Autonomous RC Rally Car for Jump and Drift Racing

Presentation – Final Presentation

Om Bhatt|AEOP Intern

Who

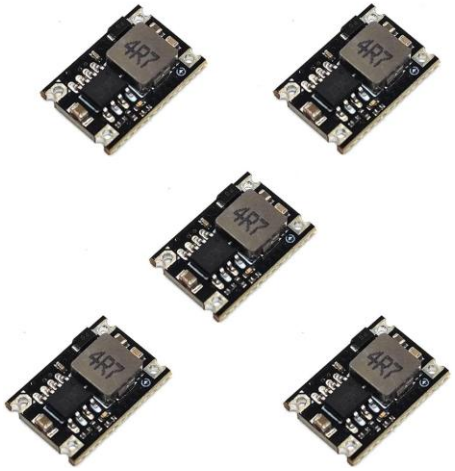
- AEOP Intern (Myself Om Bhatt)
- Autonomous Systems Group led by Dr. Ufuk Topcu
- Mentor: Dr. Christian Ellis
- AMRL Group (Prof. Joydeep Biswas and PhD Student Rwik Rana)

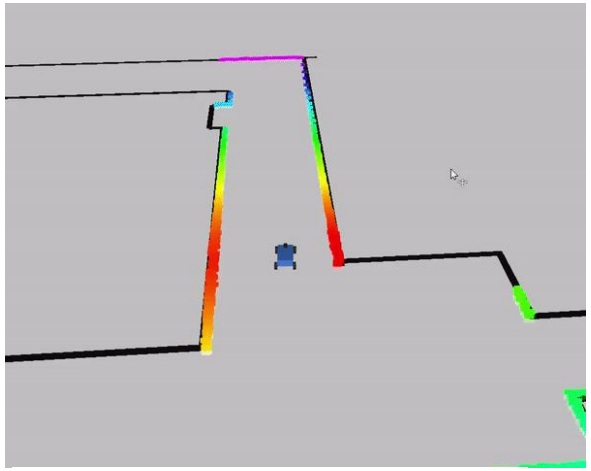
What

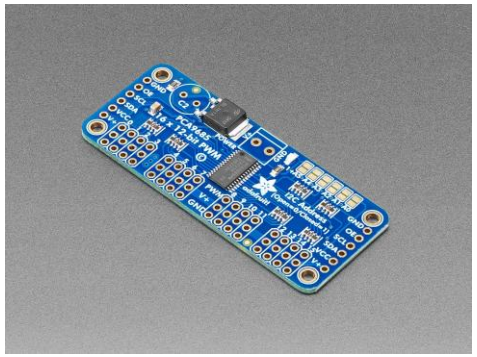
- **Objective 1:** Build a modular, autonomous 1/16 scale RC car platform and software platform using ROS2,
- Complete a "Lego-like" guide for assembly.
- **Objective 2:** Research platform: Make the RC car autonomous (Jump/Drift SLAM navigation).
- **Design:** Jetson Nano or Raspberry Pi 5 for compute, ROS2-based control stack.

Why

- What is the bigger picture/project?
  - Ongoing off-road terrain adaptation research
  - Ongoing jump racing research
  - Research platform for incoming high-school students as part of a robotics program
- Research Innovation: Enable development of a low-cost attritable vehicles for complex navigation research.
- Unique Challenge: RC Jump/Drift Racing introduces new autonomy dynamics (e.g., jumping and drifting navigation)
- Accessibility: Reduce cost of autonomous vehicle research platforms







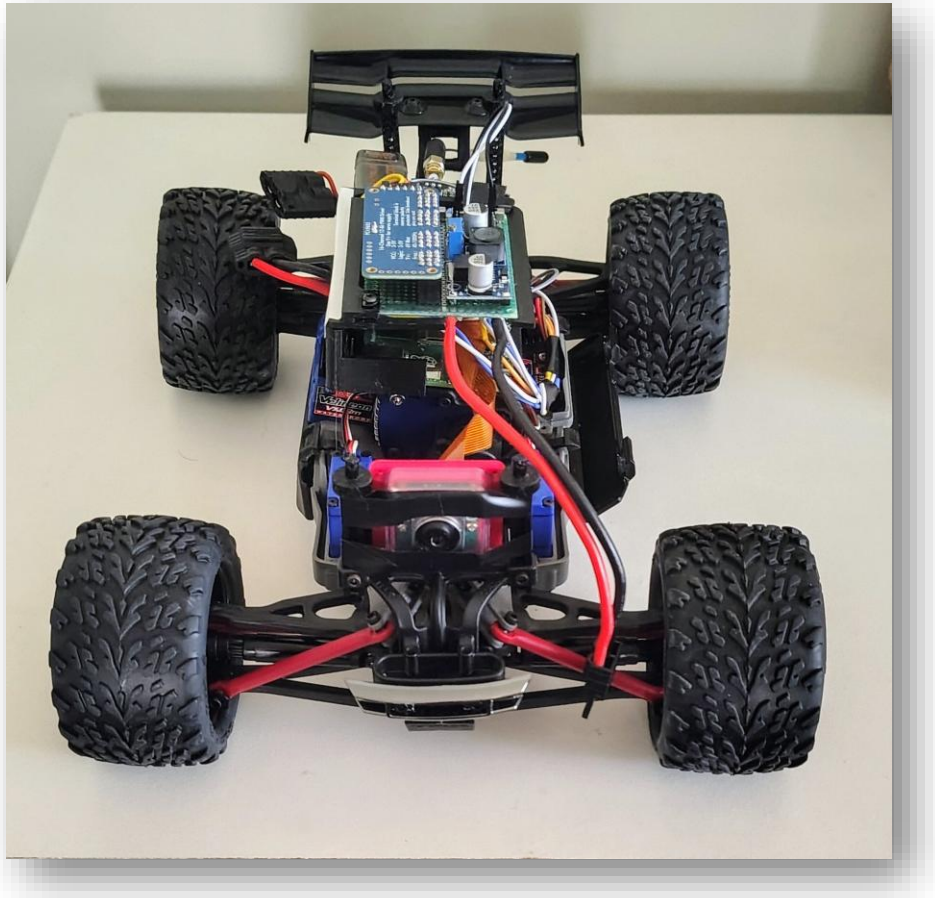
Overview of components:

- Compute: Raspberry Pi 5 (RPi5) (or Jetson Nano)
- Motors:
  - Traxxas 2080 servo (Left and Right)
  - Traxxas brushless motor: 50,000 rpm
  - Traxxas electronic speed controller (ESC): 6-12V @ 150A peak current
- Control: PCA PWM 9685 servo controller
- Perception: Arducam Rpi camera + IMU
- Communication: Crazyradio PA + Spektrum Transmitter
- Software Stack: ROS2 for control and perception integration

