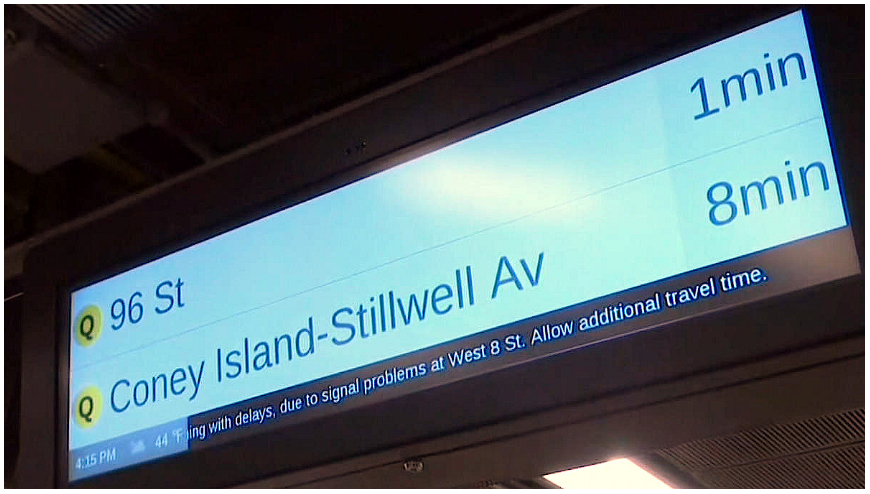
Raspberry Pi Subway Clock

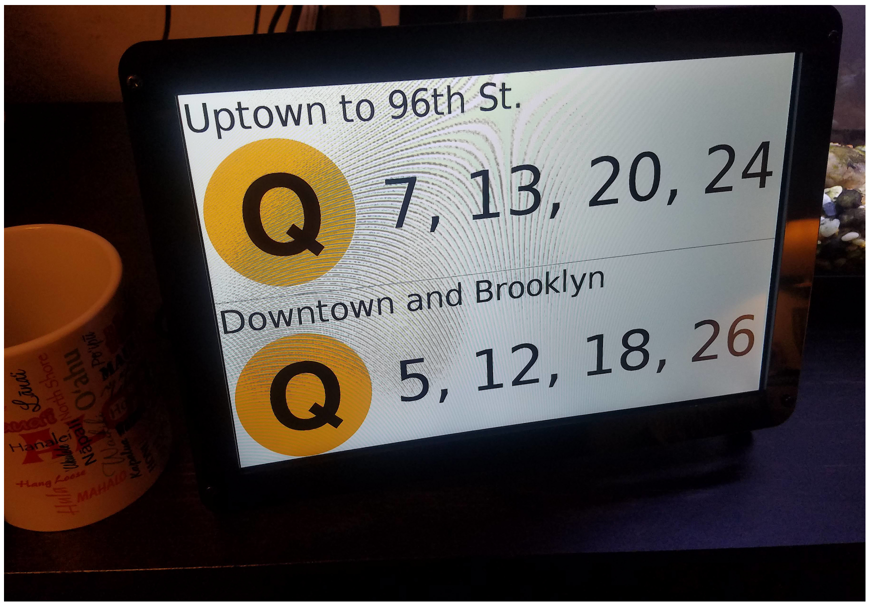
Alan Light

What is it?

Most subway stations in NYC have digital train arrival clocks which display the times of the next train arrivals and look like this:


Q 96 St
1min
Q Coney Island-Stillwell Av
8min
4:15 PM 44 °F
ing with delays, due to signal problems at West 8 St. Allow additional travel time.

Since I live near a subway station, I wanted one of these for my very own in my living room. So I made one. Mine looks like this:



The upper section shows the arrival times in minutes for uptown trains at my station and the lower section shows the same for the trains running downtown.

How it works

It's powered by a Raspberry Pi single-board computer mounted on the back of a 10 .1" monitor running custom software written by yours truly which accesses the MTA's train arrival data feeds and displays the result in a form intended to resemble the signs on the train platforms.

I used a Raspberry Pi Model 3B+ which has 1GB of RAM and a quad-core CPU running at 1.4Ghz, which is clearly overkill for this project. You could likely try it and be successful with a lower-end processor, but keep in mind that even this top-of-the-line Pi only costs $35 at the time of this writing.

How Can You Make Your Own

Please note that this text is a bit terse and assumes a certain familiarity with the underlying technologies.

You might need to fill in some gaps. Google is your friend.

This is a description of a project, not instructions for a turnkey product. You're going to be messing around a little bit with hardware, flashing an OS, installing packages, looking up some stuff, creating shell scripts and tweaking software. Make sure you're up for this before you start spending money on parts. One thing you can do for free (not counting the value of your own time) is get the software all up-and-running and configured on your own PC before you even acquire any of the dedicated hardware for this project. How you specifically approach this is of course totally up to you.

