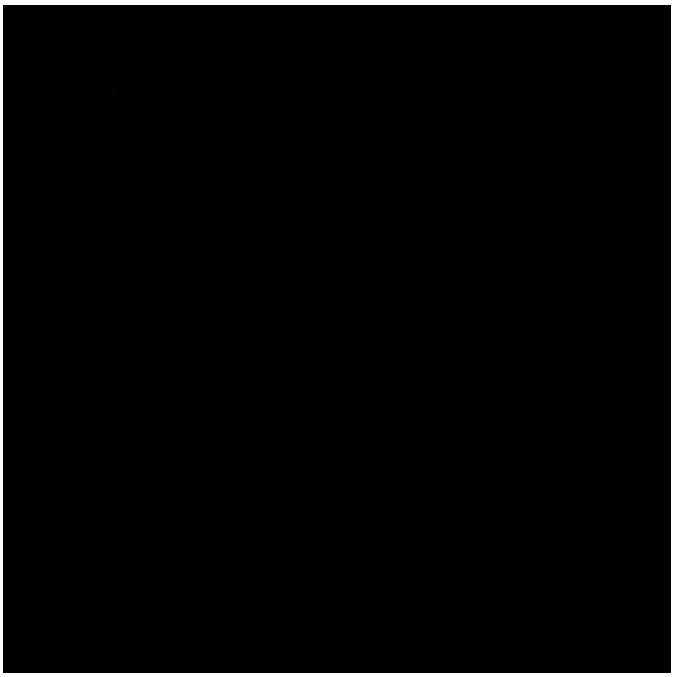


Mycodo

Environmental monitoring and regulation system

Version 8.13.0

Kyle T. Gabriel

Table of contents

1. Home

1.1 Mycodo Environmental Monitoring and Regulation System

Mycodo is open source software for the Raspberry Pi that couples inputs and outputs in interesting ways to sense and manipulate the environment.

1.1.1 Information

See the README for features, projects using Mycodo, screenshots, and other information.

This manual is also available as a PDF for offline reading.

1.1.2 Prerequisites

- Raspberry Pi single-board computer (any version: Zero, 1, 2, 3, or 4)
- Raspberry Pi Operating System installed
- An active internet connection

1.1.3 Install

Once you have the Raspberry Pi booted into the Raspberry Pi OS with an internet connection, run the following command in a terminal to initiate the Mycodo install:

```
curl -L https://kizniche.github.io/Mycodo/install | bash
```

If the install is successful, open a web browser to the Raspberry Pi's IP address and you will be greeted with a screen to create an Admin user and password.

```
https://127.0.0.1
```

1.1.4 Support

- Mycodo on GitHub
- Mycodo Wiki
- Mycodo API
- Mycodo Forum
- Mycodo Support (Android App)

1.1.5 Donate

Become a Sponsor: github.com/sponsors/kizniche

Other methods: KyleGabriel.com/donate

2. About

Mycodo is an open-source environmental monitoring and regulation system that was built to run on the Raspberry Pi.

Originally developed for cultivating edible mushrooms, Mycodo has grown to do much more. The system consists of two parts: a backend (daemon) and a frontend (web server). The backend performs tasks such as acquiring measurements from sensors and devices and coordinating a diverse set of responses to those measurements, including the ability to modulate outputs (switch relays, generate PWM signals, operate pumps, switch wireless outlets, publish/subscribe to MQTT, among others), regulate environmental conditions with PID control, schedule timers, capture photos and stream video, trigger actions when measurements meet certain conditions, and more. The frontend hosts a web interface that enables viewing and configuration from any browser-enabled device.

There are a number of different uses for Mycodo. Some users simply store sensor measurements to monitor conditions remotely, others regulate the environmental conditions of a physical space, while others capture motion-activated or time-lapse photography, among other uses.

Input controllers acquire measurements and store them in the InfluxDB time series database. Measurements typically come from sensors, but may also be configured to use the return value of linux or Python commands, or math equations, making a very powerful system for acquiring and generating data.

Output controllers produce changes to the general input/output (GPIO) pins or may be configured to execute linux or Python commands, enabling a large number of potential uses. There are a few different types of outputs: simple switching of GPIO pins (HIGH/LOW), generating pulse-width modulated (PWM) signals, switching 315/433 MHz wireless outlets, controlling Atlas Scientific peristaltic pumps, as well as executing linux and Python commands. The most common output is using a relay to switch electrical devices on and off.

When Inputs and Outputs are combined, PID controllers may be used to create a feedback loop that uses the Output device to modulate an environmental condition the Input measures. Certain Inputs may be coupled with certain Outputs to create a variety of different control and regulation applications. Beyond simple regulation, Methods may be used to create a changing setpoint over time, enabling such things as thermal cyclers, reflow ovens, environmental simulation for terrariums, food and beverage fermentation or curing, and cooking food (sous-vide), to name a few.

Triggers can be set to activate events based on specific dates and times, according to durations of time, or the sunrise/sunset at a specific latitude and longitude. Conditionals are used to activates certain events based on the truth of custom user conditional statements (e.g. "Sensor1 > 23 and 10 < Sensor2 < 30").

2.1 Web User Interface

The main frontend of Mycodo is a web user interface that allows any device with a web browser to view collected data and configure the backend, or the daemon, of the system. The web interface supports an authentication system with user/password credentials, user roles that grant/deny access to parts of the system, and SSL for encrypted browsing.

An SSL certificate with an expiration of 10 years will be generated and stored in `~/Mycodo/mycodo/mycodo_flask/ssl_certs/` during the install process to allow SSL to be used to securely connect to the web interface. If you want to use your own SSL certificates, replace them with your own.

If using the auto-generated certificate from the install, be aware that it will not be verified when visiting the web interface in your browser. You may continually receive a warning message about the security of your site unless you add the certificate to your browser's trusted list.

2.2 Languages

The Mycodo user interface has been translated from English to Dutch, German, French, Italian, Norwegian, Polish, Portuguese, Russian, Serbian, Spanish, Swedish, and Chinese. If the default language for your web browser is one of these languages, it will be automatically selected. Otherwise, you can manually set the language from the Configuration menu.

3. Frequently Asked Questions

Frequently asked questions can be found here

4. Usage

4.1 Data Viewing

There are several ways to visualize collected data.

4.1.1 Live Measurements

Page: `Data -> Live Measurements`

The `Live Measurements` page is the first page a user sees after logging in to Mycodo. It will display the current measurements being acquired from Input and Function controllers. If there is nothing displayed on the `Live` page, ensure an Input or Function controller is both configured correctly and activated. Data will be automatically updated on the page from the measurement database.

4.1.2 Asynchronous Graphs

Page: `Data -> Asynchronous Graphs`

A graphical data display that is useful for viewing data sets spanning relatively long periods of time (weeks/months/years), which could be very data- and processor-intensive to view as a Synchronous Graph. Select a time frame and data will be loaded from that time span, if it exists. The first view will be of the entire selected data set. For every view/zoom, 700 data points will be loaded. If there are more than 700 data points recorded for the time span selected, 700 points will be created from an averaging of the points in that time span. This enables much less data to be used to navigate a large data set. For instance, 4 months of data may be 10 megabytes if all of it were downloaded. However, when viewing a 4 month span, it's not possible to see every data point of that 10 megabytes, and aggregating of points is inevitable. With asynchronous loading of data, you only download what you see. So, instead of downloading 10 megabytes every graph load, only ~50kb will be downloaded until a new zoom level is selected, at which time only another ~50kb is downloaded.

> **Note**
>
> Graphs require measurements, therefore at least one Input/Output/Function/etc. needs to be added and activated in order to display data.

4.1.3 Dashboard

Page: `Data -> Dashboards`

The dashboard can be used for both viewing data and manipulating the system, thanks to the numerous dashboard widgets available. Multiple dashboards can be created as well as locked to prevent changing the arrangement.

4.1.4 Widgets

Widgets are elements on the Dashboard that have a number of uses, such as viewing data (charts, indicators, gauges, etc.) or interacting with the system (manipulate outputs, change PWM duty cycle, querying or modifying a database, etc.). Widgets can be easily rearranged and resized by dragging and dropping. For a full list of supported Widgets, see Supported Widgets.

Custom Widgets

There is a Custom Widget import system in Mycodo that allows user-created Widgets to be used in the Mycodo system. Custom Widgets can be uploaded on the `[Gear Icon] -> Configure -> Custom Widgets` page. After import, they will be available to use on the `Setup -> Widget` page.

If you develop a working Widget module, please consider creating a new GitHub issue or pull request, and it may be included in the built-in set.

Open any of the built-in modules located in the directory Mycodo/mycodo/widgets for examples of the proper formatting.

There are also example Custom Widgets in the directory Mycodo/mycodo/widgets/examples

Additionally, I have another github repository devoted to Custom Modules that are not included in the built-in set, at kizniche/Mycodo-custom.

Creating a custom widget module often requires specific placement and execution of Javascript. Several variables were created in each module to address this, and follow the following brief structure of the dashboard page that would be generated with multiple widgets being displayed.

```
<html>
<head>
  <title>Title</title>
  <script>
    {{ widget_1_dashboard_head }}
    {{ widget_2_dashboard_head }}
  </script>
</head>
<body>

<div id="widget_1">
  <div id="widget_1_titlebar">{{ widget_dashboard_title_bar }}</div>
  {{ widget_1_dashboard_body }}
  <script>
    $(document).ready(function() {
      {{ widget_1_dashboard_js_ready_end }}
    });
  </script>
</div>

<div id="widget_2">
  <div id="widget_2_titlebar">{{ widget_dashboard_title_bar }}</div>
  {{ widget_2_dashboard_body }}
  <script>
    $(document).ready(function() {
      {{ widget_2_dashboard_js_ready_end }}
    });
  </script>
</div>

<script>
  {{ widget_1_dashboard_js }}
  {{ widget_2_dashboard_js }}

  $(document).ready(function() {
    {{ widget_1_dashboard_js_ready }}
    {{ widget_2_dashboard_js_ready }}
  });
</script>

</body>
</html>
```

4.2 Inputs

Page: `Setup -> Input`

For a full list of supported Inputs, see Supported Input Devices.

Inputs, such as sensors, ADC signals, or even a response from a command, enable measuring conditions in the environment or elsewhere, which will be stored in a time-series database (InfluxDB). This database will provide measurements for Dashboard Widgets, Functions, and other parts of Mycodo to operate from. Add, configure, and activate inputs to begin recording measurements to the database and allow them to be used throughout Mycodo.

Custom Inputs

See Building a Custom Input Module Wiki page.

There is a Custom Input import system in Mycodo that allows user-created Inputs to be created an used in the Mycodo system. Custom Inputs can be uploaded and imported from the `[Gear Icon] -> Configure -> Custom Inputs` page. After import, they will be available to use on the `Setup -> Input` page.

If you develop a working Input module, please consider creating a new GitHub issue or pull request, and it may be included in the built-in set.

Open any of the built-in modules located in the directory Mycodo/mycodo/inputs for examples of the proper formatting.

There are also example Custom Inputs in the directory Mycodo/mycodo/inputs/examples

Additionally, I have another github repository devoted to Custom Modules that are not included in the built-in set, at kizniche/Mycodo-custom.

Input Commands

Input Commands are functions within the Input module that can be executed from the Web UI. This is useful for things such as calibration or other functionality specific to the input. By default, there is at least one action, Acquire Measurements Now, which will cause the input to acquire measurements rather than waiting until the next Period has elapsed.

> **Note**
>
> Actions can only be executed while the Input is active.

Input Actions

Every Period the Input will acquire measurements and store then in the time-series database. Following measurement acquisition, one or more Actions can be executed to enhance the functionality of Inputs. For example, the MQTT Publish Action can be used to publish measurements to an MQTT server.

Input Options

Setting	Description
Activate	After the sensor has been properly configured, activation begins acquiring measurements from the sensor. Any activated conditional statements will now being operating.
Deactivate	Deactivation stops measurements from being acquired from the sensor. All associated conditional statements will cease to operate.
Save	Save the current configuration entered into the input boxes for a particular sensor.
Delete	Delete a particular sensor.
Acquire Measurements Now	Force the input to conduct measurements and them in the database.
Up/Down	Move a particular sensor up or down in the order displayed.
Power Output	Select a output that powers the sensor. This enables powering cycling (turn off then on) when the sensor returns 3 consecutive errors to attempt to fix the issue. Transistors may also be used instead of a relay (note: NPN transistors are preferred over PNP for powering sensors).
Location	Depending on what sensor is being used, you will need to either select a serial number (DS18B20 temperature sensor), a GPIO pin (in the case of sensors read by a GPIO), or an I2C address. or other.
I2C Bus	The bus to be used to communicate with the I2C address.
Period (seconds)	After the sensor is successfully read and a database entry is made, this is the duration of time waited until the sensor is measured again.
Measurement Unit	Select the unit to save the measurement as (only available for select measurements).
Pre Output	If you require a output to be activated before a measurement is made (for instance, if you have a pump that extracts air to a chamber where the sensor resides), this is the output number that will be activated. The output will be activated for a duration defined by the Pre Duration, then once the output turns off, a measurement by the sensor is made.
Pre Output Duration (seconds)	This is the duration of time that the Pre Output runs for before the sensor measurement is obtained.
Pre Output During Measurement	If enabled, the Pre Output stays on during the acquisition of a measurement. If disabled, the Pre Output is turned off directly before acquiring a measurement.
Command	A linux command (executed as the user 'root') that the return value becomes the measurement
Command Measurement	The measured condition (e.g. temperature, humidity, etc.) from the linux command
Command Units	The units of the measurement condition from the linux command
Edge	Edge sensors only: Select whether the Rising or Falling (or both) edges of a changing voltage are detected. A number of devices to do this when in-line with a circuit supplying a 3.3-volt input signal to a GPIO, such as simple mechanical switch, a button, a magnet (reed/hall) sensor, a PIR motion detector, and more.
Bounce Time (ms)	Edge sensors only: This is the number of milliseconds to bounce the input signal. This is commonly called debouncing a signal [1] and may be necessary if using a mechanical circuit.
Reset Period (seconds)	Edge sensors only: This is the period of time after an edge detection that another edge will not be recorded. This enables devices such as PIR motion sensors that may stay activated for longer periods of time.

Setting	Description
Measurement	Analog-to-digital converter only: The type of measurement being acquired by the ADC. For instance, if the resistance of a photocell is being measured through a voltage divider, this measurement would be "light".
Units	Analog-to-digital converter only: This is the unit of the measurement. With the above example of "light" as the measurement, the unit may be "lux" or "intensity".
BT Adapter	The Bluetooth adapter to communicate with the input.
Clock Pin	The GPIO (using BCM numbering) connected to the Clock pin of the ADC
CS Pin	The GPIO (using BCM numbering) connected to the CS pin of the ADC
MISO Pin	The GPIO (using BCM numbering) connected to the MISO pin of the ADC
MOSI Pin	The GPIO (using BCM numbering) connected to the MOSI pin of the ADC
RTD Probe Type	Select to measure from a PT100 or PT1000 probe.
Resistor Reference (Ohm)	If your reference resistor is not the default (400 Ohm for PT100, 4000 Ohm for PT1000), you can manually set this value. Several manufacturers now use 430 Ohm resistors on their circuit boards, therefore it's recommended to verify the accuracy of your measurements and adjust this value if necessary.
Channel	Analog-to-digital converter only: This is the channel to obtain the voltage measurement from the ADC.
Gain	Analog-to-digital converter only: set the gain when acquiring the measurement.
Sample Speed	Analog-to-digital converter only: set the sample speed (typically samples per second).
Volts Min	Analog-to-digital converter only: What is the minimum voltage to use when scaling to produce the unit value for the database. For instance, if your ADC is not expected to measure below 0.2 volts for your particular circuit, set this to "0.2".
Volts Max	Analog-to-digital converter only: This is similar to the Min option above, however it is setting the ceiling to the voltage range. Units Min Analog-to-digital converter only: This value will be the lower value of a range that will use the Min and Max Voltages, above, to produce a unit output. For instance, if your voltage range is 0.0 -1.0 volts, and the unit range is 1 -60, and a voltage of 0.5 is measured, in addition to 0.5 being stored in the database, 30 will be stored as well. This enables creating calibrated scales to use with your particular circuit.
Units Max	Analog-to-digital converter only: This is similar to the Min option above, however it is setting the ceiling to the unit range.
Weighting	The This is a number between 0 and 1 and indicates how much the old reading affects the new reading. It defaults to 0 which means the old reading has no effect. This may be used to smooth the data.
Pulses Per Rev	The number of pulses for a complete revolution.
Port	The server port to be queried (Server Port Open input).
Times to Check	The number of times to attempt to ping a server (Server Ping input).
Deadline (seconds)	The maximum amount of time to wait for each ping attempt, after which 0 (offline) will be returned (Server Ping input).
Number of Measurement	The number of unique measurements to store data for this input.
Application ID	The Application ID on The Things Network.
App API Key	The Application API Key on The Things Network.
Device ID	The Device ID of the Application on The Things Network.

1. Debouncing a signal

The Things Network

The Things Network (TTN, v2 and v3) Input module enables downloading of data from TTN if the Data Storage Integration is enabled in your TTN Application. The Data Storage Integration will store data for up to 7 days. Mycodo will download this data periodically and store the measurements locally.

The payload on TTN must be properly decoded to variables that correspond to the "Variable Name" option under "Channel Options", in the lower section of the Input options. For instance, in your TTN Application, if a custom Payload Format is selected, the decoder code may look like this:

```
function Decoder(bytes, port) {
    var decoded = {};
    var rawTemp = bytes[0] + bytes[1] * 256;
    decoded.temperature = sflt162f(rawTemp) * 100;
    return decoded;
}

function sflt162f(rawSflt16) {
    rawSflt16 &= 0xFFFF;
    if (rawSflt16 === 0x8000)
        return -0.0;
    var sSign = ((rawSflt16 & 0x8000) !== 0) ? -1 : 1;
    var exp1 = (rawSflt16 >> 11) & 0xF;
    var mant1 = (rawSflt16 & 0x7FF) / 2048.0;
    return sSign * mant1 * Math.pow(2, exp1 - 15);
}
```

This will decode the 2-byte payload into a temperature float value with the name "temperature". Set "Number of Measurements" to "1", then set the "Variable Name" for the first channel (CH0) to "temperature" and the "Measurement Unit" to "Temperature: Celsius (°C)".

Upon activation of the Input, data will be downloaded for the past 7 days. The latest data timestamp will be stored so any subsequent activation of the Input will only download new data (since the last known timestamp).

This Input also allows multiple measurements to be stored. You merely have to change "Number of Measurements" to a number larger than 1, save, and there will now be multiple variable names and measurement units to set.

There are several example Input modules that, in addition to storing the measurements of a sensor in the influx database, will write the measurements to a serial device. This is useful of you have a LoRaWAN transmitter connected via serial to receive measurement information from Mycodo and transmit it to a LoRaWAN gateway (and subsequently to The Things Network). The data on TTN can then be downloaded elsewhere with the TTN Input. These example Input modules are located in the following locations:

`~/Mycodo/mycodo/inputs/examples/bme280_ttn.py`

`~/Mycodo/mycodo/inputs/examples/k30_ttn.py`

For example, the following excerpt from `bme_280.py` will write a set of comma-separated strings to the user-specified serial device with the first string (the letter "B") used to denote the sensor/measurements, followed by the actual measurements (humidity, pressure, and temperature, in this case).

```
string_send = 'B,{},{},{}'.format(
    return_dict[1]['value'],
    return_dict[2]['value'],
    return_dict[0]['value'])
self.serial_send = self.serial.Serial(self.serial_device, 9600)
self.serial_send.write(string_send.encode())
```

This is useful if multiple data strings are to be sent to the same serial device (e.g. if both `bme280_ttn.py` and `k30_ttn.py` are being used at the same time), allowing the serial device to distinguish what data is being received.

The full code used to decode both `bme280_ttn.py` and `k30_ttn.py`, with informative comments, is located at `~/Mycodo/mycodo/inputs/examples/ttn_data_storage_decoder_example.js`.

These example Input modules may be modified to suit your needs and imported into Mycodo through the `[Gear Icon] -> Configure -> Custom Inputs` page. After import, they will be available to use on the `Setup -> Input` page.

4.3 Outputs

Page: `Setup -> Output`

For a full list of supported Outputs, see Supported Outputs Devices.

Outputs are various signals that can be generated that operate devices. An output can be a HIGH/LOW signal on a GPIO pin, a pulse-width modulated (PWM) signal, a 315/433 MHz signal to switch a radio frequency-operated relay, driving of pumps and motors, or an execution of a linux or Python command, to name a few.

4.3.1 Custom Outputs

There is a Custom Output import system in Mycodo that allows user-created Outputs to be created an used in the Mycodo system. Custom Outputs can be uploaded and imported from the `[Gear Icon] -> Configure -> Custom Outputs` page. After import, they will be available to use on the `Setup -> Output` page.

If you develop a working Output module, please consider creating a new GitHub issue or pull request, and it may be included in the built-in set.

Open any of the built-in modules located in the directory Mycodo/mycodo/outputs for examples of the proper formatting.

There are also example Custom Outputs in the directory Mycodo/mycodo/outputs/examples

Additionally, I have another github repository devoted to Custom Modules that are not included in the built-in set, at kizniche/Mycodo-custom.

For Outputs that require new measurements/units, they can be added on the `[Gear Icon] -> Configure -> Measurements` page.

4.3.2 Output Options

Setting	Description
Pin (GPIO)	This is the GPIO that will be the signal to the output, using BCM numbering.
WiringPi Pin	This is the GPIO that will be the signal to the output, using WiringPi numbering.
On State	This is the state of the GPIO to signal the output to turn the device on. HIGH will send a 3.3-volt signal and LOW will send a 0-volt signal. If you output completes the circuit (and the device powers on) when a 3.3-volt signal is sent, then set this to HIGH. If the device powers when a 0-volt signal is sent, set this to LOW.
Protocol	This is the protocol to use to transmit via 315/433 MHz. Default is 1, but if this doesn't work, increment the number.
UART Device	The UART device connected to the device.
Baud Rate	The baud rate of the UART device.
I2C Address	The I2C address of the device.
I2C Bus	The I2C bus the device is connected to.
Output Mode	The Output mode, if supported.
Flow Rate	The flow rate to dispense the volume (ml/min).
Pulse Length	This is the pulse length to transmit via 315/433 MHz. Default is 189 ms.
Bit Length	This is the bit length to transmit via 315/433 MHz. Default is 24-bit.
Execute as User	Select which user executes Linux Commands.
On Command	This is the command used to turn the output on. For wireless relays, this is the numerical command to be transmitted, and for command outputs this is the command to be executed. Commands may be for the linux terminal or Python 3 (depending on which output type selected).
Off Command	This is the command used to turn the output off. For wireless relays, this is the numerical command to be transmitted, and for command outputs this is the command to be executed. Commands may be for the linux terminal or Python 3 (depending on which output type selected).
Force Command	If an Output is already on, enabling this option will allow the On command to be executed rather than returning "Output is already On".
PWM Command	This is the command used to set the duty cycle. The string "((duty_cycle))" in the command will be replaced with the actual duty cycle before the command is executed. Ensure "((duty_cycle))" is included in your command for this feature to work correctly. Commands may be for the linux terminal or Python 3 (depending on which output type selected).
Current Draw (amps)	The is the amount of current the device powered by the output draws. Note: this value should be calculated based on the voltage set in the Energy Usage Settings.
Startup State	This specifies whether the output should be ON or OFF when mycodo initially starts. Some outputs have an additional options.
Startup Value	If the Startup State is set to User Set Value (such as for PWM Outputs), then this value will be set when Mycodo starts up.
Shutdown State	This specifies whether the output should be ON or OFF when mycodo initially shuts down. Some outputs have an additional options.
Shutdown Value	If the Shutdown State is set to User Set Value (such as for PWM Outputs), then this value will be set when Mycodo shuts down.
Trigger at Startup	Select to enable triggering Functions (such as Output Triggers) when Mycodo starts and if Start State is set to ON.
Seconds to turn On	This is a way to turn a output on for a specific duration of time. This can be useful for testing the outputs and powered devices or the measured effects a device may have on an environmental condition.

4.3.3 On/Off (GPIO)

The On/Off (GPIO) output merely turns a GPIO pin High (3.3 volts) or Low (0 volts). This is useful for controlling things like electromechanical switches, such as relays, to turn electrical devices on and off.

Relays are electromechanical or solid-state devices that enable a small voltage signal (such as from a microprocessor) to activate a much larger voltage, without exposing the low-voltage system to the dangers of the higher voltage.

Add and configure outputs in the Output tab. Outputs must be properly set up before they can be used in the rest of the system.

To set up a wired relay, set the "GPIO Pin" (using BCM numbering) to the pin you would like to switch High (5 volts) and Low (0 volts), which can be used to activate relays and other devices. On Trigger should be set to the signal state (High or Low) that induces the device to turn on. For example, if your relay activates when the potential across the coil is 0-volts, set On Trigger to "Low", otherwise if your relay activates when the potential across the coil is 5 volts, set it to "High".

4.3.4 Pulse-Width Modulation (PWM)

Pulse-width modulation (PWM) is a modulation technique used to encode a message into a pulsing signal, at a specific frequency in Hertz (Hz). The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast rate. The longer the switch is on compared to the off periods, the higher the total power supplied to the load.

The PWM switching frequency has to be much higher than what would affect the load (the device that uses the power), which is to say that the resultant waveform perceived by the load must be as smooth as possible. The rate (or frequency) at which the power supply must switch can vary greatly depending on load and application, for example

> **Quote**
>
> Switching has to be done several times a minute in an electric stove; 120 Hz in a lamp dimmer; between a few kilohertz (kHz) to tens of kHz for a motor drive; and well into the tens or hundreds of kHz in audio amplifiers and computer power supplies.

The term duty cycle describes the proportion of 'on' time to the regular interval or 'period' of time; a low duty cycle corresponds to low power, because the power is off for most of the time. Duty cycle is expressed in percent, with 0% being always off, 50% being off for half of the time and on for half of the time, and 100% being always on.

4.3.5 Pulse-Width Modulation (PWM) Options

Setting	Description
Library	Select the method for producing the PWM signal. Hardware pins can produce up to a 30 MHz PWM signal, while any other (non-hardware PWM) pin can produce up to a 40 kHz PWM signal. See the table, below, for the hardware pins on various Pi boards.
Pin (GPIO)	This is the GPIO pin that will output the PWM signal, using BCM numbering.
Frequency (Hertz)	This is frequency of the PWM signal.
Invert Signal	Send an inverted duty cycle to the output controller.
Duty Cycle	This is the proportion of the time on to the time off, expressed in percent (0 -100).

Non-hardware PWM Pins

When using non-hardware PWM pins, there are only certain frequencies that can be used. These frequencies in Hertz are 40000, 20000, 10000, 8000, 5000, 4000, 2500, 2000, 1600, 1250, 1000, 800, 500, 400, 250, 200, 100, and 50 Hz. If you attempt to set a frequency that is not listed here, the nearest frequency from this list will be used.

Hardware PWM Pins

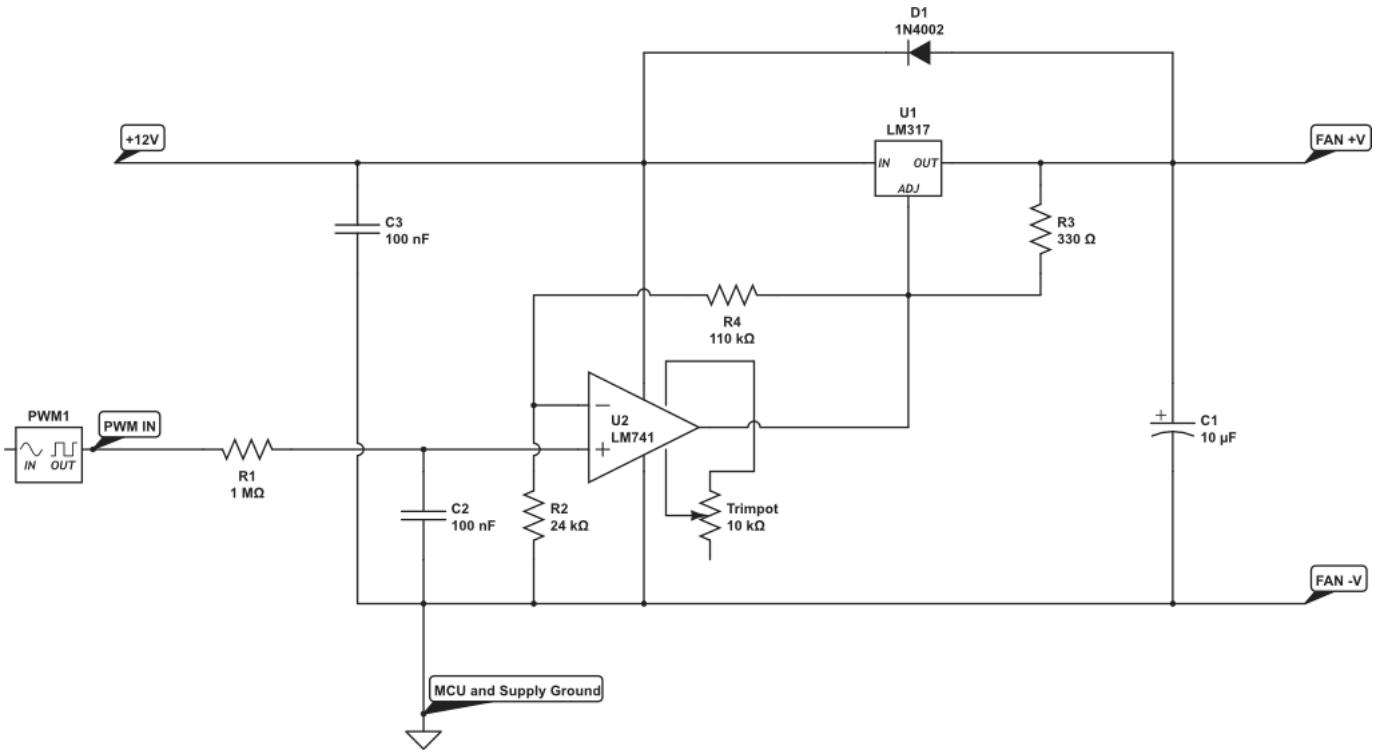
The exact frequency may be set when using hardware PWM pins. The same PWM channel is available on multiple GPIO. The latest frequency and duty cycle setting will be used by all GPIO pins which share a PWM channel.

BCM Pin	PWM Channel	Raspberry Pi Version
12	0	All models except A and B
13	1	All models except A and B
18	0	All models
19	1	All models except A and B
40	0	Compute module only
41	1	Compute module only
45	1	Compute module only
52	0	Compute module only
53	1	Compute module only

Schematics for DC Fan Control

Below are hardware schematics that enable controlling direct current (DC) fans from the PWM output from Mycodo.

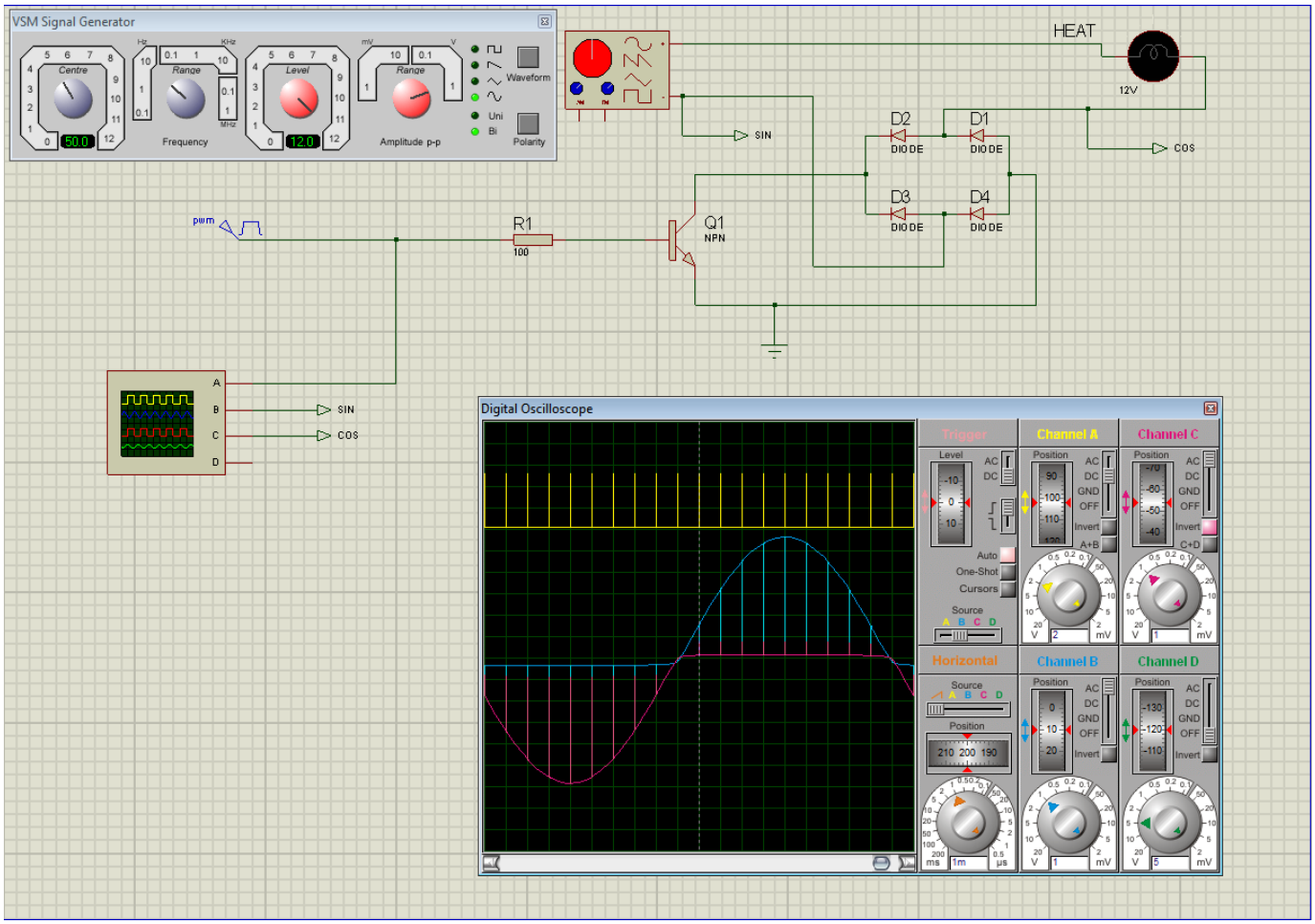
PWM output controlling a 12-volt DC fan (such as a PC fan)



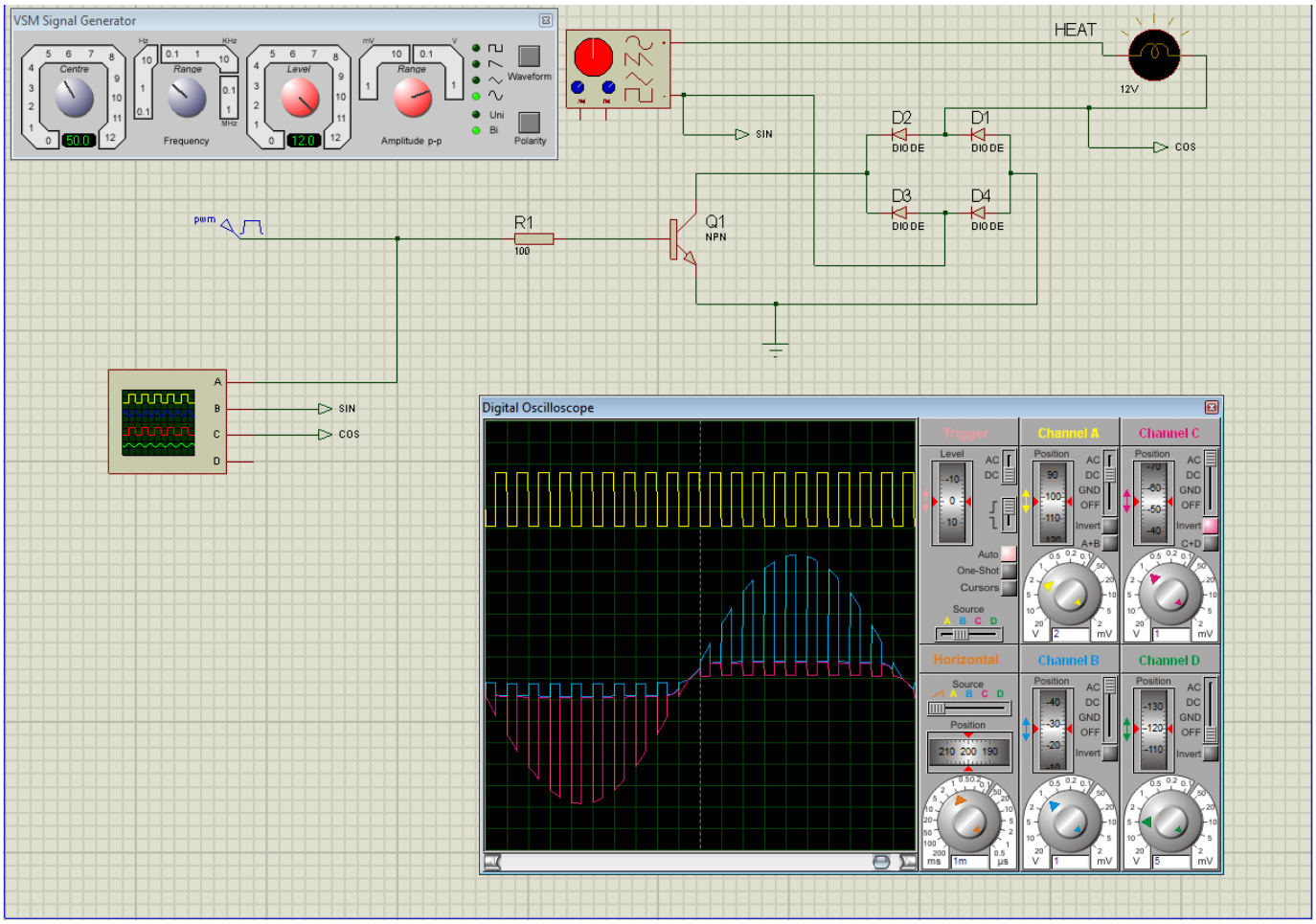
Schematics for AC Modulation

Below are hardware schematics that enable the modulation of alternating current (AC) from the PWM output from Mycodo.

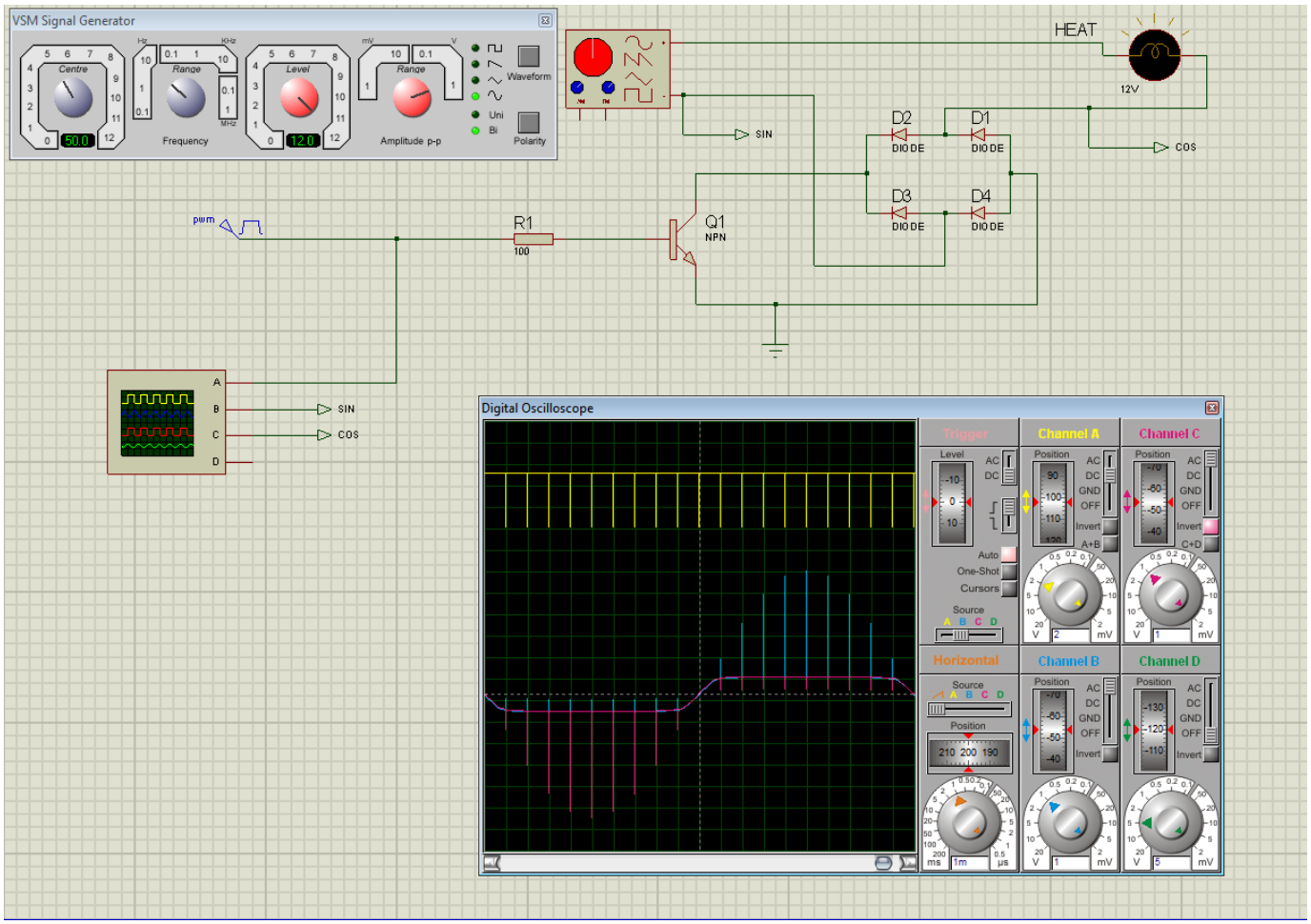
PWM output modulating alternating current (AC) at 1% duty cycle



PWM output modulating alternating current (AC) at 50% duty cycle



PWM output modulating alternating current (AC) at 99% duty cycle



4.3.6 Peristaltic Pump

There are two peristaltic pump Output modules that Mycodo supports, a generic peristaltic pump Output, and the Atlas Scientific EZO-PMP peristaltic pump.

Generic Peristaltic Pump

Any peristaltic pump can be used with the Generic Peristaltic Pump Output to dispense liquids. The most basic dispensing abilities are to start dispensing, stop dispensing, or dispense for a duration of time. If the pump rate has been measured, this value can be entered into the Fastest Rate (ml/min) setting and the Output controller will then be able to dispense specific volumes rather than merely for durations of time. In oder to dispense specific volumes, the Output Mode will also need to be set in addition to the Desired Flow Rate (ml/min), if the Output Mode has been set to Specify Flow Rate.

To determine your pump's flow rate, first purge all air from your pump's hose. Next, instruct the pump to dispense for 60 seconds and collect the liquid it dispenses. Once finished, measure the amount of liquid and enter this value, in milliliters into the Fastest Rate (ml/min) setting. Once your pump's flow rate is set, you can now start dispensing specific volumes rather than durations.

This Output module relies on switching a GPIO pin High and Low to switch the peristaltic pump on and off. This is most easily accomplished with the use of a relay in-line with your pump's power supply or using the GPIO as an input signal directly to the pump (if supported). When using a relay, it's important to develop your circuit to provide the fastest possible switching of the pump. Since the volume dispensed by the pump is dependent on time, the faster the pump switching can occur, the more accurate the dispensing will be. Many peristaltic pumps operate on DC voltage and require an AC-DC converter. These converters can take a significant amount of time to energize once power is applied as well as de-energize once power is removed, causing significant delays that can impact dispensing accuracy. To alleviate this issue, the DC power should be switched, rather than the AC power, which will remove this potential delay.

