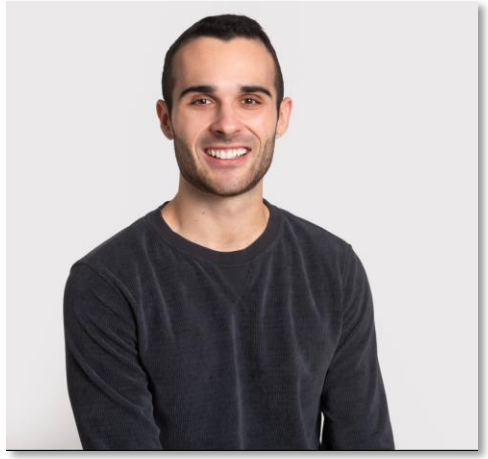
RACE

(Remote Access Control for Electric) Cars

Final Design Presentation – Group 30

The Team



Austin Silva
Electrical Engineering

Michael Mallette
Electrical Engineering

Jesvany Colon
Electrical Engineering

Sean Chaney
Computer Engineering

Motivation & Background

- For this project, we want to merge the thrill of traditional RC car racing with online accessibility into a competitive gaming system where users can race against one another remotely.
- As avid gamers, this is intended to be a medium between the physical world and virtual space, offering an experience unique to users since it blends elements of physical RC car racing and online video game racing.
- We aim to create an engaging platform and complete racing system that can connect people of common interests together, enjoying a fun racing experience.

Goals

- Develop an immersive internet-controlled (IC) car racing experience.
- Enable real-time first-person view (FPV) video streaming.
- Implement lap and time tracking for accurate race results.
- Display race times on online leaderboards.

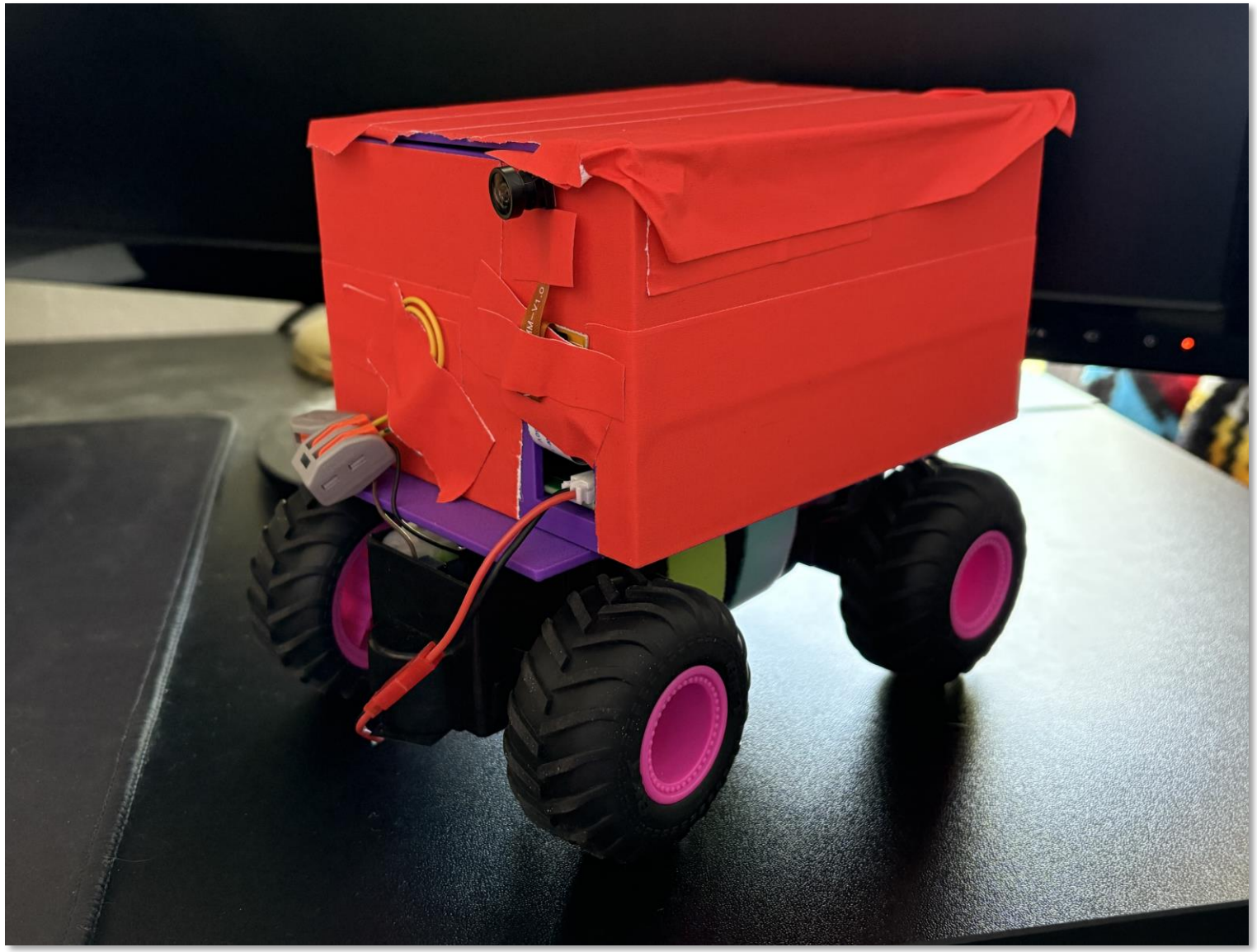
Objectives

- Build two IC car models with driving motors, steering motors, and microcontrollers.
- Integrate FPV cameras for real-time streaming and an immersive user experience.
- Develop an automated system for accurate lap tracking and race timing.
- Create a user-friendly interface for remote control and live video access.
- Focus on accessibility for a seamless and enjoyable user experience.
- Optimize performance for low latency and multi-user functionality.

Project Features

- **Internet-Based Remote Control:** Users can control IC cars via a low-latency web interface.
- **Web Interface:** Provides display of control inputs, live video, and race stats (lap count/time and leaderboard) in a browser.
- **FPV Camera System:** Real-time video streaming that offers an immersive first-person racing experience.
- **Multi-User Support:** A scalable server enables simultaneous racing with smooth performance.
- **Finish Line Detection:** Camera sensor that can distinguish cars based on their color, to determine winners.
- **Lap Tracking:** Automated sensor to ensure accurate recording of race results.
- **Custom PCB:** Integrates microcontroller-based command execution for motors, steering, and sensors.

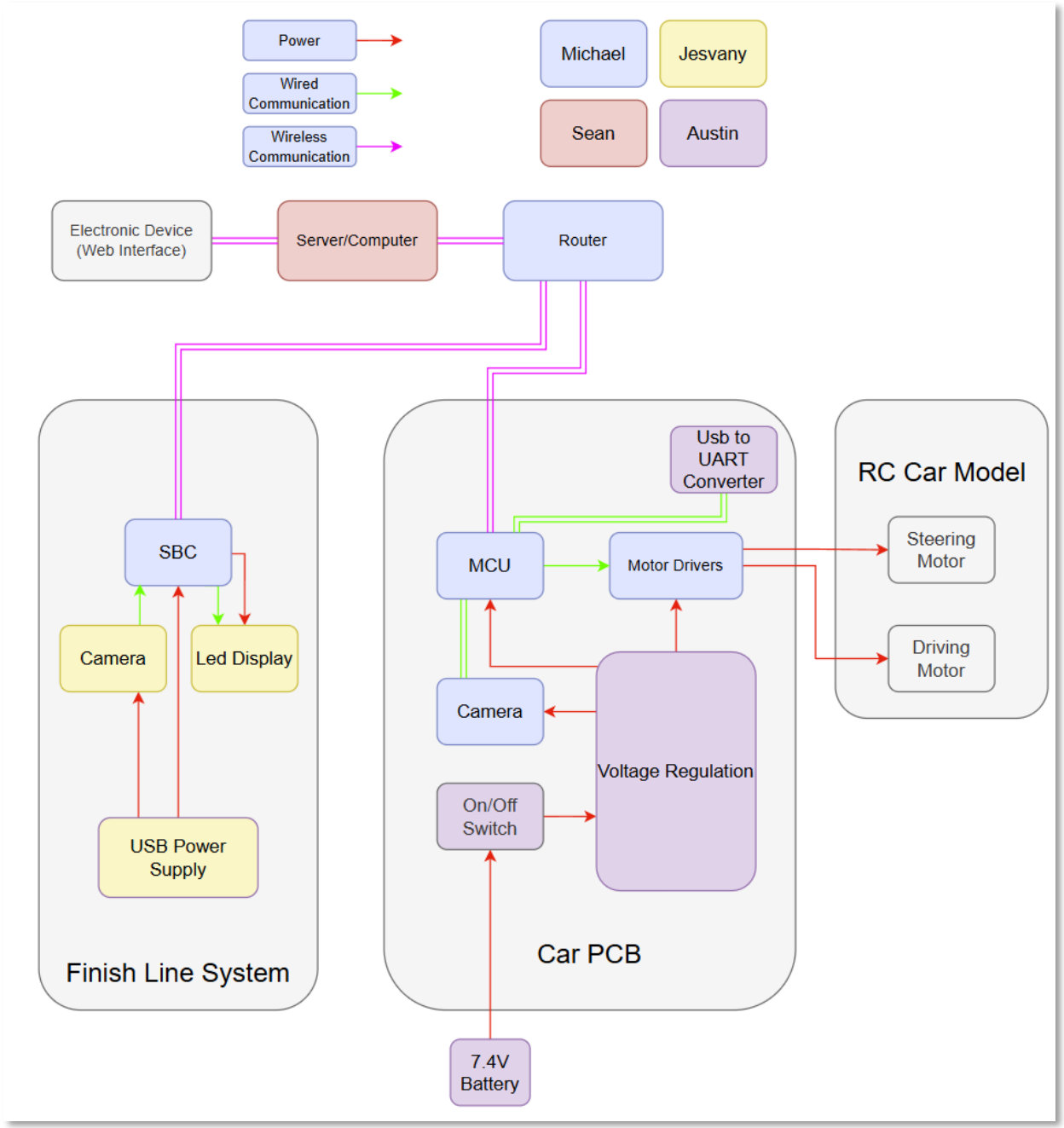
RACE Car Model



Engineering Specifications

Specifications	
Car Size	≤6”x12”
Car Control Delay	≤1.5 seconds
Car Chassis Maintainability	Chassis lasts for 100 runtime hours
Car Battery Life	≥15 minutes
Video FPS	10-30 FPS
Video Resolution	≥240x240
Total Cost of 2 Cars	≤$200
Connectivity Distance	10-30 ft
Finish Line Detection Accuracy	≥75%

