Lab 1 – Pi Setup

Overview

The MXET 300 lab illustrates the meaning of being a mechatronics engineer: utilizing both the mechanical and electronics fields. In the pre-lab, you were exposed to the Linux environment, Git, GitHub, as well as the Python language. Now you will use some of these concepts to: prepare the device for future labs by flashing the SD card with an OS image, connect the board to a network for headless development (headless refers to using the Pi remotely – without a keyboard or monitor), and prepare local and remote Git repositories for backing up your work. You will be provided a lab kit that includes your team's Raspberry Pi 4 in a custom enclosure and a micro-SD card.

Resources:
The following are links to useful resources for this lab:

- SCUTTLE Homepage
- MXET300-SCUTTLE GitHub
- Guides and Resources

Software Needed:

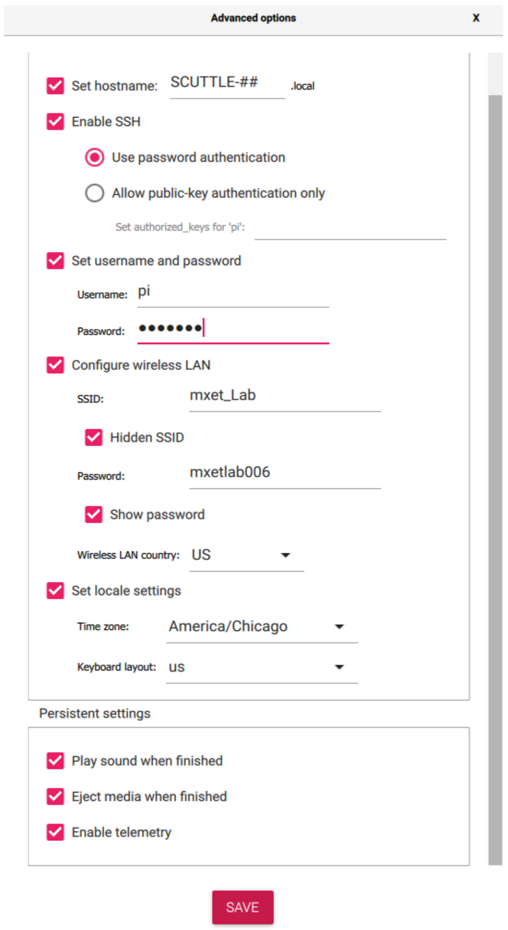
- Raspberry Pi Imager
- SCUTTLE Raspberry Pi OS Image
- Visual Studio Code

NOTE: The SCUTTLE image file is about a gigabyte and should be downloaded ahead of time. This is a part of the pre-lab you must complete before attending lab 1!

Deploying the SCUTTLE Image

The Raspberry Pi 4 used by the SCUTTLE runs a Linux distribution called Raspberry Pi OS, which is the official OS for Raspberry Pi computers. A pre-configured image of this OS has been prepared for the MXET 300 lab and made available for download. It includes additional software like the Node-RED, and Python robotics libraries. A flashing tool called Raspberry Pi Imager will be used to configure a micro-SD card with the SCUTTLE image. This software and the image should have been downloaded ahead of time and can be found in the resources section.

1. Connect the storage device to your laptop, then open the Raspberry Pi Imager program.
2. Select the downloaded `MXET300-SCUTTLE.img.xz` image for the operating system. This is done by scrolling down the drop-down list to the bottom and selecting "Use Custom". Then select micro-SD card as the storage device.
3. Open the settings by clicking the gear icon and copy the following figure. If the gear icon is missing, activate the settings by pressing CTRL + SHIFT + X. Set the user password to "`scuttle`" and make sure the username is "`pi`". For the hostname, you will use "`SCUTTLE-##`", where the ## is your SCUTTLE number. This number can be found on your lab kit as well as on the side of the ethernet port on your Pi. Reference figure 1 below for guidance.


