2: Raspberry Pi Setup

ECE 180D: Systems Design Laboratory

Contents

1 Introduction

This tutorial will focus on getting a Raspberry Pi setup in a **headless** configuration so that you can access the small computer that you now own and run a simple program on it **without the use of a dedicated monitor and keyboard**. If this tutorial proves confusing, more pictures and tips can be found in the following two tutorials: Raspberry Pi Zero Headless Quick Start and Connect To A Raspberry Pi Zero With A USB Cable And SSH.

Because the RPi has a headless configuration, one must understand command-line tools to access, deploy and run scripts on the remote computer. Useful links to brush up these skills can be found here and a more thorough coverage here.

2 Materials and Preparation

1. Raspberry Pi Zero WH.
2. MicroUSB cable that allows for data communication.
3. MicroSD card.
4. (Optional, but preferred) Power supply.
5. A personal computer with an SD card reader and a USB port.
6. Access to a network with an internet connection.
7. (Advanced) Your Logitech camera and your OTG connector

3 Overview of the Hardware

Please find the overview of the HW below. Note the number of GPIO pins, processing power, power inputs / data inputs, headers, WIFI/Bluetooth chip etc...

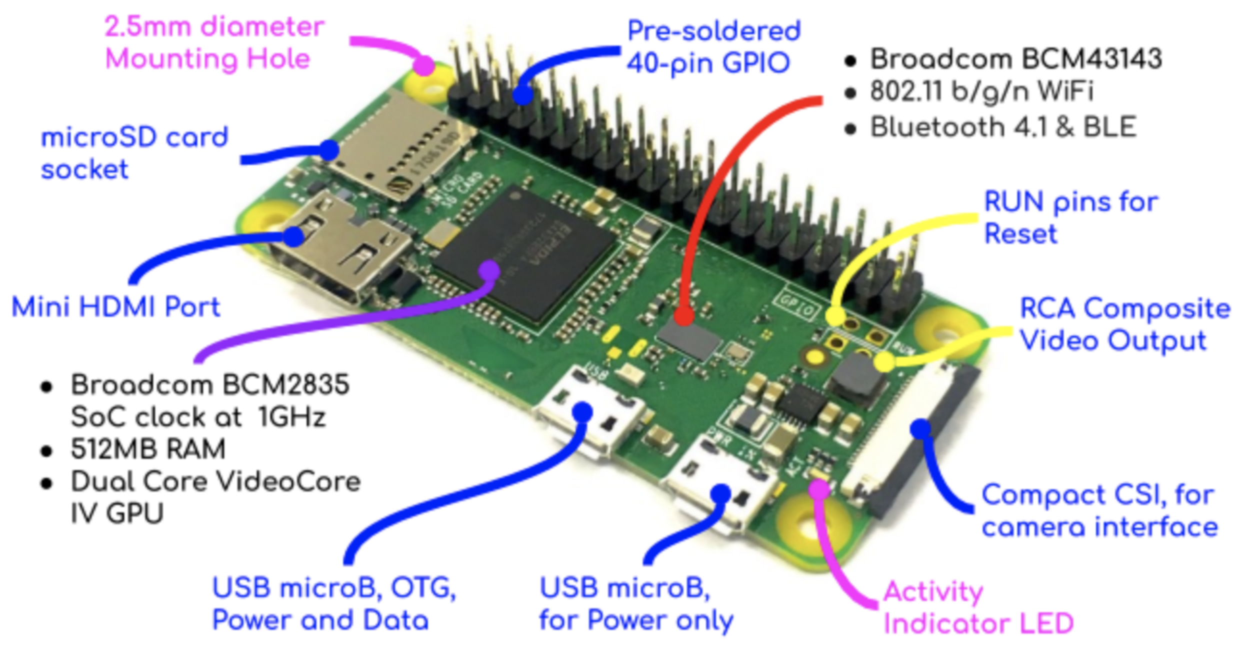

2.5mm diameter Mounting Hole
Pre-soldered 40-pin GPIO
• Broadcom BCM43143
• 802.11 b/g/n WiFi
• Bluetooth 4.1 & BLE
microSD card socket
RUN pins for Reset
Mini HDMI Port
RCA Composite Video Output
• Broadcom BCM2835 SoC clock at 1GHz
• 512MB RAM
• Dual Core VideoCore IV GPU
Compact CSI, for camera interface
USB microB, OTG, Power and Data
USB microB, for Power only
Activity Indicator LED

Figure 1: An overview of the RPi as obtained from Cytron

The abstractions that go into implementing a full-stack application on hardware can be reduced to the following layers: note that in this class we will be working with the higher levels of coding (user-friendly Python and C++).

4 Installing the OS

The MicroSD card we recommended on the materials page came pre-installed with this OS, so you can skip everything except the last step in this section if you have that. We are going to request the RPi to boot from the OS directly installed in the microSD card. To do so, we first need to dedicate the card to the OS and **flash** the drive.

