MXET 300
Mechatronics I

Multidisciplinary
Engineering Technology
COLLEGE OF ENGINEERING

LABORATORY # 3

Compass Calibration

Submission Date: 02/17/2025

Name: Kyle Rex and Rex Worley
Section: 501
UIN: 932008894 and 432006442

Introduction

The objective of this lab was to calibrate the magnetometer on the SCUTTLE robotic platform to obtain accurate compass readings and improve navigation capabilities. The magnetometer, an essential sensor for determining directional headings, often produces raw data affected by external magnetic interference and sensor inaccuracies. To address this, the lab involved a step-by-step calibration process, including data collection, transformation, and validation. The main tasks included mounting the Raspberry Pi and battery pack to the SCUTTLE chassis, executing Python scripts to collect and calibrate magnetometer readings, and performing rotational movements to ensure proper calibration. After calibration, the compass heading was continuously displayed using a Node-RED dashboard, providing both numerical values and cardinal directions. This lab reinforced key concepts related to sensor calibration, data processing, and real-time visualization, which are crucial for ensuring accurate and reliable navigation in mechatronic systems.

Procedures and Lab Results

This lab begins by setting up the physical mounting connections between the Raspberry Pi and Battery pack to the SCUTTLE chassis DIN rail following the provided configuration in the manual along with the connections between the Raspberry Pi and Battery Pack themselves which is done following the same setup procedure from Lab 2. Once this is complete the procedure used in Lab 1 to power and connect to the Raspberry Pi can be completed. This will allow the Raspberry Pi to be accessed through Visual Code Studio's terminal over a wireless network. In Visual Studio code the user will then use the "cd" command to go into the basics directory. The required Python scripts for this lab include: "Magnetometer_Compass_Calibration.ipynb", "L2_compass_heading.py", and "L1_log.py". "L1_log.py" was imported in the previous lab but "Magnetometer_Compass_Calibration.ipynb" and "L2_compass_heading.py" need to be imported using the "wget" command. Both those files can be accessed through the provided GitHub repository link in the MXET300-SCUTTLE/software/basics directory. Once this is complete the next step is to go to a web browser and open the jupyter notebook file by typing 192.168.1.107:8888 into the search bar and navigating to the "Magnetometer_Compass_Calibration.ipynb" notebook. Run the cells in this notebook one at a time to calibrate the magnetometer while the SCUTTLE is in an open area with no electronics in its general vicinity. Running the "Reading the Raw Magnetometer Values" section will show the raw magnetometer readings over time which can be seen in Figure 1.

