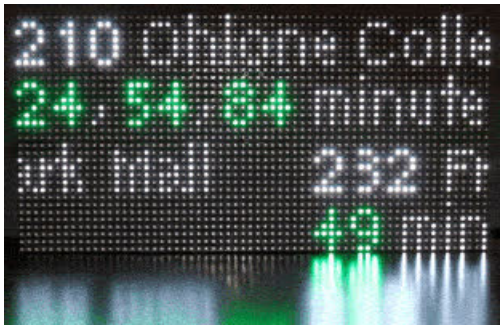
adafruit learning system

NextBus transit clock for Raspberry Pi

Created by Phillip Burgess



Last updated on 2020-11-24 04:19:43 PM EST

Overview

NextBus (https://adafru.it/eCA) is a free internet service using GPS and cellular networks to provide realtime arrival data for 60+ transit agencies in the United States and Canada.

For transit-bound people, the NextBus service is a tremendous convenience. Knowing when a bus is due means less standing out in the rain...one can use that time inside to get a little extra work done, or finish that cup of coffee. (Possibly even a life saver...I have seen people killed by traffic — *literally killed dead* — stepping off the curb to peer down the road for an arriving bus!)

NextBus provides web and mobile phone access, and there are some nice smartphone apps around. As a "heavy user," I wanted to take it one step further, creating a wall clock of sorts...a continuous feed of the stops relevant to my needs...no need to even pull out a phone or click a bookmark, the information's always there at a glance.

Parts and Tools Required

- Raspberry Pi computer — **Model** A+ **(http://adafru.it/2266),** B+ **(http://adafru.it/1914) or** Pi 2 **(https://adafru.it/eCB) only.**
- Adafruit RGB Matrix Pi Hat **(https://adafru.it/eCC)**

- RGB LED matrix **(https://adafru.it/emd):** we have many sizes and pitches (pixel spacing). Our example code is written for a **single 64x32 matrix**, but with some work could be adapted to smaller matrices, or could span across multiple matrices. **This works with "matrix panels" only, *not* NeoPixel matrices!**
- 4GB **(http://adafru.it/102)** or larger microSD card
- USB WiFi module (https://adafru.it/elp) (or Ethernet connection on B+ or Pi 2)
- **5V 10A** switching power supply (https://adafru.it/eCD). If you're using the matrix for this bus project only, this **5V 4A** supply (https://adafru.it/e50) is sufficient...but if you might repurpose the matrix later for projects with more LEDs active, the larger supply is recommended.
- Soldering iron and related soldering paraphernalia
- A monitor, keyboard and mouse are required for initial system setup, but the system can run "headless" once configured.

Our Adafruit RGB Matrix Pi HAT **(https://adafru.it/eCC)** interfaces the LED matrix to the Raspberry Pi. **HATs use a 40-pin header and only work with the Model A+, B+ or Pi 2 — not older models.**