Advanced options

- [✓] Set hostname: SCUTTLE-## .local
- [✓] Enable SSH
  - ● Use password authentication
  - ○ Allow public-key authentication only
  - Set authorized_keys for 'pi':
- [✓] Set username and password
  - Username: pi
  - Password: •••••••
- [✓] Configure wireless LAN
  - SSID: mxet_Lab
  - [✓] Hidden SSID
  - Password: mxetlab006
  - [✓] Show password
  - Wireless LAN country: US
- [✓] Set locale settings
  - Time zone: America/Chicago
  - Keyboard layout: us

Persistent settings

- [✓] Play sound when finished
- [✓] Eject media when finished
- [✓] Enable telemetry

Figure 1: Raspberry Pi Imager Settings

4. Save the settings and start the writing process. This will take several minutes, so one team member should start creating a GitHub account or use an existing one to generate an access token. See steps 3 to 5 of the "GitHub Repository Setup" section.
5. Once complete, the micro-SD may reappear on your laptop and Windows will recommend reformatting. Ignore this and eject the micro-SD card, which is now loaded with the Raspberry Pi OS image for the SCUTTLE.

Booting the Raspberry Pi

Once the micro-SD is flashed with the SCUTTLE image the Pi will be able to boot from it. Do not remove the card while the Pi is powered on. Follow the steps below to boot the Raspberry Pi for the first time, connect to the lab's wireless network, and open the VS Code.

1. Locate the micro-SD slot on the underside of the Raspberry Pi as shown in figure 2 and insert the micro-SD.

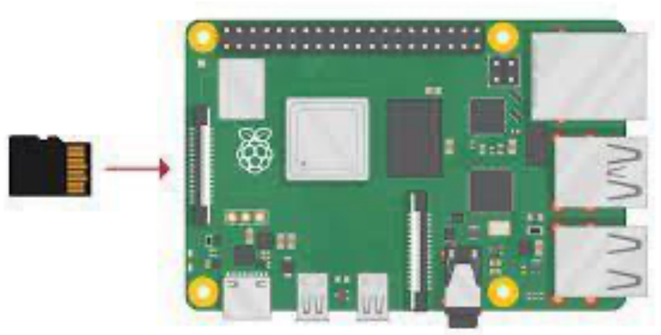


Figure 2: Micro-SD Slot Location

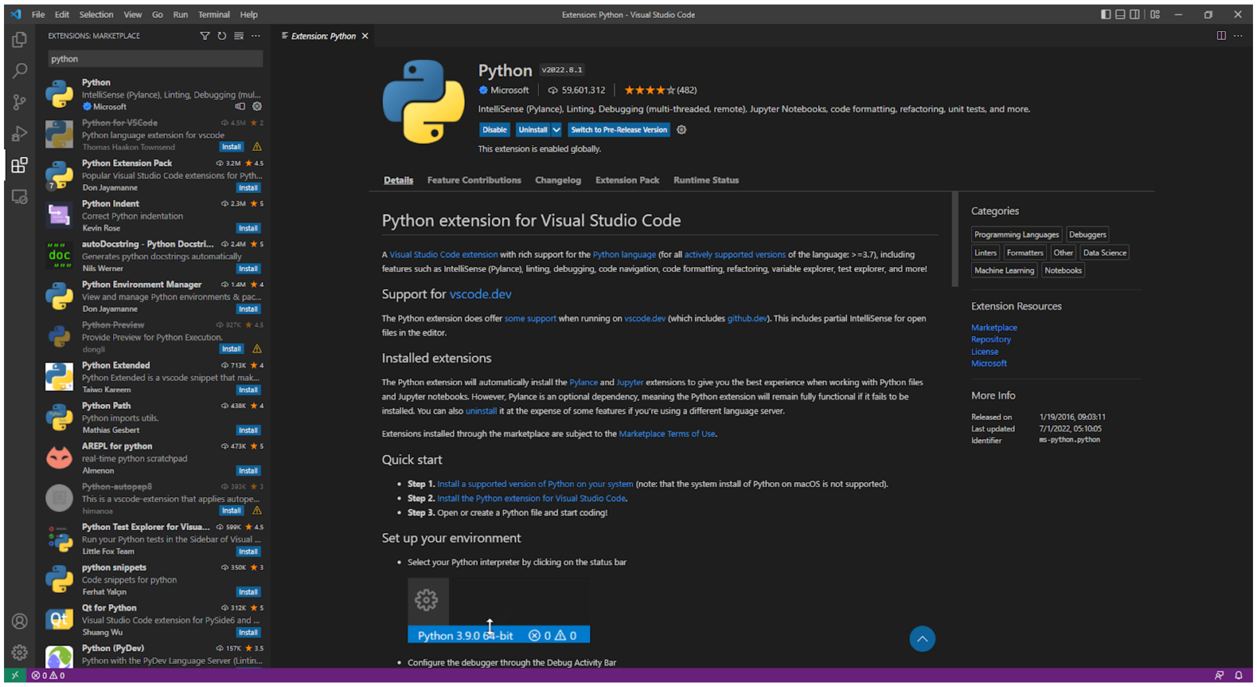
2. Your TA will inform you on how to supply power to the Pi. Once powered, allow a minute to boot the Pi for the first time.
3. While you wait, connect your laptop to the lab's wireless network.
   a. This network is hidden and will not appear in the list of nearby Wi-Fi networks. Instead, select the "Hidden" option.
   b. For the SSID, enter "`mxet_Lab`" and for the password enter "`mxetlab006`".
   c. This is necessary to connect to your Pi through a shared network for use in headless mode.
4. Once the Pi boots, the OLED display on the enclosure should turn on and indicate the network SSID that the Pi is connected to along with the IP address and battery voltage. The four bytes of the IP address should be 192.168.1.1##, where the pound signs are the unique value indicating the unique ID of the Pi labeled on the side of the ethernet port.
   a. If the display does not start at all the electrical connection is the first suspect. Remove the enclosure lid and verify that LED indicators are lit for the devices connected to the Pi.
   b. If the IP is all zeros, the display service ran but the Pi did not connect to any networks or the network interface being used is not the one the display service expects to look for. Most likely, the Pi's default network was entered incorrectly when flashing. Ask your TA for assistance in modifying the wireless configuration to fix this without re-flashing.

