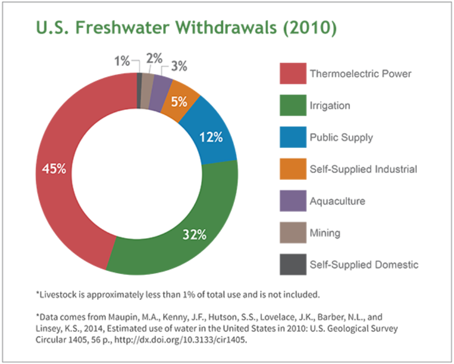
Modular Garden Monitoring System
Poster Presentation

Team CE12
Sadie Gladden, Eric Krenz, Zuguang Liu, Alan Trester
Advisor: Dr. Zachariah Fuchs

March 26, 2021

Problem Statement

- Increasing interest in garden and lawn care among new home owners
- High demand of water for use in lawns and garden



Solution

A modular garden monitoring system that

- Assists people with lawn or garden care
  - Real-time vital statistics
  - User configurable setup
  - Modular to mold to a variety of use-cases
- Alleviates the over-irrigation problem
  - Control system to keep garden soil moisture at healthy levels
  - Predicts weather patterns and conserve total water usage

Design Methodology

Attribute Table				
Characteristic	Objective	Constraint	Function	Means
Hardware				
Measures Environmental Conditions			✓	
Accurate	✓			
Expandable and Modular	✓		✓	
Waterproof / Weatherproof		✓		
Inexpensive	✓	✓		
Wireless Communication & Power		✓		✓
Easy to Set Up Outside	✓	✓		
Low Power Consumption		✓		✓
Software				
Easy to Use, Intuitive UI	✓			
Saves Historical Data			✓	
Shows Real-Time Conditions			✓	
Links to Recommended Growing Conditions			✓	
Predicts Weather			✓	
User Configurable	✓		✓	

A Strategy of Design Decomposition was used for development

Technology Selection

1. Wireless Technology:	XBee and IEEE 802.15.4
2. Microcontroller:	ATMEGA328p
3. Programming Language:	Embedded C, Python
4. Soil Sensors:	Vegetronix VH400 & THERM200
5. Environment Sensors:	Seedstudio DHT11 & ALS-PT19

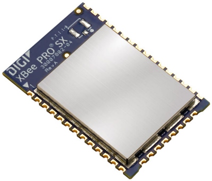
Wireless Technology

XBee Radios

- Portable and easy to configure
- Automatic network association
- Sleep mode for minimum power usage

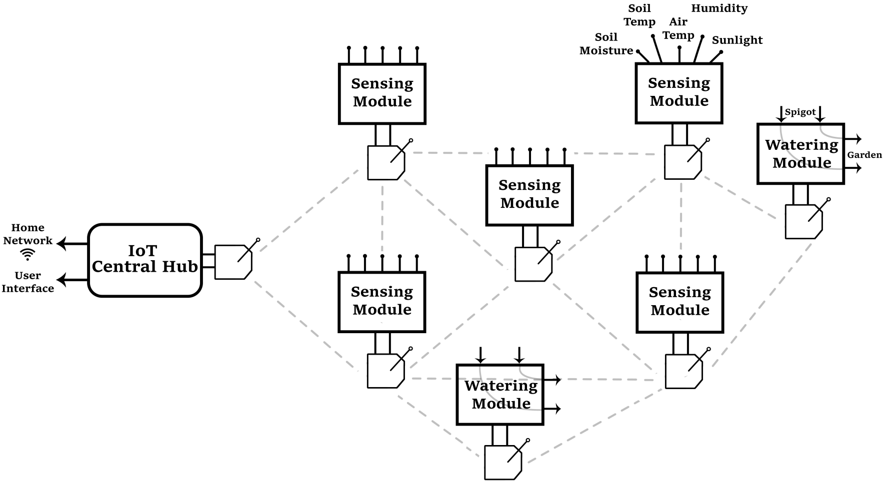
IEEE 802.15.4

- Plenty of available channels
- Point-Multipoint and P2P topologies
- Easy-to-manage addressing