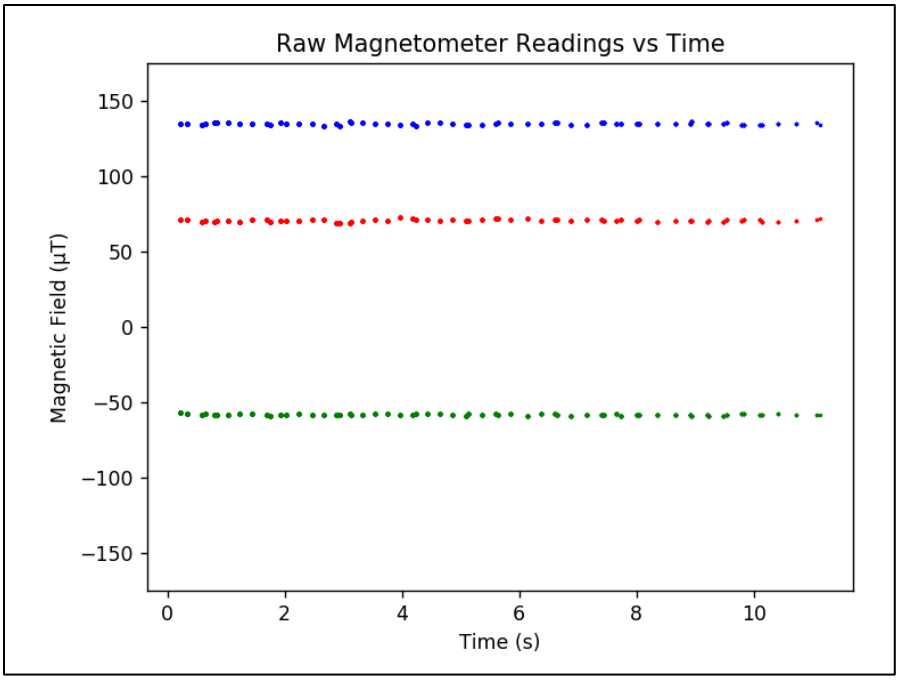


Figure 1: Raw Magnetometer Readings

With the raw magnetometer readings being outputted the next step is to calibrate the magnetometer itself which can be done by first running the first cell in the "Using Magnetometer Reading To Get A Compass Heading" section. While it is running the SCUTTLE must be rotated 10 times clockwise and 10 times counterclockwise about all three of its axes (x,y,z) which can be done by placing it in a swivel chair and then changing its orientation for each axis. The resulting plot should look similar to what can be seen in Figure 2.

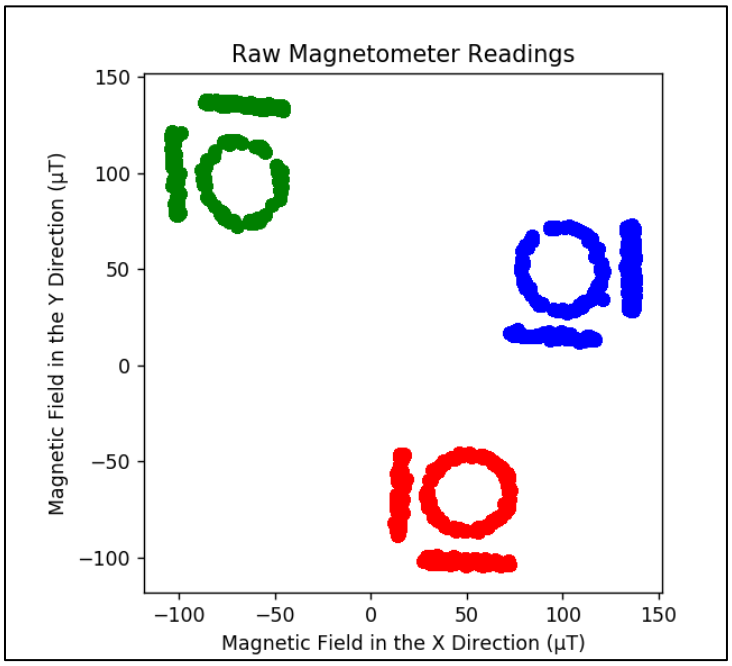


Figure 2: Raw Magnetometer Readings in the X and Y Magnetic Field

With the final raw magnetometer values recorded they must be converted into calibrated magnetometer values. This is done using the next cell of code in the "Using Magnetometer Reading To Get A Compass Heading" section which converts each X and Y value at each point using the following equations:

$$X_{calibrated} = \frac{2*(X_{raw} - X_{min})}{(X_{max} - X_{min})} - 1 \qquad \textit{Equation 1}$$

$$Y_{calibrated} = \frac{2*(Y_{raw} - Y_{min})}{(Y_{max} - Y_{min})} - 1 \qquad \textit{Equation 2}$$

This will result in the plot that can be seen in Figure 3 which displays the calibrated magnetometer readings.

