First Technical Update

Progress Update from SSG Robotics on Prototype Development for HND

Prepared by Group 10

Sebastien Sauter, Jamie Strain, and Marlow Gaddes

February 2, 2025

1.0 Technical Achievement

1.1 Design Decisions

SSG Robotics chose to prototype the airport delivery rover using a Lynxmotion Rover. This rover is an optimal prototype due to its lightweight, compact, and durable design. The design decisions section discusses pin locations, rover orientation, and direction conventions. Due to limited availability of digital and analog pins on the Arduino Uno WiFi Rev 2 pins have been chosen for input signals. For instance, it is necessary to use Pulse-Width-Modulation (PWM) pins to control motor speed. The development board has a total of five pins capable of this function. Hence, signals EA and EB are connected to PWM pins six and five respectively. Additionally, external interrupts are used to read the wheel encoder signals A and B. This does not present problems as the Uno WiFi Rev 2 has external interrupt registers on all pins. Therefore, any pins could be chosen. Lastly, the signals I1 - I4 drive the transistors in each H-bridge. These transistors only require a discrete digital voltage signal of one or zero hence, any pins can be chosen.

As seen in Figure **1**, the front of the rover is indicated by the 12 V battery. The right and left sides of the rover are determined by looking at the back of the rover.

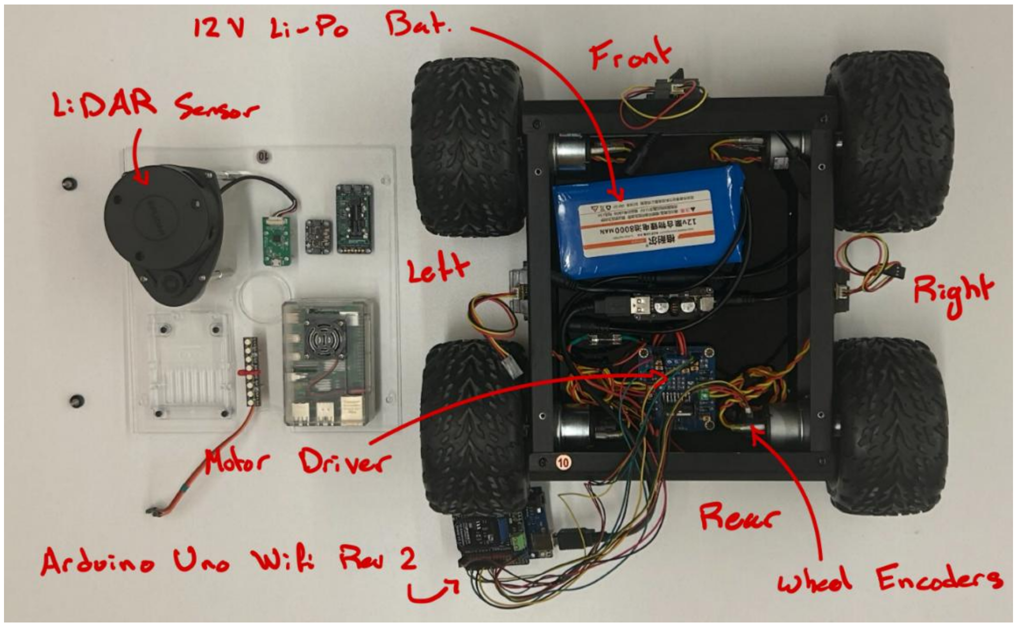

Figure 1: Lynxmotion Rover Overview with Key Components Highlighted

The design of the motors on this rover is such that the right motors are flipped with respect to the left motors. Hence, when the left and right wheels spin in the same direction the rover will pivot. To have a consistent convention of motion the forwards direction is positive and backwards is negative. The left wheels drive in the positive forwards direction as they rotate counterclockwise. Whereas the right wheels drive in the positive forwards directions as they rotate clockwise. Since positive wheel ticks are associated to counterclockwise rotation in software the right wheel speed in multiplied by the constant negative one.

1.2 Drive Motors and Wheel Encoders

The team has been utilizing wheel encoders to provide feedback to the microcontroller, allowing the rover to adjust power output and maintain the desired speed and direction. Tin Equation **1** is used to calculate the estimate for the rotational speed of the wheels, in (rad/s).

$$\widehat{\omega}_L = 2\pi * (\# \, of \, encoder \, ticks \, / \, TPR) * 1000 \, / \, T \qquad (1)$$

Where TPR the constant Ticks per Revolution was measured to be 3000. Taking the estimates for the rotational speed and multiplying it with the wheel radius gives an estimated translational velocity for the left (V_L) and right (V_R) wheels. The overall estimate of the translational velocity is found by averaging V_L and V_R. As well, by taking the difference in V_L and V_R, then dividing it by the rover track, gives the approximate turning rate.

