**Part 1:** Create a Linux 8G uSD Boot Disk and Boot Raspbian OS on the Pi (15%)

**Lab Demo:** Show the TA that you have a Linux command line (or GUI) running on the Pi

**Additional Details:** The first step is to create a NOOBS (New Out Of Box Software) boot disk by downloading a large zip file with NOOBS (not NOOBS lite!) with the RASBIAN disk image file to the 8G (or larger) uSD card using your PC. A 4GB SD card will work with the “Lite: OS image, but it does not include the GUI. [It can be added, but care is needed for only 4GB](https://www.raspberrypi.org/forums/viewtopic.php?f=66&t=133691) of storage and it is a lot more work. The OS setup is also possible without a monitor (i.e., headless), so two setup options are provided in the instructions that follow. It is worth the small price difference for the faster Class 10 uSD cards and the 16G ones do not cost much more than 8G. The mbed starter kit has a tiny white USB to uSD card adapter that can be used on any PC, or the SD to uSD adapter can be used on PCs with an SD slot. New mbed starter kits now come with 16GB micro SD cards that can be used (older ones are 4GB). This lab should also work using a Pi 3, since it has the same processor, OS, and I/O setup as the Pi Zero W.



A Sparkfun uSD to USB adapter can be used to write the uSD card on a PC

Install and initial setup of Linux takes some time and patience, but the good news is that it only needs to be done once.

If the uSD card has been used, reformat it using the [SD card association formatter](https://www.sdcard.org/downloads/formatter_4/index.html) (it works better than the OS tools). The OS requires a very large download file(1.4G) and might take an hour, so make sure the network connection used is fast! While it is downloading, read ahead to work on the cable setup for the Pi.

.

After the NOOBS file is downloaded, unzip to a new empty directory, and copy it all to the uSD card using the PC, remove it from the PC and plug it into the Pi board. [Etcher](https://learn.adafruit.com/introducing-the-raspberry-pi-zero/making-an-sd-card-using-a-windows-vista-slash-7) works better for making uSD cards on the PC, it unzips and copies files in one step.

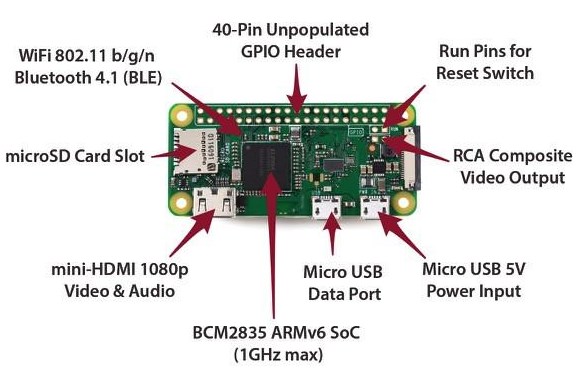
NOOBS prompts the user to setup several things, if you just install the OS distribution directly on the SD card, some OS configuration steps may be needed later for audio, Wi FI, and keyboard country. You can use Raspi-config at any time. To start it, enter the following at the Linux command line:

sudo raspi-config

It is also possible to directly edit the config file with the command below, but you need to know what you are doing!

sudo nano /boot/config.txt

**Cables:** There are many ways to hook up all the cables to the Pi, some are costlier than others, require even more cables, and longer cables. Pis are available in “starter kits” with cables, but they don’t always include the best options and all the cables may not be needed. Very few starter kits include anything to connect to the I/O header pins. The cables suggested here minimize the cable mess, cost and minimize the breadboard space used. The image below shows the location of the main I/O, USB, uSD, HDMI, Camera, and power connectors.



Raspberry Pi Zero W I/O and Connector Layout

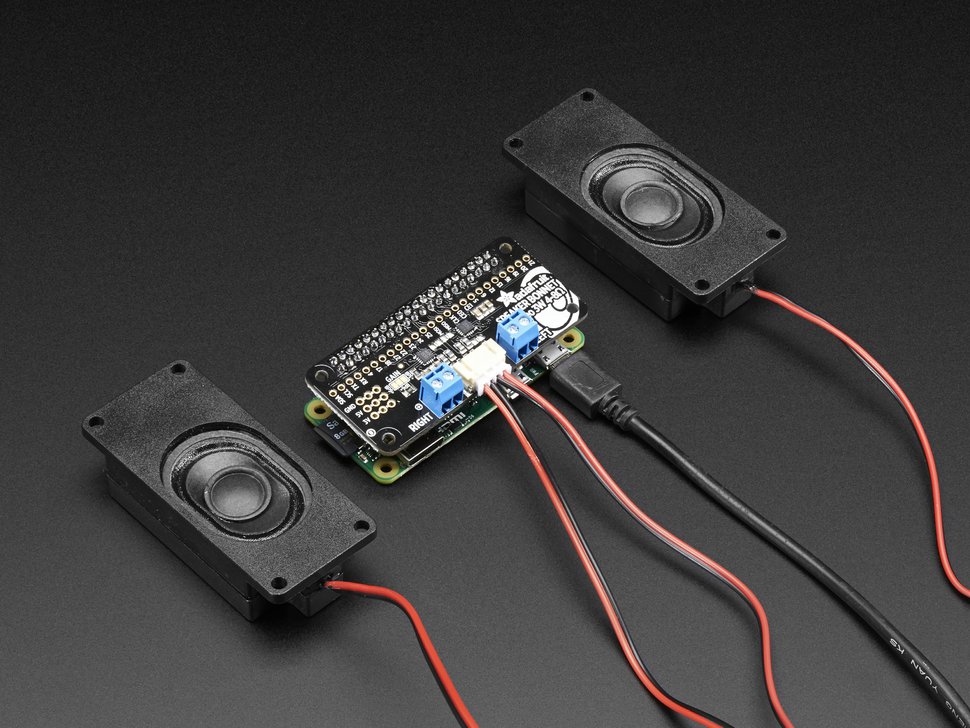
In the photo above, a typical cable setup is shown to use the Pi’s Pixel GUI with an attached monitor, keyboard, and mouse. Note the SD card on the right side and the white Pi camera connector on the left side. On the top left, is the 5VDC power cable. A 5VDC 2A AC power supply adapter with a barrel jack output is plugged into a barrel jack to micro USB adapter (many starter kits have an AC adapter with a micro USB output connector which would not require the barrel jack adapter). In the top center, a USB mini keyboard mouse combo wireless dongle is plugged into a tiny micro USB to USB adapter. On the top right, a mini HDMI to HDMI adapter is used to connect to an HDMI monitor cable (the HDMI monitor cable is not currently plugged in for this photo). Note the green activity LED seen in the upper right corner of the board near the mounting hole. Most new TVs also have an HDMI input that could be used.

For breadboard interfacing, a right-angle header socket has been soldered into the Pi’s 40 pin I/O connector PCB pins. The header socket is them plugged into a Pi Cobbler which has pin spacing that allows it to be plugged into a breadboard (in the center area just like an IC).

**Be very careful**, there is no key on the Pi right angle header socket and it is possible to plug the Pi in backwards to the Pi Cobbler board! The micro SD card should be above the PCB over the tiny “Pi Cobbler” silkscreen label. This setup allows access to all of the Pi’s GPIO and power pins on a breadboard. The GPIO pins can have different functions. In addition to GPIO pins, they support I2S digital audio, I2C, SPI, one PWM pin, and one Serial Port. Failure to do this could result in **5volts blowing up Pi I/O pins in some setups!**

**Caution:** The connector housing on the Cobbler is a bit larger than the right-angle header. It is possible to plug in the Pi with one row of exposed pins at either end of the Cobbler. This can short the power supply! Look down from the top, and double check that it is in the middle and not shifted over a pin.

There are also various Pi Hats (or Bonnets) that allow another PCB to plug into the Pi’s 40-pin I/O header. These can provide additional I/O features without requiring a breadboard. If all of the hat board’s features are needed, they may be a worthwhile option to consider. The Adafruit stereo bonnet adds [I2S](https://en.wikipedia.org/wiki/I%C2%B2S) high quality digital audio output to the Pi. I2S is a industry standard used to send PWM audio data between ICs. It has a clock and 16-bit data for two channels.

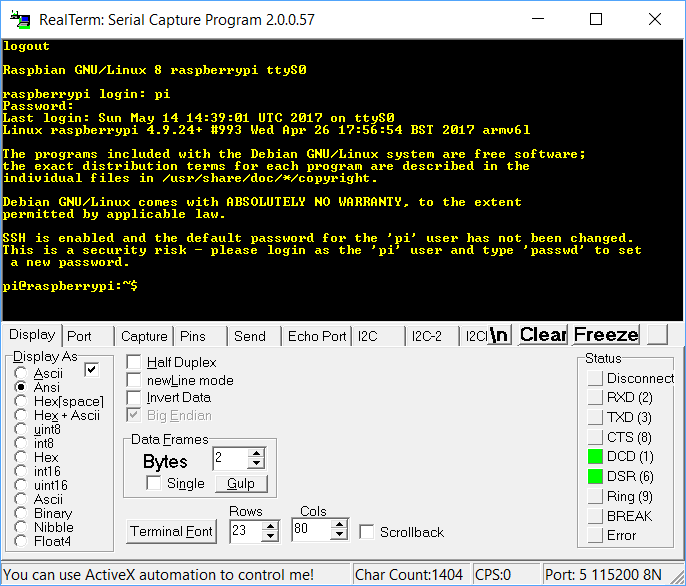


[Adafruit Stereo Pi Bonnect with Speakers](https://www.adafruit.com/product/3346)

The [Pi camera](https://www.raspberrypi.org/learning/addons-guide/picamera/) can also be added using the industry standard camera serial interface ([CSI](http://www.petervis.com/Raspberry_PI/Raspberry_Pi_CSI/Raspberry_Pi_CSI_Camera_Interface.html)) connector provided on the right side of the board. With 8 megapixels, the camera is capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60, and 640x480p90 video. There are a number of [applications and projects](https://learn.adafruit.com/search?q=pi%20camera&) posted on the web using the camera. Pi cases that include camera mounts and shorter flex ribbon cables to fit inside small cases are also available.



**OS Setup Option 2 Headless (a bit more work):** If an HDMI monitor, [a mini HDMI to HDMI adapter](https://www.adafruit.com/product/2819) mouse and keyboard are not available on the Pi to use the GUI, mbed can be used for “serial console cable” (for free) by hooking up three wires after setting up the OS following the [headless setup instructions at Adafruit](https://learn.adafruit.com/raspberry-pi-zero-creation?view=all) (but install the full and not lite version if you ever want to use the GUI). The console is a (non GUI – text only) command line interface that can be used to watch Linux boot and verify it worked and then type Linux commands. Details are provided at <https://developer.mbed.org/users/4180_1/notebook/using-mbed-for-a-pi-console-cable/> to make mbed a free console cable. This web page also contains additional information and an overview that will be useful in many parts of the lab, so be sure to read through it carefully. It also shows several options to supply 5V power to the Pi board. The default username is **pi** and the password is **raspberry**.