Snapshot of Project Milestones

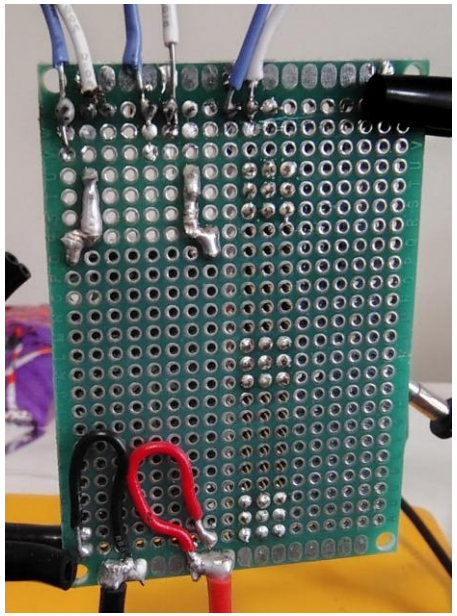
Milestones

1. Custom Circuit Board Design and Soldering
2. Testing Switch Functions
3. Testing Buck Convertor Functions
4. 3D Printing Mounts
5. Electromechanical Assembly
6. Chassis Layout and Wiring
7. Raspberry Pi Setup
8. Control Software Codebase
9. Testing Low-level control via Spectrum Radio Wireless + PCA
10. Final Demo

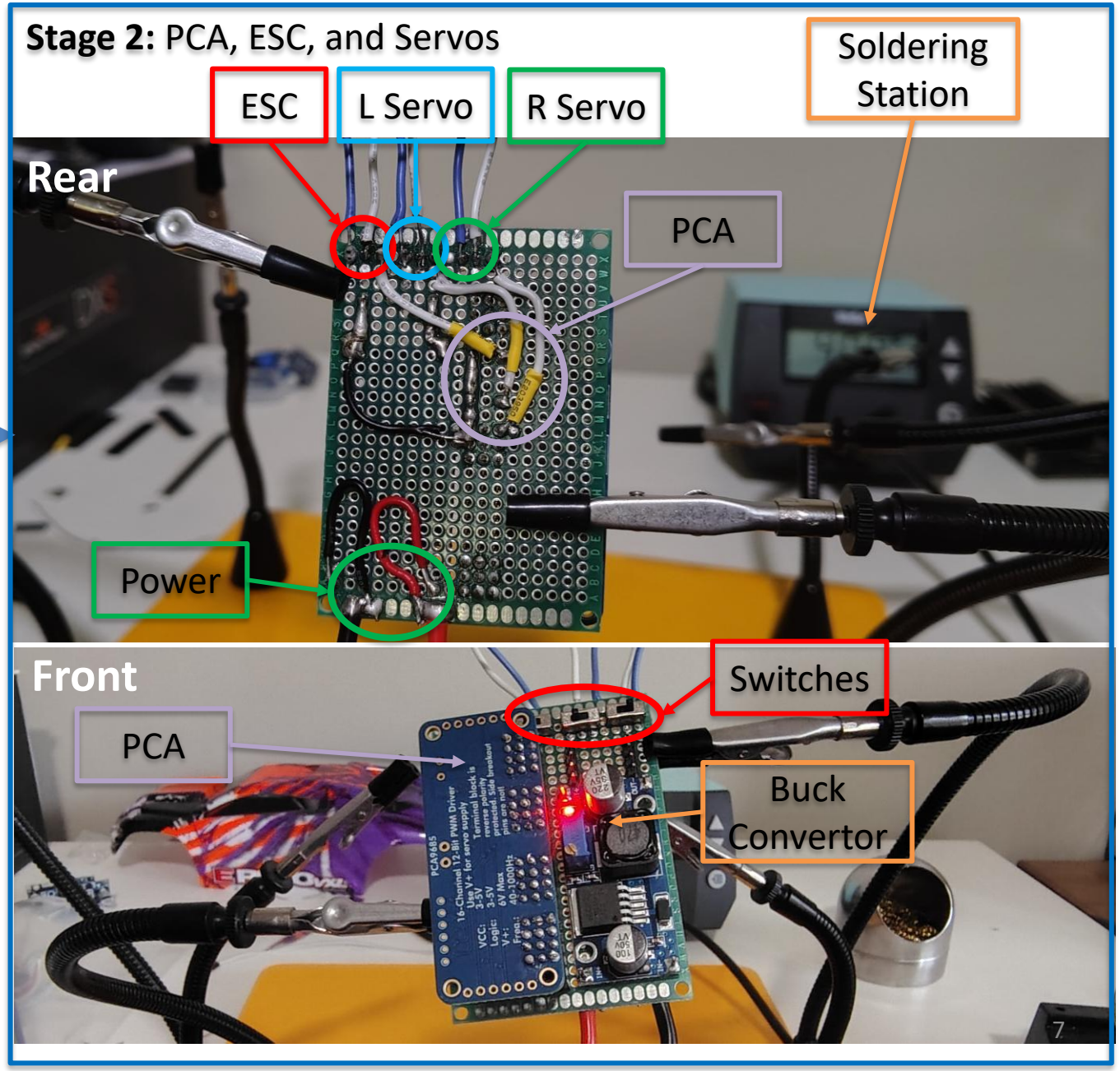


Milestone 1: Custom Circuit Board Design and Soldering

Stage 1: Power and Switches



Stage 2: PCA, ESC, and Servos

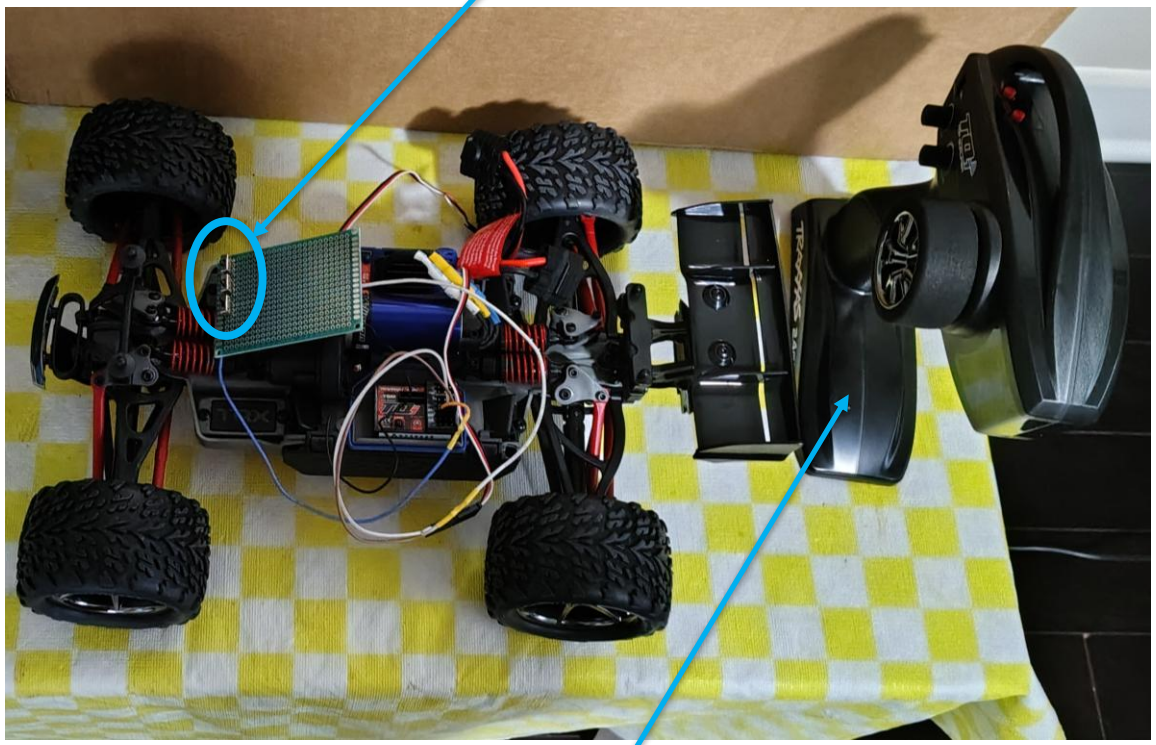


Key accomplishment:

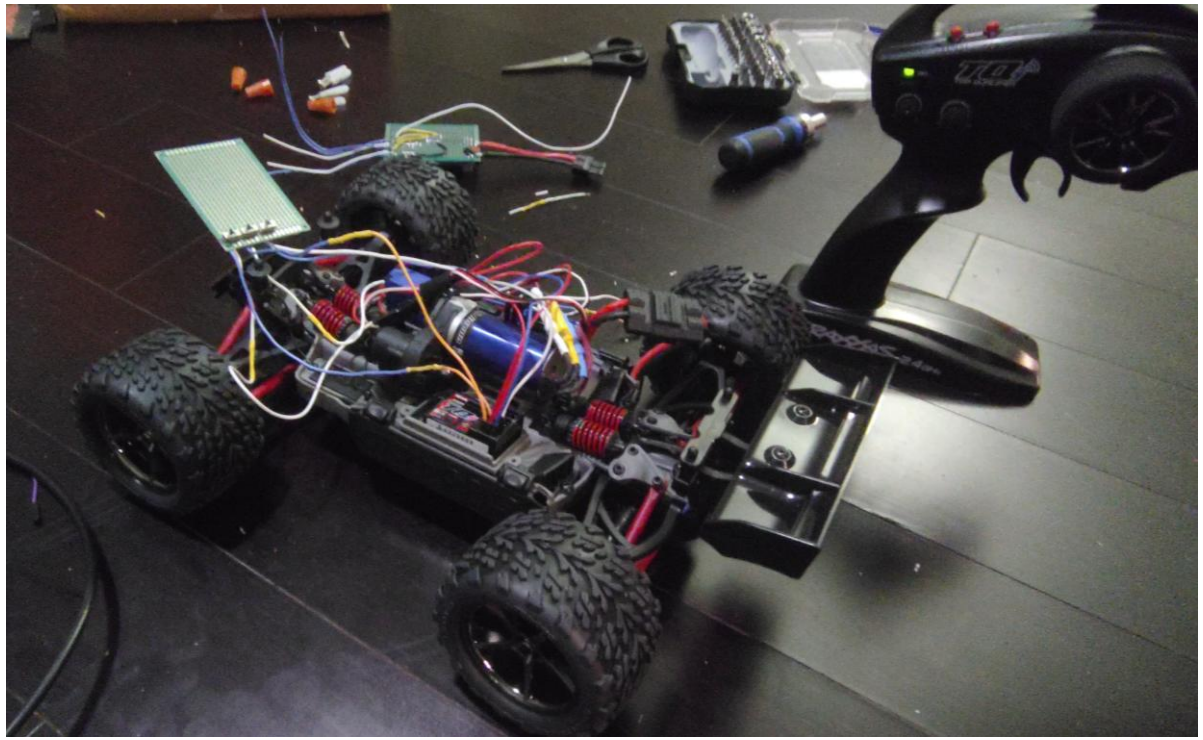
- All electrical components connected to each other in a neat manner

Milestone 2: Testing Switch Functions

3 switches for individual control of ESC, L Servo, R Servo



Stock Traxxas controller for testing



Key accomplishment:

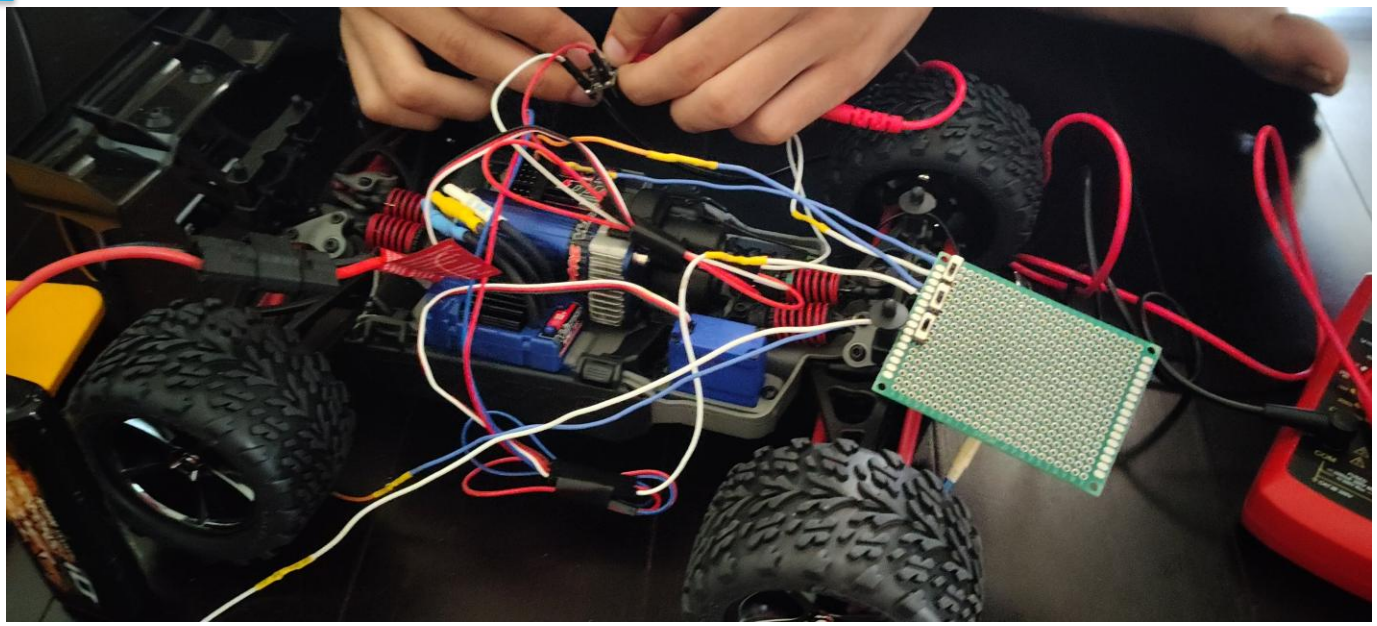
This video demonstrates a test of controlling the ESC and Servo individually by turning the respective switches on and off

Milestone 3: Testing Buck Convertor Functions

Connected Traxxas 7.2V NIMH Battery to Buck Convertor

Battery wire connectors

Multimeter showing desired voltage output (5V)