RACE System Hardware Design



Hardware Block Diagram

Car Selection (Chassis and Motors)

- We gathered and tested three different RC cars.
- Deciding Factors,
  - Most Durable Chassis
  - DC Motors Possess Suitable Voltage Operating Range of 2V-5V
  - Speed is More Balanced than other cars tested
  - Greater User Control

MCU Selection

Specification	ESP32 Wrover	ESP8266	Raspberry Pi Pico 2 W	Arduino Nano
Microcontroller	Espressif (Tensilica Xtensa LX6)	Espressif (Tensilica Xtensa L106)	RP2040	ATmega328
Camera Ribbon Connector	Yes	No	Yes	No
Number of Cores	2	1	2	1
Operating Voltage	3.3V	3.3V	3.3V	5V
Clock Speed	240 MHz	80 MHz	133 MHz	16 MHz
Flash Memory	8MB/16MB	4MB	2MB	32 KB
SRAM	520 KB + 8MB PSRAM	64 KB	264 KB	2 KB
Wi-Fi	Yes	Yes	No	No
Bluetooth	Yes (v4.2)	No	No	No
GPIO Pins	30	17	26	14
PWM Pins	16	8	16	6
ADC Channels	18	1	3	8
Power Consumption	Moderate to Low	Low	Low	Low
Cost (Amazon)	10+	8	10	6

- Deciding Factors,
  - Camera Ribbon Connection
  - Most Number of Pins
  - High Clock Speed
  - High RAM

Motor Driver Selection

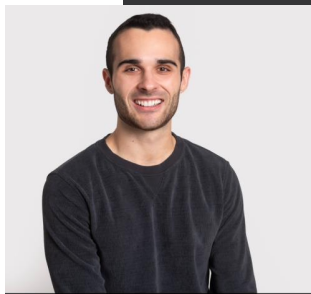
Motor Driver	Voltage Range	Current Handling	Features	Pros	Cons
L298H	4.5V - 46V	Up to 2A per channel	Dual H-bridge, simple and robust, suitable for basic applications	Widely available, simple to use, handles higher voltages	Large size, less efficient due to older technology, heat dissipation issues
DRV8871	6.5V - 45V	3.6A peak, 2A continuous	Single-channel H-bridge, efficient, built-in protection features	High current handling, good for higher power motors, reliable protection	Single channel, may need two for dual motor control
DRV8833	2.7V - 10.8V	Up to 1.5A per channel	Dual H-bridge, compact, efficient, capable of driving two motors	Compact size, efficient power usage, suitable for low voltage motors	Lower current handling compared to others, limited to lower voltage applications

- Deciding Factors
  - Small Size
  - Within motor voltage range

Camera Technologies

Features	FPV Cameras	Onboard Action Cameras	Webcams	IoT Cameras
Video Transmission	Analog RF	Digital via Wi-Fi or USB	Digital via USB or Wi-Fi	Digital via Wi-Fi
Latency	Very Low	Low-Medium	Medium	Medium
Field of View	Wide FOV, Either 120° or 170°	Variable FOV, Ranges from 90° to 170°	Small FOV, Ranges from 60° to 90°	Variable FOV, Ranges from 60° to 160°
Resolution	Low, 480p-720p	High, 1080p-4K	Medium, 720p-1080p	Medium, 720p-1080p
Image Sensor	CMOS or CCD	CMOS	CMOS or CCD	CMOS
Size	Compact	Bulkiest	Medium	Compact
Weight	Usually < 30g	100g or More	50g-100g	10g-100g
Cost	Range of $20-$100	Range of $100-$400+	Range of $30-$150	Range of $10-$100

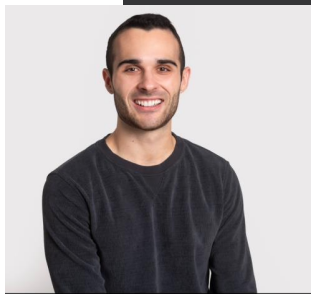
- Deciding Factors,
  - Transmission for live video streaming through Wi-Fi connection
  - FOV for ideal first-person FPV
  - Compact size
  - Low Cost
  - Compatibility with Microcontrollers



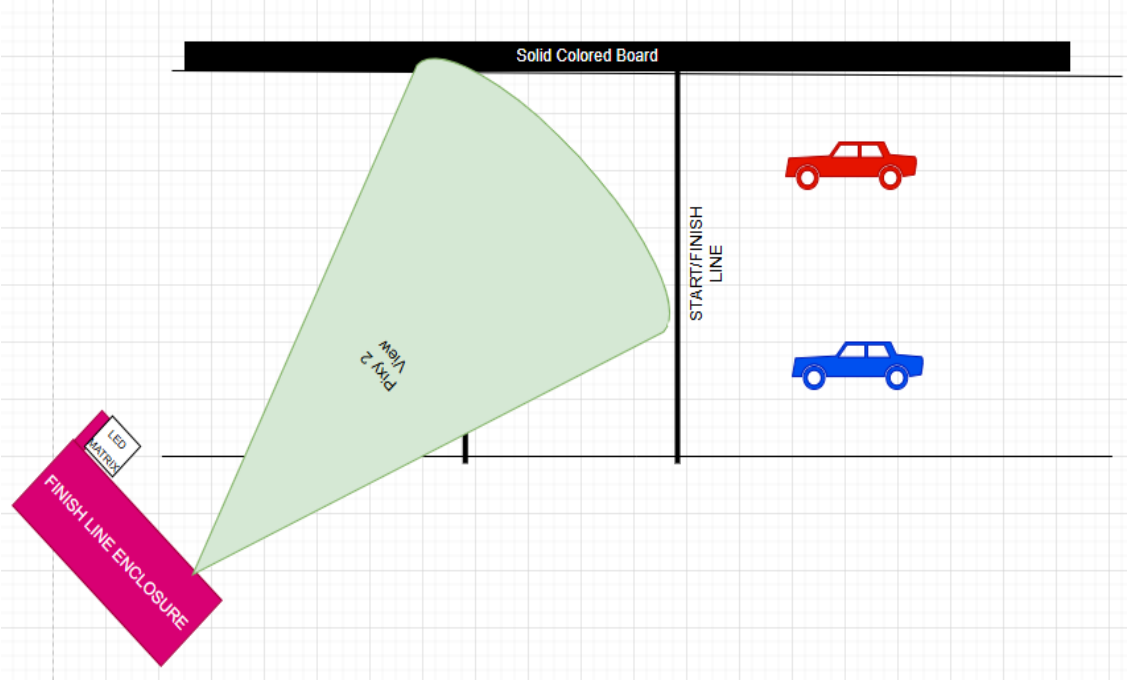
Camera Selection

Features	OV2640	OV7670
Resolution	2 MP (1600x1200)	VGA (640x480)
Frame Rate	Adjustable, 30 fps recommended	Typically 15 fps, but can go to 30 fps
Image Sensor Type	CMOS	CMOS
Interface	I2C	I2C
Field of View	Up to 160° FOV	Smaller FOV
Compatibility	Directly Compatible with ESP32	Compatible, does require additional interfacing
Power Consumption	Higher	Lower
Ease of Integration	More Support	Less Support
Cost	$6-$9	$3-$6

- Deciding Factors,
  - Higher Resolution for image capture
  - Better Quality for video streaming
  - Smoother Integration with ESP32



Finish Line Hardware Overview



The finish line consists of 3 main hardware parts:

- Pixy 2 Smart Camera Sensor
- LED Matrix
- MCU/Development Board

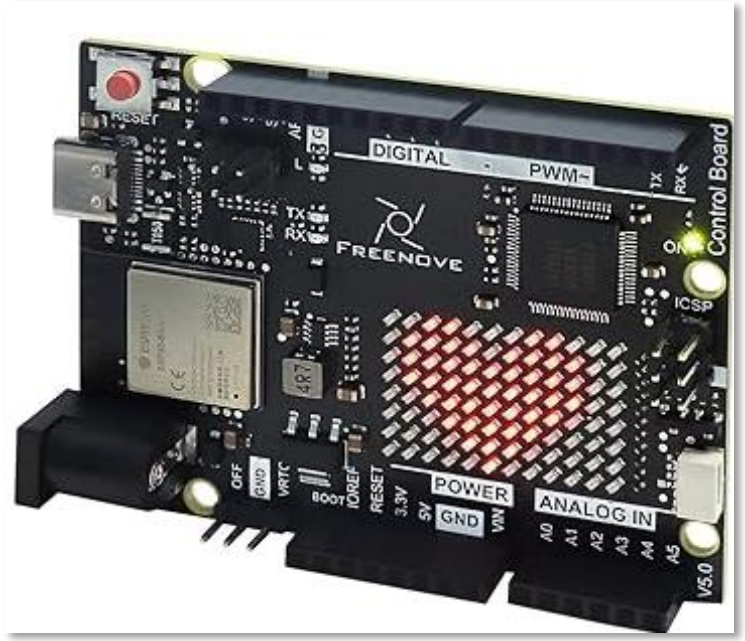
- The camera sensor is specially configured for color detection, enabling us to distinguish between racers

- LED Matrix will act as an active display for the race countdown sequence, lap tracking, and announcement of the winning car for those spectating.

- The single board computer will be able to distribute power between the other peripherals and communicate with our server.

Development Board Selection

- Originally planned to use the ESP32-WROOM-32 but it would not communicate with the PIXY 2 via SPI communication.
- We ended up deciding on the Freenove Control Board V5 Rev4 Wi-Fi due to timing in having ESP32 ready,
  - This is a cheaper but nearly identical version of the Arduino Uno Rev4 Wi-Fi.
  - This board was selected due to it being the most compatible MCU with the Pixy 2.