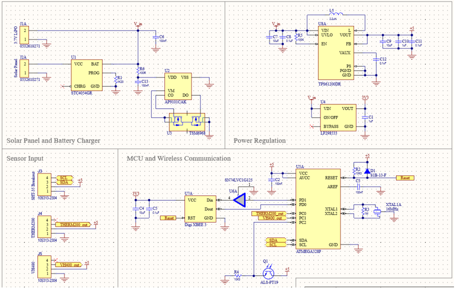
Wireless Technology

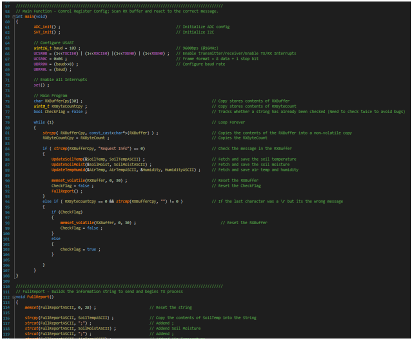
Diagram.jpg



Sensor Module Hardware Design

- Radio and sensor functions were programmed, prototyped on arduino, and tested individually
- Individual components were brought together to create a complete embedded program and hardware schematic

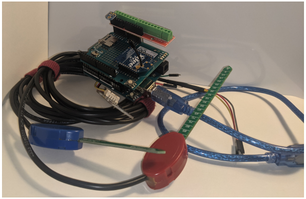




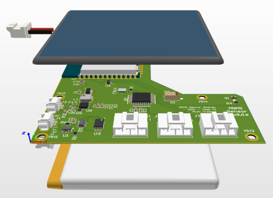
```
///////////////////////////////////////////////////////////////////////////////////////
// Main Function - Conrol Register Config; Scan RX buffer and react to the correct message.
int main(void)
{
        ADC_init() ;                                            // Initialize ADC config
        SHT_init() ;                                            // Initialize I2C

        // Configure USART
        uint16_t baud = 103 ;                                   // 9600bps (@16MHz)
        UCSR0B = (1<<TXCIE0) | (1<<RXCIE0) |(1<<TXEN0) | (1<<RXEN0) ;  // Enable transmitter/receiver/Enable TX/RX Interrupts
        UCSR0C = 0x06 ;                                         // Frame format = 8 data + 1 stop bit
        UBRR0H = (baud>>8) ;                                    // Configure baud rate
        UBRR0L = (baud) ;

        // Enable all Interrupts
        sei() ;

        // Main Program
        char RXBufferCpy[30] ;                                  // Copy stores contents of RXBuffer
        uint8_t RXByteCountCpy ;                                // Copy stores contents of RXByteCount
        bool CheckFlag = false ;                                // Tracks whether a string has already been checked (Need to check twice to avoid bugs)

        while (1)                                               // Loop Forever
        {
            strcpy( RXBufferCpy, const_cast<char*>(RXBuffer) ) ;     // Copies the contents of the RXBuffer into a non-volatile copy
            RXByteCountCpy = RXByteCount ;                      // Copies the RXByteCount

            if ( strcmp(RXBufferCpy, "Request Info") == 0)      // Check the message in the RXBuffer
            {
                UpdateSoilTemp(&SoilTemp, SoilTempASCII) ;      // Fetch and save the soil temperature
                UpdateSoilMoist(&SoilMoist, SoilMoistASCII) ;   // Fetch and save the soil moisture
                UpdateTempHumid(&AirTemp, AirTempASCII, &Humidity, HumidityASCII) ;  // Fetch and save air temp and humidity

                memset_volatile(RXBuffer, 0, 30) ;              // Reset the RXBuffer
                CheckFlag = false ;                             // Reset the CheckFlag
                FullReport() ;
            }
            else if ( RXByteCountCpy == 0 && strcmp(RXBufferCpy, "") != 0 )   // If the last character was a \r but its the wrong message
            {
                if (CheckFlag)
                {
                    memset_volatile(RXBuffer, 0, 30) ;          // Reset the RXBuffer
                    CheckFlag = false ;
                }
                else
                {
                    CheckFlag = true ;
                }

            }
        }
}

///////////////////////////////////////////////////////////////////////////////////////
// FullReport - Builds the information string to send and begins TX process
void FullReport()
{
    memset(FullReportASCII, 0, 28) ;                // Reset the string

    strcpy(FullReportASCII, SoilTempASCII) ;        // Copy the contents of SoilTemp into the String
    strcat(FullReportASCII, ";") ;                  // Addend ;
    strcat(FullReportASCII, SoilMoistASCII) ;       // Addend Soil Moisture
    strcat(FullReportASCII, ";") ;                  // Addend ;
```