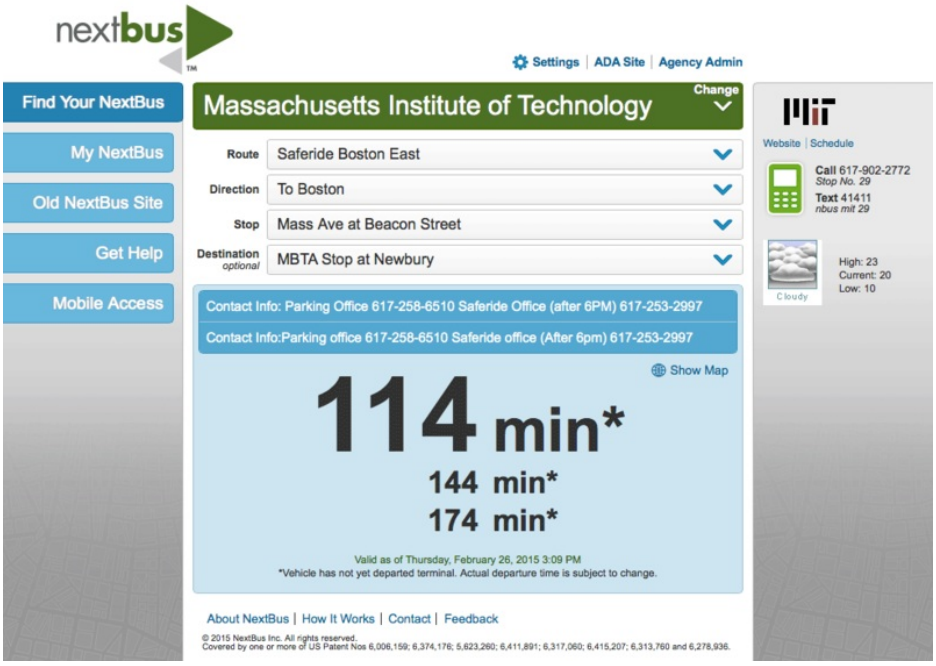
This project is *not* computationally demanding...so if you recently upgraded to a Pi 2 and have an A+ or B+ now sitting idle, this is an ideal project for repurposing "last year's" board.

Realistic Expectations

Before commiting to this project, I'd suggest trying the NextBus service for a couple weeks with your regular web browser and/or on your phone, in order to understand its limitations.

While very convenient and fairly reliable overall, the system is not 100% perfect. Not all vehicles are equipped with working tracking hardware. Sometimes GPS or cell signals are lost and tracking estimates may jump forward or back by several minutes.

Get to know how much lead time you need to safely and reliably make your transit connection, and whether the service meets your needs.



Pi Setup

Download

We'll be installing the Raspbian operating system on the Pi in a moment. If you don't have the latest version downloaded, you can get that started and then work on the hardware bits while it transfers...

https://adafru.it/eCE

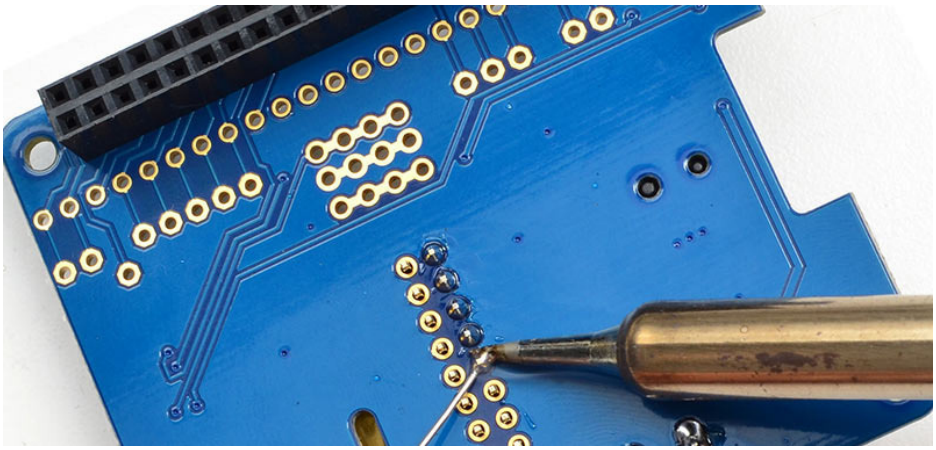
https://adafru.it/eCE

It's about a gigabyte and will take a while. If you don't have access to a fast broadband connection, we do offer Raspbian pre-installed on a microSD card (https://adafru.it/eCF).

After downloading, the OS can be installed on a microSD card using the directions in this guide (https://adafru.it/jd1).

Hardware

Assemble the Pi HAT using the directions in this guide (https://adafru.it/Cfd). This requires a bit of soldering. Make sure you get the HUB75 connector (the 8x2 pin socket) pointed the right way!



Install the HAT on the Raspberry Pi and connect the ribbon cable between the matrix and HAT. There are two sockets on the back of the matrix (to allow "chaining")...but they're not always labeled. If not, look for the arrows showing the direction of data from IN to OUT.

