MXET 300
Mechatronics I

Multidisciplinary
Engineering Technology
COLLEGE OF ENGINEERING

LABORATORY # 4

Motor Drivers and Inverse Kinematics

Submission Date: 02/24/2025

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

Introduction

The purpose of this lab is to explore motor drivers and inverse kinematics in the context of mobile robot control. Using the SCUTTLE robotic platform, the lab focused on assembling and wiring essential components such as the Raspberry Pi, battery pack, and L298N H-bridge motor driver. Proper wiring and verification procedures were followed to ensure accurate signal transmission between the motor driver and the Raspberry Pi’s GPIO pins. Once the system was operational, motor functionality was validated before implementing inverse kinematics to translate chassis movement commands into specific wheel speeds. By developing a sequence of motion instructions, the SCUTTLE was programmed to follow an S-shaped path, demonstrating the practical application of inverse kinematics for mobile robot navigation. Additionally, joystick-based manual control and velocity data visualization were integrated using Node-RED to further analyze the robot’s movement characteristics.

Procedures and Lab Results

The lab begins with the assembly of the SCUTTLE robot, ensuring that all components, including the Raspberry Pi, battery pack, and motor driver bracket, are securely placed on the DIN rail, before wiring, following the same procedure for mounting these parts that is described in the Lab 3 manual. Placing the parts first, before wiring, prevents potential damage to connectors. Once all the parts are placed on the DIN rail the connections between the Raspberry Pi and Battery Pack can be completed following the same procedure for connecting these two parts described in the Lab 2 manual. For this lab some additional connections to the motors and L298N H-bridge are required on the SCUTTLE. The L298N H-bridge motor driver is connected with motor cables attached to the outer terminals and the battery pack connected to the center terminal. PWM signal cables are then routed from the H-bridge to the Raspberry Pi's GPIO header, using physical pins 11, 12, 15, and 16. To prevent wiring errors, connections must be verified before proceeding. These connection schemes can be seen in Figure 1 and the final setup for the SCUTTLE can be seen in Figure 2.

