Lab 2 – Displays and GUI

Overview

This lab introduces the program hierarchy that will be used for this lab and on the SCUTTLE. The activity involves editing your kit's OLED display script, which is called on startup to run the display automatically. You will also learn how to use Node-RED to create a browser accessible GUI for displaying information. This interface is hosted by the Pi and is accessible on your laptop or mobile device, so long as they share the same network as your Pi.

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

- MXET300-SCUTTLE GitHub
- Import and export Node-RED flows
- Display data using Node-RED
- SCUTTLE homepage
- Guides and resources

Software Needed:

- Python scripts:
  - L1_ina.py
  - L1_oled.py
  - L1_log.py

Power to the Pi Setup

For this lab, you will need to power your Raspberry Pi utilizing the battery pack and the custom made "Y" power cable. The reason for this custom made "Y" power cable is so that the source power to the Pi can be read by the INA219 current sensor that is assembled into the lid of the Pi case. As with everything you need, this cable can be found in your lab kit.

NOTE: The "Y" power cable has two ends with Anderson power pole connectors and one end has a set of ferrule ends. There is a short power pole length end and a longer power pole length end. The longer power pole length end will go into your battery pack and the shorter power pole length end will go into the DC converter. The ferrule ends will be screwed down into the terminal block of the INA219. Refer to figure 1 to see how this cable is used.

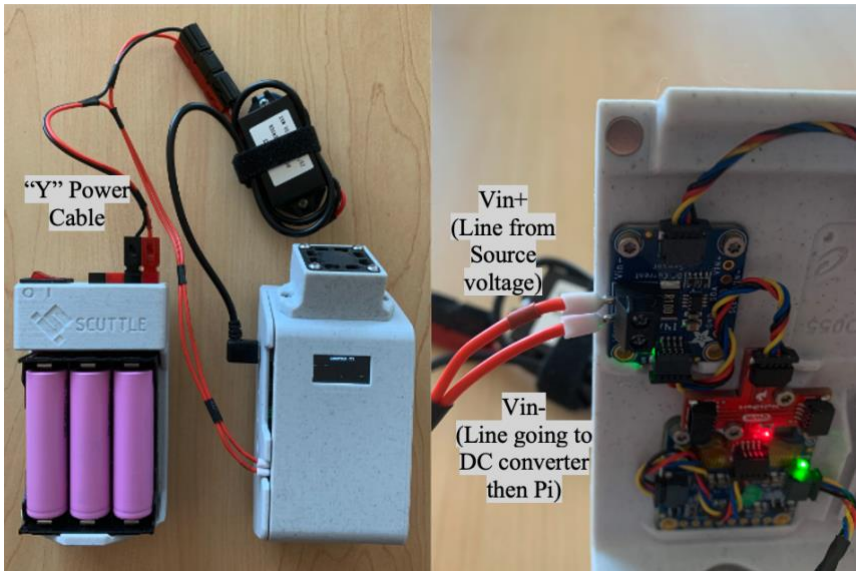

Figure 1: Connecting power to Pi while reading source voltage (Battery Pack) with INA219

1. In your kit there is a 3mm Phillips screwdriver. Use the provided Phillips screwdriver to gently unscrew the screws on terminal block just enough to slide the ferrule ends into the housing.

NOTE: Please use some caution when driving the screws on the terminal block. You only need to screw down the screws just enough for the ferrule ends to not slip out. **DO NOT overtighten or use the wrong sized driver**. This will cause stripping of the screws and over time can cause the need to replace the terminal block, a costly, inconvenient and time-consuming task.

2. Connect the red line from the battery pack (source voltage) to the Vin+ port of the terminal block and the red line going to the DC converter to the Vin- port of the terminal block. Note that the Vin+ line has a small piece of red heat shrink that will discern it from the Vin- line.

NOTE: The INA219 is capable of reading up to +26VDC and the DC converter takes in 24/12 VDC and steps it down to the necessary 5VDC 3A needed to power the Pi. Thus, the Pi can also be supplied power while reading voltage using another source such as the DC power supply in your kit since it outputs 12 VDC. Refer to figure 2 below for an example setup.