1.3 Internal Measurement Unit (IMU)

The team has begun testing on the 6 degree of freedom LSM6DS3 Internal Measurement Unit (IMU). The IMU returns accelerometer data in x, y, and z directions as well as gyroscope data for all three corresponding rotational axes. The positive and negative conventions relative to the Arduino board are illustrated in Figure **2**.

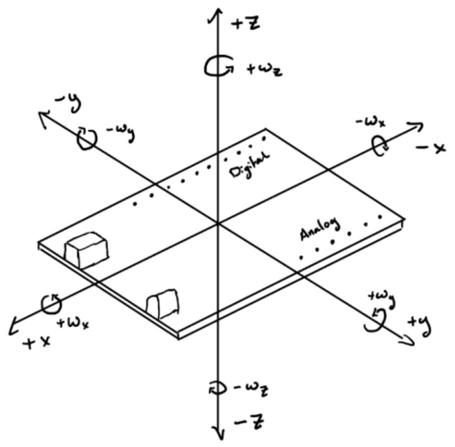

Figure 2: IMU Sign Conventions with Respect to Arduino Uno Wifi Rev 2

The accelerometer returns acceleration data in units of gravitational constants ($g = 9.81\ m/s^2$) and the gyroscope returns angular velocity data in units of degrees per second. When the IMU is stationary, the team identified the following systematic error in the data.

$$\underline{a} = [x \quad y \quad z] = [0.02 \quad 0.00 \quad 1.01]\ g$$

$$\underline{\omega} = [\omega_x \quad \omega_y \quad \omega_z] = [0.24 \quad 0.31 \quad -0.24]\ {}^\circ/s$$

The acceleration error in the z-axis makes sense because all objects on earth experience 1g of acceleration. To eliminate this error, subtract the average systematic error from the measurements.

The team identified that the IMU is well suited to measure the angular velocity. Since the Lynxmotion rover is unable to steer by rotating its wheels, anytime the rover needs to turn the wheels will slip with respect to the ground. The math behind the wheel encoders relies on a no slip condition to measure both straight line velocity and angular velocity. Therefore, whenever the rover needs to turn the wheel encoders cannot be trusted to provide accurate data.

The team conducted a test in which one wheel had a strong forward drive, and the other wheel had a medium reverse drive. The wheel encoders returned various unclear data averaging at 1.9 rad/s. The gyroscope on the other hand effectively returned a consistent 1 rad/s. To further test the gyroscope, we pulsed the angular velocity by slowly increasing the PWM signal for the forward and reverse driving wheels. The objective was to gauge the noise and consistency of the gyroscope. As shown in Figure **3**, the gyroscope returns consistent and relatively low noise data, therefore the team decided to use gyroscope data as the main feedback data whilst turning. The complete implementation with the PID controller is incomplete, refer to section **3.0** Future Objectives.