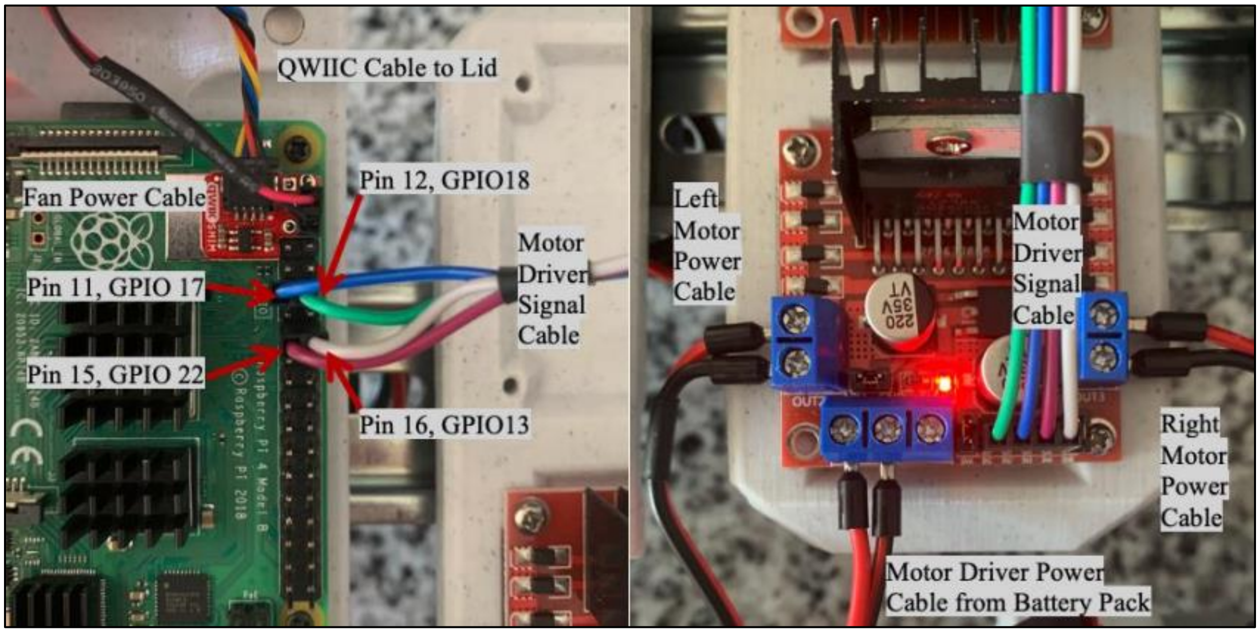


Figure 1: GPIO and H-Bridge Connections from Lab Manual

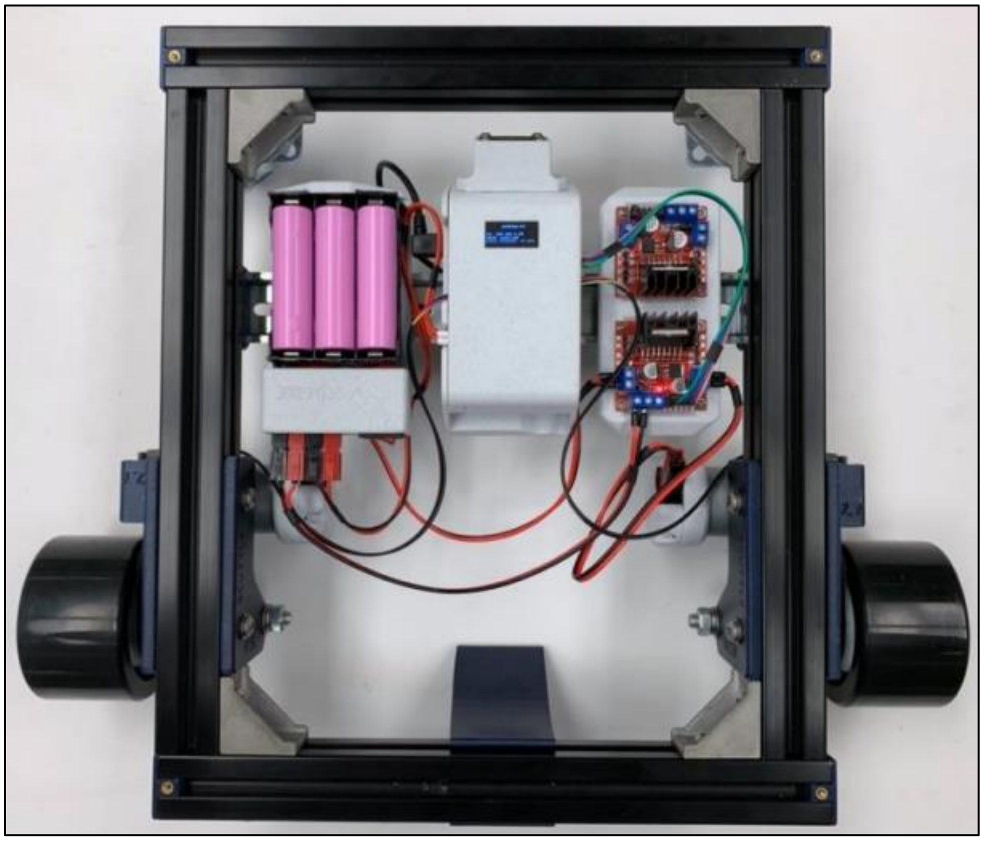


Figure 2: Complete SCUTTLE Physical Setup from Lab Manual

After checking the wiring and setup are correct the next step is to complete the booting and connecting procedure that are described in the Lab 1 manual. This will allow the Raspberry Pi to be accessed through Visual Code Studio's terminal over a wireless network. When the pi powers on connect to the network and open the VS code. In VS code use the "cd" command to go into the basics directory. Once in this directory the "wget" command can be used, following the procedure in the Lab 2 manual, in the MXET300-SCUTTLE/software/basics directory, to download and import L1_motor.py, L2_inverse_kinematics.py, L2_speed_control.py, and L3_noderedControl.py. Do this same procedure to get L3_path_template.py from the templates directory in in the MXET300- SCUTTLE GitHub that will be used later in the lab.

For this next part of the lab, to prevent unintended movement, the SCUTTLE must be placed on a stand with its back wheels lifted. The motor control of the SCUTTLE is tested by running "python3 L1_motor.py", which will result in the wheels spinning. The direction of wheel rotation is confirmed against terminal output, and any incorrect wiring is corrected accordingly. Since there were no issues with the terminal output and the wheel output being different the wiring was confirmed to be in the correct configuration.

