MXET 300 - 501

Team: 07

Lab 2 - Displays and GUI

Members: Rex Worley & Kyle Rex

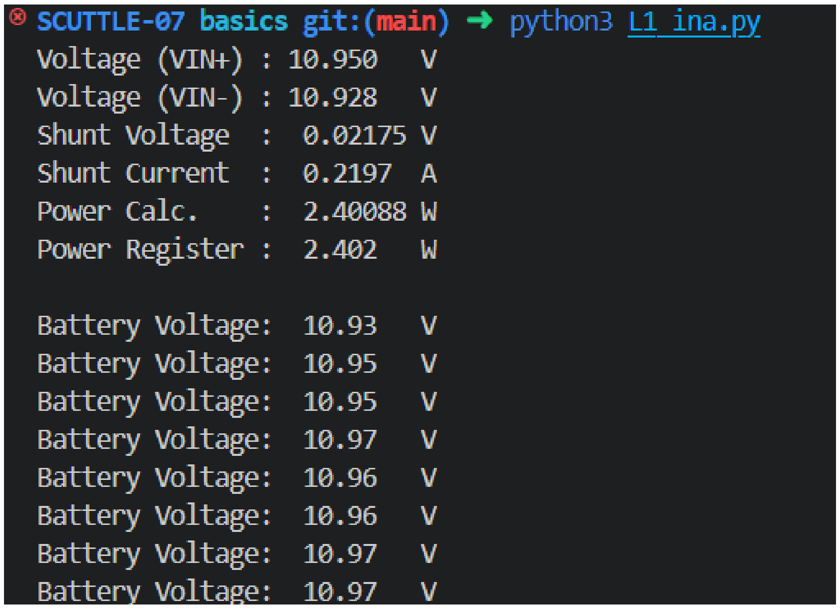
Date: 02/05/2024

Introduction

This lab focuses on integrating and modifying display and GUI elements in a SCUTTLE robot system. The main tasks include running a voltage sensor script, editing the OLED display script, and configuring Node-RED flows into an externally accessible GUI that displays the current battery voltage level. The OLED display provides an updated battery voltage reading at consistent time intervals. The Visual Studio Code IDE was utilized to modify the existing Python scripts. By modifying Python scripts and utilizing Node-RED, hands-on experience is gained in understanding program hierarchy, data visualization, and remote monitoring of system performance. The INA219 sensor is used to measure voltage, current, and power, providing valuable real-time data to the Raspberry Pi. Additionally, this lab reinforces skills in using the command line interface, GitHub version control, and Python programming for hardware interaction. Understanding how to extract and visualize sensor data is a key component of embedded systems development, and this lab provides practical experience in these areas. Finally, commits and pushes to GitHub are introduced and practiced.

Procedures and Lab Results

The Raspberry Pi uses a battery pack and custom "Y" power cable as a power source. The longer power pole was connected to the battery pack, while the shorter end was attached to the DC converter. The ferrule ends were secured to the INA219 terminal block, enabling voltage readings. To prevent damage and ensure proper connections, the ferrule ends were carefully inserted, and the screws were tightened just enough to hold them in place. Over-tightening can strip the terminal block screws, making future connections difficult. The Raspberry Pi can be accessed through Visual Code Studio's terminal over a wireless network. The required Python scripts, "L1_ina.py," "L1_oled.py," and "L1_log.py," can be accessed through the provided GitHub repository link in the MXET300-SCUTTLE/software/basics directory. After ensuring the basics folder is the current directory, the "wget" command followed by the raw script URL downloads the file. This process is repeated for the remaining files. The L1_ina.py file is run using the "python3 L1_ina.py" command and produces voltage, current, and power readings. The terminal output of the previous command is displayed in Figure 1.



```
⊗ SCUTTLE-07 basics git:(main) ➜ python3 L1_ina.py
  Voltage (VIN+) : 10.950   V
  Voltage (VIN-) : 10.928   V
  Shunt Voltage  :  0.02175 V
  Shunt Current  :  0.2197  A
  Power Calc.    :  2.40088 W
  Power Register :  2.402   W

  Battery Voltage:  10.93   V
  Battery Voltage:  10.95   V
  Battery Voltage:  10.95   V
  Battery Voltage:  10.97   V
  Battery Voltage:  10.96   V
  Battery Voltage:  10.96   V
  Battery Voltage:  10.97   V
  Battery Voltage:  10.97   V
```

Figure 1: Battery Readings in Terminal

The “L1_ina.py” script accesses the INA219 sensor via I2C communication and reads voltage and current values. These values are displayed in the terminal in real-time. The INA219 uses a shunt resistor to measure current, which allows it to calculate power based on Ohm's law.

To update the OLED display, the default service was first stopped using “sudo systemctl stop oled.service”. The “L1_oled.py” script was then modified to display the SCUTTLE’s unique robot name and real-time battery voltage with appropriate labels and units. After saving the changes, the script “python3 L1_oled.py” was executed to verify correct display functionality. Once verified, the modified OLED script was set as the default startup script using “sudo mv oled.py /usr/share/pyshared/oled.py”. The final OLED display can be seen in Figure 2.

