CPEG Capstone: Hyperdrive Final Report

Andrew Congdon, Manny Hamer, and Ryan Paillet

December 17th, 2025

Hyperdrive

Prepared by

Group 4

University of Notre Dame

Notre Dame, IN 46556

Approved by

Ryan Paillet
Manny Hamer
Andrew Congdon

On 12/17/2025

Objectives and Expected Significance

Project Motivation

We designed a modern take on common slot car racing toys that were popularized in the 1960's and 1970's. With nearly all children's entertainment coming from video games and short-form video media today, we believed there is a market for revitalizing once-popular physical toys using modern technology. This project is intended to bridge the gap between hands-on play with physical toy cars and the interactive, competitive elements commonly found in video games. We anticipate that this concept could appeal not only to children, but also to teens and young adults interested in the growing smart toy market.

Additional details, visuals, and technical documentation for the project can be found at https://hyperdrive-capstone.dev, which readers are encouraged to explore.

System Features

The proposed system features two miniature cars equipped with motors, sensors, and Bluetooth/Wi-Fi communication capabilities, operating on a closed-loop racetrack with two lanes. Each car can autonomously remain within its lane using magnetic hall effect sensing. Players can interact with the system through an IOS or Android application, allowing them to command their car to change lanes, adjust speed, determine distance to the finish line, and activate certain virtual power-ups.

Technical and Research Goals

Beyond functioning as a game, the project is intended to serve as a technical foundation for future car designs for Hyperdrive and the overall robotics industry. A key objective was to explore low-cost, miniaturized implementations of line-following and magnetic sensing techniques commonly used in robotics. At the outset of the project, there were limited known examples of robust control system strategies for this type of magnetic sensing technique in such a fast-paced, game-oriented system, especially at this small of a scale. Our goal was to create a proof-of-concept of concept of this technology in action, so that we can leave the door open for further iterations.

Table 1: Key Modifications from Original Proposal

Design Element	Original Proposal	Final Implementation
Number of Cars	4	2
Visual Feedback (LEDs)	Integrated LEDs	No LEDs
Onboard Video Feed	Possible Live Video Feed	Not Implemented

Rationale for Design Modifications

1. **Reduction from four cars to two cars:** Implementing two cars was sufficient to demonstrate competitive behavior while enabling the development of a stable minimum viable product (MVP).

This reduction simplified system synchronization, control complexity, and debugging without compromising the core competitive objective.

2. **Removal of LED-based visual effects:** While LEDs were originally intended to enhance visual feedback, they were determined to be non-essential for demonstrating core system functionality and were removed to allow greater focus on control, sensing, and system stability.

3. **Elimination of the live video feed:** Although initially considered as a potential extension feature, the Bluetooth module lacked sufficient bandwidth to reliably multiplex real-time video streaming alongside control and telemetry data.

Rationale and Related Background

We chose this project because of its diverse technical requirements, with equal parts software and hardware development. There is some mechanical design for the car itself, but that is relatively small compared to the complexity of the hardware and software. A couple of our group members were inspired by a product already in existence called Anki Overdrive, the successor to Anki Drive. Anki Drive was originally released in 2013, with Anki Overdrive following in 2015.

The choice of this project is driven by a unique market opportunity. Anki, the company behind Anki Overdrive, went out of business in 2019. While their intellectual property and assets were acquired, there has been a significant gap in the market for a modern, app-controlled robotic car racing system that offers the same level of sophistication and integrated gameplay. Existing alternatives, like traditional slot cars (e.g., Scalextric[1] or Carrera[2]), offer physical racing but lack the integrated AI, autonomous features, and video-game-like elements that made Anki Overdrive so popular. Such companies are more focused on replicating real life car models and races rather than simulating fast-paced battles on the track with power-ups. Other similar toys, such as WowWee's R.E.V. line[3], focus only on having car battles without any racing element involved.

Our project aims to directly address this gap in the market, offering several key distinctions:

Streamlined Design, Targeted Innovation: In contrast to Anki Overdrive's modular track system, our approach relies on a single mat. This simplification reduces mechanical complexity and enables us to concentrate development on the core computer engineering challenges. The central innovation of our design is the vehicle's onboard sensor system for track detection. Anki's system relied on track pieces embedded with proprietary patterns for infrared sensors. We are primarily exploring a more novel system that would utilize hall effect sensors and magnetic strips to keep the cars on the track. This trade-off is essential to keeping the project achievable (due to no custom track pattern fabrication) within a semester while still showcasing our ability to tackle a fundamental problem in a novel and practical way.

Our system serves as a strong demonstration of **fundamental computer engineering concepts**. The project challenges us to design and implement a complete computing system that integrates hardware and software seamlessly. It requires the application of embedded systems design, digital signal processing, and wireless communication. In this way, the project provides a direct connection between classroom theory and hands-on engineering practice.
Although designed as an advanced toy, the underlying technologies carry **significant potential beyond entertainment**:

- **Logistics and Warehousing:** The AI-driven path-following and sensor system could be adapted to develop autonomous guided vehicles (AGVs) for transporting goods in industrial environments.
- **STEM Education:** The platform could be commercialized as an open-source educational kit,

enabling students to explore robotics, AI, and embedded systems by programming their own miniature autonomous vehicles.

Importantly, Anki's failure was not caused by a lack of consumer interest, but by **financial and operational challenges**. This leaves a substantial opening in the market for a system that can truly succeed in blending **physical play with digital interactivity**. With the global smart toy market projected to reach **$77.8 billion by 2033** (Market.us), the demand for innovative hybrid play experiences is both clear and rapidly expanding.

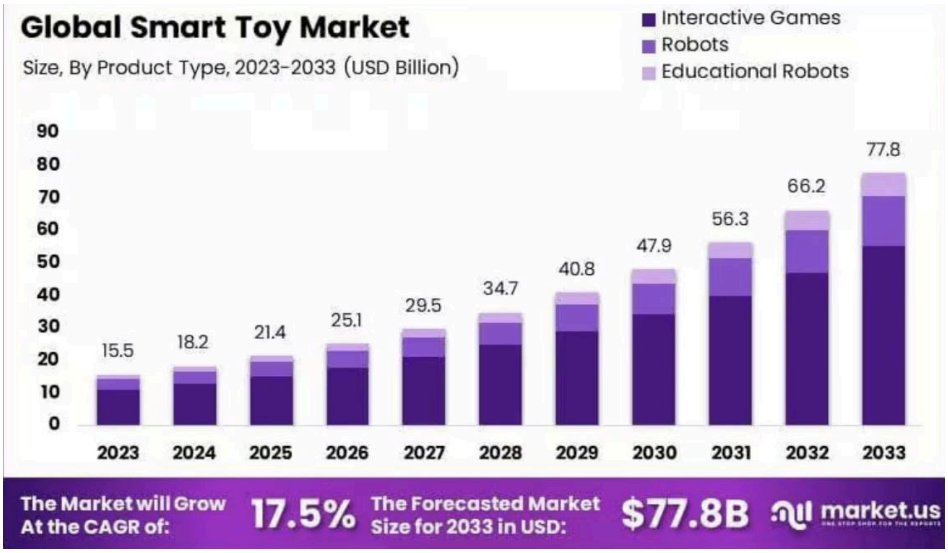


Figure 1: Market graph of electronic physical toys projection [4]

References

[1] Scalextric Slot Racing Homepage

[2] Carrera Slot Racing Homepage

[3] Wowwee Rev Battle Cars

[4] Global Smart Toy Market Review

Approach

Early Design Review

Following refinement of the initial design proposal, the intended electrical architecture was narrowed down and a preliminary concept for Hall Effect sensor-based magnetic line following was developed. While the proposed approach appeared feasible, external validation was sought before proceeding further with detailed implementation.

To this end, a design review was conducted with the former Anki Overdrive AI Lead, one of the first engineers to join the company and a key contributor to the development of their autonomous lane-following system. This meeting provided valuable insight from an industry expert with direct experience in magnetic lane tracking for small autonomous vehicles.

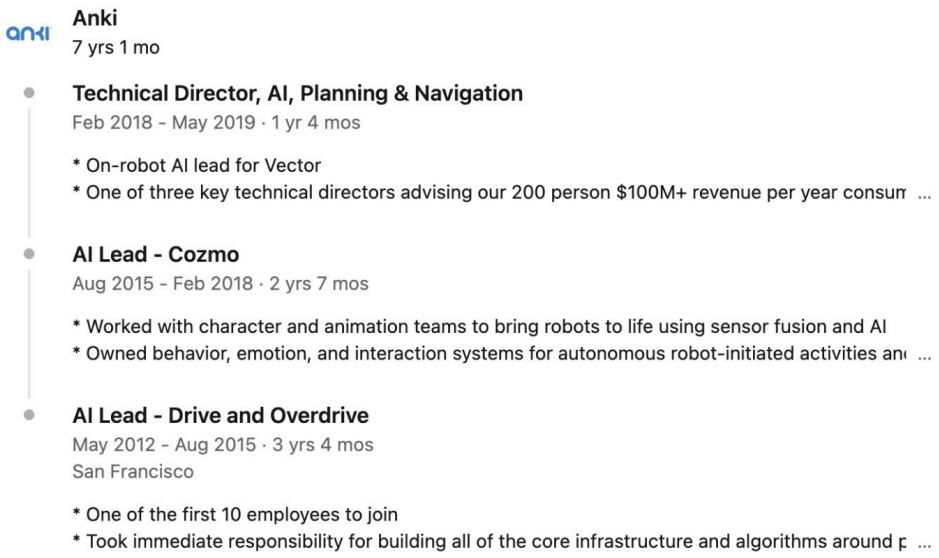


Figure 2: Former Anki Overdrive AI Lead we performed design review with

Several key questions were presented during the discussion to assess the feasibility of the proposed sensing and control approach and to identify potential pitfalls early in the design process. The questions and corresponding feedback are summarized in Table 2.

Table 2: Expert Feedback from Anki Overdrive AI Lead

Question	Feedback
Would a Hall Effect sensor-based magnetic lane-following approach be viable?	The approach was considered feasible and appropriate for the intended application.
Are there any potential issues or limitations to be aware of?	Sensor placement was identified as critical to achieving smooth and responsive control.
Do you have any general advice for the design moving forward?	Recommended placing Hall Effect sensors as far forward as possible and discussed potential strategies for implementing lane changes using magnetic sensing.

The feedback reinforced confidence in the proposed sensing strategy while also highlighting specific design considerations that were incorporated into later iterations. In particular, the recommendation to move the Hall Effect sensors toward the front of the vehicle directly influenced subsequent mechanical and electrical design decisions. Additionally, early discussion of lane-changing concepts helped guide the development of control algorithms later in the project.

With this information in mind, several key future design priorities were established, including optimized sensor placement. These priorities are reflected in the design choices and algorithms discussed in subsequent sections.