With motor functionality established, inverse kinematics is introduced to translate chassis movement commands into specific wheel speeds. Without a device to measure wheel speeds, the SCUTTLE will not precisely follow the intended path because there is no feedback on whether the actuators reach their targets. This concept is known as open loop control since there is no feedback loop being referenced like there would be in a closed loop system. Running “python3 L2_inverse_kinematics.py” prompts the user for forward ($\dot{x}$) and angular ($\dot{\theta}$) velocities, which are converted into left and right wheel speeds (pdl and pdr). Using this knowledge, a sequence of movement commands must be developed to guide SCUTTLE along an "S" path to demonstrate the primary purpose of the lab which is to convert desired chassis speeds to wheel speeds. An example of what the path should look like starting at the green star and ending at the red star can be seen in Figure 3.

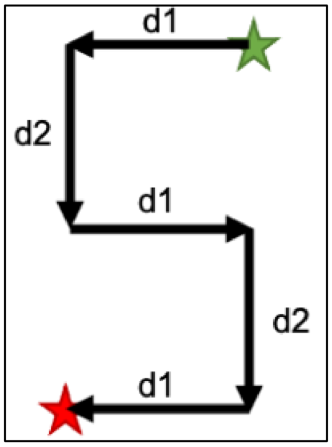


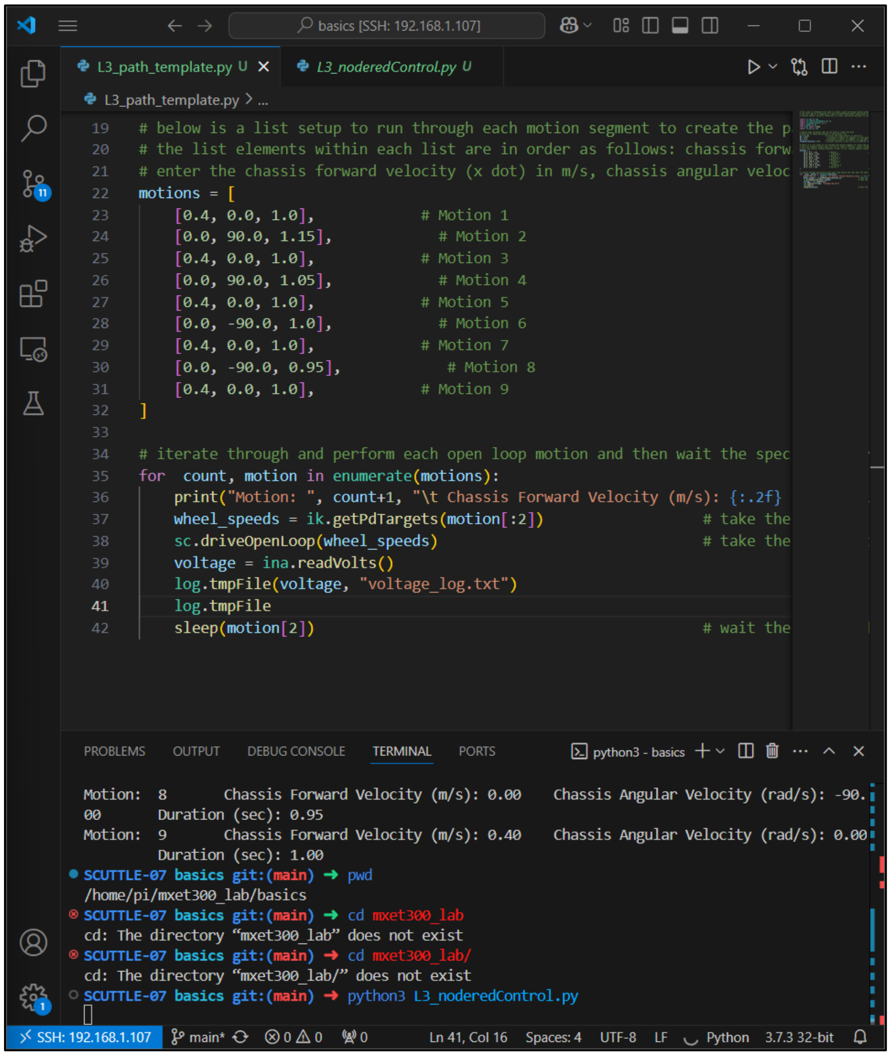
Figure 3: “S” Curve for SCUTTLE Path from Lab Manual

An S-shaped path is to be driven by the SCUTTLE robot through a series of motion instructions which include defined distance, forward velocity, angular velocity, and duration parameters for each motion instruction. The path has horizontal distances defined by d1 and vertical distances defined by d2 for which 0.8 meters and 0.6 meters were chosen respectively. The maximum forward velocity for the SCUTTLE robots is 0.4 meters per second. Table 1 displays the discrete values chosen to perform this movement operation for each motion instruction’s parameters.