c. An alternate connection option is to use an ethernet cable to connect the Pi to your laptop. This is good for debugging or quick access without needing to connect your Pi to any wireless networks. This is likely what your TA will do if your Pi needs to be reconfigured to connect correctly for wireless use. If you do this and the display is running properly, the IP should update automatically in about ten seconds to the IP dynamically assigned by the host (your laptop). This is not the same static IP assigned by the lab's network and will not match the format given in step 4.

d. If you use a USB-C power supply instead of the kit's battery, the display will not include a battery voltage.

Visual Studio Code Setup

1. Download Visual Studio Code, run the installer, and follow its instructions.
2. Once installation is complete you will need to install the extensions.
   a. Select the "extensions" tab from the toolbar on the left.
   b. Search "python" and select the Python extension from Microsoft. This extension allows VS Code to interpret and highlight Python along with some other useful features. Install the extension.



Python — IntelliSense (Pylance), Linting, Debugging (mul... — Microsoft
~~Python-for-VSCode~~ 4.5M ★ 2 — Python language extension for vscode — Thomas Haakon Townsend
Python Extension Pack 3.2M ★ 4.5 — Popular Visual Studio Code extensions for Pyth... — Don Jayamanne
Python Indent 2.3M ★ 5 — Correct Python indentation — Kevin Rose
autoDocstring - Python Docstri... 2.4M ★ 5 — Generates python docstrings automatically — Nils Werner
Python Environment Manager 1.4M ★ 4 — View and manage Python environments & pac... — Don Jayamanne
~~Python-Preview~~ 927K ★ 4.5 — Provide Preview for Python Execution. — dongli
Python Extended 713K ★ 4 — Python Extended is a vscode snippet that mak... — Taiwo Kareem
Python Path 438K ★ 4 — Python imports utils. — Mathias Gesbert
AREPL for python 473K ★ 5 — real-time python scratchpad — Almenon
~~Python-autopep8~~ 393K ★ 3 — This is a vscode-extension that applies autope... — himanoa
Python Test Explorer for Visua... 599K ★ 4.5 — Run your Python tests in the Sidebar of Visual ... — Little Fox Team
python snippets 350K ★ 3 — Code snippets for python — Ferhat Yalçın
Qt for Python 312K ★ 5 — Visual Studio Code extension for PySide6 and ... — Shuang Wu
Python (PyDev) 157K ★ 3.5 — Python with the PyDev Language Server (Lintin...

Python v2022.8.1
Microsoft | 59,601,312 | ★★★★☆ (482)
IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote), Jupyter Notebooks, code formatting, refactoring, unit tests, and more.
This extension is enabled globally.

Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the Python language (for all actively supported versions of the language: >=3.7), including features such as IntelliSense (Pylance), linting, debugging, code navigation, code formatting, refactoring, variable explorer, test explorer, and more!