High Level Diagram

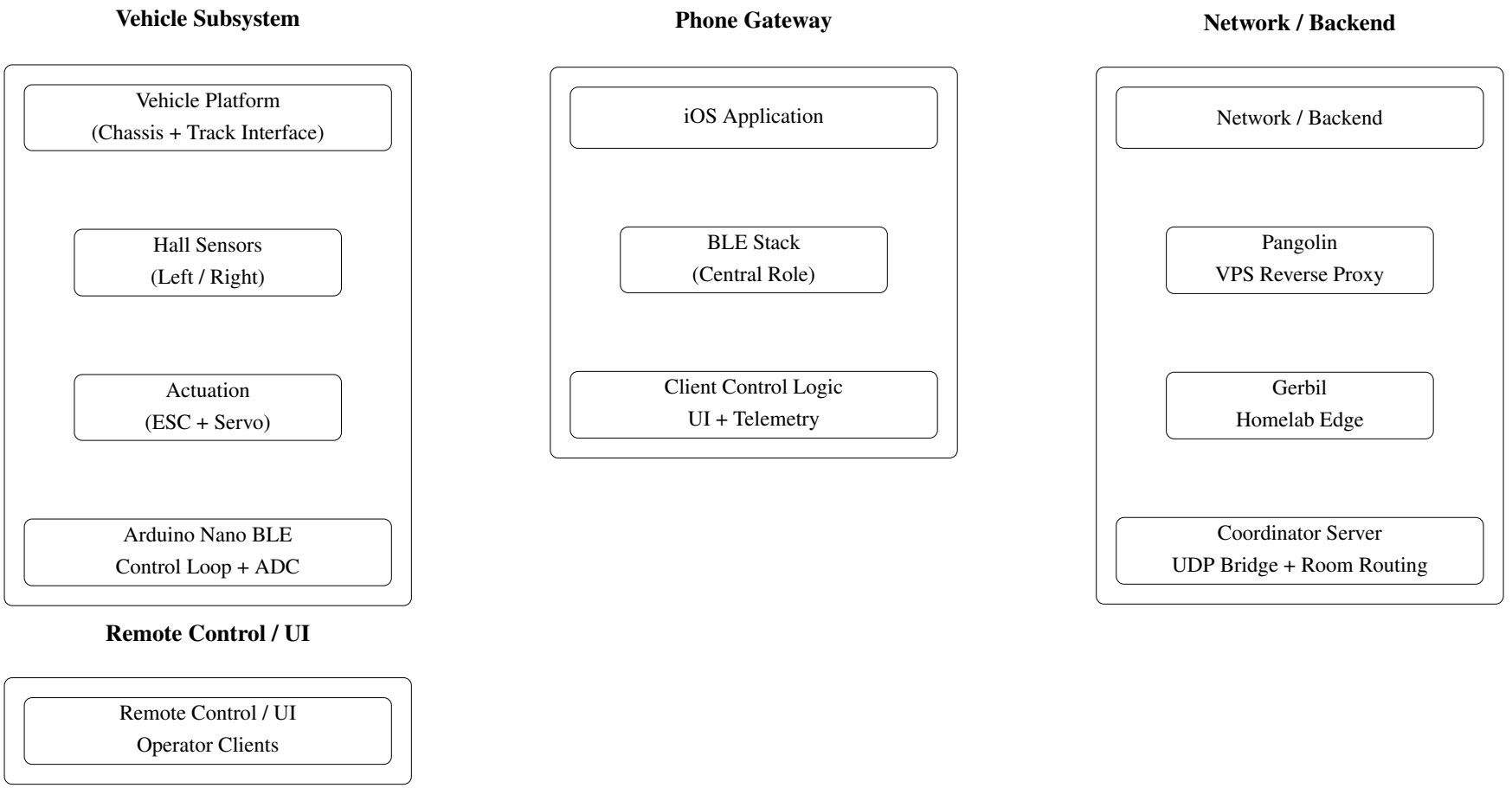


Figure 3: High-level system component diagram.

Hall Effect Sensor Testing

Initial Hall effect sensor characterization was performed on a benchtop using an ESP32-WROOM development module. The sensor was interfaced in an analog configuration, with its output voltage routed directly into the ESP32's analog-to-digital converter (ADC) input pins. This setup allowed the ESP32 to continuously sample the Hall effect sensor's analog output and record raw ADC measurements under controlled conditions, independent of the vehicle platform. This benchtop testing approach closely mirrors the sensing architecture used by the SEED ESP32 platform, ensuring consistency between early characterization results and the final embedded implementation.

During testing, the ESP32 ADC was configured to report sensor data in both 12-bit and 16-bit raw representations. These raw values reflect the quantized digital output of the ADC prior to any filtering, scaling, or thresholding, and were used to evaluate sensor noise, baseline stability, and response behavior in the presence and absence of a magnetic field. Collecting both resolutions enabled direct comparison of effective resolution and quantization noise across ADC configurations.

In the corresponding plots, the blue trace represents the 12-bit raw ADC output, while the red trace represents the 16-bit raw ADC output. The two traces exhibit the same underlying sensor behavior, with differences arising solely from ADC resolution and scaling. This visualization highlights the relative stability of the Hall effect sensor under steady conditions, as well as its dynamic response when exposed to a magnetic strip during controlled approach and withdrawal testing.

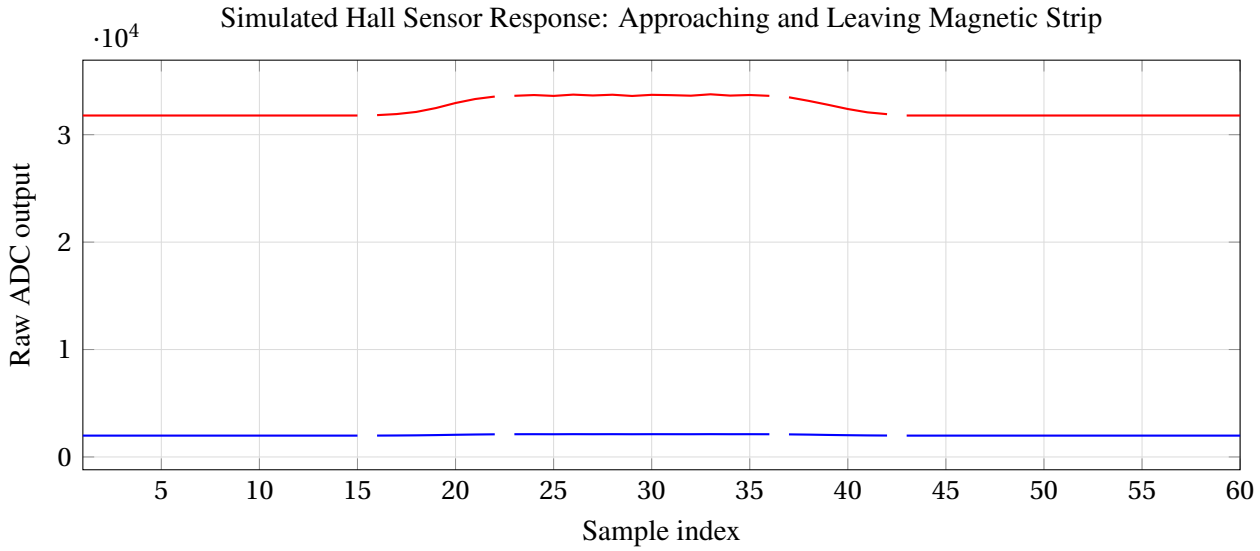


Figure 4: Simulated Hall-effect sensor raw outputs showing baseline readings, a rise as the sensor approaches a magnetic strip, a near-strip plateau, and a return to baseline as the sensor moves away.

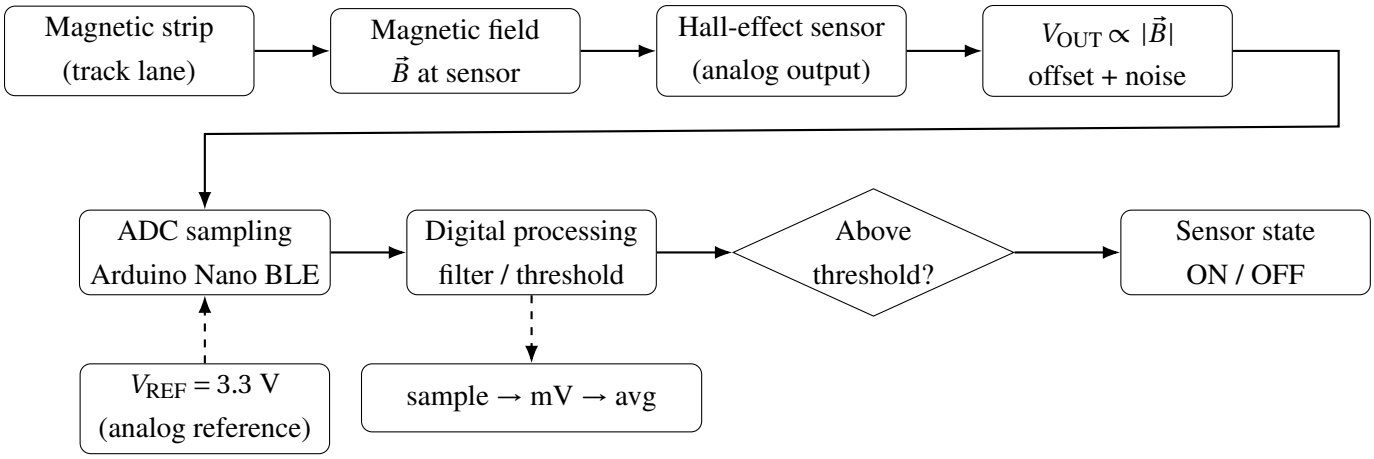


Figure 5: Hall-effect sensing and acquisition chain.

Communication

An iOS application was used as both the primary user interface and a communication bridge between the vehicle and the coordinator server. The application connects to each car over Bluetooth Low Energy (BLE), acting as a central device and interfacing with the car's GATT service through dedicated telemetry and command characteristics. Telemetry data is received via BLE notifications, while control commands are issued using write operations to minimize latency and protocol overhead.

Upon reception, BLE telemetry frames are immediately forwarded by the phone as UDP datagrams to the coordinator server. These packets are routed through a reverse proxy and homelab edge before reaching the internal coordination service. Commands generated by the coordinator follow the reverse path: they are sent as UDP datagrams to the phone application, which then translates them into BLE write operations and delivers them to the appropriate vehicle.

This multi-stage pipeline introduces minimal overhead in practice. Measured end-to-end latency from vehicle to coordinator and back averages approximately 120ms, with negligible packet loss observed during testing. By leveraging the phone as a gateway, the system maintains low-latency, bidirectional communication while keeping the vehicle hardware simple and power-efficient.

Table 3: Minimal Binary Protocol Frame Definitions (Little-Endian)

Frame	Dir.	Type (hex)	Purpose
Telemetry	Client → Server	0x01	Periodic status update including sequencing, device-relative timestamping, event flags, battery estimate, and track position.
Command	Server → Client	0x10	Low-latency control message used to trigger discrete actions (single-byte command with an optional 0/1 value).

Table 4: Command Frame Layout (Type 0x10, Little-Endian)

Byte(s)	Field	Type	Notes
0	type	`UInt8`	Frame discriminator. Fixed value `0x10`.
1	cmd	`UInt8`	Command identifier (enumeration defined by application).
2	val	`UInt8`	Command value. Typically `0` or `1` for boolean actions.

Table 5: Telemetry Frame Layout (Type 0x01, Little-Endian)

Byte(s)	Field	Type	Notes
0	type	`UInt8`	Frame discriminator. Fixed value `0x01`.
1–4	seq	`UInt32 (LE)`	Monotonic sequence counter for loss detection and ordering.
5–8	monotonicMillis	`UInt32 (LE)`	Device-relative timestamp (ms) used for ordering and latency analysis.
9	flags	`UInt8`	Bitfield. `bit0 = powerUpHit`; remaining bits reserved for future events.
10	batteryPct	`UInt8`	Battery estimate as an integer percentage (0–100).
11–12	position	`Int16 (LE)`	Signed position index / lane position encoding (implementation-defined).