I'm also assuming you're starting from scratch. If you already have a running Raspberry Pi, you can skip many of these steps.

What You'll Need

1. A Raspberry Pi. If you don't want to mess around with adding a wifi adaptor, get one with on-board wifi like the model 3B or 3B+.

2. A computer of some kind (PC/Mac/Linux). These instructions assume you're using a Windows PC, so if you're using something else, you'll need to adjust. You'll only be using this computer for the initial configuration, once it's setup your device will be free-standing.

3. A Raspberry Pi compatible monitor. In my case I used a 10.1" monitor (made by SunFounder) that had a built-in mount for the Raspberry Pi on the back (see pic below). It also has a port and a cable which allows the Pi to be powered from the monitor. This is pretty much an ideal setup for this project, but it's not required. If you're going to buy a monitor for this, **don't** pay the extra cost for a touch-screen, it's not necessary for this application.

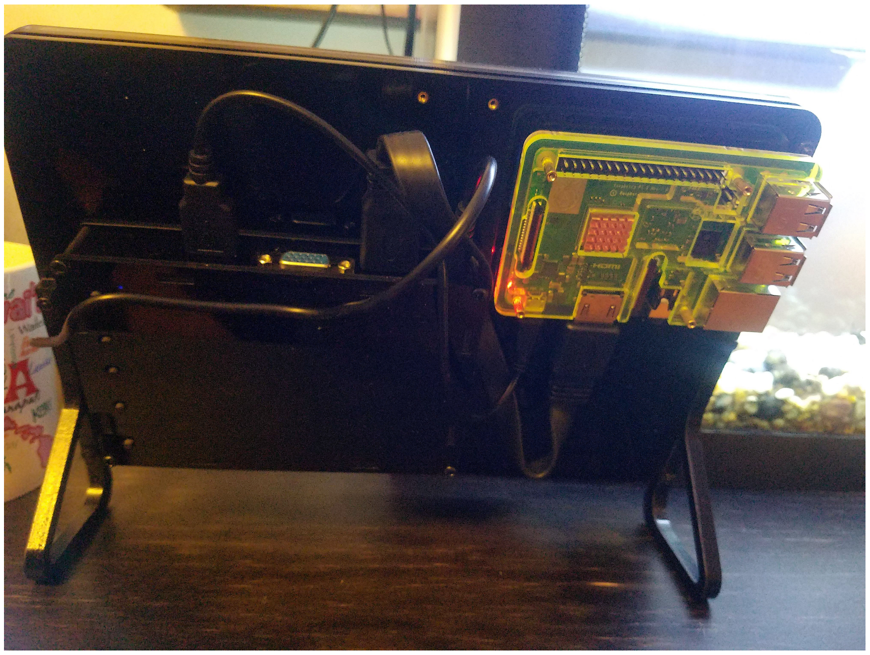
4. A HDMI cable to connect the Pi to the monitor.

5. A class-10 micro SD card. Any that you have laying around should probably be large enough, but if you're going to buy one, you might as well get a 32GB.

6. Either an adapter or card reader to be able to access the micro SD card from your PC.

7. I recommend that you get a case for the Raspberry Pi, even if your final state consists of mounting your Pi to your monitor like I did. You'll want a case for the Pi while you're messing around with it and getting it to work. Get a cheap one.

8. A power supply for your Raspberry Pi



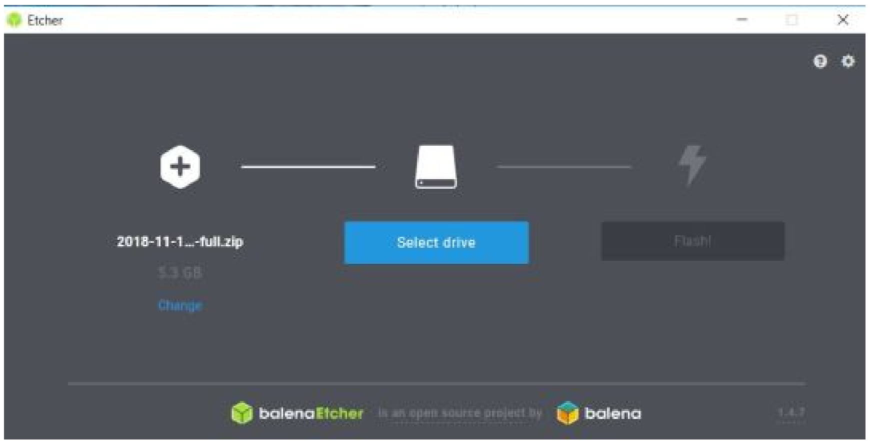
Preparing your Raspberry Pi and allowing headless operation

I recommend you do all the setup work with the Raspberry Pi running headless and only attach the monitor once you pretty much have everything working.