Support for vscode.dev

The Python extension does offer some support when running on vscode.dev (which includes github.dev). This includes partial IntelliSense for open files in the editor.

Installed extensions

The Python extension will automatically install the Pylance and Jupyter extensions to give you the best experience when working with Python files and Jupyter notebooks. However, Pylance is an optional dependency, meaning the Python extension will remain fully functional if it fails to be installed. You can also uninstall it at the expense of some features if you're using a different language server.

Extensions installed through the marketplace are subject to the Marketplace Terms of Use.

Quick start

- **Step 1.** Install a supported version of Python on your system (note: that the system install of Python on macOS is not supported).
- **Step 2.** Install the Python extension for Visual Studio Code.
- **Step 3.** Open or create a Python file and start coding!

Set up your environment

- Select your Python interpreter by clicking on the status bar
- Configure the debugger through the Debug Activity Bar

Categories
Programming Languages | Debuggers | Linters | Formatters | Other | Data Science | Machine Learning | Notebooks

Extension Resources
Marketplace
Repository
License
Microsoft

More Info

Released on	1/19/2016, 09:03:11
Last updated	7/1/2022, 05:10:05
Identifier	ms-python.python

Figure 3: Installing Visual Studio Code Extensions

   c. Next, search for the "remote SSH" extension. Select and install the first option, which should be "Remote - SSH" from Microsoft.
   d. Once installation is complete, restart your VS Code client.
3. Now you will connect to the Pi using remote SSH.
   a. First, switch to the "Remote Explorer" tab by clicking a monitor icon on the left side. Click the plus icon besides "SSH". It should look like figure 4. Then enter

"ssh pi@<ip address>". The IP address is displayed on the OLED. Next, select the user SSH configuration file to update. The file name should contain your laptop's host name and "/.ssh/config". Hit connect button add the bottom right corner of the VS Code window.

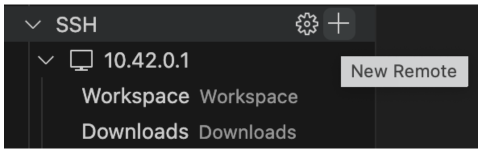


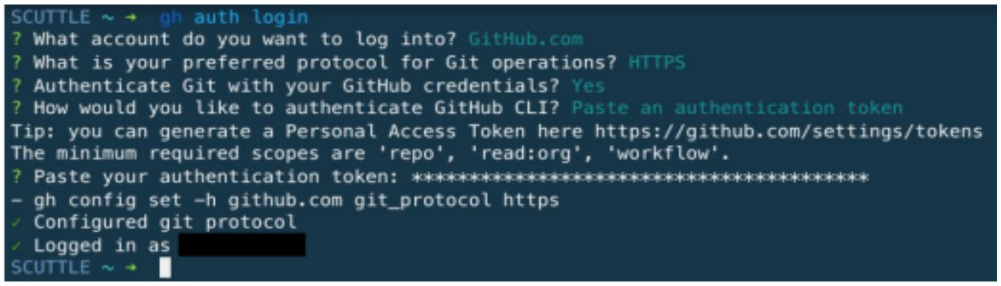
Figure 4. add new host to Remote Explorer

b. Now, a new window would pop up. Select the platform, Linux. Enter your Raspberry Pi password, "scuttle". It would take about 5 minutes to load the VS Code remote server to your Raspberry Pi.
c. Once complete, your VS Code client will connect to your Raspberry Pi. After this connection completes you can read or write files remotely as well as interpret and highlight Python syntax.
d. If the connection failed due to the connection time out, try this solution. Open your settings in your Preferences tab > Settings. Search for "useLocalServer" and disable this setting.
"remote.SSH.useLocalServer": false

GitHub Repository Setup

To maintain a backup of your work you will set up a local Git repository and configure it to upload changes to a remote repository on GitHub. This acts as a backup of your team's project on the Pi and allows you to track or revert committed changes. Your team must commit and push the SCUTTLE workspace at the end of every lab to maintain up to date backups.