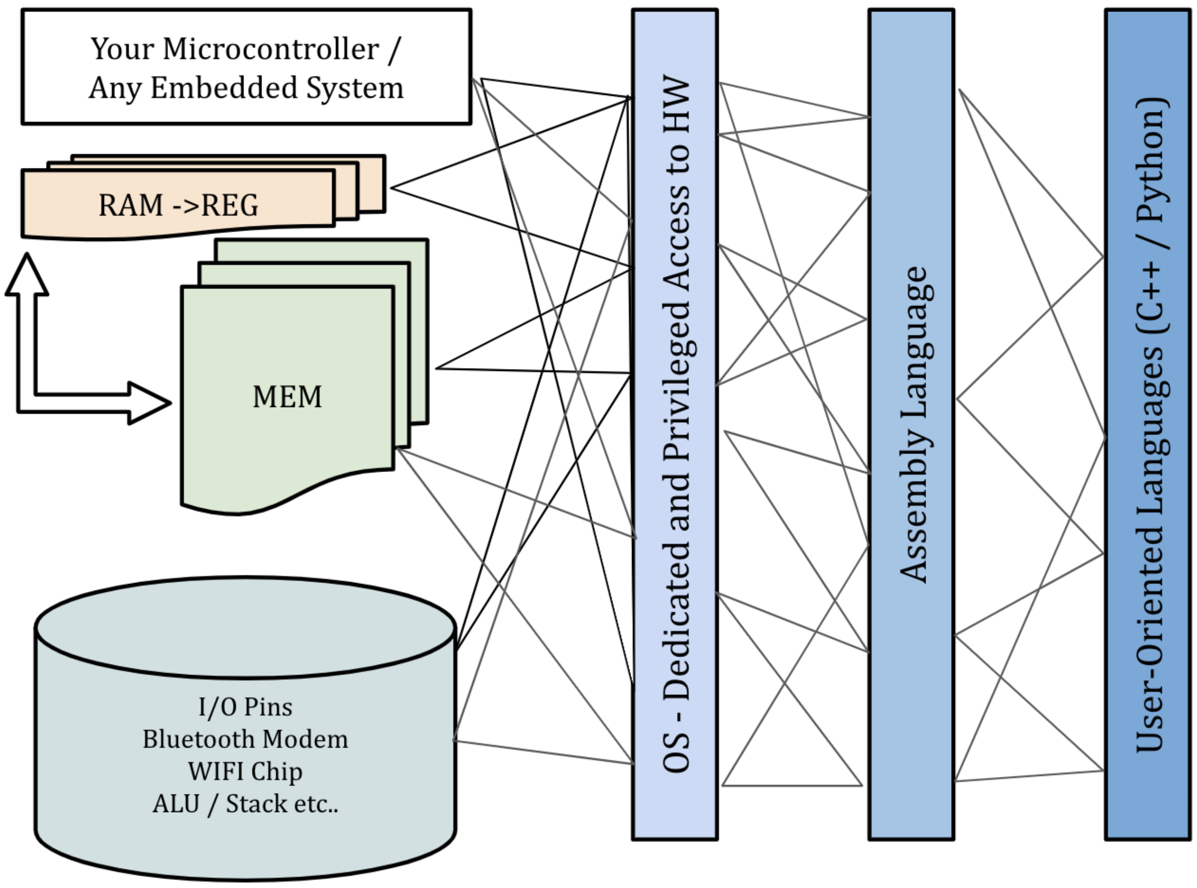

Your Microcontroller / Any Embedded System
RAM ->REG
MEM
I/O Pins
Bluetooth Modem
WIFI Chip
ALU / Stack etc..
OS - Dedicated and Privileged Access to HW
Assembly Language
User-Oriented Languages (C++ / Python)

Figure 2: Standard hierarchy in most hardware embedded systems - unknown

1. Download Raspberry Pi OS. The Lite version is sufficient for what we are planning on doing, but get the option that you prefer.

2. Connect your microSD card to your computer.

3. Download and use Etcher (easy) or some other installation method.

    (a) After installing and opening Etcher, open it up.

    (b) You want to simply add your Raspbian image file (the zip folder that you downloaded off of the Raspberry Pi website).

    (c) Then, select your SD card on the spot, like in Figure 3.

    (d) The window should look like Figure 4 before you flash.

4. After properly imaging your OS onto the SD card, its icon should disappear from your computer.

5. Try turning on your Raspberry Pi with the MicroSD card in it. Plug in your MicroSD card and also your MicroUSB cable to either opening (PWR or USB) just to get the power into the Raspberry Pi.

   A green LED should turn on and flash a couple times, indicating that the OS is correct and that the Raspberry Pi is working as intended. The green LED should stay on after the loading period has completed.

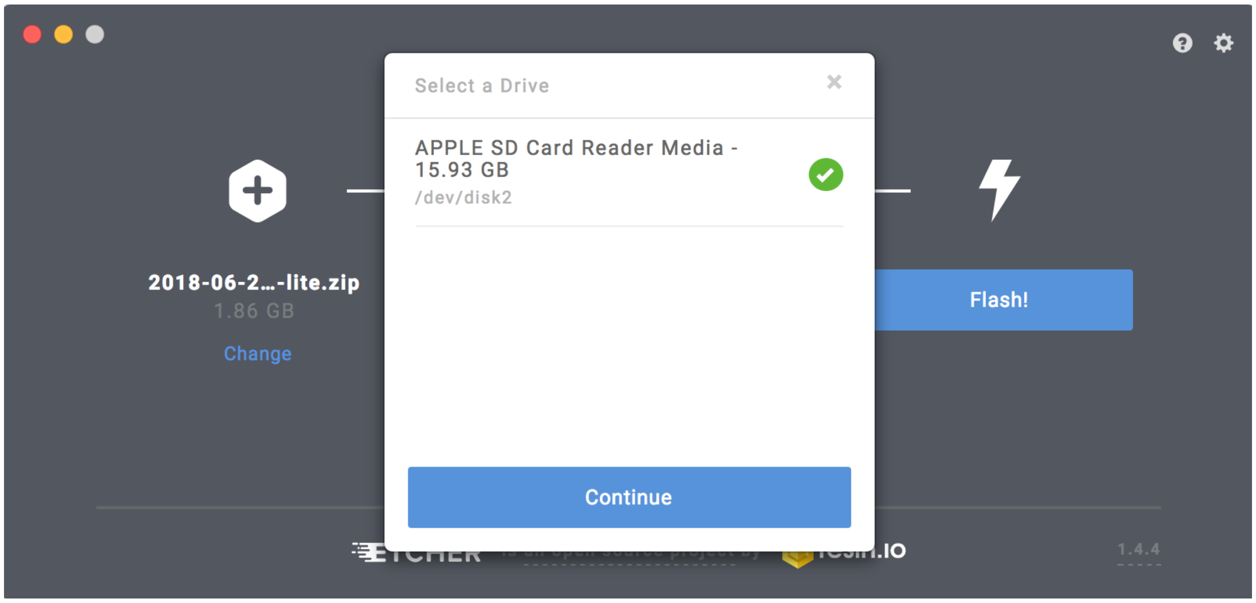


Figure 3: Selecting your SD Card (old version of Etcher)

5 Wired Connection via USB using SSH

In this part, we will setup the MicroSD card along with the ability to connect to the Raspberry Pi using a USB connection by emulating ethernet over USB.