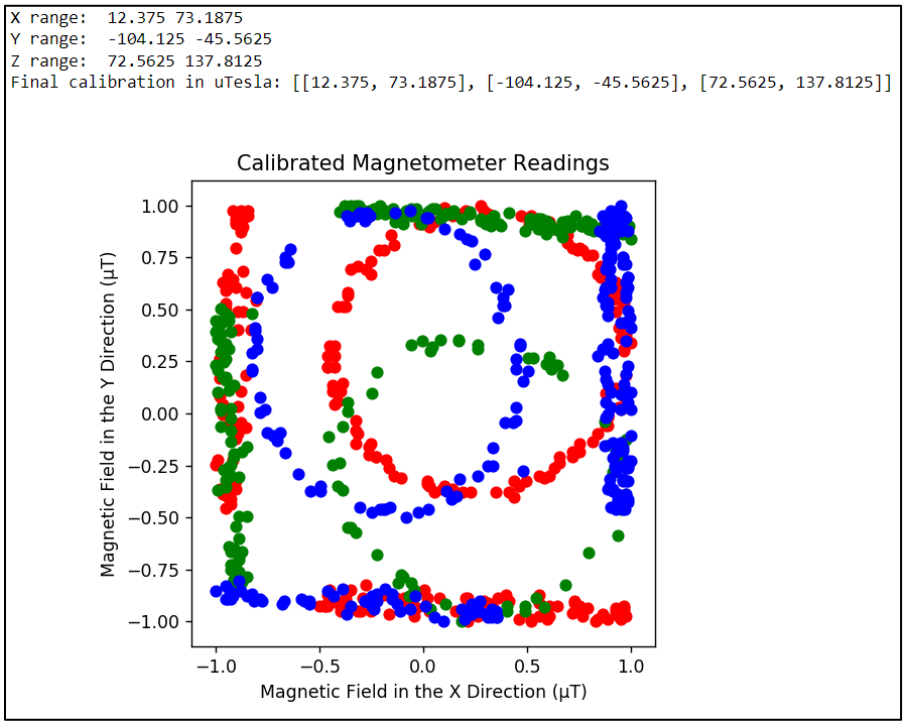


Figure 3: Calibrated Magnetometer Readings in the X and Y Magnetic Field

In order to validate the calibration the next step is to run the next cell of code in the "Validate Calibrated Magnetometer Readings" section and then rotate the SCUTTLE 10 times clockwise and 10 times counterclockwise about all three of its axes (x,y,z). Doing this will result in another plot that should be similar to the previous plot that can be seen in Figure 4.

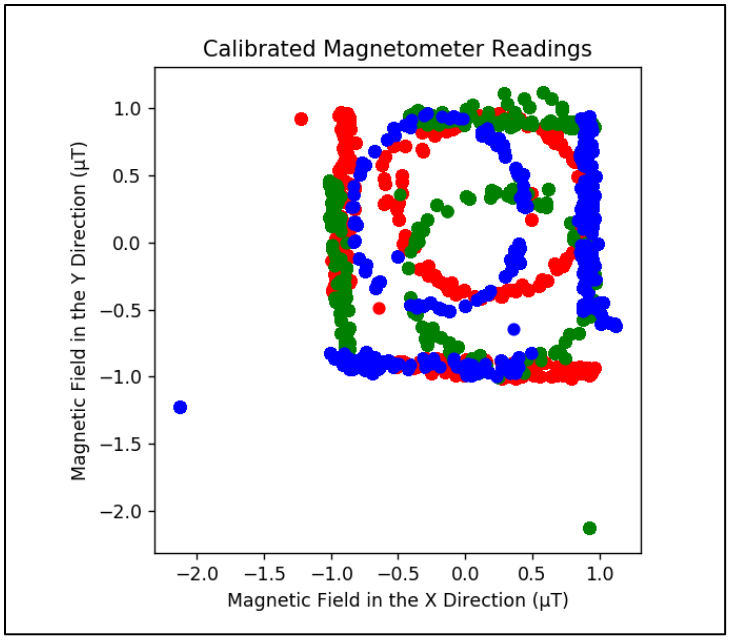


Figure 4: Validated Calibrated Magnetometer Readings in the X and Y Magnetic Field

If the result of this plot is similar to the previous plot, which it is, the validation is completed, and the magnetometer is calibrated. With the validation complete the next step is to actually calculate the compass heading of the SCUTTLE. This is done by running the "Calculate A Compass Heading" section and rotating the SCUTTLE about the z-axis. The resulting plot will display the compass heading changing. The result of this plot can be seen in Figure 5.