Table 1: Selected Velocity Data for SCUTTLE

d1 = 0.8 m		d2 = 0.6 m	
Motion	**$\dot{x}$ (m/s)**	**$\dot{\theta}$ (rad/s)**	**Duration (s)**
1	0.4	0	1
2	0	90	1.15
3	0.4	0	1
4	0	90	1.05
5	0.4	0	1
6	0	-90	1
7	0.4	0	1
8	0	-90	0.95
9	0.4	0	1
10	0	0	0

The forward velocity for each translational movement was chosen to be the maximum 0.4 meters per second for time efficiency. The decreased duration of each turn on the even-numbered motions 2, 4, 6, and 8 produced the best experimental results from numerous trials. The robot did not follow the path exactly due to the lack of a closed-loop system and only traversed the rough path templated in the code. The angular velocity was coded as if it were in degrees per second and the SCUTTLE still moved through the path accurately despite the difference in units. This movement was completed using the "python3 L3_path_template.py" script that had to be altered to match the previous table values. The required alterations to this code and the subsequent terminal output can be seen in Figure 4 and Figure 5 respectively.

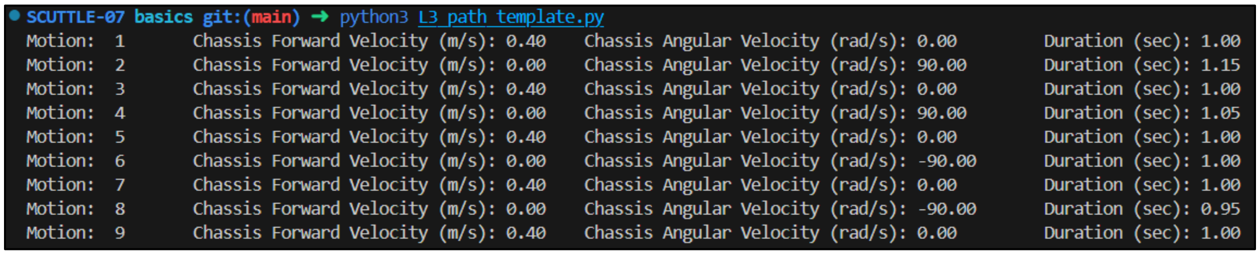


```python
# below is a list setup to run through each motion segment to create the p
# the list elements within each list are in order as follows: chassis forw
# enter the chassis forward velocity (x dot) in m/s, chassis angular veloc
motions = [
    [0.4, 0.0, 1.0],                # Motion 1
    [0.0, 90.0, 1.15],                # Motion 2
    [0.4, 0.0, 1.0],                # Motion 3
    [0.0, 90.0, 1.05],                # Motion 4
    [0.4, 0.0, 1.0],                # Motion 5
    [0.0, -90.0, 1.0],                # Motion 6
    [0.4, 0.0, 1.0],                # Motion 7
    [0.0, -90.0, 0.95],                # Motion 8
    [0.4, 0.0, 1.0],                # Motion 9
]

# iterate through and perform each open loop motion and then wait the spec
for  count, motion in enumerate(motions):
    print("Motion: ", count+1, "\t Chassis Forward Velocity (m/s): {:.2f}
    wheel_speeds = ik.getPdTargets(motion[:2])                # take the
    sc.driveOpenLoop(wheel_speeds)                            # take the
    voltage = ina.readVolts()
    log.tmpFile(voltage, "voltage_log.txt")
    log.tmpFile
    sleep(motion[2])                                          # wait the
```

```
Motion:  8       Chassis Forward Velocity (m/s): 0.00    Chassis Angular Velocity (rad/s): -90.
00       Duration (sec): 0.95
Motion:  9       Chassis Forward Velocity (m/s): 0.40    Chassis Angular Velocity (rad/s): 0.00
         Duration (sec): 1.00
● SCUTTLE-07 basics git:(main) ➜ pwd
/home/pi/mxet300_lab/basics
⊗ SCUTTLE-07 basics git:(main) ➜ cd mxet300_lab
cd: The directory "mxet300_lab" does not exist
⊗ SCUTTLE-07 basics git:(main) ➜ cd mxet300_lab/
cd: The directory "mxet300_lab/" does not exist
○ SCUTTLE-07 basics git:(main) ➜ python3 L3_noderedControl.py
```