Development Board (Continued)

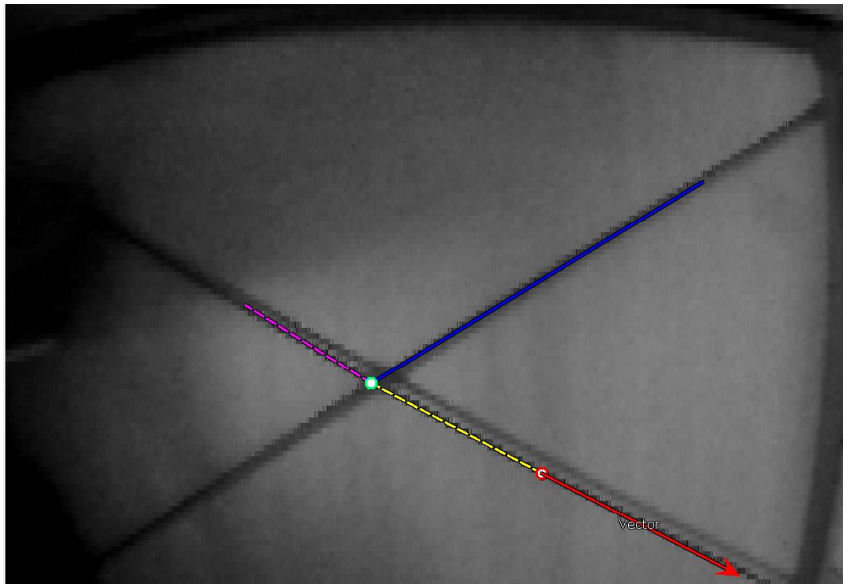
Features	ESP32-WROOM-32	Freenove Control Board V5 Rev4 Wi-Fi
Clock Speed	**Up to 240 MHz**	**48 MHz**
Flash Memory	**4 MB (32 Mbits)**	**256 KB**
SRAM	520KB	32KB
Wireless Connectivity	**Wi-Fi 802.11 b/g/n, Bluetooth v4.2 BR/EDR and BLE**	**Wi-Fi via onboard ESP32-S3 module**
Interfaces	**UART, SPI, I2C, I2S, SDIO, Ethernet, capacitive touch sensors, ADC, DAC**	**UART, I2C, SPI, CAN**
Operating Voltage	**3.0 V ~ 3.6 V**	**5 V (ESP32-S3 operates at 3.3 V)**
Additional Features	**Integrated PCB antenna, supports secure boot and flash encryption**	**Onboard ESP32-S3 module for Wi-Fi, 12x8 red LED matrix, detailed tutorials and example projects provided**
Price	$10.00	$19.99

Deciding Factors: The compatibility and support between the Freenove Control Board and Pixy 2 Camera

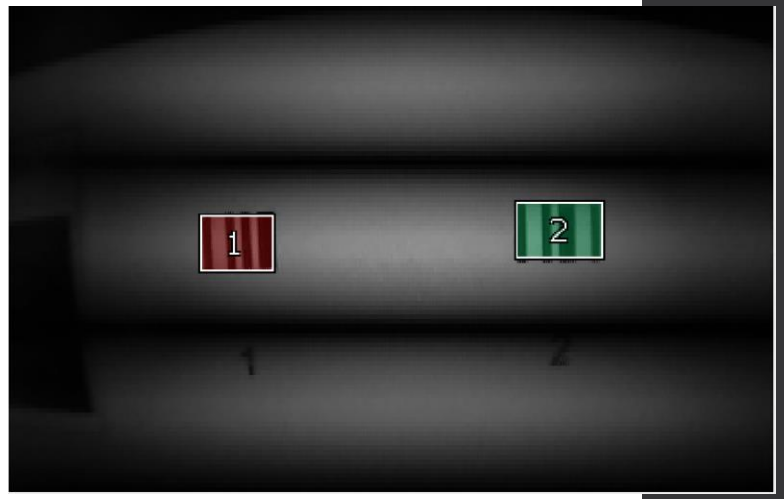
Camera Technology Research

- Color Detection

- Line Detection



- Barcode Detection



Deciding Factors:

- Allows for easier distinction between cars
- Ease of implementation
- Reliability to work for our purposes

Camera Selection

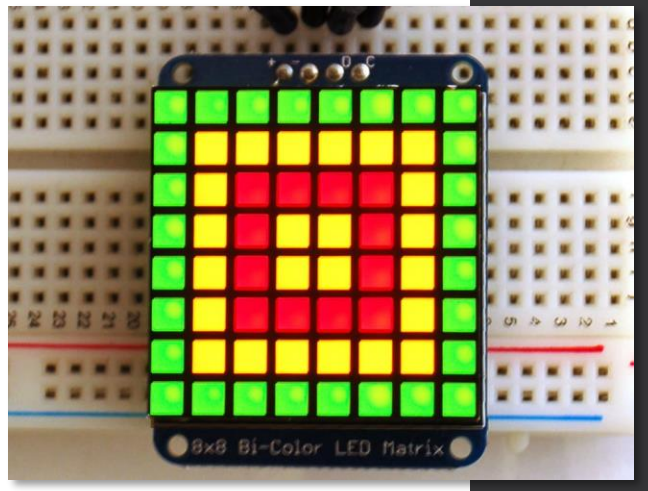
Features	PIXY2	ESP32-CAM	JeVois A33
Microcontroller/Processor	NXP LPC4330, dual-core ARM Cortex-M4/M0	ESP32-S, dual-core Xtensa LX6, 240 MHz	Allwinner A33, quad-core Cortex-A7, 1.34 GHz
Camera Sensor	Aptina MT9M114, 1296×976 resolution with integrated image flow processor	OV2640, 2 MP	1.3 MP OmniVision (supports other modules)
Frame Rate	Up to 60 fps	30 fps (at lower resolutions)	Up to 120 fps (QVGA)
Connectivity	USB, SPI, I2C, UART	Wi-Fi, Bluetooth, UART	USB, UART
Machine Vision Features	Object tracking, color detection, line tracking	Basic face detection, streaming, custom algorithms	Object tracking, face detection, custom vision algorithms
Power Requirements	5V (via USB or connector)	3.3V or 5V (via pins)	5V (via USB)
Price Range	~$60-70	~$10	~$50-60

Deciding Factors:

- Built in color detection algorithms
- SPI connectivity
- Moderate power consumption
- Sufficient frame rate

LED Matrix Selection

Features	Adafruit Bicolor LED Square Pixel Matrix with I2C Backpack	Adafruit Small 1.2" 8x8 Bi-Color (Red/Green) Square LED Matrix	MAX7219 8x8 LED Matrix
LED Matrix Size	8x8 (64 LEDs)	8x8 (64 LEDs)	8x8 (64 LEDs)
LED Color(s)	Red ,Yellow, Green (Bi-Color)	Red,Yellow, Green (Bi-Color)	Single Color (usually Red)
Driver Chip	HT16K33 (I2C)	External driver (requires an I2C backpack or controller)	MAX7219 (SPI)
Interface	I2C (with Qwiic/STEMMA QT connectors)	Requires external driver (typically I2C)	SPI
Voltage	3.3V to 5V	3.3V to 5V	5V
Dimensions	1.5" x 1.5" (38mm x 38mm)	1.2" x 1.2" (30mm x 30mm)	2.5" x 1.5" (64mm x 38mm)
Power Consumption	~20-30mA (depends on brightness)	~30-40mA (depends on brightness)	~50mA (depends on brightness and LEDs lit)
Price	$15.95	$7.95	$8.39



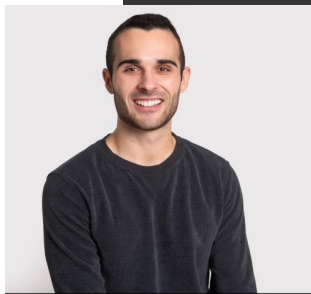
Deciding Factors:

- Bicolor capability
- Low power consumption
- Communication protocol compatibility
- Moderate price
- Bigger display

RACE System Power Needs

Device	Voltage (V)	Current (mA)	Peak Power (W)
MCU	3.3	150-300	~1
Camera	3.3	100-150	~0.5
Driving Motor	5	500	2.5
Steering Motor	5	800	4
Dual Motor Driver	Range of 2.7-10.8	1500 (Max per Channel)	6.5 (Max at 5V Input)

- Our IC (Internet-Controlled) Cars utilize a suitable 7.4 V Battery with significant capacity (mAh) to efficiently and effectively power all the devices for our cars' desired battery lifetime.
- Our IC Cars utilize 3.3 V and 5 V Voltage Regulators to step-down the 7.4 V for sufficient power delivery to their respective devices.
- Our Finish Line System is powered by a 5 V USB power supply that is connected to the Freenove Control Board. From that connection, the control board distributes power to both the Pixy 2 camera and the LED Matrix.

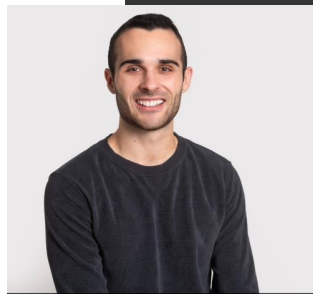


Power Supply

- Due to the properties of lithium-based batteries and their effectiveness in RC car applications, our IC cars utilize a Li-Po battery product as their power source.

Features	URGENEX 2S	Turnigy 2S	Gens Ace 2S
Voltage	7.4 V	7.4 V	7.4 V
Capacity	1100 mAh	1300 mAh	1200 mAh
C Rating	35	30	30
Weight	52g	76g	60g
Dimensions	60x34x16 mm	76x34x17 mm	125x20x12 mm
Quantity	2	1	1
Cost	$24 (including charger)	$9	$22

- 1100 mAh Battery Capacity was determined to significantly exceed our desired battery lifetime for more than 15 minutes of runtime.