1. Decide amongst your team whom will host your team's GitHub repository under their account. Then have that teammate create a **private** repository on GitHub using their account. Name the repository "`mxet300_lab`". Be sure to add a README file. This is important as it will initialize your repository with the README file and set "main" as the default branch. It is also important and good practice to include a README file that describes what the repo is about.
2. Add your TAs as collaborators on the GitHub repository.
3. Next, the GitHub repository creator should generate a personal access token (classic). In the Developer settings and under the Personal access tokens, click Tokens (classic). Select Generate new token, then click Generate new token (classic). This video also explains the process. The PAT is used to authenticate your account access without a password when you want to push to the remote repository.
    a. Set the expiration to be at least the end of the semester and select the 'repo', 'workflow', and 'admin:org' scopes.
4. Next you will cache the PAT. Open the terminal with Ctrl + `. Enter the command "gh auth login" and follow the same responses to the prompts as shown in figure 5.



```
SCUTTLE ~ → gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Paste an authentication token
Tip: you can generate a Personal Access Token here https://github.com/settings/tokens
The minimum required scopes are 'repo', 'read:org', 'workflow'.
? Paste your authentication token: ****************************************
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as [redacted]
SCUTTLE ~ → 
```

Figure 5: GitHub PAT Configuration

5. Now configure Git with your username and email using these commands:
    a. git config --global user.name "USERNAME"
    b. git config --global user.email "EMAIL"

   NOTE: These should be the username and email associated to your GitHub
6. Next, ensure you are in your home directory. If you are not, run the command `cd ~`. This will return you back to the user's (pi) home directory. You can run the command `pwd` to double check. The command line should return back "*`/home/pi`*"
7. After you have ensured you are back in this present working directory, run the command: `git clone THE URL TO YOUR REPOSITORY`
    a. This command will "clone" the repository you created in your GitHub account and initialize the repository within the present working directory that you are in on your device. This is perhaps the quickest and easiest method to establishing a GitHub repository that you intend on working with on a remote device.
    b. Note there are other methods that can be used. They would first entail creating a local git file on the device and running a few commands to setup the link between the local git file and the GitHub repository. These methods were explored and shown in your pre-lab exercise and videos, and you can see them detailed in GitHub upon creating a repo without initializing it with a README file.
8. To add "`mxet300_lab`" folder to your host, go to the Remote Explorer and click the folder-plus icon which should said "Connect in New Window…" when your cursor hovers above the icon. Enter the password in the new window. In the Explorer, click Open Folder button, enter the path, "`/home/pi/mxet300_lab`", and hit enter. Enter the password again after the window is reloaded. Now, go to the Remote Explorer and refresh the remotes like figure 8. You should see "`mxet300_lab`" folder under your host.

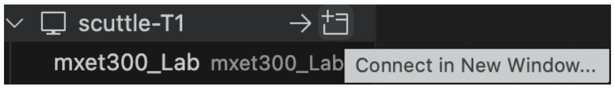


Figure 6: Add mxet300_lab folder in Remote Explorer

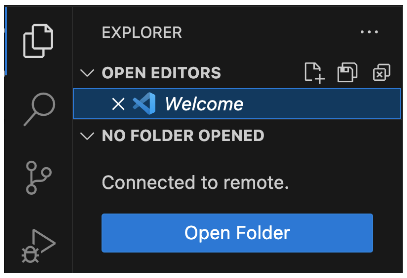


Figure 7: Open folder with the path, "/home/pi/mxet300_lab"

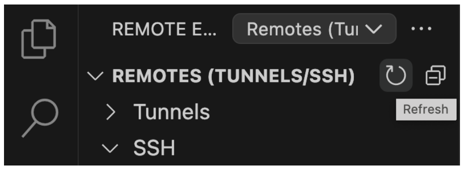


Figure 8: Refresh your remote explorer and check if mxet300_lab folder shows up

9. Now you can see a README.md file that was created when you initialized your GitHub repo. The VS Code workspace panel would look like the following in figure 9.

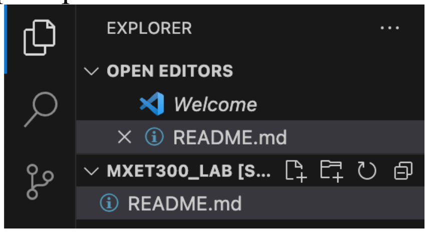
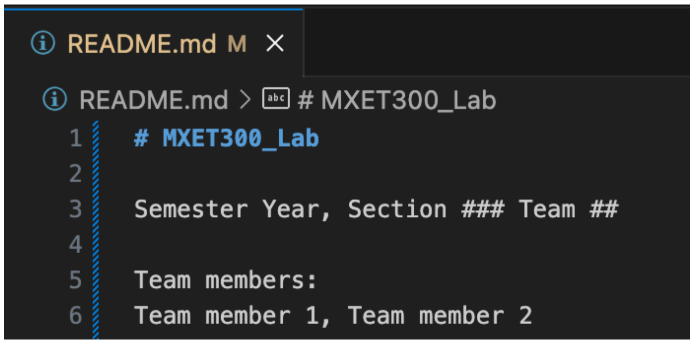