You're going to need to download a Raspbian OS-image from the Raspian official website, `https://www.raspberrypi.org/downloads/raspbian/`. I recommend you use the torrent link as the direct download link can be quite slow. The result will be a ZIP file, don't unzip it. The utility you're going to use to flash the SD card will handle the full ZIP file just fine.

Download Etcher from `https://www.balena.io/etcher/` which will allow you to flash an OS image directly to your SD card from the downloaded OS ZIP file.

Insert your SD card into your PC and run Etcher.



Etcher is about as simple as it could be to use. On the left, select the downloaded zip file with the OS image. In the middle, select the drive location of your SD card and then click the **flash!** button on the right. Note: Obviously everything on your SD card will be deleted by this process.

Once Etcher is done flashing and validating the OS image, you should pull out the SD card from your PC and re-insert it. Once your re-insert, you should see a filesytem called `boot` on your PC. This is just one partition on the SD card, but it's the only one you need for our purposes at the moment.

You're going to create two files on the root directory of this filesystem.

1. An empty file called `ssh` with no suffix. This will enable SSH access to your Raspberry Pi which is normally disabled by default.

2. A file which will setup wifi access. This file should be called `wpa_supplicant.conf` and have the following contents:

```
ctrl_interface=/var/run/wpa_supplicant
update_config=1
country=US

network={
  ssid="YOUR_NETWORK_NAME"
  psk="YOUR_WIFI_PASSWORD"
  }
```

Where `YOUR_NETWORK_NAME` is the name of your wifi network and `YOUR_WIFI_PASSWORD` is your wifi password (remember that both network names and wifi passwords are case-sensitive). Also if you're not in the US, set `country` to the two-character country code for your country. But if you're not in the US, I'm not sure why NYC subway arrival times would interest you. :)

Also, you're going to need to save this file using Unix-style line terminators, so use an editor like Notepad++ or Emacs which will allow you to set this.

When you're done, dismount the SD card from your PC, insert it into your Raspberry Pi and power it up. You should now be able to access it from your PC via wifi.

Accessing your Raspberry Pi from your PC

You'll need to install two applications on your PC.

1. A SSH client such as Putty or Bitvise (Putty is simpler and good enough if you're only going to use it occasionally, Bitvise is more full-featured. Both are free). In either case, enable the *X11 Forwarding* option.

2. An X11 server for Windows such as Xming.

See if you can ping your Pi from your PC using the hostname `raspberrypi` which will be your Pi's default hostname. If that doesn't work, log into your wifi router and look at the client table and find the ip address of your Pi and you should be able to ping the ip address.

If you can't see the Pi listed in your router's client table, then it's either not running or not connected to wifi. Re-do and re-check the steps above. If you remount the SD card in your PC and you see the `ssh` and `wpa_supplicant.conf` files you created are gone, you at least know that the system is booting and the problem is likely somewhere in your `wpa_supplicant.conf` file. Unfortunately, to repeat the process, you'll need to re-flash the SD card from the beginning.

Once you can ping your Pi, Fire up your X11 server, fire up your SSH client and connect. Login using the user name `pi` and the password `raspberry`.

Once you're successfully logged on, change your password from the default using the `passwd` command.

Setting the Time Zone

You'll need to set the Raspberry Pi to the correct time zone. Type `sudo tzselect` and follow the prompts.

Updating your software

Make sure your Pi has all its software up-to-date by entering the following two commands:

```
sudo apt-get update
sudo apt-get upgrade
```

This could take a few minutes to update all your packages.

Making sure X11 is working and displaying back to your PC

Try running `x-terminal-emulator` and see if a new terminal window for your Pi opens on your PC, if it doesn't either xMing is not running and/or your SSH client is not set for X11 forwarding.

Changing Your Hostname

You Pi will come with a hostname of `raspberrypi` by default. You'll likely want to change this. Do so by running

```
sudo raspi-config
```

You will be able to change the host name under *Network Options.*

Installing the necessary Python packages

There are a few Python packages that you are going to need to install. Since we will be using Python 3.x, you can use `pip3` to install them all.

```
pip3 install gtfs_realtime_bindings
pip3 install protobuf3_to_dict
```

Create a directory on your Raspberry Pi for the Subway Clock Software

Create a directory called `subway` to hold your software and a subdirectory to hold the artwork:

```
mkdir subway
mkdir subway/icons
```

The resulting directories should be `/home/pi/subway` and `/home/pi/subway/icons`.

Get your MTA API Key

You'll need your own MTA developer's API key in order to access realtime MTA data. You can request one here: `http://mtadatamine.s3-website-us-east-1.amazonaws.com/#/AccessKey`. Once you get your key, put it in a file called `/home/pi/subway/apikey.txt`.