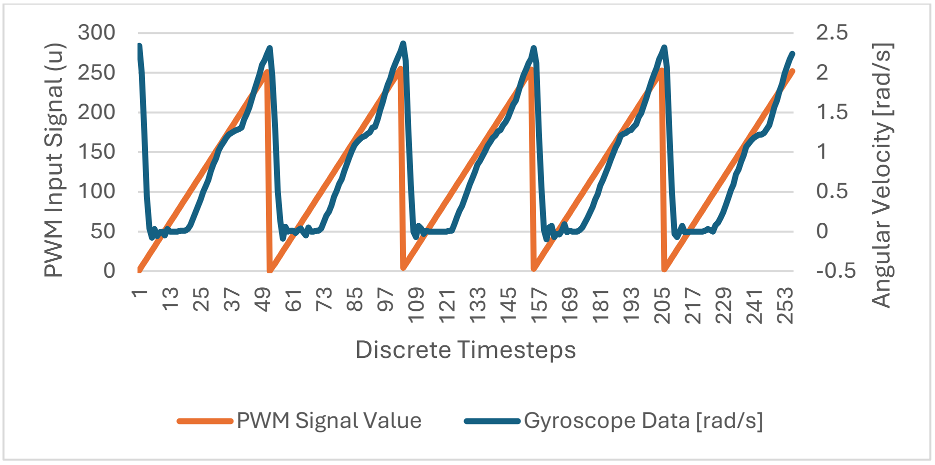

Figure 3: Gyroscope Data vs. PWM Input Signal

1.4 Proportional Integral Derivative (PID) Controller

The Lynxmotion Rover employs a Proportional-Integral-Derivative (PID) controller to drive at a precise and stable speed set by the user. A PID control system is a form of closed-loop control. Hence, a PID controller uses feedback to compare the output and the setpoint. The PID controller applies a correction based on the proportional, integral, and derivative errors. The proportional error is the difference between the setpoint and the measured speed. The integral is the sum of errors, and the derivative error is the rate of change in error over time.

To date, SSG Robotics has only implemented the proportional and integral parts of the controller, as a derivative term requires further tuning and testing. Hence, derivative control may be added in the future (refer to section **3.0** Future Objectives). Currently the team is satisfied with the results from the current PI controller.

The first step in building a PI controller is determining the desired wheel speeds. Equations **2** and **3** are used to calculate the desired wheel speed using the desired translational and angular rover speeds. The formulation of this equations makes a few assumptions. The first assumption is the track of the rover is significantly larger than the distance between the wheels on any one side. Another assumption made is both wheels on any one side must have the same speed.

$$V_{DR} = V_{DT} + \frac{l}{2}\omega_D \tag{2}$$

$$V_{DL} = V_{DT} - \frac{l}{2}\omega_D \tag{3}$$

The next step in building the PI controller is to calculate the proportional error. As stated earlier the proportional error is the difference in the measured and desired wheel speed multiplied by the proportional constant. SSG Robotics is in the process of determining the appropriate proportional constant, although further tuning and experimentation is required. The Arduino serial plotter is used to measure the proportional error over time to help determine an appropriate constant.

The reason for including the integral term in addition to proportional control is the integral control can eliminate steady-state error which is uncorrected by proportional control. The integral control has a few key components. Firstly, the calculation of the integral error is shown in Equation **4**. This error term is the sum of the error for the past five-time steps. The problem with integral error is it is initially very large which results in wind-up. The use of the anti-wind-up flag aims to reduce wind-up by removing the integral error from the PI controller when the rover reaches max speed.

$$E_{integral} = \int_{-\infty}^{t} E(t)\,dt \tag{4}$$

To ensure the PI controller does not exceed max speed the PI function is implemented. This function also corrects for negative backwards motion. When the rover is moving backwards the error is negative resulting in a negative PWM value. A negative PWM does not exist, hence the PI function corrects for this by multiply by the constant negative one.

2.0 Deviations from Project Scope

At this point in the project timeline the team has had no major deviations from the plan but has decided to investigate expanding the applications of the rover within airports. Originally the rover was being designed as only able to deliver objects from person to person, but the team will also be looking into an application to automate wheelchairs with the use of the rover.

3.0 Future Objectives

In the coming weeks, SSG Robotics will continue to make strides in prototype development. Firstly, SSG Robotics seeks to refine and tune the PID controller and test under a wider range of uses. The team still needs to implement the derivative component of the PID controller. This would involve integrating the IMU gyroscope data into the PID controller, to get more accurate feedback data while the rover is turning. The team will begin tests on integrating IMU acceleration data over time to find the corresponding velocity and position data. An application of this data is feedback to avoid veering when attempting to drive in a straight line. Other applications are to be determined with more testing.

The team also aims to integrate the While IR and LiDAR sensors into the development of navigation and obstacle avoidance features. The range sensors will provide the necessary feedback data to start development on a path finding algorithm. The team has identified the A star path finding algorithm to be best suited for faster computation on larger maps and overall familiarity with the program from previous experience.

SSG Robotics will continue research into potential secondary applications for the mobile rover. Inspiration for transporting passengers and or luggage in the terminal comes from an autonomous wheelchair company called WHILL. WHILL has been able to implement autonomous wheelchairs in Winnipeg Richardson, Los Angeles, and Miami International Airports **[1]**. SSG Robotics hopes to use this case study to reinforce the objective and feasibility of the project.

References

[1] “Industries,” *Whill.inc*, 2025. https://whill.inc/ca/mobility-service/industry (accessed Feb. 02, 2025).