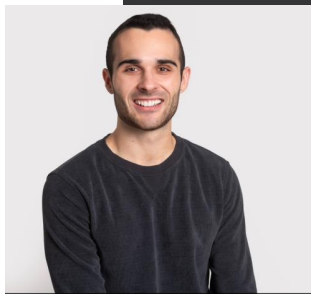


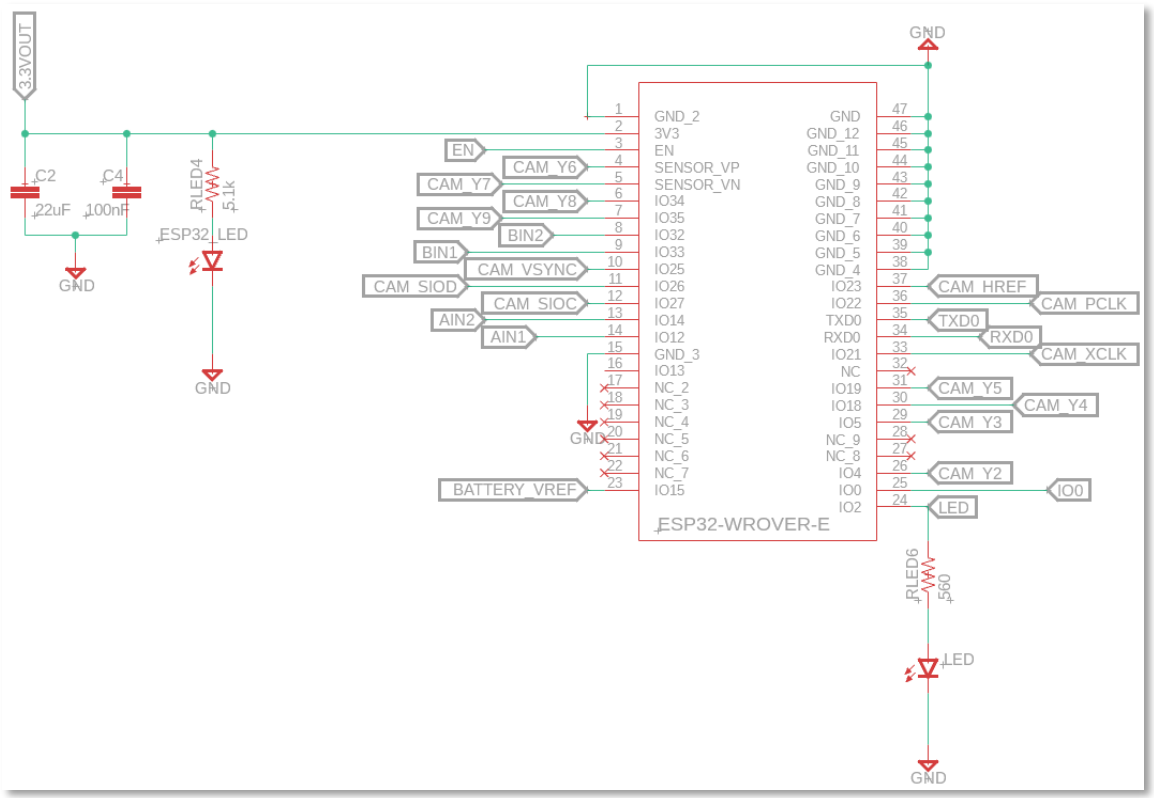
Power Distribution

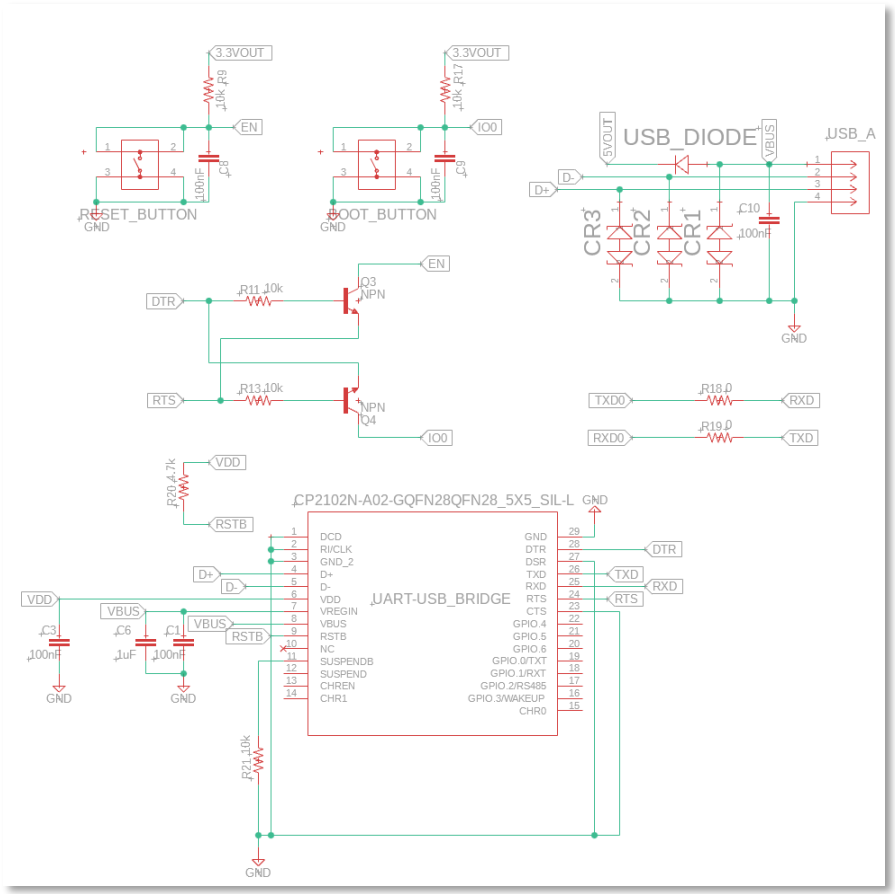
Specifications	Linear Regulators	Switching Regulators
Efficiency	Moderate	High
Design Complexity	Lower complexity with fewer components	Higher complexity with additional components
Thermal Performance	Poor due to more heat generation	Better due to less heat generation
Transient Response	Faster	Slower
Noise	Less	More (due to switching)
Input Voltage	Limited, needs input to be greater than output	Wide range, can step-up or step-down
Output Voltage	Fixed or Adjustable	Flexible
Cost	Lower	Higher

Features	TPS62142	TPS62143	TPS561208	TPS563231
Input Voltage	3-17 V	3-17 V	4.5-17 V	4.5-17 V
Output Voltage	3.3 V	5 V	0.76-7 V	0.6-7 V
Max Current	2 A	2 A	1 A	3 A
Efficiency	93.1%	93.1 %	93.6%	92.2%
Switching Frequency	1250-kHz to 2500-kHz	1250-kHz to 2500-kHz	580-kHz	600-kHz
Topology	Buck	Buck	Buck	Buck
Price	$0.57	$0.57	$0.16	$0.13

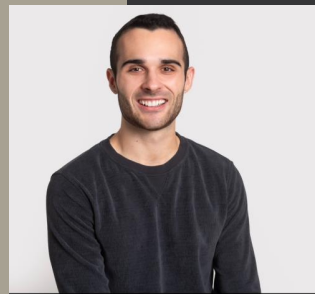
- Deciding Factors,
  - High Efficiency
  - Good Thermal Performance
  - Adjustable Input/Output Voltage Range
  - Lower Switching Frequency