Figure 2: Connecting power to Pi while reading source voltage (DC Power Cable) with INA219

3. Check with your TA if you need assistance or clarification. Proceed to the next task once your setup is done.

Read Battery Voltage

Level 1 programs are intended to access data from a sensor or send commands to an actuator. They exist at the lowest level in the SCUTTLE's program hierarchy; handling individual components or basic processes. In this task, you will use an existing L1 Python script to read a current sensor's measurements and print them to the terminal. The sensor is an INA219 configured to measure current through a fixed-resistance shunt, which then allows voltage to be calculated to determine the battery's charge level. (*Recall Ohm's law*)

1. Power the Raspberry Pi by toggling on the battery pack, wait for Pi to connect to the network, and open VS Code terminal `Ctrl+``.
2. Now, `cd` into your mxet300_lab directory/folder and use the command `mkdir basics` to make a directory named basics. Your workspace tree should look similar to figure 3.

NOTE: Your workspace may look different if you named your GitHub repository slightly different than `mxet300_lab`, though the basics folder should still be within.

Figure 3: Create a basics folder

3. In a different browser tab, access the SCUTTLE GitHub repository and navigate to the `MXET300-SCUTTLE/software/basics` directory. This contains the Python scripts needed for all labs.
4. Locate and open the L1_ina.py file and then click on the "Raw" button, as seen in figure 4, to open the raw file in a new tab.

Figure 4: Highlighted raw link on GitHub

5. Navigate back to the terminal in VS Code, and then `cd` into the *basics* folder. Remember you can use the `pwd` command to check your present working directory and ensure you are in the *basics* folder.
6. Now you will utilize the `wget` command to download the following files to the *basics* directory from GitHub:
   L1_ina.py
   L1_oled.py
   L1_log.py

NOTE: `wget` is a program that retrieves content from web servers by providing the URL to the web page. The structure of the command is: `wget URL_OF_WEB_PAGE`

To retrieve the L1_ina.py file, run the command:
`wget https://raw.githubusercontent.com/MXET-Lab/MXET300-SCUTTLE/master/software/basics/L1_ina.py` as shown in figure 5 to download the *L1_ina.py* file to your *basics* folder.

```
SCUTTLE-T1 mxet300_lab git:(main) → cd basics/
SCUTTLE-T1 basics git:(main) → wget https://raw.githubusercontent.com/MXET-Lab/MXET300-SCUTTLE/master/software/basics/L1_ina.py
--2023-08-24 16:42:44--  https://raw.githubusercontent.com/MXET-Lab/MXET300-SCUTTLE/master/software/basics/L1_ina.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.110.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 952 [text/plain]
Saving to: 'L1_ina.py.1'

L1_ina.py.1                         100%[=============================================================================>]

2023-08-24 16:42:44 (523 KB/s) - 'L1_ina.py.1' saved [952/952]
```

Figure 5: Usage example of wget command

NOTE: You must always click on the raw button to view just the code to properly utilize wget to download the actual file from GitHub you are wanting. Otherwise, you will be downloading HTML code of the previous page prior to clicking the "Raw" button.

7. Repeat the same process to `wget` the *L1_oled.py* and *L1_log.py* files.
8. Now you will run the current sensor script, *L1_ina.py*, in the VS Code using the terminal.
    a. Use the following command to run the *L1_ina.py* script and obtain measurements from the INA219: `python3 L1_ina.py`
    b. Observe the output on the terminal, understand the units and magnitudes of the values. Terminate running the script with `Ctrl+C`
    c. Capture a screenshot of the output for your lab report.

Edit OLED Display

Now that you can determine the SCUTTLE's battery voltage with the INA219, you will modify an incomplete version of the Python script currently running the OLED to recreate the voltage readout available by default. You should also give your SCUTTLE a unique name, which is entirely up to your team to decide. Finally, the default OLED script will be replaced with your team's version once it is verified to work.

1. Stop the default OLED service script that runs on boot in order to prevent conflicts when running the *L1_oled.py* script, with the command:
    `sudo systemctl stop oled.service`
2. Run the *L1_oled.py* and notice the behavior on the OLED display and the terminal. The script by default will only display the IP when ran. Terminate running the script with `Ctrl+C`
3. Now open the *L1_oled.py* script and read through it to understand the comments and code. Double click *L1_oled.py* in the VS Code file explorer to open the file and edit.
    a. Pay attention to the comments and the sections marked as "# Task:" for guidance toward step 4.
4. Modify the script and add the following information to the display:
    a. Robot name
    b. Battery voltage, with a label and units