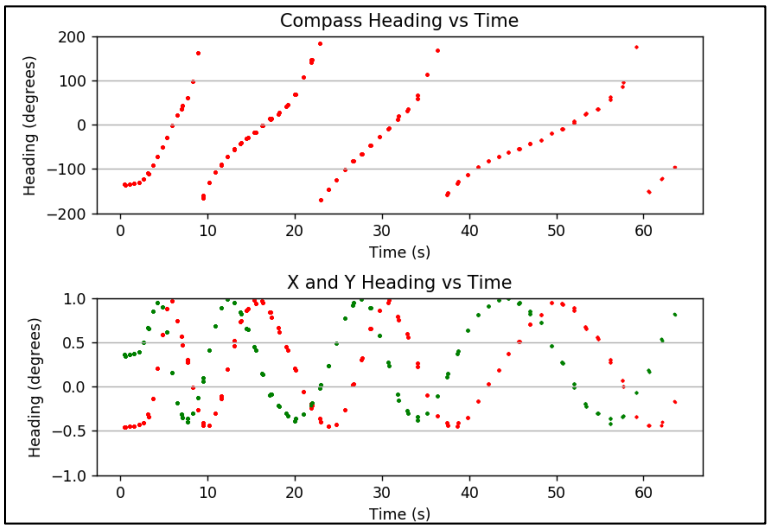


Figure 5: X and Y Magnetometer Readings for Different Compass Headings

With the calibration complete the data collected from it can be used in the compass heading python file to continuously print out the compass heading in degrees. This can then be used to create a Node-RED dashboard display L3 code displaying the compass and its heading. To do this the L2_compass_heading.py script must first be downloaded from the MXET300-SCUTTLE GitHub repository using wget and opened

in VS Code for review. The calibrated magnetometer values are then inserted into the script. After saving the changes, the script is executed using python3, and the compass heading is continuously output in degrees while rotating SCUTTLE on a swivel chair. Next, a new Python script, L3_Compass.py, is created. This script includes two functions: one retrieves the compass heading from L2_compass_heading.py, and the other converts the heading degrees (-180 to 180) into eight cardinal directions (North, Northwest, West, Southwest, South, Southeast, East, and Northeast). The assigned degree values for each direction can be seen in Figure 6.

