2.2 Mechatronics Engineering

I was involved in mechatronics engineering for Science Olympiad, specifically in the Robot Tour and Electric Vehicle events. Throughout the seasons, my partners and I tested many different designs, some of which will hopefully serve as references for future generations of team members. Currently, I have transitioned into a coaching role rather than working on the projects myself. My past designs have continued to inspire new members of the team!

2.2.1 Autonomous Navigation Robot

Over the course of the year, I experimented with both kit-based and increasingly customized robot designs, starting from a standard Pololu kit and progressing toward more refined control implementations. I used the Romi 32U4 Control Board and chassis from Pololu as the base platform, which includes built-in quadrature encoders for precise wheel position and velocity feedback.

Figure 8: The first Romi 32U4 Control Board kit we recieved

Because encoder feedback was readily available, I focused first on closed-loop motor control using PID principles. Rather than relying on open-loop motor commands, I implemented proportional–integral–derivative (PID) control to regulate wheel speed and improve trajectory accuracy. This approach allowed the robot to correct for disturbances such as battery voltage changes, surface friction, and minor mechanical asymmetries.

As shown in **Figure 9**, I began by implementing proportional (P) control, where the motor output is adjusted based on the instantaneous error between the target speed and the measured encoder speed. Proportional control significantly improved responsiveness compared to open-loop control;

However, I observed steady-state error when the robot was subjected to sustained load or resistance. This highlighted the limitation of P-only control and motivated further exploration of full PID tuning.

```
// Encoder setup
float prevLeftEncoderCount = 0;
float prevRightEncoderCount = 0;
const float countsPerRevolution = 1440.0; // Adjust based on Romi wheel encoders
const float wheelDiameter = 70.0;         // mm (adjust to your robot's wheel diameter)
const float wheelBase = 142.0;            // distance between the wheels in mm (adjust to your robot)

class Movement {
  public:
    void advance(int targetCountLeft, int targetCountRight) {
      unsigned long startTime = millis();  // Start time for sequence
      encoders.getCountsAndResetLeft();
      encoders.getCountsAndResetRight();
      int encoderCountLeft = 0;
      int encoderCountRight = 0;

      while (encoderCountLeft <= targetCountLeft && encoderCountRight <= targetCountRight) {
        encoderCountLeft = encoders.getCountsLeft();
        encoderCountRight = encoders.getCountsRight();
        updateOdometry(encoderCountLeft, encoderCountRight);
        int error = encoderCountLeft - encoderCountRight;
        int integralCorrection;
        motors.setLeftSpeed(defaultForwardspeed - error);
        motors.setRightSpeed(defaultforwardspeed + error);
      }
      motors.setLeftSpeed(0);
      motors.setRightSpeed(0);

      unsigned long endTime = millis();  // End time for sequence
      Serial.print("Advance sequence time: ");
      Serial.print(endTime - startTime);  // Calculate time taken
      Serial.println(" ms");

      x=0;
      y=0;

      delay(standard_delay); // Short delay before next command
    }
```

Figure 9: Example code for proportional (P) control

Subsequent experiments incorporated integral and derivative terms to address accumulated error and reduce oscillations. The integral term helped eliminate steady-state bias caused by friction and motor imbalance, while the derivative term dampened rapid changes in error, improving stability during acceleration and deceleration. Through the calibrations shown in **Figure 10**, I was able to make the robot travel more accurately.

Multiple lines of code were implemented to control the robot's movements, including forward motion, right turns, and left turns. Each command was carefully programmed to correspond to specific distances and angles, allowing the robot to navigate the square grid accurately. Encoder feedback was used to measure wheel rotations, ensuring precise distance control, while conditional statements adjusted motor speeds to correct deviations in real time.

Figure 10: Robot calibration through repeated traversal of square cells

The competition track consists of a 5×4 grid of square cells, with each square measuring 50 cm $\times$ 50 cm. The robot must navigate from a designated start point to a target location by traveling cell-to-cell across the grid. Please refer to **Figure 11** for an example view of the track.

Figure 11: Sample track from one of the local invitationals

Barriers placed between certain squares act as obstacles, preventing direct movement through blocked paths. These barriers simulate walls, requiring the robot to plan alternate routes and execute precise turns to avoid collisions.

Additionally, there are bonus gates where points are awarded if the robot successfully reaches them. However, there is also a target time, and if the robot significantly exceeds this time limit, a substantial number of points are lost. Therefore, carefully balancing speed and bonus point collection is essential for achieving a high score.