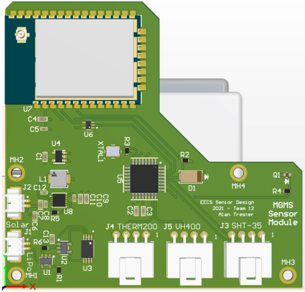
Sensor Module Hardware Design

Prototype



Final Hardware





Software Design

Implementing the Design Methodology - Decomposition

- Software was broken down into smaller parts that were then blended together.
- Each portion was designed then tested before being combined.
- Software development was broken into these stages
    1. Serial Communication and Radio Scanning
    2. Raw Data Transmission
    3. Data Parsing
    4. Data Storage
    5. Data Graphing
    6. UI Functionality

System Demo

UI Functionality
and
Typical Workflow

Budget

Prototyping BoM Rev. 10/13/2020				
Description	**Mfr. Part Number**	**Unit Price**	**Quantity**	**Notes**
Arduino ATMega328p Prototyping Board	A000066	$22	3	ATSAMD21G18 or MSP430 are also options
Temperature/Humidity Sensor	SHT35	$14.18	1	
Vegetronix Soil Moisture Sensor	VH400	$39.95	1	
Vegetronix Soil Temperature Sensor	THERM200	$33.95	1	
I2C Light Sensor	VEML7700	$10	1	Assess which light sensor will be used in final design
Analog Light Sensor	ALS-PT19	$7.95	1	
CO2 Sensor	CCS811	$20	1	For indoor use but will be assessed in prototyping
XBEE S1 PRO 802.15.4 Radio	XBP24-AWI-001J	~$20	2	Obsolete but Already Owned
Raspberry Pi 3 Model B+	N/A	$35	1	
	Total:	**$267**		

Timeline

2020
2021
Aug
Sep
Oct
Nov
Dec
Jan
Feb
Mar
Apr
Definition
100% complete
Rough system diagram
100% complete
Final preliminary system design
100% complete
Bill of Material
100% complete
Design
100% complete
Order parts
100% complete
Firmware development
100% complete
Front end development
100% complete
Control system development
100% complete
Additional hardware
100% complete
Implementation
100% complete
Finalize and order PCB
100% complete
System Prototype
100% complete
System assembly
100% complete
Finalize UI
100% complete
Testing
100% complete
Test UI
100% complete
Test Hardware
100% complete
System Testing
100% complete
Delivery
56% complete
Poster
100% complete
Abstract
100% complete
Video Presentation
100% complete
Design Expo
0% complete
TODAY

Conclusion

- Challenges
  - Building and developing project in home office, not in EECS lab
  - Creating a prototype with outdated technology on hand
  - Working and collaborating virtually instead of in-person
- For the Future
  - Create features to control watering and predict weather patterns
  - Stress-test and document the limitations of hardware/software
  - Continue adding more features and hardware/software components
- Final Remarks
  - A prototype that solves our problem was successfully created
  - The MGMS helps people conserve water in their gardens
  - There are possibilities to expand and improve the MGMS in the future