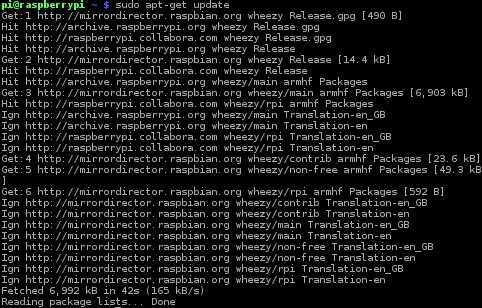
Command Line Linux running in a terminal application window on a PC using mbed for a console cable

**Power:** The USB supply in the Pi kit is only 1A**.** Each USB 2.0 device is allowed to use up to 500MA, so more power is needed whenever more USB devices are plugged in.The 5V 2A AC wall adapter in the 4180 kit could also be used when the Pi is mounted on a breadboard with the PCB mount barrel jack adapter in the kit used earlier (be careful it sometimes comes lose from the breadboard) by connecting wires to the Pi’s 5V and gnd header pins. Adafruit also makes a very handy [barrel jack to micro USB adapter plug](https://www.adafruit.com/product/2789) which could be used to plug power directly to the Pi’s micro USB power connector. A micro USB power only cable from the PC is another option (when no other USB devices are added to the Pi). A phone or tablet charger with a micro USB output might also work. The green LED on the Pi is an activity LED that turns on during boot. To see the green LED come on, the Pi needs both power and a valid uSD card inserted that starts to boot.

Even if the GUI option is used, a command line terminal window can be opened in the GUI. Some things in Linux will still require use of the command line. The Pi can also be setup to send serial data or the console setup to run on the micro USB data cable hooked to the PC (for a headless device - when this is active other USB devices cannot connect to the Pi). This would allow the use of the Pi as a USB gadget for a PC. This [tutorial](https://learn.adafruit.com/turning-your-raspberry-pi-zero-into-a-usb-gadget/ethernet-gadget?view=all) has the details. [SSH](https://thepihut.com/blogs/raspberry-pi-tutorials/17593900-remote-access-the-command-line) can also be used with Wi Fi.

**SD Backup/Copy/Clone**: Once the OS boots, things change on the micro SD card. Once everything is installed and updated, making a disk image or SD backup can save time if something bad ever happens to your SD card. If you move the SD card back to the PC, many of the files are no longer directly accessible on the PC. Once you have a micro SD card that boots and works, there is a handy SD card copy program in the Pi’s GUI. The second SD card can be attached using the USB to micro SD adapter. For reliable SD write operations on Pi Zeros, an external 5V power supply is recommended. Sometimes there are SD write errors, if only a USB cable from a PC is used during constant write operations (limit is 500MA and flash writes require more power than reads). Some USB Hubs may also need power to write micro SD cards reliably. Without a powered USB Hub, it seems to be more reliable, if it is plugged directly into the Pi’s micro USB slot and VNCview is used to control the GUI without any other USB devices attached. Even though it is faster, it takes fifteen minutes or longer to copy an entire SD card. The copy of the SD card is a complete clone that will also boot the OS.

**Part 2:** Configure Wi Fi, install both updates and upgrades using the Wi Fi to access the Internet (15%)



Even the latest version of the OS will need a lot of updates and have some upgrades

**Lab Demo:** Run a *sudo apt-get update* or other command that shows Wi Fi is working. A screen capture with your name typed someplace is acceptable for those working on home networks

**Additional Details:** To install updates and upgrades with “*sudo apt-get*”, the Wi Fi must be configured and connected to a network to access the Internet. Instructions are available at:

<https://www.raspberrypi.org/documentation/configuration/wireless/wireless-cli.md>

<https://www.raspberrypi.org/documentation/remote-access/>

If the GUI option is used with an HDMI monitor, mouse and keyboard, there is a Wi Fi icon in the upper right that can be used to select and configure Wi Fi. There is a web browser in the GUI that can be used for those Wi Fi networks (like GT Lawn) that require additional logon steps beyond the Wi Fi SSID and password once Wi Fi is setup.

On the GT campus, connect to [GTother using the key](https://auth.lawn.gatech.edu/key/) (GTwifi does not work with the default Pi setup) On Wi Fi networks that also require a user name and password (like GTother), open the Chromium web browser to logon (at GT GTLawn) or [use wget for a command line autologon](http://www.lawn.gatech.edu/help/command_line.html) (handy for headless devices). Try the browser to check the network setup (it is a bit slow on complex web pages).

For Eduroam, enterprise Wi Fi authentication needs to be turned on by editing a Wi Fi config file and then rebooting. The current OS cannot do this with the drop down Wi Fi GUI. Here is some info on how to use Raspberry Pi boards with Eduroam:

[**https://normally.online/2017/07/11/how-to-connect-your-raspberry-pi-to-eduroam/**](https://normally.online/2017/07/11/how-to-connect-your-raspberry-pi-to-eduroam/)

using edits at command line or

This video using the GUI to do about the same things:

<https://www.youtube.com/watch?v=z4MDYrLhP_A>

Even though the newest kernel version was just installed, it will still have a lot of available updates and upgrades. Go ahead and do this now once Wi Fi is working. Instructions to update and upgrade the Pi OS kernel are available at:

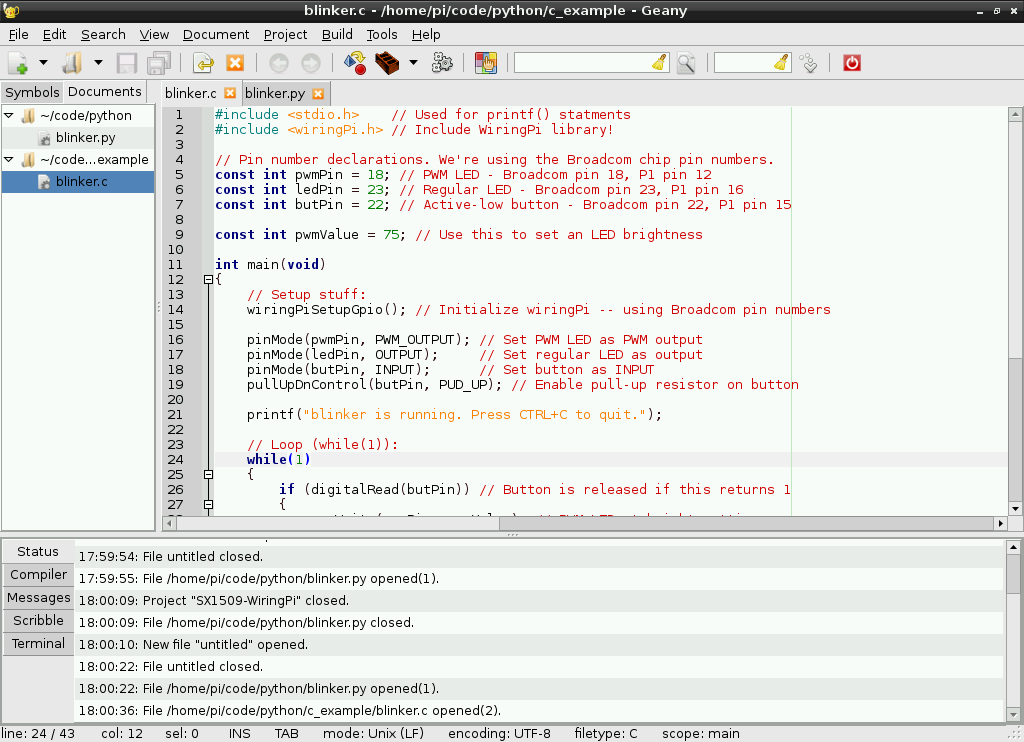
<https://www.raspberrypi.org/documentation/raspbian/updating.md> . It will take several minutes.

If it installs updates and upgrades, that also verifies that the network is working.

If a Pi GUI screen capture is ever needed for documentation, it can be done on the Pi with this [free screen capture utility](https://thepihut.com/blogs/raspberry-pi-tutorials/16018008-how-to-take-screenshots-on-the-raspberry-pi). It seems to be installed already in the non-lite distribution. If VNCviewer is used, the PC can be used to capture the Pi’s GUI window, but if the VNC view window is selected and not the entire PC screen, the screen capture file goes to the Pi’s SD card and not the PC.

Now that the Pi is on a network, consider changing the default password and hostname using “sudo raspi-config” if you have not do so earlier.

**Part 3:** Run the Pi’s GUI over the network using VNCviewer on a PC and compile a hello world C++ program using the Geany IDE (15%)



Geany IDE with C Code running on Pi

**Lab Demo:** Run the C++ hello world program in a terminal window to see output. A screen capture with your name typed someplace is acceptable for those working on home networks.

**Additional Details:** To run the Pi’s GUI around $50 of adapter cables, an HDMI monitor, a USB mouse, a USB keyboard, and a USB hub would be required. The good news is that once the OS and Wi Fi is setup, the GUI can be run on a PC for code development using the network and VNCviewer. For most setups, it is quicker and easier this way, since it avoids the need for all the extra cables and devices. The Pis IP address is needed and the two devices may need to be on the same subnet, if the Pi has a local IP address. VNCviewer will also work on a Phone or Tablet.

First enable VNC on the Pi. Install [VNCviewer](https://www.realvnc.com/download/viewer/) on a laptop (or even a smart phone or tablet) and connect to Wi Fi using GTother on the GT campus (Lab PCs using wired networking will not work – GTother/GTwifi is a different network). Then start VNCviewer using the Pi’s IP address. If the mouse cursor is left over the GUI Wi Fi icon, the IP address will display. The default username is **pi** and the password is **raspberry**. Once it connects, the Pi’s GUI window will appear on the PC. Detailed instructions are available at:

<https://www.raspberrypi.org/documentation/remote-access/vnc/README.md>

To use the Geany IDE for code development, it first needs to be installed. Instructions to install and run the Geany IDE are in the first section of the tutorial at: <https://learn.sparkfun.com/tutorials/raspberry-gpio/using-an-ide> . The wiringpi I/O code library does not need to be installed (stop before that). In the Geany IDE, write a basic C++ hello world program that also prints your name. The program can be as simple as:

**#include<stdio.h>**

**int main()**

**{**

**printf("hello world *myname*");**

**return(0);**

**}**

Save the file as \*.c for C code, or \*.cxx for C++ (this sets the language for syntax coloring and to compile) and then clicking **Build** will compile and link. **Execute** under **Build** or the paper airplane icon will run it and a new terminal window should pop up. The default text output will appear in a terminal window in the Pi GUI.