1. Unplug your Raspberry Pi and remove the MicroSD card. Reconnect the MicroSD card to your computer. You should see a new drive called **boot**. Access this file.
   - On Windows, the computer may alert you that there is something wrong with this drive. There is nothing wrong with the drive. Just cancel this warning: **do not reformat the drive**.
2. Open `config.txt`. At the end of the document, add the following two lines. These will allow you to implement all the connections that you need.

   ```
   dtoverlay=dwc2
   enable_uart=1
   ```

   - Preferably type this in, so that there won't be some strange invisible characters that enter your config file.
   - Don't add any extraneous spaces. Only use the return character in between.
   - **Careful! On Windows, because the return is actually encoded differently, it may cause errors when the Raspberry Pi tries to read it**. Try using Notepad++, Sublime Text, or Microsoft VS Code to type.
3. Now open `cmdline.txt`. You will find that the contents are just a single line, and an empty line underneath. Without changing anything else, add the following at the end of this line.

   ```
   modules-load=dwc2,g_ether
   ```

   - Only put a space after the final `rootwait`. Do not put any extraneous commas or spaces.
   - If you did not actually start up your Raspberry Pi in the previous part, you will want to put this after the `rootwait` and not the end of the line.

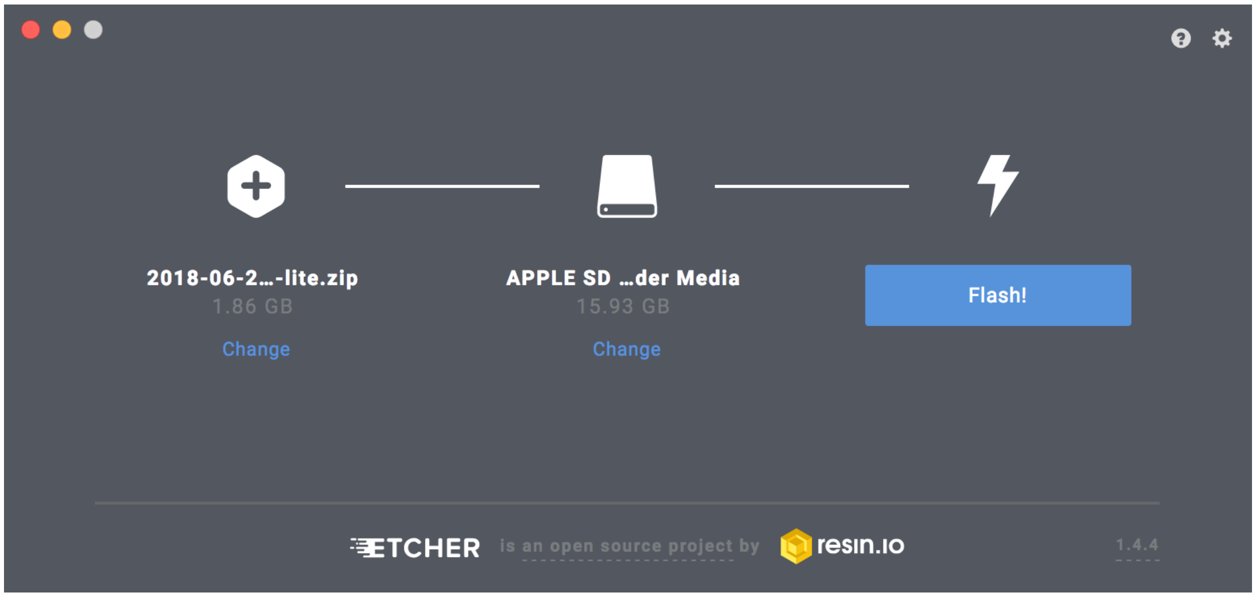


Figure 4: Final Etcher Screen

4. Finally, you want to create a single empty file called `ssh`. There should be no file extension to this, which is especially important on Windows. For example, using `touch ssh` will create this file in a UNIX system.

5. You're done with messing with the files! Try plugging it back in the Raspberry Pi.

   - Note: the `ssh` file that you created will be removed. However, if the instructions worked up to this point, this does not need to be done again. Creating it again, however, does not change anything, so do not worry if you do it multiple times.

6. When plugging your Raspberry Pi into your computer using the MicroUSB, you should still see the Green LED flash and then remain on.

   - If not, you might need to try the previous steps again and really make sure that nothing is wrong. Worst comes to worst, start over by reflashing the image onto the MicroSD card (a quick solution if already done).

7. You will need to have some form of Bonjour or Zeroconf on your computer. This will allow us to access the Raspberry Pi using the name `raspberrypi.local` instead of an IP address (which we don't have). This is built into Mac computers, so this is not necessary. However, if you have any other OS, you may need to install things (come back to this step if it turns out it doesn't work).

   - On Windows, if you have iTunes, this is already done for you. Some other programs such as Photoshop, etc. may have it installed too. If not, install Bonjour Print Services for Windows (an older version, but should work) or iTunes. If you end up installing something, you may need to restart your computer in order to continue. This is the most inconsistent part of the Windows procedure.
   - On Linux, you may need to install the Avahi daemon.

   If everything worked properly, your computer will now recognize the Raspberry Pi as `raspberrypi.local`. Sometimes, `raspberrypi` might work for Windows and Linux, but it's not consistent if you use multiple Raspberry Pis.

   This is, by far, the most inconsistent step for anyone not using Macs. If you get stuck here, don't feel discouraged. Since you are working at home, you have the nice option of completely skipping this step and going directly to connecting wirelessly in section 6.1.