Atlas Scientific Peristaltic Pump

The Atlas Scientific peristaltic pump is a peristaltic pump and microcontroller combined that allows it to be communicated with via I2C or Serial and can accurately dispense specific volumes of fluid. There are several commands the pump can accept, including commands to calibrate, turn on, turn off, and dispense at a specific rate, among others. Atlas Scientific peristaltic pumps are good options, but are more expensive than generic peristaltic pumps.

Peristaltic Pump Options

Setting	Description
Output Mode	"Fastest low Rate" will pump liquid at the fastest rate the pump can perform. "Specify Flow Rate" will pump liquid at the rate set by the "Flow Rate (ml/min)" option.
Flow Rate (ml/min)	This is how fast liquid will be pumped if the "Specify Flow Rate" option is selected for the Output Mode option.
Fastest Rate (ml/min)	This is the rate at which the pump dispenses liquid, in ml/min.
Minimum On (sec/min)	This is the minimum duration (seconds) the pump should be turned on for every 60 second period of pumping. This option is only used when Specify Flow Rate is selected as the output Mode.

4.3.7 Wireless 315/433 MHz

Certain 315/433 MHz wireless relays may be used, however you will need to set the pin of the transmitter (using BCM numbering), pulse length, bit length, protocol, on command, and off command. To determine your On and Off commands, connect a 315/433 MHz receiver to your Pi, then run the receiver script, below, replacing 17 with the pin your receiver is connected to (using BCM numbering), and press one of the buttons on your remote (either on or off) to detect the numeric code associated with that button.

```
sudo ~/Mycodo/env/bin/python ~/Mycodo/mycodo/devices/wireless_rpi_rf.py -d 2 -g 17
```

433 MHz wireless relays have been successfully tested with SMAKN 433MHz RF Transmitters/Receivers and Etekcity Wireless Remote Control Electrical Outlets (see Issue 88 for more information). If you have a 315/433 MHz transmitter/receiver and a wireless relay that does not work with the current code, submit a new issue with details of your hardware.

4.3.8 Linux Command

Another option for output control is to execute a terminal command when the output is turned on, off, or a duty cycle is set. Commands will be executed as the user 'root'. When a Linux Command output is created, example code is provided to demonstrate how to use the output.

4.3.9 Python Command

The Python Command output operates similarly to the Linux Command output, however Python 3 code is being executed. When a Python Command output is created, example code is provided to demonstrate how to use the output.

4.3.10 Output Notes

Wireless and Command (Linux/Python) Outputs: Since the wireless protocol only allows 1-way communication to 315/433 MHz devices, wireless relays are assumed to be off until they are turned on, and therefore will appear red (off) when added. If a wireless relay is turned off or on outside Mycodo (by a remote, for instance), Mycodo will ***not*** be able to determine the state of the relay and will indicate whichever state the relay was last. This is, if Mycodo turns the wireless relay on, and a remote is used to turn the relay off, Mycodo will still assume the relay is on.

4.4 Functions

Page: `Setup -> Function`

For a full list of supported Functions, see Supported Functions.

Function controllers perform tasks that often involve the use of Inputs and Outputs.

> **Note**
>
> "Last" means the Function will only acquire the last (latest) measurement in the database. "Past" means the Function will acquire all measurements from the present until the "Max Age (seconds)" that's been set (e.g. if measurements are acquired every 10 seconds, and a Max Age is set to 60 seconds, there will on average be 6 measurements returned to the Function to operate with).

4.4.1 Custom Functions

There is a Custom Function import system in Mycodo that allows user-created Functions to be used in the Mycodo system. Custom Functions can be uploaded on the `[Gear Icon] -> Configure -> Custom Functions` page. After import, they will be available to use on the `Setup -> Function` page.

If you develop a working Function module, please consider creating a new GitHub issue or pull request, and it may be included in the built-in set.

Open any of the built-in modules located in the directory Mycodo/mycodo/functions for examples of the proper formatting.

There are also example Custom Functions in the directory Mycodo/mycodo/functions/examples

Additionally, I have another github repository devoted to Custom Modules that are not included in the built-in set, at kizniche/Mycodo-custom.

For Functions that require new measurements/units, they can be added on the `[Gear Icon] -> Configure -> Measurements` page.

4.4.2 PID Controller

A proportional-derivative-integral (PID) controller is a control loop feedback mechanism used throughout industry for controlling systems. It efficiently brings a measurable condition, such as the temperature, to a desired state and maintains it there with little overshoot and oscillation. A well-tuned PID controller will raise to the setpoint quickly, have minimal overshoot, and maintain the setpoint with little oscillation.

PID settings may be changed while the PID is activated and the new settings will take effect immediately. If settings are changed while the controller is paused, the values will be used once the controller resumes operation.

PID Controller Options

Setting	Description
Activate/ Deactivate	Turn a particular PID controller on or off.
Pause	When paused, the control variable will not be updated and the PID will not turn on the associated outputs. Settings can be changed without losing current PID output values.
Hold	When held, the control variable will not be updated but the PID will turn on the associated outputs, Settings can be changed without losing current PID output values.
Resume	Resume a PID controller from being held or paused.
Direction	This is the direction that you wish to regulate. For example, if you only require the temperature to be raised, set this to "Up," but if you require regulation up and down, set this to "Both."
Period	This is the duration between when the PID acquires a measurement, the PID is updated, and the output is modulated.
Start Offset (seconds)	Wait this duration before attempting the first calculation/measurement.
Max Age	The time (in seconds) that the sensor measurement age is required to be less than. If the measurement is not younger than this age, the measurement is thrown out and the PID will not actuate the output. This is a safety measure to ensure the PID is only using recent measurements.
Setpoint	This is the specific point you would like the environment to be regulated at. For example, if you would like the humidity regulated to 60%, enter 60.
Band (+/- Setpoint)	Hysteresis option. If set to a non-0 value, the setpoint will become a band, which will be between the band_max=setpoint+band and band_min=setpoint-band. If Raising, the PID will raise above band_max, then wait until the condition falls below band_min to resume regulation. If Lowering, the PID will lower below band_min, then wait until the condition rises above band_max to resume regulating. If set to Both, regulation will only occur to the outside min and max of the band, and cease when within the band. Set to 0 to disable Hysteresis.
Store Lower as Negative	Checking this will store all output variables (PID and output duration/duty cycle) as a negative values in the measurement database. This is useful for displaying graphs that indicate whether the PID is currently lowering or raising. Disable this if you desire all positive values to be stored in the measurement database.
K_P Gain	Proportional coefficient (non-negative). Accounts for present values of the error. For example, if the error is large and positive, the control output will also be large and positive.
K_I Gain	Integral coefficient (non-negative). Accounts for past values of the error. For example, if the current output is not sufficiently strong, the integral of the error will accumulate over time, and the controller will respond by applying a stronger action.
K_D Gain	Derivative coefficient (non-negative). Accounts for predicted future values of the error, based on its current rate of change.
Integrator Min	The minimum allowed integrator value, for calculating Ki_total: (Ki_total = Ki * integrator; and PID output = Kp_total + Ki_total + Kd_total)
Integrator Max	The maximum allowed integrator value, for calculating Ki_total: (Ki_total = Ki * integrator; and PID output = Kp_total + Ki_total + Kd_total)
Output (Raise/ Lower)	This is the output that will cause the particular environmental condition to rise or lower. In the case of raising the temperature, this may be a heating pad or coil.
Min On Duration, Duty Cycle, or Amount (Raise/ Lower)	This is the minimum value that the PID output must be before Output (Lower) turns on. If the PID output is less than this value, Duration Outputs will not turn on, and PWM Outputs will be turned off unless Always Min is enabled.

Setting	Description
Max On Duration, Duty Cycle, or Amount (Raise/Lower)	This is the maximum duration, volume, or duty cycle the Output (Raise) can be set to. If the PID output is greater than this value, the Max value set here will be used.
Min Off Duration (Raise/Lower)	For On/Off (Duration) Outputs, this is the minimum amount of time the Output must have been off for before it is allowed to turn back on. Ths is useful for devices that can be damaged by rapid power cycling (e.g. fridges).
Always Min (Raise/Lower)	For PWM Outputs only. If enabled, the duty cycle will never be set below the Min value.
Setpoint Tracking Method	Set a method to change the setpoint over time.

PID Output Calculation

PID Controllers can control a number of different output types (e.g. duration, volume, or PWM duty cycle). For most output types, the PID output (Control Variable) will be proportional (i.e. `Output Duration = PID Control Variable`). However, when outputting a duty cycle, it will be calculated as `Duty Cycle = (Control Variable / Period) * 100`.

> **Note**
>
> Control Variable = P Output + I Output + D Output. Duty cycle is limited within the 0 - 100 % range and the set Min Duty Cycle and Max Duty Cycle. An output duration is limited by the set Min On Duration and Max On Duration, and output volume similarly.

PID Tuning

PID tuning can be a complex process, depending on the output device(s) used and the environment or system under control. A system with large perturbations will be more difficult to control than one that is stable. Similarly, output devices that are unsuitable may make PID tuning difficult or impossible. Learning how PID controllers operate and the theory behind their tuning will not only better prepare you to operate a PID controller, but also in the development of your system and selection and implementation of the output devices used to regulate your system.

PID TUNING RESOURCES

- Sous Vide PID Tuning and the Unexpected Electrical Fire

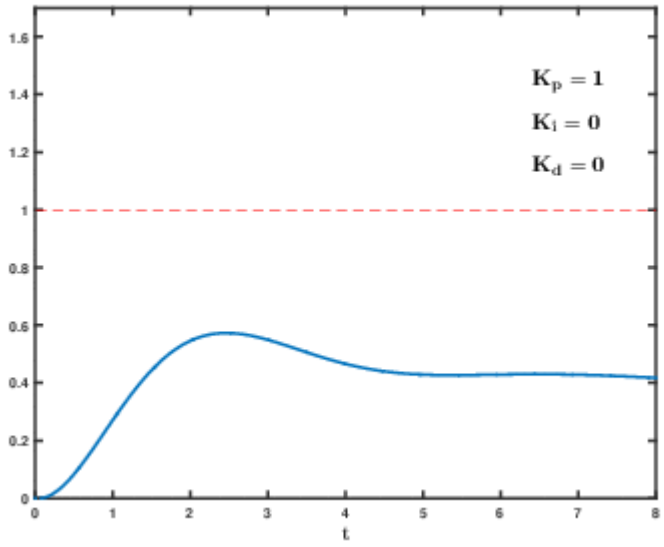
PID CONTROL THEORY

The PID controller is the most common regulatory controller found in industrial settings, for it"s ability to handle both simple and complex regulation. The PID controller has three paths, the proportional, integral, and derivative.

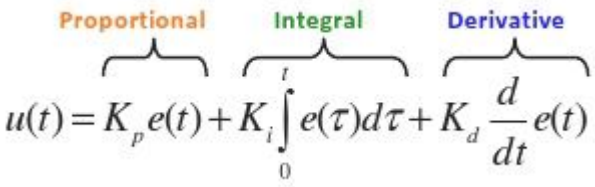
The **P**roportional takes the error and multiplies it by the constant K_P, to yield an output value. When the error is large, there will be a large proportional output.

The **I**ntegral takes the error and multiplies it by K_I, then integrates it ($K_I \cdot 1/s$). As the error changes over time, the integral will continually sum it and multiply it by the constant K_I. The integral is used to remove perpetual error in the control system. If using K_P alone produces an output that produces a perpetual error (i.e. if the sensor measurement never reaches the Set Point), the integral will increase the output until the error decreases and the Set Point is reached.

The **D**erivative multiplies the error by K_D, then differentiates it ($K_D \cdot s$). When the error rate changes over time, the output signal will change. The faster the change in error, the larger the derivative path becomes, decreasing the output rate of change. This has the effect of dampening overshoot and undershoot (oscillation) of the Set Point.



The K_P, K_I, and K_D gains determine how much each of the P, I, and D variables influence the final PID output value. For instance, the greater the value of the gain, the more influence that variable has on the output.



The output from the PID controller can be used in a number of ways. A simple use is to use this value as the number of seconds an output is turned on during a periodic interval (Period). For instance, if the Period is set to 30 seconds, the PID equation has the desired measurement and the actual measurement used to calculate the PID output every 30 seconds. The more the output is on during this period, the more it will affect the system. For example, an output on for 15 seconds every 30 seconds is at a 50 % duty cycle, and would affect the system roughly half as much as when the output is on for 30 seconds every 30 seconds, or at at 100 % duty cycle. The PID controller will calculate the output based on the amount of error (how far the actual measurement is from the desired measurement). If the error increases or persists, the output increases, causing the output to turn on for a longer duration within the Period, which usually in term causes the measured condition to change and the error to reduce. When the error reduces, the control variable decreases, meaning the output is turned on for a shorter duration of time. The ultimate goal of a well-tuned PID controller is to bring the actual measurement to the desired measurement quickly, with little overshoot, and maintain the setpoint with minimal oscillation.

Using temperature as an example, the Process Variable (PV) is the measured temperature, the Setpoint (SP) is the desired temperature, and the Error (e) is the distance between the measured temperature and the desired temperature (indicating if the actual temperature is too hot or too cold and to what degree). The error is manipulated by each of the three PID components, producing an output, called the Manipulated Variable (MV) or Control Variable (CV). To allow control of how much each path contributes to the output value, each path is multiplied by a gain (represented by K_P, K_I, and K_D). By adjusting the gains, the sensitivity of the system to each path is affected. When all three paths are summed, the PID output is produced. If a gain is set to 0, that path does not contribute to the output and that path is essentially turned off.

The output can be used a number of ways, however this controller was designed to use the output to affect the measured value (PV). This feedback loop, with a properly tuned PID controller, can achieve a set point in a short period of time, maintain regulation with little oscillation, and respond quickly to disturbance.

Therefor, if one would be regulating temperature, the sensor would be a temperature sensor and the feedback device(s) would be able to heat and cool. If the temperature is lower than the Set Point, the output value would be positive and a heater would

activate. The temperature would rise toward the desired temperature, causing the error to decrease and a lower output to be produced. This feedback loop would continue until the error reaches 0 (at which point the output would be 0). If the temperature continues to rise past the Set Point (this is may be acceptable, depending on the degree), the PID would produce a negative output, which could be used by the cooling device to bring the temperature back down, to reduce the error. If the temperature would normally lower without the aid of a cooling device, then the system can be simplified by omitting a cooler and allowing it to lower on its own.

Implementing a controller that effectively utilizes K_P, K_I, and K_D can be challenging. Furthermore, it is often unnecessary. For instance, the K_I and K_D can be set to 0, effectively turning them off and producing the very popular and simple P controller. Also popular is the PI controller. It is recommended to start with only K_P activated, then experiment with K_P and K_I, before finally using all three. Because systems will vary (e.g. airspace volume, degree of insulation, and the degree of impact from the connected device, etc.), each path will need to be adjusted through experimentation to produce an effective output.

QUICK SETUP EXAMPLES

These example setups are meant to illustrate how to configure regulation in particular directions, and not to achieve ideal values to configure your K_P, K_I, and K_D gains. There are a number of online resources that discuss techniques and methods that have been developed to determine ideal PID values (such as here, here, here, here, and here) and since there are no universal values that will work for every system, it is recommended to conduct your own research to understand the variables and essential to conduct your own experiments to effectively implement them.

Provided merely as an example of the variance of PID values, one of my setups had temperature PID values (up regulation) of $K_P = 30$, $K_I = 1.0$, and $K_D = 0.5$, and humidity PID values (up regulation) of $K_P = 1.0$, $K_I = 0.2$, and $K_D = 0.5$. Furthermore, these values may not have been optimal but they worked well for the conditions of my environmental chamber.

EXACT TEMPERATURE REGULATION

This will set up the system to raise and lower the temperature to a certain level with two regulatory devices (one that heats and one that cools).

Add a sensor, then save the proper device and pin/address for each sensor and activate the sensor.

Add two outputs, then save each GPIO and On Trigger state.

Add a PID, then select the newly-created sensor. Change Setpoint to the desired temperature, Regulate Direction to "Both". Set Raise Output to the relay attached to the heating device and the Lower Relay to the relay attached to the cooling device.

Set $K_P = 1$, $K_I = 0$, and $K_D = 0$, then activate the PID.

If the temperature is lower than the Set Point, the heater should activate at some interval determined by the PID controller until the temperature rises to the set point. If the temperature goes higher than the Set Point (or Set Point + Buffer), the cooling device will activate until the temperature returns to the set point. If the temperature is not reaching the Set Point after a reasonable amount of time, increase the K_P value and see how that affects the system. Experiment with different configurations involving only Read Interval and K_P to achieve a good regulation. Avoid changing the K_I and K_D from 0 until a working regulation is achieved with K_P alone.

View graphs in the 6 to 12 hour time span to identify how well the temperature is regulated to the Setpoint. What is meant by well-regulated will vary, depending on your specific application and tolerances. Most applications of a PID controller would like to see the proper temperature attained within a reasonable amount of time and with little oscillation around the Setpoint.

Once regulation is achieved, experiment by reducing K_P slightly (~25%) and increasing K_I by a low amount to start, such as 0.1 (or lower, 0.01), then start the PID and observe how well the controller regulates. Slowly increase K_I until regulation becomes both quick and with little oscillation. At this point, you should be fairly familiar with experimenting with the system and the K_D value can be experimented with once both K_P and K_I have been tuned.

HIGH TEMPERATURE REGULATION

Often the system can be simplified if two-way regulation is not needed. For instance, if cooling is unnecessary, this can be removed from the system and only up-regulation can be used.

Use the same configuration as the Exact Temperature Regulation example, except change Regulate Direction to "Raise" and do not touch the "Down Relay" section.

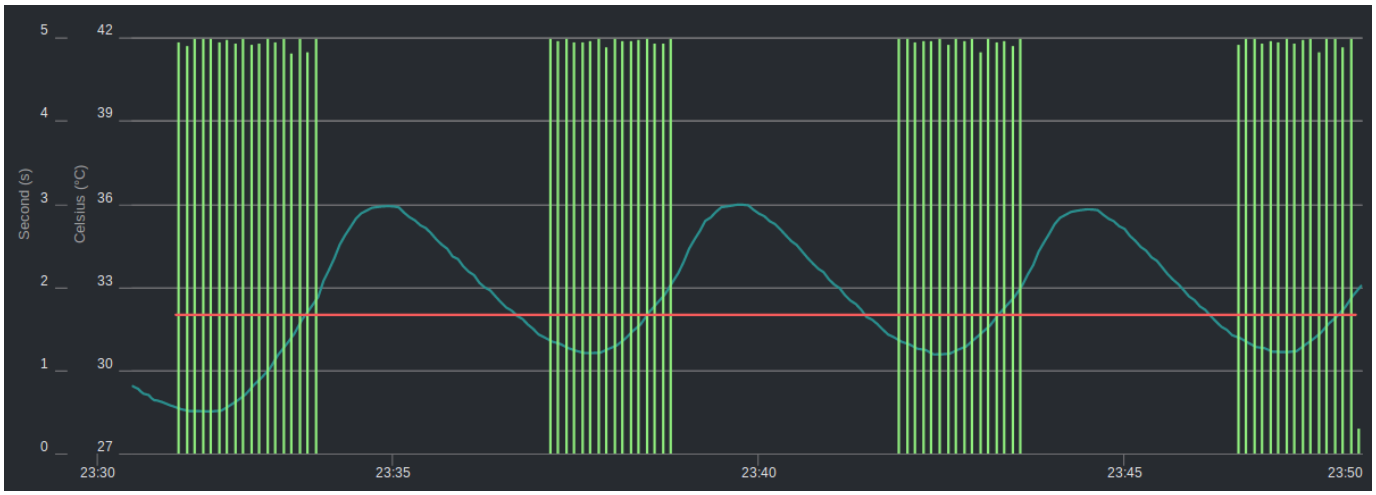
4.4.3 PID Autotune

> **Warning**
>
> This is an experimental feature. It is best not used until you are familiar with the theory, operation, and tuning of a PID.

The Autotune function is a standalone controller that is useful for determining appropriate Kp, Ki, and Kd gains for use in the a PID controller. The autotuner will manipulate an output and analyze the measured response in a particular environment/system. It will take several cycles of perturbing the system with the chosen output before enough data is available to calculate the PID gains. In order to use this feature, select a Measurement and an Output that can module the specific condition being measured. Then, configure the Noise Band and Outstep and activate the function. Log lines of the autotuner will appear in the daemon log (`[Gear Icon] -> Mycodo Logs -> Daemon Log`). While the autotune is being performed, it is recommended to create a dashboard graph that includes the Measurement and Output in order to see what the PID Autotuner is doing and to notice any potential issues with the autotune settings that have been configured. If the autotune is taking a long time to complete, there may not be enough stability in the system being manipulated to calculate a reliable set of PID gains. This may be because there are too many perturbations to the system, or conditions are changing too rapidly to acquire consistent measurement oscillations. If this is the case, try modifying your system to increase stability and yield consistent measurement oscillations. Once the autotune successfully completes, perturbations may be reintroduced in order to further tune the PID controller to handle them.

Setting	Description
Measurement	This is the Input or Function measurement that is measuring the specific condition that the Output will affect. For instance, this could be a temperature measurement and the output could be a heater.
Output	This is the Output that will affect the measurement when it's activated. The autotune function will periodically turn this output on in order to raise the measurement beyond the setpoint.
Period	This is the period of time between the Output being turned on. This should be set to the same Period you wish to use for your PID controller. A different Period can significantly affect the PID gains that the autotune produces.
Setpoint	This is the desired measurement condition value. For instance, if temperature is being measured, this should be set a several degrees higher than the current temperature so the output, when activated, will cause the temperature to rise beyond the setpoint.
Noise Band	This is the amount above the setpoint the measured condition must reach before the output turns off. This is also how much below the setpoint the measured condition must fall before the output turns back on.
Outstep	This is how many seconds the output will turn on every PID Period. For instance, to autotune with 50% power, ensure the Outstep is half the value of the PID Period.
Direction	This is the direction for which the Output will push the Measurement. For instance, a heater will raise temperature, whereas a cooler will lower temperature.

Typical graph output will look like this:



And typical Daemon Log output will look like this:

```
2018-08-04 23:32:20,876 - mycodo.pid_3b533dff - INFO - Activated in 187.2 ms
2018-08-04 23:32:20,877 - mycodo.pid_autotune - INFO - PID Autotune started
2018-08-04 23:33:50,823 - mycodo.pid_autotune - INFO -
2018-08-04 23:33:50,830 - mycodo.pid_autotune - INFO - Cycle: 19
2018-08-04 23:33:50,831 - mycodo.pid_autotune - INFO - switched state: relay step down
2018-08-04 23:33:50,832 - mycodo.pid_autotune - INFO - input: 32.52
2018-08-04 23:36:00,854 - mycodo.pid_autotune - INFO -
2018-08-04 23:36:00,860 - mycodo.pid_autotune - INFO - Cycle: 45
2018-08-04 23:36:00,862 - mycodo.pid_autotune - INFO - found peak: 34.03
2018-08-04 23:36:00,863 - mycodo.pid_autotune - INFO - peak count: 1
2018-08-04 23:37:20,802 - mycodo.pid_autotune - INFO -
2018-08-04 23:37:20,809 - mycodo.pid_autotune - INFO - Cycle: 61
2018-08-04 23:37:20,810 - mycodo.pid_autotune - INFO - switched state: relay step up
2018-08-04 23:37:20,811 - mycodo.pid_autotune - INFO - input: 31.28
2018-08-04 23:38:30,867 - mycodo.pid_autotune - INFO -
2018-08-04 23:38:30,874 - mycodo.pid_autotune - INFO - Cycle: 75
2018-08-04 23:38:30,876 - mycodo.pid_autotune - INFO - found peak: 32.17
2018-08-04 23:38:30,878 - mycodo.pid_autotune - INFO - peak count: 2
2018-08-04 23:38:40,852 - mycodo.pid_autotune - INFO -
2018-08-04 23:38:40,858 - mycodo.pid_autotune - INFO - Cycle: 77
2018-08-04 23:38:40,860 - mycodo.pid_autotune - INFO - switched state: relay step down
2018-08-04 23:38:40,861 - mycodo.pid_autotune - INFO - input: 32.85
2018-08-04 23:40:50,834 - mycodo.pid_autotune - INFO -
2018-08-04 23:40:50,835 - mycodo.pid_autotune - INFO - Cycle: 103
2018-08-04 23:40:50,836 - mycodo.pid_autotune - INFO - found peak: 33.93
2018-08-04 23:40:50,836 - mycodo.pid_autotune - INFO - peak count: 3
2018-08-04 23:42:05,799 - mycodo.pid_autotune - INFO -
2018-08-04 23:42:05,805 - mycodo.pid_autotune - INFO - Cycle: 118
2018-08-04 23:42:05,806 - mycodo.pid_autotune - INFO - switched state: relay step up
2018-08-04 23:42:05,807 - mycodo.pid_autotune - INFO - input: 31.27
2018-08-04 23:43:15,816 - mycodo.pid_autotune - INFO -
2018-08-04 23:43:15,822 - mycodo.pid_autotune - INFO - Cycle: 132
2018-08-04 23:43:15,824 - mycodo.pid_autotune - INFO - found peak: 32.09
2018-08-04 23:43:15,825 - mycodo.pid_autotune - INFO - peak count: 4
2018-08-04 23:43:25,790 - mycodo.pid_autotune - INFO -
2018-08-04 23:43:25,796 - mycodo.pid_autotune - INFO - Cycle: 134
2018-08-04 23:43:25,797 - mycodo.pid_autotune - INFO - switched state: relay step down
2018-08-04 23:43:25,798 - mycodo.pid_autotune - INFO - input: 32.76
2018-08-04 23:45:30,802 - mycodo.pid_autotune - INFO -
2018-08-04 23:45:30,808 - mycodo.pid_autotune - INFO - Cycle: 159
2018-08-04 23:45:30,810 - mycodo.pid_autotune - INFO - found peak: 33.98
2018-08-04 23:45:30,811 - mycodo.pid_autotune - INFO - peak count: 5
2018-08-04 23:45:30,812 - mycodo.pid_autotune - INFO -
2018-08-04 23:45:30,814 - mycodo.pid_autotune - INFO - amplitude: 0.9099999999999989
2018-08-04 23:45:30,815 - mycodo.pid_autotune - INFO - amplitude deviation: 0.06593406593406595
2018-08-04 23:46:40,851 - mycodo.pid_autotune - INFO -
2018-08-04 23:46:40,857 - mycodo.pid_autotune - INFO - Cycle: 173
2018-08-04 23:46:40,858 - mycodo.pid_autotune - INFO - switched state: relay step up
2018-08-04 23:46:40,859 - mycodo.pid_autotune - INFO - input: 31.37
2018-08-04 23:47:55,860 - mycodo.pid_autotune - INFO -
2018-08-04 23:47:55,866 - mycodo.pid_autotune - INFO - Cycle: 188
2018-08-04 23:47:55,868 - mycodo.pid_autotune - INFO - found peak: 32.36
2018-08-04 23:47:55,869 - mycodo.pid_autotune - INFO - peak count: 6
2018-08-04 23:47:55,870 - mycodo.pid_autotune - INFO -
2018-08-04 23:47:55,871 - mycodo.pid_autotune - INFO - amplitude: 0.9149999999999979
2018-08-04 23:47:55,872 - mycodo.pid_autotune - INFO - amplitude deviation: 0.032786885245900406
2018-08-04 23:47:55,873 - mycodo.pid_3b533dff - INFO - time:  16 min
2018-08-04 23:47:55,874 - mycodo.pid_3b533dff - INFO - state: succeeded
2018-08-04 23:47:55,874 - mycodo.pid_3b533dff - INFO -
2018-08-04 23:47:55,875 - mycodo.pid_3b533dff - INFO - rule: ziegler-nichols
2018-08-04 23:47:55,876 - mycodo.pid_3b533dff - INFO - Kp: 0.40927018474290117
2018-08-04 23:47:55,877 - mycodo.pid_3b533dff - INFO - Ki: 0.05846588600007114
2018-08-04 23:47:55,879 - mycodo.pid_3b533dff - INFO - Kd: 0.7162385434443115
2018-08-04 23:47:55,880 - mycodo.pid_3b533dff - INFO -
2018-08-04 23:47:55,881 - mycodo.pid_3b533dff - INFO - rule: tyreus-luyben
```

```
2018-08-04 23:47:55,887 - mycodo.pid_3b533dff - INFO - Kp: 0.3162542336649691
2018-08-04 23:47:55,889 - mycodo.pid_3b533dff - INFO - Ki: 0.010165091543194185
2018-08-04 23:47:55,890 - mycodo.pid_3b533dff - INFO - Kd: 0.7028026111719073
2018-08-04 23:47:55,891 - mycodo.pid_3b533dff - INFO -
2018-08-04 23:47:55,892 - mycodo.pid_3b533dff - INFO - rule: ciancone-marlin
2018-08-04 23:47:55,892 - mycodo.pid_3b533dff - INFO - Kp: 0.21083615577664605
2018-08-04 23:47:55,893 - mycodo.pid_3b533dff - INFO - Ki: 0.06626133746674728
2018-08-04 23:47:55,893 - mycodo.pid_3b533dff - INFO - Kd: 0.3644161687558038
2018-08-04 23:47:55,894 - mycodo.pid_3b533dff - INFO -
2018-08-04 23:47:55,894 - mycodo.pid_3b533dff - INFO - rule: pessen-integral
2018-08-04 23:47:55,895 - mycodo.pid_3b533dff - INFO - Kp: 0.49697093861638
2018-08-04 23:47:55,895 - mycodo.pid_3b533dff - INFO - Ki: 0.0887428626786794
2018-08-04 23:47:55,896 - mycodo.pid_3b533dff - INFO - Kd: 1.04627757151908
2018-08-04 23:47:55,896 - mycodo.pid_3b533dff - INFO -
2018-08-04 23:47:55,897 - mycodo.pid_3b533dff - INFO - rule: some-overshoot
2018-08-04 23:47:55,898 - mycodo.pid_3b533dff - INFO - Kp: 0.23191977135431066
2018-08-04 23:47:55,898 - mycodo.pid_3b533dff - INFO - Ki: 0.03313066873337365
2018-08-04 23:47:55,899 - mycodo.pid_3b533dff - INFO - Kd: 1.0823160212047374
2018-08-04 23:47:55,899 - mycodo.pid_3b533dff - INFO -
2018-08-04 23:47:55,900 - mycodo.pid_3b533dff - INFO - rule: no-overshoot
2018-08-04 23:47:55,900 - mycodo.pid_3b533dff - INFO - Kp: 0.1391518628125864
2018-08-04 23:47:55,901 - mycodo.pid_3b533dff - INFO - Ki: 0.01987840124002419
2018-08-04 23:47:55,901 - mycodo.pid_3b533dff - INFO - Kd: 0.6493896127228425
2018-08-04 23:47:55,902 - mycodo.pid_3b533dff - INFO -
2018-08-04 23:47:55,902 - mycodo.pid_3b533dff - INFO - rule: brewing
2018-08-04 23:47:55,903 - mycodo.pid_3b533dff - INFO - Kp: 5.566074512503456
2018-08-04 23:47:55,904 - mycodo.pid_3b533dff - INFO - Ki: 0.11927040744014512
2018-08-04 23:47:55,904 - mycodo.pid_3b533dff - INFO - Kd: 4.101408080354794
```

4.4.4 Conditional

Conditional controllers are used to perform certain Actions based a user-generated Conditional Statement.

Conditional Options

Setting	Description
Conditional Statement	User-created Python 3 code that will be executed.
Conditional Status	A dictionary can be returned that allows information to be passed to other controllers and widgets. For example, the Function Status Widget will display this information on the Dashboard. This code can be removed if you do not want to return any information.
Period (seconds)	The period (seconds) that the Conditional Statement will be executed.
Start Offset (seconds)	The duration (seconds) to wait before executing the Conditional for the first after it is activated.
Log Level: Debug	Show debug lines in the daemon log.
Message Includes Code	Include the Conditional Statement code in the message (self.message) that is passed to Actions.

Conditions are functions that can be used within the Conditional Statement, and return specific information.

Condition	Description
Measurement (Single, Last)	Acquires the latest measurement from an Input or device. Set Max Age (seconds) to restrict how long to accept values. If the latest value is older than this duration, "None" is returned.
Measurement (Single, Past, Average)	Acquires the past measurements from an Input or device, then averages them. Set Max Age (seconds) to restrict how long to accept values. If all values are older than this duration, "None" is returned.
Measurement (Single, Past, Sum)	Acquires the past measurements from an Input or device, then sums them. Set Max Age (seconds) to restrict how long to accept values. If all values are older than this duration, "None" is returned.
Measurement (Multiple, Past)	Acquires the past measurements from an Input or device. Set Max Age (seconds) to restrict how long to accept values. If no values are found in this duration, "None" is returned. This differs from the "Measurement (Single)" Condition because it returns a list of dictionaries with 'time' and 'value' key pairs.
GPIO State	Acquires the current GPIO state and returns 1 if HIGH or 0 if LOW. If the latest value is older than this duration, "None" is returned.
Output State	Returns 'on' if the output is currently on, and 'off' if it's currently off.
Output Duration On	Returns how long the output has currently been on, in seconds. Returns 0 if off.
Controller Running	Returns True if the controller is active, False if inactive.
Max Age (seconds)	The minimum age (seconds) the measurement can be. If the last measurement is older than this, "None" will be returned instead of a measurement.

Conditional Setup Guide

Python 3 is the environment that these conditionals will be executed. The following functions can be used within your Conditional Statement code.

> **Note**
>
> Python code indentations must use 4 spaces (not 2 spaces, tabs, or anything else).

Function	Description
self.condition("{ID}")	Returns a measurement for the Condition with ID.
self.condition_dict("{ID}")	Returns a dictionary of measurement for the Condition with ID.
self.run_action("{ID}")	Executes the Action with ID.
self.run_all_actions()	Executes all actions.
self.logger.info()	Writes a log line to the Daemon log. "info" may also be changed to "warning", "error" or "debug". Debug log lines will only appear in the Daemon log when Logging Level: Debug is enabled for the Input.
self.set_custom_option("option", value)	Writes the value to the database for retrieval later. The option argument should be a string, and value can be a string, integer, float, list, or dictionary.
self.get_custom_option("option")	Reads the value from the database that was previously written with self.set_custom_option(). Returns None if the option is not found or there is no value.

There are additional functions that can be used, but these must use the full UUID (not an abridged version as the functions above). See /home/pi/Mycodo/mycodo/mycodo_client.py for the functions available for use. These may be accessed via the 'control' object. An example, below, will return how long the output has been on (or 0 if it's currently off):

```
output_on_seconds = control.output_sec_currently_on("1b6ada50-1e69-403a-9fa6-ec748b16dc23")
```

Since the Python code contained in the Conditional Statement must be formatted properly, it's best to familiarize yourself with the basics of Python.

Note

There are two different IDs in use here, one set of IDs are found under the `Conditions` section of the Conditional Controller, and one set of IDs are found under the `Actions` section of the Conditional Controller. Read all of this section, including the examples, below, to fully understand how to properly set up a Conditional Controller.

Info

If a measurement hasn't been acquired within the set `Max Age`, "None" will be returned when self.condition("{ID}") is called in the code. It is very important that you account for this. All examples below incorporate a test for the measurement being None, and this should not be removed. If an error occurs (such as if the statement resolves to comparing None to a numerical value, such as "if None < 23"), then the code will stop there and an error will be logged in the daemon log. Accounting for None is useful for determining if an Input is no longer acquiring measurements (e.g. dead sensor, malfunction, etc.).

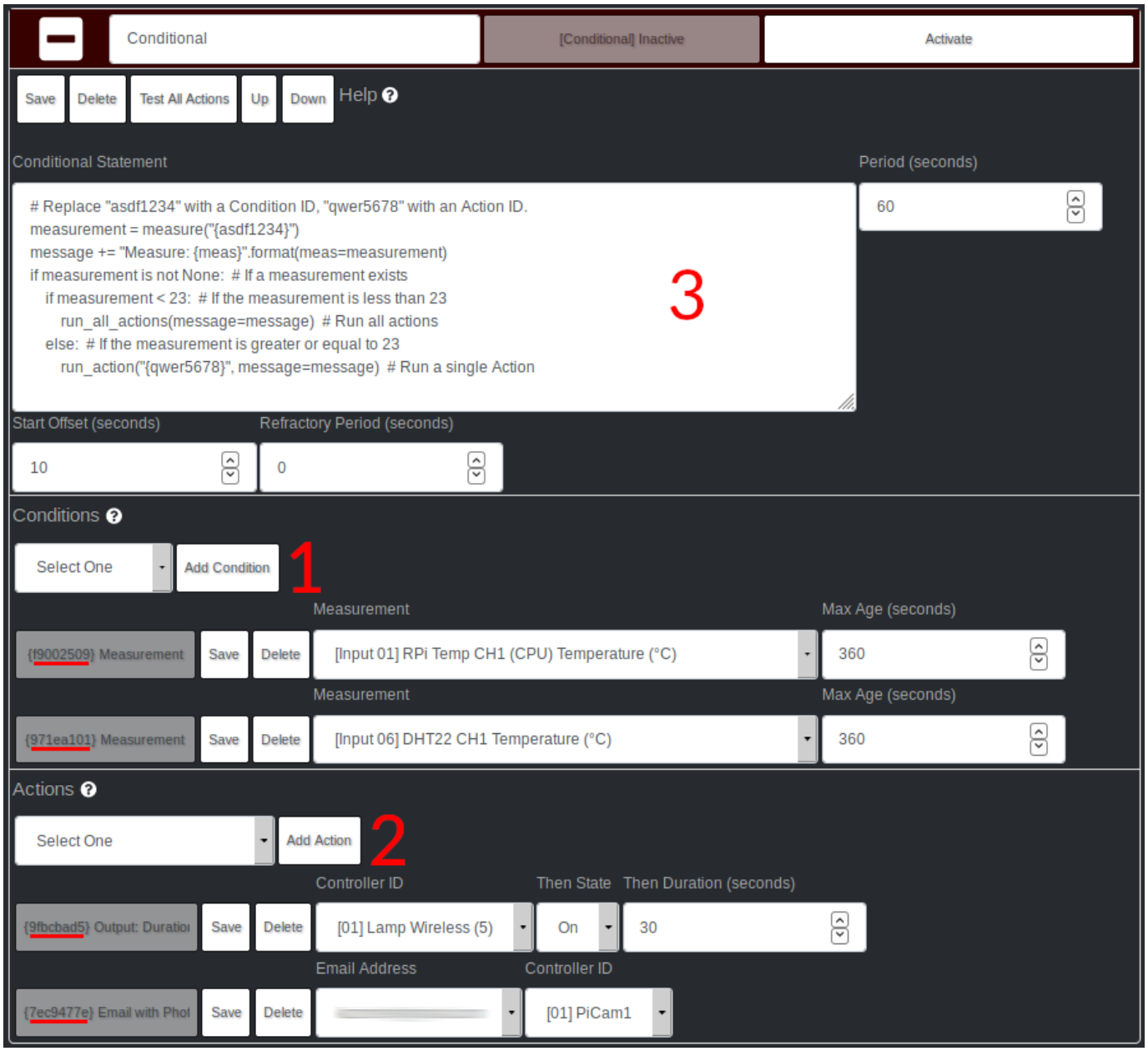
To create a basic conditional, follow these steps, using the numbers in the screenshots, below, that correspond to the numbers in parentheses:

- Navigate to the `Setup -> Function` page.
- Select "Controller: Conditional", then click `Add`.
- Under Conditions (1), select a condition option, then click `Add Condition`.
- Configure the newly-added Condition then click `Save`.
- Under Actions (2), select an action option, then click `Add Action`.
- Configure the newly-added Action then click `Save`.
- Notice that each Condition and each Action has its own ID (underlined).
- The default Conditional Statement (3) contains placeholder IDs that need to be changed to your Condition and Action IDs. Change the ID in self.condition("{asdf1234}") to your Condition ID. Change the ID in self.run_action("{qwer5678}", message=message) to your Action ID. Click `Save` at the top of the Conditional.
- The logic used in the Conditional Statement will need to be adjusted to suit your particular needs. Additionally, you may add more Conditions or Actions. See the `Advanced Conditional Statement examples`, below, for usage examples.

If your `Conditional Statement` has been formatted correctly, your Conditional will save and it will be ready to activate. If an error is returned, your options will not have been saved. Inspect the error for which line is causing the issue and read the error message itself to try to understand what the problem is and how to fix it. There are an unfathomable number of ways to configure a Conditional, but this should hopefully get you started to developing one that suits your needs.

Note

Mycodo is constantly changing, so the screenshots below may not match what you see exactly. Be sure to read this entire section of the manual to understand how to use Conditional Controllers.