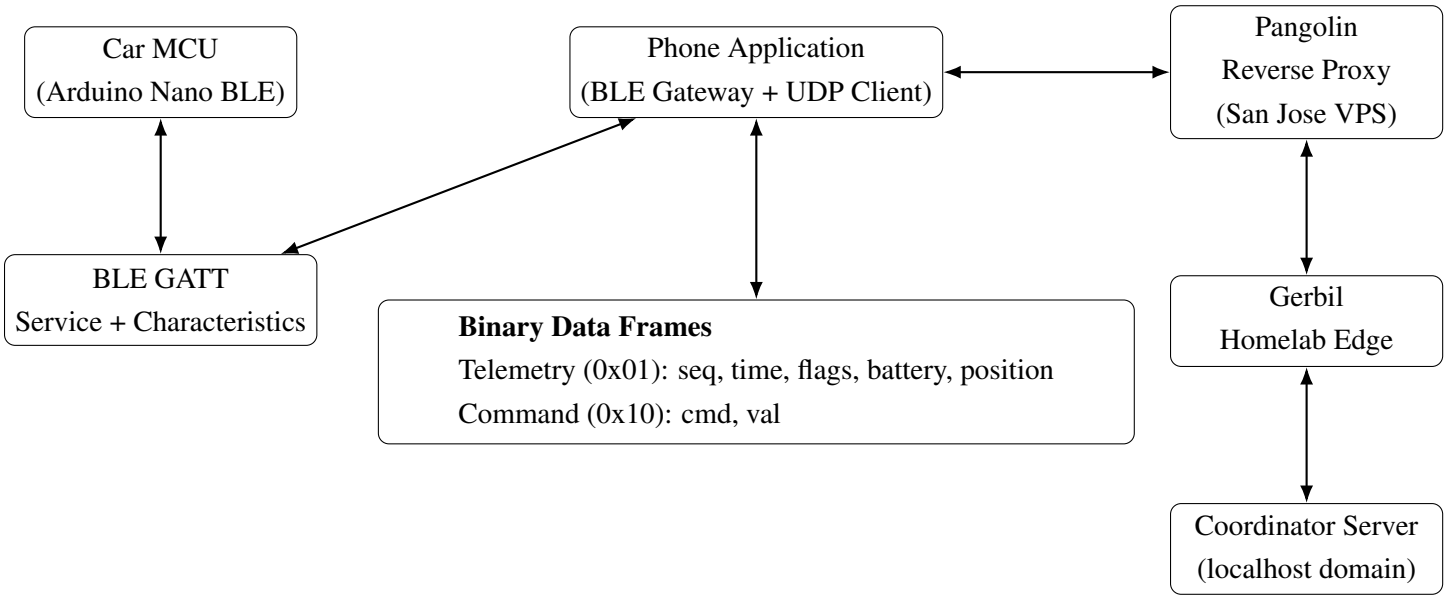


Figure 6: End-to-end communication architecture with expanded vertical spacing. The vehicle communicates with the phone over BLE, while the phone operates as a gateway that forwards telemetry and commands as binary UDP datagrams through a reverse proxy and homelab edge to the internal coordinator server.

Table 6: Coordinator UDP Bridge Server Operation Summary

Function	Behavior in Coordinator Server
Transport / Socket Setup	Creates a UDP socket (`AF_INET`, `SOCK_DGRAM`), binds to a configured `host:port` (default `0.0.0.0:9531`), and enables `SO_REUSEADDR` for robust restarts.
Packet Ingress	Continuously receives UDP datagrams via `recvfrom()`, capturing both the payload and the sender's endpoint (`IP:port`). Malformed packets are dropped early to avoid corrupt room state.
Application Envelope Parsing	Parses an outer "room envelope" format: `[roomLen][role][room bytes][payload...]`. The `role` identifies senders as *Device* (car/phone gateway) or *Operator* (controller client). The remainder is treated as the protocol payload.
Room-Based Multiplexing	Maintains an in-memory room table keyed by room name. Each room stores: (1) a list of **devices** and their last-seen endpoints, and (2) a list of **operators**. Rooms are created on-demand when first referenced.
Endpoint Tracking and Mobility	Updates device endpoints automatically when a device's source `IP:port` changes (e.g., NAT rebinding). Devices may optionally include a stable identifier appended to telemetry/commands; if present, this ID is used to associate the device with the correct room entry even when the network endpoint changes.
Telemetry Forwarding (0x01)	Telemetry payloads are forwarded **only to operators** in the same room. Frames are validated for minimum length prior to forwarding to prevent malformed or truncated telemetry from propagating.
Command Forwarding (0x10)	Command payloads are forwarded from a **device** to all **other devices** in the same room and to all **operators** in the room. Echo suppression is enforced: the server will not send a command back to the originating device (by device ID when present, otherwise by endpoint match).
Global Powerup Broadcast (Slow-Down)	A specific command subtype (slow-down) triggers a cross-room broadcast. When received, the server transmits a minimal command frame `[0x10][0x05][0x01]` to every known device across all rooms, excluding the originator. This enables "global" effects independent of room membership.

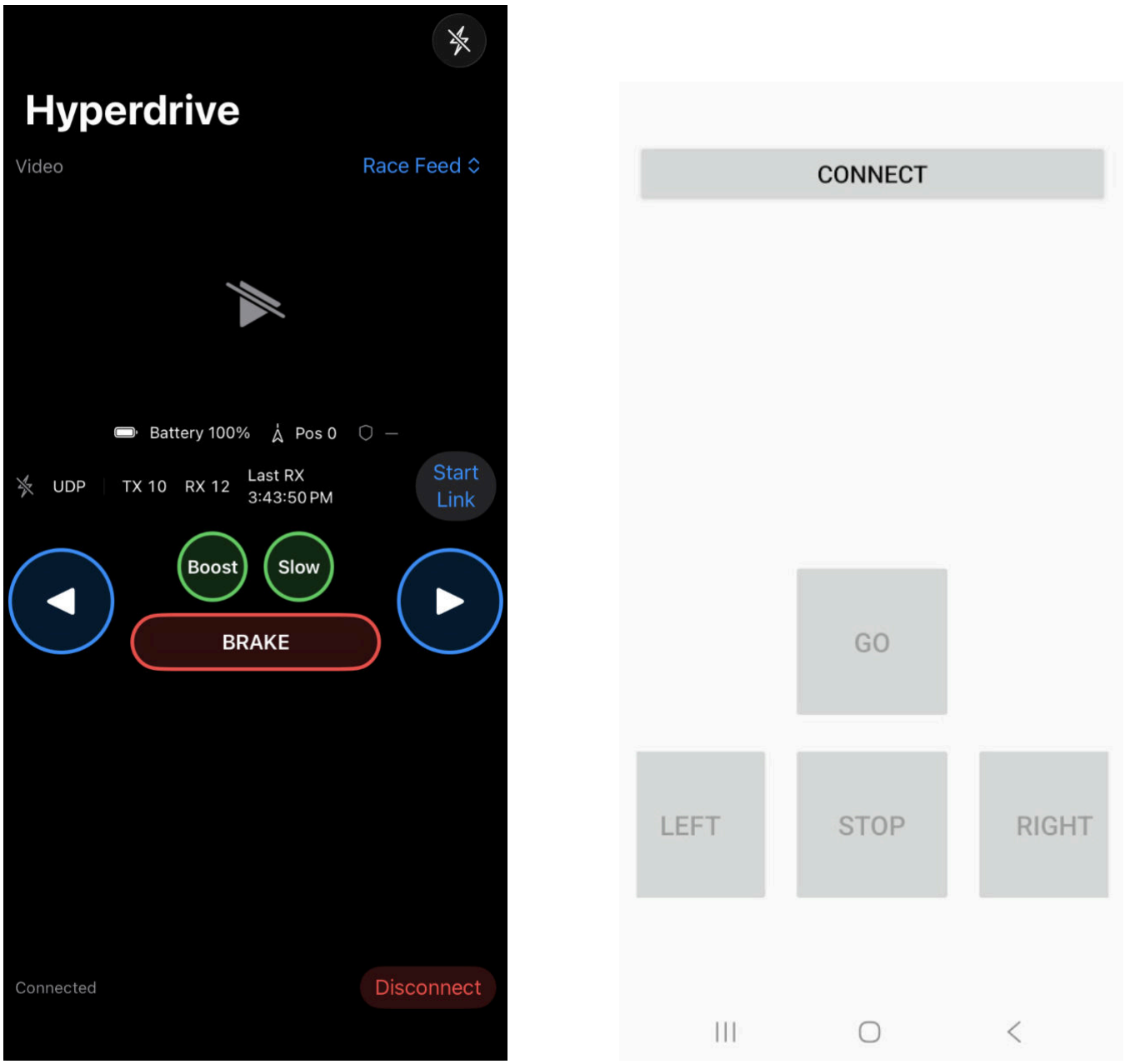


Figure 7: iOS and Android Phone Applications

Two mobile applications were developed to support system operation and testing. An Android application was used primarily during early development and debugging to validate BLE connectivity, command formatting, and basic vehicle control. Its functionality was intentionally limited and focused on rapid iteration during firmware and protocol bring-up.

The iOS application serves as the full-featured production client and primary user interface for the system. In addition to establishing BLE connections to the vehicles and issuing real-time control commands, the iOS app implements the complete networking stack required to interface with the coordinator server. The application supports user-selectable UDP server endpoints, enabling flexible deployment across local, cloud-hosted, or homelab environments. It also integrates WebSocket

connectivity for higher-level coordination and state synchronization, allowing dynamic interaction between vehicles and remote operators.

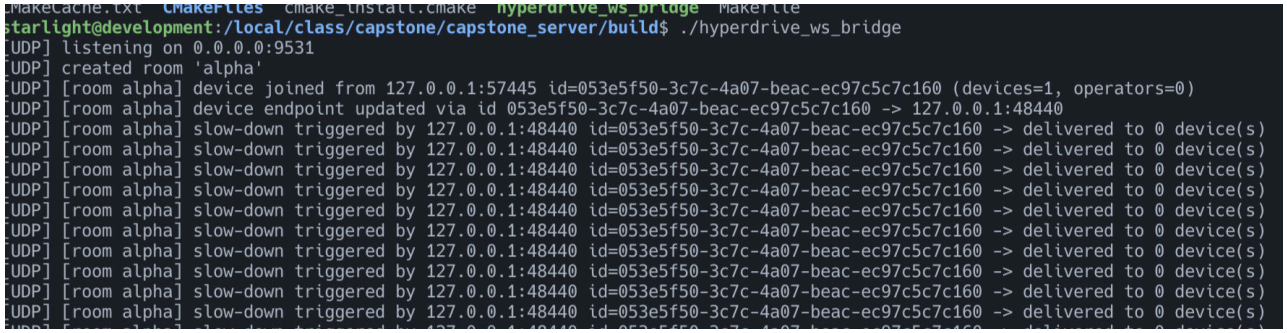


Figure 8: Coordinator Server

The coordinator server acts as the central routing and arbitration component of the system. It receives telemetry and command packets forwarded by the mobile applications over UDP and multiplexes traffic based on logical room identifiers. This design allows multiple vehicles and operators to share a common backend while remaining logically isolated from one another.

By operating entirely over UDP with minimal per-packet processing, the coordinator server provides low-latency, best-effort message delivery suitable for real-time control, while remaining simple, scalable, and easy to deploy across cloud and homelab environments.

Mechanical Design

The mechanical design was based on a free print-in-place remote control car from the website Cults3D. This allowed us to 3D print a miniature car that, after support removal, has rotatable wheels without the need of bearings. The only off-the-shelf mechanical component required was a single screw to control the turning of the front wheels.

Table 7: Benefits of Starting With the 3D Print-In-Place Car Design

Benefit	Explanation
Rapid Iteration	Since the entire design was an STL, we could quickly adjust the space for electrical components.
Easily Reproducible	In case a car broke or for rapid manufacturing production down the line, the design could be easily reproduced.
Focus on Novel Aspects	Allowed us to focus on more novel parts of the project, such as the magnetic hall effect sensing control algorithm.