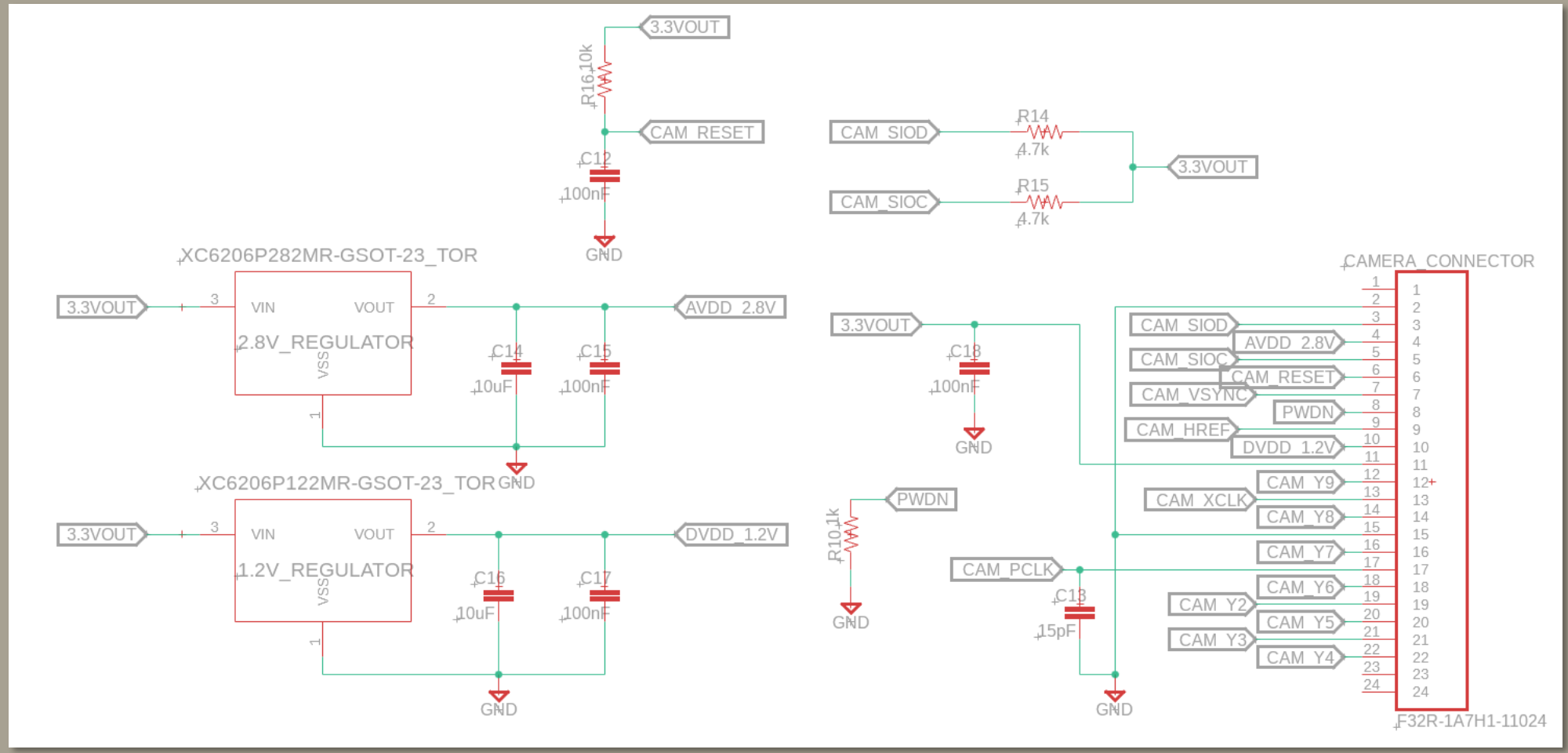




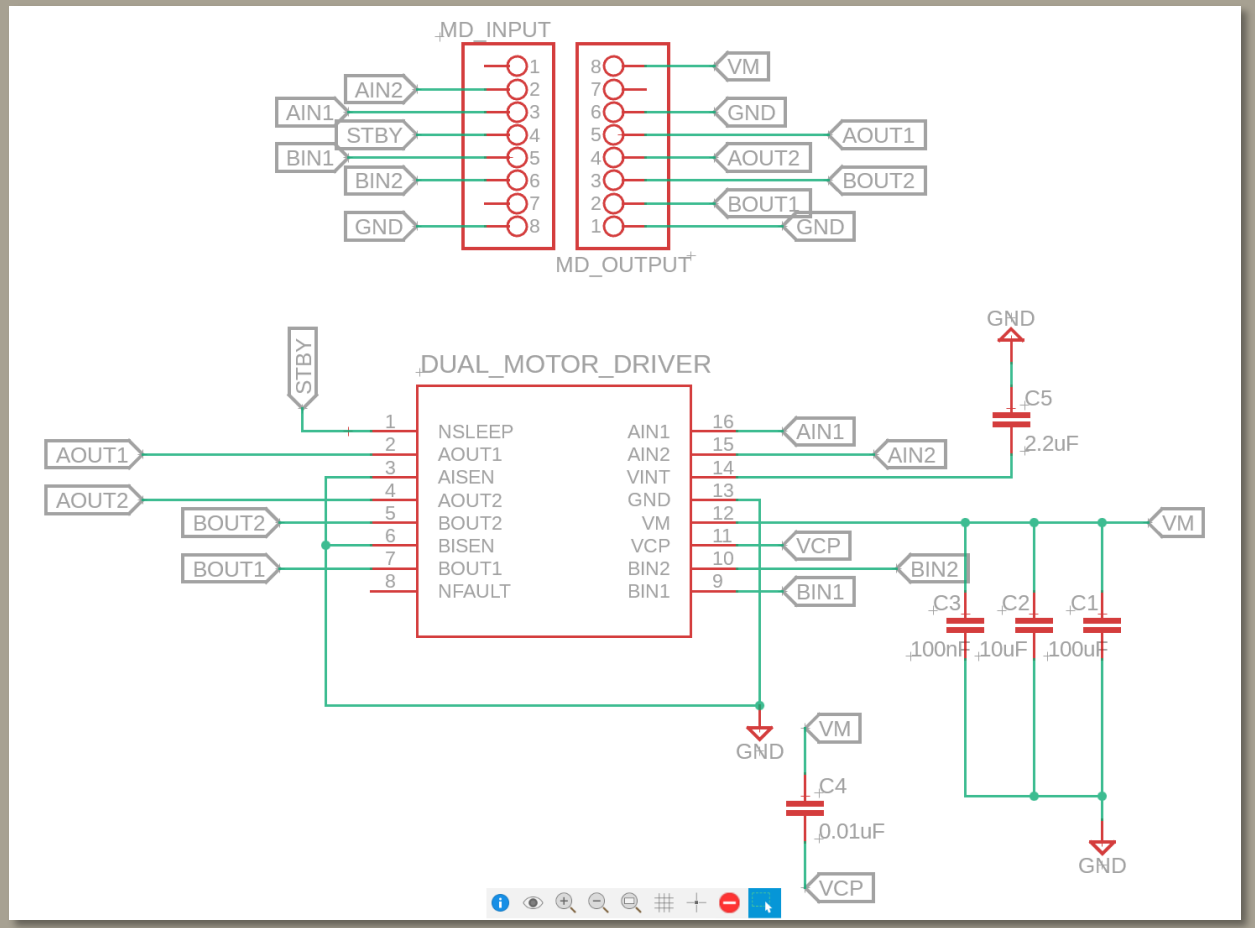


MCU & USB-UART Schematics

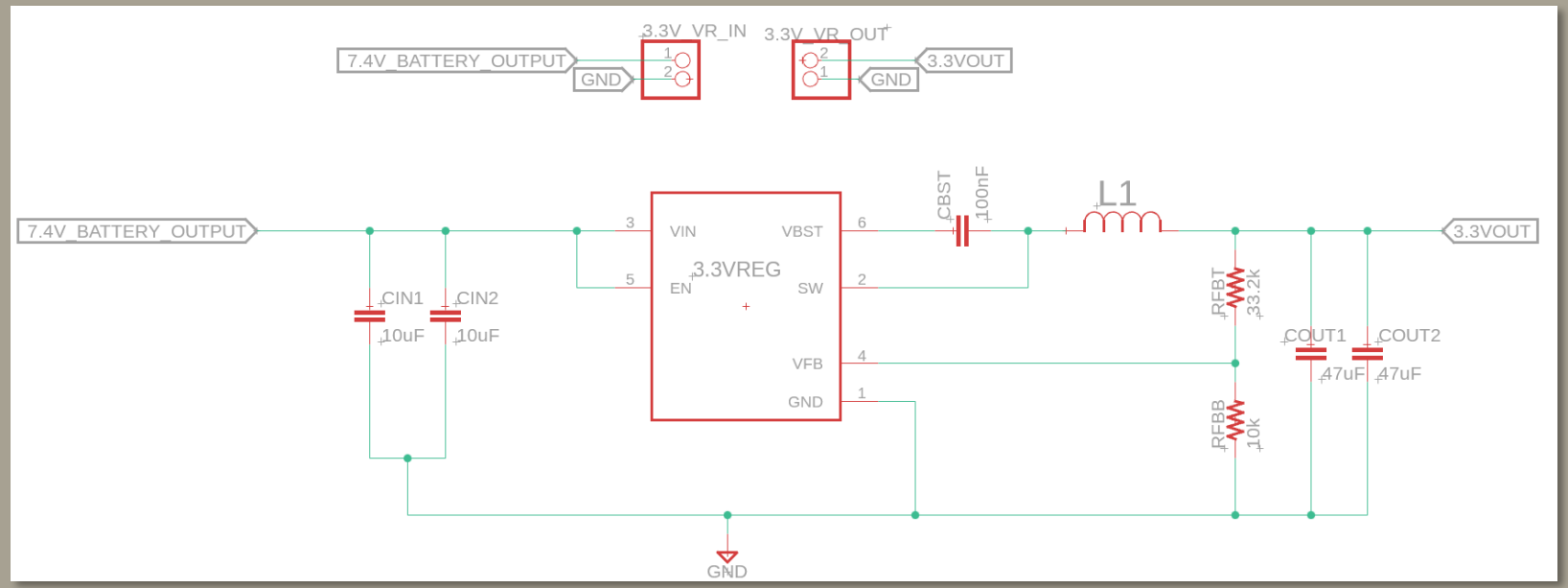


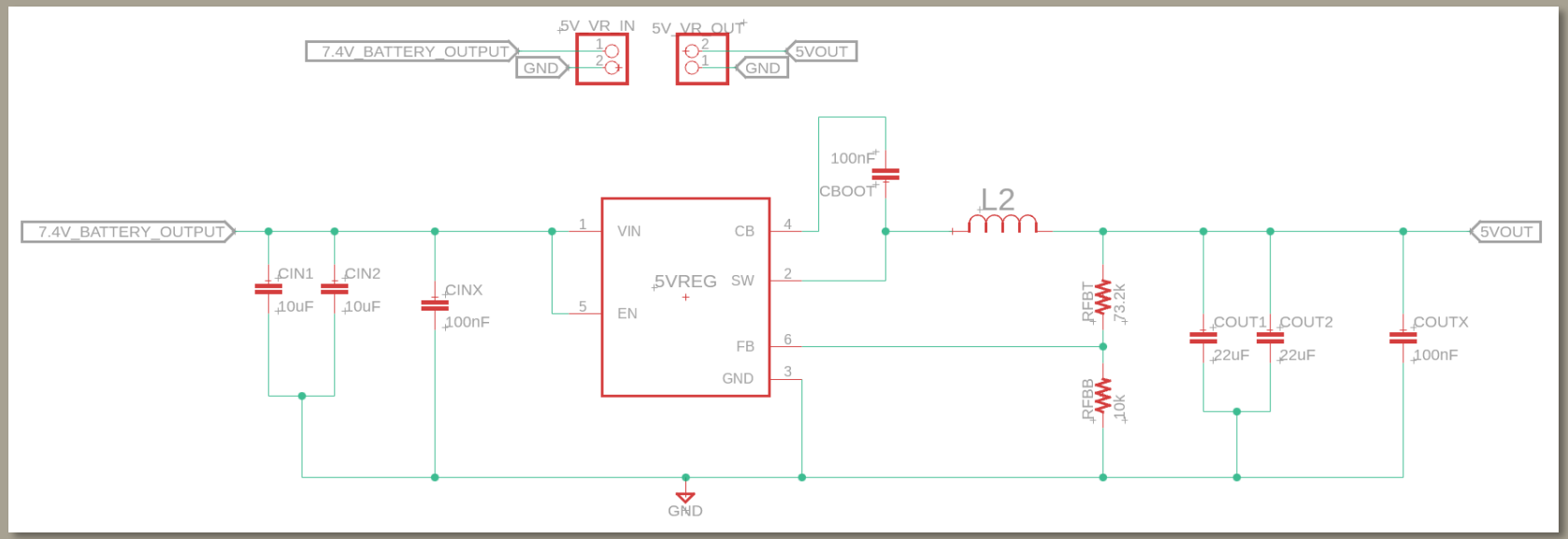


OV2640 Schematic

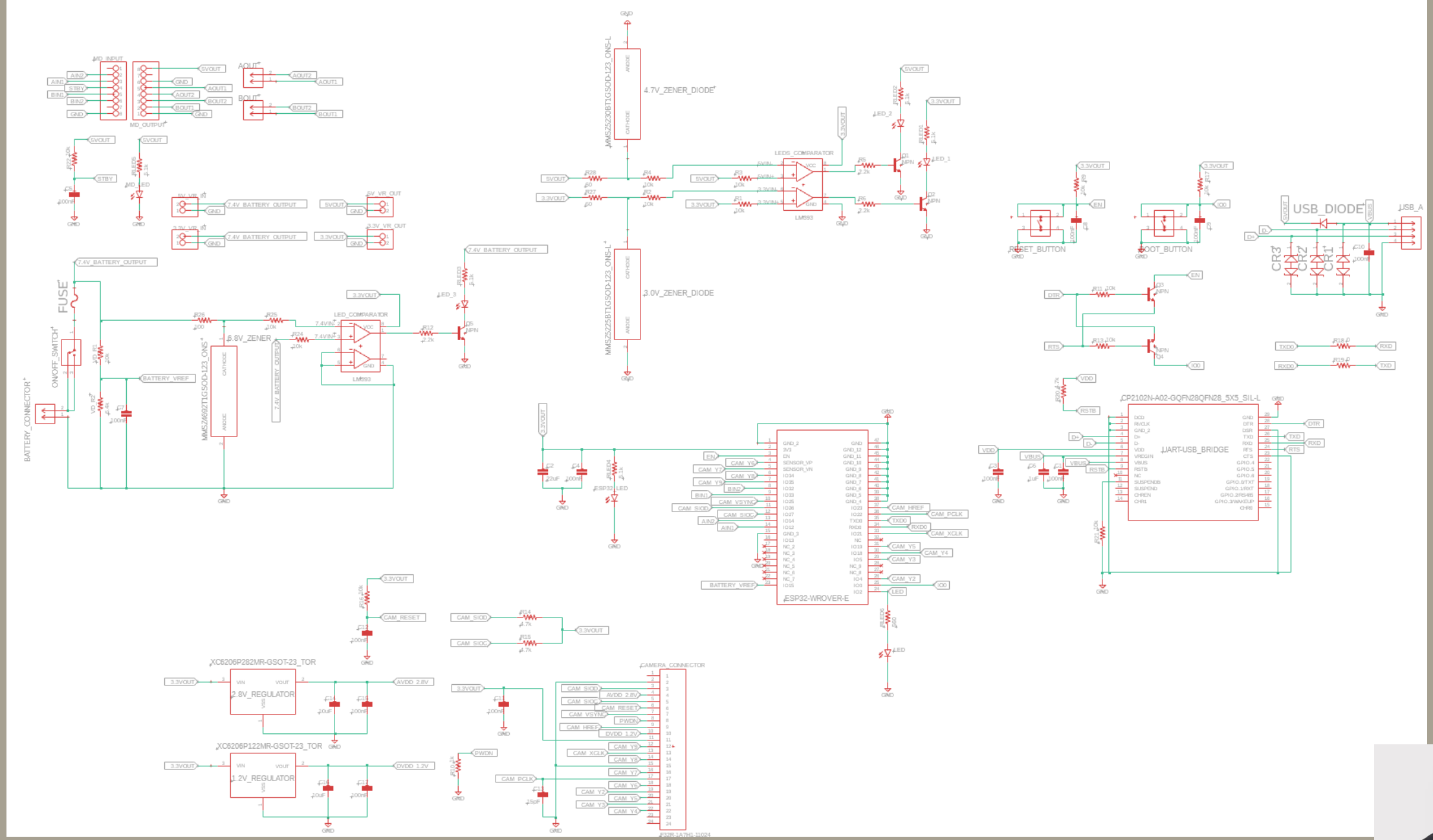


Motor Driver Schematic

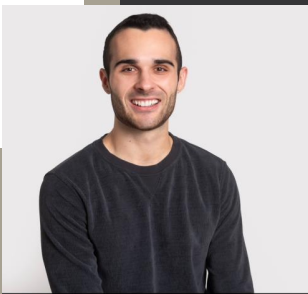


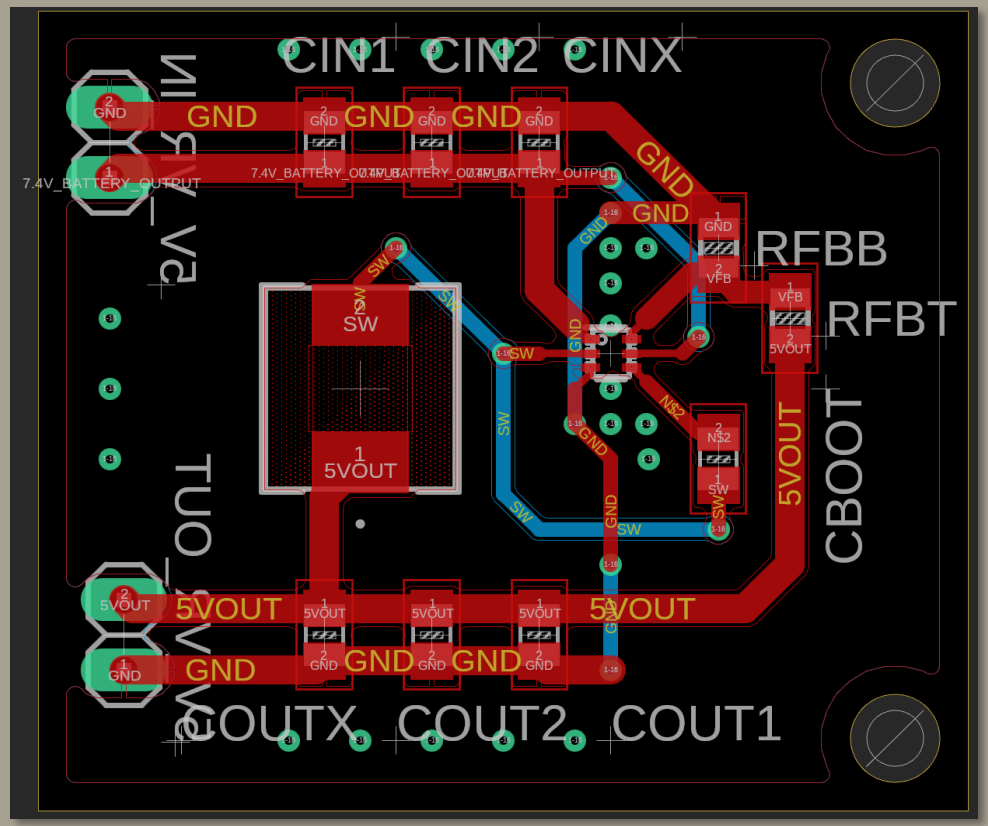


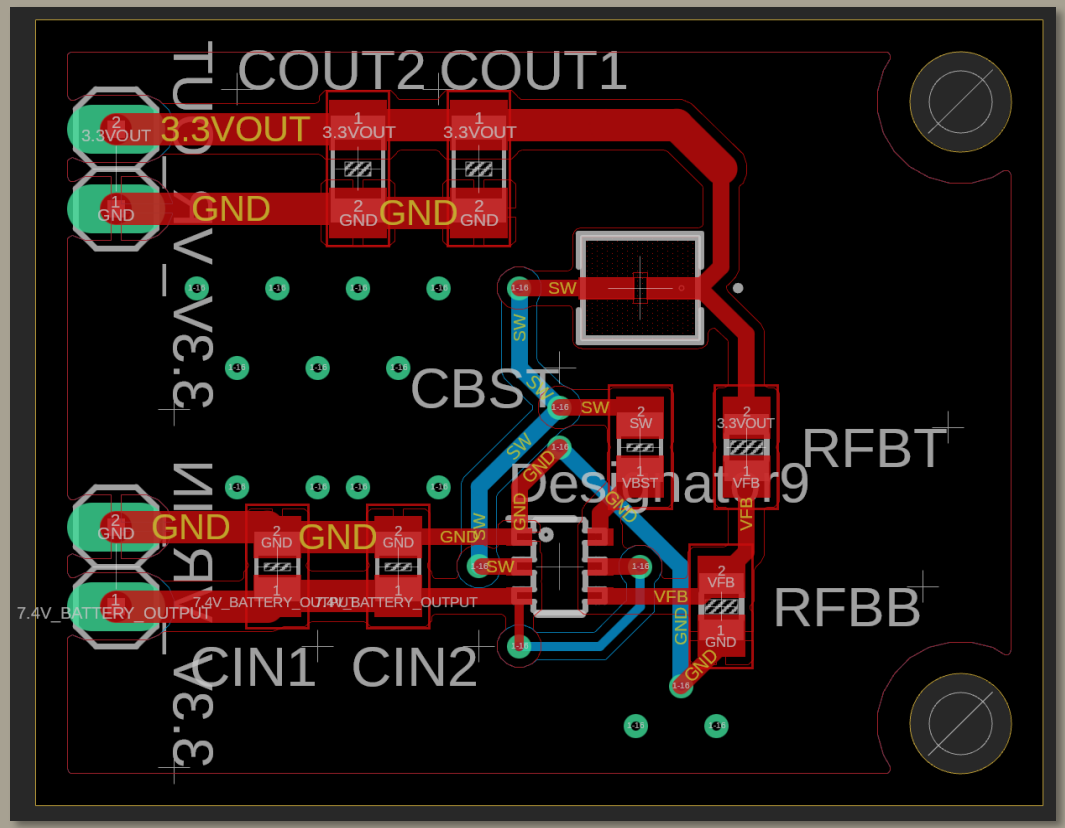
Voltage Regulation Schematics



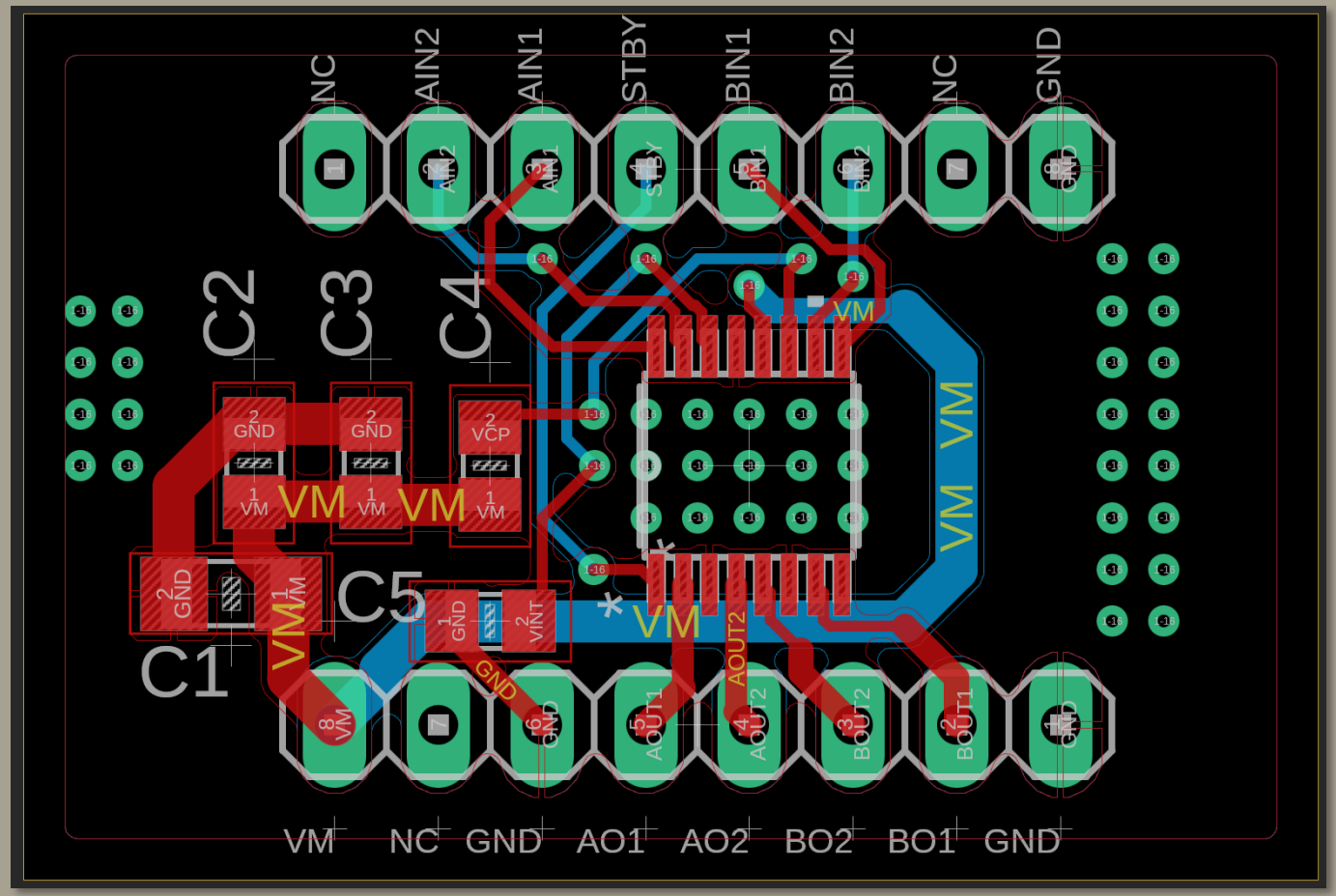
Overall Schematic





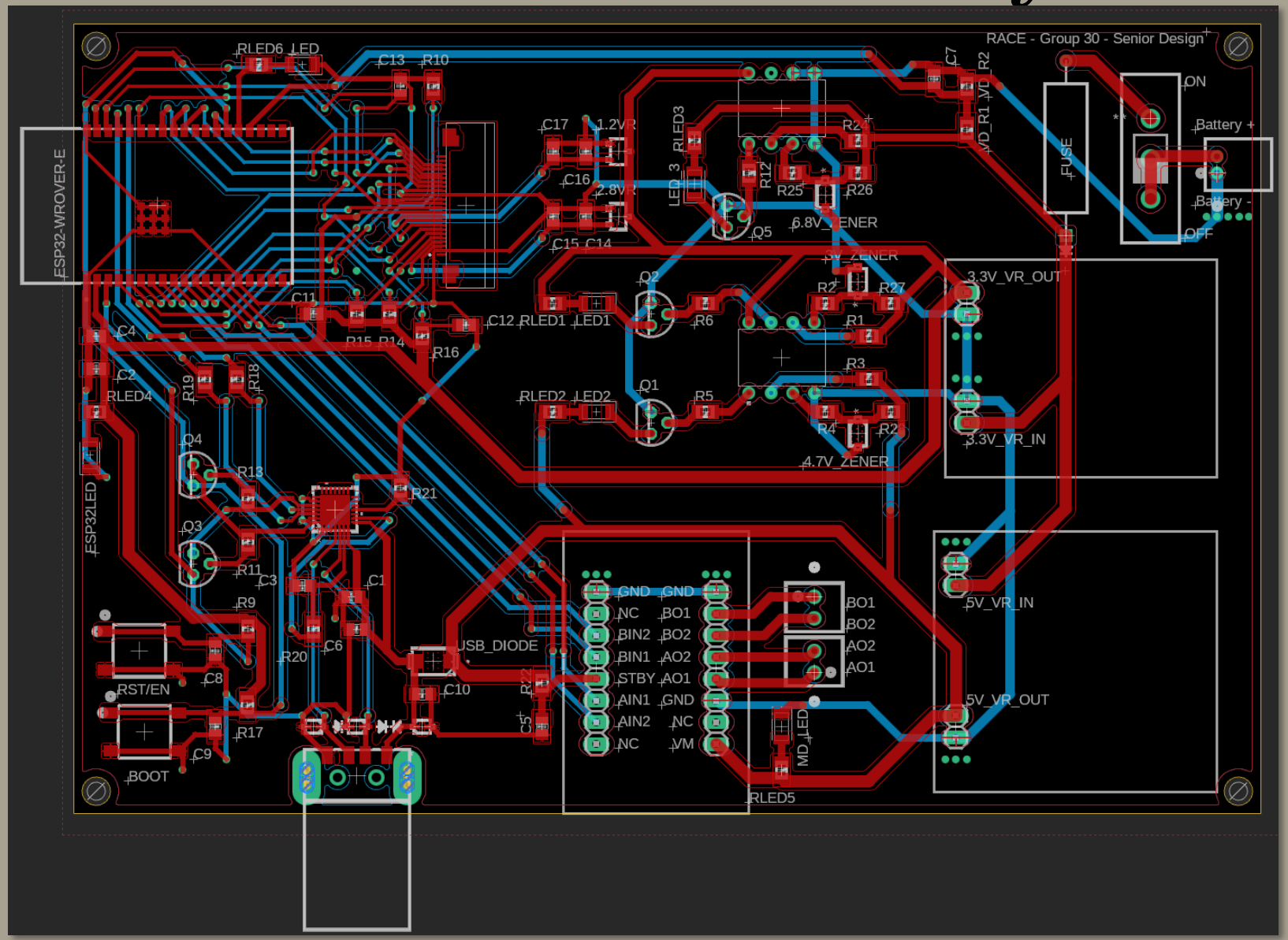


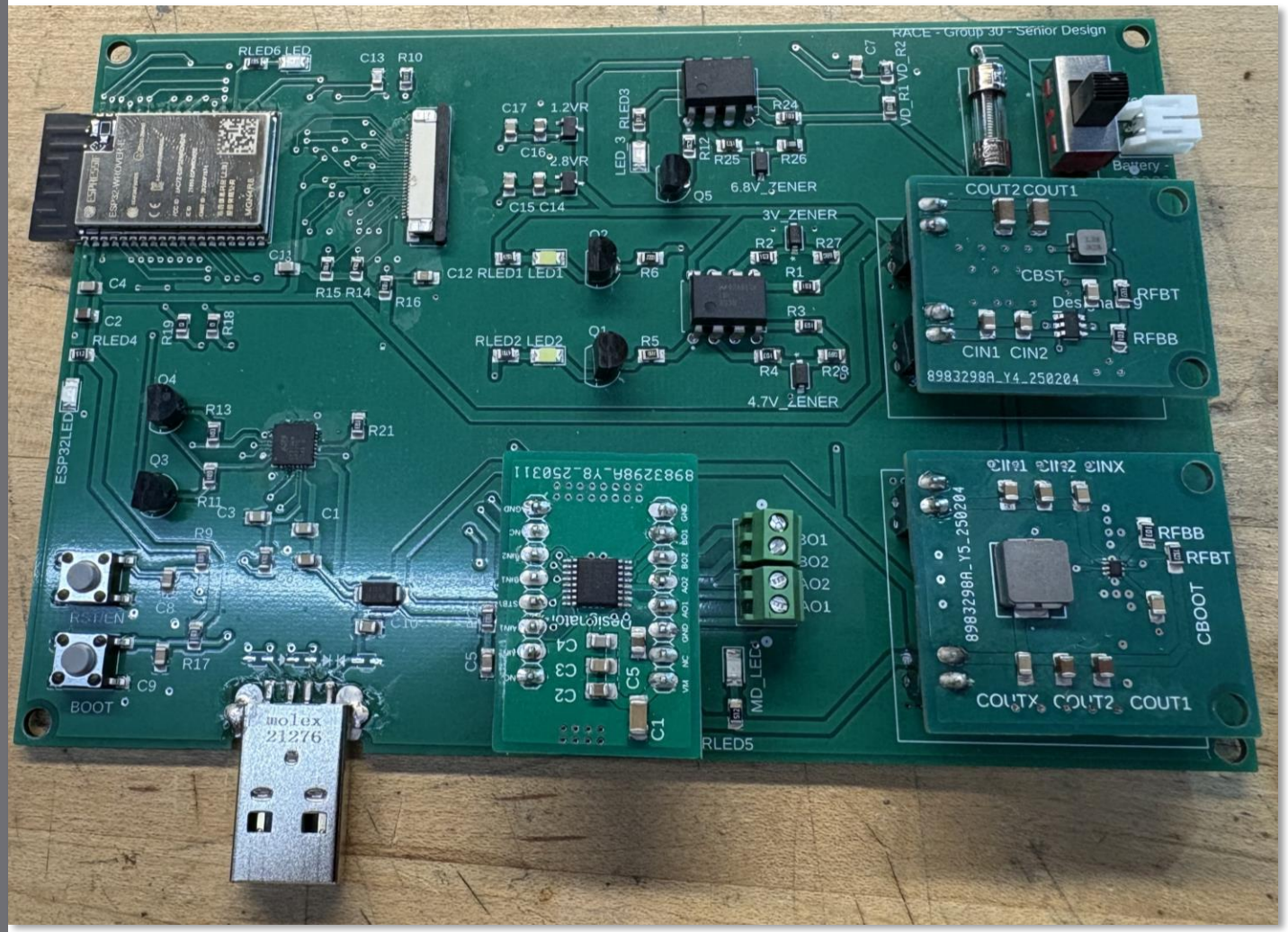
Voltage Regulation Board Layouts

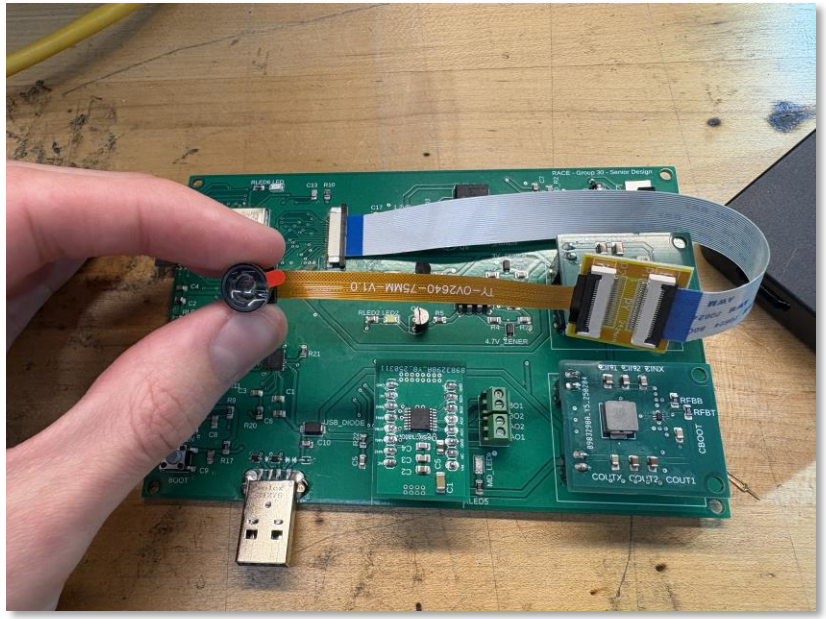


Motor Driver Board Layout

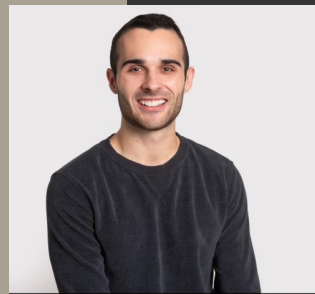
Main PCB Board Layout







Fully Assembled PCB



RACE System Communication Network

Wireless Communication Protocols

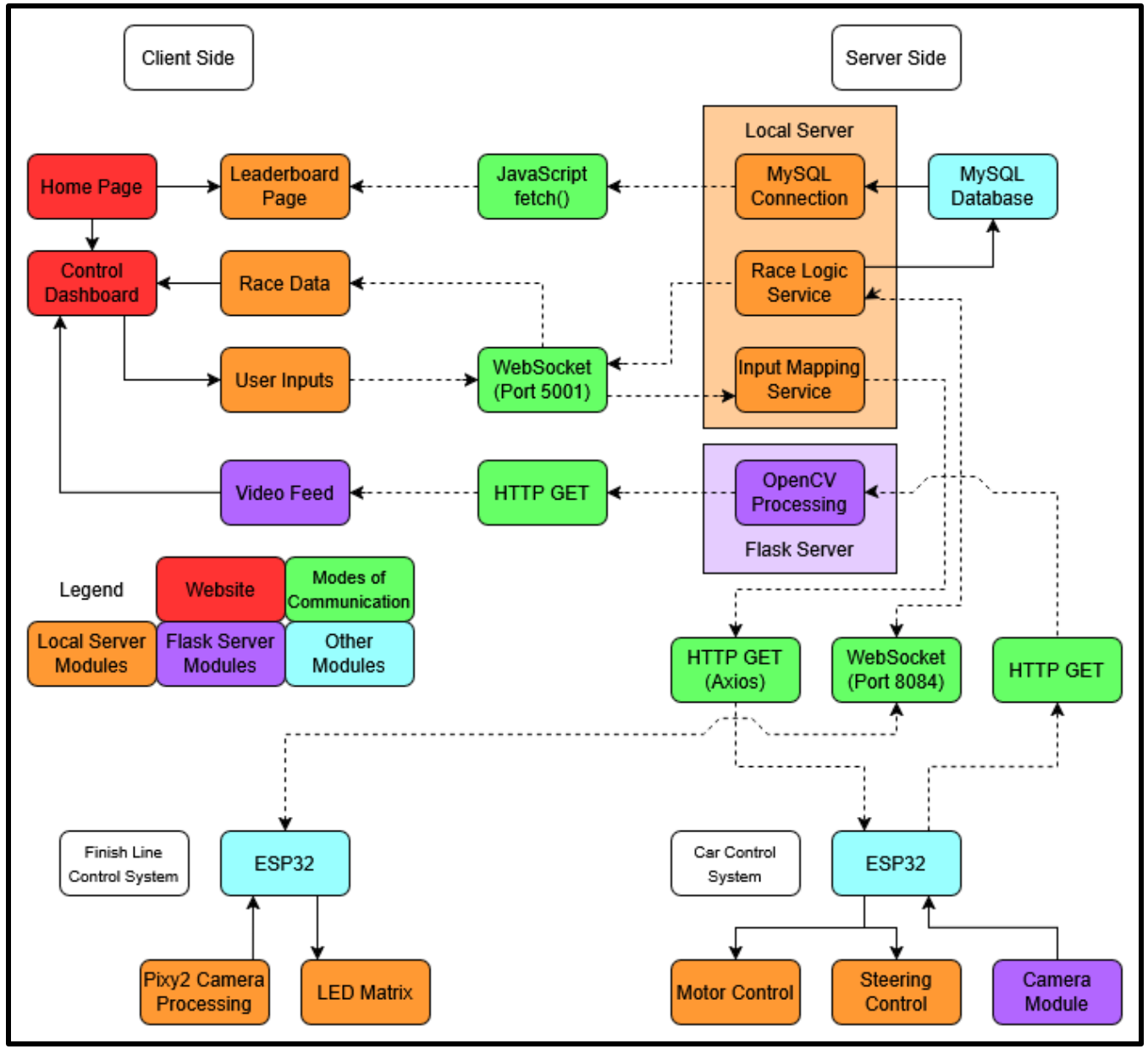
Mode	Type	Strengths	Weaknesses	Best Use in Project