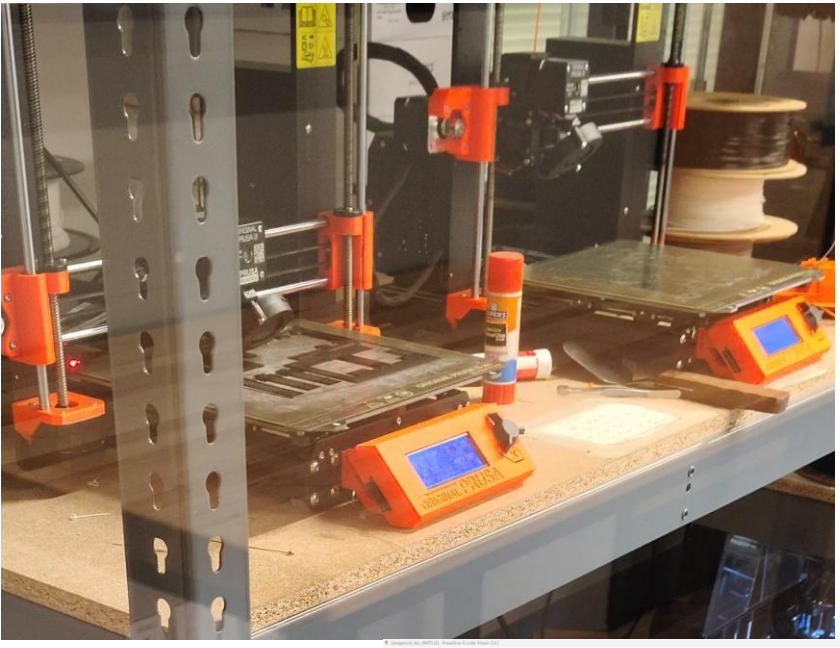


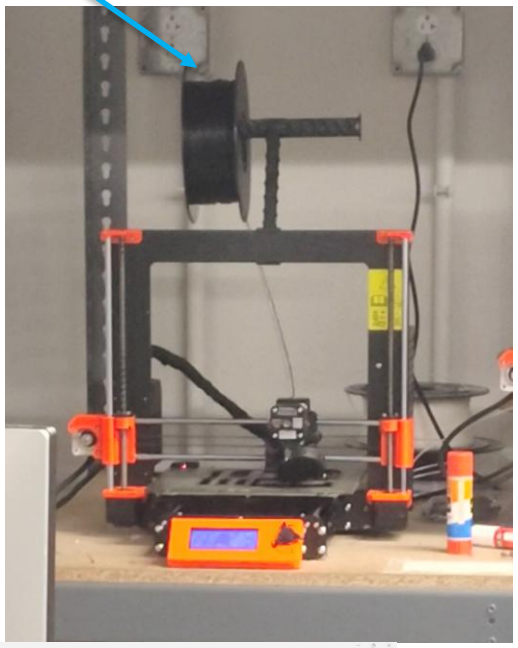
Key accomplishment:

- This video demonstrates a test of the 5V buck convertor
- This test is needed to confirm that the buck convertor is stepping down the voltage from the 2 cell NIMH battery (7.2V) to 5V
- This is the required voltage for the RPi 5 or Jetson Nano needs to run

Milestone 4: 3D Printing Mounts

PLA Filament Installed

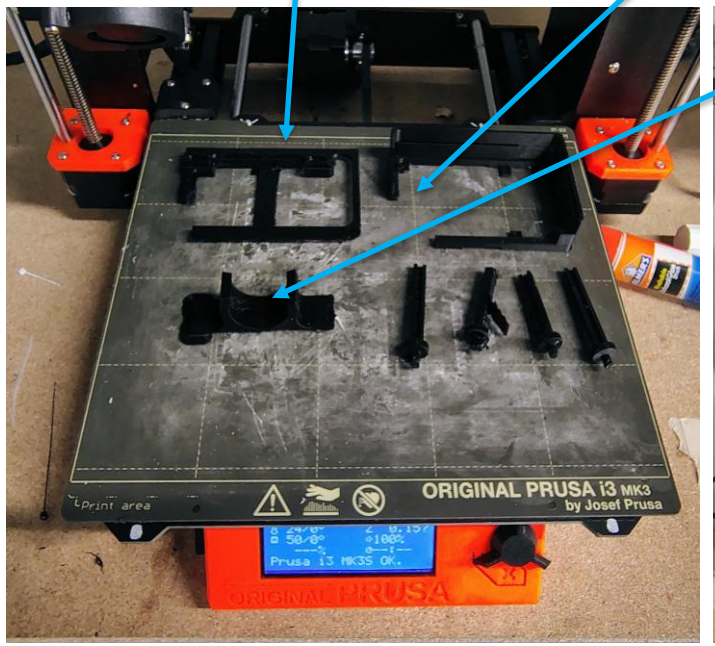


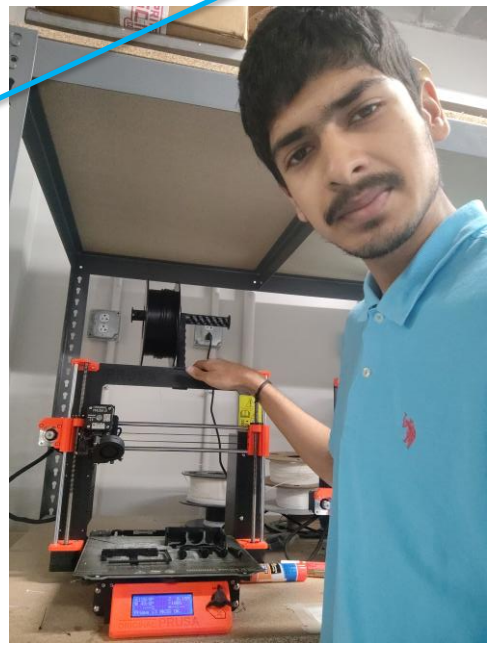


Casing - Top

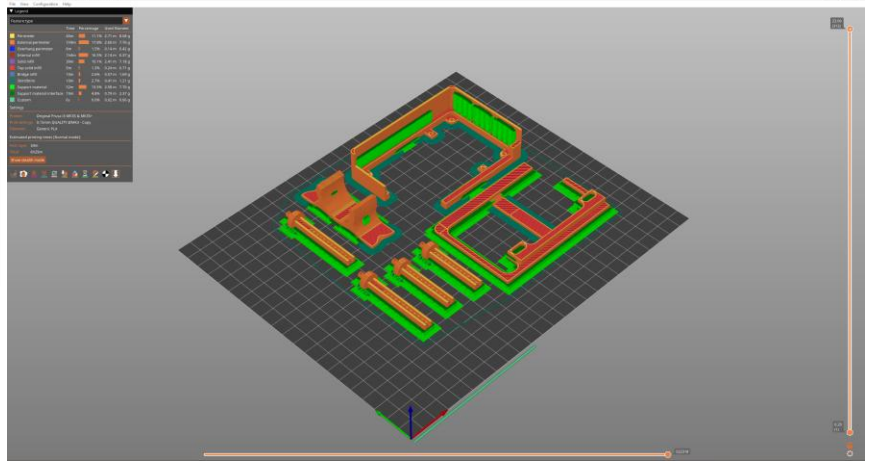
Casing - Bottom

Motor Mount





Prusa Slicer

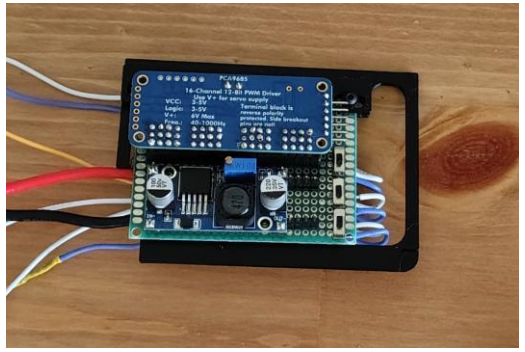


Key accomplishment:

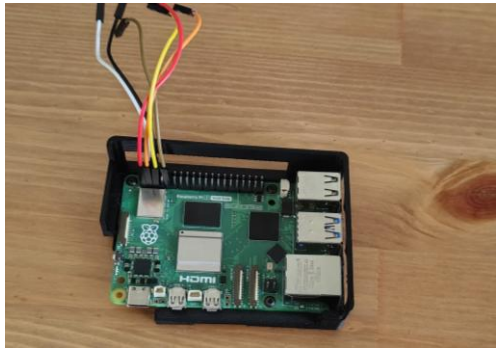
- Prepared CAD parts (mount and casing for circuit board and RPi 5) for 3D printing
- Setup Prusa 3D printer for fabricating the parts by installing PLA filament
- Properly removed the parts and unloaded PLA Filament from the 3D Printer

Milestone 5: Electromechanical Assembly

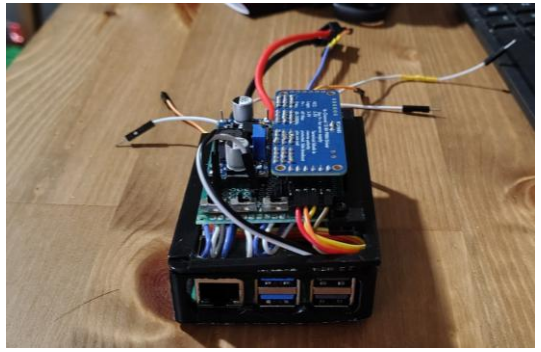
Circuit board with case

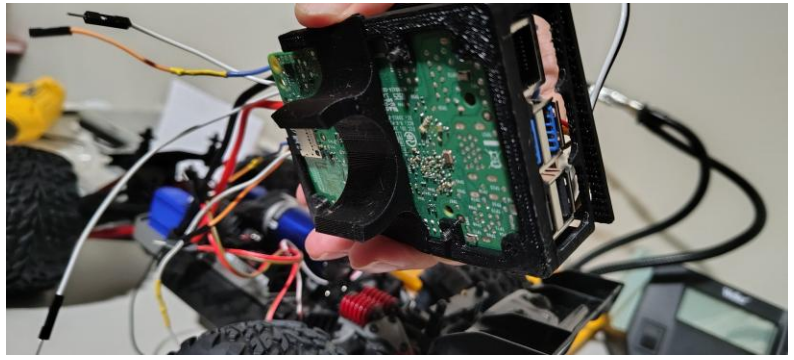


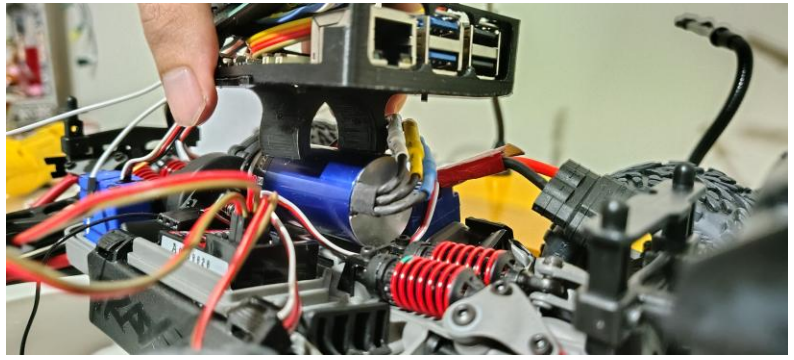
Raspberry Pi 5 with bottom case

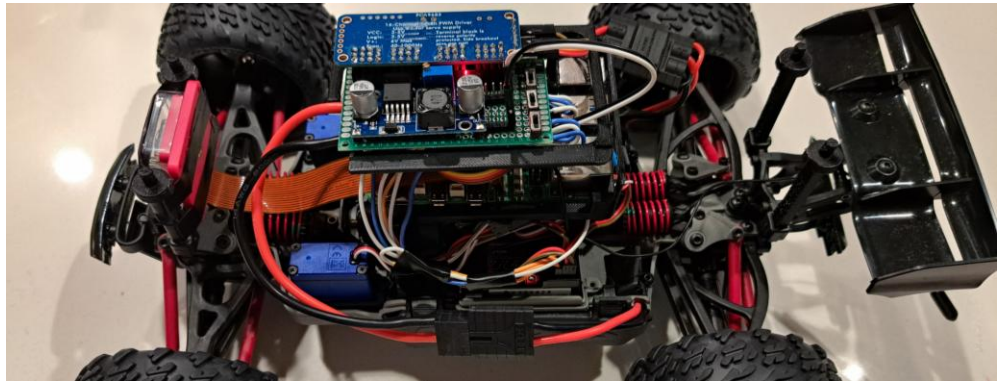


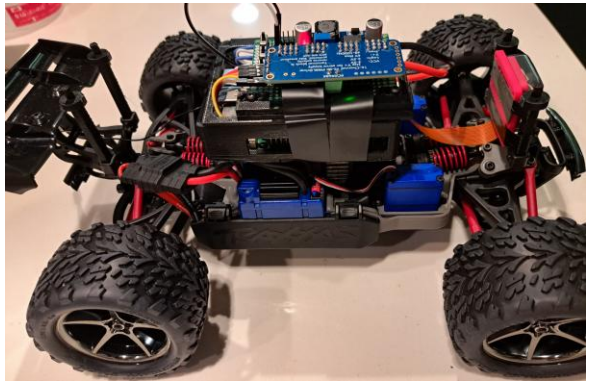
Raspberry Pi 5 and Circuit Board mounted on completed case