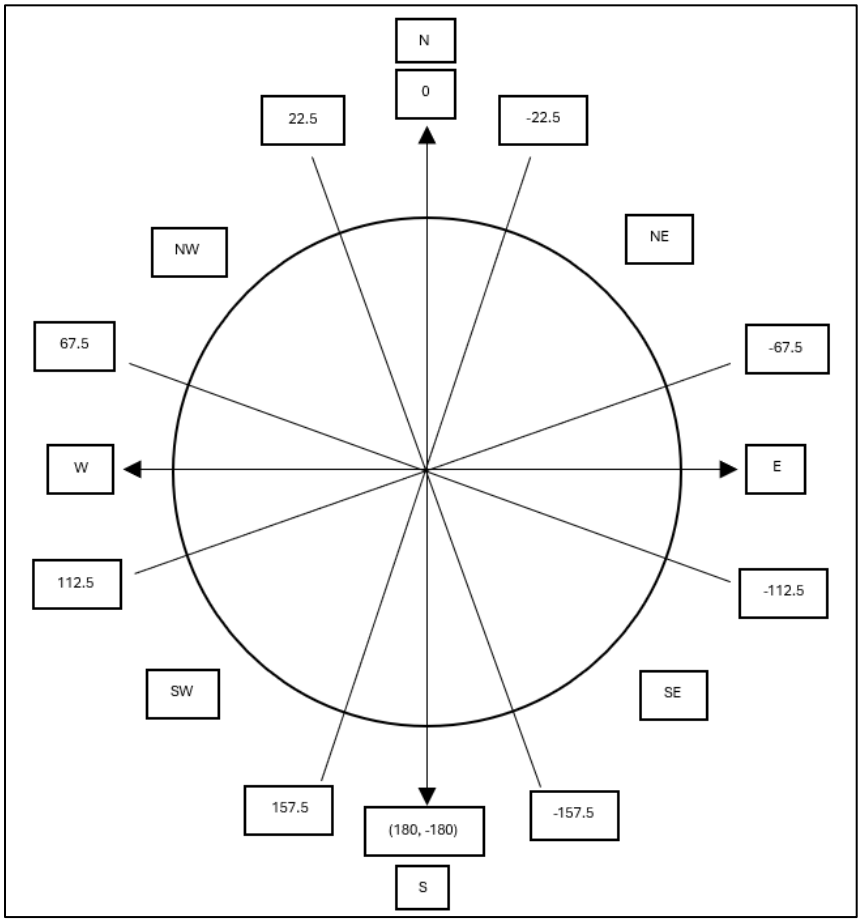


Figure 6: Directions with Corresponding Degrees

The compass heading is logged using the tmpFile function, while the cardinal direction is logged using the stringTmpFile function from L1_log, both running in a continuous loop. Finally, a Node-RED dashboard is developed to display the compass heading and cardinal directions. The heading is shown using a compass gauge, and the cardinal direction is displayed in a text node. The gauge is properly labeled with the correct unit and title, ensuring clear visualization of the data. The final gauge can be seen in Figure 7.

INSERT NODE-RED DASHBOARD

Figure 7: Node-RED Guage on Dashboard

Conclusion

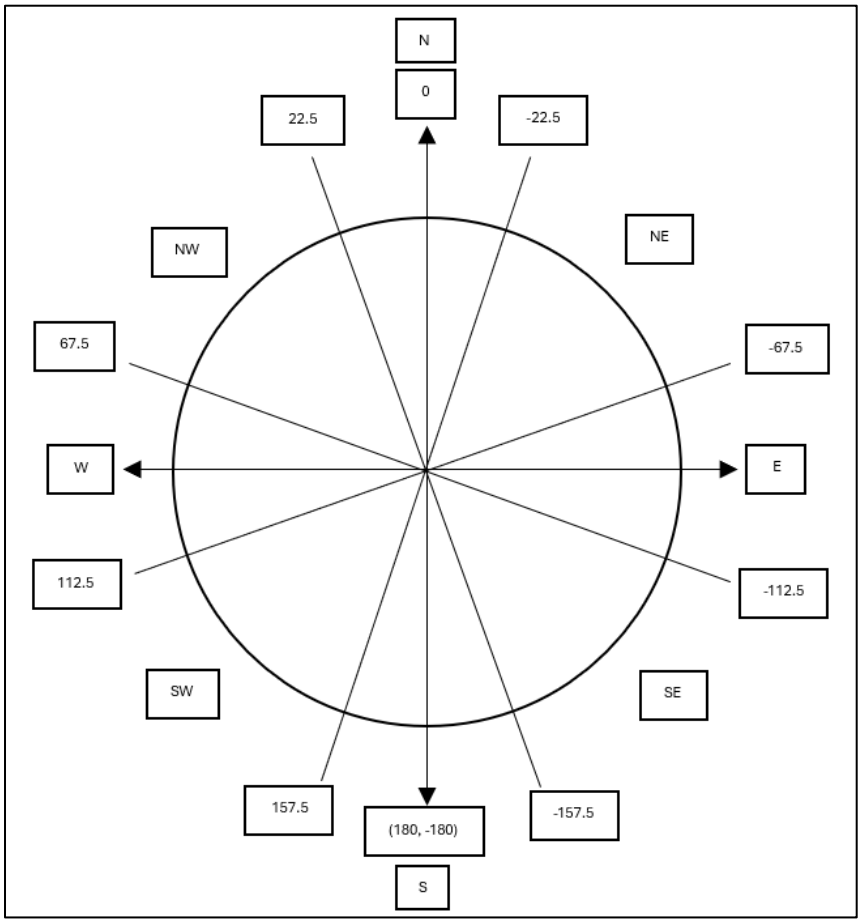
The lab successfully demonstrated the process of magnetometer calibration, resulting in improved accuracy for determining the SCUTTLE's heading. By following the calibration steps and validating the results through rotational testing, reliable directional data was obtained. The final implementation allowed for real-time monitoring through Node-RED, effectively visualizing the compass heading and corresponding cardinal directions. Throughout the lab, challenges were encountered, including potential interference from surrounding electronic components and inconsistencies in initial magnetometer readings. These issues were mitigated by conducting calibrations in an open area and carefully adjusting the collected data. This lab highlighted the importance of precise sensor calibration in robotics, emphasizing the need to account for external influences and refine raw data to enhance system performance.