Figure 9: Workspace after git clone

10. Now you will do your first GitHub push after first editing the README.md file.
    a. Open the README.md file by clicking on it in the workspace pane or use nano to edit the file.
    b. With the README.md file open, please add your section number, team number and team members in the format you can see in figure 10 below. Then save and close the file.



```
# MXET300_Lab

Semester Year, Section ### Team ##

Team members:
Team member 1, Team member 2
```

Figure 10: README.md file template

11. Another very useful command to be aware of is `cat`, this will display the entire contents of a text file within the command line. Now run the command `cat README.md` to try it out and see what is in the README.md file.
12. Now you have completed setting up the GitHub repo on your Pi and have made your first edits that you want to have "pushed" to the GitHub repo. To push edits going forward, you will only need three commands. Run these commands in this order and you will have completed your first push to your new MXET300_Lab repo.
    a. `git add .`
    b. `git commit –m "first commit message"`
    c. `git push`
13. Navigate to your GitHub repo and refresh the page. You should now see the effects of the push with the changes visible. After you have completed this, ask your TA for check-off.

NOTE: The first commit message can be whatever you want it to be, this is a message/ note that you leave with the push so you when you go back to read your commits, you can know what that commit was about. This is a note to yourself, make it clear and concise so you know what was going on.

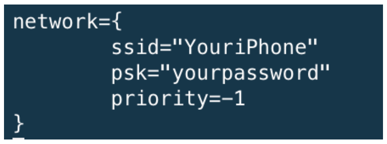
The local Git repository you have created is set up on the account configured above, so only one GitHub account is required per team.

Configuring Wireless Connections

At this point your Pi will connect automatically to the wireless network in the lab, however this will only allow access to the Pi in the Fermier building. In order to connect elsewhere you will need to add the networks you want to use to the `wpa_supplicant.conf` file. Once added, your Pi will attempt to connect to these networks on startup. You need to establish at least one reliable, non-ethernet method to connect to the Pi outside the lab to complete this task.

1. Now that you can connect to the Pi and use an IDE to read or write files, you will need to configure your Pi to connect to additional networks outside the lab.
2. In the terminal, enter the following command to use the terminal-based text editor Nano to edit a configuration file: `sudo nano /etc/wpa_supplicant/wpa_supplicant.conf`

   You should see the file's contents in the terminal. Now you will see how to add networks to this file so that your Pi automatically connects when they become available.
   NOTE: The hotspot name should not include any spaces or special characters (e.g. *'/\)($%^). Keep the name simple with characters and numbers only.



```
network={
        ssid="YouriPhone"
        psk="yourpassword"
        priority=-1
}
```

Figure 11: Network block example

   To save the file after editing in Nano, hit "Ctrl + X" to exit, enter "Y" for change confirmation, and click "Enter" to confirm the filename.

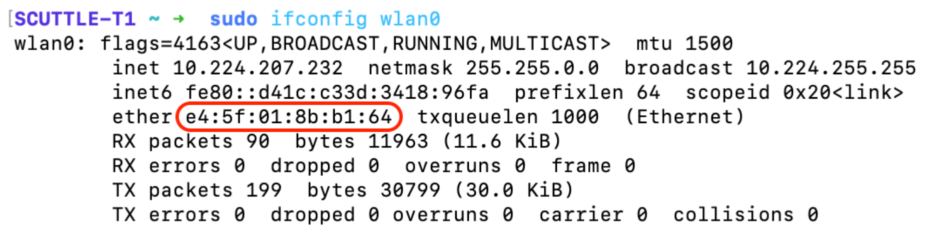
NOTE: We cannot use VS Code's built in text editor to modify this file because it exists outside the IDE's workspace directory.