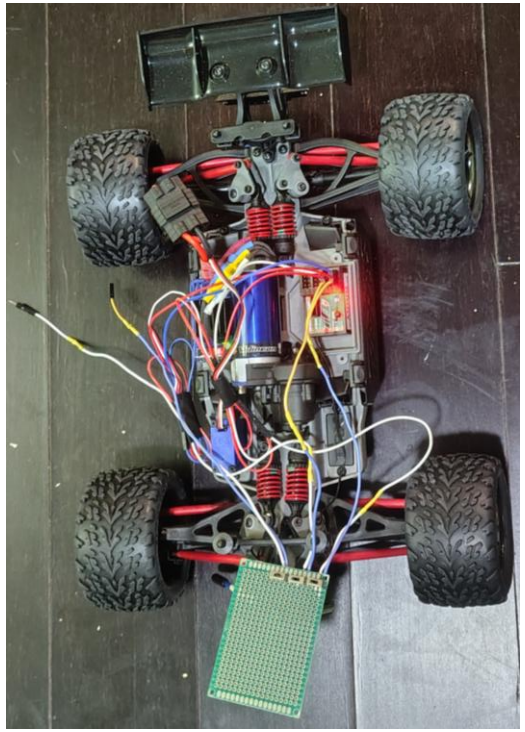


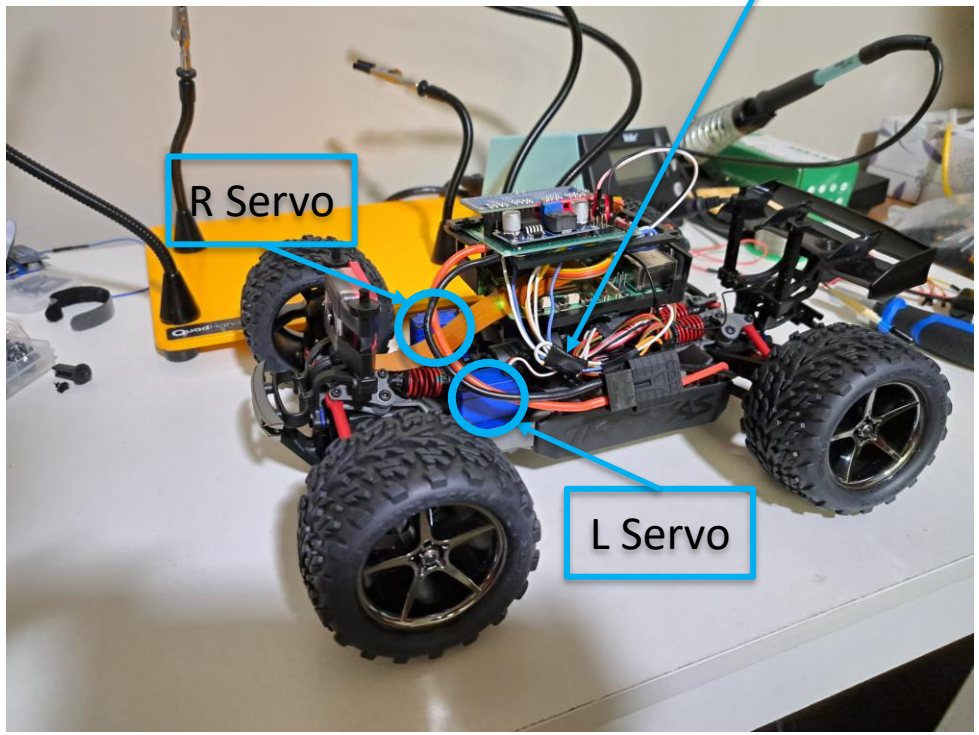






Milestone 6: Chassis Layout and Wiring



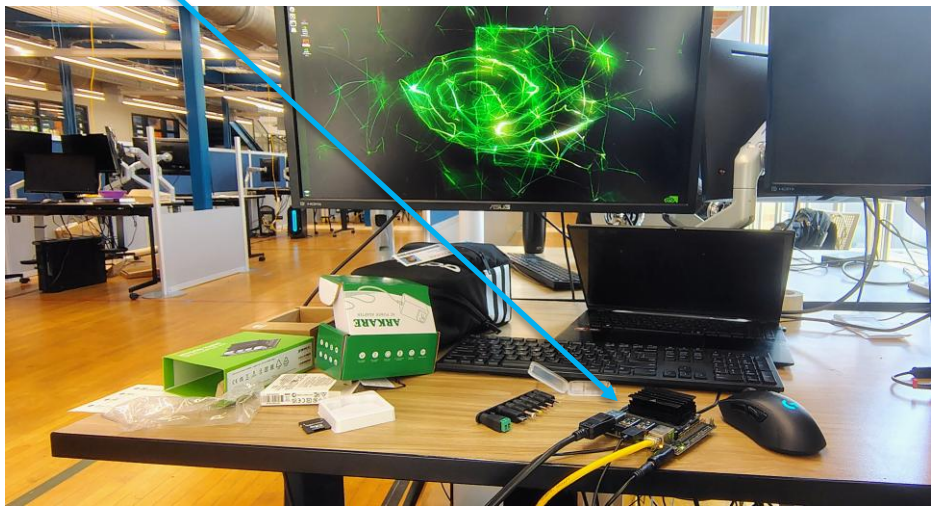


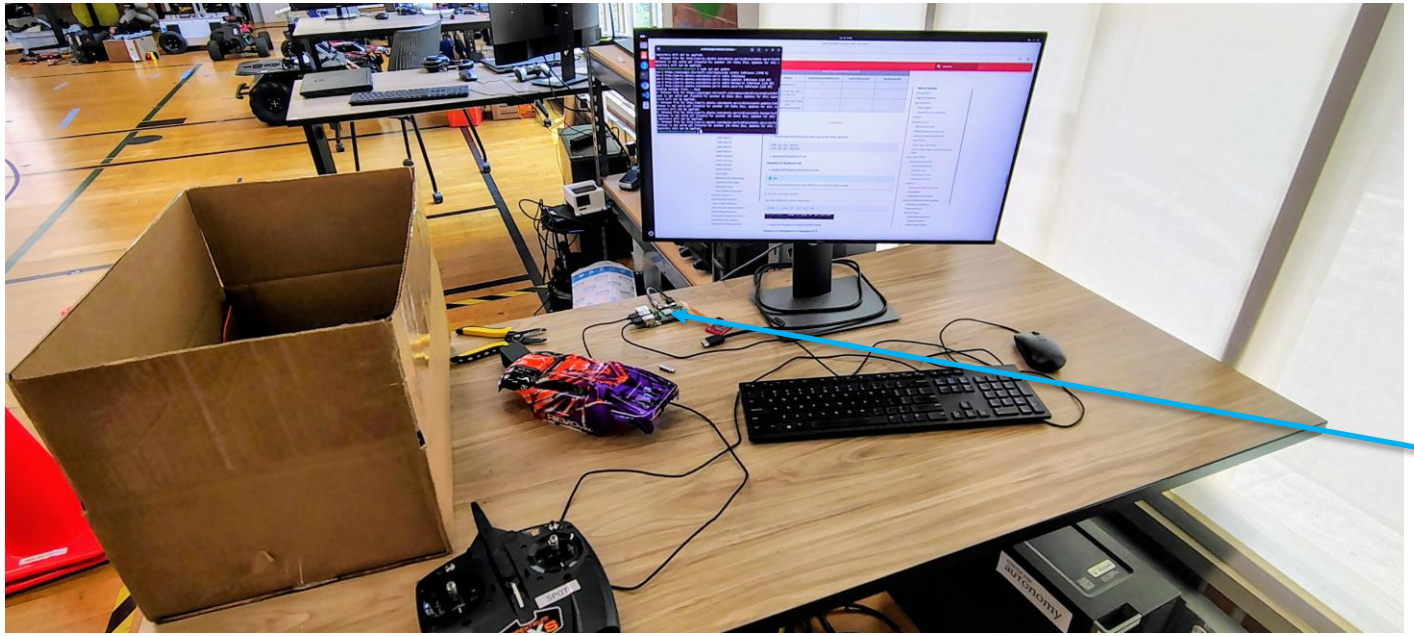
Key accomplishment:

- Cleaned up initial messy wiring for organized wire management, leading to ease of access
- Installed second servo motor for enhanced steering control
- Ensured that the 3D printed mounts for circuit board and RPi 5 are sturdy to prevent any damage when drifting and jumping

Milestone 7: Raspberry Pi Setup

Jetson Nano Booted With Ubuntu 22.04





Key accomplishment:

- Flashed Ubuntu on both RPi 5 and Jetson Nano
- Installed ROS2 on RPi 5
- Ready for development