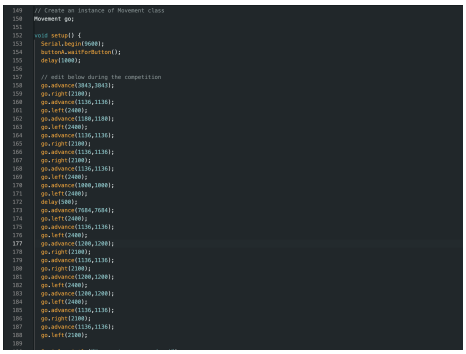

Figure 12: Example lines of codes utilized for track run

Please refer to **Figure 12** for the example lines of codes which I actually utilized for the competition run. Each movements lists the encoder count necessary to travel certain distance. The conversion is:

$$N_{\text{encoder}} = x \cdot \frac{64 \times 50}{2\pi \times 4}$$

Table 1: Variables and constants in the distance-to-encoder conversion

Symbol	**Meaning / Value**
N_{encoder}	Encoder count (number of ticks recorded)
x	Linear distance traveled by the robot (cm)
64	Encoder counts per motor revolution
50	Gear ratio (motor revolutions per wheel revolution)
2π	Constant from wheel circumference
4	Wheel radius (cm)

Along with PID control and calibrations, the robot was able to perform very well in many tournaments. The major purpose of the robot is to navigate through the track and reach the target point as close as possible as shown in **Figure 13**.

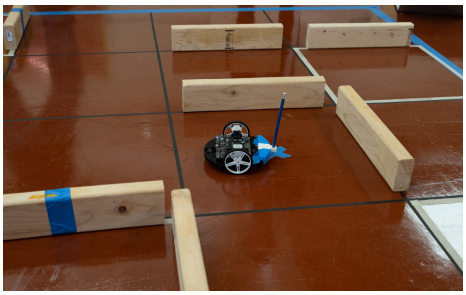

Figure 13: Romi robot successfully reaching the target point

2.2.2 New Custom Design

Nevertheless, we realized that the quadrature encoders on the Romi kit were not entirely accurate, especially over extended operation. Additionally, the round shape of the Romi made it difficult to align precisely to the track—a critical step to ensure the robot stayed on course.

To address these issues, we implemented a **custom robot design**. The hardware components included:

- **Control board:** A-Star 32U4 Prime Board with Dual G2 High-Power Motor Driver 18v18 Shield
- **Motors:** 50:1 Metal Gear Motors (12 V) with 64 CPR encoders

In addition, we designed the chassis to be **perfectly rectangular**, which made it easier to align the robot accurately to the track. This custom design improved both navigation precision and overall reliability during operation. Please refer to the **Figure 14** for the design of the robot.

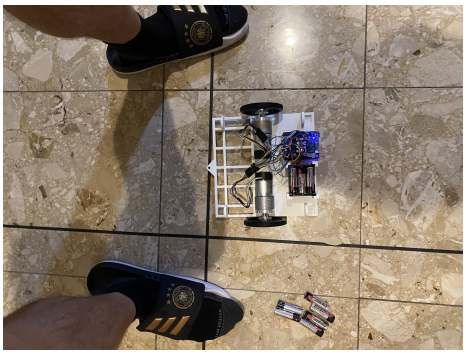

Figure 14: Overview of the customized robot

The major modification is the installation of the control board with the motor shield connected to the motors. Please refer to **Figure 15** for specific details.

The figure shows the black and red wires supplying power to the motor driver, as well as four additional wires, two of which power one of the motors. Each motor also has four wires connected to the encoders.

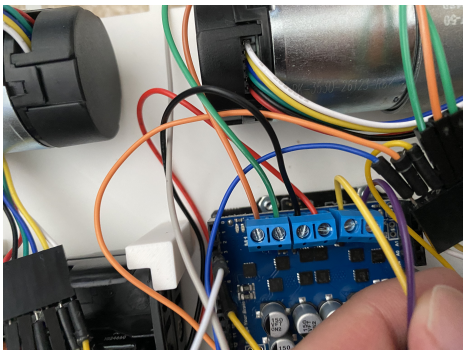

Figure 15: Diagram of motor driver connnected to the motors

Other modifications include the implementation of diagonal turns, as shown in **Figure 16**. Previously, we only employed 90° turns, but with the custom design, we have incorporated 45° turns to minimize the distance the robot must travel.

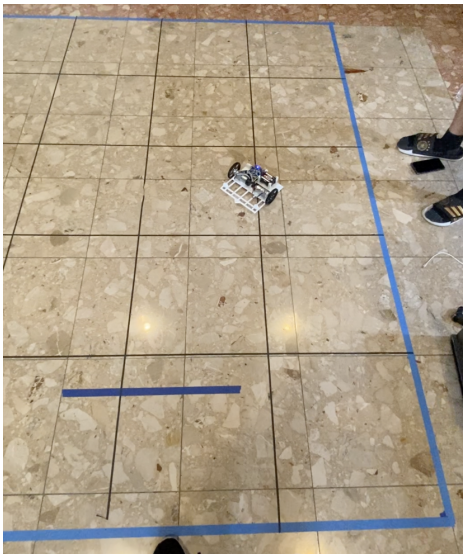

Figure 16: The robot performing diagonal turns on the practice track

2.2.3 Electric Vehicle

Below is the reflection my partner and I wrote for future generations. It includes detailed design notes, our trial-and-error process, and advice! A sample future design is attached at the bottom, although it may or may not be used by future teams