8. Now, let's test the wired connection. We will need to try to **ssh** into your connected Raspberry Pi. After the green LED has stabilized (give it a minute or two for your first attempt),

   - On Mac / Linux, you can just open your console and type in `ping -c 3 raspberrypi.local`. If everything has been configured correctly to this point, it should not give you the warning `ping: cannot resolve raspberrypi.local: Unknown host.` Wait a while longer to make sure that the loading has been completed. After connecting, you should see lines saying

     ```
     PING raspberrypi.local (169.254.150.137): 56 data bytes
     64 bytes from 169.254.150.137: icmp_seq=0 ttl=64 time=0.615 ms
     64 bytes from 169.254.150.137: icmp_seq=1 ttl=64 time=0.462 ms
     64 bytes from 169.254.150.137: icmp_seq=2 ttl=64 time=0.347 ms

     --- raspberrypi.local ping statistics ---
     3 packets transmitted, 3 packets received, 0.0% packet loss
     round-trip min/avg/max/stddev = 0.347/0.475/0.615/0.110 ms
     ```

     If this is the case, you are connected! All you have to do now is simply type

     ```
     ssh pi@raspberrypi.local
     ```

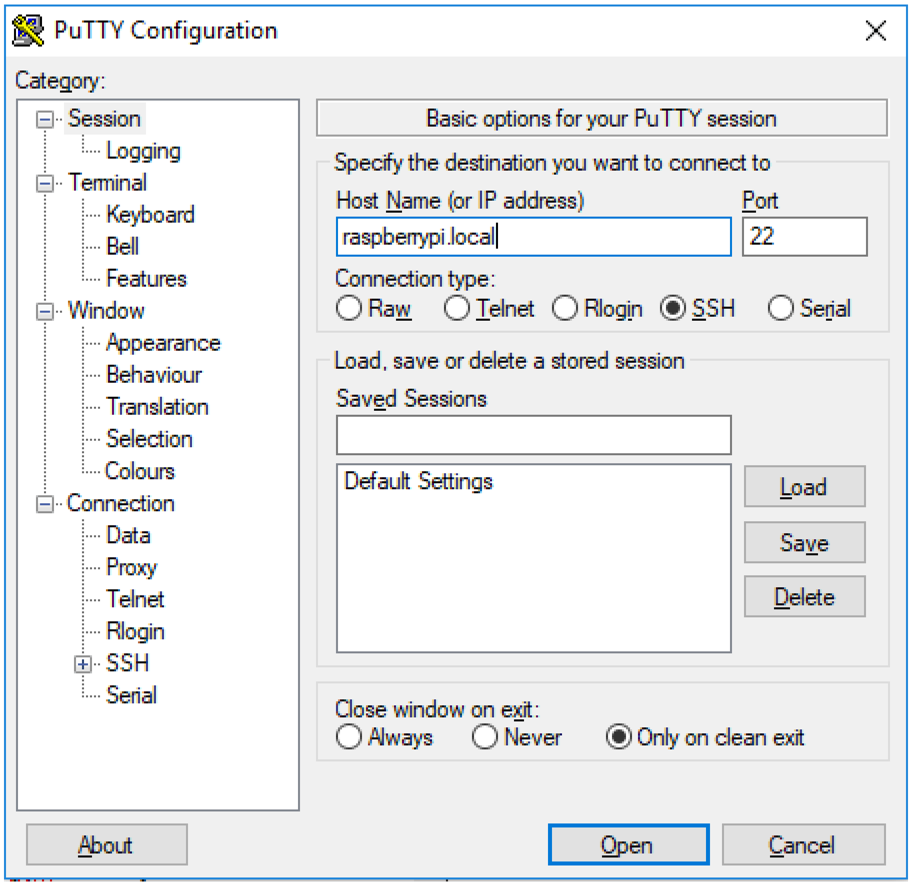
     The computer may ask if you want to add the key to your system, which you should reply yes to. Then, you simply have to use the default password '`raspberry`' to enter. While we recommend a change to some simple password for your team to use, you can change that whenever by using the '`passwd`' command.

   - On Windows, (if you don't want to use the Ubuntu subsystem,) you will have to download and install PuTTY. PuTTY is a great way to interface with SSH on Windows. PuTTY is a standalone application (i.e. does not need installation, just run the application) that simplifies the process quite a bit. Just input `raspberrypi.local` in the Host Name category, like shown in Figure 5. Everything else can be in default settings, but just check that the Port is set to 22 and the Connection type is SSH.

     If everything worked out correctly, a prompt will show up asking if you want to add this key to your list of keys. Hit yes and then log in with the username **pi** and the default password of `raspberry`. If it says that `raspberry.local` is not found, then there is something wrong, most likely with the Bonjour installation.

   - Quick warning: For the most part, you will only be able to use `raspberrypi.local` as a name for a single Raspberry Pi at a time. We will see what we can do to get around this in the future so that you can access multiple Raspberry Pis from the same computer. However, if you try to access another Raspberry Pi rather than your previous one, you will get the following scary-looking warning:

     ```
     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
     @       WARNING: POSSIBLE DNS SPOOFING DETECTED!          @
     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
     The ECDSA host key for raspberrypi.local has changed,
     and the key for the corresponding IP address 2605:e000:1314:47e9::ab15:c5e3:6b76
     is unknown. This could either mean that
     DNS SPOOFING is happening or the IP address for the host
     and its host key have changed at the same time.
     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
     @    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
     IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
     Someone could be eavesdropping on you right now (man-in-the-middle attack)!
     It is also possible that a host key has just been changed.
     ```



Host Name (or IP address): raspberrypi.local
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Close window on exit: Always Never [✓ Only on clean exit]

Figure 5: Putty Setup

```
The fingerprint for the ECDSA key sent by the remote host is
SHA256:QlIMWuE+fvEztLZSc2PUGdLUJ6oBO/mPdHSHvizqOsA.
Please contact your system administrator.
Add correct host key in /Users/jeffreyjiang/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /Users/jeffreyjiang/.ssh/known_hosts:22
ECDSA host key for raspberrypi.local has changed and you have requested strict checking.
Host key verification failed.
```

To setup a new Raspberry Pi, simply go into the folder mentioned and remove your current key for your name. This will allow you to create a new key for your new Raspberry Pi that you are setting up.

9. Great! Now you can access the command line of your Raspberry Pi through a wired connection. We will be having a Linux tutorial after this to help you with the basic commands, just in case this is your first time. One thing to note is that with just this setup, you will not have access to the Internet, and you can only do wired SFTP (we will discuss this in a bit). Press Ctrl-D to log out of your Raspberry Pi.

6 Wireless Connection

Since we are working with IoT devices, eventually what we want is to access our device wirelessly. As such, we will need to do a couple other small things in order to allow our devices to work wirelessly.

6.1 Standard Wireless Connection

Since we are at home this year, you guys get this part easy! This is because there isn't as much security protections over the network as **eduroam**, making it possible to do the connection without the wires.