With the HAT installed, USB power for the Pi is usually not required, unless you have power-hungry USB peripherals connected. For most situations the single 5V barrel jack on the HAT will work for both the matrix and Pi.

System Configuration

Insert the microSD card containing the Raspbian OS and plug a monitor and keyboard into the Raspberry Pi. (The monitor and keyboard are only used during setup — later the system can be run "headless.")

On first boot, the raspi-config utility will run automatically. Select the following options:

- **Expand Filesystem**
- **Change User Password**
- **Internationalization Options:**
  - **Change Locale**

    - **Change Timezone** and...
    - **Change Keyboard Layout** to suit your language and location preferences
- **Advanced Options**:
    - **Hostname** — assign the system a unique name (e.g. “nextbus”) to distinguish it from other Raspberry Pis on the network
    - **Memory Split** — since we’ll be using the system “headless,” this can be set to 16 MB to maximize the RAM available to the CPU
    - **SSH** — again, because it will be a headless system, enabling this will help later when administering the system remotely via network
    - **I2C** can optionally be enabled if you plan to use the RTC feature of the Matrix HAT (or example code doesn’t use it, but maybe you have projects in mind).

Tab to the “Finish” button and reboot the system when prompted.

Configure Networking

If using a wired Ethernet connection, there’s nothing to do here...skip to the next heading below.

The easiest way to get wireless networking set up is to type “startx” and then use the mouse-driven visual tool with a nice GUI — the icon is right on the desktop. This guide (https://adafru.it/jd2) explains wireless network configuration, including command-line (keyboard-only) options.

It’s a little more challenging with a Model A+ Raspberry Pi, with only a single USB port. One option there is to use a powered USB hub during setup, so you can have WiFi, keyboard and mouse all connected. An alternative, if you have a Model B+ board available, is to move the micro SD card over and do the configuration on that system, then move the card and USB WiFi adapter back to the A+, which should now be able to boot headless and connect to the network.

Don’t continue until you have a system that boots and can connect to the internet (e.g. ping adafruit.com).

Install Prerequisite Software

Once WiFi is active (or you’re using Ethernet), log in and enter the following to install some prerequisite packages:

```
sudo apt-get update
sudo apt-get install netatalk python-dev python-imaging
```

netatalk enables *Bonjour networking,* so the system is accessible as “nextbus.local” (or whatever hostname you assigned) on your network instead of just an IP address number.

python-dev and **python-imaging** are necessary packages for using the RGB LED matrix.

Download and Install Matrix and NextBus Software

```
cd
git clone https://github.com/adafruit/rpi-rgb-led-matrix
cd rpi-rgb-led-matrix
make
cd ..
git clone https://github.com/adafruit/Adafruit-NextBus
cd Adafruit-NextBus
ln -s ../rpi-rgb-led-matrix/rgbmatrix.so .
```

You can test the matrix and network connection with:

```
sudo python nextbus-matrix.py
```

On the next page we'll configure this software for our own bus routes of interest.

If there's no response from the LED matrix...

Try one of the test programs in the rpi-rgb-led-matrix directory, for example:

```
sudo ./led-matrix -D 0 -c 2
```

If this throws up an error message, something's not correctly installed or compiled. Check all of the steps above.

If the software runs without complaint but nothing appears on the LED matrix, check the following:

- Are the power wires connected to both the matrix and the Pi HAT with the correct polarity? (Red to +, black to –)
- Check for the green power LED on the Pi HAT.
- Is the ribbon cable connected to the INPUT socket on the matrix? It's not always labeled...check for the arrows showing the direction of data from IN to OUT.

If everything looks OK but won't run, check the Adafruit Forums (https://adafru.it/cer)...you can search to see if similar topics have come up before, or post a new thread. A description of the symptoms, steps you've tried, and some clear photos of the wiring are all helpful for troubleshooting.