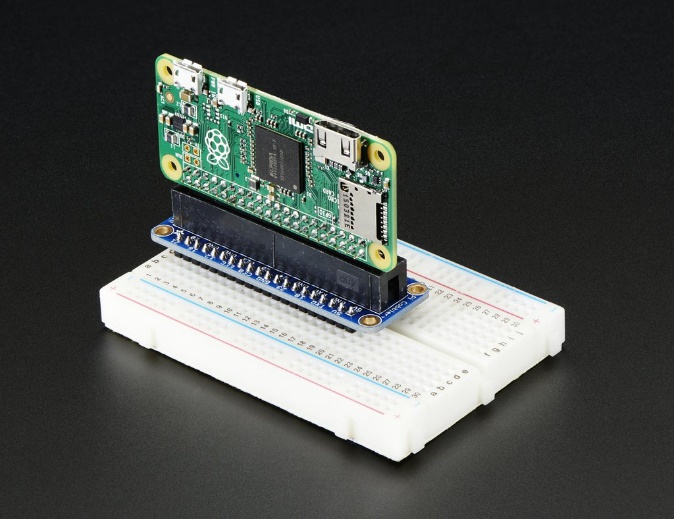
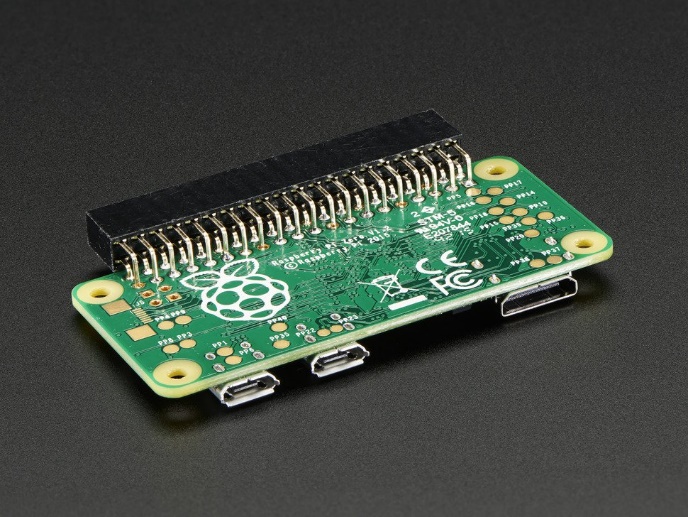
If you ever have trouble seeing the compiled code execute in the terminal window, try **Edit**->**Preferences**->**Terminal Tab** and Check “execute programs in the VTE” and it will execute in the IDE instead of a new window. This sometimes happens in some OS releases in the later steps that run the blinking LED I/O code in the next section, so consider setting this up now and trying it.

A screen capture can be used for checkoff, if you are working at home. On a PC, the entire screen might need to be saved, the VNCviewer window capture goes on the Pi’s SD card, if only that window is selected on the PC.

C/C++ Hello World on Pi GUI using the Geany IDE, terminal window on the right is running code.

While one would expect that VNC over the network would be a lot slower, VNC seems to actually run the GUI a bit faster than a local keyboard, mouse, and monitor setup, since the PC’s more powerful processor is generating the graphics and handling keyboard and mouse I/O.

**Part 4:** Blink an external LED using the PIGPIO library with C/C++ (not Python) code compiled on the Pi (20%)

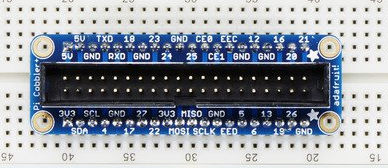


The Pi must connect to a breadboard to use the I/O header pins for the blinking LED demo

**Lab Demo:** Show the TA a blinking LED and the C++ code in the Geany IDE

**Additional Details:** The I/O pins needed are on the Pi’s header pin. Low-cost Pi Zeros come without the header pins soldered in. Headers pins to solder to the Pi and small PCBs to plug pins into a breadboard are available. Most have ribbon cables to connect everything, but the handy setup from Adafruit seen above allows a Pi to plug directly into a breadboard standing up. This one is called the [Pi cobbler](https://www.adafruit.com/product/2029) and it needs a [right-angle header socket](https://www.adafruit.com/product/2823) soldered onto the Pi (hard to find, but they are available from Adafruit). There is also a [T Pi Cobbler](https://www.adafruit.com/product/2028) that can be used to move cables to the top of breadboards when projects require a lot of cabling. The Pis I/O pins are labeled on the PCBs silkscreen and this feature is also handy. Unfortunately, everything needed here costs almost as much as the low-cost Pi Zero W board!

**Once Again, Be very careful**, there is no key on the Pi right angle header socket and it is possible to plug the Pi in backwards to the Pi Cobbler board! The micro SD card should be above the PCB over the tiny “Pi Cobbler” silkscreen label. This setup allows access to all of the Pi’s GPIO and power pins on a breadboard. The GPIO pins can have different functions. In addition to GPIO pins, they support I2S digital audio, I2C, SPI, one PWM pin, and one Serial Port. Failure to do this could result in **5 volts blowing up Pi I/O pins in some setups!**

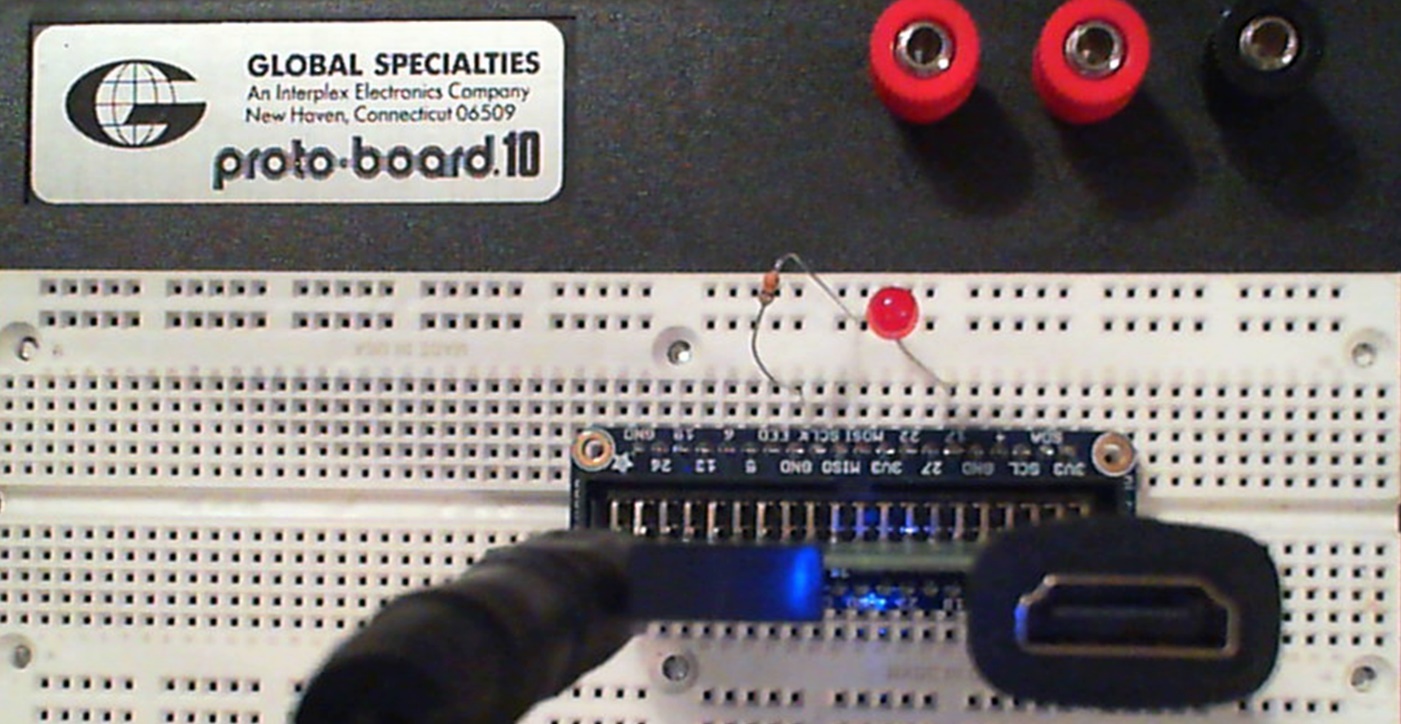


Adafruit right angle female header socket to solder to Pi board and Pi Cobbler used to plug Pi into breadboard.

**Caution:** The connector housing on the Cobbler is a bit larger than the right-angle header. It is possible to plug in the Pi with one row of exposed pins at either end of the Cobbler. Look down from the top, and double check that it is in the middle of the cobbler and not shifted over a pin.

**Caution: Unlike the mbed, the Pi’s 3.3V I/O pins are not 5V tolerant. A 5V input signal could blow out an I/O pin! I/O. Some devices expecting a 5V output signal like a servo may require a 3.3V to 5V voltage level conversion circuit.**

Add the LED to the breadboard. Don’t forget to put the voltage dropping resistor in series with the external LED going from the I/O pin to ground. **Do not connect the LED circuit in anyplace to 5V!** The longer lead on the LED (+) goes to header pin 11 (Pi Cobbler silkscreen says 17 for GPIO bit 17) and the other LED lead (-) connects to a series resistor (around 100-300 ohms) that is connected to ground (any gnd pin on Pi Cobbler). Use only the 3.3(3V3) pin to power breadboard devices to minimize risk to Pi I/O pins blowing out with 5V.



Pi Zero W on breadboard showing LED attached to GPIO pin with a series resistor to ground for the blink demo

There are several GPIO libraries for the Pi. PIGPIO has become the new de facto standard as it now comes installed in the standard OS distribution. Many older Pi projects use the BCM2835, WiringPi, and RPIO libraries instead of the newer PIGPIO. They need to be installed first. In some cases, they no longer compile and ongoing support seems to have stopped for many of them. PIGPIO has similar APIs that can be used instead of the older I/O libraries. In most cases, it is a simple one to one mapping.

PIGPIO has significantly more features and even supports software PWM on multiple pins (others have only one hardware PWM pin). Motors and servos each need a PWM pin, so this is a significant feature especially for robotic’s projects. PIGPIO runs threads and a daemon to do all of this even using DMA for faster software PWM. There are [API documents and code examples available for PIGPIO](http://abyz.co.uk/rpi/pigpio/). PIGPIO can also be used with Python, Java, and Node.js using wrappers.

In the Geany IDE, setup a /home/pi/project directory and then a blink directory in the new project directory. Cut the code below and paste it into the Geany IDE editor and save the file as /home/pi/projects/blink/blink.c. Cut and Paste should work in the VNCviewer, so that the code does not need to be typed in.

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <pigpio.h> // include GPIO library**

**#include <signal.h> // needed to clean up CTL C abort**

**#include <sys/types.h>**

**#include <unistd.h>**

**#define LED 17 //LED pin is GPIO\_17**

**// Called when CTL C or STOP button hit**

**static void err\_handler (int sig){**

**gpioTerminate(); //release GPIO locks & resources**

**signal(SIGINT, SIG\_DFL); //exit program**

**kill(getppid(), SIGINT); //kill it off**

**kill(getpid(), SIGINT);**

**exit(0);**

**}**

**static void exit\_handler(void) {**

**gpioTerminate(); //release GPIO locks & resources on exit**

**}**

**int main(int argc, char \*argv[])**

**{**

**if (gpioInitialise()<0) return 1; // init I/O library**

**signal (SIGQUIT, err\_handler);// CTL C and STOP button**

**signal (SIGINT, err\_handler); // GPIO exit & cleanup**

**signal (SIGTERM, err\_handler);**

**signal (SIGABRT, err\_handler);**

**atexit(exit\_handler); // exit handler cleanup**

**//IO code starts here**

**gpioSetMode(LED, PI\_OUTPUT); // set LED pin to output**

**while(1){**

**gpioWrite (LED, PI\_ON); // LED on**

**time\_sleep(0.25); // Delay .25 seconds**

**gpioWrite (LED, PI\_OFF); // LED off**

**time\_sleep(0.25); // Delay .25 seconds**

**}**

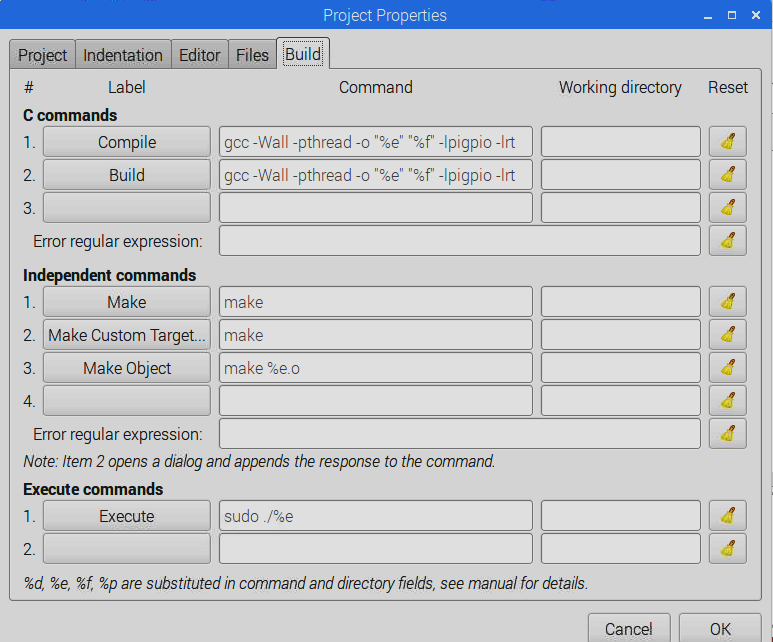
**gpioTerminate();**

**return 0;**

**}**

Once the file is saved as blink.c, the syntax coloring should appear.