1. Plug your MicroSD card back into your computer. We will need to change the `wpa_supplicant.conf` file.
2. Refer to the attached `wpa_supplicant_home.conf` or create a new file that contains the following:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US

network={
    ssid="YOURSSID"
    psk="YOURPASSWORD"
    scan_ssid=1
}
```

3. Change the SSID and password to what your WiFi name and password are, respectively and then save the file. If you used the attached file, change the name to `wpa_supplicant.conf`.
4. Plug this back into your Raspberry Pi and boot the device up, using the same method to do the wired connection.
5. From this point, you can do the same thing as the wired setup and just refer to the Raspberry Pi as `raspberrypi.local` whenever it is powered on (if you got the Bonjour thing working properly). This does not stop you from using `ifconfig` and getting the IP address to access, if you need to use multiple Raspberry Pis with the same hostname.

   Alternatively, you may be able to access your network modem to get the IP address of the Raspberry Pi directly. Then, all you have to do is to `ssh` into the Raspberry Pi using the IP address directly. Furthermore, since usually devices don't get re-assigned IP addresses in a home network, this can definitely be the permanent way for you to access your Raspberry Pi in the future.

6.2 Eduroam (for potential future reference)

Luckily, you guys are primarily at home with home WIFI networks, so we don't have to deal with the mess it is to connect with Eduroam. We leave this section in just in case Winter Quarter can be in the lab, but we are uncertain.

For this part, we will need to still do a wired connection, so keep your MicroUSB cord around. Just as a reminder, eduroam can be accessed by all students with the following credentials:

```
Username: <your MyUCLA username>@ucla.edu
Password: <your MyUCLA password>
```

For the purposes of these wireless connections, remember that in order to actually connect to things, you will need to have both your laptop (the connecting device) and your Raspberry Pi to be on the same network.

Just as a reason why **eduroam** requires special attention: The **eduroam** network utilizes the **Dynamic Host Configuration Protocol** (DHCP) to assign IP addresses; as it must service users connecting and disconnecting at unknown time intervals. DHCP allows devices to request an IP address on connection initialization. Devices may be assigned a static IP address, or have one automatically assigned to them by DHCP. As such, devices that are not assigned a static IP address may not rely upon having the same IP address on initialization of a connection with the Wi-Fi network utilizing DHCP.

The ultimate meaning behind this is that **eduroam** creates a new IP address for every device each time it joins the network and that it has a bunch of hidden security that protect the IP addresses from being seen.

As such, we cannot just access the Raspberry Pi using the phrase `raspberrypi.local` because **eduroam** hides that information.

1. Plug your MicroSD card back into your computer. We will need to add one more file in your MicroSD card in order for this to work properly: `wpa_supplicant.conf`.

2. Refer to the attached `wpa_supplicant_eduroam.conf` or create a new file that contains the following:

   ```
   ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
   update_config=1
   country=US

   network={
           ssid="eduroam"
           scan_ssid=1
           key_mgmt=WPA-EAP
           eap=PEAP
           identity="yourusername@ucla.edu"
           password="yourpassword"
           phase1="peaplabel=0"
           phase2="auth=MSCHAPV2"
   }
   ```

3. Change the username and password to match your login credentials and then save the file. If you used the attached file, change the name to `wpa_supplicant.conf`.

4. Plug this back into your Raspberry Pi and boot the device up, using the same method to do the wired connection.

   - As a note, the Raspberry Pi will delete the `wpa_supplicant.conf` file after it successfully boots up. For simplicity sake, you may want to keep an extra copy of your `wpa_supplicant.conf` somewhere safe. This allows you to switch in between networks easily by just copying the file onto the MicroSD card whenever it is necessary. The main reason against it is that it has your MyUCLA password in plaintext. If you aren't going to be switching networks much or you are willing to try out the Multiple WiFi network setup, this can be avoided. Alternatively, you can just keep using the files without your information and move them in every time you need to fix something up.

5. From here on out, these are the instructions you need to complete every time you want to use the Raspberry Pi on **eduroam**. Connect to the Raspberry Pi using the wired connection as before.

6. Once you log into your Raspberry Pi, try to

   ```
   ping google.com
   ```

   If you see that the pings are being successful like before, you are already connected to **eduroam** on your Raspberry Pi!

   Ctrl-C to quit out of the ping cycle.

7. Now, we want to find the IP address of the Raspberry Pi on this specific connection. Here, you want to type

   ```
   ifconfig
   ```

   You will see a large chunk appear. The important one to look for is `wlan0`. As an example, you will see:

```
wlan0:  flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 131.179.45.164 netmask 255.255.254.0 broadcast 131.179.45.255
        inet6 2607:f010:2e9:1b:d757:3d9d:4b43:ffd1 prefixlen 64 scopeid 0x0<gl
```

The important number to look out for is `inet 131.179.45.164`, meaning that my Raspberry Pi has the IP address of 131.179.45.164 right now.

8. Press Ctrl-D to log out of the Raspberry Pi.
9. Now, try to use this IP address to log into the Raspberry Pi instead of the `raspberrypi.local` name. On Mac / Linux, this would be

```
ssh pi@131.179.45.164
```

And on Windows, this would be just typing the IP address into the hostname on PuTTY.

And that's it! You should be able to log in using the IP address. Now, as long as you don't lose power to your Raspberry Pi, you will be able to use this IP address to log into this specific Raspberry Pi while it's connected to **eduroam**. With a battery, this does not even need to be connected to your computer anymore.

6.3 Multiple WiFi Networks (optional)

If you wanted to actually have the Raspberry Pi connect to whatever network that is available to you, you can refer to Connect To Multiple Wireless Networks With A Raspberry Pi. Play around with it if it is convenient. You can just keep two copies of the `wpa_supplicant.conf` files.

7 Initial Setup

Now that you are in your Raspberry Pi, from here on out you will be able to command it to do things using the command line. Don't worry if command line seems unfamiliar to you right now. We will go over the basics in the next tutorial. For now, just follow the instructions as given by this tutorial. Just to get your Raspberry Pi started, you will want to run:

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

These lines will update the standard Linux package manager. From here, you can get a lot of the software necessary to run Linux. One important example would be **git**. While we don't need it today, if you want, you can run