RPi 5 Booted With Ubuntu 24.04.2 LTS

Milestone 8: Control Software Codebase

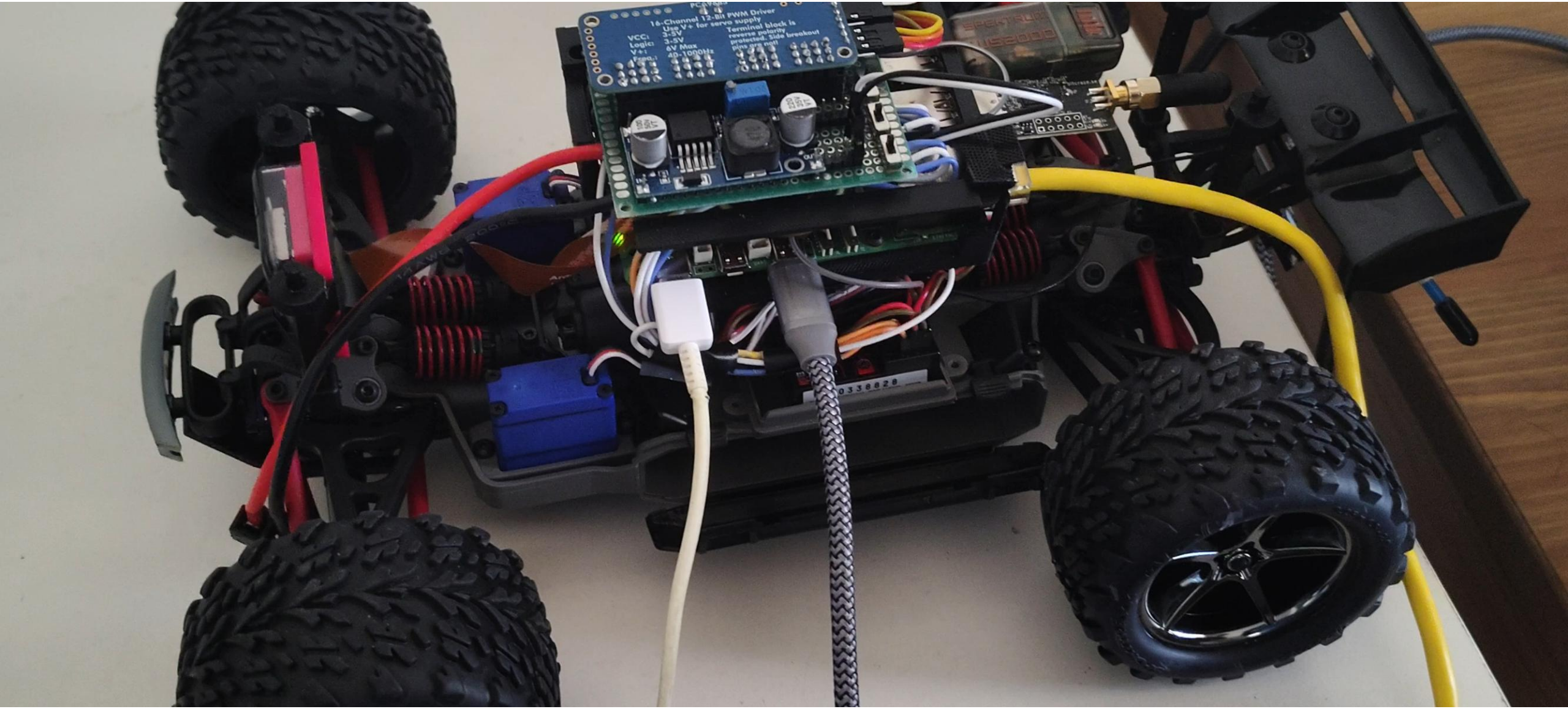
Steering Control (Ackerman)

```
41  void set_steering_ackermann(PCA9685 &pca, float steering, float throttle) {
42
43      float servo_mid_angle_rad = (SERVO_MID_ANGLE - 90) * M_PI / 180 ; // converting to radians
44      float servo_mid_angle_reverse_rad = (SERVO_MID_ANGLE_REVERSE - 90) * M_PI / 180 ; // converti
45      float ackermann_steering = steering;
46
47
48      std::cout << "inp steer " << ackermann_steering << std::endl;
49      ackermann_steering = clamp(ackermann_steering, -STEER_CLAMP, STEER_CLAMP);
50      ackermann_steering = ackermann_steering * STEER_SCALE;
51
52
53
54      float theta_left_steer_hat = center_wheel_to_right_servo(-ackermann_steering);
55      theta_left_steer_hat = -(fmod(theta_left_steer_hat + M_PI /2.0f, M_PI) - M_PI/2.0f);
56
57
58      float theta_right_steer_hat = center_wheel_to_right_servo(ackermann_steering);
59      theta_right_steer_hat = fmod(theta_right_steer_hat + M_PI / 2.0f, M_PI) - M_PI / 2.0f;
60
61      float mid_offset = SERVO_MID_ANGLE;
62      if (throttle <0){
63          mid_offset = SERVO_MID_ANGLE_REVERSE;
64      }
65
66      float theta_left_steer_hat_deg = theta_left_steer_hat * 180 / M_PI + mid_offset;
67      float theta_right_steer_hat_deg = theta_right_steer_hat * 180 / M_PI + mid_offset;
68
69      std::cout << "left_angle " << theta_left_steer_hat_deg << " right_angle " << theta_right_stee
70
71      uint16_t left_pwm_value = angle_to_pwm(theta_left_steer_hat_deg);
72      uint16_t right_pwm_value = angle_to_pwm(theta_right_steer_hat_deg);
73
74
75      pca.set_pwm(SERVO_CHANNEL_LEFT, 0, left_pwm_value);
76      pca.set_pwm(SERVO_CHANNEL_RIGHT, 0, right_pwm_value);
77  }
78
```