3. Figure 11 shows an example of a "network block" in the `wpa_supplicant.conf` file. Follow this format when adding networks and do not remove existing blocks. You can ignore the text followed by "#" indicating a comment. The next steps describe some recommended options for adding networks to this file.
   a. If your phone or laptop can broadcast a mobile hotspot, add a network block for that connection. You may need to change your hotspot name from the default to avoid conflict with other students' devices. This method is highly recommended even if you do not plan to use the hotspot often. Later in the lab you will be able

to drive the SCUTTLE using your mobile device as a gamepad and a mobile hotspot would permit driving the SCUTTLE anywhere regardless of network range or availability.

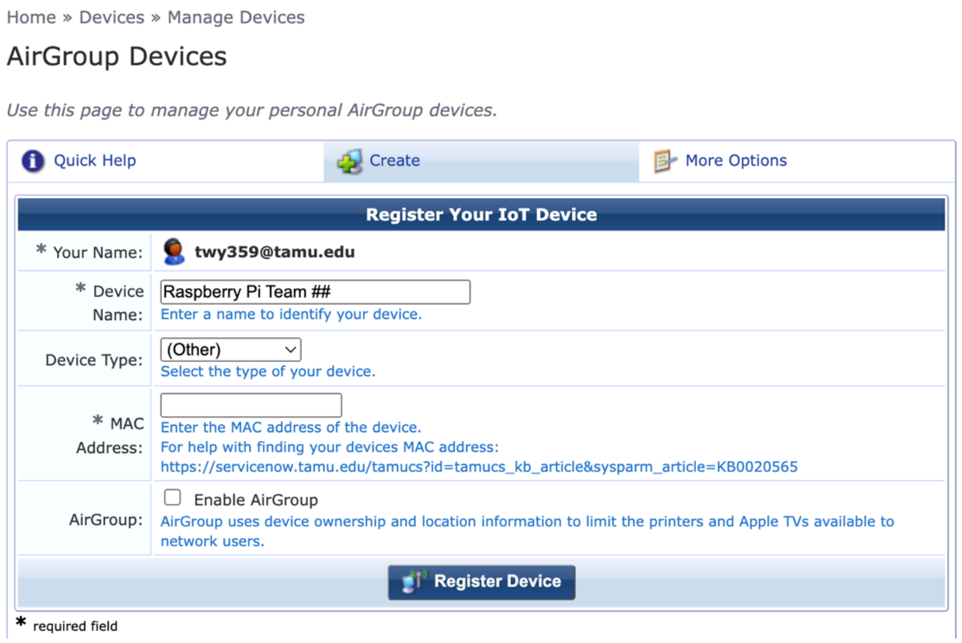
b. You can also add your home's WiFi network to the list. If you do not remember the exact name of your home network, you will need to add this network block later. Ensure you have either an ethernet cable or a mobile hotspot configured so that you can connect to edit `wpa_supplicant.conf` at home.

c. Adding the TAMU IoT network allows you to connect to Raspberry Pi while you are on campus outside of the lab. First, copy your Raspberry Pi WIFI MAC address which can be found using the command, `sudo ifconfig wlan0`. The example command is shown in figure 12. Second, open the [TAMU IoT portal](), and create a device in the whitelist. Shown in figure 13. In this case, your device is Raspberry Pi but you can rename it to "Raspberry Pi Team ##". Leave device type as *(Other)*. Enter your Raspberry Pi WIFI MAC address, in the MAC Address column. And hit Register Device once you finished. Third, in the `wpa_supplicant.conf` file, add TAMU IoT network as shown in figure 14.



```
[SCUTTLE-T1 ~ ➜  sudo ifconfig wlan0
 wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.224.207.232  netmask 255.255.0.0  broadcast 10.224.255.255
        inet6 fe80::d41c:c33d:3418:96fa  prefixlen 64  scopeid 0x20<link>
        ether e4:5f:01:8b:b1:64  txqueuelen 1000  (Ethernet)
        RX packets 90  bytes 11963 (11.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 199  bytes 30799 (30.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 12: Read Raspberry Pi WIFI MAC address.



Home » Devices » Manage Devices

AirGroup Devices

Use this page to manage your personal AirGroup devices.

Register Your IoT Device

* Your Name:	twy359@tamu.edu
* Device Name:	Raspberry Pi Team ## Enter a name to identify your device.
Device Type:	(Other) Select the type of your device.
* MAC Address:	Enter the MAC address of the device. For help with finding your devices MAC address: https://servicenow.tamu.edu/tamucs?id=tamucs_kb_article&sysparm_article=KB0020565
AirGroup:	☐ Enable AirGroup AirGroup uses device ownership and location information to limit the printers and Apple TVs available to network users.

* required field

Figure 13: Use TAMU IoT Portal to create your device.