Beginner `Conditional Statement` examples:

Each self.condition("{ID}") will return the most recent measurement obtained from that particular measurement under the `Conditions` section of the Conditional Controller, as long as it's within the set Max Age.

```
# Example 1, no measurement (i.e. None) returned
# useful with the Email Notify Action to email when an Input stops working
if self.condition("{asdf1234}") is None:
    self.run_all_actions()  # Execute all configured actions

# Example 2, test two measurement conditions
measure_1 = self.condition("{asdf1234}")
measure_2 = self.condition("{hjkl5678}")
if None not in [measure_1, measure_2]:
    # If neither measurement is None (both are working)
    if measure_1 < 20 and measure_2 > 10:
        # If measure_1 is less than 20 and measure_2 is greater than 10
        self.run_all_actions()  # Execute all configured actions

# Example 3, test two measurements and sum of measurements
measure_1 = self.condition("{asdf1234}")
measure_2 = self.condition("{hjkl5678}")
if None not in [measure_1, measure_2]:
    sum_ = measure_1 + measure_2
    if measure_1 > 2 and 10 < measure_2 < 23 and sum_ < 30.5:
        self.run_all_actions()
```

```
# Example 4, combine into one conditional
measurement = self.condition("{asdf1234}")
if measurement is not None and 20 < measurement < 30:  # combine conditions
    self.run_all_actions()

# Example 5, test two measurements
# convert Edge Input from 0 or 1 to True or False
measure_1 = self.condition("{asdf1234}")
measure_2 = self.condition("{hjkl5678}")
if None not in [measure_1, measure_2]:
    if bool(measure_1) and measure_2 > 10:
        self.run_all_actions()

# Example 6, test measurement with "or" and a rounded measurement
measure_1 = self.condition("{asdf1234}")
measure_2 = self.condition("{hjkl5678}")
if None not in [measure_1, measure_2]:
    if measure_1 > 20 or int(round(measure_2)) in [20, 21, 22]:
        self.run_all_actions()

# Example 7, use self to store variables across multiple executions
measurement = self.condition("{asdf1234}")
if not hasattr(self, "stored_measurement"):  # Initialize variable
    self.stored_measurement = measurement
if measurement is not None:
    if abs(measurement - self.stored_measurement) > 10:
        self.run_all_actions()  # if difference is greater than 10
    self.stored_measurement = measurement  # Store measurement
```

The "Measurement (Multiple)" Condition is useful if you desire to check if a particular value has been stored in any of the past measurements (within the set Max Age), not just the last measurement. This is useful if you have an alert system that each numerical value represents a different alert that you need to check each past value if it occurred. Here is an example that retrieves all measurements from the past 30 minutes and checks if any of the measurements in the returned list is equal to "119". If "119" exists, the Actions are executed and `break` is used to exit the `for` loop.

```
# Example 1, find a measurement in the past 30 minutes (Max Age: 1800 seconds)
measurements = self.condition_dict("{asdf1234}")
if measurements:  # If the list is not empty
    for each_measure in measurements:  # Loop through each measurement in the list
        if each_measure['value'] == 119:
            self.logger.info("Alert 119 found at timestamp {time}".format(
                time=each_measure['time']))
            self.run_all_actions()
            break  # Exit the for loop
```

Advanced `Conditional Statement` examples:

These examples expand on the beginner examples, above, by activating specific actions. The following examples will reference actions with IDs that can be found under the `Actions` section of the Conditional Controller. Two example action IDs will be used: "qwer1234" and "uiop5678". Additionally, self.run_all_actions() is used here, which will run all actions in the order in which they were created.

```
# Example 1
measurement = self.condition("{asdf1234}")
if measurement is None:
    self.run_action("{qwer1234}")
elif measurement > 23:
    self.run_action("{uiop5678}")
else:
    self.run_all_actions()

# Example 2, test two measurements
measure_1 = self.condition("{asdf1234}")
measure_2 = self.condition("{hjkl5678}")
if None not in [measure_1, measure_2]:
    if measure_1 < 20 and measure_2 > 10:
        self.run_action("{qwer1234}")
        self.run_action("{uiop5678}")

# Example 3, test two measurements and sum of measurements
measure_1 = self.condition("{asdf1234}")
measure_2 = self.condition("{hjkl5678}")
if None not in [measure_1, measure_2]:
    sum_ = measure_1 + measure_2
    if measure_1 > 2 and 10 < measure_2 < 23 and sum_ < 30.5:
        self.run_action("{qwer1234}")
    else:
        self.run_action("{uiop5678}")

# Example 4, combine into one conditional
measurement = self.condition("{asdf1234}")
if measurement is not None and 20 < measurement < 30:
    self.run_action("{uiop5678}")

# Example 5, test two measurements, convert Edge Input from 0/1 to True/False
```

```
measure_1 = self.condition("{asdf1234}")
measure_2 = self.condition("{hjkl5678}")
if None not in [measure_1, measure_2]:
    if bool(measure_1) and measure_2 > 10:
        self.run_all_actions()

# Example 6, test measurement with "or" and a rounded measurement
measure_1 = self.measure("{asdf1234}")
measure_2 = self.measure("{hjkl5678}")
if None not in [measure_1, measure_2]:
    if measure_1 > 20 or int(round(measure_2)) in [20, 21, 22]:
        self.run_action("{qwer1234}")
        if measure_1 > 30:
            self.run_action("{uiop5678}")
```

If your Action is a type that receives a message (E-Mail or Note), you can modify this message to include extra information before it is passed to the function (so the new information is passed to the Note, E-Mail, etc.). To do this, append a string to the variable `self.message` and add this to the `message` parameter of self.run_action() or self.run_all_actions(). Below are some examples. Note the use of "+=" instead of "=", which appends the string to the variable `self.message` instead of overwriting it.

```
# Example 1
measurement = self.measure("{asdf1234}")
if measurement is None and measurement > 23:
    self.message += "Measurement was {}".format(measurement)
    self.run_action("{uiop5678}", message=self.message)

# Example 2
measure_1 = self.measure("{asdf1234}")
measure_2 = self.measure("{hjkl5678}")
if None not in [measure_1, measure_2]:
    if measure_1 < 20 and measure_2 > 10:
        self.message += "Measurement 1: {m1}, Measurement 2: {m2}".format(
            m1=measure_1, m2=measure_2)
        self.run_all_actions(message=self.message)
```

Logging can also be used to log messages to the daemon log using `self.logger`. Logging levels include "info", "warning", "error" and "debug". Debug log lines will only appear in the Daemon log when Logging Level: Debug is enabled for the Input.

```
# Example 1
measurement = self.measure("{asdf1234}")
if measurement is None and measurement > 23:
    self.logging.error("Warning, measurement was {}".format(measurement))
    self.message += "Measurement was {}".format(measurement)
    self.run_action("{uiop5678}", message=self.message)
```

Before activating any conditionals, it's advised to thoroughly explore all possible scenarios and plan a configuration that eliminates conflicts. Some devices or outputs may respond atypically or fail when switched on and off in rapid succession. Therefore, trial run your configuration before connecting devices to any outputs.

4.4.5 Trigger

A Trigger Controller will execute actions when events are triggered, such as an output turning on or off, a GPIO pin changing it's voltage state (Edge detection, rising or falling), timed events that include various timers (duration, time period, time point, etc), or the sunrise/sunset time at a specific latitude and longitude. Once the trigger is configured, add any number of Actions to be executed when that event is triggered.

Output (On/Off) Options

Monitor the state of an output.

Setting	Description
If Output	The Output to monitor for a change of state.
If State	If the state of the output changes to On or Off the conditional will trigger. If "On (any duration) is selected, th trigger will occur no matter how long the output turns on for, whereas if only "On" is selected, the conditional will trigger only when the output turns on for a duration of time equal to the set "Duration (seconds)".
If Duration (seconds)	If "On" is selected, an optional duration (seconds) may be set that will trigger the conditional only if the Output is turned on for this specific duration.

Output (PWM) Options

Monitor the state of a PWM output.

Setting	Description
If Output	The Output to monitor for a change of state.
If State	If the duty cycle of the output is greater than,less than, or equal to the set value, trigger the Conditional Actions.
If Duty Cycle (%)	The duty cycle for the Output to be checked against.

Edge Options

Monitor the state of a pin for a rising and/or falling edge.

Setting	Description
If Edge Detected	The conditional will be triggered if a change in state is detected, either Rising when the state changes from LOW (0 volts) to HIGH (3.5 volts) or Falling when the state changes from HIGH (3.3 volts) to LOW (0 volts), or Both (Rising and Falling).

Run PWM Method Options

Select a Duration Method and this will set the selected PWM Output to the duty cycle specified by the method.

Setting	Description
Duration Method	Select which Method to use.
PWM Output	Select which PWM Output to use.
Period (seconds)	Select the interval of time to calculate the duty cycle, then apply to the PWM Output.
Trigger Every Period	Trigger Conditional Actions every period.
Trigger when Activated	Trigger Conditional Actions when the Conditional is activated.

Sunrise/Sunset Options

Trigger events at sunrise or sunset (or a time offset of those), based on latitude and longitude.

Setting	Description
Rise or Set	Select which to trigger the conditional, at sunrise or sunset.
Latitude (decimal)	Latitude of the sunrise/sunset, using decimal format.
Longitude (decimal)	Longitude of the sunrise/sunset, using decimal format.
Zenith	The Zenith angle of the sun.
Date Offset (days)	Set a sunrise/sunset offset in days (positive or negative).
Time Offset (minutes)	Set a sunrise/sunset offset in minutes (positive or negative).

Timer (Duration) Options

Run a timer that triggers Conditional Actions every period.

Setting	Description
Period (seconds)	The period of time between triggering Conditional Actions.
Start Offset (seconds)	Set this to start the first trigger a number of seconds after the Conditional is activated.

Timer (Daily Time Point) Options

Run a timer that triggers Conditional Actions at a specific time every day.

Setting	Description
Start Time (HH:MM)	Set the time to trigger Conditional Actions, in the format "HH:MM", with HH denoting hours, and MM denoting minutes. Time is in 24-hour format.

Timer (Daily Time Span) Options

Run a timer that triggers Conditional Actions at a specific period if it's between the set start and end times. For example, if the Start Time is set to 10:00 and End Time set to 11:00 and Period set to 120 seconds, the Conditional Actions will trigger every 120 seconds when the time is between 10 AM and 11 AM.

This may be useful, for instance, if you desire an Output to remain on during a particular time period and you want to prevent power outages from interrupting the cycle (which a simple Time Point Timer could not prevent against because it only triggers once at the Start Time). By setting an Output to turn the lights on every few minutes during the Start -> End period, it ensured the Output remains on during this period.

Setting	Description
Start Time (HH:MM)	Set the start time to trigger Conditional Actions, in the format "HH:MM", with HH denoting hours, and MM denoting minutes. Time is in 24-hour format.
End Time (HH:MM)	Set the end time to trigger Conditional Actions, in the format "HH:MM", with HH denoting hours, and MM denoting minutes. Time is in 24-hour format.
Period (seconds)	The period of time between triggering Conditional Actions.

4.5 Actions

These are the actions that can be added to Controllers (i.e. Input, Conditional, and Trigger Controllers) to provide a way to add additional functionality or interact with other parts of Mycodo. Actions may work with one or more controller type, depending on how the Action has been designed.

For a full list of supported Actions, see Supported Actions.

4.5.1 Custom Actions

There is a Custom Action import system in Mycodo that allows user-created Actions to be used in the Mycodo system. Custom Actions can be uploaded on the `[Gear Icon] -> Configure -> Custom Actions` page. After import, they will be available to use on the `Setup -> Function` page.

If you develop a working Action module, please consider creating a new GitHub issue or pull request, and it may be included in the built-in set.

Open any of the built-in modules located in the directory Mycodo/mycodo/actions for examples of the proper formatting.

There are also example Custom Actions in the directory Mycodo/mycodo/actions/examples

Additionally, I have another github repository devoted to Custom Modules that are not included in the built-in set, at kizniche/Mycodo-custom.

4.6 Calibration

Calibration can be performed for any Input, Output, or Function if that functionality has been built in to the module. Some common modules that have calibration are several of the Atlas Scientific, MH-Z19, and DS-type Inputs and many of the peristaltic pump Outputs. Calibration actions can be found on the options page for the particular device. Refer to the calibration instructions at this location for how to perform a successful calibration.

4.7 Methods

Page: `Setup -> Method`

Methods enable Setpoint Tracking in PIDs and time-based duty cycle changes in timers. Normally, a PID controller will regulate an environmental condition to a specific setpoint. If you would like the setpoint to change over time, this is called setpoint tracking. Setpoint Tracking is useful for applications such as reflow ovens, thermal cyclers (DNA replication), mimicking natural daily cycles, and more. Methods may also be used to change a duty cycle over time when used with a Run PWM Method Conditional.

4.7.1 Method Options

These options are shared with several method types.

Setting	Description
Start Time/Date	This is the start time of a range of time.
End Time/Date	This is the end time of a range of time.
Start Setpoint	This is the start setpoint of a range of setpoints.
End Setpoint	This is the end setpoint of a range of setpoints.

4.7.2 Time/Date Method

A time/date method allows a specific time/date span to dictate the setpoint. This is useful for long-running methods, that may take place over the period of days, weeks, or months.

4.7.3 Duration Method

A Duration Method allows a **Setpoint** (for PIDs) or **Duty Cycle** (for Conditional) to be set after specific durations of time. Each new duration added will stack, meaning it will come after the previous duration, meaning a newly-added **Start Setpoint** will begin after the previous entry's **End Setpoint**.

If the "Repeat Method" option is used, this will cause the method to repeat once it has reached the end. If this option is used, no more durations may be added to the method. If the repeat option is deleted then more durations may be added. For instance, if your method is 200 seconds total, if the Repeat Duration is set to 600 seconds, the method will repeat 3 times and then automatically turn off the PID or Conditional.

4.7.4 Daily (Time-Based) Method

The daily time-based method is similar to the time/date method, however it will repeat every day. Therefore, it is essential that only the span of one day be set in this method. Begin with the start time at 00:00:00 and end at 23:59:59 (or 00:00:00, which would be 24 hours from the start). The start time must be equal or greater than the previous end time.

4.7.5 Daily (Sine Wave) Method

The daily sine wave method defines the setpoint over the day based on a sinusoidal wave. The sine wave is defined by y = [A * sin(B * x + C)] + D, where A is amplitude, B is frequency, C is the angle shift, and D is the y-axis shift. This method will repeat daily.

4.7.6 Daily (Bezier Curve) Method

A daily Bezier curve method define the setpoint over the day based on a cubic Bezier curve. If unfamiliar with a Bezier curve, it is recommended you use the graphical Bezier curve generator and use the 8 variables it creates for 4 points (each a set of x and y).

The x-axis start (x3) and end (x0) will be automatically stretched or skewed to fit within a 24-hour period and this method will repeat daily.

4.7.7 Cascade Method

This method combines multiple methods and outputs the average of the methods. For examples, let's combine a Duration method set to 100 for 60 seconds and 0 for 60 seconds (and set to repeat forever) with a Daily Method that rises from 0 at 00:00:00 to 50 at 12:00:00, and falls back to 0 at 23:59:59. At 00:00:00, the combined methods would produce an output that oscillates from 0 ((0 / 100) * (0 / 100) = 0) to 0 ((100 / 100) * (0 / 100) = 0) every 60 seconds, and gradually increase until at 12:00:00 the output would be oscillating from 0 ((0 / 100) * (50 / 100)) to 50 ((100 / 100) * (50 / 100)) every 60 seconds. This is a simple example, but combinations can become very complex.

4.8 Alerts

Alerts can be used to notify users about the state of the system. For things like sensor monitoring, this could be a threshold that indicates something needs attention. E-Mail notifications are built-in to Mycodo in a number of places, however there are several places (Inputs, Outputs, Controllers) that allow custom Python code to be used, enabling many other notification options to be built.

See Alert Settings for more information about setting up Alerts.

4.9 Notes

Page: `More -> Notes`

Notes may be created that can then be displayed on graphs or referenced at a later time. All notes are timestamped with the date/time of creation or may be created with a custom date/time. Each note must have at least one tag selected. Tags are what are selected to be displayed on a graph and all notes with that tag will appear in the time frame selected on the graph.

4.9.1 Tag Options

Setting	Description
Name	A name for the tag. Must not contain spaces.
Rename	Rename the tag.

4.9.2 Note Options

Setting	Description
Name	A name for the note.
Use Custom Date/Time	Check to enter a custom date/time for the note.
Custom Date/Time	Store the note with this custom date/time.
Attached Files	Attach one or more files to the note.
Tags	Associate the note with at least one tag.
Note	The text body of the note. The text will appear monospaced, so code will format properly.

4.10 Camera

Page: `More -> Camera`

Cameras can be used to capture still images, create time-lapses, and stream video. Cameras may also be used by Functions to trigger a camera image or video capture.

There are several libraries that may be used to access your camera, which includes picamera (Raspberry Pi Camera), fswebcam, opencv, urllib, and requests (among potentially others). These libraries enable images to be acquired from the Raspberry Pi camera, USB cameras and webcams, and IP cameras that are accessible by a URL. Furthermore, using the urllib and request libraries, any image URL can be used to acquire images.

4.11 Energy Usage

Page: `More -> Energy Usage`

There are two methods for calculating energy usage. The first relies on determining how long Outputs have been on. Based on this, if the number of Amps the output draws has been set in the output Settings, then the kWh and cost can be calculated. Discovering the number of amps the device draws can be accomplished by calculating this from the output typically given as watts on the device label, or with the use of a current clamp while the device is operating. The limitation of this method is PWM Outputs are not currently used to calculate these figures due to the difficulty determining the current consumption of devices driven by PWM signals.

The second method for calculating energy consumption is more accurate and is the recommended method if you desire the most accurate estimation of energy consumption and cost. This method relies on an Input or Function measuring Amps. One way to do this is with the used of an analog-to-digital converter (ADC) that converts the voltage output from a transformer into current (Amps). One wire from the AC line that powers your device(s) passes thorough the transformer and the device converts the current that passes through that wire into a voltage that corresponds to the amperage. For instance, the below sensor converts 0 -50 amps input to 0 - 5 volts output. An ADC receives this output as its input. One would set this conversion range in Mycodo and the calculated amperage will be stored. On the Energy Usage page, add this ADC Input measurement and a report summary will be generated. Keep in mind that for a particular period (for example, the past week) to be accurate, there needs to be a constant measurement of amps at a periodic rate. The faster the rate the more accurate the calculation will be. This is due to the amperage measurements being averaged for this period prior to calculating kWh and cost. If there is any time turing this period where amp measurements aren't being acquired when in fact there are devices consuming current, the calculation is likely to not be accurate.

Greystone CS-650-50 AC Solid Core Current Sensor (Transformer)

The following settings are for calculating energy usage from an amp measurement. For calculating based on Output duration, see Energy Usage Settings.

Setting	Description
Select Amp Measurement	This is a measurement with the amp (A) units that will be used to calculate energy usage.

4.12 Python Code

There are numerous places where Python 3 code can be used within Mycodo, including the Python Code Input, the Python Code Output, and Conditional Controller Functions.

Here are a few example that demonstrates some useful ways to interact with Mycodo with Python 3 code.

In all the Mycodo environments where your code will be executed, the DaemonControl() Class of mycodo/mycodo_client.py is available to communicate with the daemon using the object "control".

4.12.1 Outputs

PWM Fan with a Minimum Duty Cycle to Spin

Some PWM-controlled fans do not start spinning until a minimum duty cycle is set. Once the fan is spinning, the duty cycle can be set much lower and the fan will continue to spin. Because of this, there needs to be a "charging" step if the fan is turning on from a duty cycle of 0. This code detects if the requested duty cycle will need to execute the charging step prior to setting the duty cycle. For this, you will need A GPIO PWM Output and a Python Code PWM Output. The GPIO PWM Output will be configured for the fan, and the Python Code PWM Output will be configured with the following code:

```
import time

# Set the variables the first time the code is executed
if not hasattr(self, "output_id_gpio_pwm"):
    self.logger.debug("Initializing")
    self.output_id_gpio_pwm = "a3dade60-091a-49d7-9c79-cd2adf41bc23"  # UUID of GPIO PWM Output
    self.fan_spinning = False  # saves the state of the fan
    self.fan_min_duty_cycle = 2  # The lowest duty cycle that will keep the fan spinning
    self.fan_spin_duty_cycle = 25  # The minimum duty cycle to get the fan spinning if it's been off
    self.fan_charge_duty_cycle = 45  # The charging duty cycle to get the fan initially spinning
    self.fan_spin_duration_sec = 1.5  # The duration to run the fan at the charge duty cycle

# Charge the fan if it's not spinning and the desired duty cycle is too low
if duty_cycle and not self.fan_spinning and duty_cycle < self.fan_spin_duty_cycle:
    self.logger.debug("Duty cycle too low and fan is off. Charging.")
    self.logger.debug("Setting duty cycle of {} %".format(self.fan_charge_duty_cycle))
    control.output_on(self.output_id_gpio_pwm,
                      output_type='pwm',
                      amount=self.fan_charge_duty_cycle,
                      output_channel=0)
    time.sleep(self.fan_spin_duration_sec)
    self.fan_spinning = True

if duty_cycle == 0:
    self.logger.debug("Fan turned off")
    self.fan_spinning = False
elif duty_cycle > self.fan_spin_duty_cycle:
    self.fan_spinning = True

self.logger.debug("Setting duty cycle of {} %".format(duty_cycle))
control.output_on(self.output_id_gpio_pwm,
                  output_type='pwm',
                  amount=duty_cycle,
                  output_channel=0)
```

5. Supported Devices

5.1 Inputs Sorted by Measurement

Measurements

- Acceleration
- Acceleration (X)
- Acceleration (Y)
- Acceleration (Z)
- ADC
- Altitude
- Angle
- Battery
- Boolean
- CO2
- Color (Y)
- Color (Blue)
- Color (Green)
- Color (Red)
- Color (x)
- Color (y)
- CPU Load (15 min)
- CPU Load (1 min)
- CPU Load (5 min)
- Dewpoint
- Direction
- Disk
- Dissolved Oxygen
- Duration
- Duty Cycle
- GPIO Edge
- Electrical Conductivity
- Electrical Current
- Electrical Potential
- Energy
- Frequency
- GPIO State
- Humidity
- Ion Concentration
- Length
- Light
- Magnetic Flux Density
- Moisture
- Oxidation Reduction Potential
- PM10
- PM1
- PM2.5
- Power

- Apparent Power
- Power Factor
- Reactive Power
- Pressure
- Pulse Width
- Volume Flow Rate
- Resistance
- Revolutions
- Salinity
- Specific Gravity
- Speed
- Temperature
- Total Dissolved Solids
- Vapor Pressure Deficit
- Version
- VOC
- Volume

5.1.1 Acceleration

Ruuvi: RuuviTag

5.1.2 Acceleration (X)

Analog Devices: ADXL34x (343, 344, 345, 346)

Raspberry Pi Foundation: Sense HAT

Ruuvi: RuuviTag

5.1.3 Acceleration (Y)

Analog Devices: ADXL34x (343, 344, 345, 346)

Raspberry Pi Foundation: Sense HAT

Ruuvi: RuuviTag

5.1.4 Acceleration (Z)

Analog Devices: ADXL34x (343, 344, 345, 346)

Raspberry Pi Foundation: Sense HAT

Ruuvi: RuuviTag

5.1.5 ADC

AMS: AS7262

5.1.6 Altitude

BOSCH: BME280

BOSCH: BME280

BOSCH: BME280

BOSCH: BME680

BOSCH: BME680

BOSCH: BMP180

BOSCH: BMP280

BOSCH: BMP280

5.1.7 Angle

Raspberry Pi Foundation: Sense HAT

5.1.8 Battery

Ruuvi: RuuviTag

Sensorion: SHT31 Smart Gadget

Xiaomi: Miflora

Xiaomi: Mijia LYWSD03MMC (ATC and non-ATC modes)

5.1.9 Boolean

Mycodo: Server Ping

Mycodo: Server Port Open

5.1.10 CO2

AMS: CCS811 (with Temperature)

AMS: CCS811 (without Temperature)

Atlas Scientific: Atlas CO2

CO2Meter: K30

Cozir: Cozir CO2

Sensirion: SCD-4x (SCD-40, SCD-41)

Sensirion: SCD30

Sensirion: SCD30

Winsen: MH-Z16

Winsen: MH-Z19

Winsen: MH-Z19B

5.1.11 Color (Y)

Atlas Scientific: Atlas Color

5.1.12 Color (Blue)

Atlas Scientific: Atlas Color

5.1.13 Color (Green)

Atlas Scientific: Atlas Color

5.1.14 Color (Red)

Atlas Scientific: Atlas Color

5.1.15 Color (x)

Atlas Scientific: Atlas Color

5.1.16 Color (y)

Atlas Scientific: Atlas Color

5.1.17 CPU Load (15 min)

Mycodo: CPU Load

5.1.18 CPU Load (1 min)

Mycodo: CPU Load

5.1.19 CPU Load (5 min)

Mycodo: CPU Load

5.1.20 Dewpoint

AOSONG: AM2315/AM2320

AOSONG: DHT11

AOSONG: DHT22

Atlas Scientific: Atlas Humidity

BOSCH: BME280

BOSCH: BME280

BOSCH: BME280

BOSCH: BME680

BOSCH: BME680

Cozir: Cozir CO2

Ruuvi: RuuviTag

Seeedstudio: DHT11/22

Sensirion: SCD-4x (SCD-40, SCD-41)

Sensirion: SCD30

Sensirion: SCD30

Sensirion: SHT1x/7x

Sensirion: SHT2x

Sensirion: SHT2x

Sensirion: SHT31-D

Sensirion: SHT3x (30, 31, 35)

Sensirion: SHT4X

Sensirion: SHTC3

Sensorion: SHT31 Smart Gadget

Silicon Labs: Si7021

Sonoff: TH16/10 (Tasmota firmware) with AM2301/Si7021

Sonoff: TH16/10 (Tasmota firmware) with AM2301

TE Connectivity: HTU21D

TE Connectivity: HTU21D

Texas Instruments: HDC1000

Weather: OpenWeatherMap (City, Current)

Weather: OpenWeatherMap (Lat/Lon, Current/Future)

5.1.21 Direction

Raspberry Pi Foundation: Sense HAT

Weather: OpenWeatherMap (City, Current)

Weather: OpenWeatherMap (Lat/Lon, Current/Future)

5.1.22 Disk

Mycodo: Free Space

Mycodo: Mycodo RAM

5.1.23 Dissolved Oxygen

Atlas Scientific: Atlas DO

5.1.24 Duration

Weather: OpenWeatherMap (Lat/Lon, Current/Future)

5.1.25 Duty Cycle

Raspberry Pi: Signal (PWM)

5.1.26 GPIO Edge

Raspberry Pi: Edge Detection

5.1.27 Electrical Conductivity

AnyLeaf: AnyLeaf EC

Atlas Scientific: Atlas EC

Texas Instruments: ADS1115: Generic Analog pH/EC

Texas Instruments: ADS1256: Generic Analog pH/EC

Xiaomi: Miflora

5.1.28 Electrical Current

Tasmota: Tasmota Outlet Energy Monitor (HTTP)

Texas Instruments: INA219x

5.1.29 Electrical Potential

Microchip: MCP3008

Microchip: MCP342x (x=2,3,4,6,7,8)

Tasmota: Tasmota Outlet Energy Monitor (HTTP)

Texas Instruments: ADS1015

Texas Instruments: ADS1115

Texas Instruments: ADS1256: Generic Analog pH/EC

Texas Instruments: ADS1256

Texas Instruments: ADS1x15

Texas Instruments: INA219x

5.1.30 Energy

Tasmota: Tasmota Outlet Energy Monitor (HTTP)

5.1.31 Frequency

Raspberry Pi: Signal (PWM)

5.1.32 GPIO State

Raspberry Pi: GPIO State

5.1.33 Humidity

AOSONG: AM2315/AM2320

AOSONG: DHT11

AOSONG: DHT22

ASAIR: AHTx0

Atlas Scientific: Atlas Humidity

BOSCH: BME280

BOSCH: BME280

BOSCH: BME280

BOSCH: BME680

BOSCH: BME680

Cozir: Cozir CO2

Raspberry Pi Foundation: Sense HAT

Ruuvi: RuuviTag

Seeedstudio: DHT11/22

Sensirion: SCD-4x (SCD-40, SCD-41)

Sensirion: SCD30

Sensirion: SCD30

Sensirion: SHT1x/7x

Sensirion: SHT2x

Sensirion: SHT2x

Sensirion: SHT31-D

Sensirion: SHT3x (30, 31, 35)

Sensirion: SHT4X

Sensirion: SHTC3

Sensorion: SHT31 Smart Gadget

Silicon Labs: Si7021

Sonoff: TH16/10 (Tasmota firmware) with AM2301/Si7021

Sonoff: TH16/10 (Tasmota firmware) with AM2301

TE Connectivity: HTU21D

TE Connectivity: HTU21D

Texas Instruments: HDC1000

Weather: OpenWeatherMap (City, Current)

Weather: OpenWeatherMap (Lat/Lon, Current/Future)

Xiaomi: Mijia LYWSD03MMC (ATC and non-ATC modes)

5.1.34 Ion Concentration

AnyLeaf: AnyLeaf pH

Atlas Scientific: Atlas pH

Texas Instruments: ADS1115: Generic Analog pH/EC

Texas Instruments: ADS1256: Generic Analog pH/EC

5.1.35 Length

Atlas Scientific: Atlas Color

Multiple Manufacturers: HC-SR04

STMicroelectronics: VL53L0X

STMicroelectronics: VL53L1X

Silicon Labs: SI1145

5.1.36 Light

AMS: TSL2561

AMS: TSL2591

Atlas Scientific: Atlas Color

Catnip Electronics: Chirp

ROHM: BH1750

Silicon Labs: SI1145

Xiaomi: Miflora

5.1.37 Magnetic Flux Density

Melexis: MLX90393

Raspberry Pi Foundation: Sense HAT

5.1.38 Moisture

Adafruit: I2C Capacitive Moisture Sensor

Catnip Electronics: Chirp

Xiaomi: Miflora

5.1.39 Oxidation Reduction Potential

AnyLeaf: AnyLeaf ORP

Atlas Scientific: Atlas ORP

5.1.40 PM10

Winsen: ZH03B

5.1.41 PM1

Winsen: ZH03B

5.1.42 PM2.5

Winsen: ZH03B

5.1.43 Power

Tasmota: Tasmota Outlet Energy Monitor (HTTP)

5.1.44 Apparent Power

Tasmota: Tasmota Outlet Energy Monitor (HTTP)

5.1.45 Power Factor

Tasmota: Tasmota Outlet Energy Monitor (HTTP)

5.1.46 Reactive Power

Tasmota: Tasmota Outlet Energy Monitor (HTTP)

5.1.47 Pressure

Atlas Scientific: Atlas Pressure

BOSCH: BME280

BOSCH: BME280

BOSCH: BME280

BOSCH: BME680

BOSCH: BME680

BOSCH: BMP180

BOSCH: BMP280

BOSCH: BMP280

Infineon: DPS310

Raspberry Pi Foundation: Sense HAT

Ruuvi: RuuviTag

Weather: OpenWeatherMap (City, Current)

Weather: OpenWeatherMap (Lat/Lon, Current/Future)

5.1.48 Pulse Width

Raspberry Pi: Signal (PWM)

5.1.49 Volume Flow Rate

Atlas Scientific: Atlas Flow Meter

Generic: Hall Flow Meter

5.1.50 Resistance

BOSCH: BME680

BOSCH: BME680

5.1.51 Revolutions

Raspberry Pi: Signal (Revolutions)

5.1.52 Salinity

Atlas Scientific: Atlas EC

5.1.53 Specific Gravity

Atlas Scientific: Atlas EC

5.1.54 Speed

Weather: OpenWeatherMap (City, Current)

Weather: OpenWeatherMap (Lat/Lon, Current/Future)

5.1.55 Temperature

AMS: CCS811 (with Temperature)

AOSONG: AM2315/AM2320

AOSONG: DHT11

AOSONG: DHT22

ASAIR: AHTx0

Adafruit: I2C Capacitive Moisture Sensor

Analog Devices: ADT7410

Atlas Scientific: Atlas Humidity

Atlas Scientific: Atlas PT-1000

BOSCH: BME280

BOSCH: BME280

BOSCH: BME280

BOSCH: BME680

BOSCH: BME680

BOSCH: BMP180

BOSCH: BMP280

BOSCH: BMP280

Catnip Electronics: Chirp

Cozir: Cozir CO2

Infineon: DPS310

MAXIM: DS1822

MAXIM: DS1825

MAXIM: DS18B20

MAXIM: DS18B20

MAXIM: DS18S20

MAXIM: DS28EA00

MAXIM: MAX31850K

MAXIM: MAX31855

MAXIM: MAX31856

MAXIM: MAX31865

MAXIM: MAX31865

Melexis: MLX90614

Microchip: MCP9808

Panasonic: AMG8833

Raspberry Pi Foundation: Sense HAT

Raspberry Pi: CPU/GPU Temperature

Ruuvi: RuuviTag

Seeedstudio: DHT11/22

Sensirion: SCD-4x (SCD-40, SCD-41)

Sensirion: SCD30

Sensirion: SCD30

Sensirion: SHT1x/7x

Sensirion: SHT2x

Sensirion: SHT2x

Sensirion: SHT31-D

Sensirion: SHT3x (30, 31, 35)

Sensirion: SHT4X

Sensirion: SHTC3

Sensorion: SHT31 Smart Gadget

Silicon Labs: Si7021

Sonoff: TH16/10 (Tasmota firmware) with AM2301/Si7021

Sonoff: TH16/10 (Tasmota firmware) with AM2301

Sonoff: TH16/10 (Tasmota firmware) with DS18B20

TE Connectivity: HTU21D

TE Connectivity: HTU21D

Texas Instruments: HDC1000

Texas Instruments: TMP006

Weather: OpenWeatherMap (City, Current)

Weather: OpenWeatherMap (Lat/Lon, Current/Future)

Xiaomi: Miflora

Xiaomi: Mijia LYWSD03MMC (ATC and non-ATC modes)

5.1.56 Total Dissolved Solids

Atlas Scientific: Atlas EC

5.1.57 Vapor Pressure Deficit

AOSONG: AM2315/AM2320

AOSONG: DHT11

AOSONG: DHT22

BOSCH: BME280

BOSCH: BME280

BOSCH: BME280

BOSCH: BME680

BOSCH: BME680

Ruuvi: RuuviTag

Seeedstudio: DHT11/22

Sensirion: SCD-4x (SCD-40, SCD-41)

Sensirion: SCD30

Sensirion: SCD30

Sensirion: SHT1x/7x

Sensirion: SHT2x

Sensirion: SHT2x

Sensirion: SHT31-D

Sensirion: SHT3x (30, 31, 35)

Sensirion: SHT4X

Sensirion: SHTC3

Sensorion: SHT31 Smart Gadget

Silicon Labs: Si7021

Sonoff: TH16/10 (Tasmota firmware) with AM2301/Si7021

Sonoff: TH16/10 (Tasmota firmware) with AM2301

TE Connectivity: HTU21D

TE Connectivity: HTU21D

Texas Instruments: HDC1000

5.1.58 Version

Mycodo: Mycodo Version

5.1.59 VOC

AMS: CCS811 (with Temperature)

AMS: CCS811 (without Temperature)

5.1.60 Volume

Atlas Scientific: Atlas Flow Meter

Generic: Hall Flow Meter

5.2 Supported Inputs

Supported Inputs are listed below.

5.2.1 Built-In Inputs (System)

Linux: Bash Command

- Manufacturer: Linux
- Measurements: Return Value
- Interfaces: Mycodo

This Input will execute a command in the shell and store the output as a float value. Perform any unit conversions within your script or command. A measurement/unit is required to be selected.

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Command Timeout

- Type: Integer
- Default Value: 60
- Description: How long to wait for the command to finish before killing the process.

User

- Type: Text
- Default Value: mycodo
- Description: The user to execute the command

Current Working Directory

- Type: Text
- Default Value: /home/pi
- Description: The current working directory of the shell environment.

Linux: Python 3 Code

- Manufacturer: Linux
- Measurements: Store Value(s)
- Interfaces: Mycodo
- Dependencies: pylint

All channels require a Measurement Unit to be selected and saved in order to store values to the database.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Mycodo: CPU Load

- Manufacturer: Mycodo
- Measurements: CPULoad
- Libraries: os.getloadavg()

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Mycodo: Free Space

- Manufacturer: Mycodo
- Measurements: Unallocated Disk Space
- Libraries: os.statvfs()

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Mycodo: Mycodo RAM

- Manufacturer: Mycodo
- Measurements: Daemon RAM Allocation
- Libraries: resource.getrusage()

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Mycodo: Mycodo Version

- Manufacturer: Mycodo
- Measurements: Version as Major.Minor.Revision

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Mycodo: Server Ping

- Manufacturer: Mycodo
- Measurements: Boolean
- Libraries: ping

This Input executes the bash command "ping -c [times] -w [deadline] [host]" to determine if the host can be pinged.

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal

- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Mycodo: Server Port Open

- Manufacturer: Mycodo
- Measurements: Boolean
- Libraries: nc

This Input executes the bash command "nc -zv [host] [port]" to determine if the host at a particular port is accessible.

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Mycodo: Spacer

- Manufacturer: Mycodo

A spacer to organize Inputs.

OPTIONS

Color

- Type: Text
- Default Value: #000000
- Description: The color of the name text

Raspberry Pi: CPU/GPU Temperature

- Manufacturer: Raspberry Pi
- Measurements: Temperature
- Interfaces: RPi

The internal CPU and GPU temperature of the Raspberry Pi.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Path for CPU Temperature

- Type: Text
- Default Value: /sys/class/thermal/thermal_zone0/temp
- Description: Reads the CPU temperature from this file

Path to vcgencmd

- Type: Text
- Default Value: /usr/bin/vcgencmd
- Description: Reads the GPU from vcgencmd

Raspberry Pi: Edge Detection

- Manufacturer: Raspberry Pi
- Measurements: Rising/Falling Edge
- Interfaces: GPIO
- Libraries: RPi.GPIO
- Dependencies: RPi.GPIO

OPTIONS

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Pin Mode

- Type: Select
- Options: [**Floating** | Pull Down | Pull Up] (Default in **bold**)
- Description: Enables or disables the pull-up or pull-down resistor

Raspberry Pi: GPIO State

- Manufacturer: Raspberry Pi

- Measurements: GPIO State
- Interfaces: GPIO
- Libraries: RPi.GPIO
- Dependencies: RPi.GPIO

Measures the state of a GPIO pin, returning either 0 (low) or 1 (high).

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Pin Mode

- Type: Select
- Options: [**Floating** | Pull Down | Pull Up] (Default in **bold**)
- Description: Enables or disables the pull-up or pull-down resistor

Raspberry Pi: Signal (PWM)

- Manufacturer: Raspberry Pi
- Measurements: Frequency/Pulse Width/Duty Cycle
- Interfaces: GPIO
- Libraries: pigpio
- Dependencies: pigpio, pigpio

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Raspberry Pi: Signal (Revolutions)

- Manufacturer: Raspberry Pi
- Measurements: RPM
- Interfaces: GPIO
- Libraries: pigpio
- Dependencies: pigpio, pigpio

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

5.2.2 Built-In Inputs (Devices)

AMS: AS7262

- Manufacturer: AMS
- Measurements: Light at 450, 500, 550, 570, 600, 650 nm
- Interfaces: I^2C
- Libraries: as7262
- Dependencies: as7262
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Gain

- Type: Select
- Options: [1x | 3.7x | 16x | **64x**] (Default in **bold**)
- Description: Set the sensor gain

Illumination LED Current

- Type: Select
- Options: [**12.5 mA** | 25 mA | 50 mA | 100 mA] (Default in **bold**)
- Description: Set the illumination LED current (milliamps)

Illumination LED Mode

- Type: Select
- Options: [**On** | Off] (Default in **bold**)
- Description: Turn the illumination LED on or off during a measurement

Indicator LED Current

- Type: Select
- Options: [**1 mA** | 2 mA | 4 mA | 8 mA] (Default in **bold**)
- Description: Set the indicator LED current (milliamps)

Indicator LED Mode

- Type: Select
- Options: [**On** | Off] (Default in **bold**)
- Description: Turn the indicator LED on or off during a measurement

Integration Time

- Type: Decimal
- Default Value: 15.0
- Description: The integration time (0 - ~91 ms)

AMS: CCS811 (with Temperature)

- Manufacturer: AMS
- Measurements: CO2/VOC/Temperature
- Interfaces: I^2C
- Libraries: Adafruit_CCS811
- Dependencies: Adafruit_CCS811, Adafruit-GPIO
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URLs: Link 1, Link 2

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

AMS: CCS811 (without Temperature)

- Manufacturer: AMS
- Measurements: CO2/VOC
- Interfaces: I^2C
- Libraries: Adafruit_CircuitPython_CCS811
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-ccs811
- Manufacturer URL: Link

- Datasheet URL: Link
- Product URL: Link
- Additional URL: Link

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

AMS: TSL2561

- Manufacturer: AMS
- Measurements: Light
- Interfaces: I²C
- Libraries: tsl2561
- Dependencies: Adafruit-GPIO, Adafruit-PureIO, tsl2561
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

AMS: TSL2591

- Manufacturer: AMS
- Measurements: Light
- Interfaces: I²C
- Libraries: maxlklaxl/python-tsl2591
- Dependencies: tsl2591
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

AOSONG: AM2315/AM2320

- Manufacturer: AOSONG
- Measurements: Humidity/Temperature
- Interfaces: I^2C
- Libraries: quick2wire-api
- Dependencies: quick2wire-api
- Datasheet URL: Link
- Product URL: Link

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

AOSONG: DHT11

- Manufacturer: AOSONG
- Measurements: Humidity/Temperature
- Interfaces: GPIO
- Libraries: pigpio
- Dependencies: pigpio, pigpio
- Datasheet URL: Link
- Product URL: Link

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

AOSONG: DHT22

- Manufacturer: AOSONG
- Measurements: Humidity/Temperature
- Interfaces: GPIO
- Libraries: pigpio
- Dependencies: pigpio, pigpio
- Datasheet URL: Link
- Product URL: Link

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal

- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

ASAIR: AHTx0

- Manufacturer: ASAIR
- Measurements: Temperature/Humidity
- Interfaces: I^2C
- Libraries: Adafruit-CircuitPython-AHTx0
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-ahtx0
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Adafruit: I2C Capacitive Moisture Sensor

- Manufacturer: Adafruit
- Measurements: Moisture/Temperature
- Interfaces: I^2C
- Libraries: adafruit_seesaw
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-seesaw
- Manufacturer URL: Link
- Product URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Analog Devices: ADT7410

- Manufacturer: Analog Devices
- Measurements: Temperature
- Interfaces: I^2C
- Libraries: Adafruit_CircuitPython
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-adt7410
- Datasheet URL: Link
- Product URL: Link

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Analog Devices: ADXL34x (343, 344, 345, 346)

- Manufacturer: Analog Devices
- Measurements: Acceleration
- Interfaces: I²C
- Libraries: Adafruit_CircuitPython
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-adxl34x
- Datasheet URLs: Link 1, Link 2, Link 3, Link 4
- Product URLs: Link 1, Link 2, Link 3, Link 4

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Range

- Type: Select
- Options: [±2 g (±19.6 m/s/s) | ±4 g (±39.2 m/s/s) | ±8 g (±78.4 m/s/s) | **±16 g (±156.9 m/s/s)**] (Default in **bold**)
- Description: Set the measurement range

AnyLeaf: AnyLeaf EC

- Manufacturer: AnyLeaf
- Measurements: Electrical Conductivity
- Interfaces: UART
- Libraries: anyleaf
- Dependencies: libjpeg-dev, zlib1g-dev, Pillow, scipy, pyusb, Adafruit-extended-bus, anyleaf
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Conductivity Constant

- Type: Decimal
- Default Value: 1.0
- Description: Conductivity constant K

AnyLeaf: AnyLeaf ORP

- Manufacturer: AnyLeaf
- Measurements: Oxidation Reduction Potential
- Interfaces: I^2C
- Libraries: anyleaf
- Dependencies: libjpeg-dev, zlib1g-dev, Pillow, scipy, pyusb, Adafruit-extended-bus, anyleaf
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Calibrate: Voltage (Internal)

- Type: Decimal
- Default Value: 0.4
- Description: Calibration data: internal voltage

Calibrate: ORP (Internal)

- Type: Decimal
- Default Value: 400.0
- Description: Calibration data: internal ORP

COMMANDS

Calibrate: Buffer ORP (mV)

- Type: Decimal
- Default Value: 400.0
- Description: This is the nominal ORP of the calibration buffer in mV, usually labelled on the bottle.

Calibrate

- Type: Button

Clear Calibration Slots

- Type: Button

AnyLeaf: AnyLeaf pH

- Manufacturer: AnyLeaf
- Measurements: Ion concentration
- Interfaces: I^2C

- Libraries: anyleaf
- Dependencies: libjpeg-dev, zlib1g-dev, Pillow, scipy, pyusb, Adafruit-extended-bus, anyleaf
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Temperature Compensation: Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Select a measurement for temperature compensation

Temperature Compensation: Max Age

- Type: Integer
- Default Value: 120
- Description: The maximum age (seconds) of the measurement to use

Cal data: V1 (internal)

- Type: Decimal
- Description: Calibration data: Voltage

Cal data: pH1 (internal)

- Type: Decimal
- Default Value: 7.0
- Description: Calibration data: pH

Cal data: T1 (internal)

- Type: Decimal
- Default Value: 23.0
- Description: Calibration data: Temperature

Cal data: V2 (internal)

- Type: Decimal
- Default Value: 0.17
- Description: Calibration data: Voltage

Cal data: pH2 (internal)

- Type: Decimal

- Default Value: 4.0
- Description: Calibration data: pH

Cal data: T2 (internal)

- Type: Decimal
- Default Value: 23.0
- Description: Calibration data: Temperature

Cal data: V3 (internal)

- Type: Decimal
- Description: Calibration data: Voltage

Cal data: pH3 (internal)

- Type: Decimal
- Description: Calibration data: pH

Cal data: T3 (internal)

- Type: Decimal
- Description: Calibration data: Temperature

COMMANDS

Calibration buffer pH

- Type: Decimal
- Default Value: 7.0
- Description: This is the nominal pH of the calibration buffer, usually labelled on the bottle.

Calibrate, slot 1

- Type: Button

Calibrate, slot 2

- Type: Button

Calibrate, slot 3

- Type: Button

Clear Calibration Slots

- Type: Button

Atlas Scientific: Atlas CO2

- Manufacturer: Atlas Scientific
- Measurements: CO2
- Interfaces: I^2C, UART, FTDI
- Libraries: pylibftdi/fcntl/io/serial
- Dependencies: pylibftdi
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

FTDI Device

- Type: Text
- Description: The FTDI device connected to the input/output/etc.

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

COMMANDS

A one- or two-point calibration can be performed. After exposing the probe to a concentration of CO2 between 3,000 and 5,000 ppmv until readings stabilize, press Calibrate (High). You can place the probe in a 0 CO2 environment until readings stabilize, then press Calibrate (Zero). You can also clear the currently-saved calibration by pressing Clear Calibration, returning to the factory-set calibration. Status messages will be set to the Daemon Log, accessible from Config -> Mycodo Logs -> Daemon Log.