```
sudo apt-get install git
```

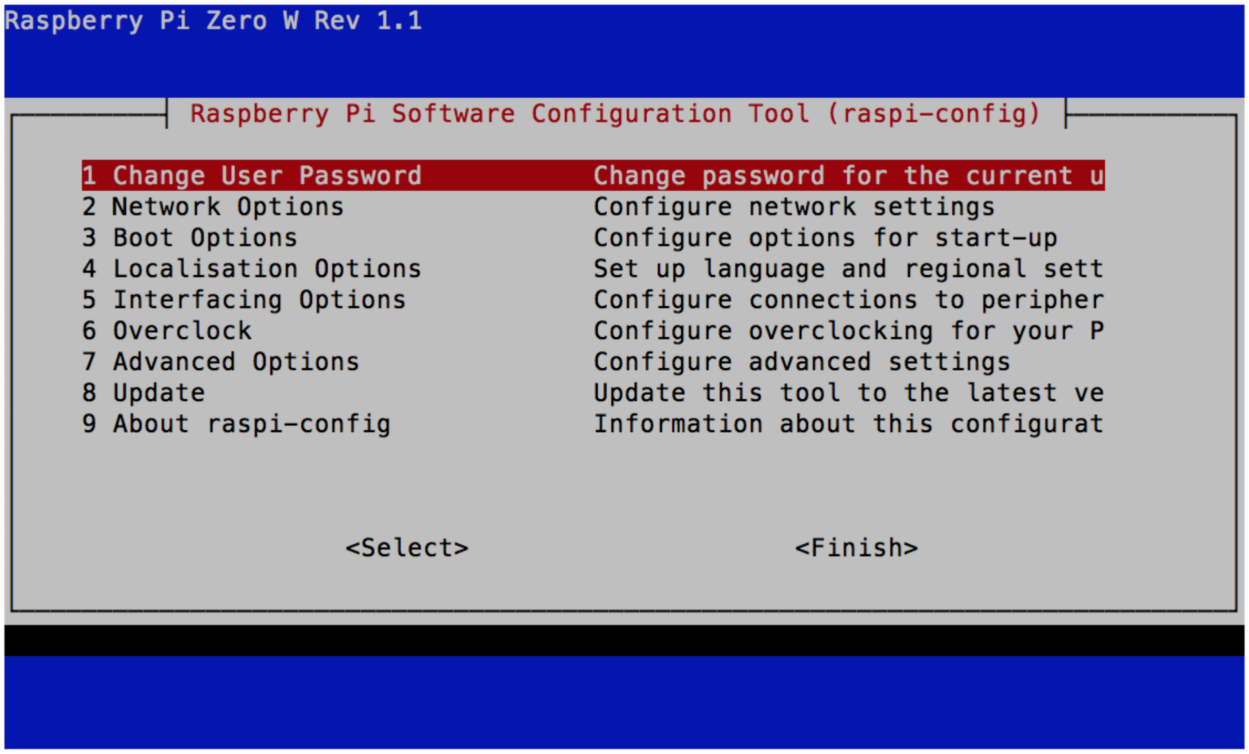
Another thing you can do to change some settings around would be to run

```
sudo raspi-config
```

By doing so, you will open a configuration window similar to a computer's BIOS, shown in Figure 6. For example, one thing you can do is change the hostname from `raspberrypi.local` to some other name, should this be a better solution than just finding the IP address in your eyes.

8 Data Transfer and SFTP

SFTP is known as the *SSH File Transfer Protocol.* It is the standard protocol for transmissions of data files from one computer to another.



```
Raspberry Pi Zero W Rev 1.1

        ┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

   1 Change User Password          Change password for the current u
   2 Network Options               Configure network settings
   3 Boot Options                  Configure options for start-up
   4 Localisation Options          Set up language and regional sett
   5 Interfacing Options           Configure connections to peripher
   6 Overclock                     Configure overclocking for your P
   7 Advanced Options              Configure advanced settings
   8 Update                        Update this tool to the latest ve
   9 About raspi-config            Information about this configurat