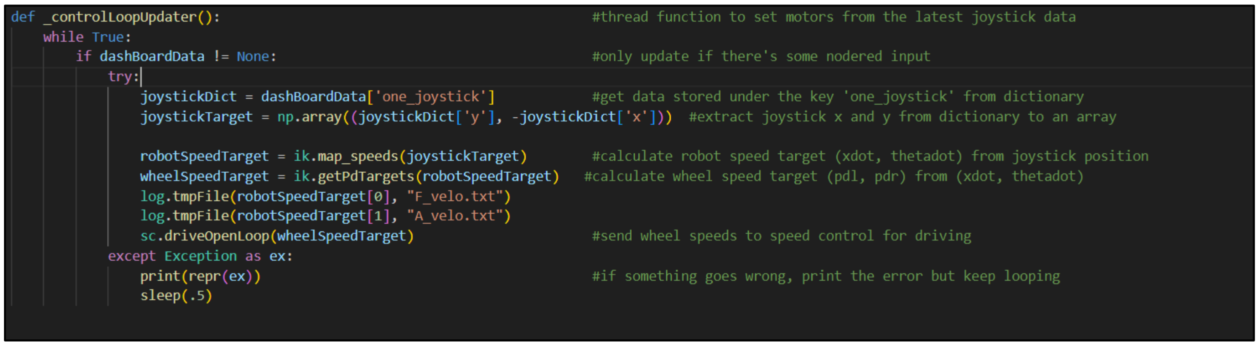
Figure 4: "S" Curve Motion Code Changes and Additions



```
● SCUTTLE-07 basics git:(main) ➜ python3 L3_path_template.py
Motion:  1       Chassis Forward Velocity (m/s): 0.40    Chassis Angular Velocity (rad/s): 0.00       Duration (sec): 1.00
Motion:  2       Chassis Forward Velocity (m/s): 0.00    Chassis Angular Velocity (rad/s): 90.00      Duration (sec): 1.15
Motion:  3       Chassis Forward Velocity (m/s): 0.40    Chassis Angular Velocity (rad/s): 0.00       Duration (sec): 1.00
Motion:  4       Chassis Forward Velocity (m/s): 0.00    Chassis Angular Velocity (rad/s): 90.00      Duration (sec): 1.05
Motion:  5       Chassis Forward Velocity (m/s): 0.40    Chassis Angular Velocity (rad/s): 0.00       Duration (sec): 1.00
Motion:  6       Chassis Forward Velocity (m/s): 0.00    Chassis Angular Velocity (rad/s): -90.00     Duration (sec): 1.00
Motion:  7       Chassis Forward Velocity (m/s): 0.40    Chassis Angular Velocity (rad/s): 0.00       Duration (sec): 1.00
Motion:  8       Chassis Forward Velocity (m/s): 0.00    Chassis Angular Velocity (rad/s): -90.00     Duration (sec): 0.95
Motion:  9       Chassis Forward Velocity (m/s): 0.40    Chassis Angular Velocity (rad/s): 0.00       Duration (sec): 1.00
```