The Geany IDE’s default build option command lines for the project will need to be changed in the GUI to link the PIGPIO and Pthreads and RT library when it compiles and to execute code using “sudo”. Note that it just runs the standard Linux C compiler, gcc. Open the window **with Build-> Set Build Commands** and set it up as seen below and **click OK**:



Once the build commands are setup correctly, hit **build** and the code should compile without errors. Then run the code, and the LED on the breadboard should blink as soon as a blank terminal windows pops open in the GUI.

Hit CTL C in the terminal window to stop execution or close the terminal window. The stop button in the Geany IDE does not work for this code. When the program stops, the LED may stay on or be off, depending on where it was stopped. The program can be started again and the LED should start blinking again. Leave the LED connected, it will be used again later in the lab for more interesting things.

Since PIGIO starts a daemon and has threads it needs to be shutdown carefully or things might get locked or keep running. The code example has some ugly signal handlers to respond to program termination signals and attempts to shutdown gracefully (i.e. a CTL C or STOP button). Typing a CTL C in the Terminal window or closing the terminal window shuts down correctly.

Despite this, it is possible you may encounter an error message on program restart during code debugging whenever *gpioTerminate()* is not executed such as “*Can’t initialize…*” or “*Can’t lock…*”. This means the pigpio daemon is already (or still) running. If your own program is acting as the daemon (as in the lab code example) it may be removed as follows. Find its process id (pid).

**cat /var/run/pigpio.pid**

Kill the program with

**sudo kill -9 *pid***

Some other PIGPIO code may start the pigio daemon*.* The default daemon is called *pigpiod* and may be removed as follows*.* Check that it is running with the command

**ps aux | grep pigpiod**

Kill the daemon with

**sudo killall pigpiod**

If the above doesn't work do the following and try starting the daemon again

**sudo rm /var/run/pigpio.pid**

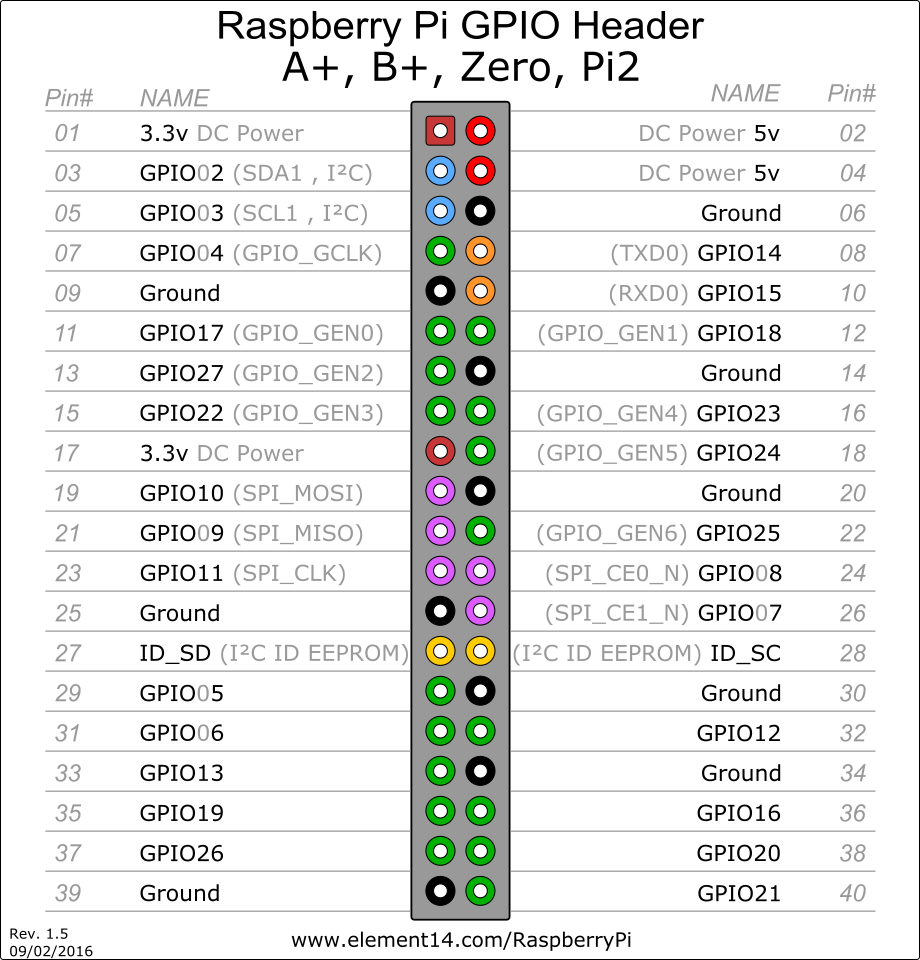
To start the daemon do

**sudo pigpiod**

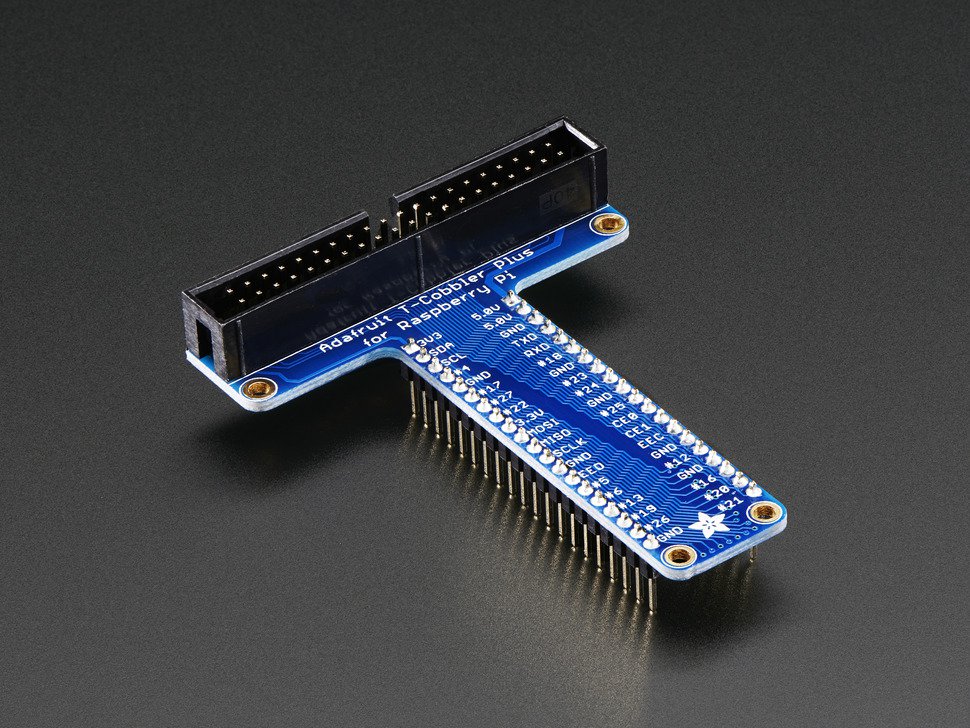
A reboot also fixes it, but commands are a lot faster.

Pigpio code that uses the pigpio deamon alternative, does not need to execute as “sudo” since it calls the daemon for I/O and the daemon is running in the kernel.

A list of Pi header pinouts is provided below. To write code using Pi GPIO pins you need to know which bit number is used on the port. To wire up things, these charts are handy to then find the correct pin number on the header for that GPIO bit. Like the mbed, pins have multiple functions and not all are shown in the diagram. If you have one of the PCB adapters, most of them also have labels on the PCB. I2C pins already have internal pullups. 3.3V power is limited to perhaps 50MA or less.

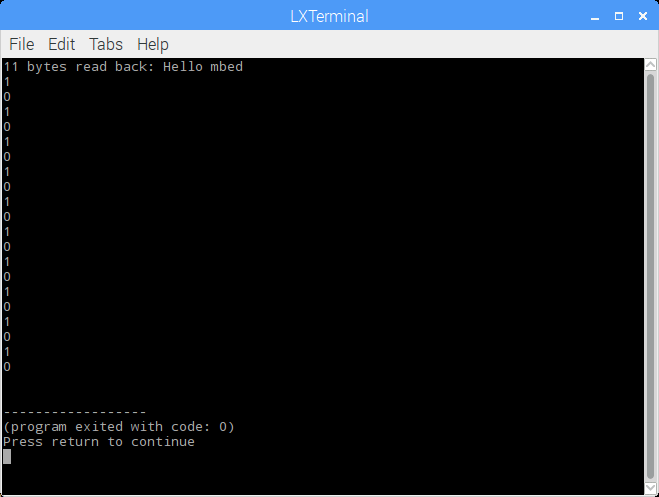


Another option is to use the official [Linux device driver for the GPIO pins](http://hertaville.com/introduction-to-accessing-the-raspberry-pis-gpio-in-c.html). Code using this approach does not need to run as “sudo” since the actual I/O instructions that change pins runs in the OS kernel device driver. This also makes it slower, but using the OS to check and keep track of things makes it safer. Some of the various Linux distributions also do not turn on or can be configured to turn off the hardware protection of the GPIO register memory area and on those “sudo” is not required for execution. This approach is an extra credit option.



A T-style Pi Cobbler can also be used in projects to move the Pi and the cables to the very top of a breadboard. It works out a bit better for Pi 3s since the ribbon cable is moved off of the breadboard.