Optional: Shutdown Button

Linux systems don't like to have the power cord yanked. If you need to do maintenance on the hardware, the system should be cleanly halted first...tricky if there's no keyboard and monitor attached. There are a couple of options:

- Network login via **ssh**, then use the **shutdown** command.

or

- Add a momentary pushbutton (http://adafru.it/1489) between an unused GPIO input and GND, then install software to initiate a shutdown when pressed.

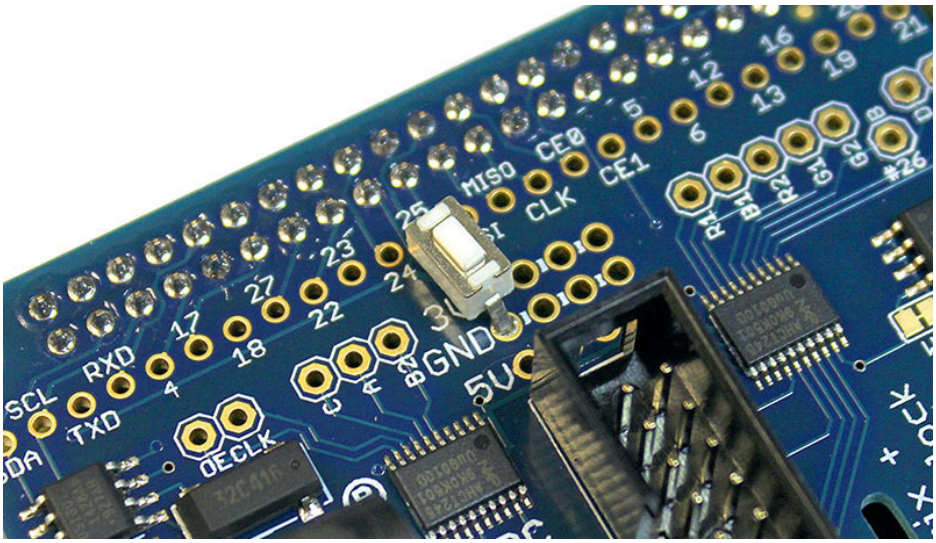
```
cd
git clone https://github.com/adafruit/Adafruit-GPIO-Halt
cd Adafruit-GPIO-Halt
make
sudo make install
sudo vi /etc/rc.local
```

(You can substitute your editor of preference instead of 'vi' on that last line)

Before the final 'exit 0' line, insert this additional line:

```
/usr/local/bin/gpio-halt 25 &
```

GPIO pin #25 is available and is close to the extra GND connections on the Matrix HAT. Solder a momentary pushbutton between those two points. If installed in a case, run wires to an exterior button.



A few other pins could be used instead, as listed in the RGB Matrix HAT guide (https://adafru.it/Cff). Change the pin number passed to gpio-halt in /etc/rc.local to match.

Reboot to activate the changes in rc.local.

```
sudo reboot
```

You should now be able to tap the button to initiate a clean shutdown. Allow 15-20 seconds for this sequence to complete before disconnecting power.