Figure 5: Terminal Output for "S" Curve Motion Code

Following successful execution of the scripted path, the SCUTTLE can now be manually controlled using a Node-RED joystick. This is done using the "L3_noderedControl.py" script that receives joystick inputs from Node-RED to drive the motors. This code can be seen in Figure 6.



```python
def _controlLoopUpdater():                                                   #thread function to set motors from the latest joystick data
    while True:
        if dashBoardData != None:                                            #only update if there's some nodered input
            try:
                joystickDict = dashBoardData['one_joystick']                 #get data stored under the key 'one_joystick' from dictionary
                joystickTarget = np.array((joystickDict['y'], -joystickDict['x']))  #extract joystick x and y from dictionary to an array

                robotSpeedTarget = ik.map_speeds(joystickTarget)             #calculate robot speed target (xdot, thetadot) from joystick position
                wheelSpeedTarget = ik.getPdTargets(robotSpeedTarget)        #calculate wheel speed target (pdl, pdr) from (xdot, thetadot)
                log.tmpFile(robotSpeedTarget[0], "F_velo.txt")
                log.tmpFile(robotSpeedTarget[1], "A_velo.txt")
                sc.driveOpenLoop(wheelSpeedTarget)                           #send wheel speeds to speed control for driving
            except Exception as ex:
                print(repr(ex))                                              #if something goes wrong, print the error but keep looping
                sleep(.5)
```

Figure 6: Code for Joystick Control

The next step is to navigate into the "Node-RED_Flows" folder that was created in Lab 2. Once in this folder the "wget" command can be used, following the procedure in the Lab 2 manual, in the MXET300-SCUTTLE nodered directory, to download and import a "flow_joystickControl.json" file. The L3_noderedControl.py script is run to receive joystick inputs from Node-RED, which are sent to the Raspberry Pi through the "flow_joystickControl.json" file. This allows real-time joystick-based movement of SCUTTLE. Running the "python3 L3_noderedControl.py" script with this new file imported and configured properly in a new Node-RED flow will result in a fully functioning GUI joystick control for the SCUTTLE on the Node-RED dashboard that can be seen in Figure 7.

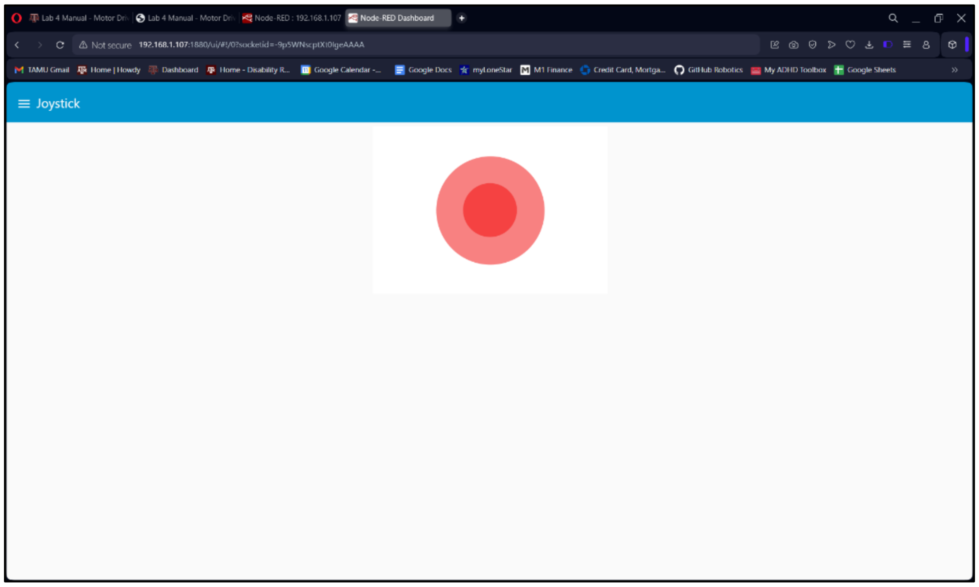

Node-RED Dashboard
Not secure 192.168.1.107:1880/ui/#!/0?socketid=-9p5WNscptXt0geAAAA
Joystick

Figure 7: Joystick Control Node-RED Dashboard

With a joystick control that moves the SCUTTLE in any direction at varying speeds the next step involves plotting its turning (angular) and forward velocities using Node-Red plots for a better understanding of its movement. The Node-Red screen required was created by obtaining flow_joystickControl.json and running the L3_noderedControl.py file. The json file is imported to control the SCUTTLE robot's movement through a digital joystick GUI. The joystick values are sent to the Raspberry Pi and used to control the SCUTTLE's motion directly. The voltage measurement flow from a previous laboratory experiment was utilized to display the recorded voltage in the Node-Red dashboard. Chart blocks were chosen and formatted to display the forward and angular velocity data from their respective temporary files "F_velo.txt" and "A_velo.txt" in the Node-Red dashboard. The final Node-Red dashboard can be seen in Figure 8.