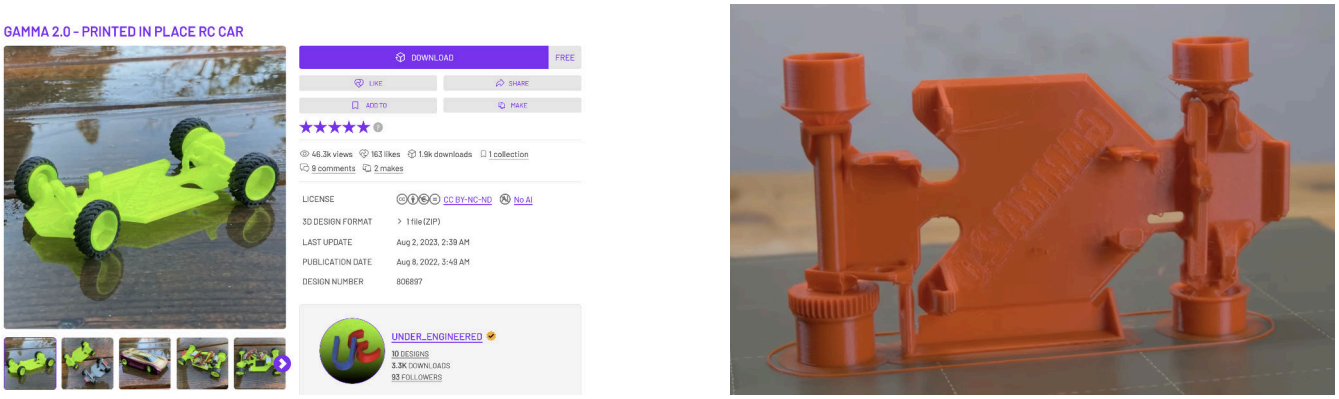


Figure 9: Original 3D Model Our Design is Based On

Modifications to the Original Model

The car design incorporated multiple electrical components and wiring configurations not present in the original model. It was necessary to create space for and design new components, including:

1. A new motor
2. Routing locations and indents for Hall Effect sensors and associated wiring
3. A new pinion gear to interface with the rear axle

4. Shell modifications, battery compartment, and switch location

Challenges in Editing the 3D Model:

Access to the original CAD files, which would have allowed extrusion or cutting of faces (e.g., a .f3d file), was not available. Attempts to contact the model creator were unsuccessful. With only the STL format available, all alterations were performed using TinkerCAD's "combine" feature, enabling addition or subtraction of shapes to modify the model. This required precise measurements of component dimensions. The shell and chassis were printed from PLA, while the tires were printed from TPU to provide improved track friction.

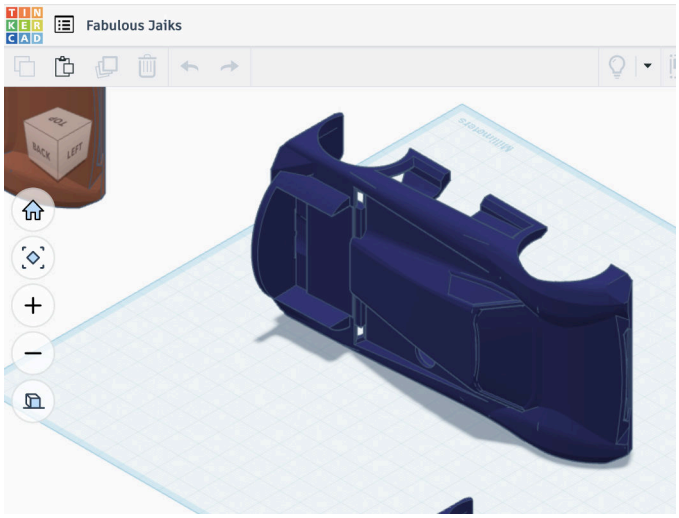


Figure 10: TinkerCAD view showing 3D modeling workflow

Motor Housing

The DC motor was the most critical component. A housing was designed based on the motor datasheet, with a hole precisely sized for the motor shaft. This press-fit design secures the motor without additional mechanical latches or adhesives and avoids interference with the motor's rotating components.

Figure 11: Motor integration: datasheet, 3D model, and actual motor placement

Hall Effect Sensor Integration

Hall Effect sensors were mounted underneath the car chassis toward the front. Two holes were incorporated to route wires from the Arduino BLE module to the sensors. Indents were created to ensure precise spacing and alignment of the sensors during installation.

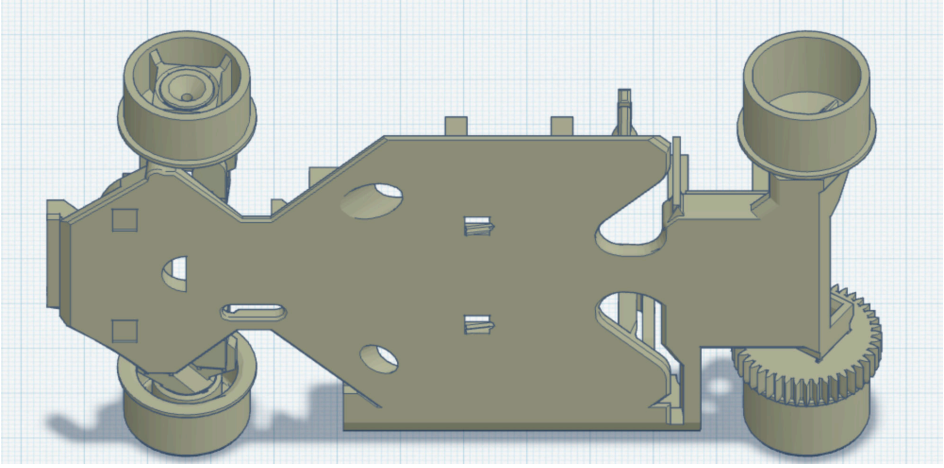


Figure 12: TinkerCAD view of Hall Effect sensor mounting holes and indents

Pinion Gear Adaptation

The new motor shaft size and its position relative to the rear axle required a custom pinion gear. STL limitations prevented exact measurement of rear axle tooth properties, necessitating six iterative gear adjustments to achieve a snug fit and reliable axle engagement without skipping teeth.

Figure 13: Pinion gear: Old (left) vs. New (right)

Shell Modifications

Limited internal space prompted the creation of a battery compartment in the outer shell, facilitating easy battery access. A hole was added to route power cables, and a gap was introduced for the power switch. The wheel well was expanded to accommodate larger tires.

Figure 14: Updated car shell showing battery compartment, switch gap, and expanded wheel well

The resulting car can be fully equipped with all electrical components necessary for our use case and performed reliably after running the car for multiple hours straight.

Figure 15: Final car chassis and shell design

Electrical Design

The electrical architecture of the vehicle was designed around a centralized power distribution model, with the electronic speed controller (ESC) serving as both the primary motor driver and power management interface. System power is supplied by a 7.4 V (2S) lithium-polymer battery, which is directly connected to the ESC. The ESC regulates this input and provides a stabilized 5 V rail via its onboard battery eliminator circuit (BEC). This regulated 5 V output is used to power both the Arduino Nano BLE microcontroller and the steering servo, eliminating the need for a separate external buck converter and reducing overall system complexity and wiring overhead.

Analog sensing is performed using two Hall effect sensors positioned at the front of the chassis for lane detection. These sensors are connected to the Arduino Nano BLE's analog input pins A0 and A4, respectively. To improve measurement stability and ensure consistent thresholding behavior, the Hall effect sensors are referenced to the microcontroller's 3.3 V supply rail. This 3.3 V reference is generated locally by the Arduino Nano BLE's onboard linear voltage regulator, which steps down the incoming 5 V supply for low-voltage peripherals and analog reference consistency.

Motor and steering actuation are controlled via standard pulse-width modulation (PWM) signaling. The ESC receives its throttle control signal on digital pin D10, while the steering servo is driven from digital pin D8.

System-level electrical protection is primarily handled by the ESC, which incorporates internal short-circuit and over-current protection to safeguard the battery, motor, and downstream electronics. By sourcing all regulated power through the ESC's BEC, the design ensures that fault conditions such as motor stalls or wiring shorts result in a controlled shutdown rather than catastrophic component failure. This power hierarchy—battery to ESC, ESC to logic and actuation, and onboard regulation for low-voltage domains—provides a robust and fault-tolerant electrical foundation for the vehicle.

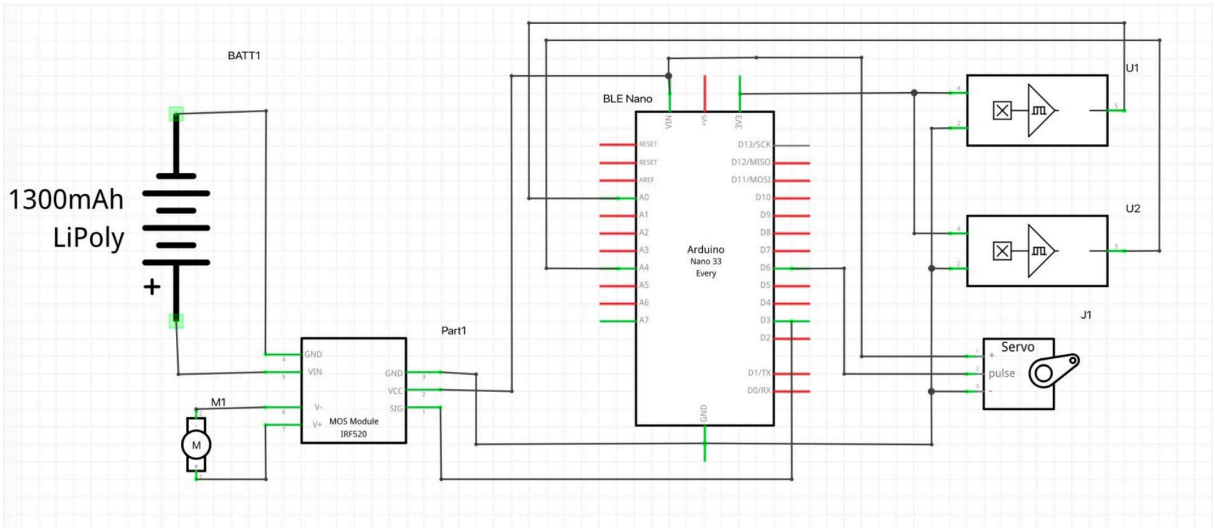


Figure 16: Hall Effect Sensors on Front of Car

Track Design

The track design was guided by several primary objectives:

- A two-lane layout to support lane-following and lane-changing behaviors
- Equidistant lanes arranged in an oval configuration for consistent lap geometry
- Sufficient portability to allow transportation and setup by a small group of people

Table 8: Track Surface Material Evaluation

Material	Evaluation
Rollable plastic tarps	Too flimsy to provide a flat, consistent driving surface.
Rigid plastic slabs	Provided adequate rigidity, but were cost-prohibitive and had long shipping times.
Foam padding	Too compressible, resulting in an uneven and inconsistent surface.
Dry erase whiteboard material (Selected)	Smooth, rigid, relatively inexpensive, and well-suited for magnetic strip placement.

The first step in the track design process was selecting an appropriate material for the track surface. Several candidate materials were evaluated based on rigidity, surface smoothness, cost, and ease of acquisition.

Dry erase whiteboard material proved to be a promising option; however, many commercially available whiteboards were excessively large and heavy, making transportation difficult. To address this, three smaller whiteboard slabs were sourced and arranged to form the complete track. This modular approach allowed the track to be easily transported and assembled in different locations.

Figure 17: Modular whiteboard slabs used to construct the track surface

With the track surface selected, attention shifted to choosing a magnetic strip that was sufficiently strong to be detected by the Hall Effect sensors while remaining flexible enough to conform to the track geometry. A 0.5 in. wide magnetic strip with adhesive backing was selected to meet these requirements.

Figure 18: Adhesive-backed magnetic strip used for lane definition

Initial track layout prototyping revealed that the lane-following algorithm performed more smoothly and accurately when operating over surfaces defined by two parallel magnetic strips. Based on this observation, the track was designed accordingly. For consistency and simplicity in the minimum viable product, the overall layout was chosen to be a classic oval race track.

To ensure even spacing of the magnetic strips, measurements were taken along the edges of the whiteboard slabs and between each lane. The full track layout was traced in pencil prior to strip placement. The magnetic strips were then applied carefully in three separate segments, allowing the track to be disassembled for transport.