High Point CO2

- Type: Integer
- Default Value: 3000
- Description: The high CO2 calibration point (3000 - 5000 ppmv)

Calibrate (High)

- Type: Button

Calibrate (Zero)

- Type: Button

Clear Calibration

- Type: Button

The I2C address can be changed. Enter a new address in the 0xYY format (e.g. 0x22, 0x50), then press Set I2C Address. Remember to deactivate and change the I2C address option after setting the new address.

New I2C Address

- Type: Text
- Default Value: 0x69
- Description: The new I2C to set the device to

Set I2C Address

- Type: Button

Atlas Scientific: Atlas Color

- Manufacturer: Atlas Scientific
- Measurements: RGB, CIE, LUX, Proximity
- Interfaces: I^2C, UART, FTDI
- Libraries: pylibftdi/fcntl/io/serial
- Dependencies: pylibftdi
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

FTDI Device

- Type: Text
- Description: The FTDI device connected to the input/output/etc.

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

LED Only For Measure

- Type: Boolean
- Default Value: True
- Description: Turn the LED on only during the measurement

LED Percentage

- Type: Integer
- Default Value: 30
- Description: What percentage of power to supply to the LEDs during measurement

Gamma Correction

- Type: Decimal
- Default Value: 1.0
- Description: Gamma correction between 0.01 and 4.99 (default is 1.0)

COMMANDS

The EZO-RGB color sensor is designed to be calibrated to a white object at the maximum brightness the object will be viewed under. In order to get the best results, Atlas Scientific strongly recommends that the sensor is mounted into a fixed location. Holding the sensor in your hand during calibration will decrease performance.
1. Embed the EZO-RGB color sensor into its intended use location.
2. Set LED brightness to the desired level.
3. Place a white object in front of the target object and press the Calibration button.
4. A single color reading will be taken and the device will be fully calibrated.

Calibrate

- Type: Button

The I2C address can be changed. Enter a new address in the 0xYY format (e.g. 0x22, 0x50), then press Set I2C Address. Remember to deactivate and change the I2C address option after setting the new address.

New I2C Address

- Type: Text
- Default Value: 0x70
- Description: The new I2C to set the device to

Set I2C Address

- Type: Button

Atlas Scientific: Atlas DO

- Manufacturer: Atlas Scientific
- Measurements: Dissolved Oxygen
- Interfaces: I^2C, UART, FTDI
- Libraries: pylibftdi/fcntl/io/serial
- Dependencies: pylibftdi

- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

FTDI Device

- Type: Text
- Description: The FTDI device connected to the input/output/etc.

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Temperature Compensation: Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Select a measurement for temperature compensation

Temperature Compensation: Max Age

- Type: Integer
- Default Value: 120
- Description: The maximum age (seconds) of the measurement to use

COMMANDS

A one- or two-point calibration can be performed. After exposing the probe to air for 30 seconds until readings stabilize, press Calibrate (Air). If you require accuracy below 1.0 mg/L, you can place the probe in a 0 mg/L solution for 30 to 90 seconds until readings stabilize, then press Calibrate (0 mg/L). You can also clear the currently-saved calibration by pressing Clear Calibration. Status messages will be set to the Daemon Log, accessible from Config -> Mycodo Logs -> Daemon Log.

Calibrate (Air)

- Type: Button

Calibrate (0 mg/L)

- Type: Button

Clear Calibration

- Type: Button

The I2C address can be changed. Enter a new address in the 0xYY format (e.g. 0x22, 0x50), then press Set I2C Address. Remember to deactivate and change the I2C address option after setting the new address.

New I2C Address

- Type: Text
- Default Value: 0x66
- Description: The new I2C to set the device to

Set I2C Address

- Type: Button

Atlas Scientific: Atlas EC

- Manufacturer: Atlas Scientific
- Measurements: Electrical Conductivity
- Interfaces: I^2C, UART, FTDI
- Libraries: pylibftdi/fcntl/io/serial
- Dependencies: pylibftdi
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

FTDI Device

- Type: Text
- Description: The FTDI device connected to the input/output/etc.

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Temperature Compensation: Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Select a measurement for temperature compensation

Temperature Compensation: Max Age

- Type: Integer
- Default Value: 120
- Description: The maximum age (seconds) of the measurement to use

COMMANDS

Calibration: a one- or two-point calibration can be performed. It's a good idea to clear the calibration before calibrating. Always perform a dry calibration with the probe in the air (not in any fluid). Then perform either a one- or two-point calibration with calibrated solutions. If performing a one-point calibration, use the Single Point Calibration field and button. If performing a two-point calibration, use the Low and High Point Calibration fields and buttons. Allow a minute or two after submerging your probe in a calibration solution for the measurements to equilibrate before calibrating to that solution. The EZO EC circuit default temperature compensation is set to 25 °C. If the temperature of the calibration solution is +/- 2 °C from 25 °C, consider setting the temperature compensation first. Note that at no point should you change the temperature compensation value during calibration. Therefore, if you have previously enabled temperature compensation, allow at least one measurement to occur (to set the compensation value), then disable the temperature compensation measurement while you calibrate. Status messages will be set to the Daemon Log, accessible from Config -> Mycodo Logs -> Daemon Log.

Clear Calibration

- Type: Button

Calibrate Dry

- Type: Button

Single Point EC (µS)

- Type: Integer
- Default Value: 84
- Description: The EC (µS) of the single point calibration solution

Calibrate Single Point

- Type: Button

Low Point EC (µS)

- Type: Integer
- Default Value: 12880
- Description: The EC (µS) of the low point calibration solution

Calibrate Low Point

- Type: Button

High Point EC (µS)

- Type: Integer
- Default Value: 80000
- Description: The EC (µS) of the high point calibration solution

Calibrate High Point

- Type: Button

The I2C address can be changed. Enter a new address in the 0xYY format (e.g. 0x22, 0x50), then press Set I2C Address. Remember to deactivate and change the I2C address option after setting the new address.

New I2C Address

- Type: Text
- Default Value: 0x64
- Description: The new I2C to set the device to

Set I2C Address

- Type: Button

Atlas Scientific: Atlas Flow Meter

- Manufacturer: Atlas Scientific
- Measurements: Total Volume, Flow Rate
- Interfaces: I^2C, UART, FTDI
- Libraries: pylibftdi/fcntl/io/serial
- Dependencies: pylibftdi
- Manufacturer URL: Link
- Datasheet URL: Link

Set the Measurement Time Base to a value most appropriate for your anticipated flow (it will affect accuracy). This flow rate time base that is set and returned from the sensor will be converted to liters per minute, which is the default unit for this input module. If you desire a different rate to be stored in the database (such as liters per second or hour), then use the Convert to Unit option.

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

FTDI Device

- Type: Text
- Description: The FTDI device connected to the input/output/etc.

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Flow Meter Type

- Type: Select
- Options: [**Atlas Scientific 3/8" Flow Meter** | Atlas Scientific 1/4" Flow Meter | Atlas Scientific 1/2" Flow Meter | Atlas Scientific 3/4" Flow Meter | Non-Atlas Scientific Flow Meter] (Default in **bold**)
- Description: Set the type of flow meter used

Atlas Meter Time Base

- Type: Select
- Options: [Liters per Second | **Liters per Minute** | Liters per Hour] (Default in **bold**)
- Description: If using an Atlas Scientific flow meter, set the flow rate/time base

Internal Resistor

- Type: Select
- Options: [**Use Atlas Scientific Flow Meter** | Disable Internal Resistor | 1 K Ω Pull-Up | 1 K Ω Pull-Down | 10 K Ω Pull-Up | 10 K Ω Pull-Down | 100 K Ω Pull-Up | 100 K Ω Pull-Down] (Default in **bold**)
- Description: Set an internal resistor for the flow meter

Custom K Value(s)

- Type: Text
- Description: If using a non-Atlas Scientific flow meter, enter the meter's K value(s). For a single K value, enter '[volume per pulse],[number of pulses]'. For multiple K values (up to 16), enter '[volume at frequency],[frequency in Hz];[volume at frequency],[frequency in Hz];...'. Leave blank to disable.

K Value Time Base

- Type: Select
- Options: [**Use Atlas Scientific Flow Meter** | Liters per Second | Liters per Minute | Liters per Hour] (Default in **bold**)
- Description: If using a non-Atlas Scientific flow meter, set the flow rate/time base for the custom K values entered.

COMMANDS

The total volume can be cleared with the following button or with the Clear Total Volume Function Action.

Clear Total Volume

- Type: Button

The I2C address can be changed. Enter a new address in the 0xYY format (e.g. 0x22, 0x50), then press Set I2C Address. Remember to deactivate and change the I2C address option after setting the new address.

New I2C Address

- Type: Text
- Default Value: 0x68
- Description: The new I2C to set the device to

Set I2C Address

- Type: Button

Atlas Scientific: Atlas Humidity

- Manufacturer: Atlas Scientific
- Measurements: Humidity/Temperature
- Interfaces: I^2C, UART, FTDI
- Libraries: pylibftdi/fcntl/io/serial
- Dependencies: pylibftdi
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

FTDI Device

- Type: Text
- Description: The FTDI device connected to the input/output/etc.

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Measurements Enabled

- Type: Multi-Select

- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

LED Mode

- Type: Select
- Options: [**Always On** | Always Off | Only On During Measure] (Default in **bold**)
- Description: When to turn the LED on

COMMANDS

New I2C Address

- Type: Text
- Default Value: 0x6f
- Description: The new I2C to set the device to

Set I2C Address

- Type: Button

Atlas Scientific: Atlas ORP

- Manufacturer: Atlas Scientific
- Measurements: Oxidation Reduction Potential
- Interfaces: I^2C, UART, FTDI
- Libraries: pylibftdi/fcntl/io/serial
- Dependencies: pylibftdi
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer

- Description: The I2C bus the device is connected to

FTDI Device

- Type: Text
- Description: The FTDI device connected to the input/output/etc.

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Temperature Compensation: Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Select a measurement for temperature compensation

Temperature Compensation: Max Age

- Type: Integer
- Default Value: 120
- Description: The maximum age (seconds) of the measurement to use

COMMANDS

A one-point calibration can be performed. Enter the solution's mV, set the probe in the solution, then press Calibrate. You can also clear the currently-saved calibration by pressing Clear Calibration. Status messages will be set to the Daemon Log, accessible from Config -> Mycodo Logs -> Daemon Log.

Calibration Solution mV

- Type: Integer
- Default Value: 225
- Description: The value of the calibration solution, in mV

Calibrate

- Type: Button

Clear Calibration

- Type: Button

The I2C address can be changed. Enter a new address in the 0xYY format (e.g. 0x22, 0x50), then press Set I2C Address. Remember to deactivate and change the I2C address option after setting the new address.

New I2C Address

- Type: Text
- Default Value: 0x62
- Description: The new I2C to set the device to

Set I2C Address

- Type: Button

Atlas Scientific: Atlas PT-1000

- Manufacturer: Atlas Scientific
- Measurements: Temperature
- Interfaces: I^2C, UART, FTDI
- Libraries: pylibftdi/fcntl/io/serial
- Dependencies: pylibftdi
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

FTDI Device

- Type: Text
- Description: The FTDI device connected to the input/output/etc.

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

COMMANDS

New I2C Address

- Type: Text
- Default Value: 0x66
- Description: The new I2C to set the device to

Set I2C Address

- Type: Button

Atlas Scientific: Atlas Pressure

- Manufacturer: Atlas Scientific
- Measurements: Pressure
- Interfaces: I^2C, UART, FTDI
- Libraries: pylibftdi/fcntl/io/serial
- Dependencies: pylibftdi
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

FTDI Device

- Type: Text
- Description: The FTDI device connected to the input/output/etc.

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

LED Mode

- Type: Select
- Options: [**Always On** | Always Off | Only On During Measure] (Default in **bold**)
- Description: When to turn the LED on

COMMANDS

New I2C Address

- Type: Text
- Default Value: 0x6a
- Description: The new I2C to set the device to

Set I2C Address

- Type: Button

Atlas Scientific: Atlas pH

- Manufacturer: Atlas Scientific
- Measurements: Ion Concentration
- Interfaces: I^2C, UART, FTDI
- Libraries: pylibftdi/fcntl/io/serial
- Dependencies: pylibftdi
- Manufacturer URL: Link
- Datasheet URL: Link

Calibration Measurement is an optional setting that provides a temperature measurement (in Celsius) of the water that the pH is being measured from.

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

FTDI Device

- Type: Text
- Description: The FTDI device connected to the input/output/etc.

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Temperature Compensation: Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Select a measurement for temperature compensation

Temperature Compensation: Max Age

- Type: Integer
- Default Value: 120
- Description: The maximum age (seconds) of the measurement to use

COMMANDS

Calibration: a one-, two- or three-point calibration can be performed. It's a good idea to clear the calibration before calibrating. The first calibration must be the Mid point. The second must be the Low point. And the third must be the High point. You can perform a one-, two- or three-point calibration, but they must be performed in this order. Allow a minute or two after submerging your probe in a calibration solution for the measurements to equilibrate before calibrating to that solution. The EZO pH circuit default temperature compensation is set to 25 °C. If the temperature of the calibration solution is +/- 2 °C from 25 °C, consider setting the temperature compensation first. Note that if you have a Temperature Compensation Measurement selected from the Options, this will overwrite the manual Temperature Compensation set here, so be sure to disable this option if you would like to specify the temperature to compensate with. Status messages will be set to the Daemon Log, accessible from Config -> Mycodo Logs -> Daemon Log.

Compensation Temperature (°C)

- Type: Decimal
- Default Value: 25.0
- Description: The temperature of the calibration solutions

Set Temperature Compensation

- Type: Button

Clear Calibration

- Type: Button

Mid Point pH

- Type: Decimal

- Default Value: 7.0
- Description: The pH of the mid point calibration solution

Calibrate Mid

- Type: Button

Low Point pH

- Type: Decimal
- Default Value: 4.0
- Description: The pH of the low point calibration solution

Calibrate Low

- Type: Button

High Point pH

- Type: Decimal
- Default Value: 10.0
- Description: The pH of the high point calibration solution

Calibrate High

- Type: Button

Calibration Export/Import: Export calibration to a series of strings. These can later be imported to restore the calibration. Watch the Daemon Log for the output.

Export Calibration

- Type: Button

Calibration String

- Type: Text
- Description: The calibration string to import

Import Calibration

- Type: Button

The I2C address can be changed. Enter a new address in the 0xYY format (e.g. 0x22, 0x50), then press Set I2C Address. Remember to deactivate and change the I2C address option after setting the new address.

New I2C Address

- Type: Text
- Default Value: 0x63
- Description: The new I2C to set the device to

Set I2C Address

- Type: Button

BOSCH: BME280

- Manufacturer: BOSCH
- Measurements: Pressure/Humidity/Temperature
- Interfaces: I^2C
- Libraries: Adafruit_BME280
- Dependencies: Adafruit-GPIO, Adafruit_BME280
- Manufacturer URL: Link

- Datasheet URL: Link
- Product URLs: Link 1, Link 2

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

BOSCH: BME280

- Manufacturer: BOSCH
- Measurements: Pressure/Humidity/Temperature
- Interfaces: I^2C
- Libraries: Adafruit_CircuitPython_BME280
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-bme280
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URLs: Link 1, Link 2

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

BOSCH: BME280

- Manufacturer: BOSCH
- Measurements: Pressure/Humidity/Temperature
- Interfaces: I^2C
- Libraries: RPi.bme280
- Dependencies: RPi.bme280
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URLs: Link 1, Link 2

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

BOSCH: BME680

- Manufacturer: BOSCH
- Measurements: Temperature/Humidity/Pressure/Gas
- Interfaces: I^2C
- Libraries: Adafruit_CircuitPython_BME680
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-bme680
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URLs: Link 1, Link 2

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Humidity Oversampling

- Type: Select
- Options: [NONE | 1X | **2X** | 4X | 8X | 16X] (Default in **bold**)
- Description: A higher oversampling value means more stable readings with less noise and jitter. However each step of oversampling adds ~2 ms latency, causing a slower response time to fast transients.

Temperature Oversampling

- Type: Select
- Options: [NONE | 1X | 2X | 4X | **8X** | 16X] (Default in **bold**)
- Description: A higher oversampling value means more stable readings with less noise and jitter. However each step of oversampling adds ~2 ms latency, causing a slower response time to fast transients.

Pressure Oversampling

- Type: Select
- Options: [NONE | 1X | 2X | **4X** | 8X | 16X] (Default in **bold**)
- Description: A higher oversampling value means more stable readings with less noise and jitter. However each step of oversampling adds ~2 ms latency, causing a slower response time to fast transients.

IIR Filter Size

- Type: Select
- Options: [0 | 1 | **3** | 7 | 15 | 31 | 63 | 127] (Default in **bold**)
- Description: Optionally remove short term fluctuations from the temperature and pressure readings, increasing their resolution but reducing their bandwidth.

Temperature Offset

- Type: Decimal
- Description: The amount to offset the temperature, either negative or positive

Sea Level Pressure (ha)

- Type: Decimal
- Default Value: 1013.25
- Description: The pressure at sea level for the sensor location

BOSCH: BME680

- Manufacturer: BOSCH
- Measurements: Temperature/Humidity/Pressure/Gas
- Interfaces: I^2C
- Libraries: bme680
- Dependencies: bme680, smbus2

- Manufacturer URL: Link
- Datasheet URL: Link
- Product URLs: Link 1, Link 2

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Humidity Oversampling

- Type: Select
- Options: [NONE | 1X | **2X** | 4X | 8X | 16X] (Default in **bold**)
- Description: A higher oversampling value means more stable readings with less noise and jitter. However each step of oversampling adds ~2 ms latency, causing a slower response time to fast transients.

Temperature Oversampling

- Type: Select
- Options: [NONE | 1X | 2X | 4X | **8X** | 16X] (Default in **bold**)
- Description: A higher oversampling value means more stable readings with less noise and jitter. However each step of oversampling adds ~2 ms latency, causing a slower response time to fast transients.

Pressure Oversampling

- Type: Select
- Options: [NONE | 1X | 2X | **4X** | 8X | 16X] (Default in **bold**)

- Description: A higher oversampling value means more stable readings with less noise and jitter. However each step of oversampling adds ~2 ms latency, causing a slower response time to fast transients.

IIR Filter Size

- Type: Select
- Options: [0 | 1 | **3** | 7 | 15 | 31 | 63 | 127] (Default in **bold**)
- Description: Optionally remove short term fluctuations from the temperature and pressure readings, increasing their resolution but reducing their bandwidth.

Gas Heater Temperature (°C)

- Type: Integer
- Default Value: 320
- Description: What temperature to set

Gas Heater Duration (ms)

- Type: Integer
- Default Value: 150
- Description: How long of a duration to heat. 20-30 ms are necessary for the heater to reach the intended target temperature.

Gas Heater Profile

- Type: Select
- Description: Select one of the 10 configured heating durations/set points

Temperature Offset

- Type: Decimal
- Description: The amount to offset the temperature, either negative or positive

BOSCH: BMP180

- Manufacturer: BOSCH
- Measurements: Pressure/Temperature
- Interfaces: I^2C
- Libraries: Adafruit_BMP
- Dependencies: Adafruit-BMP, Adafruit-GPIO
- Datasheet URL: Link

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

BOSCH: BMP280

- Manufacturer: BOSCH
- Measurements: Pressure/Temperature
- Interfaces: I^2C
- Libraries: Adafruit_GPIO
- Dependencies: Adafruit-GPIO
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

BOSCH: BMP280

- Manufacturer: BOSCH
- Measurements: Pressure/Temperature
- Interfaces: I^2C
- Libraries: bmp280-python
- Dependencies: smbus2, bmp280
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

This is similar to the other BMP280 Input, except it uses a different library, whcih includes the ability to set forced mode.

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Enable Forced Mode

- Type: Boolean
- Description: Enable heater to evaporate condensation. Turn on heater x seconds every y measurements.

CO2Meter: K30

- Manufacturer: CO2Meter
- Measurements: CO2

- Interfaces: UART
- Libraries: serial
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Catnip Electronics: Chirp

- Manufacturer: Catnip Electronics
- Measurements: Light/Moisture/Temperature
- Interfaces: I^2C
- Libraries: smbus2
- Dependencies: smbus2
- Manufacturer URL: Link
- Product URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Cozir: Cozir CO2

- Manufacturer: Cozir
- Measurements: CO2/Humidity/Temperature
- Interfaces: UART
- Libraries: pierre-haessig/pycozir
- Dependencies: cozir
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Generic: Hall Flow Meter

- Manufacturer: Generic
- Measurements: Flow Rate, Total Volume
- Interfaces: GPIO
- Libraries: pigpio
- Dependencies: pigpio, pigpio

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Pulses per Liter

- Type: Decimal
- Default Value: 1.0
- Description: Enter the conversion factor for this meter (pulses to Liter).

COMMANDS

Clear Total Volume

- Type: Button

Infineon: DPS310

- Manufacturer: Infineon
- Measurements: Pressure/Temperature
- Interfaces: I^2C
- Libraries: Adafruit-CircuitPython-DPS310
- Dependencies: Adafruit-extended-bus, adafruit-circuitpython-dps310
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URLs: Link 1, Link 2, Link 3

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

MAXIM: DS1822

- Manufacturer: MAXIM
- Measurements: Temperature
- Interfaces: 1-Wire
- Libraries: w1thermsensor
- Dependencies: w1thermsensor
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

COMMANDS

Set the resolution, precision, and response time for the sensor. This setting will be written to the EEPROM to allow persistence after power loss. The EEPROM has a limited amount of writes (>50k).

Resolution

- Type: Select
- Description: Select the resolution for the sensor

Set Resolution

- Type: Button

MAXIM: DS1825

- Manufacturer: MAXIM
- Measurements: Temperature
- Interfaces: 1-Wire
- Libraries: w1thermsensor
- Dependencies: w1thermsensor
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

COMMANDS

Set the resolution, precision, and response time for the sensor. This setting will be written to the EEPROM to allow persistence after power loss. The EEPROM has a limited amount of writes (>50k).

Resolution

- Type: Select
- Description: Select the resolution for the sensor

Set Resolution

- Type: Button

MAXIM: DS18B20

- Manufacturer: MAXIM
- Measurements: Temperature
- Interfaces: 1-Wire
- Libraries: ow-shell
- Dependencies: ow-shell, owfs
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URLs: Link 1, Link 2, Link 3
- Additional URL: Link

Warning: Counterfeit DS18B20 sensors are common and can cause a host of issues. Review the Additional URL for more information about how to determine if your sensor is authentic.

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

MAXIM: DS18B20

- Manufacturer: MAXIM
- Measurements: Temperature
- Interfaces: 1-Wire
- Libraries: w1thermsensor
- Dependencies: w1thermsensor
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URLs: Link 1, Link 2, Link 3
- Additional URL: Link

Warning: Counterfeit DS18B20 sensors are common and can cause a host of issues. Review the Additional URL for more information about how to determine if your sensor is authentic.

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

COMMANDS

Set the resolution, precision, and response time for the sensor. This setting will be written to the EEPROM to allow persistence after power loss. The EEPROM has a limited amount of writes (>50k).

Resolution

- Type: Select
- Description: Select the resolution for the sensor

Set Resolution

- Type: Button

MAXIM: DS18S20

- Manufacturer: MAXIM
- Measurements: Temperature
- Interfaces: 1-Wire
- Libraries: w1thermsensor
- Dependencies: w1thermsensor
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

COMMANDS

Set the resolution, precision, and response time for the sensor. This setting will be written to the EEPROM to allow persistence after power loss. The EEPROM has a limited amount of writes (>50k).

Resolution

- Type: Select
- Description: Select the resolution for the sensor

Set Resolution

- Type: Button

MAXIM: DS28EA00

- Manufacturer: MAXIM
- Measurements: Temperature
- Interfaces: 1-Wire
- Libraries: w1thermsensor
- Dependencies: w1thermsensor
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

COMMANDS

Set the resolution, precision, and response time for the sensor. This setting will be written to the EEPROM to allow persistence after power loss. The EEPROM has a limited amount of writes (>50k).

Resolution

- Type: Select
- Description: Select the resolution for the sensor

Set Resolution

- Type: Button

MAXIM: MAX31850K

- Manufacturer: MAXIM
- Measurements: Temperature
- Interfaces: 1-Wire
- Libraries: w1thermsensor
- Dependencies: w1thermsensor
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

COMMANDS

Set the resolution, precision, and response time for the sensor. This setting will be written to the EEPROM to allow persistence after power loss. The EEPROM has a limited amount of writes (>50k).

Resolution

- Type: Select
- Description: Select the resolution for the sensor

Set Resolution

- Type: Button

MAXIM: MAX31855

- Manufacturer: MAXIM
- Measurements: Temperature (Object/Die)
- Interfaces: UART
- Libraries: Adafruit_MAX31855
- Dependencies: Adafruit_MAX31855, Adafruit-GPIO
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

CS Pin

- Type: Integer
- Description: The GPIO (using BCM numbering) connected to the Cable Select pin

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

MAXIM: MAX31856

- Manufacturer: MAXIM
- Measurements: Temperature (Object/Die)
- Interfaces: UART
- Libraries: RPi.GPIO
- Dependencies: RPi.GPIO
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

CS Pin

- Type: Integer
- Description: The GPIO (using BCM numbering) connected to the Cable Select pin

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

MAXIM: MAX31865

- Manufacturer: MAXIM
- Measurements: Temperature
- Interfaces: SPI
- Libraries: Adafruit-CircuitPython-MAX31865
- Dependencies: adafruit-circuitpython-max31865
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

This module was added to allow support for multiple sensors to be connected at the same time, which the original MAX31865 module was not designed for.

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Chip Select Pin

- Type: Integer
- Default Value: 8
- Description: Enter the GPIO Chip Select Pin for your device.

Number of wires

- Type: Select
- Options: [**2 Wires** | 3 Wires | 4 Wires] (Default in **bold**)
- Description: Select the number of wires your thermocouple has.

MAXIM: MAX31865

- Manufacturer: MAXIM
- Measurements: Temperature
- Interfaces: UART
- Libraries: RPi.GPIO
- Dependencies: RPi.GPIO
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

Note: This module does not allow for multiple sensors to be connected at the same time. For multi-sensor support, use the MAX31865 CircuitPython Input.

OPTIONS

CS Pin

- Type: Integer
- Description: The GPIO (using BCM numbering) connected to the Cable Select pin

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

MQTT: MQTT Subscribe (JSON payload)

- Manufacturer: MQTT
- Measurements: Variable measurements
- Interfaces: Mycodo
- Libraries: paho-mqtt, jmespath
- Dependencies: paho-mqtt, jmespath

A single topic is subscribed to and the returned JSON payload contains one or more key/value pairs. The given JSON Key is used as a JMESPATH expression to find the corresponding value that will be stored for that channel. Be sure you select and save the Measurement Unit for each channel. Once the unit has been saved, you can convert to other units in the Convert Measurement section. Example expressions for jmespath (https://jmespath.org) include temperature, sensors[0].temperature, and bathroom.temperature which refer to the temperature as a direct key within the first entry of sensors or as a subkey of bathroom, respectively. Jmespath elements and keys that contain special characters have to be enclosed in double quotes, e.g. "sensor-1".temperature. Warning: If using multiple MQTT Inputs or Functions, ensure the Client IDs are unique.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Host

- Type: Text
- Default Value: localhost
- Description: Host address or IP

Port

- Type: Integer
- Default Value: 1883
- Description: Host port number

Topic

- Type: Text
- Default Value: mqtt/test/input
- Description: The topic to subscribe to

Keep Alive

- Type: Integer
- Default Value: 60
- Description: Maximum amount of time between received signals. Set to 0 to disable.

Client ID

- Type: Text
- Default Value: client_bZCtOuMT
- Description: Unique client ID for connecting to the server

Use Login

- Type: Boolean
- Description: Send login credentials

Use TLS

- Type: Boolean
- Description: Send login credentials using TLS

Username

- Type: Text
- Default Value: user
- Description: Username for connecting to the server

Password

- Type: Text
- Description: Password for connecting to the server. Leave blank to disable.

CHANNEL OPTIONS

Name

- Type: Text
- Description: A name to distinguish this from others

JSON Key

- Type: Text
- Description: JMES Path expression to find value in JSON response

MQTT: MQTT Subscribe (Value payload)

- Manufacturer: MQTT
- Measurements: Variable measurements
- Interfaces: Mycodo
- Libraries: paho-mqtt
- Dependencies: paho-mqtt

A topic is subscribed to for each channel Subscription Topic and the returned payload value will be stored for that channel. Be sure you select and save the Measurement Unit for each of the channels. Once the unit has been saved, you can convert to other units in the Convert Measurement section. Warning: If using multiple MQTT Inputs or Functions, ensure the Client IDs are unique.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Host

- Type: Text
- Default Value: localhost
- Description: Host address or IP

Port

- Type: Integer
- Default Value: 1883
- Description: Host port number

Keep Alive

- Type: Integer
- Default Value: 60
- Description: Maximum amount of time between received signals. Set to 0 to disable.

Client ID

- Type: Text
- Default Value: client_JBFze0AI
- Description: Unique client ID for connecting to the server

Use Login

- Type: Boolean
- Description: Send login credentials

Use TLS

- Type: Boolean
- Description: Send login credentials using TLS

Username

- Type: Text
- Default Value: user
- Description: Username for connecting to the server

Password

- Type: Text
- Description: Password for connecting to the server. Leave blank to disable.

CHANNEL OPTIONS

Name

- Type: Text
- Description: A name to distinguish this from others

Subscription Topic

- Type: Text
- Description: The MQTT topic to subscribe to

Melexis: MLX90393

- Manufacturer: Melexis
- Measurements: Magnetic Flux
- Interfaces: I^2C
- Libraries: Adafruit-CircuitPython-MLX90393
- Dependencies: Adafruit-extended-bus, adafruit-circuitpython-mlx90393
- Manufacturer URL: Link

- Datasheet URL: Link
- Product URLs: Link 1, Link 2, Link 3

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Melexis: MLX90614

- Manufacturer: Melexis
- Measurements: Temperature (Ambient/Object)
- Interfaces: I^2C
- Libraries: smbus2
- Dependencies: smbus2
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Microchip: MCP3008

- Manufacturer: Microchip
- Measurements: Voltage (Analog-to-Digital Converter)
- Interfaces: UART
- Libraries: Adafruit_MCP3008
- Dependencies: Adafruit-MCP3008
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

CS Pin

- Type: Integer
- Description: The GPIO (using BCM numbering) connected to the Cable Select pin

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

VREF (volts)

- Type: Decimal
- Default Value: 3.3
- Description: Set the VREF voltage

Microchip: MCP342x (x=2,3,4,6,7,8)

- Manufacturer: Microchip
- Measurements: Voltage (Analog-to-Digital Converter)
- Interfaces: I^2C
- Libraries: MCP342x
- Dependencies: smbus2, MCP342x
- Manufacturer URLs: Link 1, Link 2, Link 3, Link 4, Link 5
- Datasheet URLs: Link 1, Link 2

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Microchip: MCP9808

- Manufacturer: Microchip
- Measurements: Temperature
- Interfaces: I^2C
- Libraries: Adafruit_MCP9808
- Dependencies: Adafruit-GPIO, Adafruit_MCP9808
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Multiple Manufacturers: HC-SR04

- Manufacturer: Multiple Manufacturers
- Measurements: Ultrasonic Distance
- Interfaces: GPIO
- Libraries: Adafruit-CircuitPython-HCSR04
- Dependencies: libgpiod-dev, pyusb, adafruit-circuitpython-hcsr04
- Manufacturer URL: Link

- Datasheet URL: Link
- Product URL: Link
- Additional URL: Link

OPTIONS

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Trigger Pin

- Type: Integer
- Description: Enter the GPIO Trigger Pin for your device (BCM numbering).

Echo Pin

- Type: Integer
- Description: Enter the GPIO Echo Pin for your device (BCM numbering).

Panasonic: AMG8833

- Manufacturer: Panasonic
- Measurements: 8x8 Temperature Grid
- Interfaces: I^2C
- Libraries: Adafruit_AMG88xx/Pillow/colour
- Dependencies: libjpeg-dev, zlib1g-dev, colour, Pillow, Adafruit_AMG88xx

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

ROHM: BH1750

- Manufacturer: ROHM
- Measurements: Light
- Interfaces: I^2C
- Libraries: smbus2
- Dependencies: smbus2
- Datasheet URL: Link
- Product URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Raspberry Pi Foundation: Sense HAT

- Manufacturer: Raspberry Pi Foundation
- Measurements: hum/temp/press/compass/magnet/accel/gyro
- Interfaces: I^2C
- Libraries: sense-hat
- Dependencies: sense-hat
- Manufacturer URL: Link

This module acquires measurements from the Raspberry Pi Sense HAT sensors, which include the LPS25H, LSM9DS1, and HTS221.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Ruuvi: RuuviTag

- Manufacturer: Ruuvi
- Measurements: Acceleration/Humidity/Pressure/Temperature
- Interfaces: BT
- Libraries: ruuvitag_sensor
- Dependencies: psutil, bluez, bluez-hcidump, ruuvitag-sensor
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

MAC (XX:XX:XX:XX:XX:XX)

- Type: Text
- Description: The MAC address of the Bluetooth device

BT Adapter (hci[X])

- Type: Integer
- Description: The adapter of the Bluetooth device

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

STMicroelectronics: VL53L0X

- Manufacturer: STMicroelectronics
- Measurements: Millimeter (Time-of-Flight Distance)
- Interfaces: I^2C
- Libraries: VL53L0X_rasp_python
- Dependencies: VL53L0X
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URLs: Link 1, Link 2

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Accuracy

- Type: Select
- Options: [**Good Accuracy (33 ms, 1.2 m range)** | Better Accuracy (66 ms, 1.2 m range) | Best Accuracy (200 ms, 1.2 m range) | Long Range (33 ms, 2 m) | High Speed, Low Accuracy (20 ms, 1.2 m)] (Default in **bold**)
- Description: Set the accuracy. A longer measurement duration yields a more accurate measurement

COMMANDS

New I2C Address

- Type: Text
- Default Value: 0x52
- Description: The new I2C to set the device to

Set I2C Address

- Type: Button

STMicroelectronics: VL53L1X

- Manufacturer: STMicroelectronics
- Measurements: Millimeter (Time-of-Flight Distance)
- Interfaces: I^2C
- Libraries: VL53L1X
- Dependencies: smbus2, vl53l1x
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URLs: Link 1, Link 2

Notes when setting a custom timing budget: A higher timing budget results in greater measurement accuracy, but also a higher power consumption. The inter measurement period must be >= the timing budget, otherwise it will be double the expected value.

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Range

- Type: Select
- Options: [**Short Range** | Medium Range | Long Range | Custom Timing Budget] (Default in **bold**)
- Description: Select a range or select to set a custom Timing Budget and Inter Measurement Period.

Timing Budget (microseconds)

- Type: Integer
- Default Value: 66000
- Description: Set the timing budget. Must be less than or equal to the Inter Measurement Period.

Inter Measurement Period (milliseconds)

- Type: Integer
- Default Value: 70
- Description: Set the Inter Measurement Period

Seeedstudio: DHT11/22

- Manufacturer: Seeedstudio
- Measurements: Humidity/Temperature
- Interfaces: GROVE

- Libraries: grovepi
- Dependencies: libatlas-base-dev, grovepi
- Manufacturer URLs: Link 1, Link 2

Enter the Grove Pi+ GPIO pin connected to the sensor and select the sensor type.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Sensor Type

- Type: Select
- Options: [**DHT11 (Blue)** | DHT22 (White)] (Default in **bold**)
- Description: Sensor type

Sensirion: SCD-4x (SCD-40, SCD-41)

- Manufacturer: Sensirion
- Measurements: CO2/Temperature/Humidity
- Interfaces: I^2C
- Libraries: Adafruit-CircuitPython-SCD4x
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-scd4x
- Manufacturer URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Temperature Offset

- Type: Decimal
- Default Value: 4.0
- Description: Set the sensor temperature offset

Altitude (m)

- Type: Integer
- Description: Set the sensor altitude (meters)

Automatic Self-Calibration

- Type: Boolean
- Description: Set the sensor automatic self-calibration

Persist Settings

- Type: Boolean
- Default Value: True
- Description: Settings will persist after powering off

COMMANDS

You can force the CO2 calibration for a specific CO2 concentration value (in ppmv).

CO2 Concentration (ppmv)

- Type: Decimal
- Default Value: 400.0
- Description: Calibrate to this CO2 concentration that the sensor is being exposed to (in ppmv)

Calibrate CO2

- Type: Button

Sensirion: SCD30

- Manufacturer: Sensirion
- Measurements: CO2/Humidity/Temperature
- Interfaces: I^2C
- Libraries: Adafruit-CircuitPython-SCD30
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitPython-scd30
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URLs: Link 1, Link 2

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

I2C Frequency: The SCD-30 has temperamental I2C with clock stretching. The datasheet recommends starting at 50,000 Hz.

I2C Frequency (Hz)

- Type: Integer
- Default Value: 50000

Automatic Self Ccalibration (ASC): To work correctly, the sensor must be on and active for 7 days after enabling ASC, and exposed to fresh air for at least 1 hour per day. Consult the manufacturer's documentation for more information.

Enable Automatic Self Calibration

- Type: Boolean

Temperature Offset: Specifies the offset to be added to the reported measurements to account for a bias in the measured signal. Value is in degrees Celsius with a resolution of 0.01 degrees and a maximum value of 655.35 C.

Temperature Offset

- Type: Decimal

Ambient Air Pressure (mBar): Specify the ambient air pressure at the measurement location in mBar. Setting this value adjusts the CO2 measurement calculations to account for the air pressure's effect on readings. Values must be in mBar, from 700 to 1200 mBar.

Ambient Air Pressure (mBar)

- Type: Integer
- Default Value: 1200

Altitude: Specifies the altitude at the measurement location in meters above sea level. Setting this value adjusts the CO2 measurement calculations to account for the air pressure's effect on readings.

Altitude (m)

- Type: Integer
- Default Value: 100

COMMANDS

A soft reset restores factory default values.

Soft Reset

- Type: Button

Forced Re-Calibration: The SCD-30 is placed in an environment with a known CO2 concentration, this concentration value is entered in the CO2 Concentration (ppmv) field, then the Foce Calibration button is pressed. But how do you come up with that known value? That is a caveat of this approach and Sensirion suggests three approaches: 1. Using a separate secondary calibrated CO2 sensor to provide the value. 2. Exposing the SCD-30 to a controlled environment with a known value. 3. Exposing the SCD-30 to fresh outside air and using a value of 400 ppm.

CO2 Concentration (ppmv)

- Type: Integer
- Default Value: 800
- Description: The CO2 concentration of the sensor environment when forcing calibration

Force Recalibration

- Type: Button

Sensirion: SCD30

- Manufacturer: Sensirion
- Measurements: CO2/Humidity/Temperature
- Interfaces: I^2C
- Libraries: scd30_i2c
- Dependencies: scd30-i2c
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URLs: Link 1, Link 2

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal

- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Automatic Self Ccalibration (ASC): To work correctly, the sensor must be on and active for 7 days after enabling ASC, and exposed to fresh air for at least 1 hour per day. Consult the manufacturer's documentation for more information.

Enable Automatic Self Calibration

- Type: Boolean

COMMANDS

A soft reset restores factory default values.

Soft Reset

- Type: Button

Sensirion: SHT1x/7x

- Manufacturer: Sensirion
- Measurements: Humidity/Temperature
- Interfaces: GPIO
- Libraries: sht_sensor
- Dependencies: sht-sensor
- Manufacturer URLs: Link 1, Link 2

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal

- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Sensirion: SHT2x

- Manufacturer: Sensirion
- Measurements: Humidity/Temperature
- Interfaces: I^2C
- Libraries: sht20
- Dependencies: sht20
- Manufacturer URL: Link

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Temperature Resolution

- Type: Select
- Options: [11-bit | 12-bit | 13-bit | **14-bit**] (Default in **bold**)
- Description: The resolution of the temperature measurement

Sensirion: SHT2x

- Manufacturer: Sensirion
- Measurements: Humidity/Temperature
- Interfaces: I^2C
- Libraries: smbus2
- Dependencies: smbus2

- Manufacturer URL: Link

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Sensirion: SHT31-D

- Manufacturer: Sensirion
- Measurements: Humidity/Temperature
- Interfaces: I^2C
- Libraries: Adafruit_CircuitPython_SHT31
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-sht31d
- Manufacturer URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Sensirion: SHT3x (30, 31, 35)

- Manufacturer: Sensirion
- Measurements: Humidity/Temperature
- Interfaces: I^2C
- Libraries: Adafruit_SHT31
- Dependencies: Adafruit-GPIO, Adafruit-SHT31
- Manufacturer URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean

- Description: Check to turn the output off after (opposed to before) the measurement is complete

Enable Heater

- Type: Boolean
- Description: Enable heater to evaporate condensation. Turn on heater x seconds every y measurements.

Heater On Seconds

- Type: Decimal
- Default Value: 1.0
- Description: How long to turn the heater on (seconds).

Heater On Period

- Type: Integer
- Default Value: 10
- Description: After how many measurements to turn the heater on. This will repeat.

Sensirion: SHT4X

- Manufacturer: Sensirion
- Measurements: Humidity/Temperature
- Interfaces: I^2C
- Libraries: Adafruit_CircuitPython_SHT4X
- Dependencies: pyusb, Adafruit-extended-bus, adafruit_circuitpython_sht4x
- Manufacturer URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Sensirion: SHTC3

- Manufacturer: Sensirion
- Measurements: Humidity/Temperature
- Interfaces: I^2C
- Libraries: Adafruit_CircuitPython_SHT3C
- Dependencies: pyusb, Adafruit-extended-bus, adafruit_circuitpython_shtc3
- Manufacturer URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Sensorion: SHT31 Smart Gadget

- Manufacturer: Sensorion
- Measurements: Humidity/Temperature
- Interfaces: BT
- Libraries: bluepy

- Dependencies: pi-bluetooth, libglib2.0-dev, bluepy
- Manufacturer URL: Link

OPTIONS

MAC (XX:XX:XX:XX:XX:XX)

- Type: Text
- Description: The MAC address of the Bluetooth device

BT Adapter (hci[X])

- Type: Integer
- Description: The adapter of the Bluetooth device

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Download Stored Data

- Type: Boolean
- Default Value: True
- Description: Download the data logged to the device.

Set Logging Interval

- Type: Integer
- Default Value: 600
- Description: Set the logging interval (seconds) the device will store measurements on its internal memory.