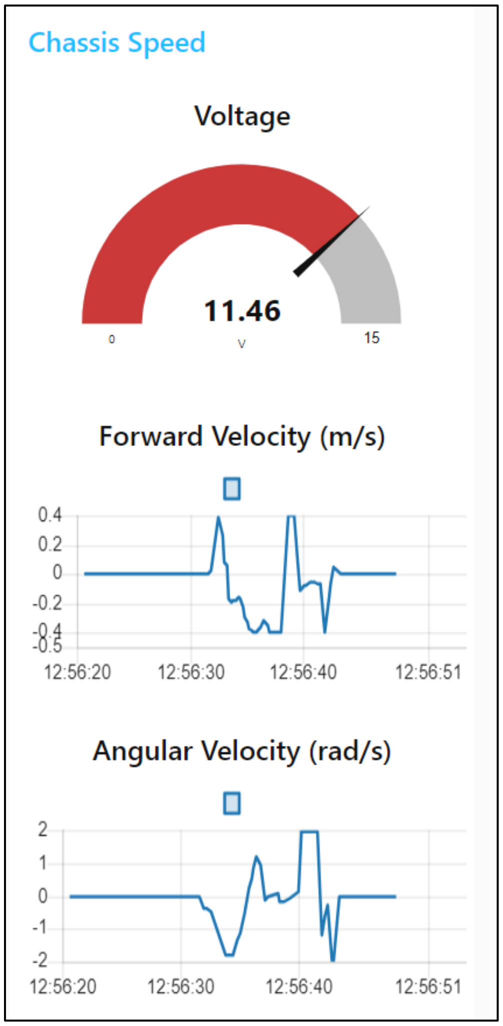


Figure 8: Node-RED Dashboard of Forward and Angular Velocities of the SCUTTLE

Conclusion

The lab successfully demonstrated the implementation of motor drivers and inverse kinematics to control the SCUTTLE robotic platform. The integration of open-loop control techniques enabled the conversion of desired chassis velocities into corresponding wheel speeds, allowing the SCUTTLE to approximate an S-shaped trajectory. However, due to the absence of a feedback system, minor deviations from the intended path were observed. The incorporation of joystick-based control and real-time velocity plotting using Node-RED provided additional insight into the robot's movement behavior. Through this lab, essential concepts such as motor control, motion planning, and data visualization were reinforced, laying a foundation for further advancements in mobile robot navigation and control systems.

Post-Lab Questions

1. Why won't the SCUTTLE, as configured in this lab, follow the planned path exactly?

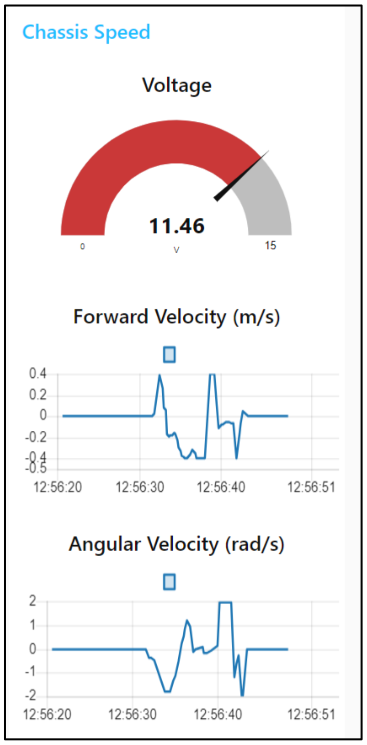
The SCUTTLE does not follow the planned path exactly due to variability in the predetermined lengths of d1 and d2. Another reason is the lack of consistency in the period allotted for each movement as each movement required a finely tuned time that required an inefficient amount of time to drastically increase precision. Also, it's important to keep in mind that without a device to measure the wheel speeds the SCUTTLE will not actually follow the exact path given due to there being no feedback control and it being an open-loop system, not closed, therefore causing open-loop control. In this system there is no telling whether or not the actuators actually reach their targets.

2. Go through your L3_path_template.py file script for completing the S path and briefly explain how the code works.

After importing the necessary files and libraries, the code holds the user's values for distance 1, distance 2, and the desired forward velocity. The forward velocity, angular velocity, and length of time are placed in an array for execution. The function "_controlLoopUpdater" checks if a NodeRed input exists and calculates the robot target speeds from the joystick position and wheel target speeds from the forward and angular velocities calculated right before. The robot's forward and angular velocities were logged in the temporary files "F_velo.txt" and "A_velo.txt" respectively. An exception condition was added if the function's logic was disrupted.