**Part 5:** Communicate with and control mbed from the Pi using the USB virtual com port (15%)



Pi terminal sending commands to the mbed to control an LED

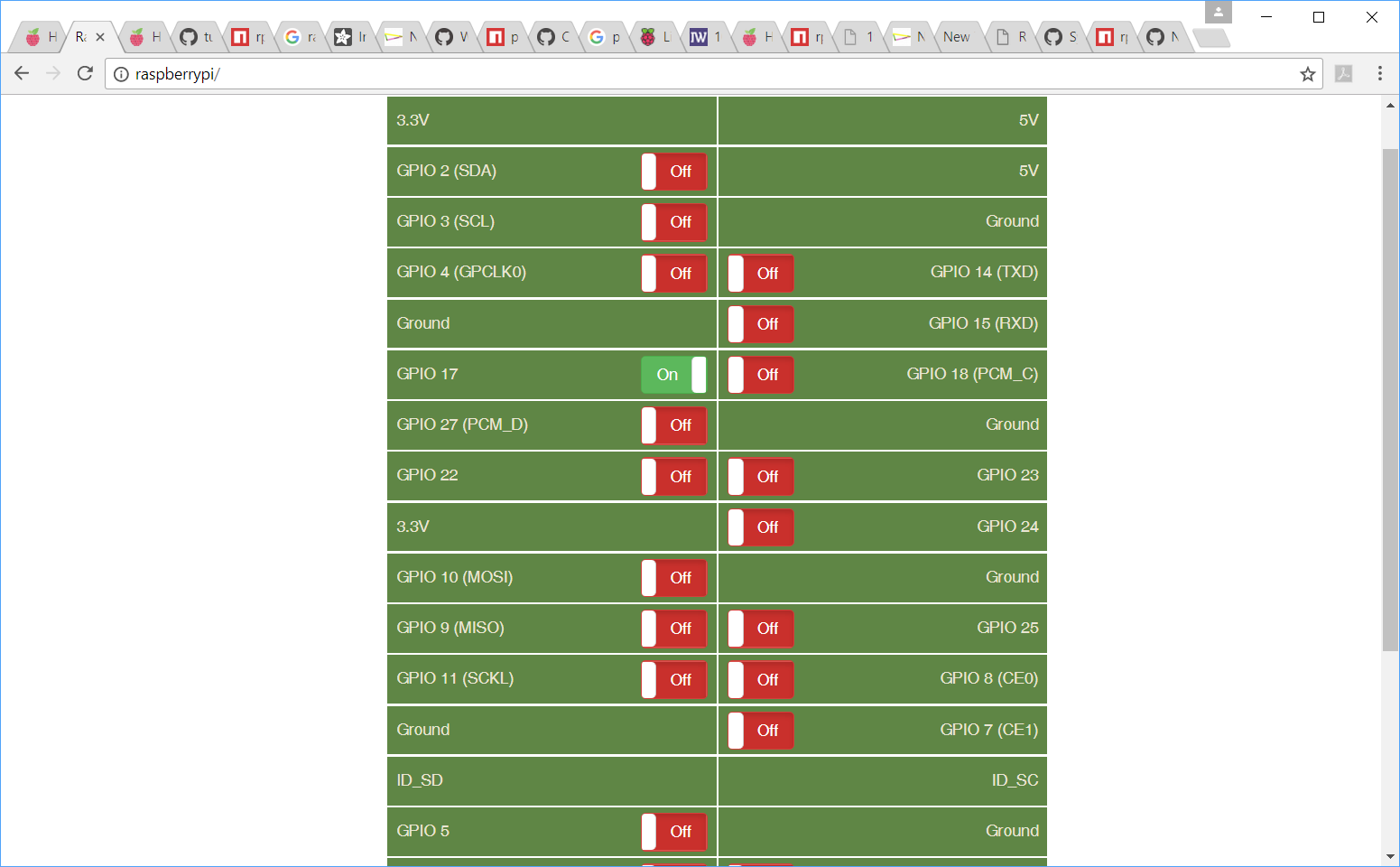
**Lab Demo:** Type commands on the Pi to turn ON/OFF the mbed LED. Hitting CTL C will exit a program.

**Additional Details:** This example will use a standard Linux device driver using OS file I/O APIs to communicate with the mbed’s USB serial port or virtual com port.Device drivers can be called from regular user programs (“sudo” not required to execute). It does take another couple of software layers inside the kernel to access the I/O device and will be a bit slower than the previous approach, or the approach used on mbed which directly communicated with the I/O hardware. The advantage is that I/O transfers use standard OS APIs so they look and act in a uniform way for user programmers and are automatically buffered, scheduled, and protected (i.e. MMU hardware protection of GPIO register memory address area and mutex access locks) by the OS. Note that several serial port settings needed to be changed in the code in addition to the basic File I/O APIs.

Setup and run the Pi mbed LED control demo found at:

<https://developer.mbed.org/users/4180_1/notebook/raspberry-pi-and-mbed-communication-using-the-usb-/>

**Part 6:** Raspberry Pi Node.js IOT Web Server with GPIO controlling the LED (10%)



IOT Web Page on PC from Raspbery Pi web server running Node.js controlling the LED

**Lab Demo:** Control the LED from the web page in a browser on the PC by clicking on GPIO 17

**Additional Details:** [Node.js](https://en.wikipedia.org/wiki/Node.js) is a newer language often used for JavaScript server side applications. [PHP](https://en.wikipedia.org/wiki/PHP) is another older language alternative and it is an extra credit option using an Apache server. Follow the instructions on the [How to setup a Raspberry Pi Node.js Webserver and control GPIOs tutorial](https://tutorials-raspberrypi.com/setup-raspberry-pi-node-js-webserver-control-gpios/) with a small patch for RPIO. Node.js now comes preinstalled, so skip that install step for Node. You can also skip the first basic web server demo, but note how little code is required to setup a web server in node.js.

NPM must be installed for the second part of the tutorial (it assumes it is already installed, but it currently is not on the Pi). Use the following two commands to install npm:

**sudo apt-get update**

**sudo apt-get install npm**

After installing NPM in the tutorial, a [patch is currently needed](https://github.com/fivdi/onoff/wiki/Node.js-v0.10.29-and-native-addons-on-the-Raspberry-Pi) (as of 7/17) to compile RPIO without errors during the install RPIO step. There is a [Debian Unstable](https://sources.debian.net/src/libv8-3.14/3.14.5.8-10/debian/patches/nodejsREPLACE_INVALID_UTF8.patch/) patch that [is disputed](https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=797545) for fixing the issue. It works, at least for this demo code. The patch is on the next page.

This patch is needed to compile RPIO. It can be manually applied by replacing the following snippet of code in  /usr/include/nodejs/deps/v8/include/v8.h using an editor:

enum WriteOptions {

NO\_OPTIONS = 0,

HINT\_MANY\_WRITES\_EXPECTED = 1,

NO\_NULL\_TERMINATION = 2,

PRESERVE\_ASCII\_NULL = 4,

};

with:

enum WriteOptions {

NO\_OPTIONS = 0,

HINT\_MANY\_WRITES\_EXPECTED = 1,

NO\_NULL\_TERMINATION = 2,

PRESERVE\_ASCII\_NULL = 4**,**

**REPLACE\_INVALID\_UTF8 = 0**

};

The GUI text editor will not be able to save the file. “sudo” will be needed to start the editor (such as vi or nano) to be able to save the file. Note that /usr/include/nodejs/deps/v8/include/v8.h will not exist, if npm hasn't been installed. After the patch, return to the tutorial and RPIO will install and compile. If the code patch was needed, repeat the command to install RPIO and it should compile without errors.

Once the server is started, on a PC browse to [**http:***//your\_pi\_ip*](http://your_pi_ip) and the IoT web page should appear after several seconds. Click the GUI switch for GPIO 17 and the LED should turn on or off. The PC used should be on the same sub net. The final auto boot section of the tutorial is not required for the lab checkoff. There are quite a few small node.js files in several directories to setup the server and web page. Ajax.js contains the code that controls the LED using the RPIO library.

**Part 7:** Using Node Red and Tweets to control the LED (10%)

**Lab Demo:** When a tweet is sent the LED comes on for 15 seconds and the tweet is read aloud.

**10/26/18 Last Minute Fix/Update to instructions for this part:**

Node Red is no longer pre-installed on the newest Pi OS image that just came out around a week ago! There are manual node red install instructions, some are a big mess, throw warnings & errors, and take forever. There are install instructions at <https://nodered.org/docs/hardware/raspberrypi> that have worked for some students, but it still takes a fair amount of time. You can also use the old SD boot disk and do this checkoff first before making a new boot disk, or get an old boot disk from a kit from the TA or another group. Don’t forget you will need to reconfigure Wi Fi for your account to get it working, since Node Red uses it to read email and for VNC, if you are using it.

Twitter starting requiring a developer account and API key code for the node red block to work this summer. Sounds like people were doing “fake news” tweets and collecting user data to sell using it. You can apply for one, but it now takes weeks since they are trying to filter out bad actors. I even saw a node red block to automatically send fake news Tweets! Some CS students in the class have a key from a CS class that still works from before they got hard to get and if you have one, you can still do the checkoff using Tweets. Most free web API keys come back with a code right away, but this one is now a mess.

The HDMI monitors in the lab do not have speakers and do not support audio output from the Pi or any other boards. To hear the audio output from the email (or tweet), a laptop with Wi Fi should be used and the web browser used to open node red needs to support text to speech output, or use VNC viewer on a laptop to connect and it will play audio on the laptop. The lab PCs have Ethernet networking, so they will not be able to open the node red web page since it is a local IP address and Wi Fi is not connected to the Ethernet subnet in the lab.

I wanted the demo to get data using the web and play it out as it arrives, so the closest easy alternative is email. The fix for lab 4 on for the Twitter API key needed issue is to switch to a block that reads email. Just put the email read block (output node on left) where the Twitter block is in the flow figures in the lab instructions that follow. There is a pre-installed email read block. Instead of sending a tweet, send yourself an email and it reads it and flashes the LED.

To configure the node red email block, the email account needs to be setup for an IMAP incoming server, here is an OIT web page with the details for GT’s IMAP Server:

<https://faq.oit.gatech.edu/content/how-can-i-access-my-office-365-mailbox-using-imap>

Note that the GT IMAP strange server name is ***outlook.office365.com*** and userid name is **yourGTid@gatech.edu** and click **SSL**.

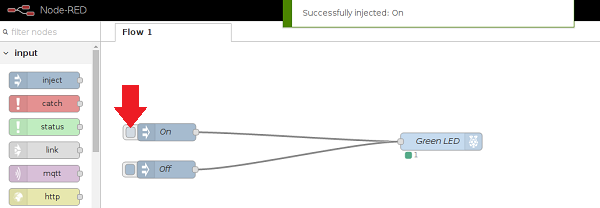
I verified that it will connect to GT and get email. It prints “connect error” below the block, if you get it wrong. Note: If you can’t type a “@” – You did not configure the Pi for a US and not UK keyboard. You might also change the option to mark email as read.

Use Outlook on a PC to send an email to try it. In the new email compose window, select **Format** and send as **plain text**. This is necessary since default HTML emails from some versions of Outlook do not seem to parse correctly and the output from the email block, *msg.payload*, shows up as an undefined string. The string with the email’s body text is needed to play audio and read the email. To avoid setting up an email account on a lab PC to try it, open *mail.gatech.edu* in a web browser to run Outlook. In the new email compose window click on “…” and select *switch to plain text*.

If you use a different email client or a phone and have the no string issue, there may be a way to tell it to send plain text email. Adding a debug node, displays the string in the debug tab window and this would allow you to verify this is the problem. Another option that I have not had time to try, is to uninstall the existing node red email block and install a newer one (it seems to be years out of date). If anyone tries a new node red block and it fixes it, let me know. One of the recent release notes says they switched to the new MIT parser and that sounds like it might fix something like this.