Silicon Labs: SI1145

- Manufacturer: Silicon Labs
- Measurements: Light (UV/Visible/IR), Proximity (cm)
- Interfaces: I^2C
- Libraries: si1145
- Dependencies: SI1145

- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Silicon Labs: Si7021

- Manufacturer: Silicon Labs
- Measurements: Temperature/Humidity
- Interfaces: I^2C
- Libraries: Adafruit-CircuitPython-Si7021
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-si7021
- Datasheet URL: Link

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer

- Description: The I2C bus the device is connected to

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Sonoff: TH16/10 (Tasmota firmware) with AM2301/Si7021

- Manufacturer: Sonoff
- Measurements: Humidity/Temperature
- Libraries: requests
- Dependencies: requests
- Manufacturer URL: Link

This Input module allows the use of any temperature/huidity sensor with the TH10/TH16. Changing the Sensor Name option changes the key that's queried from the returned dictionary of measurements. If you would like to use this module with a version of this device that uses the AM2301, change Sensor Name to AM2301.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean

- Description: Check to turn the output off after (opposed to before) the measurement is complete

IP Address

- Type: Text
- Default Value: 192.168.0.100
- Description: The IP address of the device

Sensor Name

- Type: Text
- Default Value: SI7021
- Description: The name of the sensor connected to the device (specific key name in the returned dictionary)

Sonoff: TH16/10 (Tasmota firmware) with AM2301

- Manufacturer: Sonoff
- Measurements: Humidity/Temperature
- Libraries: requests
- Dependencies: requests
- Manufacturer URL: Link

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

IP Address

- Type: Text
- Default Value: 192.168.0.100
- Description: The IP address of the device

Sonoff: TH16/10 (Tasmota firmware) with DS18B20

- Manufacturer: Sonoff

- Measurements: Temperature
- Libraries: requests
- Dependencies: requests
- Manufacturer URL: Link

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

IP Address

- Type: Text
- Default Value: 192.168.0.100
- Description: The IP address of the device

TE Connectivity: HTU21D

- Manufacturer: TE Connectivity
- Measurements: Humidity/Temperature
- Interfaces: I^2C
- Libraries: Adafruit-CircuitPython-HTU21D
- Dependencies: Adafruit-extended-bus, adafruit-circuitpython-HTU21D
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

TE Connectivity: HTU21D

- Manufacturer: TE Connectivity
- Measurements: Humidity/Temperature
- Interfaces: I²C
- Libraries: pigpio
- Dependencies: pigpio, pigpio
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Tasmota: Tasmota Outlet Energy Monitor (HTTP)

- Manufacturer: Tasmota
- Measurements: Total Energy, Amps, Watts
- Interfaces: HTTP
- Libraries: requests
- Manufacturer URL: Link
- Product URL: Link

This input queries the energy usage information from a WiFi outlet that is running the tasmota firmware. There are many WiFi outlets that support tasmota, and many of of those have energy monitoring capabilities. When used with an MQTT Output, you can both control your tasmota outlets as well as mionitor their energy usage.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Host

- Type: Text
- Default Value: 192.168.0.50
- Description: Host address or IP

Texas Instruments: ADS1015

- Manufacturer: Texas Instruments
- Measurements: Voltage (Analog-to-Digital Converter)
- Interfaces: I^2C
- Libraries: Adafruit_CircuitPython
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-ads1x15

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Measurements to Average

- Type: Integer
- Default Value: 5
- Description: The number of times to measure each channel. An average of the measurements will be stored.

Texas Instruments: ADS1115: Generic Analog pH/EC

- Manufacturer: Texas Instruments
- Measurements: Ion Concentration/Electrical Conductivity
- Interfaces: I^2C
- Libraries: Adafruit_CircuitPython_ADS1x15
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-ads1x15

This input relies on an ADS1115 analog-to-digital converter (ADC) to measure pH and/or electrical conductivity (EC) from analog sensors. You can enable or disable either measurement if you want to only connect a pH sensor or an EC sensor by selecting which measurements you want to under Measurements Enabled. Select which channel each sensor is connected to on the ADC. There are default calibration values initially set for the Input. There are also functions to allow you to easily calibrate your sensors with calibration solutions. If you use the Calibrate Slot actions, these values will be calculated and will replace the currently-set values. You can use the Clear Calibration action to delete the database values and return to using the default values. If you delete the Input or create a new Input to use your ADC/sensors with, you will need to recalibrate in order to store new calibration data.

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

ADC Channel: pH

- Type: Select
- Options: [**Channel 0** | Channel 1 | Channel 2 | Channel 3] (Default in **bold**)
- Description: The ADC channel the pH sensor is connected

ADC Channel: EC

- Type: Select
- Options: [Channel 0 | **Channel 1** | Channel 2 | Channel 3] (Default in **bold**)
- Description: The ADC channel the EC sensor is connected

Temperature Compensation

Temperature Compensation: Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Select a measurement for temperature compensation

Temperature Compensation: Max Age

- Type: Integer
- Default Value: 120
- Description: The maximum age (seconds) of the measurement to use

pH Calibration Data

Cal data: V1 (internal)

- Type: Decimal
- Default Value: 1.5
- Description: Calibration data: Voltage

Cal data: pH1 (internal)

- Type: Decimal
- Default Value: 7.0
- Description: Calibration data: pH

Cal data: T1 (internal)

- Type: Decimal
- Default Value: 25.0
- Description: Calibration data: Temperature

Cal data: V2 (internal)

- Type: Decimal
- Default Value: 2.032
- Description: Calibration data: Voltage

Cal data: pH2 (internal)

- Type: Decimal
- Default Value: 4.0
- Description: Calibration data: pH

Cal data: T2 (internal)

- Type: Decimal
- Default Value: 25.0
- Description: Calibration data: Temperature

EC Calibration Data

EC cal data: V1 (internal)

- Type: Decimal
- Default Value: 0.232
- Description: EC calibration data: Voltage

EC cal data: EC1 (internal)

- Type: Decimal
- Default Value: 1413.0
- Description: EC calibration data: EC

EC cal data: T1 (internal)

- Type: Decimal
- Default Value: 25.0
- Description: EC calibration data: EC

EC cal data: V2 (internal)

- Type: Decimal
- Default Value: 2.112
- Description: EC calibration data: Voltage

EC cal data: EC2 (internal)

- Type: Decimal
- Default Value: 12880.0
- Description: EC calibration data: EC

EC cal data: T2 (internal)

- Type: Decimal
- Default Value: 25.0
- Description: EC calibration data: EC

COMMANDS

pH Calibration Actions: Place your probe in a solution of known pH.

```
Set the known pH value in the "Calibration buffer pH" field, and press "Calibrate pH, slot 1".
Repeat with a second buffer, and press "Calibrate pH, slot 2".
You don't need to change the values under "Custom Options".
```

Calibration buffer pH

- Type: Decimal
- Default Value: 7.0
- Description: This is the nominal pH of the calibration buffer, usually labelled on the bottle.

Calibrate pH, slot 1

- Type: Button

Calibrate pH, slot 2

- Type: Button

Clear pH Calibration Slots

- Type: Button

EC Calibration Actions: Place your probe in a solution of known EC.

```
Set the known EC value in the "Calibration standard EC" field, and press "Calibrate EC, slot 1".
Repeat with a second standard, and press "Calibrate EC, slot 2".
You don't need to change the values under "Custom Options".
```

Calibration standard EC

- Type: Decimal
- Default Value: 1413.0
- Description: This is the nominal EC of the calibration standard, usually labelled on the bottle.

Calibrate EC, slot 1

- Type: Button

Calibrate EC, slot 2

- Type: Button

Clear EC Calibration Slots

- Type: Button

Texas Instruments: ADS1115

- Manufacturer: Texas Instruments
- Measurements: Voltage (Analog-to-Digital Converter)
- Interfaces: I^2C
- Libraries: Adafruit_CircuitPython_ADS1x15
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-ads1x15

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal

- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Measurements to Average

- Type: Integer
- Default Value: 5
- Description: The number of times to measure each channel. An average of the measurements will be stored.

Texas Instruments: ADS1256: Generic Analog pH/EC

- Manufacturer: Texas Instruments
- Measurements: Ion Concentration/Electrical Conductivity
- Interfaces: UART
- Libraries: wiringpi, kizniche/PiPyADC-py3
- Dependencies: wiringpi, pipyadc_py3

This input relies on an ADS1256 analog-to-digital converter (ADC) to measure pH and/or electrical conductivity (EC) from analog sensors. You can enable or disable either measurement if you want to only connect a pH sensor or an EC sensor by selecting which measurements you want to under Measurements Enabled. Select which channel each sensor is connected to on the ADC. There are default calibration values initially set for the Input. There are also functions to allow you to easily calibrate your sensors with calibration solutions. If you use the Calibrate Slot actions, these values will be calculated and will replace the currently-set values. You can use the Clear Calibration action to delete the database values and return to using the default values. If you delete the Input or create a new Input to use your ADC/sensors with, you will need to recalibrate in order to store new calibration data.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

ADC Channel: pH

- Type: Select
- Options: [Not Connected | **Channel 0** | Channel 1 | Channel 2 | Channel 3 | Channel 4 | Channel 5 | Channel 6 | Channel 7] (Default in **bold**)
- Description: The ADC channel the pH sensor is connected

ADC Channel: EC

- Type: Select
- Options: [Not Connected | Channel 0 | **Channel 1** | Channel 2 | Channel 3 | Channel 4 | Channel 5 | Channel 6 | Channel 7] (Default in **bold**)
- Description: The ADC channel the EC sensor is connected

Temperature Compensation

Temperature Compensation: Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Select a measurement for temperature compensation

Temperature Compensation: Max Age

- Type: Integer
- Default Value: 120
- Description: The maximum age (seconds) of the measurement to use

pH Calibration Data

Cal data: V1 (internal)

- Type: Decimal
- Default Value: 1.5
- Description: Calibration data: Voltage

Cal data: pH1 (internal)

- Type: Decimal
- Default Value: 7.0
- Description: Calibration data: pH

Cal data: T1 (internal)

- Type: Decimal
- Default Value: 25.0
- Description: Calibration data: Temperature

Cal data: V2 (internal)

- Type: Decimal
- Default Value: 2.032
- Description: Calibration data: Voltage

Cal data: pH2 (internal)

- Type: Decimal
- Default Value: 4.0
- Description: Calibration data: pH

Cal data: T2 (internal)

- Type: Decimal
- Default Value: 25.0
- Description: Calibration data: Temperature

EC Calibration Data

EC cal data: V1 (internal)

- Type: Decimal
- Default Value: 0.232
- Description: EC calibration data: Voltage

EC cal data: EC1 (internal)

- Type: Decimal
- Default Value: 1413.0
- Description: EC calibration data: EC

EC cal data: T1 (internal)

- Type: Decimal
- Default Value: 25.0
- Description: EC calibration data: EC

EC cal data: V2 (internal)

- Type: Decimal
- Default Value: 2.112
- Description: EC calibration data: Voltage

EC cal data: EC2 (internal)

- Type: Decimal
- Default Value: 12880.0
- Description: EC calibration data: EC

EC cal data: T2 (internal)

- Type: Decimal
- Default Value: 25.0
- Description: EC calibration data: EC

Calibration

- Type: Select
- Description: Set the calibration method to perform during Input activation

COMMANDS

pH Calibration Actions: Place your probe in a solution of known pH.

```
Set the known pH value in the `Calibration buffer pH` field, and press `Calibrate pH, slot 1`.
Repeat with a second buffer, and press `Calibrate pH, slot 2`.
You don't need to change the values under `Custom Options`.
```

Calibration buffer pH

- Type: Decimal
- Default Value: 7.0

- Description: This is the nominal pH of the calibration buffer, usually labelled on the bottle.

Calibrate pH, slot 1

- Type: Button

Calibrate pH, slot 2

- Type: Button

Clear pH Calibration Slots

- Type: Button

EC Calibration Actions: Place your probe in a solution of known EC.

```
Set the known EC value in the `Calibration standard EC` field, and press `Calibrate EC, slot 1`.
Repeat with a second standard, and press `Calibrate EC, slot 2`.
You don't need to change the values under `Custom Options`.
```

Calibration standard EC

- Type: Decimal
- Default Value: 1413.0
- Description: This is the nominal EC of the calibration standard, usually labelled on the bottle.

Calibrate EC, slot 1

- Type: Button

Calibrate EC, slot 2

- Type: Button

Clear EC Calibration Slots

- Type: Button

Texas Instruments: ADS1256

- Manufacturer: Texas Instruments
- Measurements: Voltage (Waveshare, Analog-to-Digital Converter)
- Interfaces: UART
- Libraries: wiringpi, kizniche/PiPyADC-py3
- Dependencies: wiringpi, pipyadc_py3

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal

- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Calibration

- Type: Select
- Description: Set the calibration method to perform during Input activation

Texas Instruments: ADS1x15

- Manufacturer: Texas Instruments
- Measurements: Voltage (Analog-to-Digital Converter)
- Interfaces: I^2C
- Libraries: Adafruit_ADS1x15 [DEPRECATED]
- Dependencies: Adafruit-GPIO, Adafruit-ADS1x15

The Adafruit_ADS1x15 is deprecated. It's advised to use The Circuit Python ADS1x15 Input.

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Measurements to Average

- Type: Integer
- Default Value: 5
- Description: The number of times to measure each channel. An average of the measurements will be stored.

Texas Instruments: HDC1000

- Manufacturer: Texas Instruments
- Measurements: Humidity/Temperature
- Interfaces: I^2C
- Libraries: fcntl/io
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Texas Instruments: INA219x

- Manufacturer: Texas Instruments
- Measurements: Electrical Current (DC)
- Interfaces: I^2C

- Libraries: Adafruit_CircuitPython
- Dependencies: adafruit-circuitpython-ina219, Adafruit-extended-bus
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Measurements to Average

- Type: Integer
- Default Value: 5
- Description: The number of times to measure each channel. An average of the measurements will be stored.

Calibration Range

- Type: Select
- Options: [**32V @ 2A max (default)** | 32V @ 1A max | 16V @ 400mA max | 16V @ 5A max] (Default in **bold**)
- Description: Set the device calibration range

Bus Voltage Range

- Type: Select
- Options: [(0x00) - 16V | **(0x01) - 32V (default)**] (Default in **bold**)
- Description: Set the bus voltage range

Bus ADC Resolution

- Type: Select
- Options: [(0x00) - 9 Bit / 1 Sample | (0x01) - 10 Bit / 1 Sample | (0x02) - 11 Bit / 1 Sample | **(0x03) - 12 Bit / 1 Sample (default)** | (0x09) - 12 Bit / 2 Samples | (0x0A) - 12 Bit / 4 Samples | (0x0B) - 12 Bit / 8 Samples | (0x0C) - 12 Bit / 16 Samples | (0x0D) - 12 Bit / 32 Samples | (0x0E) - 12 Bit / 64 Samples | (0x0F) - 12 Bit / 128 Samples] (Default in **bold**)
- Description: Set the Bus ADC Resolution.

Shunt ADC Resolution

- Type: Select
- Options: [(0x00) - 9 Bit / 1 Sample | (0x01) - 10 Bit / 1 Sample | (0x02) - 11 Bit / 1 Sample | **(0x03) - 12 Bit / 1 Sample (default)** | (0x09) - 12 Bit / 2 Samples | (0x0A) - 12 Bit / 4 Samples | (0x0B) - 12 Bit / 8 Samples | (0x0C) - 12 Bit / 16 Samples | (0x0D) - 12 Bit / 32 Samples | (0x0E) - 12 Bit / 64 Samples | (0x0F) - 12 Bit / 128 Samples] (Default in **bold**)
- Description: Set the Shunt ADC Resolution.

Texas Instruments: TMP006

- Manufacturer: Texas Instruments
- Measurements: Temperature (Object/Die)
- Interfaces: I^2C
- Libraries: Adafruit_TMP
- Dependencies: Adafruit-TMP
- Datasheet URL: Link
- Product URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

The Things Network: TTN Integration: Data Storage (TTN v2)

- Manufacturer: The Things Network
- Measurements: Variable measurements
- Libraries: requests
- Dependencies: requests

This Input receives and stores measurements from the Data Storage Integration on The Things Network.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Start Offset (seconds)

- Type: Integer
- Description: The duration (seconds) to wait before the first operation

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Application ID

- Type: Text
- Description: The Things Network Application ID

App API Key

- Type: Text
- Description: The Things Network Application API Key

Device ID

- Type: Text
- Description: The Things Network Device ID

CHANNEL OPTIONS

Name

- Type: Text
- Description: A name to distinguish this from others

Variable Name

- Type: Text
- Description: The TTN variable name

The Things Network: TTN Integration: Data Storage (TTN v3, Payload Key)

- Manufacturer: The Things Network
- Measurements: Variable measurements
- Libraries: requests
- Dependencies: requests

This Input receives and stores measurements from the Data Storage Integration on The Things Network. If you have key/value pairs as your payload, enter the key name in Variable Name and the corresponding value for that key will be stored in the measurement database.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Start Offset (seconds)

- Type: Integer
- Description: The duration (seconds) to wait before the first operation

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Application ID

- Type: Text
- Description: The Things Network Application ID

App API Key

- Type: Text
- Description: The Things Network Application API Key

Device ID

- Type: Text
- Description: The Things Network Device ID

CHANNEL OPTIONS

Name

- Type: Text
- Description: A name to distinguish this from others

Variable Name

- Type: Text
- Description: The TTN variable name

The Things Network: TTN Integration: Data Storage (TTN v3, Payload jmespath Expression)

- Manufacturer: The Things Network
- Measurements: Variable measurements
- Libraries: requests, jmespath
- Dependencies: requests, jmespath

This Input receives and stores measurements from the Data Storage Integration on The Things Network. The given Payload jmespath Expression is used as a JMESPATH expression to find the corresponding value that will be stored for that channel. Be sure you select and save the Measurement Unit for each channel. Once the unit has been saved, you can convert to other units in the Convert Measurement section. Example expressions for jmespath (https://jmespath.org) include temperature, sensors[0].temperature, and bathroom.temperature which refer to the temperature as a direct key within the first entry of sensors or as a subkey of bathroom, respectively. Jmespath elements and keys that contain special characters have to be enclosed in double quotes, e.g. "sensor-1".temperature.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Start Offset (seconds)

- Type: Integer
- Description: The duration (seconds) to wait before the first operation

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal

- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Application ID

- Type: Text
- Description: The Things Network Application ID

App API Key

- Type: Text
- Description: The Things Network Application API Key

Device ID

- Type: Text
- Description: The Things Network Device ID

CHANNEL OPTIONS

Name

- Type: Text
- Description: A name to distinguish this from others

Payload jmespath Expression

- Type: Text
- Description: The TTN jmespath expression to return the value to store

Weather: OpenWeatherMap (City, Current)

- Manufacturer: Weather
- Measurements: Humidity/Temperature/Pressure/Wind
- Additional URL: Link

Obtain a free API key at openweathermap.org. If the city you enter does not return measurements, try another city. Note: the free API subscription is limited to 60 calls per minute

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal

- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

API Key

- Type: Text
- Description: The API Key for this service's API

City

- Type: Text
- Description: The city to acquire the weather data

Weather: OpenWeatherMap (Lat/Lon, Current/Future)

- Manufacturer: Weather
- Measurements: Humidity/Temperature/Pressure/Wind
- Interfaces: Mycodo
- Additional URL: Link

Obtain a free API key at openweathermap.org. Notes: The free API subscription is limited to 60 calls per minute. If a Day (Future) time is selected, Minimum and Maximum temperatures are available as measurements.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

API Key

- Type: Text
- Description: The API Key for this service's API

Latitude (decimal)

- Type: Decimal
- Default Value: 33.441792
- Description: The latitude to acquire weather data

Longitude (decimal)

- Type: Decimal
- Default Value: -94.037689
- Description: The longitude to acquire weather data

Time

- Type: Select
- Options: [**Current (Present)** | 1 Day (Future) | 2 Day (Future) | 3 Day (Future) | 4 Day (Future) | 5 Day (Future) | 6 Day (Future) | 7 Day (Future) | 1 Hour (Future) | 2 Hours (Future) | 3 Hours (Future) | 4 Hours (Future) | 5 Hours (Future) | 6 Hours (Future) | 7 Hours (Future) | 8 Hours (Future) | 9 Hours (Future) | 10 Hours (Future) | 11 Hours (Future) | 12 Hours (Future) | 13 Hours (Future) | 14 Hours (Future) | 15 Hours (Future) | 16 Hours (Future) | 17 Hours (Future) | 18 Hours (Future) | 19 Hours (Future) | 20 Hours (Future) | 21 Hours (Future) | 22 Hours (Future) | 23 Hours (Future) | 24 Hours (Future) | 25 Hours (Future) | 26 Hours (Future) | 27 Hours (Future) | 28 Hours (Future) | 29 Hours (Future) | 30 Hours (Future) | 31 Hours (Future) | 32 Hours (Future) | 33 Hours (Future) | 34 Hours (Future) | 35 Hours (Future) | 36 Hours (Future) | 37 Hours (Future) | 38 Hours (Future) | 39 Hours (Future) | 40 Hours (Future) | 41 Hours (Future) | 42 Hours (Future) | 43 Hours (Future) | 44 Hours (Future) | 45 Hours (Future) | 46 Hours (Future) | 47 Hours (Future) | 48 Hours (Future)] (Default in **bold**)
- Description: Select the time for the current or forecast weather

Winsen: MH-Z16

- Manufacturer: Winsen
- Measurements: CO2
- Interfaces: UART, I^2C
- Libraries: smbus2/serial
- Dependencies: smbus2
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Winsen: MH-Z19

- Manufacturer: Winsen
- Measurements: CO2
- Interfaces: UART
- Libraries: serial
- Datasheet URL: Link

This is the version of the sensor that does not include the ability to conduct automatic baseline correction (ABC). See the B version of the sensor if you wish to use ABC.

OPTIONS

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Measurement Range

- Type: Select
- Options: [0 - 1000 ppmv | 0 - 2000 ppmv | 0 - 3000 ppmv | **0 - 5000 ppmv**] (Default in **bold**)
- Description: Set the measuring range of the sensor

COMMANDS

Calibrate Zero Point

- Type: Button

Span Point (ppmv)

- Type: Integer
- Default Value: 2000
- Description: The ppmv concentration for a span point calibration

Calibrate Span Point

- Type: Button

Winsen: MH-Z19B

- Manufacturer: Winsen
- Measurements: CO2
- Interfaces: UART
- Libraries: serial
- Manufacturer URL: Link
- Datasheet URL: Link

This is the B version of the sensor that includes the ability to conduct automatic baseline correction (ABC).

OPTIONS

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Automatic Baseline Correction

- Type: Boolean
- Description: Enable automatic baseline correction (ABC)

Measurement Range

- Type: Select

- Options: [0 - 1000 ppmv | 0 - 2000 ppmv | 0 - 3000 ppmv | **0 - 5000 ppmv** | 0 - 10000 ppmv] (Default in **bold**)
- Description: Set the measuring range of the sensor

COMMANDS

Calibrate Zero Point

- Type: Button

Span Point (ppmv)

- Type: Integer
- Default Value: 2000
- Description: The ppmv concentration for a span point calibration

Calibrate Span Point

- Type: Button

Winsen: ZH03B

- Manufacturer: Winsen
- Measurements: Particulates
- Interfaces: UART
- Libraries: serial
- Manufacturer URL: Link
- Datasheet URL: Link

OPTIONS

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Fan Off After Measure

- Type: Boolean
- Description: Turn the fan on only during the measurement

Fan On Duration

- Type: Decimal
- Default Value: 50.0
- Description: How long to turn the fan on (seconds) before acquiring measurements

Number of Measurements

- Type: Integer
- Default Value: 3
- Description: How many measurements to acquire. If more than 1 are acquired that are less than 1001, the average of the measurements will be stored.

Xiaomi: Miflora

- Manufacturer: Xiaomi
- Measurements: EC/Light/Moisture/Temperature
- Interfaces: BT
- Libraries: miflora
- Dependencies: libglib2.0-dev, miflora, bluepy

OPTIONS

MAC (XX:XX:XX:XX:XX:XX)

- Type: Text
- Description: The MAC address of the Bluetooth device

BT Adapter (hci[X])

- Type: Integer
- Description: The adapter of the Bluetooth device

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Xiaomi: Mijia LYWSD03MMC (ATC and non-ATC modes)

- Manufacturer: Xiaomi
- Measurements: Battery/Humidity/Temperature
- Interfaces: BT
- Libraries: bluepy/bluez
- Dependencies: bluez, bluetooth, libbluetooth-dev, bluepy, pybluez

More information about ATC mode can be found at https://github.com/JsBergbau/MiTemperature2

OPTIONS

MAC (XX:XX:XX:XX:XX:XX)

- Type: Text
- Description: The MAC address of the Bluetooth device

BT Adapter (hci[X])

- Type: Integer
- Description: The adapter of the Bluetooth device

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Description: The duration (seconds) between measurements or actions

Pre Output

- Type: Select
- Description: Turn the selected output on before taking every measurement

Pre Out Duration

- Type: Decimal
- Description: If a Pre Output is selected, set the duration (seconds) to turn the Pre Output on for before every measurement is acquired.

Pre During Measure

- Type: Boolean
- Description: Check to turn the output off after (opposed to before) the measurement is complete

Enable ATC Mode

- Type: Boolean
- Description: Enable sensor ATC mode

5.3 Supported Outputs

Supported Outputs are listed below.

5.3.1 Built-In Outputs (System)

On/Off: MQTT Publish

- Manufacturer: Mycodo
- Interfaces: IP
- Output Types: On/Off
- Libraries: paho-mqtt
- Dependencies: paho-mqtt
- Additional URL: Link

Publish "on" or "off" (or any other strings of your choosing) to an MQTT server.

OPTIONS

CHANNEL OPTIONS

Hostname

- Type: Text
- Default Value: localhost
- Description: The hostname of the MQTT server

Port

- Type: Integer
- Default Value: 1883
- Description: The port of the MQTT server

Topic

- Type: Text
- Default Value: paho/test/single
- Description: The topic to publish with

Keep Alive

- Type: Integer
- Default Value: 60
- Description: The keepalive timeout value for the client. Set to 0 to disable.

Client ID

- Type: Text
- Default Value: client_9cletQYK
- Description: Unique client ID for connecting to the MQTT server

On Payload

- Type: Text
- Default Value: on
- Description: The payload to send when turned on

Off Payload

- Type: Text
- Default Value: off
- Description: The payload to send when turned off

Startup State

- Type: Select
- Description: Set the state when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state when Mycodo shuts down

Force Command

- Type: Boolean
- Description: Always send the command if instructed, regardless of the current state

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

Use Login

- Type: Boolean
- Description: Send login credentials

Username

- Type: Text
- Default Value: user
- Description: Username for connecting to the server

Password

- Type: Text
- Description: Password for connecting to the server. Leave blank to disable.

Value: MQTT Publish

- Manufacturer: Mycodo
- Output Types: Value
- Libraries: paho-mqtt
- Dependencies: paho-mqtt
- Additional URL: Link

Publish a value to an MQTT server.

OPTIONS

CHANNEL OPTIONS

Hostname

- Type: Text
- Default Value: localhost
- Description: The hostname of the MQTT server

Port

- Type: Integer
- Default Value: 1883
- Description: The port of the MQTT server

Topic

- Type: Text
- Default Value: paho/test/single
- Description: The topic to publish with

Keep Alive

- Type: Integer
- Default Value: 60
- Description: The keepalive timeout value for the client. Set to 0 to disable.

Client ID

- Type: Text
- Default Value: client_kx19LPFB
- Description: Unique client ID for connecting to the MQTT server

Off Value

- Type: Integer
- Description: The value to send when an Off command is given

Use Login

- Type: Boolean
- Description: Send login credentials

Username

- Type: Text
- Default Value: user
- Description: Username for connecting to the server

Password

- Type: Text
- Description: Password for connecting to the server.

5.3.2 Built-In Outputs (Devices)

Digital Potentiometer: DS3502

- Manufacturer: Maxim Integrated
- Interfaces: I^2C
- Output Types: Value
- Dependencies: pyusb, Adafruit_Extended_Bus, adafruit-circuitpython-ds3502
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

The DS3502 can generate a 0 - 10k Ohm resistance with 7-bit precision. This equates to 128 possible steps. A value, in Ohms, is passed to this output controller and the step value is calculated and passed to the device. Select whether to round up or down to the nearest step.

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Round Step

- Type: Select
- Options: [**Up** | Down] (Default in **bold**)
- Description: Round direction to the nearest step value

Digital-to-Analog Converter: MCP4728

- Manufacturer: MICROCHIP
- Interfaces: I²C
- Output Types: Value
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-mcp4728
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

VREF (volts)

- Type: Decimal
- Default Value: 4.096
- Description: Set the VREF voltage

CHANNEL OPTIONS

Name

- Type: Text
- Description: A name to distinguish this from others

VREF

- Type: Select

- Options: [**Internal** | VDD] (Default in **bold**)
- Description: Select the channel VREF

Gain

- Type: Select
- Options: [**1X** | 2X] (Default in **bold**)
- Description: Select the channel Gain

Start State

- Type: Select
- Options: [**Previously-Saved State** | Specified Value] (Default in **bold**)
- Description: Select the channel start state

Start Value (volts)

- Type: Decimal
- Description: If Specified Value is selected, set the start state value

Shutdown State

- Type: Select
- Options: [**Previously-Saved Value** | Specified Value] (Default in **bold**)
- Description: Select the channel shutdown state

Shutdown Value (volts)

- Type: Decimal
- Description: If Specified Value is selected, set the shutdown state value

Motor: Grove I2C Motor Driver (Board v1.3)

- Manufacturer: Grove
- Interfaces: I^2C
- Output Types: Volume, On/Off
- Libraries: smbus2
- Dependencies: smbus2
- Manufacturer URL: Link

Controls the Grove I2C Motor Driver Board (v1.3). Both motors will turn at the same time. This output can also dispense volumes of fluid if the motors are attached to peristaltic pumps.

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

CHANNEL OPTIONS

Name

- Type: Text

- Description: A name to distinguish this from others

Motor Speed (0 - 100)

- Type: Integer
- Default Value: 100
- Description: The motor output that determines the speed

Flow Rate Method

- Type: Select
- Options: [**Fastest Flow Rate** | Specify Flow Rate] (Default in **bold**)
- Description: The flow rate to use when pumping a volume

Desired Flow Rate (ml/min)

- Type: Decimal
- Default Value: 10.0
- Description: Desired flow rate in ml/minute when Specify Flow Rate set

Fastest Rate (ml/min)

- Type: Decimal
- Default Value: 100.0
- Description: The fastest rate that the pump can dispense (ml/min)

Motor: Grove I2C Motor Driver (TB6612FNG, Board v1.0)

- Manufacturer: Grove
- Interfaces: I^2C
- Output Types: Volume, On/Off
- Libraries: smbus2
- Dependencies: smbus2
- Manufacturer URL: Link

Controls the Grove I2C Motor Driver Board (v1.3). Both motors will turn at the same time. This output can also dispense volumes of fluid if the motors are attached to peristaltic pumps.

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

CHANNEL OPTIONS

Name

- Type: Text
- Description: A name to distinguish this from others

Motor Speed (0 - 255)

- Type: Integer

- Default Value: 255
- Description: The motor output that determines the speed

Flow Rate Method

- Type: Select
- Options: [**Fastest Flow Rate** | Specify Flow Rate] (Default in **bold**)
- Description: The flow rate to use when pumping a volume

Desired Flow Rate (ml/min)

- Type: Decimal
- Default Value: 10.0
- Description: Desired flow rate in ml/minute when Specify Flow Rate set

Fastest Rate (ml/min)

- Type: Decimal
- Default Value: 100.0
- Description: The fastest rate that the pump can dispense (ml/min)

Minimum On (sec/min)

- Type: Decimal
- Default Value: 1.0
- Description: The minimum duration (seconds) the pump turns on for every 60 second period (only used for Specify Flow Rate mode).

COMMANDS

New I2C Address

- Type: Text
- Default Value: 0x14
- Description: The new I2C to set the sensor to

Set I2C Address

- Type: Button

Motor: L298N DC Motor Controller

- Manufacturer: STMicroelectronics
- Interfaces: GPIO
- Output Types: Volume, On/Off
- Libraries: RPi.GPIO
- Dependencies: RPi.GPIO
- Additional URL: Link

The L298N can control 2 DC motors. If these motors control peristaltic pumps, set the Flow Rate and the output can can be instructed to dispense volumes accurately in addition to being turned on for durations.

OPTIONS

CHANNEL OPTIONS

Name

- Type: Text

- Description: A name to distinguish this from others

Input Pin 1

- Type: Integer
- Description: The Input Pin 1 of the controller (BCM numbering)

Input Pin 2

- Type: Integer
- Description: The Input Pin 2 of the controller (BCM numbering)

Use Enable Pin

- Type: Boolean
- Default Value: True
- Description: Enable the use of the Enable Pin

Enable Pin

- Type: Integer
- Description: The Enable pin of the controller (BCM numbering)

Enable Pin Duty Cycle

- Type: Integer
- Default Value: 50
- Description: The duty cycle to apply to the Enable Pin (percent, 1 - 100)

Direction

- Type: Select
- Options: [**Forward** | Backward] (Default in **bold**)
- Description: The direction to turn the motor

Volume Rate (ml/min)

- Type: Decimal
- Default Value: 150.0
- Description: If a pump, the measured flow rate (ml/min) at the set Duty Cycle

Motor: Stepper Motor, Bipolar (Generic)

- Interfaces: GPIO
- Output Types: Value
- Dependencies: RPi.GPIO
- Manufacturer URLs: Link 1, Link 2
- Datasheet URLs: Link 1, Link 2
- Product URLs: Link 1, Link 2

This is a generic module for bipolar stepper motor drivers such as the DRV8825, A4988, and others. The value passed to the output is the number of steps. A positive value turns clockwise and a negative value turns counter-clockwise.

OPTIONS

CHANNEL OPTIONS

If the Direction or Enable pins are not used, make sure you pull the appropriate pins on your driver high or low to set the proper direction and enable the stepper motor to be energized. Note: For Enable Mode, always having the motor energized will use more energy and produce more heat.

Step Pin

- Type: Integer
- Description: The Step pin of the controller (BCM numbering)

Full Step Delay

- Type: Decimal
- Default Value: 0.005
- Description: The Full Step Delay of the controller

Direction Pin

- Type: Integer
- Description: The Direction pin of the controller (BCM numbering). Set to None to disable.

Enable Pin

- Type: Integer
- Description: The Enable pin of the controller (BCM numbering). Set to None to disable.

Enable Mode

- Type: Select
- Options: [**Only When Turning** | Always] (Default in **bold**)
- Description: Choose when to pull the enable pin high to energize the motor.

Enable at Shutdown

- Type: Select
- Options: [Enable | **Disable**] (Default in **bold**)
- Description: Choose whether the enable pin in pulled high (Enable) or low (Disable) when Mycodo shuts down.

If using a Step Resolution other than Full, and all three Mode Pins are set, they will be set high (1) or how (0) according to the values in parentheses to the right of the selected Step Resolution, e.g. (Mode Pin 1, Mode Pin 2, Mode Pin 3).

Step Resolution

- Type: Select
- Options: [**Full (modes 0, 0, 0)** | Half (modes 1, 0, 0) | 1/4 (modes 0, 1, 0) | 1/8 (modes 1, 1, 0) | 1/16 (modes 0, 0, 1) | 1/32 (modes 1, 0, 1)] (Default in **bold**)
- Description: The Step Resolution of the controller

Mode Pin 1

- Type: Integer
- Description: The Mode Pin 1 of the controller (BCM numbering). Set to None to disable.

Mode Pin 2

- Type: Integer
- Description: The Mode Pin 2 of the controller (BCM numbering). Set to None to disable.

Mode Pin 3

- Type: Integer

- Description: The Mode Pin 3 of the controller (BCM numbering). Set to None to disable.

Motor: ULN2003 Stepper Motor, Unipolar

- Manufacturer: STMicroelectronics
- Interfaces: GPIO
- Output Types: Value
- Dependencies: RPi.GPIO, rpimotorlib
- Manufacturer URL: Link
- Datasheet URLs: Link 1, Link 2

This is a module for the ULN2003 driver.

OPTIONS

CHANNEL OPTIONS

Notes about connecting the ULN2003...

Pin IN1

- Type: Integer
- Default Value: 18
- Description: The pin (BCM numbering) connected to IN1 of the ULN2003

Pin IN2

- Type: Integer
- Default Value: 23
- Description: The pin (BCM numbering) connected to IN2 of the ULN2003

Pin IN3

- Type: Integer
- Default Value: 24
- Description: The pin (BCM numbering) connected to IN3 of the ULN2003

Pin IN4

- Type: Integer
- Default Value: 25
- Description: The pin (BCM numbering) connected to IN4 of the ULN2003

Step Delay

- Type: Decimal
- Default Value: 0.001
- Description: The Step Delay of the controller

Notes about step resolution...

Step Resolution

- Type: Select
- Options: [**Full** | Half | Wave] (Default in **bold**)
- Description: The Step Resolution of the controller

On/Off: GPIO

- Interfaces: GPIO
- Output Types: On/Off
- Libraries: RPi.GPIO
- Dependencies: RPi.GPIO

The specified GPIO pin will be set HIGH (3.3 volts) or LOW (0 volts) when turned on or off, depending on the On State option.

OPTIONS

CHANNEL OPTIONS

GPIO Pin (BCM)

- Type: Integer
- Description: The pin to control the state of

Startup State

- Type: Select
- Description: Set the state when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state when Mycodo shuts down

On State

- Type: Select
- Options: [**HIGH** | LOW] (Default in **bold**)
- Description: The state of the GPIO that corresponds to an On state

Trigger Functions at Startup

- Type: Boolean
- Description: Whether to trigger functions when the output switches at startup

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

On/Off: Grove Multichannel Relay (4- or 8-Channel board)

- Manufacturer: Grove
- Interfaces: I^2C
- Output Types: On/Off
- Libraries: smbus2
- Dependencies: smbus2
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

Controls the 4 or 8 channel Grove multichannel relay board.

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

CHANNEL OPTIONS

Name

- Type: Text
- Description: A name to distinguish this from others

Startup State

- Type: Select
- Description: Set the state of the relay when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state of the relay when Mycodo shuts down

On State

- Type: Select
- Options: [**HIGH** | LOW] (Default in **bold**)
- Description: The state of the GPIO that corresponds to an On state

Trigger Functions at Startup

- Type: Boolean
- Description: Whether to trigger functions when the output switches at startup

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

On/Off: HS300 Kasa Smart WiFi Power Strip

- Manufacturer: TP-Link
- Interfaces: IP
- Output Types: On/Off
- Dependencies: python-kasa
- Manufacturer URL: Link

This output controls the 6 outlets of the Kasa HS300 Smart WiFi Power Strip.

OPTIONS

Host

- Type: Text
- Default Value: 192.168.0.50
- Description: Host address or IP

Status Update (Sec)

- Type: Integer
- Default Value: 60
- Description: The period (seconds) between checking if connected and output states.

CHANNEL OPTIONS

Name

- Type: Text
- Default Value: Outlet Name
- Description: A name to distinguish this from others

Startup State

- Type: Select
- Description: Set the state when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state when Mycodo shuts down

Trigger Functions at Startup

- Type: Boolean
- Description: Whether to trigger functions when the output switches at startup

Force Command

- Type: Boolean
- Description: Always send the command if instructed, regardless of the current state

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

On/Off: KP303 Kasa Smart WiFi Power Strip

- Manufacturer: TP-Link
- Interfaces: IP
- Output Types: On/Off
- Dependencies: python-kasa
- Manufacturer URL: Link

This output controls the 3 outlets of the Kasa KP303 Smart WiFi Power Strip.

OPTIONS

Host

- Type: Text
- Default Value: 192.168.0.50
- Description: Host address or IP

Status Update (Sec)

- Type: Integer
- Default Value: 60

- Description: The period (seconds) between checking if connected and output states.

CHANNEL OPTIONS

Name

- Type: Text
- Default Value: Outlet Name
- Description: A name to distinguish this from others

Startup State

- Type: Select
- Description: Set the state when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state when Mycodo shuts down

Trigger Functions at Startup

- Type: Boolean
- Description: Whether to trigger functions when the output switches at startup

Force Command

- Type: Boolean
- Description: Always send the command if instructed, regardless of the current state

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

On/Off: MCP23017 16-Channel I/O Expander

- Manufacturer: MICROCHIP
- Interfaces: I^2C
- Output Types: On/Off
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-mcp230xx
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

Controls the 16 channels of the MCP23017.

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

CHANNEL OPTIONS

Name

- Type: Text
- Description: A name to distinguish this from others

Startup State

- Type: Select
- Description: Set the state of the GPIO when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state of the GPIO when Mycodo shuts down

On State

- Type: Select
- Options: [**HIGH** | LOW] (Default in **bold**)
- Description: The state of the GPIO that corresponds to an On state

Trigger Functions at Startup

- Type: Boolean
- Description: Whether to trigger functions when the output switches at startup

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

On/Off: PCF8574 8-Channel {lazy_gettext('I/O Expander')}

- Manufacturer: Texas Instruments
- Interfaces: I^2C
- Output Types: On/Off
- Libraries: smbus2
- Dependencies: smbus2
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

Controls the 8 channels of the PCF8574.

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

CHANNEL OPTIONS

Name

- Type: Text
- Description: A name to distinguish this from others

Startup State

- Type: Select
- Description: Set the state of the GPIO when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state of the GPIO when Mycodo shuts down

On State

- Type: Select
- Options: [**HIGH** | LOW] (Default in **bold**)
- Description: The state of the GPIO that corresponds to an On state

Trigger Functions at Startup

- Type: Boolean
- Description: Whether to trigger functions when the output switches at startup

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

On/Off: Python Code

- Interfaces: Python
- Output Types: On/Off
- Dependencies: pylint

Python 3 code will be executed when this output is turned on or off.

OPTIONS

CHANNEL OPTIONS

On Command

- Description: Python code to execute when the output is instructed to turn on

Off Command

- Description: Python code to execute when the output is instructed to turn off

Startup State

- Type: Select
- Description: Set the state when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state when Mycodo shuts down

Trigger Functions at Startup

- Type: Boolean
- Description: Whether to trigger functions when the output switches at startup

Force Command

- Type: Boolean
- Description: Always send the command if instructed, regardless of the current state

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

On/Off: Shell Script

- Output Types: On/Off
- Libraries: subprocess.Popen

Commands will be executed in the Linux shell by the specified user when this output is turned on or off.

OPTIONS

CHANNEL OPTIONS

On Command

- Type: Text
- Default Value: /home/pi/script_on_off.sh on
- Description: Command to execute when the output is instructed to turn on

Off Command

- Type: Text
- Default Value: /home/pi/script_on_off.sh off
- Description: Command to execute when the output is instructed to turn off

User

- Type: Text
- Default Value: mycodo
- Description: The user to execute the command

Startup State

- Type: Select
- Description: Set the state when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state when Mycodo shuts down

Trigger Functions at Startup

- Type: Boolean
- Description: Whether to trigger functions when the output switches at startup

Force Command

- Type: Boolean

- Description: Always send the command if instructed, regardless of the current state

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

On/Off: Wireless 315/433 MHz

- Interfaces: GPIO
- Output Types: On/Off
- Libraries: rpi-rf
- Dependencies: RPi.GPIO, rpi_rf

This output uses a 315 or 433 MHz transmitter to turn wireless power outlets on or off. Run ~/Mycodo/mycodo/devices/wireless_rpi_rf.py with a receiver to discover the codes produced from your remote.

OPTIONS

CHANNEL OPTIONS

GPIO Pin (BCM)

- Type: Integer
- Description: The pin to control the state of

On Command

- Type: Text
- Default Value: 22559
- Description: Command to execute when the output is instructed to turn on

Off Command

- Type: Text
- Default Value: 22558
- Description: Command to execute when the output is instructed to turn off

Protocol

- Type: Select
- Options: [**1** | 2 | 3 | 4 | 5] (Default in **bold**)
- Description: Wireless protocol

Pulse Length

- Type: Integer
- Default Value: 189
- Description: Wireless pulse length

Startup State

- Type: Select
- Description: Set the state when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state when Mycodo shuts down

Trigger Functions at Startup

- Type: Boolean
- Description: Whether to trigger functions when the output switches at startup

Force Command

- Type: Boolean
- Description: Always send the commad if instructed, regardless of the current state

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

PWM: GPIO

- Interfaces: GPIO
- Output Types: PWM
- Libraries: pigpio
- Dependencies: pigpio, pigpio

See the PWM section of the manual for PWM information and determining which pins may be used for each library option.

CHANNEL OPTIONS

GPIO Pin (BCM)

- Type: Integer
- Description: The pin to control the state of

Startup State

- Type: Select
- Description: Set the state when Mycodo starts

Startup Value

- Type: Decimal
- Description: The value when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state when Mycodo shuts down

Shutdown Value

- Type: Decimal
- Description: The value when Mycodo shuts down

Library

- Type: Select
- Options: [**Any Pin, <= 40 kHz** | Hardware Pin, <= 30 MHz] (Default in **bold**)
- Description: Which method to produce the PWM signal (hardware pins can produce higher frequencies)

Frequency (Hertz)

- Type: Integer
- Default Value: 22000

- Description: The Herts to output the PWM signal (0 - 70,000)

Invert Signal

- Type: Boolean
- Description: Invert the PWM signal

Invert Stored Signal

- Type: Boolean
- Description: Invert the value that is saved to the measurement database

Trigger Functions at Startup

- Type: Boolean
- Description: Whether to trigger functions when the output switches at startup

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

COMMANDS

Set the Duty Cycle.