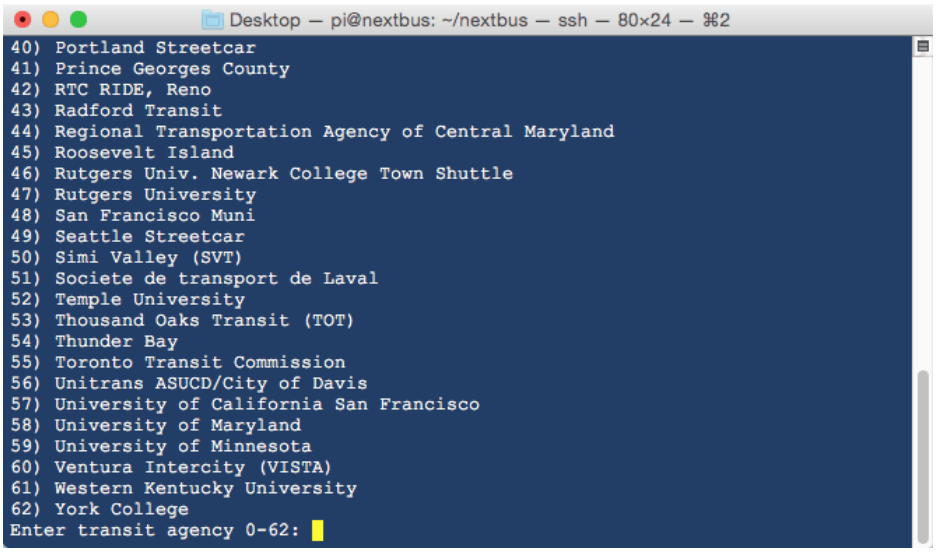
Software Config

With the hardware and code tested, let's now configure the system for your personal needs. You probably have a short list of bus stops and routes that are particularly relevant from your house, office or hackerspace.

The NextBus servers use a series of special "tags" (unique identifier strings) for naming transit agencies, routes and stops. The **routefinder.py** script (in the Adafruit-NextBus directory) helps uncover the correct tags and outputs them in a format that's easily copied into other scripts for display.

```
python routefinder.py
```

Since it's used just a few times for setup, routefinder.py isn't very glamorous to look at...simply text-based with numeric prompts. The lists it displays are sometimes long, so it's best run from a terminal with scroll-back capability.



You'll be prompted for a transit agency, route numer, direction and stop. The script then spits out a line of text similar to this:

```
COPY/PASTE INTO APPLICATION SCRIPT:
  ( 'actransit', '210', '0702640', 'To Ohlone College' ),
```

Copy and paste the resulting line (in parenthesis) into the nextbus-matrix.py script. You'll see there's a list of routes near the top of the program:

```
stops = [
  ( 'actransit', '210', '0702640', 'Ohlone College' ),
  ( 'actransit', '232', '0704440', 'Fremont BART'   ),
  ( 'actransit', '210', '0702630', 'Union Landing'  ),
  ( 'actransit', '232', '0704430', 'NewPark Mall'   ) ]
```

Run routefinder.py again for each stop and route you want arrival times for, copying the output into this list.

The fourth element on each line can be edited for brevity, to use less space on the LED matrix. For

example, ‘To Ohlone College’ could be shortened to ‘Ohlone College’ or even just ‘College’ if that’s sufficiently descriptive for your needs. The other elements on the line should NOT be edited though…the NextBus server expects these exactly as-is.

Below the route list you’ll see some other configurable settings…colors, number of predictions to show, etc. Comments accomanying each item explain their purpose. You might want to set a value (in minutes) for *minTime* — any arrivals sooner than this simply aren’t displayed — to discourage impossible or unsafe rushing for a bus. Recall the story about bus-seekers getting hit in traffic…better to wait for the next one than run across the road to make a connection.

Once configured, test it out by running the nextbus-matrix.py script. Initially all lines will show “No Prediction,” but one by one they should start appearing as server queries are made.

“No Prediction” is also shown when route data isn’t available — either a network error, or simply that the route isn’t running at this time, perhaps late at night or on weekends, depending on your location.

Auto-Start

WIth the script now configured for our routes, let’s set it up to start automatically when the system boots.

Edit the file **/etc/rc.local**:

```
sudo vi /etc/rc.local
```

(Substitute editor of choice instead of vi)

Insert this line before the final “exit 0” line:

```
python /home/pi/Adafruit-NextBus/nextbus-matrix.py &
```

Change the path if the script is located somewhere else.

Notice that you don’t need “sudo” at the start of this line…commands in rc.local are already run as root.

Reboot to confirm that everything works and starts up automatically, and the halt button (if installed) does its job.

Once it's all configured and tested to your liking, you can put the hardware on a shelf or design a case or bracket to hold everything on the wall. This is the "arts and crafts" part of the project...every installation is different, so we don't have a specific design to recommend...you'll need to come up with something that meets your personal aesthetic tastes and available tools and crafting skills.