**Additional Details:** Another tool that can be useful for IoT projects is Node RED. [Node Red](https://learn.adafruit.com/raspberry-pi-hosting-node-red/what-is-node-red) is an IoT development tool developed at IBM that uses a graphical flow editor to connect nodes (i.e. code blocks) to develop IoT applications. The lines that connect nodes represent data flows between nodes. A version comes with the Pi OS install. Many nodes are available for web resources (web pages, tweets, email, web APIs, Alexa Dot…) and Pi I/O devices. It is based on node.js. A [tutorial is available to blink the LED setup](https://www.raspberrypi.org/learning/getting-started-with-node-red/worksheet/). Since the Node Red IDE runs in a web browser, it can also control the LED from a web page (by connecting existing nodes without any new coding). Learning how to use and configure the options in the various nodes is the tricky part. Nodes can be added with custom code when existing nodes cannot perform the desired operation. The LED block in the figure below has node.js code that calls the RPIO library functions that were just used in the web page GUI control demo. Node RED can also be used on BeagleBone boards. Node Reds GUI runs a lot faster when using VNC and Firefox on a PC! The audio will play over VNC. Changes to the Pi OS config may be needed for HDMI monitor audio to work, if the node red GUI is run locally on the Pi. The lab HDMI monitors have only DVI inputs on the back of the monitor which does not support audio out. HDMI to VGA adapters have an audio output plug that would need to be used for audio.

If the Pi does not have the correct time and date, Node Red’s networking authentication does not work for several IoT web services it uses. Check the time and date. Get the time and date working correctly before proceeding, if it is not correct. It may take a couple minutes for Linux to set the time using NTP after booting. There has been as issue with this in the past on some Pi wireless setups that are locked down for security like GTvisitor.



Node RED controlling a Pi LED using GUI pushbuttons in a web browser

For the demo, on the Pi, start the Node-Red server using **Programming -> Node-Red**. On a PC **open a web browser to *my\_pis\_IP*:1880** and the Node IDE should appear in the browser window. To develop an application, you drag and drop nodes from the pallet on the left into the Flow area in the center. When a node is selected, the info tab on the right has information about that node. Put the following nodes in the center:

In Output Menu– drop a **debug node** into the center Flow area (name will change to msg.payload). This node displays any messages flowing into it on the Debug tab area on the right.

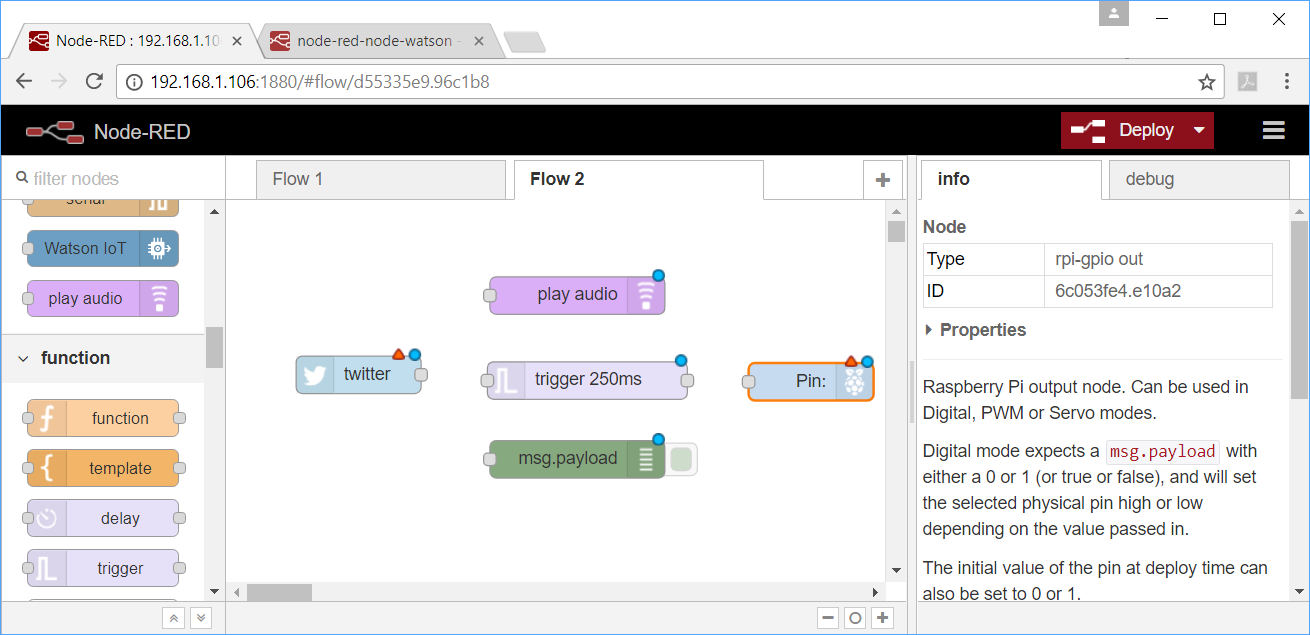
In Output Menu– drop a **play audio** node into the center Flow area. It uses a web text to speech server to convert text to an audio wave file

In Function Menu– drop a **trigger node** into the Flow. This node triggers an output message and then another message after a time delay each time it gets an input message

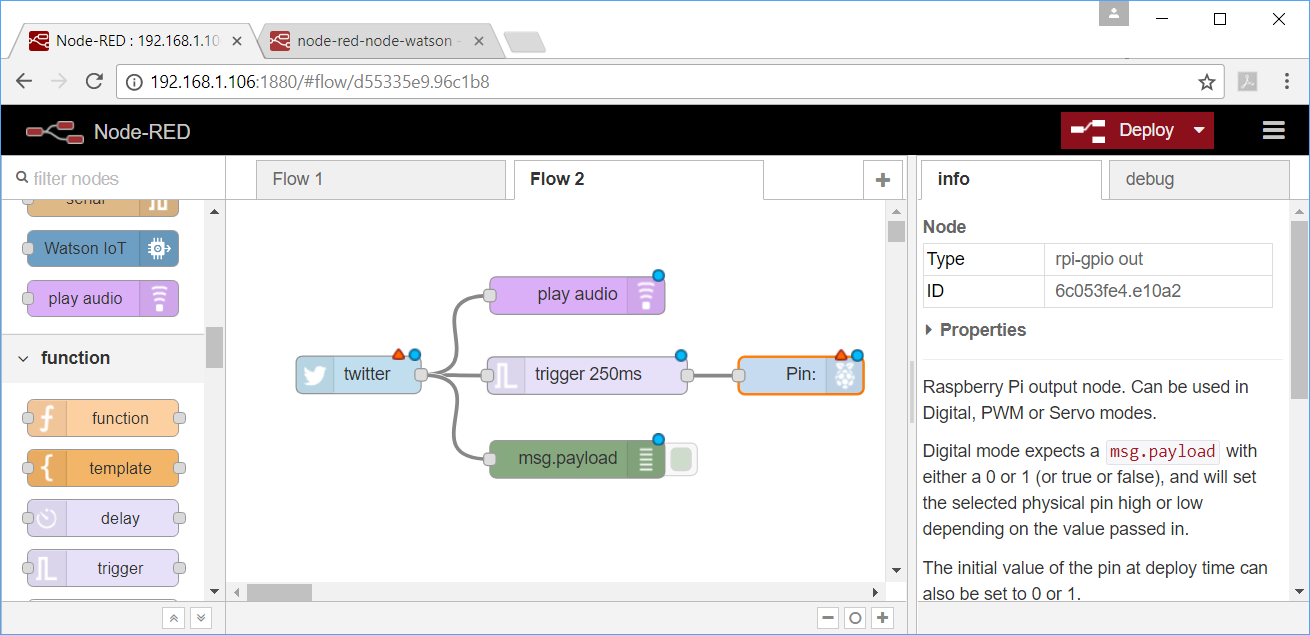
In Social Menu – drop a **twitter output node** (one with circle on right side) into the Flow area.

In Raspberry\_Pi Menu – drop an **rpio output node** (one with circle on left side) into the Flow area.