Duty Cycle

- Type: Decimal
- Description: The duty cycle to set

Set Duty Cycle

- Type: Button

PWM: PCA9685 16-Channel LED Controller

- Manufacturer: NXP Semiconductors
- Interfaces: I^2C
- Output Types: PWM
- Libraries: adafruit-pca9685
- Dependencies: adafruit-pca9685
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

The PCA9685 can output a PWM signal to 16 channels at a frequency between 40 and 1600 Hz.

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

Frequency (Hertz)

- Type: Integer
- Default Value: 1600
- Description: The Herts to output the PWM signal (40 - 1600)

CHANNEL OPTIONS

Name

- Type: Text
- Description: A name to distinguish this from others

Startup State

- Type: Select
- Description: Set the state when Mycodo starts

Startup Value

- Type: Decimal
- Description: The value when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state when Mycodo shuts down

Shutdown Value

- Type: Decimal
- Description: The value when Mycodo shuts down

Invert Signal

- Type: Boolean
- Description: Invert the PWM signal

Invert Stored Signal

- Type: Boolean
- Description: Invert the value that is saved to the measurement database

Trigger Functions at Startup

- Type: Boolean
- Description: Whether to trigger functions when the output switches at startup

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

PWM: Python 3 Code

- Interfaces: Python
- Output Types: PWM
- Dependencies: pylint

Python 3 code will be executed when this output is turned on or off. The "duty_cycle" object is a float value that represents the duty cycle that has been set.

CHANNEL OPTIONS

Python 3 Code

- Description: Python code to execute to set the PWM duty cycle (%)

User

- Type: Text
- Default Value: mycodo
- Description: The user to execute the command

Startup State

- Type: Select
- Description: Set the state when Mycodo starts

Startup Value

- Type: Decimal
- Description: The value when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state when Mycodo shuts down

Shutdown Value

- Type: Decimal
- Description: The value when Mycodo shuts down

Invert Signal

- Type: Boolean
- Description: Invert the PWM signal

Invert Stored Signal

- Type: Boolean
- Description: Invert the value that is saved to the measurement database

Trigger Functions at Startup

- Type: Boolean
- Description: Whether to trigger functions when the output switches at startup

Force Command

- Type: Boolean
- Description: Always send the command if instructed, regardless of the current state

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

COMMANDS

Set the Duty Cycle.

Duty Cycle

- Type: Decimal
- Description: The duty cycle to set

Set Duty Cycle

- Type: Button

PWM: Shell Script

- Interfaces: Shell
- Output Types: PWM
- Libraries: subprocess.Popen

Commands will be executed in the Linux shell by the specified user when the duty cycle is set for this output. The string "((duty_cycle))" in the command will be replaced with the duty cycle being set prior to execution.

OPTIONS

CHANNEL OPTIONS

Bash Command

- Type: Text
- Default Value: /home/pi/script_pwm.sh ((duty_cycle))
- Description: Command to execute to set the PWM duty cycle (%)

User

- Type: Text
- Default Value: mycodo
- Description: The user to execute the command

Startup State

- Type: Select
- Description: Set the state when Mycodo starts

Startup Value

- Type: Decimal
- Description: The value when Mycodo starts

Shutdown State

- Type: Select
- Description: Set the state when Mycodo shuts down

Shutdown Value

- Type: Decimal
- Description: The value when Mycodo shuts down

Invert Signal

- Type: Boolean
- Description: Invert the PWM signal

Invert Stored Signal

- Type: Boolean
- Description: Invert the value that is saved to the measurement database

Trigger Functions at Startup

- Type: Boolean
- Description: Whether to trigger functions when the output switches at startup

Force Command

- Type: Boolean
- Description: Always send the commad if instructed, regardless of the current state

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

Peristaltic Pump: Atlas Scientific

- Manufacturer: Atlas Scientific
- Interfaces: I^2C, UART, FTDI
- Output Types: Volume, On/Off
- Dependencies: pylibftdi
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

Atlas Scientific peristaltic pumps can be set to dispense at their maximum rate or a rate can be specified. Their minimum flow rate is 0.5 ml/min and their maximum is 105 ml/min.

OPTIONS

I^2C Address

- Type: Text
- Description: The I2C address of the device

I^2C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

FTDI Device

- Type: Text
- Description: The FTDI device connected to the input/output/etc.

UART Device

- Type: Text
- Description: The UART device location (e.g. /dev/ttyUSB1)

CHANNEL OPTIONS

Flow Rate Method

- Type: Select
- Options: [**Fastest Flow Rate** | Specify Flow Rate] (Default in **bold**)
- Description: The flow rate to use when pumping a volume

Desired Flow Rate (ml/min)

- Type: Decimal
- Default Value: 10.0
- Description: Desired flow rate in ml/minute when Specify Flow Rate set

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

COMMANDS

Calibration: a calibration can be performed to increase the accuracy of the pump. It's a good idea to clear the calibration before calibrating. First, remove all air from the line by pumping the fluid you would like to calibrate to through the pump hose. Next, press Dispense Amount and the pump will be instructed to dispense 10 ml (unless you changed the default value). Measure how much fluid was actually dispensed, enter this value in the Actual Volume Dispensed (ml) field, and press Calibrate to Dispensed Amount. Now any further pump volumes dispensed should be accurate.

Clear Calibration

- Type: Button

Volume to Dispense (ml)

- Type: Decimal
- Default Value: 10.0
- Description: The volume (ml) that is instructed to be dispensed

Dispense Amount

- Type: Button

Actual Volume Dispensed (ml)

- Type: Decimal
- Default Value: 10.0
- Description: The actual volume (ml) that was dispensed

Calibrate to Dispensed Amount

- Type: Button

The I2C address can be changed. Enter a new address in the 0xYY format (e.g. 0x22, 0x50), then press Set I2C Address. Remember to deactivate and change the I2C address option after setting the new address.

New I2C Address

- Type: Text
- Default Value: 0x67
- Description: The new I2C to set the device to

Set I2C Address

- Type: Button

Peristaltic Pump: GPIO

- Interfaces: GPIO
- Output Types: Volume, On/Off
- Libraries: RPi.GPIO
- Dependencies: RPi.GPIO

This output turns a GPIO pin HIGH and LOW to control power to a generic peristaltic pump. The peristaltic pump can then be turned on for a duration or, after determining the pump's maximum flow rate, instructed to dispense a specific volume at the maximum rate or at a specified rate.

OPTIONS

CHANNEL OPTIONS

GPIO Pin (BCM)

- Type: Integer
- Description: The pin to control the state of

On State

- Type: Select
- Options: [**HIGH** | LOW] (Default in **bold**)
- Description: The state of the GPIO that corresponds to an On state

Fastest Rate (ml/min)

- Type: Decimal
- Default Value: 150.0
- Description: The fastest rate that the pump can dispense (ml/min)

Minimum On (sec/min)

- Type: Decimal
- Default Value: 1.0
- Description: The minimum duration (seconds) the pump should be turned on for every 60 second period

Flow Rate Method

- Type: Select
- Options: [**Fastest Flow Rate** | Specify Flow Rate] (Default in **bold**)
- Description: The flow rate to use when pumping a volume

Desired Flow Rate (ml/min)

- Type: Decimal
- Default Value: 10.0
- Description: Desired flow rate in ml/minute when Specify Flow Rate set

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

Peristaltic Pump: MCP23017 16-Channel I/O Expander

- Manufacturer: MICROCHIP
- Interfaces: I^2C
- Output Types: Volume, On/Off
- Dependencies: pyusb, Adafruit-extended-bus, adafruit-circuitpython-mcp230xx
- Manufacturer URL: Link
- Datasheet URL: Link
- Product URL: Link

Controls the 16 channels of the MCP23017 with a relay and peristaltic pump connected to each channel.

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

CHANNEL OPTIONS

On State

- Type: Select
- Options: [**HIGH** | LOW] (Default in **bold**)
- Description: The state of the output channel that corresponds to the pump being on

Fastest Rate (ml/min)

- Type: Decimal
- Default Value: 150.0
- Description: The fastest rate that the pump can dispense (ml/min)

Minimum On (sec/min)

- Type: Decimal
- Default Value: 1.0
- Description: The minimum duration (seconds) the pump should be turned on for every 60 second period

Flow Rate Method

- Type: Select
- Options: [**Fastest Flow Rate** | Specify Flow Rate] (Default in **bold**)
- Description: The flow rate to use when pumping a volume

Desired Flow Rate (ml/min)

- Type: Decimal
- Default Value: 10.0
- Description: Desired flow rate in ml/minute when Specify Flow Rate set

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

Peristaltic Pump: PCF8574 8-Channel I/O Expander

- Manufacturer: Texas Instruments
- Interfaces: I²C
- Output Types: Volume, On/Off
- Libraries: smbus2
- Dependencies: smbus2
- Manufacturer URL: Link
- Datasheet URL: Link

- Product URL: Link

Controls the 8 channels of the PCF8574 with a relay and peristaltic pump connected to each channel.

OPTIONS

I²C Address

- Type: Text
- Description: The I2C address of the device

I²C Bus

- Type: Integer
- Description: The I2C bus the device is connected to

CHANNEL OPTIONS

On State

- Type: Select
- Options: [**HIGH** | LOW] (Default in **bold**)
- Description: The state of the output channel that corresponds to the pump being on

Fastest Rate (ml/min)

- Type: Decimal
- Default Value: 150.0
- Description: The fastest rate that the pump can dispense (ml/min)

Minimum On (sec/min)

- Type: Decimal
- Default Value: 1.0
- Description: The minimum duration (seconds) the pump should be turned on for every 60 second period

Flow Rate Method

- Type: Select
- Options: [**Fastest Flow Rate** | Specify Flow Rate] (Default in **bold**)
- Description: The flow rate to use when pumping a volume

Desired Flow Rate (ml/min)

- Type: Decimal
- Default Value: 10.0
- Description: Desired flow rate in ml/minute when Specify Flow Rate set

Current (Amps)

- Type: Decimal
- Description: The current draw of the device being controlled

Spacer

A spacer to organize Outputs.

OPTIONS

Color

- Type: Text

- Default Value: #000000
- Description: The color of the name text

5.4 Supported Functions

Supported Functions are listed below.

5.4.1 Built-In Functions

Average (Last, Multiple)

This function acquires the last measurement of those that are selected, averages them, then stores the resulting value as the selected measurement and unit.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 60
- Description: The duration (seconds) between measurements or actions

Start Offset

- Type: Integer
- Default Value: 10
- Description: The duration (seconds) to wait before the first operation

Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement

- Description: Measurement to replace "x" in the equation

Average (Past, Single)

This function acquires the past measurements (within Max Age) for the selected measurement, averages them, then stores the resulting value as the selected measurement and unit.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 60
- Description: The duration (seconds) between measurements or actions

Start Offset

- Type: Integer
- Default Value: 10
- Description: The duration (seconds) to wait before the first operation

Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Measurement to replace "x" in the equation

Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Backup to Remote Host (rsync)

- Dependencies: rsync

This function will use rsync to back up assets on this system to a remote system. Your remote system needs to have an SSH server running and rsync installed. This system will need rsync installed and be able to access your remote system via SSH keyfile (login without a password). You can do this by creating an SSH key on this system running Mycodo with "ssh-keygen" (leave the password field empty), then run "ssh-copy-id -i ~/.ssh/id_rsa.pub pi@REMOTE_HOST_IP" to transfer your public SSH key to your remote system (changing pi and REMOTE_HOST_IP to the appropriate user and host of your remote system). You can test if this worked by trying to connect to your remote system with "ssh pi@REMOTE_HOST_IP" and you should log in without being asked for a password. Be careful not to set the Period too low, which could cause the function to begin running before the previous operation(s) complete. Therefore, it is recommended to set a relatively long Period (greater than 10 minutes). The default Period is 15 days. Note that the Period will reset if the system or the Mycodo daemon restarts and the Function will run, generating new settings and measurement archives that will be synced. There are two common ways to use this Function: 1) A short period (1 hour), only have Backup Camera Directories enabled, and use the Backup Settings Now and Backup Measurements Now buttons manually to perform a backup, and 2) A long period (15 days), only have Backup Settings and Backup Measurements enabled. You can even create two of these Functions with one set up to perform long-Period settings and measurement backups and the other set up to perform short-Period camera backups.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 1296000
- Description: The duration (seconds) between measurements or actions

Start Offset

- Type: Integer
- Default Value: 300
- Description: The duration (seconds) to wait before the first operation

Local User

- Type: Text
- Default Value: pi
- Description: The user on this system that will run rsync

Remote User

- Type: Text
- Default Value: pi
- Description: The user to log in to the remote host

Remote Host

- Type: Text
- Default Value: 192.168.0.50
- Description: The IP or host address to send the backup to

Remote Backup Path

- Type: Text
- Default Value: /home/pi/backup_mycodo
- Description: The path to backup to on the remote host

Rsync Timeout

- Type: Integer
- Default Value: 3600
- Description: How long to allow rsync to complete (seconds)

Backup Settings Export File

- Type: Boolean
- Default Value: True
- Description: Create and backup exported settings file

Remove Local Settings Backups

- Type: Boolean
- Description: Remove local settings backups after successful transfer to remote host

Backup Measurements

- Type: Boolean
- Default Value: True
- Description: Backup all influxdb measurements

Remove Local Measurements Backups

- Type: Boolean
- Description: Remove local measurements backups after successful transfer to remote host

Backup Camera Directories

- Type: Boolean
- Default Value: True
- Description: Backup all camera directories

Remove Local Camera Images

- Type: Boolean
- Description: Remove local camera images after successful transfer to remote host

SSH Port

- Type: Integer
- Default Value: 22
- Description: Specify a nonstandard SSH port

COMMANDS

Backup of settings are only created if the Mycodo version or database versions change. This is due to this Function running periodically- if it created a new backup every Period, there would soon be many identical backups. Therefore, if you want to induce the backup of settings, measurements, or camera directories and sync them to your remote system, use the buttons below.

Backup Settings Now

- Type: Button

Backup Measurements Now

- Type: Button

Backup Camera Directories Now

- Type: Button

Bang-Bang Hysteretic (On/Off) (Raise/Lower)

A simple bang-bang control for controlling one output from one input. Select an input, an output, enter a setpoint and a hysteresis, and select a direction. The output will turn on when the input is below (lower = setpoint - hysteresis) and turn off when the input is above (higher = setpoint + hysteresis). This is the behavior when Raise is selected, such as when heating. Lower direction has the opposite behavior - it will try to turn the output on in order to drive the input lower.

OPTIONS

Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Select a measurement the selected output will affect

Output

- Type: Select Device, Measurement, and Channel
- Selections: Output
- Description: Select an output to control that will affect the measurement

Setpoint

- Type: Decimal
- Default Value: 50
- Description: The desired setpoint

Hysteresis

- Type: Decimal
- Default Value: 1
- Description: The amount above and below the setpoint that defines the control band

Direction

- Type: Select
- Options: [**Raise** | Lower] (Default in **bold**)
- Description: Raise means the measurement will increase when the control is on (heating). Lower means the measurement will decrease when the output is on (cooling)

Period (seconds)

- Type: Decimal
- Default Value: 5
- Description: The duration (seconds) between measurements or actions

Bang-Bang Hysteretic (On/Off) (Raise/Lower/Both)

A simple bang-bang control for controlling one or two outputs from one input. Select an input, a raise and/or lower output, enter a setpoint and a hysteresis, and select a direction. The output will turn on when the input is below (lower = setpoint - hysteresis) and turn off when the input is above (higher = setpoint + hysteresis). This is the behavior when Raise is selected, such as when heating. Lower direction has the opposite behavior - it will try to turn the output on in order to drive the input lower. The Both option will raise and lower. Note: This output will only work with On/Off Outputs.

OPTIONS

Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Select a measurement the selected output will affect

Output (Raise)

- Type: Select Device, Measurement, and Channel
- Selections: Output
- Description: Select an output to control that will raise the measurement

Output (Lower)

- Type: Select Device, Measurement, and Channel
- Selections: Output
- Description: Select an output to control that will lower the measurement

Setpoint

- Type: Decimal
- Default Value: 50
- Description: The desired setpoint

Hysteresis

- Type: Decimal
- Default Value: 1
- Description: The amount above and below the setpoint that defines the control band

Direction

- Type: Select
- Options: [Raise | Lower | **Both**] (Default in **bold**)
- Description: Raise means the measurement will increase when the control is on (heating). Lower means the measurement will decrease when the output is on (cooling)

Period (seconds)

- Type: Decimal
- Default Value: 5
- Description: The duration (seconds) between measurements or actions

Bang-Bang Hysteretic (PWM) (Raise/Lower/Both)

A simple bang-bang control for controlling one PWM output from one input. Select an input, a PWM output, enter a setpoint and a hysteresis, and select a direction. The output will turn on when the input is below below (lower = setpoint - hysteresis) and turn off when the input is above (higher = setpoint + hysteresis). This is the behavior when Raise is selected, such as when heating. Lower direction has the opposite behavior - it will try to turn the output on in order to drive the input lower. The Both option will raise and lower. Note: This output will only work with PWM Outputs.

OPTIONS

Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Select a measurement the selected output will affect

Output

- Type: Select Device, Measurement, and Channel
- Selections: Output
- Description: Select an output to control that will affect the measurement

Setpoint

- Type: Decimal
- Default Value: 50
- Description: The desired setpoint

Hysteresis

- Type: Decimal
- Default Value: 1
- Description: The amount above and below the setpoint that defines the control band

Direction

- Type: Select
- Options: [Raise | Lower | **Both**] (Default in **bold**)
- Description: Raise means the measurement will increase when the control is on (heating). Lower means the measurement will decrease when the output is on (cooling)

Period (seconds)

- Type: Decimal
- Default Value: 5
- Description: The duration (seconds) between measurements or actions

Duty Cycle (increase)

- Type: Decimal
- Default Value: 90
- Description: The duty cycle to increase the measurement

Duty Cycle (maintain)

- Type: Decimal
- Default Value: 55
- Description: The duty cycle to maintain the measurement

Duty Cycle (decrease)

- Type: Decimal
- Default Value: 20
- Description: The duty cycle to decrease the measurement

Duty Cycle (shutdown)

- Type: Decimal
- Description: The duty cycle to set when the function shuts down

Difference

This function acquires 2 measurements, calculates the difference, and stores the resulting value as the selected measurement and unit.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 60
- Description: The duration (seconds) between measurements or actions

Measurement A

- Type: Select Measurement
- Selections: Input, Function
- Description:

Measurement A Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement B

- Type: Select Measurement
- Selections: Input, Function
- Description:

Measurement B Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Reverse Order

- Type: Boolean
- Description: Reverse the order in the calculation

Absolute Difference

- Type: Boolean
- Description: Return the absolute value of the difference

Display: Generic LCD 16x2 (I2C)

This Function outputs to a generic 16x2 LCD display via I2C. Since this display can show 2 lines at a time, channels are added in sets of 2 when Number of Line Sets is modified. Every Period, the LCD will refresh and display the next set of lines. Therefore, the first 2 lines that are displayed are channels 0 and 1, then 2 and 3, and so on. After all channels have been displayed, it will cycle back to the beginning.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 10
- Description: The duration (seconds) between measurements or actions

I2C Address

- Type: Text
- Default Value: 0x20
- Description: The I2C address of the device

I2C Bus

- Type: Integer
- Default Value: 1
- Description: The I2C bus the device is connected to

Number of Line Sets

- Type: Integer
- Default Value: 1
- Description: How many sets of lines to cycle on the LCD

CHANNEL OPTIONS

Line Display Type

- Type: Select
- Description: What to display on the line

Measurement

- Type: Select Measurement
- Selections: Input, Function, Output, PID
- Description: Measurement to display on the line

Measurement Max Age

- Type: Decimal
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement Label

- Type: Text
- Description: Set to overwrite the default measurement label

Measurement Decimal

- Type: Integer
- Default Value: 1

- Description: The number of digits after the decimal

Text

- Type: Text
- Default Value: Text
- Description: Text to display

Display Unit

- Type: Boolean
- Default Value: True
- Description: Display the measurement unit (if available)

COMMANDS

Backlight On

- Type: Button

Backlight Off

- Type: Button

Backlight Flashing On

- Type: Button

Backlight Flashing Off

- Type: Button

Display: Generic LCD 20x4 (I2C)

This Function outputs to a generic 20x4 LCD display via I2C. Since this display can show 4 lines at a time, channels are added in sets of 4 when Number of Line Sets is modified. Every Period, the LCD will refresh and display the next set of lines. Therefore, the first 4 lines that are displayed are channels 0, 1, 2, and 3, then 4, 5, 6, and 7, and so on. After all channels have been displayed, it will cycle back to the beginning.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 10
- Description: The duration (seconds) between measurements or actions

I2C Address

- Type: Text
- Default Value: 0x20
- Description: The I2C address of the device

I2C Bus

- Type: Integer
- Default Value: 1
- Description: The I2C bus the device is connected to

Number of Line Sets

- Type: Integer
- Default Value: 1

- Description: How many sets of lines to cycle on the LCD

CHANNEL OPTIONS

Line Display Type

- Type: Select
- Description: What to display on the line

Measurement

- Type: Select Measurement
- Selections: Input, Function, Output, PID
- Description: Measurement to display on the line

Max Age

- Type: Decimal
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement Label

- Type: Text
- Description: Set to overwrite the default measurement label

Measurement Decimal

- Type: Integer
- Default Value: 1
- Description: The number of digits after the decimal

Text

- Type: Text
- Default Value: Text
- Description: Text to display

Display Unit

- Type: Boolean
- Default Value: True
- Description: Display the measurement unit (if available)

COMMANDS

Backlight On

- Type: Button

Backlight Off

- Type: Button

Display: Grove LCD 16x2 (I2C)

This Function outputs to the Grove 16x2 LCD display via I2C. Since this display can show 2 lines at a time, channels are added in sets of 2 when Number of Line Sets is modified. Every Period, the LCD will refresh and display the next set of lines. Therefore, the first 2 lines that are displayed are channels 0 and 1, then 2 and 3, and so on. After all channels have been displayed, it will cycle back to the beginning.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 10
- Description: The duration (seconds) between measurements or actions

I2C Address

- Type: Text
- Default Value: 0x3e
- Description: The I2C address of the device

I2C Bus

- Type: Integer
- Default Value: 1
- Description: The I2C bus the device is connected to

Backlight I2C Address

- Type: Text
- Default Value: 0x62
- Description: I2C address to control the backlight

Number of Line Sets

- Type: Integer
- Default Value: 1
- Description: How many sets of lines to cycle on the LCD

Backlight Red (0 - 255)

- Type: Integer
- Default Value: 255
- Description: Set the red color value of the backlight on startup.

Backlight Green (0 - 255)

- Type: Integer
- Default Value: 255
- Description: Set the green color value of the backlight on startup.

Backlight Blue (0 - 255)

- Type: Integer
- Default Value: 255
- Description: Set the blue color value of the backlight on startup.

CHANNEL OPTIONS

Line Display Type

- Type: Select
- Description: What to display on the line

Measurement

- Type: Select Measurement
- Selections: Input, Function, Output, PID

- Description: Measurement to display on the line

Max Age

- Type: Decimal
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement Label

- Type: Text
- Description: Set to overwrite the default measurement label

Measurement Decimal

- Type: Integer
- Default Value: 1
- Description: The number of digits after the decimal

Text

- Type: Text
- Default Value: Text
- Description: Text to display

Display Unit

- Type: Boolean
- Default Value: True
- Description: Display the measurement unit (if available)

COMMANDS

Backlight On

- Type: Button

Backlight Off

- Type: Button

Color (RGB)

- Type: Text
- Default Value: 255,0,0
- Description: Color as R,G,B values (e.g. "255,0,0" without quotes)

Set Backlight Color

- Type: Button

Display: SSD1306 OLED 128x32 [2 Lines] (I2C)

- Dependencies: libjpeg-dev, Pillow, pyusb, Adafruit-extended-bus, adafruit-circuitpython-framebuf, adafruit-circuitpython-ssd1306

This Function outputs to a 128x32 SSD1306 OLED display via I2C. This display Function will show 2 lines at a time, so channels are added in sets of 2 when Number of Line Sets is modified. Every Period, the LCD will refresh and display the next set of lines. Therefore, the first set of lines that are displayed are channels 0 - 1, then 2 - 3, and so on. After all channels have been displayed, it will cycle back to the beginning.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 10
- Description: The duration (seconds) between measurements or actions

I2C Address

- Type: Text
- Default Value: 0x3c
- Description: The I2C address of the device

I2C Bus

- Type: Integer
- Default Value: 1
- Description: The I2C bus the device is connected to

Number of Line Sets

- Type: Integer
- Default Value: 1
- Description: How many sets of lines to cycle on the LCD

Reset Pin

- Type: Integer
- Default Value: 17
- Description: The pin (BCM numbering) connected to RST of the display

Characters Per Line

- Type: Integer
- Default Value: 17
- Description: The maximum number of characters to display per line

Use Non-Default Font

- Type: Boolean
- Description: Don't use the default font. Enable to specify the path to a font to use.

Non-Default Font Path

- Type: Text
- Default Value: /usr/share/fonts/truetype/dejavu//DejaVuSans.ttf
- Description: The path to the non-default font to use

Font Size (pt)

- Type: Integer
- Default Value: 12
- Description: The size of the font, in points

CHANNEL OPTIONS

Line Display Type

- Type: Select
- Description: What to display on the line

Measurement

- Type: Select Measurement
- Selections: Input, Function, Output, PID
- Description: Measurement to display on the line

Max Age

- Type: Decimal
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement Label

- Type: Text
- Description: Set to overwrite the default measurement label

Measurement Decimal

- Type: Integer
- Default Value: 1
- Description: The number of digits after the decimal

Text

- Type: Text
- Default Value: Text
- Description: Text to display

Display Unit

- Type: Boolean
- Default Value: True
- Description: Display the measurement unit (if available)

Display: SSD1306 OLED 128x32 [2 Lines] (SPI)

- Dependencies: libjpeg-dev, Pillow, pyusb, Adafruit-GPIO, Adafruit-extended-bus, adafruit-circuitpython-framebuf, adafruit-circuitpython-ssd1306

This Function outputs to a 128x32 SSD1306 OLED display via SPI. This display Function will show 2 lines at a time, so channels are added in sets of 2 when Number of Line Sets is modified. Every Period, the LCD will refresh and display the next set of lines. Therefore, the first set of lines that are displayed are channels 0 - 1, then 2 - 3, and so on. After all channels have been displayed, it will cycle back to the beginning.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 10
- Description: The duration (seconds) between measurements or actions

Number of Line Sets

- Type: Integer
- Default Value: 1
- Description: How many sets of lines to cycle on the LCD

SPI Device

- Type: Integer
- Description: The SPI device

SPI Bus

- Type: Integer
- Description: The SPI bus

DC Pin

- Type: Integer
- Default Value: 16
- Description: The pin (BCM numbering) connected to DC of the display

Reset Pin

- Type: Integer
- Default Value: 19
- Description: The pin (BCM numbering) connected to RST of the display

CS Pin

- Type: Integer
- Default Value: 17
- Description: The pin (BCM numbering) connected to CS of the display

Characters Per Line

- Type: Integer
- Default Value: 17
- Description: The maximum number of characters to display per line

Use Non-Default Font

- Type: Boolean
- Description: Don't use the default font. Enable to specify the path to a font to use.

Non-Default Font Path

- Type: Text
- Default Value: /usr/share/fonts/truetype/dejavu//DejaVuSans.ttf
- Description: The path to the non-default font to use

Font Size (pt)

- Type: Integer
- Default Value: 12
- Description: The size of the font, in points

CHANNEL OPTIONS

Line Display Type

- Type: Select
- Description: What to display on the line

Measurement

- Type: Select Measurement

- Selections: Input, Function, Output, PID
- Description: Measurement to display on the line

Max Age

- Type: Decimal
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement Label

- Type: Text
- Description: Set to overwrite the default measurement label

Measurement Decimal

- Type: Integer
- Default Value: 1
- Description: The number of digits after the decimal

Text

- Type: Text
- Default Value: Text
- Description: Text to display

Display Unit

- Type: Boolean
- Default Value: True
- Description: Display the measurement unit (if available)

Display: SSD1306 OLED 128x32 [4 Lines] (I2C)

- Dependencies: libjpeg-dev, Pillow, pyusb, Adafruit-extended-bus, adafruit-circuitpython-framebuf, adafruit-circuitpython-ssd1306

This Function outputs to a 128x32 SSD1306 OLED display via I2C. This display Function will show 4 lines at a time, so channels are added in sets of 4 when Number of Line Sets is modified. Every Period, the LCD will refresh and display the next set of lines. Therefore, the first set of lines that are displayed are channels 0 - 3, then 4 - 7, and so on. After all channels have been displayed, it will cycle back to the beginning.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 10
- Description: The duration (seconds) between measurements or actions

I2C Address

- Type: Text
- Default Value: 0x3c
- Description: The I2C address of the device

I2C Bus

- Type: Integer
- Default Value: 1

- Description: The I2C bus the device is connected to

Number of Line Sets

- Type: Integer
- Default Value: 1
- Description: How many sets of lines to cycle on the LCD

Reset Pin

- Type: Integer
- Default Value: 17
- Description: The pin (BCM numbering) connected to RST of the display

Characters Per Line

- Type: Integer
- Default Value: 21
- Description: The maximum number of characters to display per line

Use Non-Default Font

- Type: Boolean
- Description: Don't use the default font. Enable to specify the path to a font to use.

Non-Default Font Path

- Type: Text
- Default Value: /usr/share/fonts/truetype/dejavu//DejaVuSans.ttf
- Description: The path to the non-default font to use

Font Size (pt)

- Type: Integer
- Default Value: 10
- Description: The size of the font, in points

CHANNEL OPTIONS

Line Display Type

- Type: Select
- Description: What to display on the line

Measurement

- Type: Select Measurement
- Selections: Input, Function, Output, PID
- Description: Measurement to display on the line

Max Age

- Type: Decimal
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement Label

- Type: Text
- Description: Set to overwrite the default measurement label

Measurement Decimal

- Type: Integer
- Default Value: 1
- Description: The number of digits after the decimal

Text

- Type: Text
- Default Value: Text
- Description: Text to display

Display Unit

- Type: Boolean
- Default Value: True
- Description: Display the measurement unit (if available)

Display: SSD1306 OLED 128x32 [4 Lines] (SPI)

- Dependencies: libjpeg-dev, Pillow, pyusb, Adafruit-GPIO, Adafruit-extended-bus, adafruit-circuitpython-framebuf, adafruit-circuitpython-ssd1306

This Function outputs to a 128x32 SSD1306 OLED display via SPI. This display Function will show 4 lines at a time, so channels are added in sets of 4 when Number of Line Sets is modified. Every Period, the LCD will refresh and display the next set of lines. Therefore, the first set of lines that are displayed are channels 0 - 3, then 4 - 7, and so on. After all channels have been displayed, it will cycle back to the beginning.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 10
- Description: The duration (seconds) between measurements or actions

Number of Line Sets

- Type: Integer
- Default Value: 1
- Description: How many sets of lines to cycle on the LCD

SPI Device

- Type: Integer
- Description: The SPI device

SPI Bus

- Type: Integer
- Description: The SPI bus

DC Pin

- Type: Integer
- Default Value: 16
- Description: The pin (BCM numbering) connected to DC of the display

Reset Pin

- Type: Integer

- Default Value: 19
- Description: The pin (BCM numbering) connected to RST of the display

CS Pin

- Type: Integer
- Default Value: 17
- Description: The pin (BCM numbering) connected to CS of the display

Characters Per Line

- Type: Integer
- Default Value: 21
- Description: The maximum number of characters to display per line

Use Non-Default Font

- Type: Boolean
- Description: Don't use the default font. Enable to specify the path to a font to use.

Non-Default Font Path

- Type: Text
- Default Value: /usr/share/fonts/truetype/dejavu//DejaVuSans.ttf
- Description: The path to the non-default font to use

Font Size (pt)

- Type: Integer
- Default Value: 10
- Description: The size of the font, in points

Display Unit

- Type: Boolean
- Default Value: True
- Description: Display the measurement unit (if available)

CHANNEL OPTIONS

Line Display Type

- Type: Select
- Description: What to display on the line

Measurement

- Type: Select Measurement
- Selections: Input, Function, Output, PID
- Description: Measurement to display on the line

Max Age

- Type: Decimal
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement Label

- Type: Text

- Description: Set to overwrite the default measurement label

Measurement Decimal

- Type: Integer
- Default Value: 1
- Description: The number of digits after the decimal

Text

- Type: Text
- Default Value: Text
- Description: Text to display

Display Unit

- Type: Boolean
- Default Value: True
- Description: Display the measurement unit (if available)

Display: SSD1306 OLED 128x64 [4 Lines] (I2C)

- Dependencies: libjpeg-dev, Pillow, pyusb, Adafruit-extended-bus, adafruit-circuitpython-framebuf, adafruit-circuitpython-ssd1306

This Function outputs to a 128x64 SSD1306 OLED display via I2C. This display Function will show 4 lines at a time, so channels are added in sets of 4 when Number of Line Sets is modified. Every Period, the LCD will refresh and display the next set of lines. Therefore, the first set of lines that are displayed are channels 0 - 3, then 4 - 7, and so on. After all channels have been displayed, it will cycle back to the beginning.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 10
- Description: The duration (seconds) between measurements or actions

I2C Address

- Type: Text
- Default Value: 0x3c
- Description: The I2C address of the device

I2C Bus

- Type: Integer
- Default Value: 1
- Description: The I2C bus the device is connected to

Number of Line Sets

- Type: Integer
- Default Value: 1
- Description: How many sets of lines to cycle on the LCD

Reset Pin

- Type: Integer
- Default Value: 17

- Description: The pin (BCM numbering) connected to RST of the display

Characters Per Line

- Type: Integer
- Default Value: 17
- Description: The maximum number of characters to display per line

Use Non-Default Font

- Type: Boolean
- Description: Don't use the default font. Enable to specify the path to a font to use.

Non-Default Font Path

- Type: Text
- Default Value: /usr/share/fonts/truetype/dejavu//DejaVuSans.ttf
- Description: The path to the non-default font to use

Font Size (pt)

- Type: Integer
- Default Value: 12
- Description: The size of the font, in points

CHANNEL OPTIONS

Line Display Type

- Type: Select
- Description: What to display on the line

Measurement

- Type: Select Measurement
- Selections: Input, Function, Output, PID
- Description: Measurement to display on the line

Max Age

- Type: Decimal
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement Label

- Type: Text
- Description: Set to overwrite the default measurement label

Measurement Decimal

- Type: Integer
- Default Value: 1
- Description: The number of digits after the decimal

Text

- Type: Text
- Default Value: Text
- Description: Text to display

Display Unit

- Type: Boolean
- Default Value: True
- Description: Display the measurement unit (if available)

Display: SSD1306 OLED 128x64 [4 Lines] (SPI)

- Dependencies: libjpeg-dev, Pillow, pyusb, Adafruit-GPIO, Adafruit-extended-bus, adafruit-circuitpython-framebuf, adafruit-circuitpython-ssd1306

This Function outputs to a 128x64 SSD1306 OLED display via SPI. This display Function will show 4 lines at a time, so channels are added in sets of 4 when Number of Line Sets is modified. Every Period, the LCD will refresh and display the next set of lines. Therefore, the first set of lines that are displayed are channels 0 - 3, then 4 - 7, and so on. After all channels have been displayed, it will cycle back to the beginning.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 10
- Description: The duration (seconds) between measurements or actions

Number of Line Sets

- Type: Integer
- Default Value: 1
- Description: How many sets of lines to cycle on the LCD

SPI Device

- Type: Integer
- Description: The SPI device

SPI Bus

- Type: Integer
- Description: The SPI bus

DC Pin

- Type: Integer
- Default Value: 16
- Description: The pin (BCM numbering) connected to DC of the display

Reset Pin

- Type: Integer
- Default Value: 19
- Description: The pin (BCM numbering) connected to RST of the display

CS Pin

- Type: Integer
- Default Value: 17
- Description: The pin (BCM numbering) connected to CS of the display

Characters Per Line

- Type: Integer

- Default Value: 17
- Description: The maximum number of characters to display per line

Use Non-Default Font

- Type: Boolean
- Description: Don't use the default font. Enable to specify the path to a font to use.

Non-Default Font Path

- Type: Text
- Default Value: /usr/share/fonts/truetype/dejavu//DejaVuSans.ttf
- Description: The path to the non-default font to use

Font Size (pt)

- Type: Integer
- Default Value: 12
- Description: The size of the font, in points

CHANNEL OPTIONS

Line Display Type

- Type: Select
- Description: What to display on the line

Measurement

- Type: Select Measurement
- Selections: Input, Function, Output, PID
- Description: Measurement to display on the line

Max Age

- Type: Decimal
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement Label

- Type: Text
- Description: Set to overwrite the default measurement label

Measurement Decimal

- Type: Integer
- Default Value: 1
- Description: The number of digits after the decimal

Text

- Type: Text
- Default Value: Text
- Description: Text to display

Display Unit

- Type: Boolean
- Default Value: True

- Description: Display the measurement unit (if available)

Display: SSD1306 OLED 128x64 [8 Lines] (I2C)

- Dependencies: libjpeg-dev, Pillow, pyusb, Adafruit-extended-bus, adafruit-circuitpython-framebuf, adafruit-circuitpython-ssd1306

This Function outputs to a 128x64 SSD1306 OLED display via I2C. This display Function will show 8 lines at a time, so channels are added in sets of 8 when Number of Line Sets is modified. Every Period, the LCD will refresh and display the next set of lines. Therefore, the first set of lines that are displayed are channels 0 - 7, then 8 - 15, and so on. After all channels have been displayed, it will cycle back to the beginning.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 10
- Description: The duration (seconds) between measurements or actions

I2C Address

- Type: Text
- Default Value: 0x3c
- Description: The I2C address of the device

I2C Bus

- Type: Integer
- Default Value: 1
- Description: The I2C bus the device is connected to

Number of Line Sets

- Type: Integer
- Default Value: 1
- Description: How many sets of lines to cycle on the LCD

Reset Pin

- Type: Integer
- Default Value: 17
- Description: The pin (BCM numbering) connected to RST of the display

Characters Per Line

- Type: Integer
- Default Value: 21
- Description: The maximum number of characters to display per line

Use Non-Default Font

- Type: Boolean
- Description: Don't use the default font. Enable to specify the path to a font to use.

Non-Default Font Path

- Type: Text
- Default Value: /usr/share/fonts/truetype/dejavu//DejaVuSans.ttf
- Description: The path to the non-default font to use

Font Size (pt)

- Type: Integer
- Default Value: 10
- Description: The size of the font, in points

CHANNEL OPTIONS

Line Display Type

- Type: Select
- Description: What to display on the line

Measurement

- Type: Select Measurement
- Selections: Input, Function, Output, PID
- Description: Measurement to display on the line

Max Age

- Type: Decimal
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement Label

- Type: Text
- Description: Set to overwrite the default measurement label

Measurement Decimal

- Type: Integer
- Default Value: 1
- Description: The number of digits after the decimal

Text

- Type: Text
- Default Value: Text
- Description: Text to display

Display Unit

- Type: Boolean
- Default Value: True
- Description: Display the measurement unit (if available)

Display: SSD1306 OLED 128x64 [8 Lines] (SPI)

- Dependencies: libjpeg-dev, Pillow, pyusb, Adafruit-GPIO, Adafruit-extended-bus, adafruit-circuitpython-framebuf, adafruit-circuitpython-ssd1306

This Function outputs to a 128x64 SSD1306 OLED display via SPI. This display Function will show 8 lines at a time, so channels are added in sets of 8 when Number of Line Sets is modified. Every Period, the LCD will refresh and display the next set of lines. Therefore, the first set of lines that are displayed are channels 0 - 7, then 8 - 15, and so on. After all channels have been displayed, it will cycle back to the beginning.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 10
- Description: The duration (seconds) between measurements or actions

Number of Line Sets

- Type: Integer
- Default Value: 1
- Description: How many sets of lines to cycle on the LCD

SPI Device

- Type: Integer
- Description: The SPI device

SPI Bus

- Type: Integer
- Description: The SPI bus

DC Pin

- Type: Integer
- Default Value: 16
- Description: The pin (BCM numbering) connected to DC of the display

Reset Pin

- Type: Integer
- Default Value: 19
- Description: The pin (BCM numbering) connected to RST of the display

CS Pin

- Type: Integer
- Default Value: 17
- Description: The pin (BCM numbering) connected to CS of the display

Characters Per Line

- Type: Integer
- Default Value: 21
- Description: The maximum number of characters to display per line

Use Non-Default Font

- Type: Boolean
- Description: Don't use the default font. Enable to specify the path to a font to use.

Non-Default Font Path

- Type: Text
- Default Value: /usr/share/fonts/truetype/dejavu//DejaVuSans.ttf
- Description: The path to the non-default font to use

Font Size (pt)

- Type: Integer
- Default Value: 10
- Description: The size of the font, in points

CHANNEL OPTIONS

Line Display Type

- Type: Select
- Description: What to display on the line

Measurement

- Type: Select Measurement
- Selections: Input, Function, Output, PID
- Description: Measurement to display on the line

Max Age

- Type: Decimal
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement Label

- Type: Text
- Description: Set to overwrite the default measurement label

Measurement Decimal

- Type: Integer
- Default Value: 1
- Description: The number of digits after the decimal

Text

- Type: Text
- Default Value: Text
- Description: Text to display

Display Unit

- Type: Boolean
- Default Value: True
- Description: Display the measurement unit (if available)

Display: SSD1309 OLED 128x64 [8 Lines] (I2C)

- Dependencies: pyusb, luma.oled, Pillow, libjpeg-dev, zlib1g-dev, libfreetype6-dev, liblcms2-dev, libopenjp2-7, libtiff5

This Function outputs to a 128x64 SSD1309 OLED display via I2C. This display Function will show 8 lines at a time, so channels are added in sets of 8 when Number of Line Sets is modified. Every Period, the LCD will refresh and display the next set of lines. Therefore, the first set of lines that are displayed are channels 0 - 7, then 8 - 15, and so on. After all channels have been displayed, it will cycle back to the beginning.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 10
- Description: The duration (seconds) between measurements or actions

I2C Address

- Type: Text
- Default Value: 0x3c
- Description: The I2C address of the device

I2C Bus

- Type: Integer
- Default Value: 1
- Description: The I2C bus the device is connected to

Number of Line Sets

- Type: Integer
- Default Value: 1
- Description: How many sets of lines to cycle on the LCD

Reset Pin

- Type: Integer
- Default Value: 17
- Description: The pin (BCM numbering) connected to RST of the display

CHANNEL OPTIONS

Line Display Type

- Type: Select
- Description: What to display on the line

Measurement

- Type: Select Measurement
- Selections: Input, Function, Output, PID
- Description: Measurement to display on the line

Max Age

- Type: Decimal
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement Label

- Type: Text
- Description: Set to overwrite the default measurement label

Measurement Decimal

- Type: Integer
- Default Value: 1
- Description: The number of digits after the decimal

Text

- Type: Text
- Default Value: Text
- Description: Text to display

Display Unit

- Type: Boolean
- Default Value: True
- Description: Display the measurement unit (if available)

Equation (Multi-Measure)

This function acquires two measurements and uses them within a user-set equation and stores the resulting value as the selected measurement and unit.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 60
- Description: The duration (seconds) between measurements or actions

Measurement A

- Type: Select Measurement
- Selections: Input, Function
- Description: Measurement to replace a

Measurement A Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement B

- Type: Select Measurement
- Selections: Input, Function
- Description: Measurement to replace b

Measurement B Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Equation

- Type: Text
- Default Value: a*(2+b)
- Description: Equation using measurements a and b

Equation (Single-Measure)

This function acquires a measurement and uses it within a user-set equation and stores the resulting value as the selected measurement and unit.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 60
- Description: The duration (seconds) between measurements or actions

Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Measurement to replace "x" in the equation

Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Equation

- Type: Text
- Default Value: x*5+2
- Description: Equation using the measurement

Humidity (Wet/Dry-Bulb)

This function calculates the humidity based on wet and dry bulb temperature measurements.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Default Value: 60
- Description: The duration (seconds) between measurements or actions

Start Offset

- Type: Integer
- Default Value: 10
- Description: The duration (seconds) to wait before the first operation

Dry Bulb Temperature

- Type: Select Measurement
- Selections: Input, Function
- Description: Dry Bulb temperature measurement

Dry Bulb Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Wet Bulb Temperature

- Type: Select Measurement
- Selections: Input, Function
- Description: Wet Bulb temperature measurement

Wet Bulb Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Pressure

- Type: Select Measurement
- Selections: Input, Function
- Description: Pressure measurement

Pressure Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

PID Autotune

This function will attempt to perform a PID controller autotune. That is, an output will be powered and the response measured from a sensor several times to calculate the P, I, and D gains. Updates about the operation will be sent to the Daemon log. If the autotune successfully completes, a summary will be sent to the Daemon log as well. Currently, only raising a Measurement is supported, but lowering should be possible with some modification to the function controller code. It is recommended to create a graph on a dashboard with the Measurement and Output to monitor that the Output is successfully raising the Measurement beyond the Setpoint. Note: Autotune is an experimental feature, it is not well-developed, and it has a high likelihood of failing to generate PID gains. Do not rely on it for accurately tuning your PID controller.