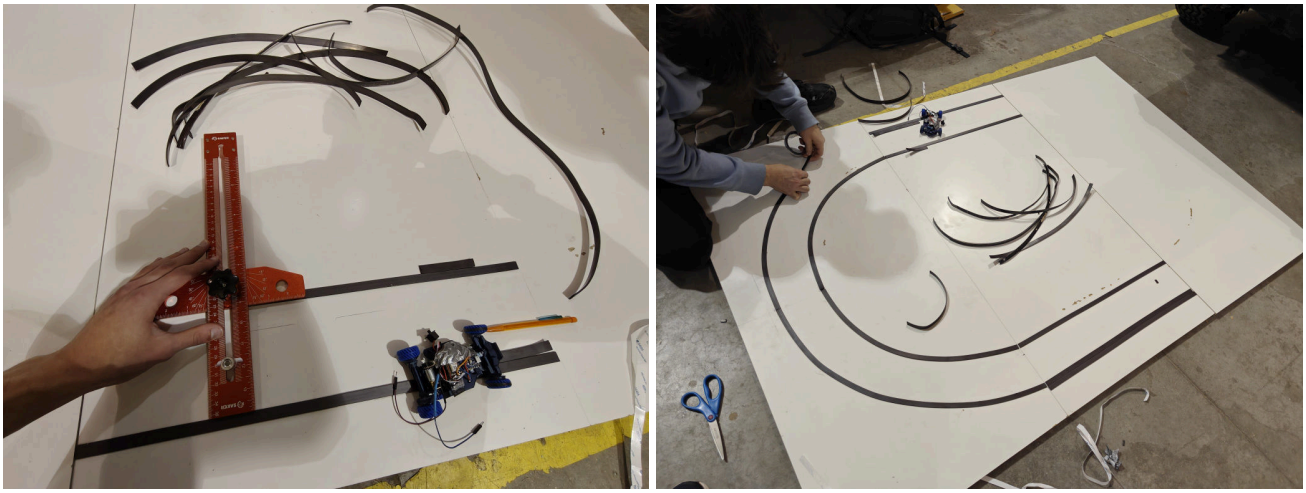


Figure 19: Track layout measurements and magnetic strip placement process

Once the track loop was completed, lane-following and lane-changing algorithms were tested. Despite the use of TPU tires, additional friction was required to ensure reliable lane changes without slipping. To address this, grip tape was applied around the entire perimeter of the track.

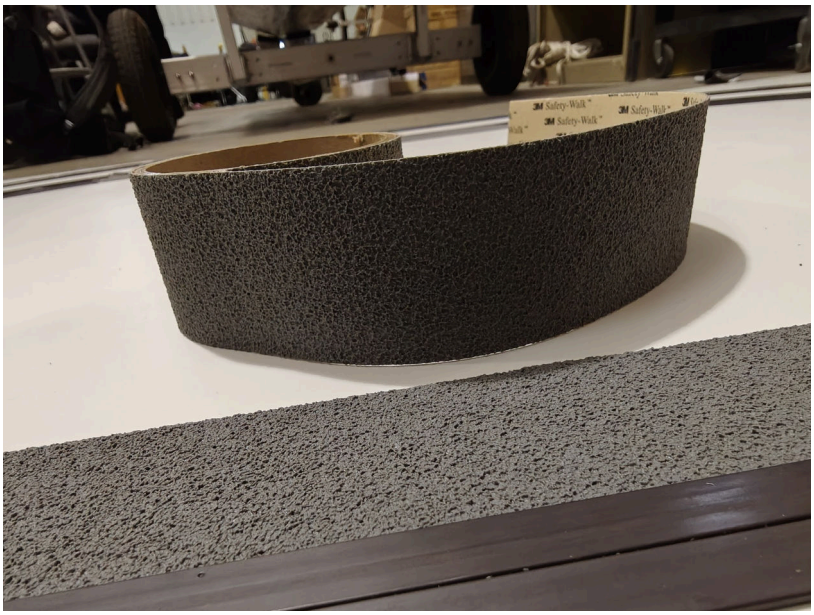


Figure 20: Grip tape applied along the track perimeter to increase friction

Ultimately, the completed track interfaced reliably with the vehicles, and all original design objectives were successfully met.

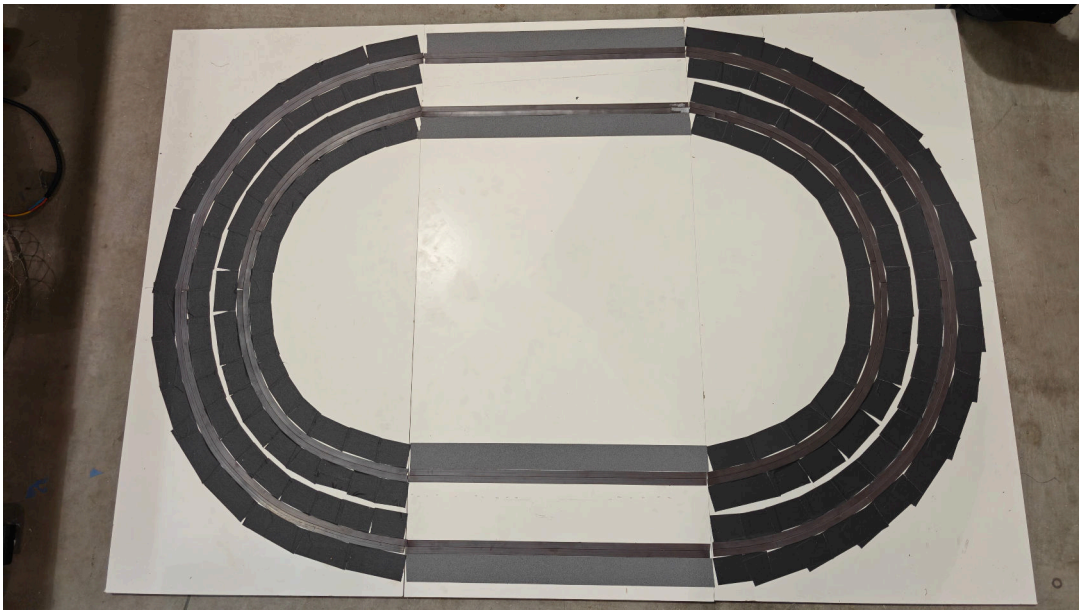


Figure 21: Final assembled track configuration

Control Algorithms

Lane Following

The lane following algorithm for each car needed to satisfy the following requirements:

- Car should smoothly follow lanes on straights, with minimal oscillations
- Car should be able to follow lanes on turns
- Car should be able to complete transitions between straights and turns to complete laps

The final algorithm design relied on two hall effect sensors placed towards the front of the car. Two hall effect sensors were chosen to give the car the ability to track in which direction it had left a lane. Further, based on input from the Anki Overdrive AI Lead, the sensors were placed at the very front of the car to receive data for servo motor adjustments at the earliest point possible.

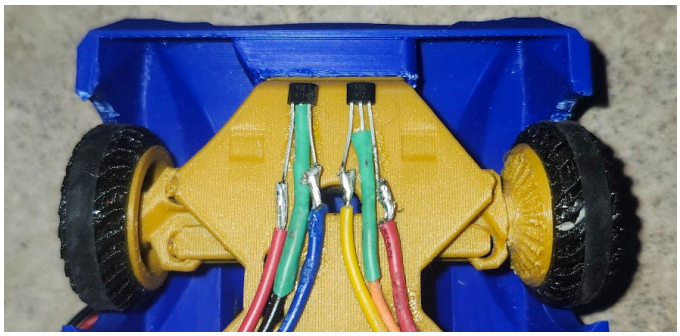


Figure 22: Hall Effect Sensors on Front of Car

The hall effect sensors were designated as left and right for algorithmic purposes. The next step was to determine how to change servo motor movements based on the values read from the hall effect sensors. Because of the variability of the hall effect sensors found during their testing, the design decision was made to treat the hall effect sensors as either ON or OFF depending on a threshold value, instead of making steering adjustments based on exact readings. Each sensor is finetuned with its own base value when it is detecting nothing, then the difference between this value and its current one is measured for each loop.

Listing 1: Hall-effect sensor baseline comparison

```
// Set base values for left and right hall effect sensors
int baseL = 76;
int baseR = 73;

// Determine difference between current sensor values and base values
int Ldif = abs(baseL - rawL_avg);
int Rdif = abs(baseR - rawR_avg);
```

Further, decisions for whether a sensor is ON or OFF were based on whether the average of the last 7 readings for each sensor are below or above the designated threshold. Based on the ON and OFF states of the sensors, three states were introduced to the car. The car could either be CENTERED, OFF_RIGHT, or OFF_LEFT.

The actions based on sensor readings and state are summarized in the table below:

Table 9: Lane Following Sensor States and Actions

Left Sensor	Right Sensor	State	Action
On	On	CENTERED	Go straight
On	Off	OFF_RIGHT	Turn left back to lane
Off	On	OFF_LEFT	Turn right back to lane
Off	Off	OFF_LEFT/OFF_RIGHT	Retain previous off state and turn until lane is found

This allowed the system to move smoothly and account for any dramatic variations in the hall effect sensors. Further, because our cars only travel counter-clockwise around the track, steering PWM values sent to the servo were fine tuned to match the turning radius of the track when turning left, and reduced heavily on right turns to further reduce oscillation.

The result of the algorithm was that the cars can consistently follow both the inner and outer lanes, with smooth transitions between turns and straights, and minimal oscillations throughout the track.

Videos of the testing processes and final demo can be found here.

Lane Changing

The lane changing routines needed to satisfy the following requirements:

- Car should be able to change lanes at any point in the track
- Lane changes should be triggered by BLE commands using app on phone
- Safety systems should be implemented to make sure user cannot initiate lane changes into lanes that do not exists

The lane changing routines represent another core component of our MVP. They were built on top of the control loop and are activated based on commands from the phone app over BLE. The first part of the design was to ensure that only "possible" lane changes could occur. For example, if the car is in the inner lane, it should not be able to turn left into nothing, and when it is in the outer lane, it should not be able to turn right and go off the track. Safety checks in the code were implemented by introducing two new states to the CAR, LANE_1 and LANE_2. A summary of the safety checks is shown in the table below:

Table 10: Lane Changing Safety Checks

Lane State	Check
LANE_1 (inner lane)	Do not allow lane changes to the left
LANE_2 (outer lane)	Do not allow lane changes to the right

Following confirmation that a requested lane change is possible, the car will enter into a routine for either changing lanes to the right, or changing lanes to the left. Each respective lane change follows these steps:

Table 11: Lane Change Procedure Steps

Step	Action
1	Interrupt normal control loop
2	Write PWM signals to servo to turn wheels in desired direction
3	Write PWM signals to motor controller to speed car up to get it off the original lane
4	Delay for a fine-tuned number of milliseconds to give the car time climb over the magnetic strip in its original lane
5	Check if any hall effect sensors are detecting a magnetic strip to make sure it left original lane
6	Turn the wheels straight once car leaves original lane
7	Wait for hall effect sensor on the side of the turn direction to detect a magnetic strip
8	Write PWM signal to servo to turn sharply in the opposite direction
9	Update car states and hand control back to normal loop

The decision to add a delay in step 4 was made because of the variability of the hall effect sensors. During testing, once the car turned in a direction to make a lane change, there were cases where at the edge of a magnetic strip the hall effect sensors would fluctuate between their ON and OFF thresholds, causing the car to prematurely believe it had left its original lane. The delay was fine-tuned for each lane change direction.