                 <Select>                     <Finish>
```

Figure 6: Configuration Page for the Raspberry Pi within the terminal

8.1 Graphical User Interface

Try using Cyberduck or WinSCP (for Windows) to use a Graphical client that does SFTP. This allows for a drag and drop interface that is much more user friendly. Simply set up Cyberduck in the following way and you will find a friendly interface similar to that of just a normal Finder / Explorer window.

The setup is fairly simple. Click “Open Connection” in the top left corner. See Figure 7 to see how you should set it up. Definitely make sure that you have selected SFTP in the top menu. Afterward, it should be fairly straightforward - treat the window just as you would any other file drag and drop like in your native OS.

Once you have successfully setup a connection, the two GUI programs provide a nice feature - the usage of your native computer’s text editors. If you commonly use your own text editor, you can Open a file directly from the Cyberduck / WinSCP directory scheme and edit it as if it were on your computer. The GUI programs will automatically upload the files onto your Raspberry Pi every time it detects a new save to the file.

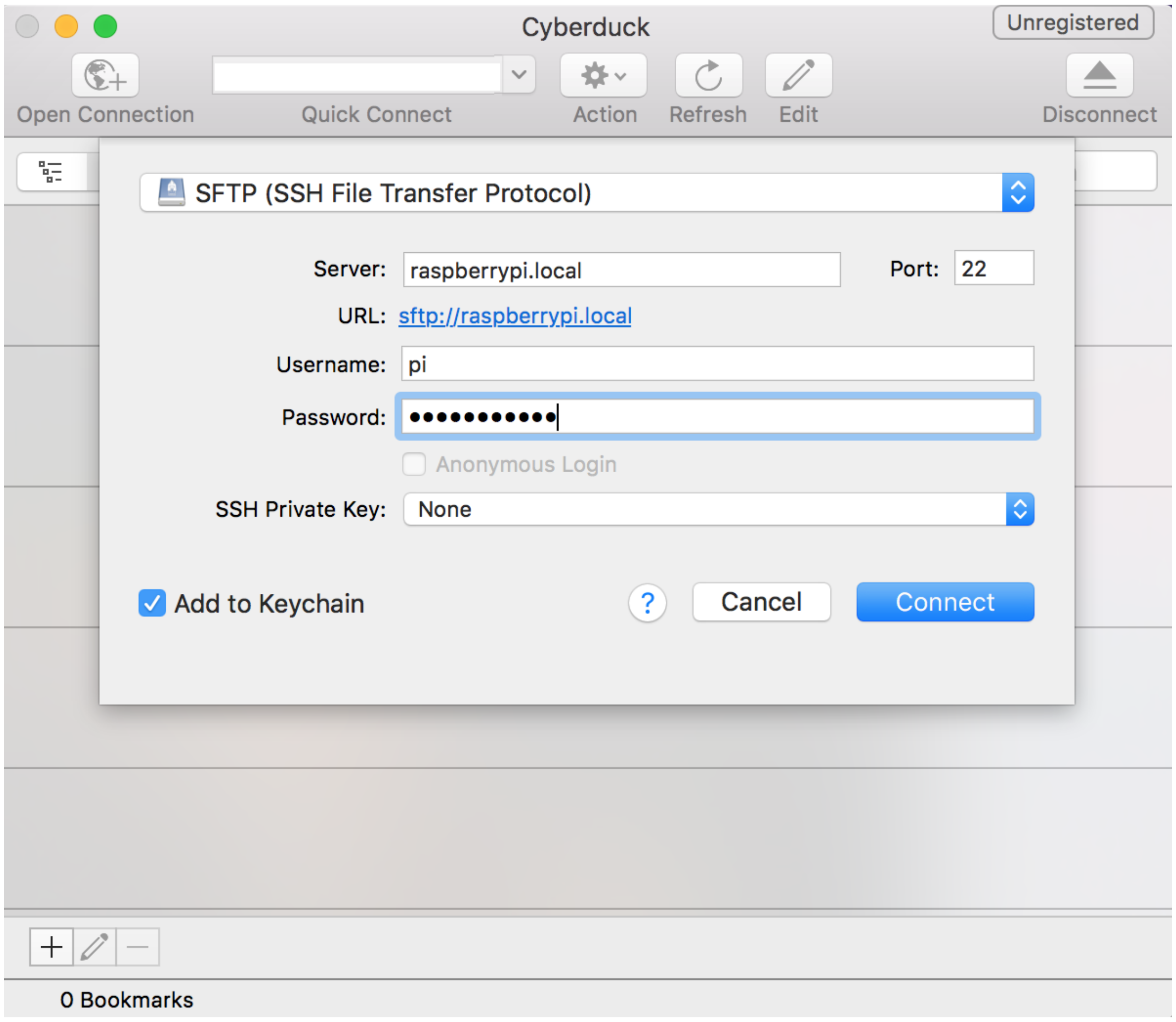
8.2 Git

In theory, this is not an actual data transfer method (this is not SFTP), but can become a makeshift method to obtain work through the Internet. Of course, we have already covered git briefly in the last tutorial, but in this case, because the only way we can interface with git is to use the command line, we do have to learn some of the commands to use git, if you want to use it.

To initially setup a git repository, you type

```
git clone GIT_REPO_URL          # initial setup.
```

in the directory you want to work in. This creates a new directory in your working directory with the name of your git repository. Once you have cloned your repository, you will change into that directory to access the files in your repository.



SFTP (SSH File Transfer Protocol)

Server: raspberrypi.local Port: 22
URL: sftp://raspberrypi.local
Username: pi
Password: ●●●●●●●●●●●
[ ] Anonymous Login
SSH Private Key: None
[✓] Add to Keychain

0 Bookmarks

Figure 7: The Cyberduck Setup Screen

Then, as a "file transfer" method, the two necessary tasks are to push new files into the server:

```
git add -A                      # adds all the new edits into a commit.
git commit -m "message"         # adds a message to your commit
git push origin-master          # or whatever branch you are on, to finalize your changes.
```

to move everything onto the server. And then to retrieve information from the server:

```
git pull
```

to get all the new changes. Git is incredibly powerful and offer much more than just what has been typed here, but these commands alone give enough for you to file share between the two systems.

While this might not be an optimal solution for small edits, and can still feel a bit complicated moving into multiple users, it is an effective way for you to code on your native computer's environment and then sending it over to the Raspberry Pi. The benefit of using this method is that you get all the version control benefits for free.

9 Camera and Library Usage on Raspberry Pi

This section provides the option for you to use to use image processing through your embedded device. However, do note that there are limitations with using the camera that you should test out (of which these issues include battery consumption, memory usage, and processing speed and power).

9.1 Installation

This part follows in a similar path as the Miniconda installation. This will take a little bit longer than the computer version, but will be within the 5-10 minute range.

1. Download Berryconda onto your Raspberry Pi. You want the `Berrycondax-2.0.0-Linux-armv6l.sh`. Using our previous tutorials, one easy way to do this is to use your SFTP / Cyberduck methods to put the file onto your Raspberry Pi. Another potential way is to type this in your Raspberry Pi.

   ```
   wget https://github.com/jjhelmus/berryconda/releases/download/v2.0.0/Berryconda3-2.0.0-Linux-armv6l.sh
   ```

2. Then, as mentioned in the Github startup,

   ```
   chmod +x Berryconda3-2.0.0-Linux-armv6l.sh
   ./Berryconda3-2.0.0-Linux-armv6l.sh
   ```

   Accept the license agreements and such.

3. Restart your Raspberry Pi.

   ```
   sudo reboot -h now
   ```

   Hopefully your eduroam IP address won't change, but there is a chance that you might have to re-get the IP address.

4. Again, as with the Conda installation on your computer, you simply

   ```
   conda install opencv
   ```

9.2 (Optional)-OpenCV Usage Notes

There are limitations that exist if you want to use OpenCV on your Raspberry Pi. In specific, the physical limitations include,

- Battery consumption of the webcam. Using the camera will draw a lot of power from your battery, so know that you may have to use it sparingly in any design that requires a long working duration.
- Memory usage. Each photo (or frame of a video) can be on the order of several MBs. If you save too many such photos, you may overload the limited memory that exists on your Raspberry Pi.
- Computational Power. Do also note that your computers have much better computational power than your Raspberry Pis. As such, some code that may run very quickly on your computer may no longer do so on your Raspberry Pi.

These are all problems that you will have to test for any implementation of camera-based code that you want to implement.

There are some other limitations that are also present with the software. Unlike your computers, which are very integrated with user interface software to view videos, you typically cannot view the pictures / videos that you take with your cameras in real-time. Instead, oftentimes the best you can do is to save a file containing your picture / video and then send them back to your computer for you to view on your

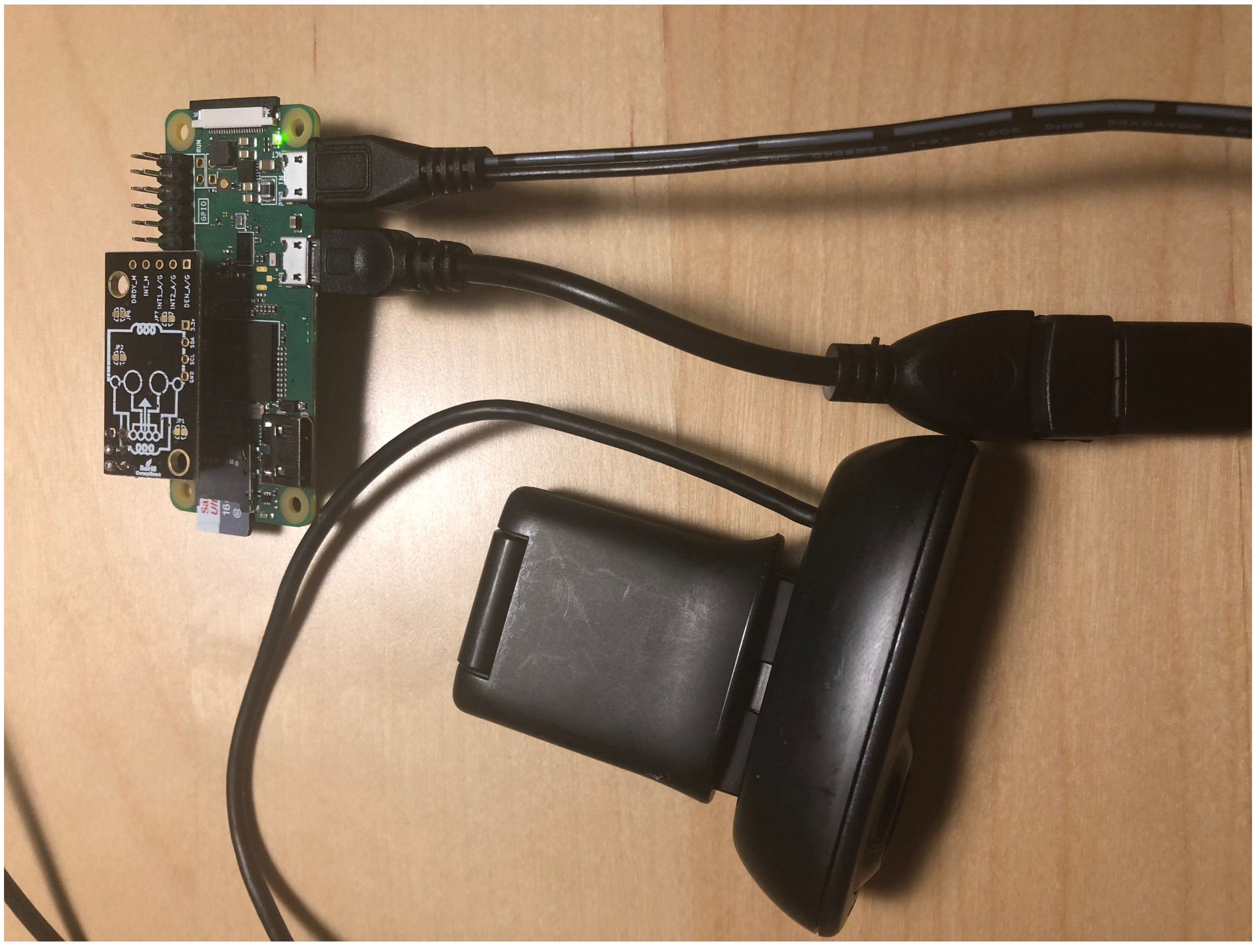


Figure 8: Setup of the Raspberry Pi to use the Camera. Off-camera there is the wire that connects from the OTG cable to the webcam.

computer. Therefore, you must be very careful in how you want to design your OpenCV-based system on Raspberry Pi, because it becomes much more difficult to test.

This section outlines the steps to get OpenCV rendered on your remote device and requires the **USB OTG Host Cable**, an **external power source** and a **USB Webcam** (ask TA for one if you don't have one).

1. Setup: See Figure 8.
   (a) Plug your Raspberry into the external power source (when able to access the ssh into the Raspberry Pi).
   (b) Use the OTG cable in the USB spot and connect it to your webcam.
2. Use OpenCV to take a picture of something on the Raspberry Pi and save it as a file in the Raspberry Pi.
3. Try to verify the success of the image capture by finding the file and the file size in terminal.
4. Use one of the file transfer methods to move this picture onto your computer to see if the picture has been taken properly.

Now, you know how to use OpenCV on your Raspberry Pi! Just remember that because of processing power, there will be slight issues with how quickly it is possible to process images. However, you will definitely be able to get some frames and processing going on.

Because of how you can't see the video feed on the Raspberry Pi, you may want to do most of your development on your computer (knowing about the processor limitations) and then testing it on the Raspberry Pi afterward.

10 Report

1. Git clone your repo from last week into your RPi.
2. Using whatever method it is you like, please have a Python script on your Raspberry Pi. The only requirement of this Python script is for you to have it output something to console, so that you can take a screenshot of it. (Cloning might give you the test.py file from last week!) **Please push the screenshot of the terminal to Github**.

   For those uncomfortable with writing Python code, please refer to this Python tutorial to get familiar with the syntax.
3. Please save the code in your cloned repo on the RPi and **push the changes to Github** WITHOUT merge conflicts from step 2.
4. **Submit** your report for this week on CCLE following guidelines from the Week 1 specification on CCLE. Please also include a small blurb about the difficulties and questions that you had while you were doing this lab (if any).
5. **Get a head-start on next week:** Install Berryconda (see Section 9.1) and clone and run the MQTT-subscriber and publisher code from here.