OPTIONS

Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Select a measurement the selected output will affect

Output

- Type: Select Device, Measurement, and Channel
- Selections: Output
- Description: Select an output to modulate that will affect the measurement

Period

- Type: Integer
- Default Value: 30
- Description: The period between powering the output

Setpoint

- Type: Decimal
- Default Value: 50

- Description: A value sufficiently far from the current measured value that the output is capable of pushing the measurement toward

Noise Band

- Type: Decimal
- Default Value: 0.5
- Description: The amount above the setpoint the measurement must reach

Outstep

- Type: Decimal
- Default Value: 10
- Description: How many seconds the output will turn on every Period

Currently, only autotuning to raise a condition (measurement) is supported.

Direction

- Type: Select
- Options: [**Raise**] (Default in **bold**)
- Description: The direction the Output will push the Measurement

Redundancy

This function stores the first available measurement. This is useful if you have multiple sensors that you want to serve as backups in case one stops working, you can set them up in the order of importance. This function will check if a measurement exits, starting with the first measurement. If it doesn't, the next is checked, until a measurement is found. Once a measurement is found, it is stored in the database with the user-set measurement and unit. The output of this function can be used as an input throughout Mycodo. If you need more than 3 measurements to be checked, you can string multiple Redundancy Functions by creating a second Function and setting the first Function's output as the second Function's input.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 60
- Description: The duration (seconds) between measurements or actions

Measurement A

- Type: Select Measurement
- Selections: Input, Function
- Description: Measurement to replace a

Measurement A Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement B

- Type: Select Measurement
- Selections: Input, Function
- Description: Measurement to replace b

Measurement B Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement C

- Type: Select Measurement
- Selections: Input, Function
- Description: Measurement to replace C

Measurement C Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Spacer

A spacer to organize Functions.

OPTIONS

Color

- Type: Text
- Default Value: #000000
- Description: The color of the name text

Statistics (Last, Multiple)

This function acquires multiple measurements, calculates statistics, and stores the resulting values as the selected unit.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Default Value: 60
- Description: The duration (seconds) between measurements or actions

Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement

- Description: Measurements to perform statistics on

Halt on Missing Measurement

- Type: Boolean
- Description: Don't calculate statistics if >= 1 measurement is not found within Max Age

Statistics (Past, Single)

This function acquires multiple values from a single measurement, calculates statistics, and stores the resulting values as the selected unit.

OPTIONS

Measurements Enabled

- Type: Multi-Select
- Description: The measurements to record

Period (seconds)

- Type: Decimal
- Default Value: 60
- Description: The duration (seconds) between measurements or actions

Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Measurement to perform statistics on

Sum (Last, Multiple)

This function acquires the last measurement of those that are selected, sums them, then stores the resulting value as the selected measurement and unit.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 60
- Description: The duration (seconds) between measurements or actions

Start Offset

- Type: Integer
- Default Value: 10
- Description: The duration (seconds) to wait before the first operation

Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement

- Description: Measurement to replace "x" in the equation

Sum (Past, Single)

This function acquires the past measurements (within Max Age) for the selected measurement, sums them, then stores the resulting value as the selected measurement and unit.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 60
- Description: The duration (seconds) between measurements or actions

Start Offset

- Type: Integer
- Default Value: 10
- Description: The duration (seconds) to wait before the first operation

Measurement

- Type: Select Measurement
- Selections: Input, Function
- Description: Measurement to replace "x" in the equation

Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Vapor Pressure Deficit

This function calculates the vapor pressure deficit based on leaf temperature and humidity.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 60
- Description: The duration (seconds) between measurements or actions

Start Offset

- Type: Integer
- Default Value: 10
- Description: The duration (seconds) to wait before the first operation

Temperature

- Type: Select Measurement
- Selections: Input, Function
- Description: Temperature measurement

Temperature Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Humidity

- Type: Select Measurement
- Selections: Input, Function
- Description: Humidity measurement

Humidity Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Verification

This function acquires 2 measurements, calculates the difference, and if the difference is not larger than the set threshold, the Measurement A value is stored. This enables verifying one sensor's measurement with another sensor's measurement. Only when they are both in agreement is a measurement stored. This stored measurement can be used in functions such as Conditional Statements that will notify the user if no measurement is avilable to indicate there may be an issue with a sensor.

OPTIONS

Period (seconds)

- Type: Decimal
- Default Value: 60
- Description: The duration (seconds) between measurements or actions

Measurement A

- Type: Select Measurement
- Selections: Input, Function
- Description: Measurement A

Measurement A Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Measurement B

- Type: Select Measurement
- Selections: Input, Function
- Description: Measurement B

Measurement A Max Age

- Type: Integer
- Default Value: 360
- Description: The maximum age (seconds) of the measurement to use

Maximum Difference

- Type: Decimal
- Default Value: 10.0
- Description: The maximum allowed difference between the measurements

5.5 Supported Actions

Actions allow certain Functions to influence other parts of Mycodo and the computer system.

Supported Actions are listed below.

5.5.1 Built-In Actions (System)

Actions: Pause

- Manufacturer: Mycodo
- Works with: Functions

Set a delay between executing Actions when self.run_all_actions() is used.

Usage: Executing **self.run_action("{ACTION_ID}")** will create a pause for the set duration. When **self.run_all_actions()** is executed, this will add a pause in the sequential execution of all actions.

OPTIONS

Duration (seconds)

- Type: Decimal
- Description: The duration to pause

Camera: Capture Photo

- Manufacturer: Mycodo
- Works with: Functions

Capture a photo with the selected Camera.

Usage: Executing **self.run_action("{ACTION_ID}")** will capture a photo with the selected Camera. Executing **self.run_action("{ACTION_ID}", value={"camera_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b"})** will capture a photo with the Camera with the specified ID.

OPTIONS

Camera

- Type: Select Device
- Description: Select the Camera to take a photo

Camera: Time-lapse: Pause

- Manufacturer: Mycodo
- Works with: Functions

Pause a camera time-lapse

Usage: Executing **self.run_action("{ACTION_ID}")** will pause the selected Camera time-lapse. Executing **self.run_action("{ACTION_ID}", value={"camera_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b"})** will pause the Camera time-lapse with the specified ID.

OPTIONS

Camera

- Type: Select Device
- Description: Select the Camera to pause the time-lapse

Camera: Time-lapse: Resume

- Manufacturer: Mycodo
- Works with: Functions

Resume a camera time-lapse

Usage: Executing **self.run_action("{ACTION_ID}")** will resume the selected Camera time-lapse. Executing **self.run_action("{ACTION_ID}", value={"camera_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b"})** will resume the Camera time-lapse with the specified ID.

OPTIONS

Camera

- Type: Select Device
- Description: Select the Camera to resume the time-lapse

Controller: Activate

- Manufacturer: Mycodo
- Works with: Functions

Activate a controller.

Usage: Executing **self.run_action("{ACTION_ID}")** will activate the selected Controller. Executing **self.run_action("{ACTION_ID}", value={"controller_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b"})** will activate the controller with the specified ID.

OPTIONS

Controller

- Type: Select Device
- Description: Select the controller to activate

Controller: Deactivate

- Manufacturer: Mycodo
- Works with: Functions

Deactivate a controller.

Usage: Executing **self.run_action("{ACTION_ID}")** will deactivate the selected Controller. Executing **self.run_action("{ACTION_ID}", value={"controller_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b"})** will deactivate the controller with the specified ID.

OPTIONS

Controller

- Type: Select Device
- Description: Select the controller to deactivate

Create: Note

- Manufacturer: Mycodo
- Works with: Functions

Create a note with the selected Tag.

Usage: Executing **self.run_action("{ACTION_ID}")** will create a note with the selected tag and note. Executing **self.run_action("{ACTION_ID}", value={"tags": ["tag1", "tag2"], "name": "My Note", "note": "this is a message"})** will execute the action with the specified list of tag(s) and note. If using only one tag, make it the only element of the list (e.g. ["tag1"]). If note is not specified, then the action message will be used as the note.

OPTIONS

Tags

- Description: Select one or more tags

Name

- Type: Text
- Default Value: Name
- Description: The name of the note

Note

- Type: Text
- Default Value: Note
- Description: The body of the note

Execute Command: Shell

- Manufacturer: Mycodo
- Works with: Functions

Execute a Linux bash shell command.

Usage: Executing **self.run_action("{ACTION_ID}")** will execute the bash command.Executing **self.run_action("{ACTION_ID}", value={"user": "mycodo", "command": "/home/pi/my_script.sh on"})** will execute the action with the specified command and user.

OPTIONS

User

- Type: Text
- Default Value: mycodo
- Description: The user to execute the command

Command

- Type: Text
- Default Value: /home/pi/my_script.sh on
- Description: Command to execute

Flow Meter: Clear Total Volume

- Manufacturer: Mycodo
- Works with: Functions

Clear the total volume saved for a flow meter Input. The Input must have the Clear Total Volume option.

Usage: Executing **self.run_action("{ACTION_ID}")** will clear the total volume for the selected flow meter Input. Executing **self.run_action("{ACTION_ID}", value={"input_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b"})** will clear the total volume for the flow meter Input with the specified ID.

OPTIONS

Controller

- Type: Select Device
- Description: Select the flow meter Input

Input: Force Measurements

- Manufacturer: Mycodo
- Works with: Functions

Force measurements to be conducted for an input

Usage: Executing **self.run_action("{ACTION_ID}")** will force acquiring measurements for the selected Input. Executing **self.run_action("{ACTION_ID}", value={"input_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b"})** will force acquiring measurements for the Input with the specified ID.

OPTIONS

Input

- Type: Select Device
- Description: Select an Input

MQTT: Publish

- Manufacturer: Mycodo
- Works with: Functions
- Dependencies: paho-mqtt

Publish to an MQTT server.

Usage: Executing **self.run_action("{ACTION_ID}")** will publish the saved payload text options to the MQTT server. Executing **self.run_action("{ACTION_ID}", value={"payload": 42})** will publish the specified payload (any type) to the MQTT server. You can also specify the topic (e.g. value={"topic": "my_topic", "payload": 42}). Warning: If using multiple MQTT Inputs or Functions, ensure the Client IDs are unique.

OPTIONS

Hostname

- Type: Text
- Default Value: localhost
- Description: The hostname of the MQTT server

Port

- Type: Integer
- Default Value: 1883
- Description: The port of the MQTT server

Topic

- Type: Text
- Default Value: paho/test/single
- Description: The topic to publish with

Payload

- Type: Text

- Description: The payload to publish

Keep Alive

- Type: Integer
- Default Value: 60
- Description: The keepalive timeout value for the client. Set to 0 to disable.

Client ID

- Type: Text
- Default Value: client_796v1NR4
- Description: Unique client ID for connecting to the MQTT server

Use Login

- Type: Boolean
- Description: Send login credentials

Username

- Type: Text
- Default Value: user
- Description: Username for connecting to the server

Password

- Type: Text
- Description: Password for connecting to the server

MQTT: Publish: Measurement

- Manufacturer: Mycodo
- Works with: Inputs
- Dependencies: paho-mqtt

Publish an Input measurement to an MQTT server.

OPTIONS

Measurement

- Description: Select the measurement to send as the payload

Hostname

- Type: Text
- Default Value: localhost
- Description: The hostname of the MQTT server

Port

- Type: Integer
- Default Value: 1883
- Description: The port of the MQTT server

Topic

- Type: Text
- Default Value: paho/test/single

- Description: The topic to publish with

Keep Alive

- Type: Integer
- Default Value: 60
- Description: The keepalive timeout value for the client. Set to 0 to disable.

Client ID

- Type: Text
- Default Value: client_YeURfmKy
- Description: Unique client ID for connecting to the MQTT server

Use Login

- Type: Boolean
- Description: Send login credentials

Username

- Type: Text
- Default Value: user
- Description: Username for connecting to the server

Password

- Type: Text
- Description: Password for connecting to the server.

Output: Duty Cycle

- Manufacturer: Mycodo
- Works with: Functions

Set a PWM Output to set a duty cycle.

Usage: Executing **self.run_action("{ACTION_ID}")** will set the PWM output duty cycle. Executing **self.run_action("{ACTION_ID}", value={"output_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b", "channel": 0, "duty_cycle": 42})** will set the duty cycle of the PWM output with the specified ID and channel.

OPTIONS

Output

- Type: Select Channel
- Selections: Output_Channels
- Description: Select an output to control

Duty Cycle

- Type: Decimal
- Description: Duty cycle for the PWM (percent, 0.0 - 100.0)

Output: On/Off/Duration

- Manufacturer: Mycodo
- Works with: Functions

Turn an On/Off Output On, Off, or On for a duration.

Usage: Executing **self.run_action("{ACTION_ID}")** will actuate an output. Executing **self.run_action("{ACTION_ID}", value={"output_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b", "channel": 0, "state": "on", "duration": 300})** will set the state of the output with the specified ID and channel. If state is on and a duration is set, the output will turn off after the duration.

OPTIONS

Output

- Type: Select Channel
- Selections: Output_Channels
- Description: Select an output to control

State

- Type: Select
- Description: Turn the output on or off

Duration (seconds)

- Type: Decimal
- Description: If On, you can set a duration to turn the output on. 0 stays on.

Output: Ramp Duty Cycle

- Manufacturer: Mycodo
- Works with: Functions

Ramp a PWM Output from one duty cycle to another duty cycle over a period of time.

Usage: Executing **self.run_action("{ACTION_ID}")** will ramp the PWM output duty cycle according to the settings. Executing **self.run_action("{ACTION_ID}", value={"output_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b", "channel": 0, "start": 42, "end": 62, "increment": 1.0, "duration": 600})** will ramp the duty cycle of the PWM output with the specified ID and channel.

OPTIONS

Output

- Type: Select Channel
- Selections: Output_Channels
- Description: Select an output to control

Duty Cycle: Start

- Type: Decimal
- Description: Duty cycle for the PWM (percent, 0.0 - 100.0)

Duty Cycle: End

- Type: Decimal
- Default Value: 50.0
- Description: Duty cycle for the PWM (percent, 0.0 - 100.0)

Increment (Duty Cycle)

- Type: Decimal
- Default Value: 1.0
- Description: How much to change the duty cycle every Duration

Duration (seconds)

- Type: Decimal
- Description: How long to ramp from start to finish.

Output: Value

- Manufacturer: Mycodo
- Works with: Functions

Send a value to the Output.

Usage: Executing **self.run_action("{ACTION_ID}")** will actuate a value output. Executing **self.run_action("{ACTION_ID}", value={"output_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b", "channel": 0, "value": 42})** will send a value to the output with the specified ID and channel.

OPTIONS

Output

- Type: Select Channel
- Selections: Output_Channels
- Description: Select an output to control

Value

- Type: Decimal
- Description: The value to send to the output

Output: Volume

- Manufacturer: Mycodo
- Works with: Functions

Instruct the Output to dispense a volume.

Usage: Executing **self.run_action("{ACTION_ID}")** will actuate a volume output. Executing **self.run_action("{ACTION_ID}", value={"output_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b", "channel": 0, "volume": 42})** will send a volume to the output with the specified ID and channel.

OPTIONS

Output

- Type: Select Channel
- Selections: Output_Channels
- Description: Select an output to control

Volume

- Type: Decimal
- Description: The volume to send to the output

PID: Lower: Setpoint

- Manufacturer: Mycodo
- Works with: Functions

Lower the Setpoint of a PID.

Usage: Executing **self.run_action("{ACTION_ID}")** will lower the setpoint of the selected PID Controller. Executing **self.run_action("{ACTION_ID}", value={"pid_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b", "amount": 2})** will lower the setpoint of the PID with the specified ID.

OPTIONS

Controller

- Type: Select Device
- Description: Select the PID Controller to lower the setpoint of

Lower Setpoint

- Type: Decimal
- Description: The amount to lower the PID setpoint by

PID: Pause

- Manufacturer: Mycodo
- Works with: Functions

Pause a PID.

Usage: Executing **self.run_action("{ACTION_ID}")** will pause the selected PID Controller. Executing **self.run_action("{ACTION_ID}", value="959019d1-c1fa-41fe-a554-7be3366a9c5b")** will pause the PID Controller with the specified ID.

OPTIONS

Controller

- Type: Select Device
- Description: Select the PID Controller to pause

PID: Raise: Setpoint

- Manufacturer: Mycodo
- Works with: Functions

Raise the Setpoint of a PID.

Usage: Executing **self.run_action("{ACTION_ID}")** will raise the setpoint of the selected PID Controller. Executing **self.run_action("{ACTION_ID}", value={"pid_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b", "amount": 2})** will raise the setpoint of the PID with the specified ID.

OPTIONS

Controller

- Type: Select Device
- Description: Select the PID Controller to raise the setpoint of

Raise Setpoint

- Type: Decimal
- Description: The amount to raise the PID setpoint by

PID: Resume

- Manufacturer: Mycodo
- Works with: Functions

Resume a PID.

Usage: Executing **self.run_action("{ACTION_ID}")** will resume the selected PID Controller. Executing **self.run_action("{ACTION_ID}", value="959019d1-c1fa-41fe-a554-7be3366a9c5b")** will resume the PID Controller with the specified ID.

OPTIONS

Controller

- Type: Select Device
- Description: Select the PID Controller to resume

PID: Set Method

- Manufacturer: Mycodo
- Works with: Functions

Select a method to set the PID to use.

Usage: Executing **self.run_action("{ACTION_ID}")** will pause the selected PID Controller. Executing **self.run_action("{ACTION_ID}", value={"pid_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b", "method_id": "fe8b8f41-131b-448d-ba7b-00a044d24075"})** will set a method for the PID Controller with the specified IDs.

OPTIONS

Controller

- Type: Select Device
- Description: Select the PID Controller to apply the method

Method

- Type: Select Device
- Description: Select the Method to apply to the PID

PID: Set: Setpoint

- Manufacturer: Mycodo
- Works with: Functions

Set the Setpoint of a PID.

Usage: Executing **self.run_action("{ACTION_ID}")** will set the setpoint of the selected PID Controller. Executing **self.run_action("{ACTION_ID}", value={"setpoint": 42})** will set the setpoint of the PID Controller (e.g. 42). You can also specify the PID ID (e.g. value={"setpoint": 42, "pid_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b"})

OPTIONS

Controller

- Type: Select Device
- Description: Select the PID Controller to pause

Setpoint

- Type: Decimal
- Description: The setpoint to set the PID Controller

Send Email

- Manufacturer: Mycodo

- Works with: Functions

Send an email.

Usage: Executing **self.run_action("{ACTION_ID}")** will email the specified recipient(s) using the SMTP credentials in the system configuration. Separate multiple recipients with commas. The body of the email will be the self-generated message. Executing **self.run_action("{ACTION_ID}", value={"email_address": ["email1@email.com", "email2@email.com"], "message": "My message"})** will send an email to the specified recipient(s) with the specified message.

OPTIONS

E-Mail Address

- Type: Text
- Default Value: email@domain.com
- Description: E-mail recipient(s) (separate multiple addresses with commas)

Send Email with Photo

- Manufacturer: Mycodo
- Works with: Functions

Take a photo and send an email with it attached.

Usage: Executing **self.run_action("{ACTION_ID}")** will take a photo and email it to the specified recipient(s) using the SMTP credentials in the system configuration. Separate multiple recipients with commas. The body of the email will be the self-generated message. Executing **self.run_action("{ACTION_ID}", value={"camera_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b", "email_address": ["email1@email.com", "email2@email.com"], "message": "My message"})** will capture a photo using the camera with the specified ID and send an email to the specified email(s) with message and attached photo.

OPTIONS

Camera

- Type: Select Device
- Description: Select the Camera to take a photo with

E-Mail Address

- Type: Text
- Default Value: email@domain.com
- Description: E-mail recipient(s). Separate multiple with commas.

System: Restart

- Manufacturer: Mycodo
- Works with: Functions

Restart the System

Usage: Executing **self.run_action("{ACTION_ID}")** will restart the system in 10 seconds.

System: Shutdown

- Manufacturer: Mycodo
- Works with: Functions

Shutdown the System

Usage: Executing **self.run_action("{ACTION_ID}")** will shut down the system in 10 seconds.

Webhook

- Manufacturer: Mycodo
- Works with: Functions

Emits a HTTP request when triggered. The first line contains a HTTP verb (GET, POST, PUT, ...) followed by a space and the URL to call. Subsequent lines are optional "name: value"-header parameters. After a blank line, the body payload to be sent follows. {{{message}}} is a placeholder that gets replaced by the message, {{{quoted_message}}} is the message in an URL safe encoding.

Usage: Executing **self.run_action("{ACTION_ID}")** will run the Action.

OPTIONS

Webhook Request

- Description: HTTP request to execute

5.5.2 Built-In Actions (Devices)

Display: Backlight: Color

- Manufacturer: Display
- Works with: Functions

Set the display backlight color

Usage: Executing **self.run_action("{ACTION_ID}")** will change the backlight color on the selected display. Executing **self.run_action("{ACTION_ID}", value={"display_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b", "color": "255,0,0"})** will change the backlight color on the controller with the specified ID and color.

OPTIONS

Display

- Type: Select Device
- Description: Select the display to set the backlight color

Color (RGB)

- Type: Text
- Default Value: 255,0,0
- Description: Color as R,G,B values (e.g. "255,0,0" without quotes)

Display: Backlight: Off

- Manufacturer: Display
- Works with: Functions

Turn display backlight off

Usage: Executing **self.run_action("{ACTION_ID}")** will turn the backlight off for the selected display. Executing **self.run_action("{ACTION_ID}", value={"display_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b"})** will turn the backlight off for the controller with the specified ID.

OPTIONS

Display

- Type: Select Device

- Description: Select the display to turn the backlight off

Display: Backlight: On

- Manufacturer: Display
- Works with: Functions

Turn display backlight on

Usage: Executing **self.run_action("{ACTION_ID}")** will turn the backlight on for the selected display. Executing **self.run_action("{ACTION_ID}", value={"display_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b"})** will turn the backlight on for the controller with the specified ID.

OPTIONS

Display

- Type: Select Device
- Description: Select the display to turn the backlight on

Display: Flashing: Off

- Manufacturer: Display
- Works with: Functions

Turn display flashing off

Usage: Executing **self.run_action("{ACTION_ID}")** will stop the backlight flashing on the selected display. Executing **self.run_action("{ACTION_ID}", value={"display_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b"})** will stop the backlight flashing on the controller with the specified ID.

OPTIONS

Display

- Type: Select Device
- Description: Select the display to stop flashing the backlight

Display: Flashing: On

- Manufacturer: Display
- Works with: Functions

Turn display flashing on

Usage: Executing **self.run_action("{ACTION_ID}")** will start the backlight flashing on the selected display. Executing **self.run_action("{ACTION_ID}", value={"display_id": "959019d1-c1fa-41fe-a554-7be3366a9c5b"})** will start the backlight flashing on the controller with the specified ID.

OPTIONS

Display

- Type: Select Device
- Description: Select the display to start flashing the backlight

5.6 Supported Widgets

Supported Widget devices are listed below.

5.6.1 Built-In Widgets

Camera

Displays a camera image or stream.

Function Status

Displays the status of a Function (if supported).

Gauge (Angular) [Highcharts]

- Libraries: Highcharts
- Dependencies: unzip, highstock-9.1.2.js, highcharts-more-9.1.2.js

Displays an angular gauge. Be sure to set the Maximum option to the last Stop High value for the gauge to display properly.

Gauge (Solid) [Highcharts]

- Libraries: Highcharts
- Dependencies: unzip, highstock-9.1.2.js, highcharts-more-9.1.2.js, solid-gauge-9.1.2.js

Displays a solid gauge. Be sure to set the Maximum option to the last Stop value for the gauge to display properly.

Graph (Synchronous) [Highstock]

- Libraries: Highstock
- Dependencies: unzip, highstock-9.1.2.js, highcharts-more-9.1.2.js, data-9.1.2.js, exporting-9.1.2.js, export-data-9.1.2.js, offline-exporting-9.1.2.js

Displays a synchronous graph (all data is downloaded for the selected period on the x-axis).

Indicator

Displays a red or green circular image based on a measurement value. Useful for showing if an Output is on or off.

Measurement

Displays a measurement value and timestamp.

Output (PWM Slider)

Displays and allows control of a PWM output using a slider.

Output Control (Channel)

Displays and allows control of an output channel. All output options and measurements for the selected channel will be displayed. E.g. pumps will have seconds on and volume as measurements, and can be turned on for a duration (seconds) or amount (volume). If NO DATA or TOO OLD is displayed, the Max Age is not sufficiently long enough to find a current measurement.

PID Controller

Displays and allows control of a PID Controller.

Python Code

Executes Python code and displays the output within the widget.

Spacer

A simple widget to use as a spacer, which includes the ability to set text in its contents.

5.7 I2C Multiplexers

All devices that connected to the Raspberry Pi by the I2C bus need to have a unique address in order to communicate. Some inputs may have the same address (such as the AM2315), which prevents more than one from being connected at the same time. Others may provide the ability to change the address, however the address range may be limited, which limits by how many you can use at the same time. I2C multiplexers are extremely clever and useful in these scenarios because they allow multiple sensors with the same I2C address to be connected.

For instance, the TCA9548A/PCA9548A: I2C Multiplexer has 8 selectable addresses, so 8 multiplexers can be connected to one Raspberry Pi. Each multiplexer has 8 channels, allowing up to 8 devices/sensors with the same address to be connected to each multiplexer. 8 multiplexers x 8 channels = 64 devices/sensors with the same I2C address.

Multiplexers can be set up by loading a kernel driver to handle the communication, producing a new I2C bus device for each multiplexer channel. To enable the driver for the TCA9548A/PCA9548A, visit GPIO-pca9548 to get the code and latest install instructions. If successfully set up, there will be 8 new I2C buses on the `[Gear Icon] -> System Information` page.

The driver for the TCA9545A can be found at https://github.com/camrex/i2c-mux-pca9545a and other drivers are available elsewhere. See the manufacturer or user forums for details. Some multiplexers I've tested are below.

- TCA9548A/PCA9548A: I2C Multiplexer link (I2C): 8 selectable addresses, 8 channels
- TCA9545A: I2C Bus Multiplexer link (I2C): The linked Grove board creates 4 new I2C buses, each with their own selectable voltage, either 3.3 or 5.0 volts.

5.8 Analog-To-Digital Converters

An analog-to-digital converter (ADC) allows the measurement of an analog voltage.

> **Note**
>
> A voltage divider may be necessary to convert your source voltage to an acceptable range for the ADC.

- ADS1x15: Analog-to-digital converter link
- ADS1256: Analog-to-digital converter link
- MCP3008: Analog-to-digital converter link
- MCP342x: Analog-to-digital converter link

5.9 Interfaces

5.9.1 I2C Information

The I2C interface should be enabled with `raspi-config` or from the `[Gear Icon] -> Configure -> Raspberry Pi` page.

5.9.2 1-Wire Information

The 1-Wire interface should be enabled with `raspi-config` or from the `[Gear Icon] -> Configure -> Raspberry Pi` page.

5.9.3 UART Information

This documentation provides specific installation procedures for configuring UART with the Raspberry Pi version 1 or 2.

Because the UART is handled differently higher after the Raspberry Pi 2 (due to the addition of bluetooth), there are a different set of instructions. If installing Mycodo on a Raspberry Pi 3 or above, you only need to perform these steps to configure UART:

Run raspi-config

```
sudo raspi-config
```

Go to `Advanced Options -> Serial` and disable. Then edit `/boot/config.txt`

```
sudo nano /boot/config.txt
```

Find the line "enable_uart=0" and change it to "enable_uart=1", then reboot.

5.10 Dependencies

Page: `[Gear Icon] -> Dependencies`

The dependency page allows viewing of dependency information and the ability to initiate their installation. This is not something you will need to normally do, as dependencies are installed on an as-needed basis. If an Input, Output, Function, or other device you're adding has unmet dependencies, you will be prompted to install them when you attempt to install that device.

5.11 Device Notes

This information may not be current, so always reference and follow manufacturer recommendations for operating their devices.

5.11.1 Edge Detection

The detection of a changing signal, for instance a simple switch completing a circuit, requires the use of edge detection. By detecting a rising edge (LOW to HIGH), a falling edge (HIGH to LOW), or both, actions or events can be triggered. The GPIO chosen to detect the signal should be equipped with an appropriate resistor that either pulls the GPIO up [to 5-volts] or down [to ground]. The option to enable the internal pull-up or pull-down resistors is not available for safety reasons. Use your own resistor to pull the GPIO high or low.

Examples of devices that can be used with edge detection: simple switches and buttons, PIR motion sensors, reed switches, hall effect sensors, float switches, and more.

5.11.2 Displays

There are only a few number fo displays that are supported. 16x2 and 20x4 character LCD displays with I2C backpacks and the 128x32 / 128x64 OLED displays are supported. The below image is the type of device with the I2C backpack that should be compatible. See Supported Functions for more information.

5.11.3 Raspberry Pi

The Raspberry Pi has an integrated temperature sensor on the BCM2835 SoC that measure the temperature of the CPU/GPU. This is the easiest sensor to set up in Mycodo, as it is immediately available to be used.

5.11.4 AM2315

From @Theoi-Meteoroi on GitHub:

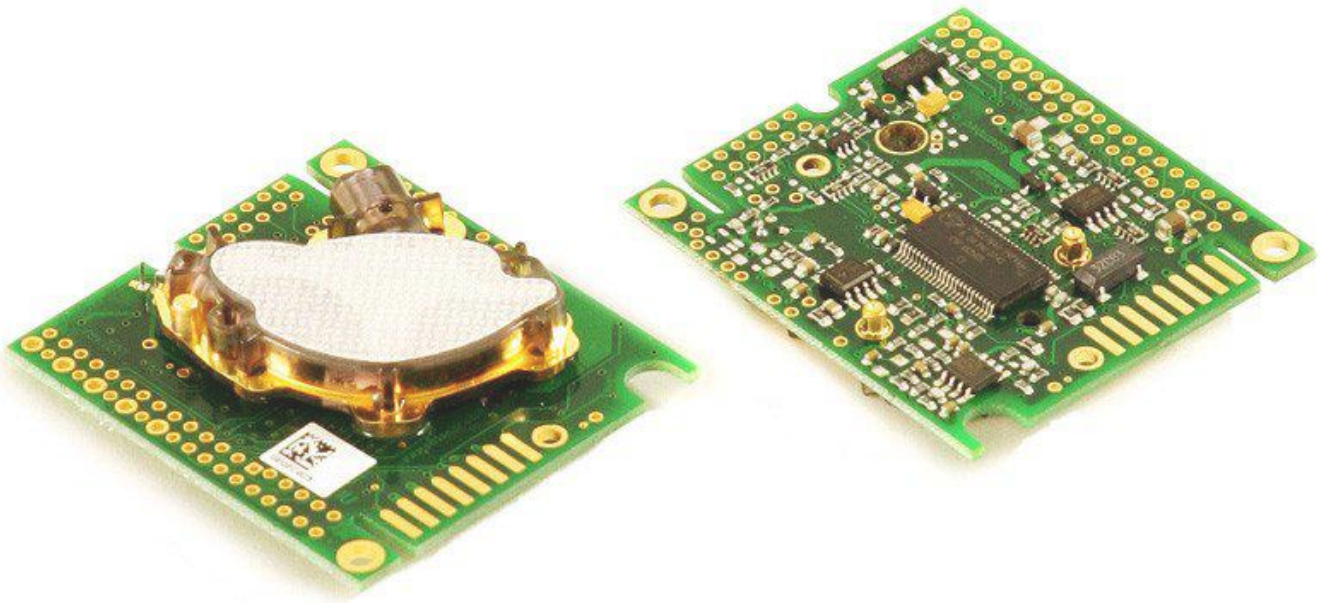
I figured out why this [AM2315] sensor is unreliable with Rpi3 hardware I2C. It is among a number of I2C devices that really hates the BCM2835 clock stretching blunder (hardware bug: raspberrypi/linux#254). The wakeup attempts fail, consistently. I checked the bitstream with a sniffer, and see that the sensor may respond once out of 20 or so tries (or not at all) but only with a single byte returned. The solution is to use a software implementation of the I2C bus. You need to add pull-up resistors (4.7k is dandy) to 3.3v and install the i2c_gpio device overlay. Seems to work fine now, will run for a few days, but the CRC failures are gone and I get good readings, every time. And no twiddling the power for the sensor is required.

To enable software I2C, add the following line to your `/boot/config.txt`

```
dtoverlay=i2c-gpio,i2c_gpio_sda=23,i2c_gpio_scl=24,i2c_gpio_delay_us=4
```

After rebooting, a new I2C bus at /dev/i2c-3 should exist with SDA on pin 23 (BCM) and SCL on pin 24 (BCM). Make sure you add the appropriate pull-up resistors before connecting any devices.

5.11.5 K-30



Be very careful when connecting the K-30, as there is no reverse-voltage protection and improper connections could destroy your sensor.

Wiring instructions for the Raspberry Pi can be found here.

5.11.6 USB Device Persistence Across Reboots

From (#547) Theoi-Meteoroi on Github:

Using USB devices, such as USB-to-serial interfaces (CP210x) to connect a sensor, while convenient, poses an issue if there are multiple devices when the system reboots. After a reboot, there is no guarantee the device will persist with the same name. For instance, if Sensor A is /dev/ttyUSB0 and Sensor B is /dev/ttyUSB1, after a reboot Sensor A may be /dev/ttyUSB1 and Sensor B may be /dev/ttyUSB0. This will cause Mycodo to query the wrong device for a measurement, potentially causing a mis-measurement, or worse, an incorrect measurement because the response is not from the correct sensor (I've seen my temperature sensor read 700+ degrees celsius because of this!). Follow the instructions below to alleviate this issue.

I use udev to create a persistent device name ('/dev/dust-sensor') that will be linked to the /dev/ttyUSBn that is chosen at device arrival in the kernel. The only requirement is some attribute returned from the USB device that is unique. The common

circumstance is that none of the attributes are unique and you get stuck with just VID and PID, which is ok as long as you don't have any other adapters that report the same VID and PID. If you have multiple adapters with the same VID and PID, then hopefully they have some unique attribute. This command will walk the attributes. Run on each USB device and then compare differences to possibly find some attribute to use.

`udevadm info --name=/dev/ttyUSB0 --attribute-walk`

I ended up using the serial number on the ZH03B to program the USB adapter serial field. This way guarantees unique serial numbers rather than me trying to remember what was the last serial number I used to increment by 1.

When you plug a USB device in it can be enumerated to different device names by the operating system. To fix this problem for this sensor on linux, I changed attributes that make the connection unique.

First - find the VID and PID for the USB device:

```
pi@raspberry:~ $ lsusb
Bus 001 Device 008: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x UART Bridge / myAVR mySmartUSB light
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp. SMC9514 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

In this case the Vendor ID is 10c4 The Product ID is ea60

> **Note**
>
> If you have multiple devices and you find your IDs to be the same, you can change IDs with the Simplicity Studio Xpress Configurator tool (discussed starting on page 6 of the AN721: USBXpress Device Configuration and Programming Guide).

Since I changed the serial number field - this will be unique.