3. Modify the L3_path_template script to log $\dot{x}$ and $\dot{\theta}$. Then create a flow to display these velocities in a chart while the program runs. a) Attach a screenshot of the flow with data included. Make sure to configure the charts with titles, units, and appropriate bounds. The figure would look like the following. b) Optionally, you could log $\dot{x}$ and $\dot{\theta}$ from Node-RED Joystick. You can use "file in" nodes to read this information or optionally try using "UDP in" nodes. For the former, you can log values from L3_noderedControl script. For the latter you can refer to the joystick flow and L3 driving script for examples on using sockets to communicate via network interface.

The answer for this post lab question can be seen in Post Lab Figure 1.



Post Lab Figure 1: Forward and Angular Velocities of the SCUTTLE

Appendix

Lab 4 Commands Used:

- pwd
- python3 L3_path_template.py
- python3 L3_noderedControl.py
- git add .
- git commit -m “Lab 4”
- git push

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

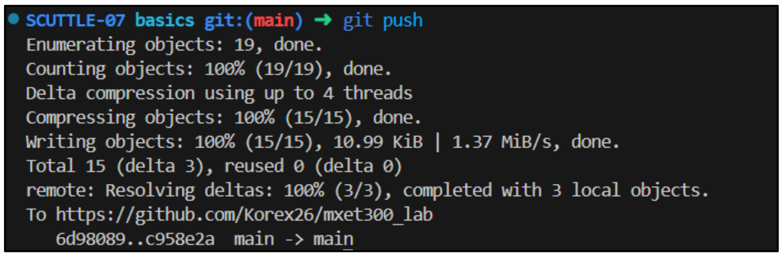
Lab 2 Commands Used:

- 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

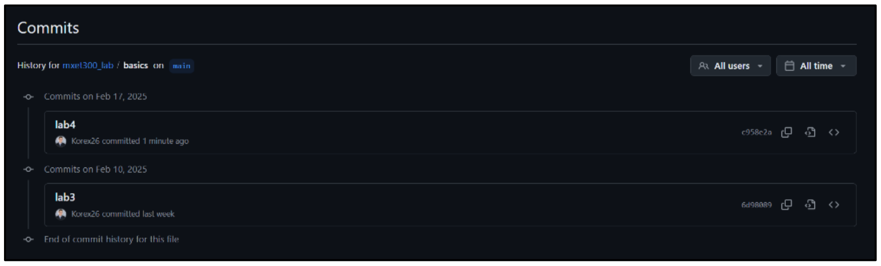
GitHub Submission

All required files were pushed to the team's repository for review and backup as seen in the GitHub Submission Figure 1 and GitHub Submission Figure 2. The repository can be found using the following link: https://github.com/Korex26/mxet300_lab.



```
● SCUTTLE-07 basics git:(main) ➜ git push
  Enumerating objects: 19, done.
  Counting objects: 100% (19/19), done.
  Delta compression using up to 4 threads
  Compressing objects: 100% (15/15), done.
  Writing objects: 100% (15/15), 10.99 KiB | 1.37 MiB/s, done.
  Total 15 (delta 3), reused 0 (delta 0)
  remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
  To https://github.com/Korex26/mxet300_lab
     6d98089..c958e2a  main -> main
```

GitHub Submission Figure 1: Commit and Push to GitHub on Visual Studio Code



Commits

History for mxet300_lab / basics on main

Commits on Feb 17, 2025

lab4
Korex26 committed 1 minute ago
c958e2a

Commits on Feb 10, 2025

lab3
Korex26 committed last week
6d98089

End of commit history for this file

GitHub Submission Figure 2: Commit and Push to GitHub on GitHub

References

K. Rex, R. Worley, *MXET 300 Lab Report 3*, 2025.

K. Rex, R. Worley, *MXET 300 Lab Report 2*, 2025.

X. Song, *Lab 4 Manual - Motor Drivers _ Inverse Kinematics*, 2025.

X. Song, *Lab 3 - Compass Calibration*, 2025.

X. Song, *Lab 2 Manual - Displays and GUI*, 2025.

X. Song, *Lab 1 Manual - Pi Setup*, 2025.