Throttle Control

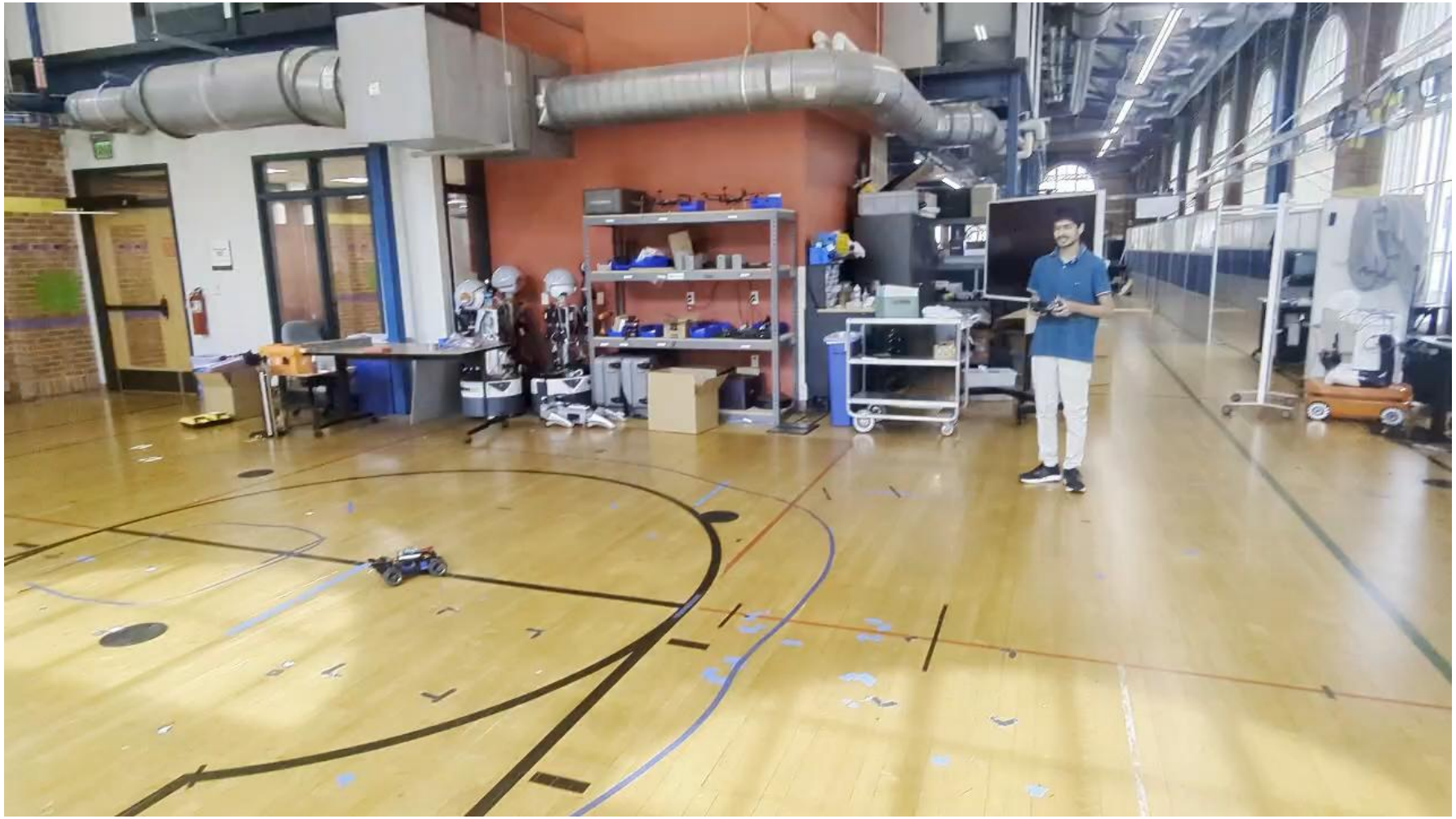
```
169 void set_throttle(PCA9685 &pca, float throttle, bool arming, float prev_throttle,
170                 bool &breaking_mode, int &breaking_pwm,
171                 std::optional<float>& curr_heading_vel_ptr, int &breaking_count_tick) {
172
173     uint16_t pwm_value;
174     ////////////////////////////////////
175     if (arming) {
176         pca.set_pwm(ESC_CHANNEL, 0, ESC_NEUTRAL);
177         return;
178     }
179
180     if (curr_heading_vel_ptr) {
181         std::cout << "curr_heading_vel : " << *curr_heading_vel_ptr << std::endl;
182     } else {
183         std::cout << "curr_heading_vel : None" << std::endl;
184     }
185
186     ////////////////////////////////////
187
188     bool direction_flip = (throttle * prev_throttle < 0.0f) && (fabs(throttle) > BRAKE_THROTTLE_THRESHOL
189     if(direction_flip){
190         pca.set_pwm(ESC_CHANNEL, 0, ESC_NEUTRAL);
191         usleep(100000);
192         breaking_mode = false;
193     }
194
195     ////////////////////////////////////
196
197     pwm_value = ESC_NEUTRAL;
198     // pwm_value = throttle_to_pwm(throttle);
199
200     ////////////////////////////////////
201
202     if (fabs(throttle) > BRAKE_THROTTLE_THRESHOLD) {
203         if(breaking_mode == true){
204             breaking_mode = false;
205             pca.set_pwm(ESC_CHANNEL, 0, ESC_NEUTRAL);
206             usleep(10000);
207         }
208         pwm_value = throttle_to_pwm(throttle);
```

Milestone 9: Testing Low-level control via Spectrum Radio Wireless + PCA



Milestone 9: Testing Low-level control via Spectrum Radio Wireless + PCA

Milestone 10: Final Demo

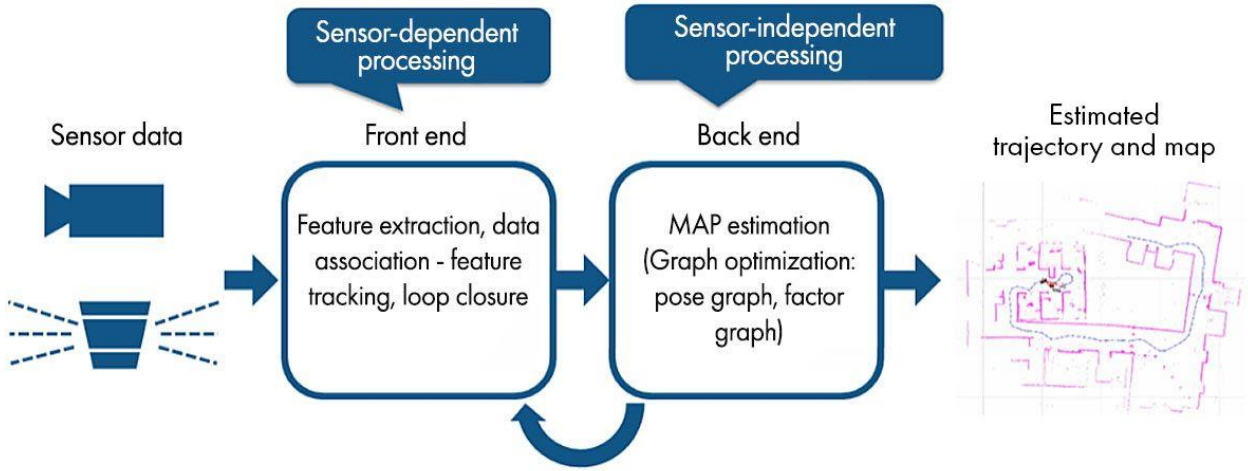


Key accomplishment:

- Full teleoperation control via raspberry pi PCA 9685 is successfully working
- The platform is ready and capable for autonomous vehicle research and development
- The platform has drifting and jumping capabilities

Next steps

- Collect sensor data for various drift and jump scenarios
- Run an off-the-shelf SLAM codebase on the stack
- Integrate a high-level controller and connect it to the low-level controller I have developed
- Work with mentors on a research publication they are aiming in the September/November



SLAM (Simultaneous Localization and Mapping):

- Enables the car to build a map of the environment and localize within it in real-time
- Essential for navigating unpredictable jump landing zones and drift corners
- Helps maintain awareness after airborne movement (where odometry fails)

Application to Jump Navigation:

- SLAM can re-localize the vehicle post-jump using visual/IMU data
- Allows estimation of optimal trajectory and landing correction

Application to Drift Navigation:

- SLAM + IMU fusion improves control when wheels lose traction
- Useful for sharp turn handling where pure kinematics are unreliable

Simulation Example: F1Tenth Simulator:

- Open-source platform with ROS2 integration
- Supports virtual testing of SLAM, planning, and control algorithms
- Reduces cost and risk by validating approaches before physical deployment

Closing Remarks

- A research platform for RC jump and drift racing is ready for autonomy
- Full autonomous low-controller is ready for use
- Will be continuing to contribute to the research and the lab once I am gone via a research publication with my mentors
- Gained valuable research experience and attended informative lectures and advice for graduate school
- Future REU interns should experience research before making their mind toward an industry focus