Setup your Subway Software

Copy the files `subway.py` and `mta.py` to your `/home/pi/subway` directory. You can use SFTP from your PC to do this either from the PC's command line or using a client like Filezilla.

Make `subway.py` and `mta.py` executable.

Test your connection to the MTA API and configure for Your Station

Run the following:

```
cd /home/pi/subway
./mta.py
```

You should see output which looks something like the following:

```
(['Q', 'Q', 'Q', 'Q', 'Q'], [2, 10, 22, 32, 40], ['Q', 'Q', 'Q', 'Q'], [5, 13, 21, 29])
```

If you get any errors, you'll need to address them.

The output above is the uptown and downtown arrival trains and times at the 72nd St. and 2nd Ave station. You will need to configure and test for your particular station.

Check this list of stations here: `http://web.mta.info/developers/data/nyct/subway/Stations.csv` and find yours.

You're going to need the value of `GTFS Stop ID` column (third column) for your station. For example, if your station is the one at Lexington Av and 86 St, you'll see that the stop ID for this station is `626`.

Note: For a given station, there are actually two IDs (one for the trains running in each direction). Take the base ID (e.g. "626" and add an "N" and an "S" to get the two actual ids to use (i.e. "626N" and "626S"). The MTA uses "N" and "S" (north and south) as directions regardless of which actual compass directions the trains are traveling.

Now, test the API with your station by specifying the stop IDs on the command line to `mta.py`, for example:

```
./mta.py 626N 626S
```

Get Your Artwork

The graphics that represent the subway lines are copyrighted and I can't re-distribute them. You should be able to find versions on your own or make them yourself. To get the best possible look, you should start with vector images and scale them to bitmaps (using Inkscape, Cairosvg or similar software) which are sized correctly for your display. For the 1280x800 monitor that I used, 300x300 PNG images worked the best. The images must be square and ideally 37.5% of the vertical resolution of your display. If they are not the optimal size, the software will scale them (provided they are still square), but the results will not be great. Put your images in the `/home/pi/subway/icons` directory with names such as `R.png` for the **R** train, `Q.png` for the **Q** train, etc. This is all case-sensitive, so make sure the train names are upper-case and .png is lower-case. You will also need an image called `unknown.png` for the edge case where an unknown train comes up.

Test the Software

From the `/home/pi/subway` directory, run `subway.py` and make sure everything is working **before** plugging in your monitor.

Connect Your Monitor

Once you pretty much have it all working, it's time to connect your monitor (to the HDMI port on the Pi). Reboot and do a test to make sure that everything displays ok on the monitor:

```
export DISPLAY=:0
/home/pi/subway/subway.py -f
```

This tries and runs the app in full-screen mode on the attached monitor.

Setup Subway Clock to run on Booting

Create a file called `/etc/xdg/autostart/subway.desktop` which contains:

```
[Desktop Entry]
Type=Application
Name=Subway
Comment=Subway Clock
NoDisplay=false
Exec=/usr/bin/lxterminal -e /home/pi/subway/startsubway.sh
NotShowIn=GNOME;KDE;XFCE;
```

Then create `/home/pi/subway/startsubway.sh` which contains:

```
#!/bin/bash
cd /home/pi/subway
sleep 15
xset s 0 0
xset s noblank
xset s noexpose
while true; do
    /usr/bin/python3 subway.py -f -u Q03N -d Q03S -U "Uptown to 96th St." -D "Downtown and Brooklyn"
done
```

This disables the screensaver and runs the subway clock (and will re-start it should it ever crash).

You should replace the `Q03N` and the `Q03S` with the IDs for your station as well as replace the station descriptions with your own.

You'll also need to make this file executable with a

```
chmod a+x /home/pi/subway/startsubway.sh
```

Reboot the Pi with `sudo reboot` and make sure that it launches the subway clock upon rebooting.

Hiding the Mouse Cursor

We're not going to want to see a mouse cursor on the screen, so install the Unclutter package:

```
sudo apt-get install unclutter
```

Then edit `/home/pi/.config/lxsession/LXDE-pi/autostart` and add the line:

```
@unclutter -idle 0
```

Turning the display on/off at night

If you want to turn the display off during "off hours," Create a script in `/home/pi/subway` called `displayoff.sh` which contains the following:

```
#!/bin/bash
vcgencmd display_power 0
```

Similarly, create a file called `displayon.sh` with the following contents:

```
#!/bin/bash
vcgencmd display_power 1
```

Don't forget to make both these scripts executable. You can now create crontab entries (using `crontab -e`) to call these scripts to enable/disable the display whenever you like. For example, if you want the display to turn off at midnight every night, and back on at 5:30am, your crontab entries would look like:

```
30 5 * * * /home/pi/subway/displayon.sh
0 0 * * * /home/pi/subway/displayoff.sh
```