```
pi@raspberry:~ $ udevadm info --name=/dev/ttyUSB0 --attribute-walk | grep serial
SUBSYSTEMS=="usb-serial"
ATTRS{serial}=="ZH03B180904"
ATTRS{serial}=="3f980000.usb"
```

Now I have an attribute to tell udev what to do. I create a file in /etc/udev/rules.d with a name like "99-dustsensor.rules". In that file I tell udev what device name to create when it sees this device plugged in:

`SUBSYSTEM=="tty", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", ATTRS{serial}=="ZH03B180904" SYMLINK+="dust-sensor"`

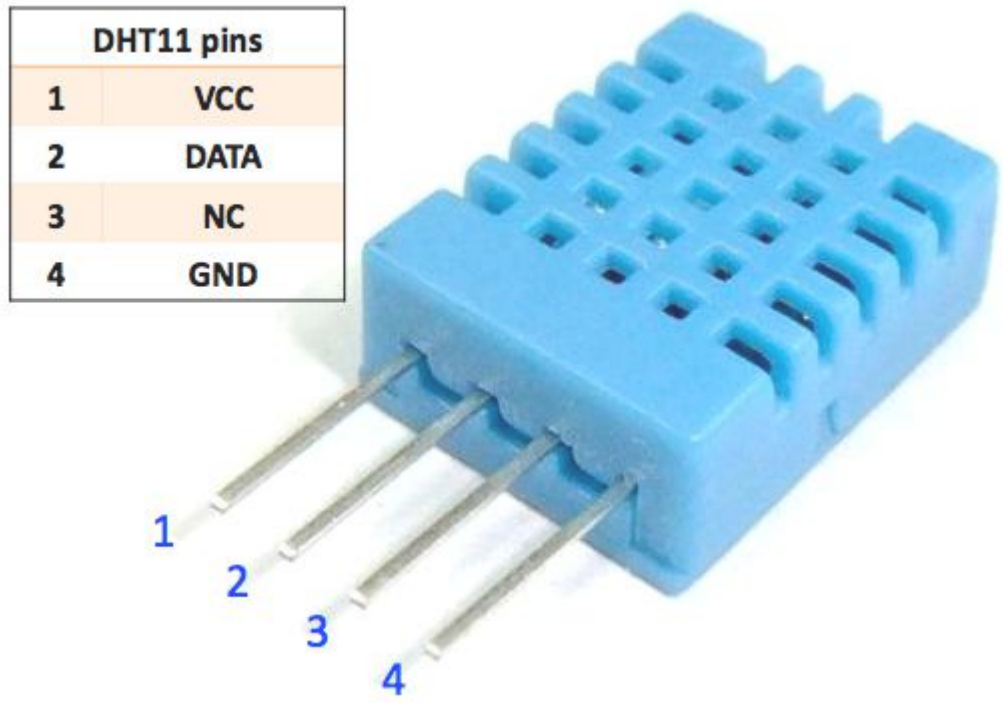
To test the new rule:

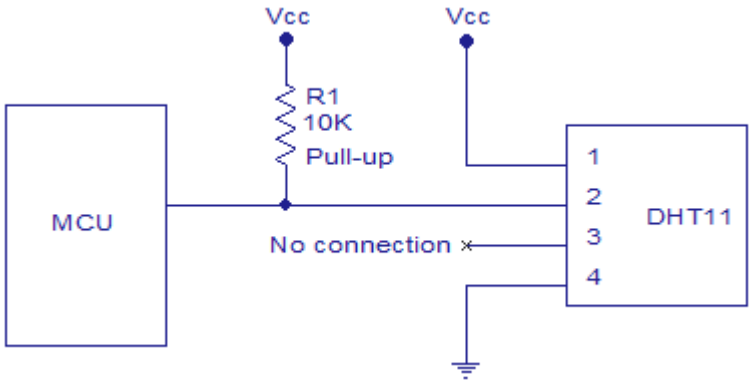
```
pi@raspberry:/dev $ sudo udevadm trigger
pi@raspberry:/dev $ ls -al dust-sensor
lrwxrwxrwx 1 root root 7 Oct 6 21:04 dust-sensor -> ttyUSB0
```

Now, every time the dust sensor is plugged in, it shows up at /dev/dust-sensor

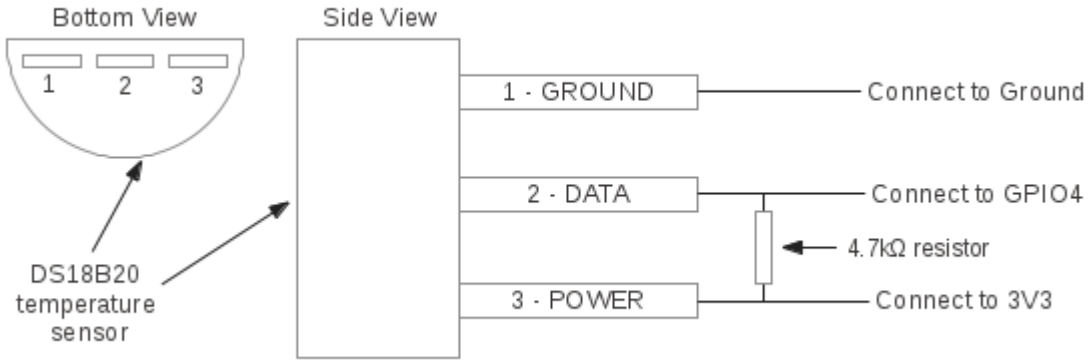
5.11.7 Diagrams

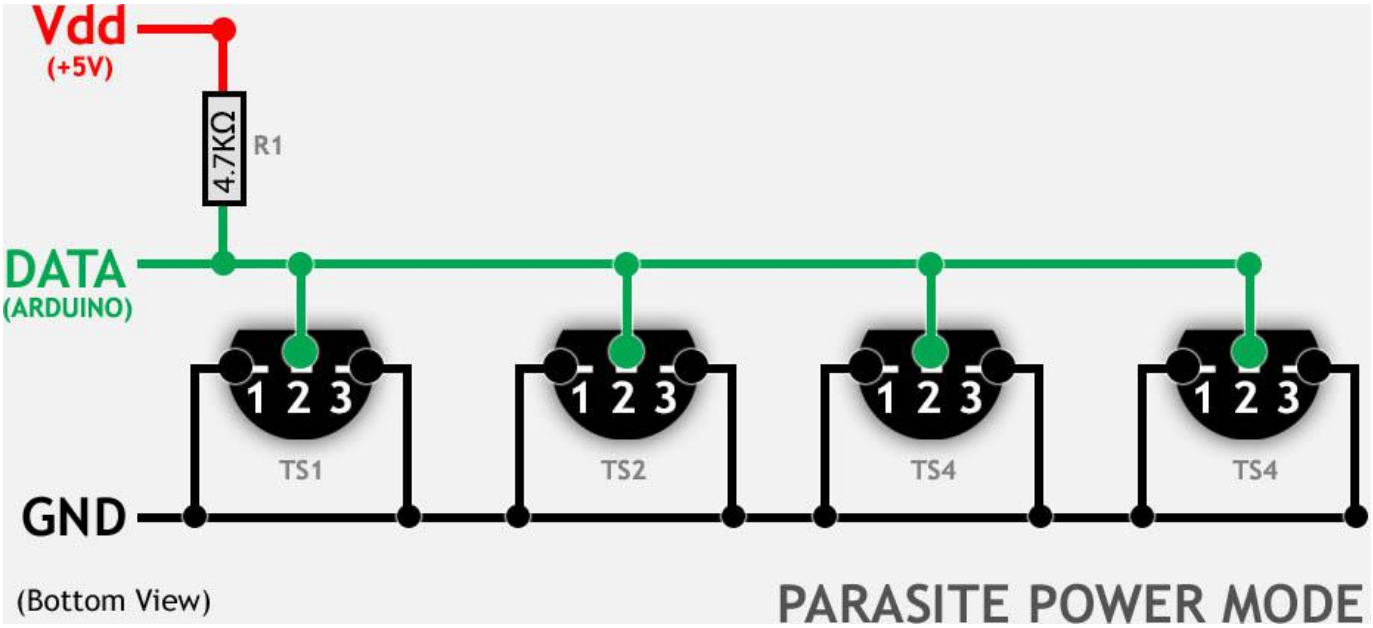
DHT11 Diagrams

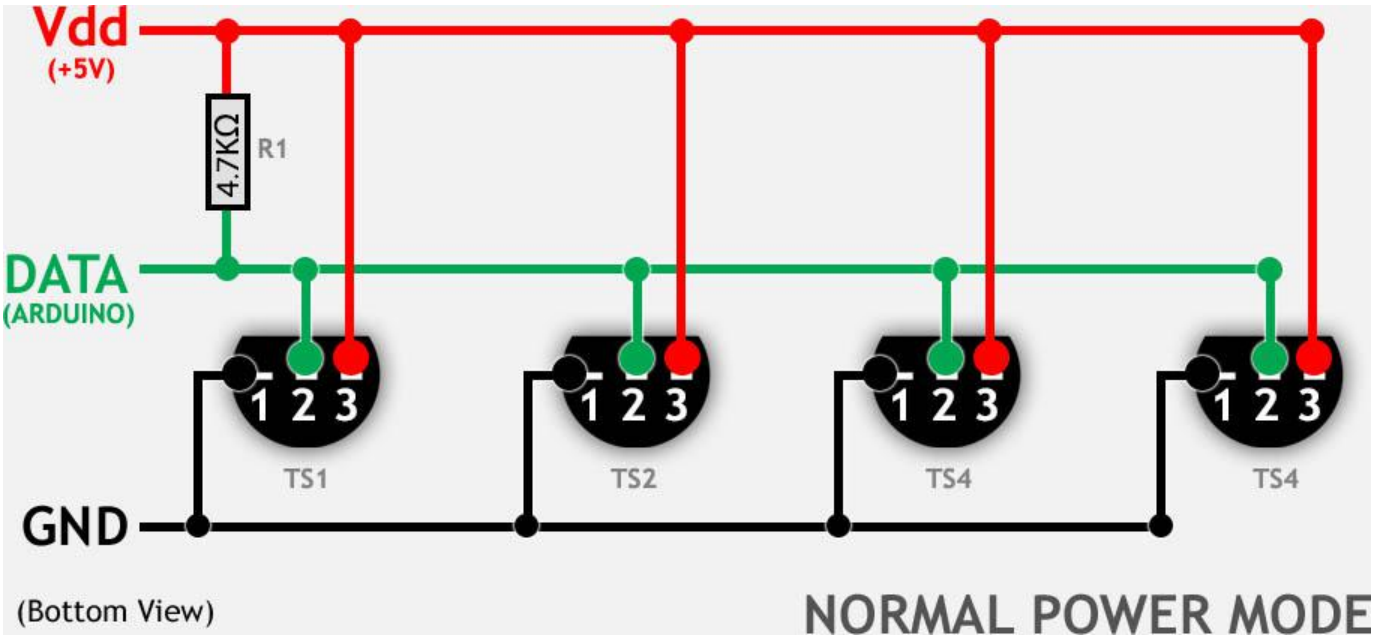




DS18B20 Diagrams

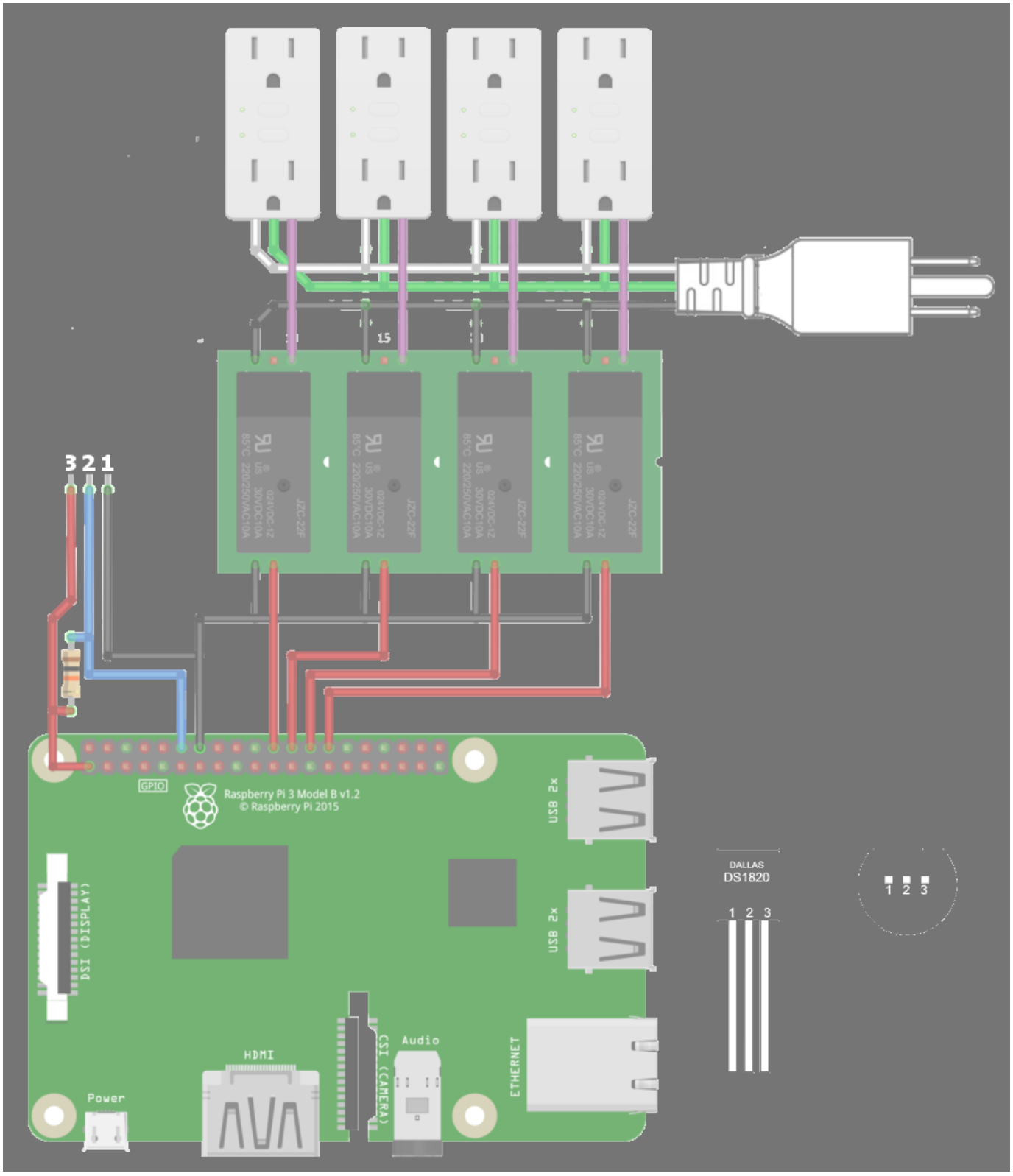




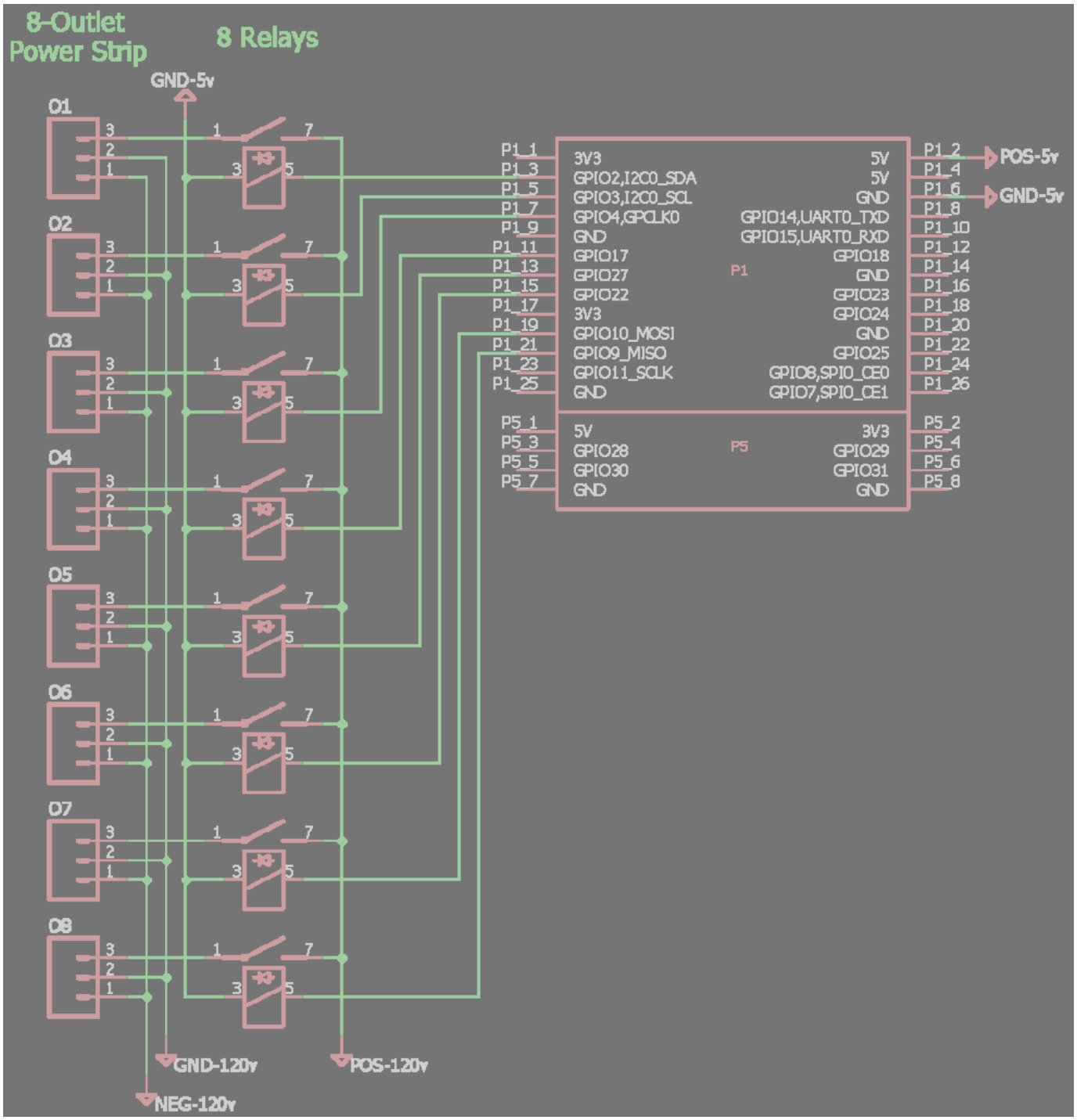


Raspberry Pi and Relay Diagrams

RASPBERRY PI, 4 RELAYS, 4 OUTLETS, 1 DS18B20 SENSOR



RASPBERRY PI, 8 RELAYS, 8 OUTLETS



6. System

6.1 System Information

Page: `[Gear Icon] -> System Information`

This page serves to provide information about the Mycodo frontend and backend as well as the linux system it's running on. Several commands and their output are listed to give the user information about how their system is running.

Command	Description
Mycodo Version	The current version of Mycodo, reported by the configuration file.
Python Version	The version of python currently running the web user interface.
Database Version	The current version of the settings database. If the current version is different from what it should be, an error will appear indicating the issue and a link to find out more information about the issue.
Daemon Status	This will be a green "Running" or a red "Stopped". Additionally, the Mycodo version and hostname text at the top-left of the screen May be Green, Yellow, or Red to indicate the status. Green = daemon running, yellow = unable to connect, and red = daemon not running.
...	Several other status indicators and commands are listed to provide information about the health of the system. Use these in addition to others to investigate software or hardware issues.

6.2 System Configuration

Page: `[Gear Icon] -> Configure`

The settings menu, accessed by selecting the gear icon in the top-right, then the Configure link, is a general area for various system-wide configuration options.

6.2.1 General Settings

Page: `[Gear Icon] -> Configure -> General`

Setting	Description
Language	Set the language that will be displayed in the web user interface.
Force HTTPS	Require web browsers to use SSL/HTTPS. Any request to http:// will be redirected to https://.
Hide success alerts	Hide all success alert boxes that appear at the top of the page.
Hide info alerts	Hide all info alert boxes that appear at the top of the page.
Hide warning alerts	Hide all warning alert boxes that appear at the top of the page.
Opt-out of statistics	Turn off sending anonymous usage statistics. Please consider that this helps the development to leave on.
Check for Updates	Automatically check for updates every 2 days and notify through the web interface. If there is a new update, the Configure (Gear Icon) as well as the Upgrade menu will turn the color red.

6.2.2 Energy Usage Settings

Page: `[Gear Icon] -> Configure -> General`

In order to calculate accurate energy usage statistics, a few characteristics of your electrical system needs to be know. These variables should describe the characteristics of the electrical system being used by the relays to operate electrical devices.

> **Note**
>
> If not using a current sensor, proper energy usage calculations will rely on the correct current draw to be set for each output (see Output Settings).

Setting	Description
Max Amps	Set the maximum allowed amperage to be switched on at any given time. If a output that's instructed to turn on will cause the sum of active devices to exceed this amount, the output will not be allowed to turn on, to prevent any damage that may result from exceeding current limits.
Voltage	Alternating current (AC) voltage that is switched by the outputs. This is usually 120 or 240.
Cost per kWh	This is how much you pay per kWh.
Currency Unit	This is the unit used for the currency that pays for electricity.
Day of Month	This is the day of the month (1-30) that the electricity meter is read (which will correspond to the electrical bill).
Generate Usage/Cost Report	These options define when an Energy Usage Report will be generated. Currently, these Only support the Output Duration calculation method.

6.2.3 Input Settings

Page: `[Gear Icon] -> Configure -> Custom Inputs`

Input modules may be imported and used within Mycodo. These modules must follow a specific format. See Custom Inputs for more details.

Setting	Description
Import Input Module	Select your input module file, then click this button to begin the import.

6.2.4 Output Settings

Page: `[Gear Icon] -> Configure -> Custom Outputs`

Output modules may be imported and used within Mycodo. These modules must follow a specific format. See Custom Outputs for more details.

Setting	Description
Import Output Module	Select your output module file, then click this button to begin the import.

6.2.5 Function Settings

Page: `[Gear Icon] -> Configure -> Custom Functions`

Function modules may be imported and used within Mycodo. These modules must follow a specific format. See Custom Functions for more details.

Setting	Description
Import Function Module	Select your function module file, then click this button to begin the import.

6.2.6 Action Settings

Page: `[Gear Icon] -> Configure -> Custom Actions`

Action modules may be imported and used within Mycodo. These modules must follow a specific format. See Custom Actions for more details.

Setting	Description
Import Action Module	Select your action module file, then click this button to begin the import.

6.2.7 Widget Settings

Page: `[Gear Icon] -> Configure -> Custom Widgets`

Widget modules may be imported and used within Mycodo. These modules must follow a specific format. See Custom Widgets for more details.

Setting	Description
Import Widget Module	Select your widget module file, then click this button to begin the import.

6.2.8 Measurement Settings

Page: `[Gear Icon] -> Configure -> Measurements`

New measurements, units, and conversions can be created that can extend functionality of Mycodo beyond the built-in types and equations. Be sure to create units before measurements, as units need to be selected when creating a measurement. A measurement can be created that already exists, allowing additional units to be added to a pre-existing measurement. For example, the measurement 'altitude' already exists, however if you wanted to add the unit 'fathom', first create the unit 'fathom', then create the measurement 'altitude' with the 'fathom' unit selected. It is okay to create a custom measurement for a measurement that already exist (this is how new units for a currently-installed measurement is added).

Setting	Description
Measurement ID	ID for the measurement to use in the measurements_dict of input modules (e.g. "length", "width", "speed").
Measurement Name	Common name for the measurement (e.g. "Length", "Weight", "Speed").
Measurement Units	Select all the units that are associated with the measurement.
Unit ID	ID for the unit to use in the measurements_dict of input modules (e.g. "K", "g", "m").
Unit Name	Common name for the unit (e.g. "Kilogram", "Meter").
Unit Abbreviation	Abbreviation for the unit (e.g. "kg", "m").
Convert From Unit	The unit that will be converted from.
Convert To Unit	The unit that will be converted to.
Equation	The equation used to convert one unit to another. The lowercase letter "x" must be included in the equation (e.g. "x/1000+20", "250*(x/3)"). This "x" will be replaced with the actual measurement being converted.

6.2.9 Users

Page: `[Gear Icon] -> Configure -> Users`

Mycodo requires at least one Admin user for the login system to be enabled. If there isn't an Admin user, the web server will redirect to an Admin Creation Form. This is the first page you see when starting Mycodo for the first time. After an Admin user has been created, additional users may be created from the User Settings page.

Setting	Description
Username	Choose a user name that is between 2 and 64 characters. The user name is case insensitive (all user names are converted to lower-case).
Email	The email associated with the new account.
Password/ Repeat	Choose a password that is between 6 and 64 characters and only contains letters, numbers, and symbols.
Keypad Code	Set an optional numeric code that is at least 4 digits for logging in using a keypad.
Role	Roles are a way of imposing access restrictions on users, to either allow or deny actions. See the table below for explanations of the four default Roles.
Theme	The web user interface theme to apply, including colors, themes, and other design elements.

Roles

Roles define the permissions of each user. There are 4 default roles that determine if a user can view or edit particular areas of Mycodo. Four roles are provided by default, but custom roles may be created.

Role	Admin	Editor	Monitor	Guest
Edit Users	X			
Edit Controllers	X	X		
Edit Settings	X	X		
View Settings	X	X	X	
View Camera	X	X	X	
View Stats	X	X	X	
View Logs	X	X	X	

The `Edit Controllers` permission protects the editing of Conditionals, Graphs, LCDs, Methods, PIDs, Outputs, and Inputs.

The `View Stats` permission protects the viewing of usage statistics and the System Information and Energy Usage pages.

6.2.10 Pi Settings

Page: `[Gear Icon] -> Configure -> Raspberry Pi`

Pi settings configure parts of the linux system that Mycodo runs on.

pigpiod is required if you wish to use PWM Outputs, as well as PWM, RPM, DHT22, DHT11, HTU21D Inputs.

Setting	Description
Enable/ Disable Feature	These are system interfaces that can be enabled and disabled from the web UI via the `raspi-config` command.
pigpiod Sample Rate	This is the sample rate the pigpiod service will operate at. The lower number enables faster PWM frequencies, but may significantly increase processor load on the Pi Zeros. pigpiod may als be disabled completely if it's not required (see note, above).

6.2.11 Alert Settings

Page: `[Gear Icon] -> Configure -> Alerts`

Alert settings set up the credentials for sending email notifications.

Setting	Description
SMTP Host	The SMTP server to use to send emails from.
SMTP Port	Port to communicate with the SMTP server (465 for SSL, 587 for TSL).
Enable SSL	Check to enable SSL, uncheck to enable TSL.
SMTP User	The user name to send the email from. This can be just a name or the entire email address.
SMTP Password	The password for the user.
From Email	What the from email address be set as. This should be the actual email address for this user.
Max emails (per hour)	Set the maximum number of emails that can be sent per hour. If more notifications are triggered within the hour and this number has been reached, the notifications will be discarded.
Send Test Email	Test the email configuration by sending a test email.

6.2.12 Camera Settings

Page: `[Gear Icon] -> Configure -> Camera`

Many cameras can be used simultaneously with Mycodo. Each camera needs to be set up in the camera settings, then may be used throughout the software.

> **Note**
>
> Not every option (such as Hue or White Balance) may be able to be used with your particular camera, due to manufacturer differences in hardware and software.

Setting	Description
Type	Select whether the camera is a Raspberry Pi Camera or a USB camera.
Library	Select which library to use to communicate with the camera. The Raspberry Pi Camera uses picamera, and USB cameras should be set to fswebcam.
Device	The device to use to connect to the camera. fswebcam is the only library that uses this option.
Output	This output will turn on during the capture of any still image (which includes timelapses).
Output Duration	Turn output on for this duration of time before the image is captured.
Rotate Image	The number of degrees to rotate the image.
...	Image Width, Image Height, Brightness, Contrast, Exposure, Gain, Hue, Saturation, White Balance. These options are self-explanatory. Not all options will work with all cameras.
Pre Command	A command to execute (as user 'root') before a still image is captured.
Post Command	A command to execute (as user 'root') after a still image is captured.
Flip horizontally	Flip, or mirror, the image horizontally.
Flip vertically	Flip, or mirror, the image vertically.

6.2.13 Diagnostic Settings

Page: `[Gear Icon] -> Configure -> Diagnostics`

Sometimes issues arise in the system as a result of incompatible configurations, either the result of a misconfigured part of the system (Input, Output, etc.) or an update that didn't properly handle a database upgrade, or other unforeseen issue. Sometimes it is necessary to perform diagnostic actions that can determine the cause of the issue or fix the issue itself. The options below are meant to alleviate issues, such as a misconfigured dashboard element causing an error on the `Data -> Dashboard` page, which may cause an inability to access the `Data -> Dashboard` page to correct the issue. Deleting all Dashboard Elements may be the most economical method to enable access to the `Data -> Dashboard` page again, at the cost of having to readd all the Dashboard Elements that were once there.

Setting	Description
Delete All Dashboard Elements	Delete all saved Dashboard Elements from the Dashboard.
Delete All Notes and Note Tags	Delete all notes and note tags.

6.3 Upgrade/Backup/Restore

6.3.1 Upgrading

Page: `[Gear Icon] -> Upgrade`

If you already have Mycodo installed, you can perform an upgrade to the latest Mycodo Release by either using the Upgrade option in the web interface (recommended) or by issuing the following command in a terminal. A log of the upgrade process is created at `/var/log/mycodo/mycodoupgrade.log` and is also available from the `[Gear Icon] -> Mycodo Logs` page.

```
sudo mycodo-commands upgrade-mycodo
```

6.3.2 Backup-Restore

Page: `[Gear Icon] -> Backup Restore`

A backup is made to /var/Mycodo-backups when the system is upgraded or instructed to do so from the web interface on the `[Gear Icon] -> Backup Restore` page.

If you need to restore a backup, this can be done on the `[Gear Icon] -> Backup Restore` page (recommended). Find the backup you would like restored and press the Restore button beside it. If you're unable to access the web interface, a restore can also be initialized through the command line. Use the following command to initialize a restore. The [backup_location] must be the full path to the backup to be restored (e.g. "/var/Mycodo-backups/Mycodo-backup-2018-03-11_21-19-15-5.6.4/" without quotes).

```
sudo mycodo-commands backup-restore [backup_location]
```

6.4 Export/Import

Page: `More -> Export Import`

Measurements that fall within the selected date/time frame may be exported as CSV with their corresponding timestamps.

Additionally, the entire measurement database (influxdb) may be exported as a ZIP archive backup. This ZIP may be imported back in any Mycodo system to restore these measurements.

Note

Measurements are associated with specific IDs that correspond to the Inputs/Outputs/etc. of your specific system. If you import measurements without also importing the associated Inputs/Outputs/etc., you will not see these measurements (e.g. on Dashboard Graphs). Therefore, it is recommended to export both Measurements and Settings at the same time so when you import them at a later time, you will have the devices associated with the measurements available on the system you're importing to.

Note

Importing measurement data will not destroy old data and will be added to the current measurement data.

Mycodo settings may be exported as a ZIP file containing the Mycodo settings database (sqlite) and any custom Inputs, Outputs, Functions, and Widgets. This ZIP file may be used to restore these to another Mycodo install, as long as the Mycodo and database versions being imported are equal or less than the system you are installing them to. Additionally, you can only import to a system with the same major version number (the first number in the version format x.x.x). For instance, you can export settings from Mycodo 8.5.0 and import them into Mycodo 8.8.0, however you can not import them into Mycodo 8.2.0 (earlier version with same major version number), 7.0.0 (not the same major version number), or 9.0.0 (not the same major version number).

Warning

An import will override the current settings and custom controller data (i.e. destroying it). It is advised to make a Mycodo backup prior to attempting an import.

6.5 Error Codes

6.5.1 Error Codes

Mycodo can return a number of different errors. Below are a few of the numbered errors that you may receive and information about how to diagnose the issue.

Error 100

Cannot set a value of '**X**' of type **Y**. Must be a float or string representing a float.

- Examples:
- Cannot set a value of '**1.33.4**' of type **str**.
- Cannot set a value of '**Output: 1.2**' of type **str**.
- Cannot set a value of '**[1.3, 2.4]**' of type **list**.
- Cannot set a value of '**{"output": 1.99}**' of type **dict**.
- Cannot set a value of '**None**' of type **Nonetype**.

This error occurs because the value provided to be stored in the influxdb time-series database is not a numerical value (integer or decimal/float) or it is not a string that represents a float (e.g. "5", "3.14"). There are a number of reasons why this error occurs, but the most common reason is the sensor being ready by an Input did not return a measurement when queried, or it returned something other than something that represents a numerical value, indicating the sensor is not working. This could be from a number of reasons, including but not limited to, faulty wiring, faulty/insufficient power supply, defective sensor, I2C bus hasn't been enabled, misconfigured settings, etc. Often, a sensor can fail or not get set up correctly during Input initialization when the daemon starts, leading to this error every measurement period. You will need to review the Daemon Log (`[Gear Icon] -> Mycodo Logs`) all the way back to when the daemon started (since this is when the Input started and potentially failed with an initial error that may be more informative). Enabling Log Level: Debug in the Controller setting can also be useful by providing debugging log lines (when available) in addition to the info and error log lines.

Error 101

X not set up properly

- Examples
- Device not set up
- Output channel **Y** not set up

This error occurs when the Controller (Input/Output/Function/etc.) could not properly initialize the device or channel when it started and is now trying to access an uninitialized device or channel. For Inputs, this could be loading the 3rd party library used to communicate with the sensor. If there was an error loading the library, then the library cannot be used to communicate with the sensor. You will often need to review the Daemon Log (`[Gear Icon] -> Mycodo Logs`) for any relevant errors that occurred when the Controller was initially activated to determine the issue setting up the device. Try deactivating, then activating the device, to see the initialization error again. Enabling Log Level: Debug in the Controller setting can also be useful by providing debugging log lines (when available) in addition to the info and error log lines.

6.6 Mycodo Client

The Mycodo client is a command-line tool used to communicate with the daemon.

```
pi@raspberry:~ $ mycodo-client --help
usage: mycodo-client [-h] [-c] [--activatecontroller CONTROLLER ID]
                     [--deactivatecontroller CONTROLLER ID] [--ramuse] [-t]
                     [--trigger_action ACTIONID]
                     [--trigger_all_actions FUNCTIONID]
                     [--input_force_measurements INPUTID]
                     [--backlight_on DEVID] [--backlight_off DEVID]
                     [--lcd_reset DEVID] [--get_measurement ID UNIT CHANNEL]
                     [--output_state OUTPUTID]
                     [--output_currently_on OUTPUTID] [--outputoff OUTPUTID]
                     [--outputon OUTPUTID] [--duration SECONDS]
                     [--dutycycle DUTYCYCLE] [--pid_pause ID] [--pid_hold ID]
                     [--pid_resume ID] [--pid_get_setpoint ID]
                     [--pid_get_error ID] [--pid_get_integrator ID]
                     [--pid_get_derivator ID] [--pid_get_kp ID]
                     [--pid_get_ki ID] [--pid_get_kd ID]
                     [--pid_set_setpoint ID SETPOINT]
                     [--pid_set_integrator ID INTEGRATOR]
                     [--pid_set_derivator ID DERIVATOR] [--pid_set_kp ID KP]
                     [--pid_set_ki ID KI] [--pid_set_kd ID KD]

Client for Mycodo daemon.

optional arguments:
  -h, --help            show this help message and exit
  -c, --checkdaemon     Check if all active daemon controllers are running
  --activatecontroller CONTROLLER ID
                        Activate controller. Options: Conditional,
                        PID, Input
  --deactivatecontroller CONTROLLER ID
                        Deactivate controller. Options: Conditional,
                        PID, Input
  --ramuse              Return the amount of ram used by the Mycodo daemon
  -t, --terminate       Terminate the daemon
  --trigger_action ACTIONID
                        Trigger action with Action ID
  --trigger_all_actions FUNCTIONID
                        Trigger all actions belonging to Function with ID
  --input_force_measurements INPUTID
                        Force acquiring measurements for Input ID
  --backlight_on DEVID
                        Turn on display backlight with device ID
  --backlight_off DEVID
                        Turn off display backlight with device ID
  --lcd_reset DEVID     Reset display with device ID
  --get_measurement ID UNIT CHANNEL
                        Get the last measurement
  --output_state OUTPUTID
                        State of output with output ID
  --output_currently_on OUTPUTID
                        How many seconds an output has currently been active
                        for
  --outputoff OUTPUTID  Turn off output with output ID
  --outputon OUTPUTID   Turn on output with output ID
  --duration SECONDS    Turn on output for a duration of time (seconds)
  --dutycycle DUTYCYCLE
                        Turn on PWM output for a duty cycle (%)
  --pid_pause ID        Pause PID controller.
  --pid_hold ID         Hold PID controller.
  --pid_resume ID       Resume PID controller.
  --pid_get_setpoint ID
                        Get the setpoint value of the PID controller.
  --pid_get_error ID    Get the error value of the PID controller.
  --pid_get_integrator ID
                        Get the integrator value of the PID controller.
  --pid_get_derivator ID
                        Get the derivator value of the PID controller.
  --pid_get_kp ID       Get the Kp gain of the PID controller.
  --pid_get_ki ID       Get the Ki gain of the PID controller.
  --pid_get_kd ID       Get the Kd gain of the PID controller.
  --pid_set_setpoint ID SETPOINT
                        Set the setpoint value of the PID controller.
  --pid_set_integrator ID INTEGRATOR
                        Set the integrator value of the PID controller.
  --pid_set_derivator ID DERIVATOR
                        Set the derivator value of the PID controller.
  --pid_set_kp ID KP    Set the Kp gain of the PID controller.
  --pid_set_ki ID KI    Set the Ki gain of the PID controller.
  --pid_set_kd ID KD    Set the Kd gain of the PID controller.
```

6.7 API

6.7.1 REST API

As of version 8, Mycodo has a REST API (See API Endpoint Documentation).

An API is an application programming interface - in short, it's a set of rules that lets programs talk to each other, exposing data and functionality across the internet in a consistent format.

REST stands for Representational State Transfer. This is an architectural pattern that describes how distributed systems can expose a consistent interface. When people use the term 'REST API,' they are generally referring to an API accessed via HTTP protocol at a predefined set of URLs. These URLs represent various resources - any information or content accessed at that location, which can be returned as JSON, HTML, audio files, or images. Often, resources have one or more methods that can be performed on them over HTTP, like GET, POST, PUT and DELETE.

Authentication

An API Key can be generated from the User Settings page (`[Gear Icon] -> Configure -> Users`). This is stored as a 128-bit bytes object in the database, but will be presented to the user as a base64-encoded string. This can be used to access HTTPS endpoints.

Mycodo supports several authentication methods. All API requests must be made over HTTPS. Calls made over plain HTTP will fail. API requests without authentication will fail.

Bash Examples

`curl` can be used, but you must either use `-k` to allow the use of an unsigned SSL certificate, or use your own certificate and domain.

```
curl -k -v -X GET "https://127.0.0.1/api/settings/users" -H "authorization: Basic OscjVcxRGiOXczregANBRXG3VMMro+oolPYdauadLblaNThd79bzFPITJjYneU1yK/
Ikc9ahHXml9JiKZO9+hogKoIp2Q8a2cMFBGevgJSd5jYVYz5D83dFE5+OBvvKKaN1U5TvPOXXcj3lkjvPzgxOnEFOCZUsKfU3MA3cFEs=" -H "accept: application/vnd.mycodo.v1+json"
```

```
curl -k -v -x GET "https://127.0.0.1/api/settings/users -H "X-API-KEY: OscjVcxRGiOXczregANBRXG3VMMro+oolPYdauadLblaNThd79bzFPITJjYneU1yK/
Ikc9ahHXml9JiKZO9+hogKoIp2Q8a2cMFBGevgJSd5jYVYz5D83dFE5+OBvvKKaN1U5TvPOXXcj3lkjvPzgxOnEFOCZUsKfU3MA3cFEs=" -H "accept: application/vnd.mycodo.v1+json"
```

```
curl -k -v -x GET "https://127.0.0.1/api/settings/users?api_key=OscjVcxRGiOXczregANBRXG3VMMro+oolPYdauadLblaNThd79bzFPITJjYneU1yK/
Ikc9ahHXml9JiKZO9+hogKoIp2Q8a2cMFBGevgJSd5jYVYz5D83dFE5+OBvvKKaN1U5TvPOXXcj3lkjvPzgxOnEFOCZUsKfU3MA3cFEs=" -H "accept: application/vnd.mycodo.v1+json"
```

Python Example (GET)

```
import json
import requests

ip_address = '127.0.0.1'
api_key = 'YOUR_API_KEY'
endpoint = 'settings/inputs'
url = 'https://{ip}/api/{ep}'.format(ip=ip_address, ep=endpoint)
headers = {'Accept': 'application/vnd.mycodo.v1+json',
           'X-API-KEY': api_key}
response = requests.get(url, headers=headers, verify=False)
print("Response Status: {}".format(response.status_code))
print("Response Headers: {}".format(response.headers))
response_dict = json.loads(response.text)
print("Response Dictionary: {}".format(response_dict))
```

Python Example (POST)

```
import json
import requests
import urllib3

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

ip_address = '127.0.0.1'
api_key = 'YOUR_API_KEY'
endpoint = 'outputs/3f5a4806-c830-432d-b329-7821da8336e4'
url = 'https://{ip}/api/{ep}'.format(ip=ip_address, ep=endpoint)
data = {"state": True}  # Turn Output On
```

```
headers = {'Accept': 'application/vnd.mycodo.v1+json',
           'X-API-KEY': api_key}
response = requests.post(url, json=data, headers=headers, verify=False)
print("Response Status: {}".format(response.status_code))
print("Response Headers: {}".format(response.headers))
response_dict = json.loads(response.text)
print("Response Dictionary: {}".format(response_dict))
```

Errors

Mycodo uses conventional HTTP response codes to indicate the success or failure of an API request. In general: Codes in the 2xx range indicate success. Codes in the 4xx range indicate an error that failed given the information provided (e.g., a required parameter was omitted, a charge failed, etc.). Codes in the 5xx range indicate an error with Mycodo's servers (these are rare).

Some 4xx errors that could be handled programmatically (e.g., a card is declined) include an error code that briefly explains the error reported.

Endpoints

A vendor-specific content type header must be included to determine which API version to use. For version 1, this is "application/vnd.mycodo.v1+json", as can be seen in the examples, above.

Visit https://{RASPBERRY_PI_IP_ADDRESS}/api for documentation of the current API endpoints of your Mycodo install.

Documentation for the latest API version is also available in HTML format: `Mycodo API Docs <https://kizniche.github.io/Mycodo/mycodo-api.html>` __

6.7.2 Daemon Control Object

DaemonControl()

class mycodo_client.DaemonControl (pyro_uri='PYRO:mycodo.pyro_server@127.0.0.1:9080', pyro_timeout=None)

The mycodo client object implements a way to communicate with a mycodo daemon and query information from the influxdb database.

Example usage:

```
from mycodo.mycodo_client import DaemonControl
control = DaemonControl()
control.terminate_daemon()
```

Parameters:

- **pyro_uri** - the Pyro5 uri to use to connect to the daemon.
- **pyro_timeout** - the Pyro5 timeout period.

controller_activate()

controller_activate (controller_id)

Activates a controller.

Parameters:

- **controller_type** - the type of controller being activated. Options are: "Function", "Input", "Output", "PID", "Trigger", or "Function".
- **controller_id** - the unique ID of the controller to activate.

controller_deactivate()

controller_deactivate (controller_id)

Deactivates a controller.

Parameters:

- **controller_type** - the type of controller being deactivated. Options are: "Conditional", "Input", "Output", "PID", "Trigger", or "Function".
- **controller_id** - the unique ID of the controller to deactivate.

get_condition_measurement()

get_condition_measurement (condition_id)

Gets the measurement from a Condition of a Conditional Controller.

Parameters:

- **condition_id** - The unique ID of the controller.

get_condition_measurement_dict()

get_condition_measurement_dict (condition_id)

Gets the measurement dictionary from a Condition of a Conditional Controller.

Parameters:

- **condition_id** - The unique ID of the controller.

input_force_measurements()

input_force_measurements (input_id)

Induce an Input to conduct a measurement.

Parameters:

- **input_id** - The unique ID of the controller.

lcd_backlight()

lcd_backlight (lcd_id, state)

Turn the backlight of an LCD on or off, if the LCD supports that functionality.

Parameters:

- **lcd_id** - The unique ID of the controller.
- **state** - The state of the LCD backlight. Options are: False for off, True for on.

lcd_flash()

lcd_flash (lcd_id, state)

Cause the LCD backlight to start or stop flashing, if the LCD supports that functionality.

Parameters:

- **lcd_id** - The unique ID of the controller.
- **state** - The state of the LCD flashing. Options are: False for off, True for on.

lcd_reset()

lcd_reset (lcd_id)

Reset an LCD to its default startup state. This can be used to clear the screen, fix display issues, or turn off flashing.

Parameters:

- **lcd_id** - The unique ID of the controller.

output_off()

output_off (output_id, trigger_conditionals=True)

Turn an Output off.

Parameters:

- **output_id** - The unique ID of the Output.
- **trigger_conditionals** - Whether to trigger controllers that may be monitoring Outputs for state changes.

output_on()

output_on (output_id, output_type='sec', amount=0.0, min_off=0.0, trigger_conditionals=True)

Turn an Output on.

Parameters:

- **output_id** - The unique ID of the Output.
- **output_type** - The type of output to send to the output module (e.g. "sec", "pwm", "vol").
- **amount** - The amount to send to the output module.
- **min_off** - How long to keep the Output off after turning on, if on for a duration.
- **trigger_conditionals** - Whether to trigger controllers that may be monitoring Outputs for state changes.

output_on_off()

output_on_off (output_id, state, output_type='sec', amount=0.0,)

Turn an Output on or off.

Parameters:

- **output_id** - The unique ID of the Output.
- **state** - The state to turn the Output. Options are: "on", "off"
- **output_type** - The type of output to send to the output module (e.g. "sec", "pwm", "vol").
- **amount** - The amount to send to the output module.

output_sec_currently_on()

output_sec_currently_on (output_id)

Get how many seconds an Output has been on.

Parameters:

- **output_id** - The unique ID of the Output.

output_setup()

output_setup (action, output_id)

Set up an Output (i.e. load/reload settings from database, initialize any pins/classes, etc.).

Parameters:

- **action** - What action to instruct for the Output. Options are: "Add", "Delete", or "Modify".
- **output_id** - The unique ID of the Output.

output_state()

output_state (output_id)

Gets the state of an Output. Returns "on" or "off" or duty cycle value.

Parameters:

- **output_id** - The unique ID of the Output.

pid_get()

pid_get (pid_id, setting)

Get a parameter of a PID controller.

Parameters:

- **pid_id** - The unique ID of the controller.
- **setting** - Which option to get. Options are: "setpoint", "error", "integrator", "derivator", "kp", "ki", or "kd".

pid_hold()

pid_hold (pid_id)

Set a PID Controller to Hold.

Parameters:

- **pid_id** - The unique ID of the controller.

pid_mod()

pid_mod (pid_id)

Refresh/Initialize the variables of a running PID controller.

Parameters:

- **pid_id** - The unique ID of the controller.

pid_pause()

pid_pause (pid_id)

Set a PID Controller to Pause.

Parameters:

- **pid_id** - The unique ID of the controller.

pid_resume()

pid_resume (pid_id)

Set a PID Controller to Resume.

Parameters:

- **pid_id** - The unique ID of the controller.

pid_set()

pid_set (pid_id, setting, value)

Set a parameter of a running PID controller.

Parameters:

- **pid_id** - The unique ID of the controller.
- **setting** - Which option to set. Options are: "setpoint", "method", "integrator", "derivator", "kp", "ki", or "kd".
- **value** - The value to set.

refresh_daemon_camera_settings()

refresh_daemon_camera_settings ()

Refresh the camera settings stored in the running daemon from the database values.

refresh_daemon_conditional_settings()

refresh_daemon_conditional_settings (unique_id)

Refresh the Conditional Controller settings of a running Conditional Controller.

Parameters:

- **unique_id** - The unique ID of the controller.

refresh_daemon_misc_settings()

refresh_daemon_misc_settings ()

Refresh the miscellaneous settings stored in the running daemon from the database values.

refresh_daemon_trigger_settings()

refresh_daemon_trigger_settings (unique_id)

Refresh the Trigger Controller settings of a running Trigger Controller.

Parameters:

- **unique_id** - The unique ID of the controller.

send_email()

send_email (recipients, message, subject)

Send an email with the credentials configured for alert notifications.

Parameters:

- **recipients** - The email address (string) or addresses (list of strings) to send the email.
- **message** - The body of the email.
- **subject** - The subject of the email.

terminate_daemon()

terminate_daemon ()

Instruct the daemon to shut down.

trigger_action()

trigger_action (action_id, message='', single_action=True, debug=False)

Instruct a Function Action to be executed.

Parameters:

- **action_id** - The unique ID of the Function Action.
- **message** - A message to send with the action that may be used by the action.
- **single_action** - True if only executing a single action.
- **debug** - Whether to show debug logging messages.

trigger_all_actions()

trigger_all_actions (function_id, message='', debug=False)

Instruct all Function Actions of a Function Controller to be executed sequentially.

Parameters:

- **function_id** - The unique ID of the controller.
- **message** - A message to send with the action that may be used by the action.
- **debug** - Whether to show debug logging messages.

7. Troubleshooting

7.1 Daemon Not Running

- Check the Logs: From the `[Gear Icon] -> Mycodo Logs` page, check the Daemon Log for any errors. If the issue began after an upgrade, also check the Upgrade Log for indications of an issue.
- Determine if the Daemon is Running: Execute `ps aux | grep '/var/mycodo-root/env/bin/python /var/mycodo-root/mycodo/mycodo_daemon.py'` in a terminal and look for an entry to be returned. If nothing is returned, the daemon is not running.
- Daemon Lock File: If the daemon is not running, make sure the daemon lock file is deleted at `/var/lock/mycodo.pid`. The daemon cannot start if the lock file is present.
- If a solution could not be found after investigating the above suggestions, submit a New Mycodo Issue on github.

7.2 Incorrect Database Version

- Check the `[Gear Icon] -> System Information` page or select the mycodo logo in the top-left.
- An incorrect database version error means the version stored in the Mycodo settings database (`~/Mycodo/databases/mycodo.db`) is not correct for the latest version of Mycodo, determined in the Mycodo config file (`~/Mycodo/mycodo/config.py`).
- This can be caused by an error in the upgrade process from an older database version to a newer version, or from a database that did not upgrade during the Mycodo upgrade process.
- Check the Upgrade Log for any issues that may have occurred. The log is located at `/var/log/mycodo/mycodoupgrade.log` but may also be accessed from the web UI (if you're able to): select `[Gear Icon] -> Mycodo Logs -> Upgrade Log`.
- Sometimes issues may not immediately present themselves. It is not uncommon to be experiencing a database issue that was actually introduced several Mycodo versions ago, before the latest upgrade.
- Because of the nature of how many versions the database can be in, correcting a database issue may be very difficult. It may be much easier to delete your database and let Mycodo generate a new one.
- Use the following commands to rename your database and restart the web UI. If both commands are successful, refresh your web UI page in your browser in order to generate a new database and create a new Admin user.

```
mv ~/Mycodo/databases/mycodo.db ~/Mycodo/databases/mycodo.db.backup
sudo service mycodoflask restart
```

7.3 More

Check out the Diagnosing Mycodo Issues Wiki Page on github for more information about diagnosing issues.

8. Translations

Mycodo has been translated to several languages. By default, the language of the browser will determine which language is used, but may be overridden in the General Settings, on the `[Gear Icon] -> Configure -> General` page. If you find an issue and would like to correct a translation or would like to add another language, this can be done at https://translate.kylegabriel.com.

Also check out the Translations section of the Wiki for details on working with translation files manually.