5. Save then execute the script using Python 3 to test your display code. Show your TA once it shows an updating battery voltage and your SCUTTLE's name.
6. Once verified by your TA, terminate the script.
7. Finally, you will modify *oled.py*. It is the default OLED script that runs on boot. It is slightly different from the L1_oled version, but still functions in a very similar way.
    a. From GitHub, download the *oled.py* script from the oled_service folder. Use the same `wget` method used earlier.
    b. Find line 98 and change the string "SCUTTLE" to the name of your robot.
    c. Next you need to replace the original *oled.py* script with the modified one. Make sure you are in the same directory as your modified script and use the command `sudo mv oled.py /usr/share/pyshared/oled.py`
8. You may also add anything else, so long as it does not prevent the display from functioning normally. See the Pillow documentation if you want to try customizing the display to include more than text.
9. Stop the previous script if you haven't already done so and restart the OLED service with the command: `sudo systemctl restart oled.service`

   Verify that your display functions as expected before continuing.

Output Data to Node-RED GUI

Node-RED is a graphical programming interface like LabView, and it is running by default on your Pi's image. You can create and "deploy" your Node-RED flows and view the dashboard for real time data in your web browser from a laptop or mobile device so long as they share the same network as the Pi.

1. First, information is needed to display in the first place. The OLED already shows battery voltage, but you need to be physically present to see it. Node-RED allows you to create a dashboard that shows voltage on a gauge, which makes it easy to monitor the SCUTTLE at a distance.
2. In order for Node-RED to actually obtain the voltage measurement, it needs to know where to find it. There are many options for inputting data to a Node-RED flow, but the simplest is to have the script obtaining data write it to a file and have Node-RED read that file's contents.
3. Read through *L1_log.py*, which provides a series of useful functions for saving information to a file.
    a. Be aware of the distinction between *tmpFile* and *uniqueFile* functions. The former creates the file in a temporary directory which is cleared on restart and the latter will create the file in a unique directory, which will be retained.
    b. Using *tmpFile* is recommended to avoid cluttering your workspace with logs, but either method is acceptable.
4. Create a new script called *L2_telemetry.py* that reads voltage measurements from *L1_ina.py* and saves them to a file using *L1_log.py*.
    a. Create the *L2_telemetry.py* using the command:

```
touch L2_telemetry.py
```

touch is a command that creates a file with the name and extension specified.

b. Refer to the OLED scripts for a reminder on how to retrieve a voltage from the INA219. Remember to import the scripts you need to extend.
c. The name of the file storing voltage data is not important so long as it matches the name of the file Node-RED is told to extract data from.
d. Make sure to include a brief delay to avoid spamming the sensor. No need to measure more than once a second for our purposes.
e. You should be able to accomplish this code with 10 lines or less.

5. Access Node-RED by navigating to this address in your browser: scuttle-##:1880 or 192.168.1.1##:1880. Replace ## with your lab kit number, which is the two-digit value labeled on the side of the ethernet port.

Figure 6: Node-RED workspace

6. Once Node-RED loads, it should look something like figure 6. Create a flow that accesses data collected and logged by *L2_telemetry.py*, then display that data to a gauge.
   a. Your L2 script will need functions from *L1_log.py* to update text files with data acquired from *L1_ina.py*. Node-RED accesses and reads these files to display their contents to the dashboard.
   b. See the Display data using Node-RED resource on the first page for a video explaining how to build a flow. Your TA should also have an example or demonstration available for reference. If you get stuck, feel free to ask for help.
   c. Your gauge should look something like the example in figure 7 below. There are many options for customizing the gauge widget, but you should always include a proper title, units, and other labels to identify your information.

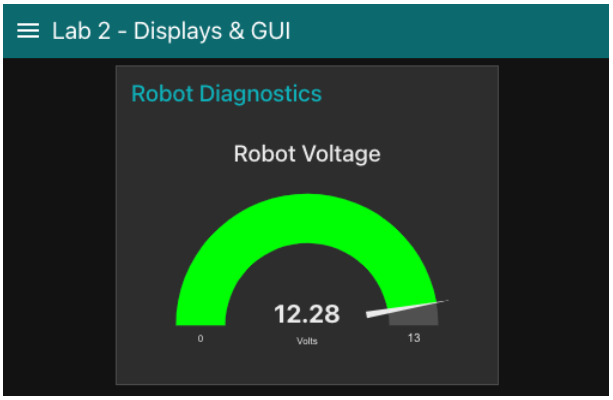

Figure 7: Example Node-RED gauge on dashboard

NOTE: Always include proper units and labels for the Node-RED GUI. Most screenshots for reports will be generated by Node-RED.