Post-Lab Questions

1. Transform the heading number into cardinal directions (N, NE, E, SE, S, SW, W, and NW)

a. Create a function in your L3 code to decide in which cardinal direction the Raspberry Pi is heading. A function that takes heading as input and returns the heading as N, NE, E, SE, S, SW, W, and NW. (Hint: North is 0 degrees, and south is both 180 and -180 degrees. West and east should have positive and negative degrees respectively.)

Using if and elif conditions a code sequence was created to return a heading direction (N, NE, E, SE, S, SW, W, and NW) based on the heading number (some number between or equal to 180 and -180). The basic logic and degree selections can be understood in the Post Lab Figure 1 below.



Post Lab Figure 1: Directions with Corresponding Degrees

b. Modify your while loop to log the cardinal direction as a .txt file.

The while loop was modified to log the cardinal direction as a text file.

c. Display cardinal direction and heading value in Node-RED. Be sure to include proper labeling and units for your gauges.

The cardinal direction and heading value were displayed using Node-RED as seen in the Post Lab Figure 2 below.

INSERT NODE-RED DASHBOARD

Post Lab Figure 2: Node-RED Guage on Dashboard

d. Be sure to push all your updated work to your team's repo. This includes your updated python file(s) and Node-RED flows.

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

2. For what orientation (local/global) is a compass required?

A compass is required for global orientation because it provides absolute directional reference based on the Earth's magnetic field. It helps determine heading relative to true or magnetic north.

3. Before calibration, the raw data from the compass shows the following minimum and maximum values seen in Post Lab Table 1.

Post Lab Table 1: Raw Data Max and Min Values

Before Calibration	Min (microtesla)	max (microtesla)
x_c	-20	50
y_c	-32	25

If the raw data reading from the compass (before calibration) shows that xc = 15; yc = -10; what is resulting scaled xc and yc values? Please be sure to show the general equation and your work..

The following function from the Magnetometer_Compass_Calibration.ipynb can be used to calculate the scaled xc and yc values for this question:

```
def calibrate_magnetometer(original_value, axis_min, axis_max):
    temp = (original_value - axis_min) / (axis_max - axis_min)
    calibrated_value = 2*temp - 1
    return calibrated_value
```

This can be simplified into these equations:

$$X_{calibrated} = \frac{2 * (X_{raw} - X_{min})}{(X_{max} - X_{min})} - 1$$

$$Y_{calibrated} = \frac{2 * (Y_{raw} - Y_{min})}{(Y_{max} - Y_{min})} - 1$$

Using these equations xc (calibrated) = (2* ((15 - (-20)) / (50 - (-20)))) -1 = 0 (microtesla) and yc (calibrated) = (2* (((-10) - (-32)) / (25 - (-32)))) -1 = -0.228 (microtesla).

Appendix

Lab 3 Commands Used:

- cd basics/
- wget
- python3 L2_compass_heading.py
- python3 L3_Compass.py
- tmpFile
- stringTmpFile
- git add .
- git commit -m “Lab 3”
- git push

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