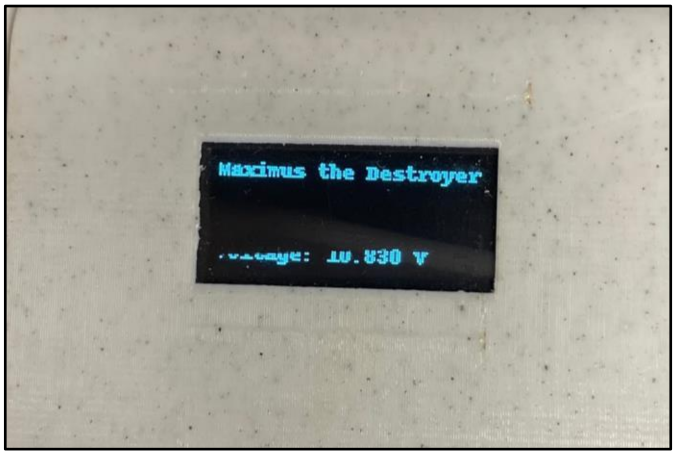


Figure 2: Modified OLED Display

The OLED display update allowed real-time monitoring of SCUTTLE’s battery voltage without needing to access the terminal. The script reads voltage values from the INA219 sensor and updates the display at regular intervals.

The next part of the lab involved using Node-RED GUI to create a real time voltage gauge display of the robot’s voltage readings. A new script to read voltage measurements was created under the name “L2_telemetry.py” by executing the command “touch L2_telemetry.py”. The newly created script reads voltage measurements from “L1_ina.py” and logs them with functions from “L1_log.py”. The script includes a delay to prevent excessive logging and ensures Node-RED can extract real-time values for the GUI. Using tmpFile ensures that logs do not clutter storage, as they are removed upon system restart. Node-RED was accessed via a browser using “scuttle-##:1880” (replacing ## with the lab kit number). A flow was created to read voltage data from the logged file and display it on a gauge in a user-friendly dashboard. An example of what this display should look like can be seen in Figure 3.

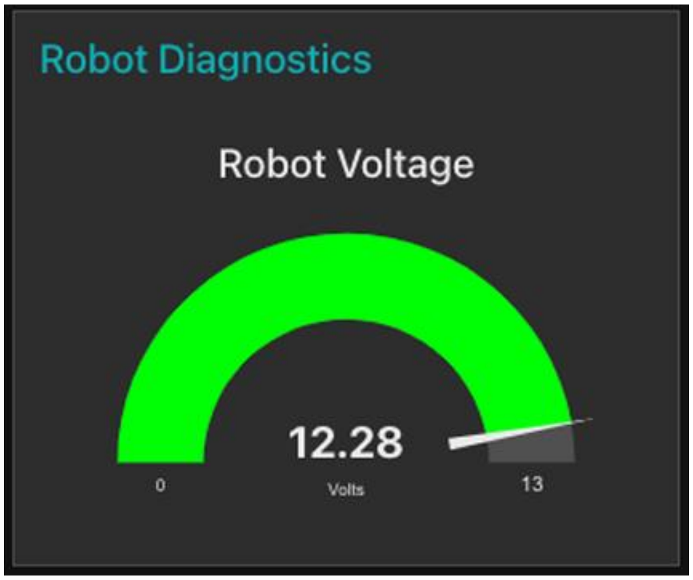


Figure 3: Example Node-RED gauge on dashboard from Lab Manual

The completed flow was exported and saved in the Node-RED_Flows directory using "Lab2_Flow.json".

Conclusion

This lab demonstrated real-time data acquisition and GUI development for monitoring SCUTTLE's battery voltage. By modifying Python scripts and utilizing Node-RED, battery voltage readings were successfully displayed on both the OLED screen and a web-accessible dashboard. The integration of the INA219 current sensor with the Raspberry Pi enabled accurate voltage measurements, which were then formatted and displayed in real-time.

The interaction between hardware and software components in an embedded system was a key focus of this lab. Python scripts such as L1_ina.py, L1_oled.py, and L1_log.py were used to read sensor data, process it, and display the results effectively. The modifications to L1_oled.py allowed for a customized OLED display, reinforcing the importance of program structure and script execution.

Additionally, implementing a Node-RED flow for real-time data visualization provided experience in GUI development and remote monitoring. Initially, configuring the Node-RED interface posed challenges, but after reviewing tutorials and practicing flow creation, the process became clearer. The ability to access SCUTTLE's battery voltage remotely enhances usability and provides a scalable method for monitoring system parameters.

Post-Lab Questions

1. Describe the outputs of the L1_ina.py script when executed directly.

<u>a) What are the outputs and their units?</u>

The script outputs input voltage at both input terminals in volts, shunt voltage in volts, shunt current in Amperes, calculated power in watts, and power register reading in watts. The current battery voltage reading was also displayed at timed intervals.

<u>b) Where does the script obtain these outputs and how?</u>

The script imports libraries to interact with the INA219 sensor, which measures voltage across a fixed-resistance shunt to determine current and power. The sensor communicates with the Raspberry Pi via I2C, and calculations use Ohm's law to derive additional values.

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

Node-RED gauge was checked off in lab but not captured in a screenshot. That is why in the report itself an example from the lab manual was included.

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

The updated scripts and Node-RED flow were committed and pushed to GitHub for grading and future reference.

Appendix

Lab 2 Commands Used:

- cd
- mkdir basics
- pwd
- wget
- python3 L1_ina.py
- sudo systemctl stop oled.service
- sudo mv oled.py/usr/share/pyshared/oled.py
- git status
- git add .
- git commit -m
- git push

GitHub Submission

All required files were pushed to the team's repository for review and backup.
https://github.com/Korex26/mxet300_lab