An example of the lane changing routine to turn to the left is provided below:

Listing 2: Lane change to the left routine (BLE-triggered) with Hall-effect validation

```
if (lane_left)
{
    // === 1) Initiate lane change: bias steering left + add throttle ===
    // Goal: force the vehicle to depart the current lane boundary quickly and commit to the transition.
    steer.writeMicroseconds(1100);     // hard left steering command (tuned PWM)
    esc.writeMicroseconds(2000);       // temporary motor boost (tuned PWM)
    delay(550);                        // allow time to cross lane boundary (tuned)

    // === 2) Confirm we have left the original lane ===
    // While either sensor still indicates strong presence of the original magnetic strip,
    // keep sampling. Exiting the loop means the car is no longer detecting the original lane.
    while (Ldif >= HALL_THRESHOLD || Rdif >= HALL_THRESHOLD)
    {
        // Sample both Hall sensors (raw ADC counts)
        rawL_instance = analogRead(SENSOR_LEFT_PIN);
        rawR_instance = analogRead(SENSOR_RIGHT_PIN);
```

```
17
18          // Smooth readings (rolling average) to reduce sensor noise / spikes
19          rawL_avg = getRollingAverage(rawL_instance, leftReadings);
20          rawR_avg = getRollingAverage(rawR_instance, rightReadings);
21
22          // Difference-from-baseline metric used for ON/OFF thresholding
23          // Larger difference implies "over magnetic strip"
24          Ldif = abs(baseL - rawL_avg);
25          Rdif = abs(baseR - rawR_avg);
26      }
27
28      // === 3) Drive straight to traverse the gap between lanes ===
29      steer.writeMicroseconds(SERVO_CENTER_US);
30
31      // === 4) Acquire the next lane (expected on the right sensor for this maneuver) ===
32      // Wait until the lane marker is detected by the sensor on the side we are transitioning toward.
33      // Loop exits when the new magnetic strip is detected (Rdif crosses threshold).
34      while (Rdif < HALL_THRESHOLD)
35      {
36          rawL_instance = analogRead(SENSOR_LEFT_PIN);
37          rawR_instance = analogRead(SENSOR_RIGHT_PIN);
38
39          rawL_avg = getRollingAverage(rawL_instance, leftReadings);
40          rawR_avg = getRollingAverage(rawR_instance, rightReadings);
41
42          Ldif = abs(baseL - rawL_avg);
43          Rdif = abs(baseR - rawR_avg);
44      }
45
46      // === 5) Re-center onto the new lane ===
47      // Snap steering back in the opposite direction to settle the vehicle over the new strip.
48      steer.writeMicroseconds(2400);     // sharp corrective steer back right (tuned)
49
50      // Restore nominal forward speed for normal lane-following control
51      esc.writeMicroseconds(ESC_FORWARD_US);
52
53      // === 6) Update state for the control loop ===
54      // After a left lane change, the vehicle should be biased as OFF_LEFT initially,
55      // and internal lane bookkeeping is updated for future safety checks.
56      car_state = OFF_LEFT;
57      lane_left = false;                // clear command flag so routine doesn't retrigger
58      car_lane  = LANE1;                // update lane state (inner lane after left change)
59
60      // === 7) Hand control back to normal lane-following loop ===
61      delay(CONTROL_PERIOD_MS);         // prevent immediate re-entry; aligns with main loop period
62      return;
63  }
```

Following fine-tuning of steering values, delay, and motor speed, lane changes were able to work semi-consistently. See the videos here for testing and a final version.

High-Level System State Diagram

Vehicle control states and end-to-end communication flow

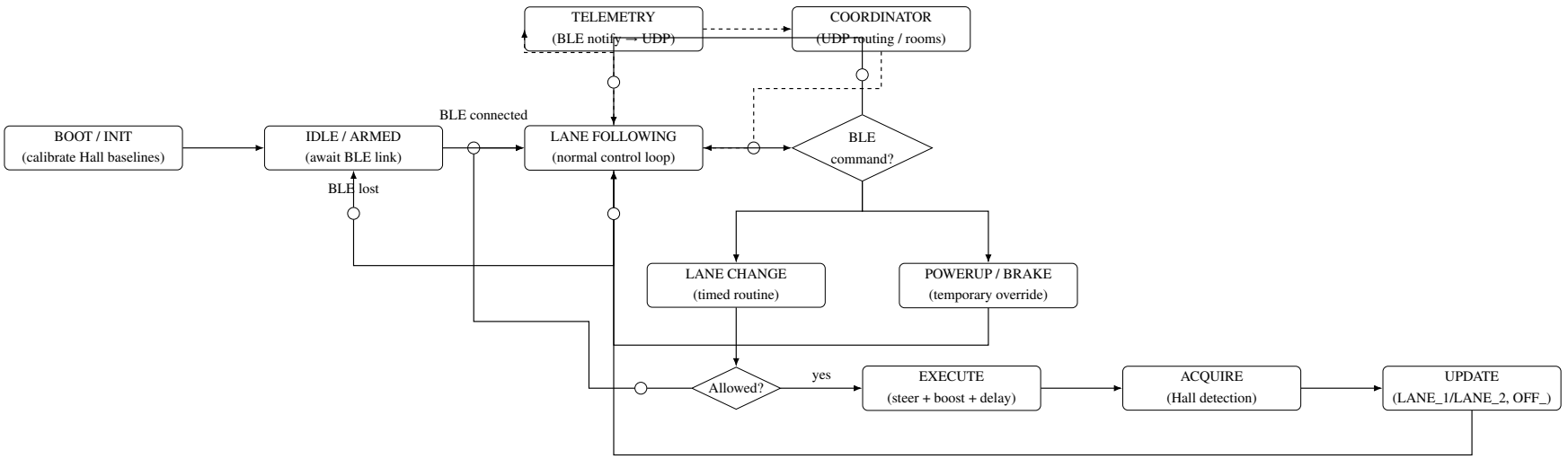


Figure 23: High-level system state diagram.

Use of Professional Standards

Overview

This project was designed and evaluated with reference to established professional standards covering electrical safety, wireless communication, data security, and mechanical integrity. Compliance was achieved through a combination of certified components, manufacturer documentation, and system-level verification testing.

Applicable Standards and Compliance

Standards Verification Approach

Verification Methods

- **Battery and power safety (IEEE 1725):** Verified using manufacturer documentation and power-system behavior during testing.
- **PCB and material standards (UL 94, RoHS):** Confirmed through supplier datasheets and component certifications.
- **Wireless standards (IEEE 802.11, BLE, FCC Part 15):** Validated through functional testing, interoperability checks, and use of pre-certified RF modules.
- **Encryption and data protection (AES-128, TLS):** Verified through packet inspection and encrypted session validation during system integration.
- **Mechanical and material standards (ISO 527, ISO 10993):** Confirmed through supplier documentation and in-house mechanical testing.

Summary: Adherence to established professional standards ensured that the system met safety, reliability, and interoperability expectations appropriate for a consumer-facing electromechanical platform. Leveraging certified components and validated protocols reduced risk while enabling rapid iteration and integration.

Table 12: Applicable Professional Standards

Category	Standard	Application in Project
Battery Safety	IEEE 1725	Rechargeable battery integration follows IEEE 1725 guidelines, including protection against reverse polarity, over-current conditions, improper charging, and long-term storage degradation.
PCB & Materials	UL 94	Plastic enclosures and PCB substrates meet a minimum UL 94 V-1 flammability rating based on manufacturer specifications.
Environmental Compliance	RoHS Directive 2011/65/EU	All electronic components are RoHS-compliant, and lead-free solder was used during assembly and testing.
Wireless Networking	IEEE 802.11 a/g/n	Wireless modules used for development and testing conform to IEEE 802.11 standards, ensuring compatibility with common Wi-Fi infrastructure.
Short-Range Communication	Bluetooth Low Energy (v5.x)	Bluetooth Low Energy is used for phone-to-car communication with standardized UUIDs, efficient advertising, and low-power operation.
Radio Emissions	FCC Part 15	All RF modules are FCC-certified consumer devices. No custom RF front-end design was performed, ensuring compliance with U.S. emission limits.
Data Security	AES-128 / TLS 1.2–1.3 / FIPS 197	All system communications are encrypted using AES-128 with TLS. Encryption behavior was verified via packet inspection during integration testing.
Mechanical Integrity	ISO 527	Structural components were designed for repeated mechanical stress. PLA and ABS parts were evaluated through tensile and impact testing.
Material Safety	ISO 10993	All 3D-printed and plastic components are sourced from manufacturers meeting safety requirements for human handling and prolonged contact.

Table 13: WiFi vs. Bluetooth (BLE) for Phone–Car Communication

Category	WiFi	Bluetooth Low Energy (BLE)
Primary use case	High throughput networking; well suited for bulk data such as video streaming.	Low-overhead point-to-point communication optimized for control and telemetry.
Latency behavior	Variable latency due to contention, re-association, and network management, which can negatively impact real-time control.	Lower and more predictable latency for small packets, producing smoother vehicle response.
Setup and configuration	Requires SSID management, IP addressing, and network discovery, increasing system complexity.	Simple pairing and GATT-based communication with minimal configuration.
Robustness during iteration	Sensitive to environmental factors and network state; more failure modes during resets and testing.	Fast reconnect and fewer external dependencies, improving reliability during development.
Power and firmware cost	Higher power consumption and larger software stack on the vehicle.	Lower power usage and smaller firmware footprint, better suited for embedded control.
Design tradeoff	Offers excess bandwidth that is not directly useful for low-rate control data.	Trades bandwidth for timing predictability, which is more critical for responsive car control.

References

[1] IEEE 1725 – Rechargeable Batteries for Portable Devices

[2] UL 94 – Flammability of Plastic Materials

[3] RoHS Directive 2011/65/EU

[4] IEEE 802.11 – Wireless LAN Standards

[5] Bluetooth Core Specification v5.x

[6] FCC Part 15 – Radio Frequency Devices

[7] NIST FIPS 197 – Advanced Encryption Standard

[8] ISO 527 – Plastics Tensile Testing

[9] ISO 10993 – Biological Evaluation of Materials

Project Outcomes

Overview

The final outcome of this project was a functional autonomous miniature car racing system designed to operate on a magnetic track. The completed system consists of 2 3D-printed vehicles integrated with onboard electronics and control software, paired with a modular two-lane track that enables reliable lane following and lane changing behavior. We'll go into further detail below.

Communication Outcomes

- Implemented reliable end-to-end bidirectional communication between the vehicle (BLE), the mobile gateway (phone), and the coordinator server (see Tables 3–5 and Table 6; Figures 3 and 6).
- Developed a multiplexing coordinator server capable of routing telemetry and commands across multiple devices using room-based isolation (see Table 6; Figure 8).
- Verified successful transmission of telemetry frames from vehicles to the coordinator (via phone gateway) and successful return-path delivery of commands to the correct vehicle (see Tables 3–5; Figure 6).
- Demonstrated stable operation under repeated connect/disconnect events, including endpoint updates due to NAT rebinding and mobile network changes (see Table 6).

Electrical & Control Outcomes

- Built two fully functioning vehicles with integrated power distribution (battery → ESC/BEC → logic and actuation), sensing, and actuation subsystems. See Hyperdrive Capstone Project Website.
- Implemented and tuned a lane-following control algorithm using dual Hall-effect sensors, achieving consistent lap completion with minimal oscillation (see Table 10 and Table 2; Figures 4 and 5).
- Implemented a lane-changing routine triggered over BLE, including safety checks to prevent invalid lane changes (inner/outer lane constraints) (see Table 10; Figure 23). (see video evidence here (Also see video evidence here)
- Integrated all electrical components with stable wiring, safe power routing, and repeatable assembly for extended testing sessions.

Software Outcomes

- Developed a iOS application that serves as the primary operator interface and gateway between BLE vehicles and the coordinator backend (UDP/WebSocket networking) (see Tables 3–5 and Table 6; Figures 6 and 3).

- Developed an kotlin based android application for testing BLE connection and sending commands to the cars (Start this video at :14 for example of app controlling car over BLE).
- Implemented a lightweight binary messaging protocol (telemetry and command frames) with sequencing and timestamping support for loss detection and latency analysis (see Tables 3–5).
- Built a room-based coordinator server to multiplex multiple vehicles and operators, including endpoint tracking to tolerate NAT rebinding and client mobility (see Table 6; Figure 8).
- Implemented control-state handling in firmware (armed/idle, lane-following, lane-change routines, and override commands) to ensure predictable behavior under asynchronous BLE commands (see Figure 23).
- Added safety-oriented control constraints in software, including lane validity checks (preventing impossible lane changes). Also see Table 10 for further explanation.
- Created reusable tooling and test workflows for validating BLE connectivity, packet formatting, and end-to-end telemetry/command round trips during development. See Table 10 for further explanation. Example code for preventing illegal lane changes even if app button is pressed to go left:

```
    void onGoLeftWritten(BLEDevice central, BLECharacteristic characteristic)
    {
        // Only allow car to turn left if it is in LANE2
        // to prevent car turning off track
        if (car_lane == LANE2)
        {
        lane_left = true;
        }
    }

```

- Created reusable tooling and test workflows for validating BLE connectivity, packet formatting, and end-to-end telemetry/command round trips during development.