Arrange the nodes something like the following screen capture

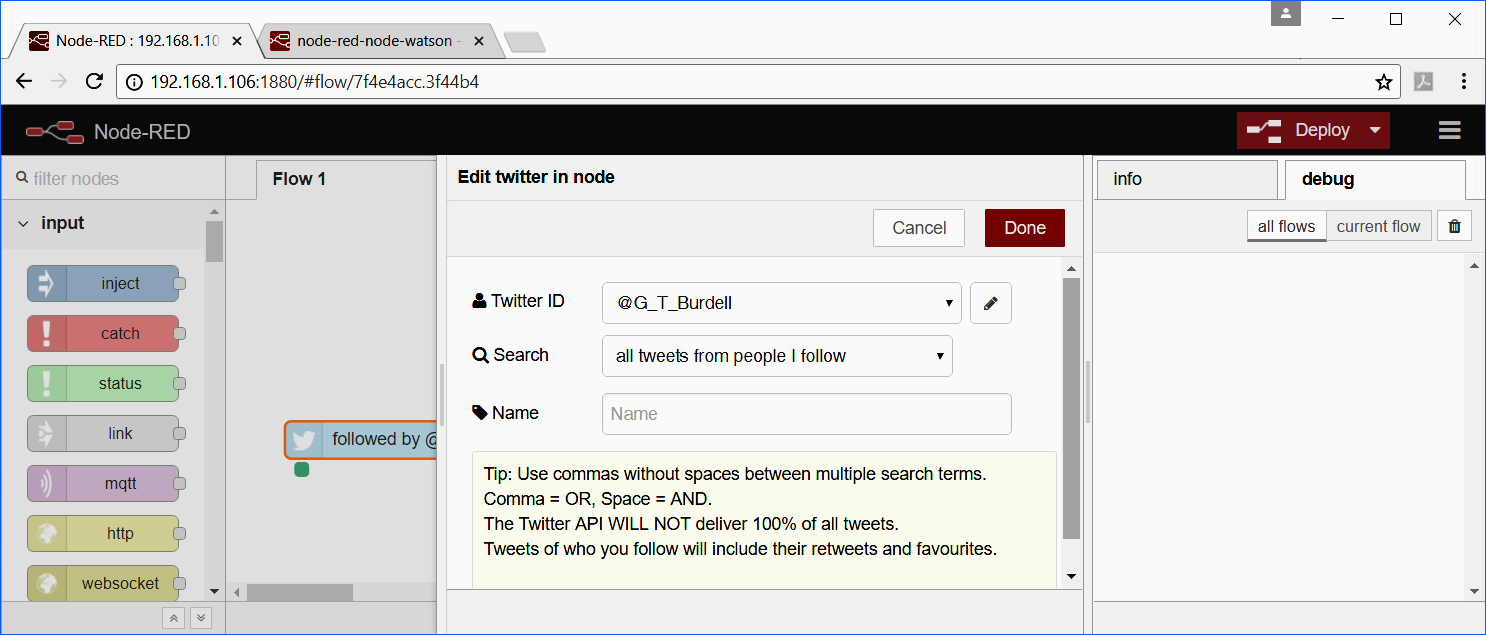


Next, using the mouse connect the nodes as seen in the screen capture below

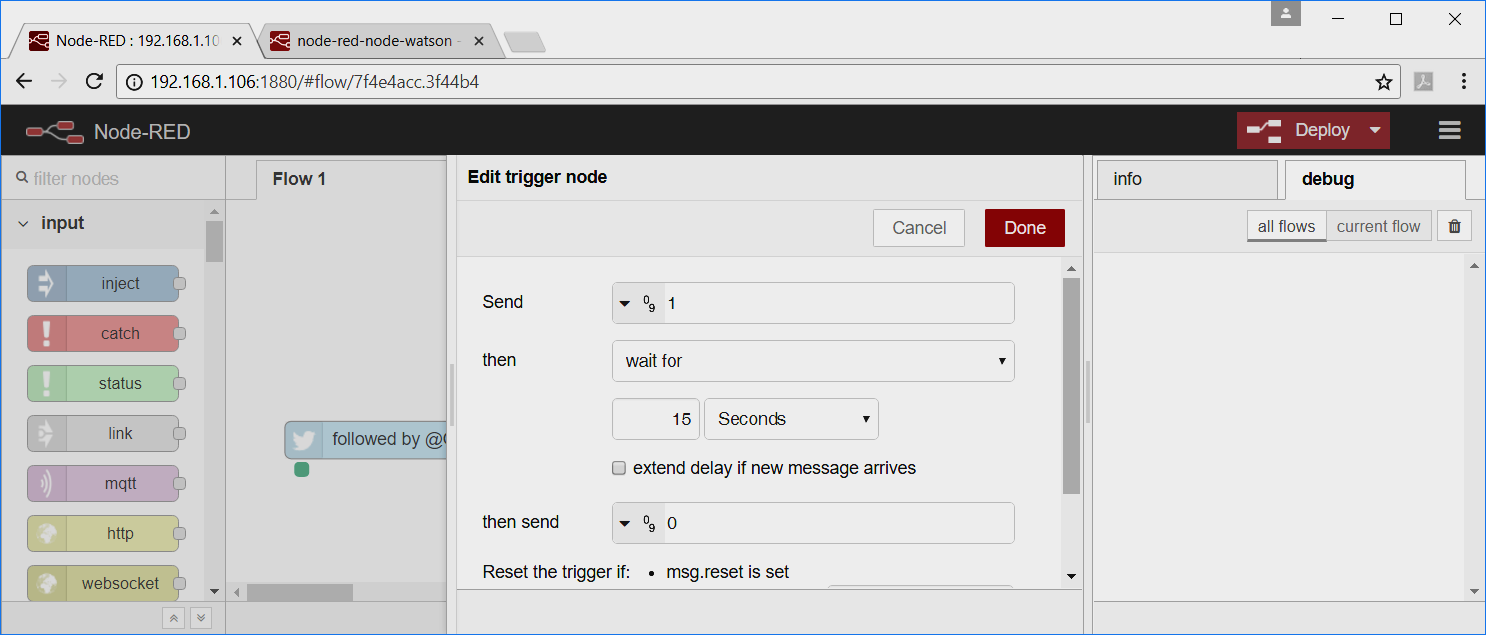


Some of the nodes still need to be edited or configured. Double click on a node in the flow area and another edit window should open. Configure them as seen in the following screen captures.

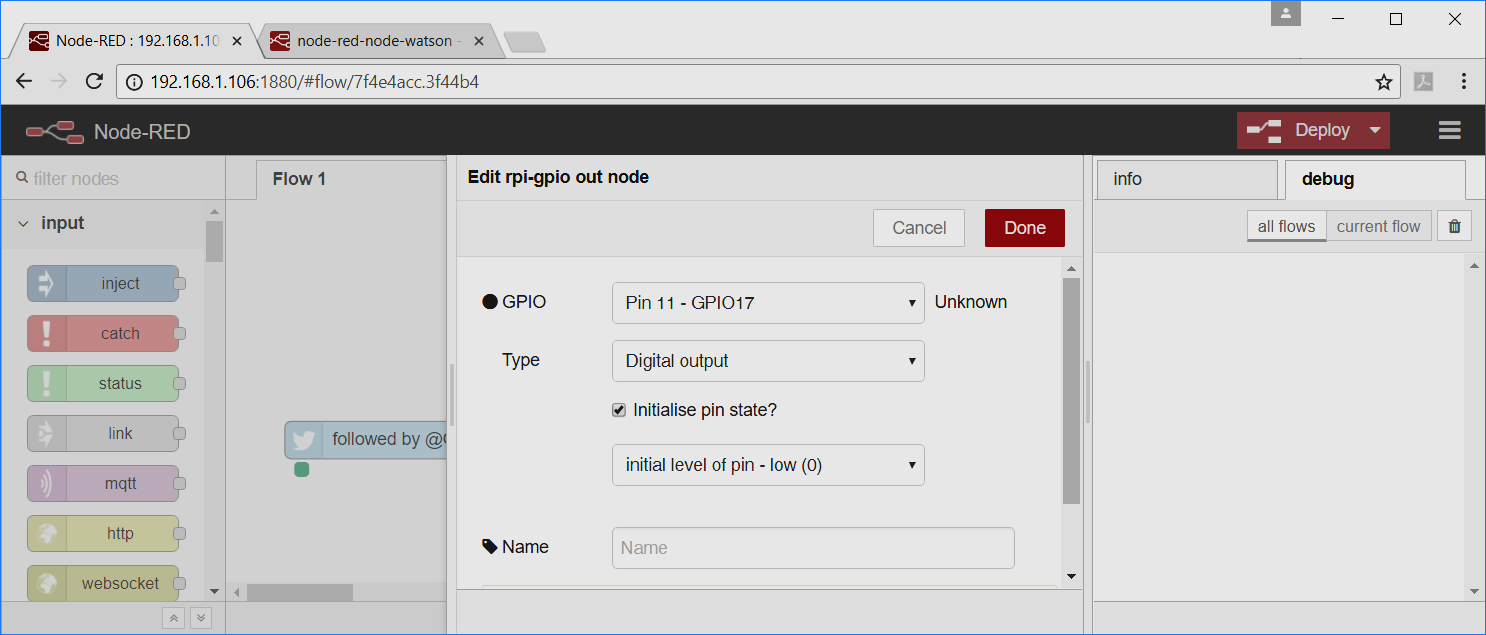
First the twitter node needs your Twitter ID. If you do not have one, create one at [www.twitter.com](http://www.twitter.com). Also, select “all tweets from people I follow” and then **click Done**. This node will send out a message containing each tweet. On some GT Wi Fi networks, Twitter might not connect properly to authenticate the first time when you hit done in the edit twitter window. If it hangs trying this, it will likely work fine on any home network. After getting it to authenticate the first time, it does not do it again, and then seems to always work on GT networks when running code.



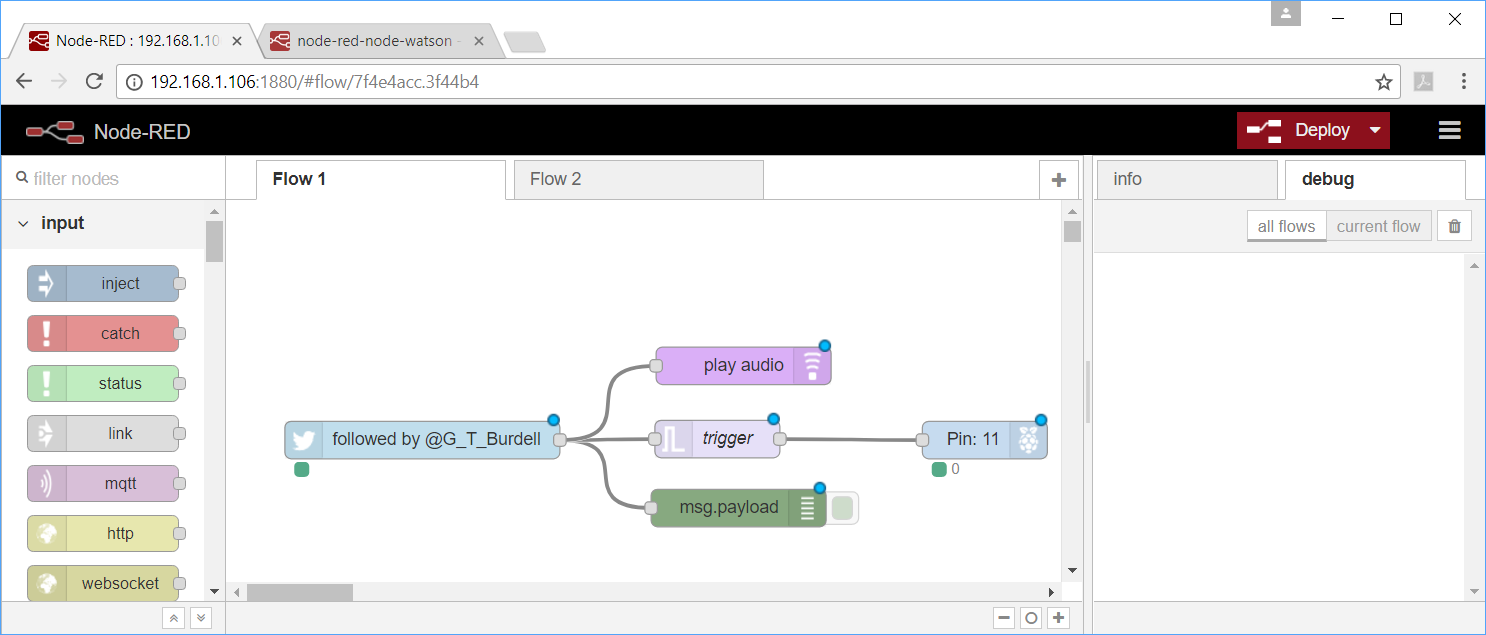
Next edit the Trigger node messages and delays. It sends out a 1, waits fifteen seconds, and sends out a 0 each time it gets an input message. 0 and 1 must be 0..9 and not A..Z type.



Next edit the rpi-gpio out node and select pin 11 (the LED), Digital output with an initial value of 0.



After configuring all of the nodes, it should look like the following screen capture.



Now it is ready to run. **Click the left side of the red Deploy Button** to run. Everything should now be running. Send a tweet and when the tweet comes in, the LED goes on for 15 seconds, the tweet is read aloud, and the text appears in the debug window.

If you don’t have a phone or the tweet web page handy, a time stamp tweet can be sent out by connecting an inject node to a tweet out node. Clicking the left button on the inject node will send a tweet.

There are other add-on nodes for web pages, email, audio input from a USB microphone, Pi camera input, and IBM web services including speech to text, natural language understanding, and image processing. There are a large number of add-on nodes available at <https://flows.nodered.org/>. There is also a new [Alexa Echo/Dot skill and node set](https://www.amazon.com/Ben-Hardill-Node-RED/dp/B01N0D97FZ).

**Extra Credit Options**

**Extra Credit** (1%) Automatically start running the Blink LED code at OS startup. A [tutorial](https://thepihut.com/blogs/raspberry-pi-tutorials/34708676-starting-something-on-boot) is available. See this [tutorial for some other options](https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/). This would also be handy to auto logon to Wi Fi networks with a password with wget.