```
network={
        ssid="TAMU_IoT"
        priority=5      # lowest priority means we always try this one last
        key_mgmt=NONE   # no password needed
}
```

Figure 14: TAMU IoT network block example

NOTE: If you are adding your Raspberry Pi to TAMU IoT, make sure to add your laptop's wireless MAC address to your TAMU IoT portal while your laptop is connected on TAMU-WiFi.

NOTE: You can establish priority levels to the networks you have in the `wpa_supplicant.conf`. You can reference figure 15 below as an example. Pay attention that the higher priority number creates the higher priority. It would be a good idea to establish your mobile hotspot as the highest priority so that it can be used as an assured connection to a network you have control over and access to.

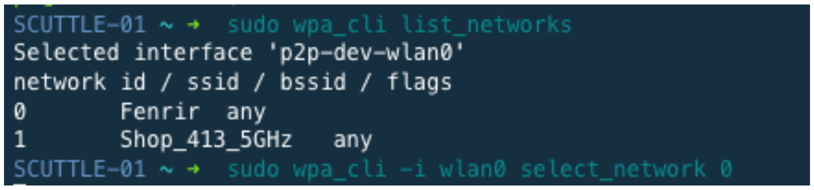
```
network={
    ssid="wifi_A"
    psk="passwordOfA"
    priority=1 #lower priority
}
network={
   ssid="wifi_B"
   psk="passwordOfB"
   priority=2 #higher priority
}
```

Figure 15: Network Priority Example

NOTE: There are a couple of commands that you can run within the command line to change networks on the fly without rebooting. They are as follows:

- `sudo wpa_cli –i wlan0 reconfig` # reconfigure network
- `sudo wpa_cli list_networks` # show listed networks in the config file
- `sudo wpa_cli –i wlan0 select_network 0` #0, 1, etc. Note: Networks are NOT the same number as your priority in the wpa_supplicant

Recall that the text after # on that line is a comment. Below is screenshot of the command in action. Ask your TA if you don't understand the commands and need some clarification on how to use this:



```
SCUTTLE-01 ~ ➜  sudo wpa_cli list_networks
Selected interface 'p2p-dev-wlan0'
network id / ssid / bssid / flags
0       Fenrir  any
1       Shop_413_5GHz   any
SCUTTLE-01 ~ ➜  sudo wpa_cli -i wlan0 select_network 0
```

Figure 16: Network Priority Example

4. Once you finish adding the networks you plan to use, press "CTRL+X" to close the editor and then "Y" to confirm saving your changes before exiting. The Pi will need to restart for the changes to take effect, so use the command `sudo reboot` to initiate a reboot and ensure everything works properly after the changes were made.
    a. If your Pi is no longer connecting after changing this file and restarting, you may need to use an ethernet cable to connect and modify the configuration again. Ask your TA to verify the contents of the file as well.

5. You can add more networks to this file if you need to. If you want to read more about the format and options available for wpa_supplicant, this page from FreeBSD includes useful, albeit somewhat technical, documentation.

NOTE: Take your kit with you and be sure to bring the entire kit to each lab. You and your lab partner(s) are responsible for each and every item that is within the kit. Failure to bring back the kit and each item with in the kit and in the order in which you received them will result in loss of points to your grade.

Terminal Fun

If you are a little curious and want to see some fun things you can do in the terminal, try running some of these commands that have already been installed.

```
fortune | cowsay | lolcat
figlet MXET 300 | lolcat
cowsay -f dragon SCUTTLE
fortune | cowsay -f ghostbusters | lolcat
```

Try playing with these to see what they are doing. Remember you can use the `-- help` argument after a command to view help info on the command. You can also ask the TA and google to look them up.

Post-Lab Questions

1. Describe the two main methods for obtaining a connection to the Raspberry Pi.

2. What is Git and why might it be useful for managing software in a project?

3. Describe the commands for adding, committing, and pushing changes in a local repository to GitHub. If the push is successful, you would see updates on your remote repository. If your push command is rejected, it is likely because your local repository is behind the remote repository or when your teammate changes the code remotely without updating on your local files, what is the command you need to update your outdated local repository?