Mechanical Outcomes

- Produced a 3D-printed vehicle platform based on a print-in-place chassis, enabling rapid iteration while minimizing external hardware requirements (see Table 7 and Figure 9).
- Designed and integrated mechanical accommodations for electronics, including sensor mounting features, wire-routing channels, and a battery compartment with switch access (see Figures 12 and 15).
- Designed and iterated a custom pinion gear to match the selected motor shaft and rear axle interface, achieving reliable drivetrain engagement without tooth skipping (see Figure 13).
- Integrated Hall-effect sensors into the chassis with repeatable positioning and alignment to ensure consistent lane detection across vehicles (see Figure 12).

- Validated mechanical durability through multi-hour testing sessions, confirming stable component retention and consistent steering and drivetrain operation (see Figure 15).
- Designed and constructed a symmetrical two-lane oval track with equidistant lane spacing, providing consistent geometry for reliable lane-following and lane-changing behavior(see Figure 21).
- Implemented a modular, rigid track surface that enabled reliable and consistent lane changes across testing sessions(see Figure 17).

Unexpected Events

This section summarizes major setbacks encountered during system development, integration, and testing. Events are grouped by subsystem domain and paired directly with the corrective actions or design revisions that were implemented.

Electronics & Power Systems

Table 14: Unexpected Events — Electronics & Power

ID	Component	Setback / Challenge	Revision / Mitigation
UE-E1	SEEED ESP32-CAM	ESP32 was shorted due to voltage regulation issues; buck converter logic and line voltage caused device failure	Replaced ESP32 with Arduino Nano BLE; eliminated WiFi requirement; routed all control through phone-based BLE link
UE-E2	Buck Converter / Power Rail	Buck converter could not supply sufficient current for system operation	Drew regulated power directly from ESC output rail; verified compatibility with Arduino supply limits
UE-E3	Power Connections	Power and ground pins frequently shorted due to direct soldering	Switched to auxiliary power (PWM) and auxiliary ground pins to reduce risk of shorts

Communication & Networking

Table 15: Unexpected Events — Communication & Networking

ID	Subsystem	Setback / Challenge	Revision / Mitigation
UE-C1	WiFi / Video	ESP32 radio could not multiplex telemetry and video streams fast enough	Routed all telemetry through BLE; video codec evaluated but abandoned due to Bluetooth bandwidth limits
UE-C2	Raspberry Pi Networking	Raspberry Pi could not obtain a static IP address	Transitioned to Starlight server infrastructure hosted on a private homelab
UE-C3	WiFi / Bluetooth Stack	WiFi and Bluetooth could not effectively share the same radio, even at low video bitrates	Used phone as communication proxy; transmitted raw UDP datagrams through cellular link
UE-C4	Coordination Server	Vehicles required dynamic identification for global broadcast coordination	Implemented dynamic vehicle ID assignment; coordinator uses localhost addressing with encrypted reverse proxy

Mechanical Design & Fabrication

Table 16: Unexpected Events — Mechanical Design

ID	Subsystem	Setback / Challenge	Revision / Mitigation
UE-M1	Gear Train	Original CAD files unavailable; exact gear dimensions unknown	Used TinkerCAD to edit STL files; redesigned gears iteratively through trial-and-error
UE-M2	Wheel Wells	Larger rubber tires caused interference with chassis wheel wells	Expanded wheel well geometry to accommodate increased tire diameter
UE-M3	Tire Traction	Hyperdrive car tires lacked sufficient grip on metal track surface	Added rubber bands and other friction-enhancing materials to increase torque transfer

Track Design & Physical Environment

Table 17: Unexpected Events — Track & Environment

ID	Subsystem	Setback / Challenge	Revision / Mitigation
UE-T1	Track Assembly	Single-piece track was difficult to transport and align	Split track into three modular sections assembled during setup
UE-T2	Track Symmetry	Maintaining symmetry during magnet placement was difficult	Traced full track layout prior to magnet installation to preserve precise alignment
UE-T3	Track Surface	Whiteboard surface provided insufficient traction	Added grip tape along track edges to improve tire adhesion

Control & Autonomy

Table 18: Unexpected Events — Control & Autonomy

ID	Subsystem	Setback / Challenge	Revision / Mitigation
UE-A1	Lane Following	Vehicle oscillated continuously on straight segments	Increased lane width from 0.5 inches to 1 inch to increase tolerance
UE-A2	Lane Change Logic	Lane change routine triggered but vehicle failed to exit lane due to sensor variance	Added delay logic to allow sufficient time for vehicle to depart lane before re-evaluating state

Summary: These unexpected events informed several key architectural and design decisions, including simplifying wireless communications, modularizing physical infrastructure, and increasing tolerance in control algorithms. Each setback directly resulted in a system revision that improved robustness, reliability, or ease of deployment.

Lessons Learned and Future Improvements

Lessons Learned

Key Lessons from Design, Integration, and Testing

- **Allocate significant time for algorithm and parameter tuning.** Control algorithms that interact with real-world systems (traction, sensors, magnetic fields) require extensive empirical tuning. Small parameter changes often produced large behavioral differences, especially for lane following and lane changing.
- **Mechanical design should complement control algorithms.** Adjustments such as increasing lane width, improving tire traction, and modifying track surfaces resulted in more stable behavior than attempting to compensate purely through software.
- **Early and continuous testing is critical.** Incremental testing of individual subsystems (electronics, sensors, control logic) significantly reduced debugging complexity compared to full-system testing late in development.
- **Simpler architectures are often more robust.** Eliminating unnecessary communication paths (e.g., removing WiFi multiplexing in favor of BLE) improved reliability and reduced failure modes under real-world constraints.
- **Formal trade studies improve decision-making.** Structured comparison of design alternatives early in the project would have reduced rework, particularly for communication protocols, power distribution, and sensing strategies.
- **Seek external expertise early.** Discussions with experienced engineers provided valuable validation and design guidance, helping avoid unproductive solution paths and reinforcing effective approaches.
- **Expect variability in low-cost sensors and consumer hardware.** Hall effect sensors, motors, and radio modules exhibited significant variance, requiring tolerance-based logic and defensive design rather than idealized assumptions.

Environmental Constraints & Track Dynamics

Environmental Characterization and System-Level Effects

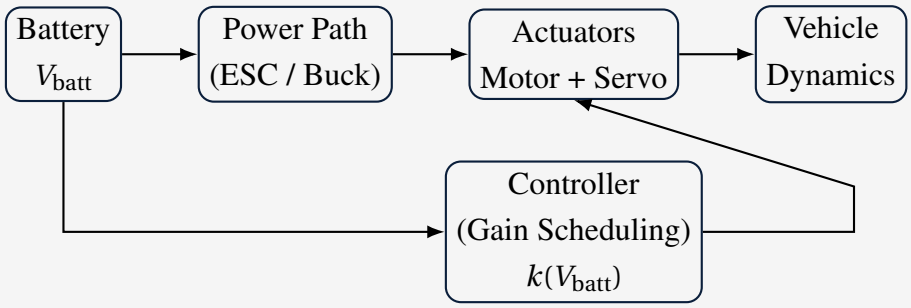
A key lesson from full-system testing was that the vehicle's behavior was strongly influenced by environmental and track-level factors that were not fully captured during early bench testing. While the control algorithms functioned as designed in isolation, real-world deployment revealed several coupled effects between surface properties, power delivery, and vehicle dynamics.

Uneven Surface and Carpeted Substrate Effects

- The track was frequently deployed on uneven flooring and non-uniform carpeted surfaces, introducing small but persistent variations in ride height and wheel loading.
- These variations altered both tire traction and the effective distance between the Hall effect sensors and the embedded magnetic tape, resulting in intermittent sensor dropouts and asymmetric lane-following behavior.
- Local surface compliance (e.g., carpet pile compression) caused transient yaw disturbances, which manifested as oscillations that could not be fully eliminated through controller tuning alone.

Battery State-of-Charge and Power Delivery Variability

- Testing revealed that vehicle behavior differed noticeably between a fully charged battery and a partially depleted battery.
- Higher initial battery voltage increased motor torque and steering aggressiveness, effectively changing the system dynamics assumed during controller tuning.
- As the battery discharged, the system gradually transitioned into a more stable—but slower—operating regime, indicating that the control parameters were implicitly voltage-dependent.



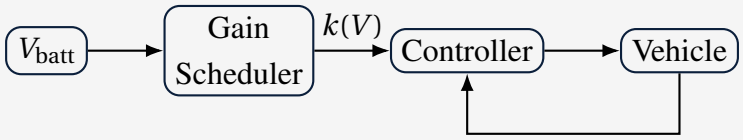
$V_{\text{batt}} \uparrow \Rightarrow$ torque/steering authority $\uparrow$

Lessons Learned

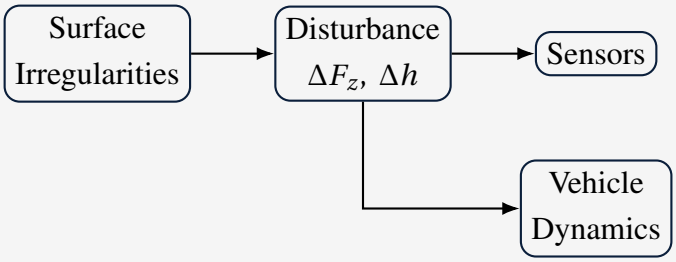
These effects highlighted that the system was operating as a tightly coupled electro-mechanical-control system rather than a purely algorithmic one. Importantly, these behaviors were not random failures but repeatable phenomena that could be observed, characterized, and explained through testing.

Future Mitigations (Conceptual Sketch)

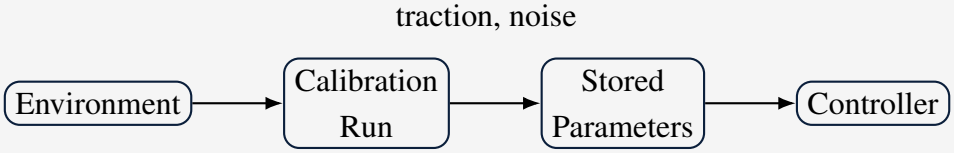
- Introduce battery-voltage-aware control scaling to normalize motor and steering response across the discharge curve.



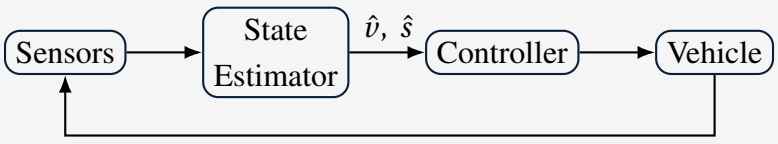
- Add simple compliance or suspension elements, or enforce stricter track flatness requirements, to reduce sensitivity to surface irregularities.



- Perform environment-specific calibration passes to adapt control gains based on measured traction and sensor signal quality.



- Incorporate basic state estimation (e.g., speed or slip inference) to detect and compensate for transient disturbances caused by surface variation.



Future Improvements

Planned Enhancements and Extensions

Control Algorithms and Detection

- Further refine lap detection logic to improve accuracy and reduce false positives.
- Improve robustness of lane change routines under sensor noise and magnetic variability.
- Introduce adaptive or self-calibrating control parameters based on track conditions.

Testing and Validation

- Develop standardized test procedures for electronics, sensors, and control algorithms.
- Automate logging and replay of test data to accelerate tuning and regression testing.
- Perform extended endurance testing to characterize long-term reliability.

Gameplay and User Experience

- Add additional game modes such as round-based races, time trials, and elimination formats.
- Support different classes of cars with distinct performance characteristics.
- Expand power-up mechanics to increase strategy and replayability.

Visual and Audio Feedback

- Integrate LED lighting for status indication and in-game effects.
- Add onboard speakers for audio cues and event feedback.