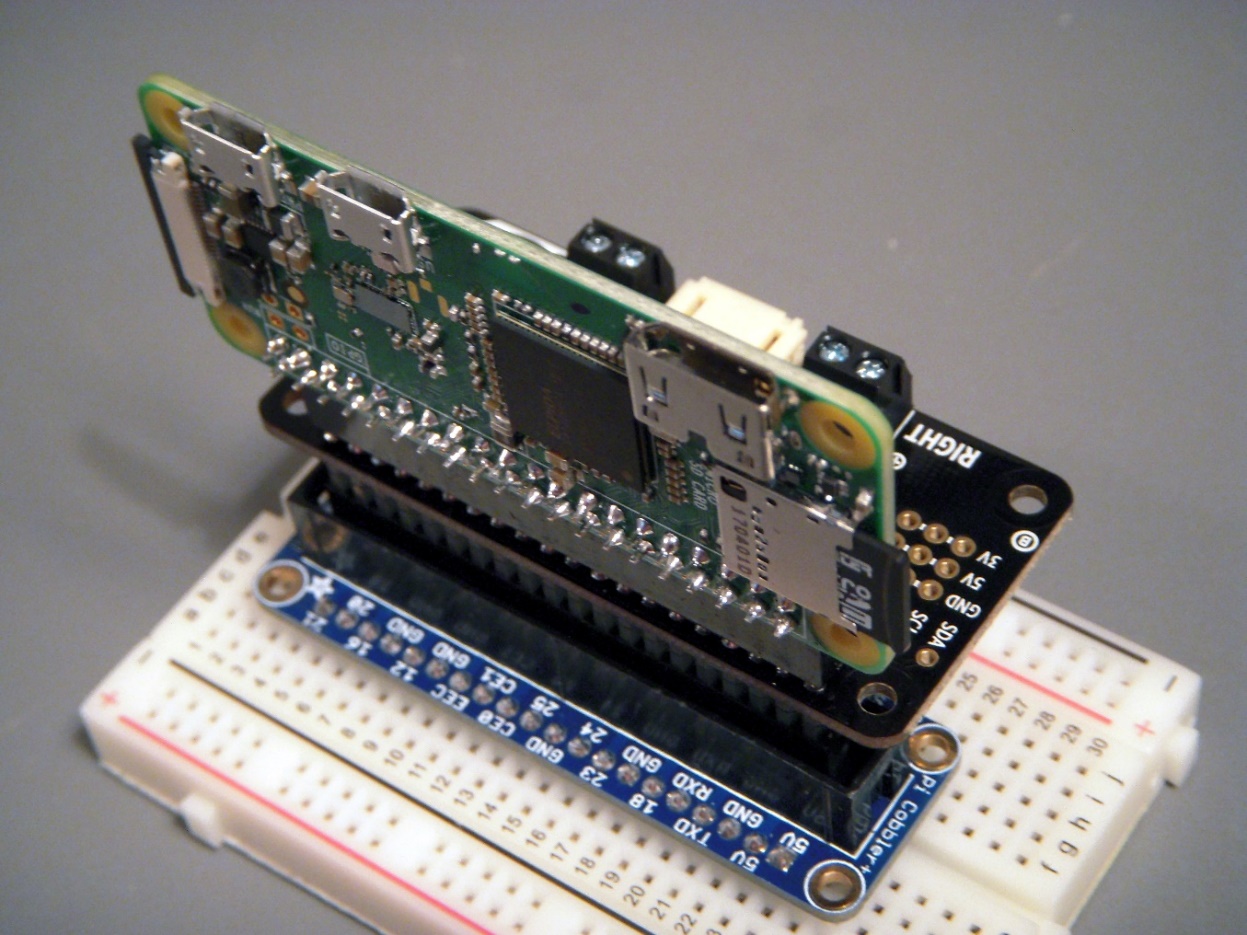
**Extra Credit** (1%)Develop C++ code using the PIGPIO library to dim the LED (instead of just blinking it) on the Pi using PWM. The Pi has one hardware PWM pin on the I/O header (GPIO pin 18 supports PWM**)**. Use “+” and “-“ user inputs from the keyboard to control the dimmer. See the pwm.c example that comes with the BCM2835 library. There are some new software PWM libraries for the Pi that can provide more than one PWM pin when needed and this feature is included in PIGPIO. Some of these use DMA from a buffer to be a bit faster than just bit banging a pin high and low in software.

**Extra Credit** (1%) Add a pushbutton switch input to toggle on/off the LED using a program loop. Without some debounce code, it may be a bit erratic, but that is OK for the demo. A pushbutton hit toggles the LED state (i.e., the pushbutton is not just kept pushed down to keep the LED on). A small time delay once the pushbutton is hit might help a bit. There is also a PIGPIO function to use internal pullups.

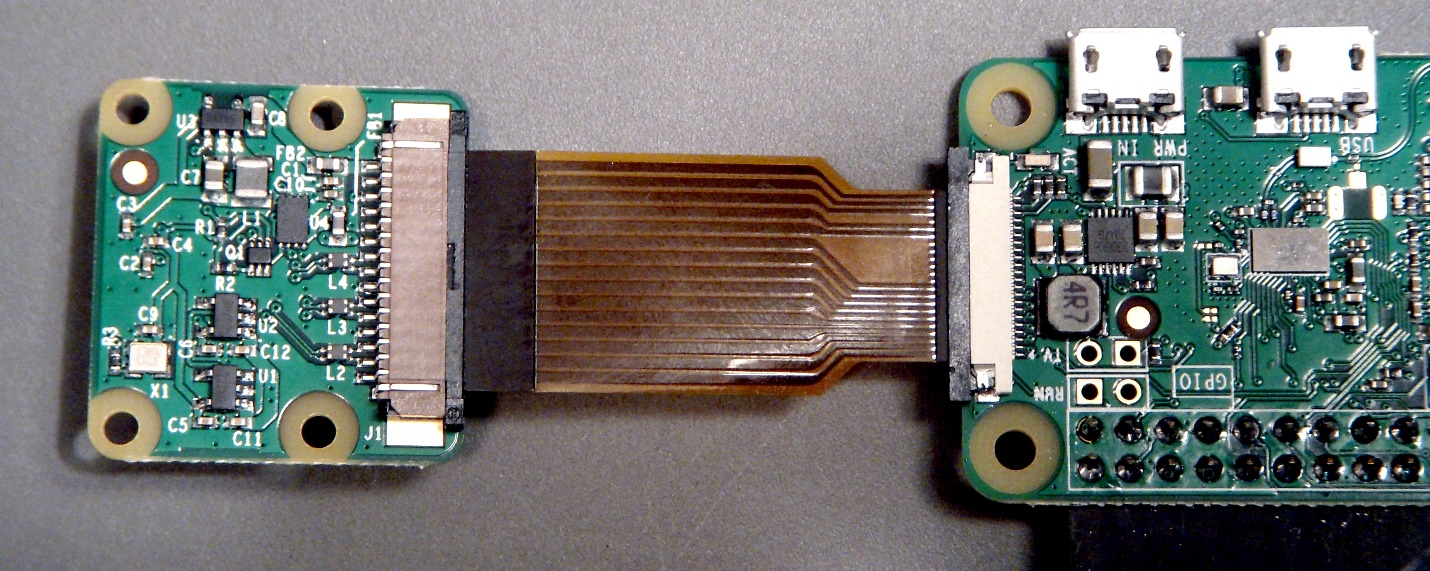
**Extra Credit** (1%) Develop C++ code to dim the LED on mbed using commands from the Pi. Use “+” and “-“ user inputs from the keyboard to control the dimmer. New code will also be required for the mbed.

**Extra Credit** (1%) Demo audio output using the Adafruit Stereo Bonnet. A [tutorial](https://learn.adafruit.com/adafruit-speaker-bonnet-for-raspberry-pi/overview) is available with complete instructions. The board has two I2S digital audio interface class D amps to drive the stereo speakers. Read carefully, two reboots are required after installing the new driver. A short video showing the setup with audio is acceptable for checkoff. Run the demo that streams the MP3 music data from soma FM. It **requires the 2A 4180 power supply** using the barrel jack adapter. Once this audio driver is installed using the complex script, it is difficult to remove and get back to the basic HDMI audio out setup. A new SD card OS install might be the easiest solution.

**Be sure to plug it in as seen below with power off. It can blow the amp if not done correctly:**



**Extra Credit** (1%) Demo use of the Pi camera. A [tutorial](https://www.raspberrypi.org/learning/addons-guide/picamera/) is available with complete instructions. A personalized screen capture showing an image from the camera is acceptable for checkoff. Use the shorter orange (not white) camera flat flexible cable (FFC) and the ends with the black stripe (other side has metal pins) are on (i.e., visible from ) the component side of the board on both the Pi and Camera. If you get a new kit in a bag, this second cable is hidden inside the red case. The longer white cable in the kits is for older Pis. Longer FFC cables are available, if needed for projects.



Pi Zero W Camera FFC setup using the CSI interface with black strips on top, Some FFC cables may look a bit different, but the exposed metal contacts are always on the bottom. The cables just push into the connector and do not latch. When they are all the way in the metal contacts cannot be seen from the bottom of the board.

**Extra Credit** (1%) Setup an Apache Web Server and PHP on the Pi. A [tutorial](https://www.raspberrypi.org/documentation/remote-access/web-server/apache.md) is available with complete instructions. A personalized web page screen capture is acceptable for checkoff. There are several [Pi projects](http://www.instructables.com/id/Simple-IOT-project-for-Beginners/) available on the web that use PHP to control LEDs from a web page, but that operation is not required for the lab checkoff.

**Extra Credit** (1%) Blink the LED using Linux’s GPIO device driver using C/C++ file system APIs (not an add-on driver library that requires “sudo” privledges).

**Extra Credit** (1%) Turn the Pi into an Amazon echo device using this [tutorial](http://lifehacker.com/the-simplest-way-to-build-a-raspberry-pi-powered-amazon-1794218212). An amp, speaker, and USB microphone will be needed. Some students report higher recognition rates using Google. The basic theory is the same for implementing it as a Google Assistant; however there are slightly more steps. The link below has the step by step process on a Pi 3 or Pi Zero W: <https://developers.google.com/assistant/sdk/guides/service/python/embed/audio>

**Extra Credit** (1%) Use the Pi and a camera (USB or Pi camera) to stream live video to a web page. There are several tutorials available. This [tutorial](http://www.instructables.com/id/How-to-Make-Raspberry-Pi-Webcam-Server-and-Stream-/) may be one of the easier setups. Another [tutorial](http://www.instructables.com/id/Raspberry-Pi-as-low-cost-HD-surveillance-camera/) uses Wi Fi with the Pi camera, but is a bit older.

**Extra Credit** (2%) If you are already familiar with the Pi and are considering using the BeagleBone board for the design project, do the same things (update OS, Blink LED, send serial data to mbed) using one of the BeagleBone Boards with Wi Fi in the Cloud 9 IDE using C++ and a different GPIO I/O library for BeagleBone. Wi Fi still needs to be setup for update and upgrade and to run the Cloud 9 IDE. See [BeagelBone.org](https://beagleboard.org/) for more information. The BeagleBone Cloud 9 IDE GUI runs over the USB cable, but Wi Fi needs to be configured for updates and upgrades.

**Early Bird Bonus** (2%) First four groups to finish the required parts of the lab.