7. Once finished, ask your TA to check your work.
8. Once checked off, create a new directory **inside your parent mxet300_lab directory** called "*Node-RED_Flows*". This should look similar to figure 8 below.

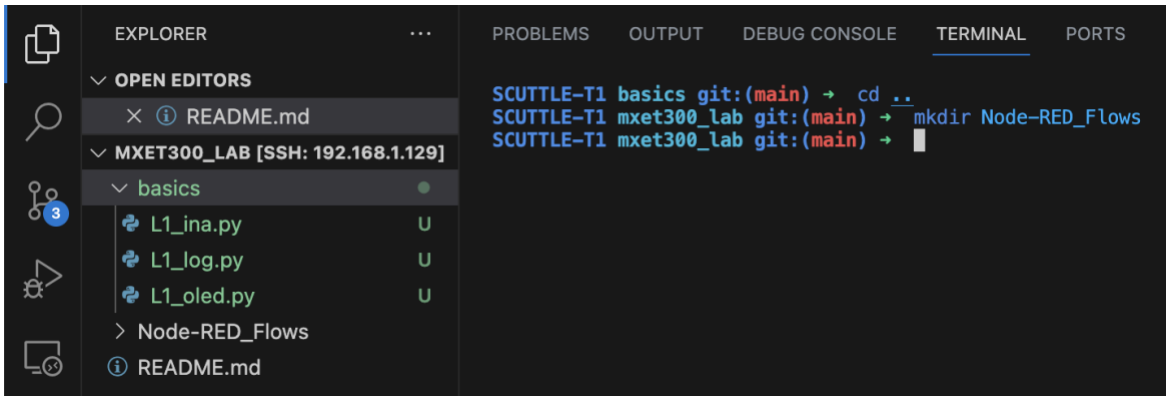

Figure 8: Node-RED_Flows

9. This is where you will export your Node-RED flows after each lab in order to submit and back them up. Since the files are all under the mxet300_lab directory, they too will be included in your GitHub repository when you push to it.
    a. To export a flow from Node-RED, follow the instructions in the resource linked on the first page titled How to Export and Import Node-RED Flows.

NOTE: Be sure to always export **JUST** the "**CURRENT FLOW"** for each newly created lab flow. This will create a .json file just for the current lab's Node-RED flow. By default, Node-RED may have the "all flows" option selected, and will create a .json file for all your flows. This can be nice as well, though having a separate file of each flow specifically can provide specific record keeping and utility.

Also, prior to exporting, you can click on the "JSON" tab and choose between the option to export as a compact or formatted JSON. The compact option generates a single line of JSON

with no whitespace. The formatted JSON option is formatted over multiple lines with full indentation - which can be easier to read.

   b. The export will be stored on your local machine, likely in your *Downloads* folder, not the Pi. Use VS Code to place the exported flow into the Node-RED_Flows directory you created previously.
   c. This is done by drag and drop into the directory directly through the Explorer pane, just be sure you're dropping in the intended directory.

10. The flow file will then be uploaded into the directory.
11. Label the exported flow so that it can be easily identified as your Lab2 flow: "Lab2_Flow.json".

Commit and Push to GitHub

Finally, you will need to commit and push your work to GitHub. This must be done at the end of every lab for grading and to ensure you have a complete backup available if needed.

1. Use the `git status` command to see the current state of your local repository.
2. **Ensure you are in the mxet300_lab directory folder.** Use the command `git add .` to track the changes made to files in the directory and add them to the staging area.
   a. Make sure you include a period after `git add`, which indicates that the command applies to all contents in the directory.
   b. Use `git status` to see the changes.
3. Commit the changes using the following command:
   a. `git commit -m "YOUR COMMIT NOTE HERE"`
   b. Again, use `git status` to see the changes.
4. Finally, use `git push` to push all commits to your remote repository on GitHub.

   NOTE: You don't need to use `git status` after each command, here it is only demonstrating what each command does when committing and pushing.

Post-Lab Questions

1. Describe the outputs of the *L1_ina.py* script when executed directly.
   a) What are the outputs and their units?
   b) Where does the script obtain these outputs and how?

2. Show a working Node-RED gauge for the SCUTTLE battery voltage, including proper units and a title.

3. Ensure your updated work is pushed to your team's repository on GitHub and ensure your python and Node-RED files are included.