Track and Physical System Improvements

- Embed magnetic strips directly into the track for improved durability and repeatability.
- Design modular track sections to support multiple layouts and race configurations.
- Improve track materials to increase traction and consistency across environments.

Summary: This project highlighted the importance of co-design between mechanical systems, electronics, communication architecture, and control algorithms. Many challenges were resolved not by increasing system complexity, but by simplifying designs, improving testing discipline, and iterating based on real-world behavior. The identified future improvements provide a clear path toward a more robust, scalable, and engaging system.

Project Performance - On Time

This section compares the **proposed schedule** to the **actual schedule**. Schedule differences are summarized as milestone slips (in weeks) to highlight where time was gained or lost.

Table 19: Proposed vs Actual Schedule (Milestone Slip Matrix)

Milestone	Proposed Week	Actual Week	Slip	Primary Cause / Notes
M1: Concept & Planning Complete	9/15	9/29	**+2 w**	Requirements locked; initial procurement; architecture baselined
M2: First Drivable Car Prototype	9/29	10/20	**+3 w**	Mechanical iteration + power constraints required redesign
M3: Stable Comms Link (Car ↔ App)	10/6	11/17	**+6 w**	Radio multiplexing issues; moved to phone proxy/BLE path
M4: Track Built & Lane Following Working	10/20	12/1	**+6 w**	Traction + magnetic strength tuning; widened lane tolerance
M5: Multi-Car Bring-Up	11/3	11/17	**+2 w**	Coordinator changes; dynamic car ID assignment
M6: Lane Change Demonstration Ready	11/17	11/17	**0 w**	Delay logic added to handle sensor variability
M7: Final Demo Ready (End-to-End)	12/8	12/8	**0 w**	Packaging + documentation + run-throughs

Table 20: Schedule Delta by Phase (Proposed vs Actual)

Phase	Proposed End	Actual End	Delta	Notes
Concept & Planning	9/15	9/29	**+2 w**	Scope/architecture locked; initial procurement
Mechanical Fabrication & Fit	9/29	10/20	**+3 w**	STL editing; gear sizing; wheel well clearance; iterative fit checks
Electronics & Power	10/6	10/20	**+2 w**	Buck converter underpowered; power/ground shorts; microcontroller swap
App/Comms Integration	10/13	11/17	**+5 w**	WiFi/BLE radio sharing limits; moved to phone-proxy + UDP datagrams; dynamic ID work
Track & Environment	10/20	12/1	**+6 w**	Track split into modular sections; symmetry/placement effort; traction improvements
Control Algorithms	11/17	11/17	**0 w**	Lane width increased; delay logic for lane change stability
Demo & Documentation	12/8	12/8	**0 w**	End-to-end run-throughs; final packaging; documentation and demo polish

Schedule Interpretation: The most significant deviations from the proposed schedule occurred during the **electronics and power integration** and **communications architecture** phases. Unexpected power delivery limitations, component failures, and radio coexistence constraints necessitated multiple design revisions, including a transition to a phone-proxy BLE communication model. Once these architectural decisions were finalized, subsequent phases stabilized. Adjustments to mechanical design, track construction, and sensor tolerances reduced algorithmic complexity and enabled the control software to converge as planned, allowing final integration and demonstration activities to remain on schedule.

Project Planning – Actual Schedule

Table 21: Actual Schedule (Weeks W1–W8: 9/1 to 10/20)

Phase	Scope	W1	W2	W3	W4	W5	W6	W7	W8
Week Start Date		9/1	9/8	9/15	9/22	9/29	10/6	10/13	10/20
Concept & Planning	Requirements, architecture, initial procurement								
Mechanical Fabrication & Fit	STL edits, gear sizing, wheel wells, fit iteration								
Electronics & Power	Buck converter issues, shorts, MCU swap, wiring								
App & Communications (in progress)	WiFi/BLE testing, phone proxy architecture								
Track & Environment (in progress)	Track build, traction, magnets (initial work)								

Table 22: Actual Schedule (Weeks W9–W15: 10/27 to 12/8)

Phase	Scope	W9	W10	W11	W12	W13	W14	W15
Week Start Date		10/27	11/3	11/10	11/17	11/24	12/1	12/8
App & Communications	Finalize comms path + multi-car coordination							
Track & Environment	Track refinement + traction + layout work							
Control Algorithms	Lane following + lane change stability							
Demo & Documentation	Integration, testing, demo prep, documentation							
Milestones					M			M

Project Performance - Budget

Table 23: Bill of Materials (BOM) – Project Cost Summary

Item	Part Name	Manufacturer	Qty	Unit Price (USD)	Total Price (USD)
1	ESP32	Seeed Studio	1	$24	$24
2	Hall Effect Sensors	ALLECIN	1	$8	$8
3	Thin Magnetic Strip	Saypacck	4	$16	$64
4	Foldable White Board	Maxtek	1	$70	$70
5	Batteries 2S 7.4V	Crazepony	2	$24	$48
6	Raspberry Pi Power Supply	RasTech	1	$13	$13
7	Raspberry Pi Active Cooler	PNY	1	$10	$10
8	SD Card	uxcell	1	$7	$7
9	DC Motor with Encoder 6V	PES	4	$17	$68
10	Servo Motor GH-S37D	hiBCTR	1	$19	$19
11	Buck Converter	Tbest	4	$10	$40
12	Motor Controller	Tbest	4	$14	$56
			Total Project BOM Cost		**$427**

Table 24: Budget Utilization Summary

Metric	Value (USD)	Percent	Visualization
Max Budget	$600	100%	
Actual BOM Cost	$427	71.2%	
Remaining Budget	$173	28.8%	

Table 25: Cost Contribution by BOM Line Item (Percent of Total)

Item	Part Name	Total (USD)	% of $427	Visual Share
4	Foldable White Board	$70	16.4%	
9	DC Motor with Encoder 6V	$68	15.9%	
3	Thin Magnetic Strip	$64	15.0%	
12	Motor Controller	$56	13.1%	
5	Batteries 2S 7.4V	$48	11.2%	
11	Buck Converter	$40	9.4%	
1	ESP32	$24	5.6%	
10	Servo Motor GH-S37D	$19	4.4%	
6	Raspberry Pi Power Supply	$13	3.0%	
7	Raspberry Pi Active Cooler	$10	2.3%	
2	Hall Effect Sensors	$8	1.9%	
8	SD Card	$7	1.6%	
	Total	**$427**	**100%**	

Budget Result: The final BOM cost was $427, which is $173 under the $600 budget (71.2% utilization).

Project Performance - Meeting Original Expectations

Implemented Features

Table 26: Comparison of Proposed Objectives and Final Project Outcomes

Project Area	Original Proposal Expectations	Final Implemented Outcome
Vehicle Control	Basic autonomous lane-following behavior using magnetic sensing, with limited robustness expectations.	Implemented a stable dual-sensor lane-following algorithm with state-based control, enabling smooth navigation on straights and turns with minimal oscillation.
Communication Architecture	BLE-based vehicle control with a simple centralized server for command dispatch.	Developed a multi-stage communication pipeline (BLE → phone gateway → UDP coordinator) supporting bidirectional, low-latency messaging and multiple concurrent devices.
Backend Coordination	Single-session or single-vehicle backend handling.	Implemented a room-based, multiplexing coordinator server capable of routing telemetry and commands across multiple vehicles and operators simultaneously.
Lane Changing	Optional stretch goal with limited validation and safety guarantees.	Fully implemented BLE-triggered lane-changing routines with explicit safety checks to prevent invalid lane transitions (inner/outer lane constraints).
Mobile Application	Basic mobile interface for manual testing and debugging.	Developed a full-featured iOS application acting as both the primary user interface and a BLE-to-UDP gateway, supporting real-time control, telemetry forwarding, and server connectivity.
Mechanical Platform	Use of an off-the-shelf or minimally modified chassis.	Significantly modified a 3D print-in-place chassis to integrate motors, sensors, wiring channels, battery compartment, switch access, and custom drivetrain components.
Electrical Integration	Proof-of-concept wiring and component integration, with theorized components.	Delivered an electrical system with, safe wiring practices, and repeatable assembly across multiple vehicles that scales.

Deferred and Unimplemented Features

Table 27: Planned Features Not Implemented in Final System

Feature	Original Motivation	Reason for Deferral
Lap Counting	Track lap counting using magnetic markers for race timing and scoring.	Deferred to prioritize reliable lane-following and lane-changing behavior. Accurate lap counting would have required additional track markers and sensor logic beyond the available development time.
Multi-Vehicle Racing	Simultaneous control and coordination of multiple vehicles on the same track.	The backend architecture supports multiple vehicles; however, testing focused on single-vehicle stability to ensure safety and robustness before scaling to concurrent operation.
Visual Indicators (LEDs)	Onboard LED indicators for state feedback (lane, command reception, fault conditions).	Not implemented to reduce wiring complexity and power draw, and to focus development effort on core sensing and control algorithms.
Audio Feedback	Onboard audio cues for events such as lane changes or power-up triggers.	Deferred due to limited available space on the vehicle chassis and to avoid additional power and timing complexity.
Onboard Video Streaming	Live video streaming from an onboard camera using the ESP32 platform.	Originally explored during early ESP32 prototyping, but removed when migrating to the Arduino Nano BLE, which does not natively support video streaming.
Portable Coordinator Server	Self-contained local server running on a portable embedded device.	Deferred in favor of a VPS and homelab-based deployment to simplify debugging, logging, and remote testing during development.
Advanced Telemetry Visualization	Real-time graphical dashboards for vehicle state, lap timing, and sensor diagnostics.	Basic telemetry forwarding was implemented; advanced visualization was deferred to focus on backend correctness and low-latency communication.

Team Contributions

Andrew Congdon

Table 28: Contributions of Andrew Congdon

Focus Area	Description of Contribution
Concept Sketches	Made initial concept sketches of the car with the proposed electrical layout
Car Mechanical Design	Found 3D model to base design on, and made all edits to the STL for fitting electrical components
Pinion Gear Sizing	Produced multiple iterations of pinion gear, eventually finding a size that interfaced with the rear axle
Scheduled Call with Former Anki AI Lead	Found Anki former AI lead on LinkedIn and scheduled time for all team members to meet and ask questions
Track Layout Constructioan	Traced track layout with exact dimensions and laid down strips around each of the 3 whiteboard sections. Placed grip tape around track as well.
Car Testing/Finetuning	Helped Manny with testing the control algorithm and finetuning different parameters

Manny Hamer

Table 29: Contributions of Manny Hamer

Focus Area	Description of Contribution
Hall Effect Sensor Testing	Completed initial hall effect sensor testing with magnetic strips. Determined that stronger strips would be needed for the final design.
Kotlin Testing App	Developed an Kotlin testing app for android. App was capable of connecting to Cars over BLE, sending commands to test lane changes and braking.
Arduino Firmware	Utilized BLE capabilities of Arduino Nano BLE microcontroller to accept commands for braking, lane changing, and powerups.
Lane Following Algorithm	Developed and fine-tuned algorithm for following magnetic lane with hall effect sensors.
Lane Changing Routines	Developed and fine-tuned lane changing routines for cars transitioning between the two lanes on the track
Track Construction	Contributed to track construction with Andrew, laying down magnetic tape and grip tape on the sides according to specific measurements.

Ryan Paillet

Table 30: Contributions of Ryan Paillet

Focus Area	Description of Contribution
Soldering Electronic Equipment	Assembled and reworked vehicle electronics, including power wiring, sensors, and communication hardware.
Coordinator Server via Homelab	Deployed and maintained the central coordination server using a personal homelab environment.
Microcontroller Programming and Toolchain	Developed and maintained embedded firmware and configured flashing and debugging workflows.
iOS Swift Development	Developed the iOS application for vehicle control and game interaction over BLE.
Electrical Debugging	Diagnosed and resolved power, grounding, and signal integrity issues during system integration.
Radio Debugging	Investigated BLE and WiFi communication issues and supported improvements to communication reliability.