Wi-Fi	Wireless local network	High bandwidth, widely supported, enables full system connectivity	Requires stable signal and shared access	Used to wirelessly connect ESP32s, servers, and website on a shared local network
WebSocket	Full-duplex communication	Real-time, bidirectional, low-latency	Requires persistent connection and setup overhead	Used to send control commands and race/lap updates between website and Node.js server
HTTP GET	Request-response protocol	Simple, fast, easy to implement	One-way only, less efficient for real-time	Used by Node.js to forward control commands to ESP32s over /command?action=...
JavaScript fetch()	Frontend HTTP client	Built-in, async support, modern browser-compatible	Requires CORS setup	Used to retrieve leaderboard data from the Node.js API on page load
Axios	HTTP client for Node.js	Cleaner syntax, async/await support, robust error handling	Slightly larger than native fetch()	Used in Node.js server to send HTTP requests to ESP32s for control forwarding
MJPEG over HTTP	Streaming technique (video)	Browser-compatible, simple to implement	High bandwidth usage, not compressed efficiently	Used in Flask to stream JPEG frames to the play page's video feed
TCP/IP	Transport/Internet layer protocol	Underlying protocol for all network communication	Not used directly, but all higher-level protocols depend on it	Provides the transport foundation for Wi-Fi, WebSocket, HTTP, and video streaming

- **Wi-Fi** enables wireless communication between all components in your system, including the ESP32 modules, Flask server, Node.js server, and the frontend website.
- **WebSocket** is used for real-time, bidirectional communication between the browser and the local Node.js server, allowing low-latency control of the cars and instant lap updates.
- **HTTP GET** is used by the Node.js server to send control commands to the ESP32 cars by requesting specific actions through simple URLs like /command?action=move_forward.
- **JavaScript fetch()** is used in the frontend (leaderboard page) to request and display the top lap times from the server's RESTful API upon page load.
- **Axios** is a Node.js HTTP client used in the server code to send commands to the ESP32s via HTTP, offering clean syntax and reliable error handling.
- **MJPEG over HTTP** is used by the Flask server to stream continuous JPEG frames from the ESP32 camera to the browser via an <img> tag in the play page.
- **TCP/IP** underlies all network communication in the project, serving as the transport layer for Wi-Fi, WebSockets, HTTP, and video streaming between devices.

Wired Communication Protocols

Mode	Type	Strengths	Weaknesses	Best Use in Project
I2C	Serial, synchronous, multi-master	Simple 2-wire design, supports multiple devices on one bus	Slower than SPI, limited cable length, requires addressing	Used to interface with the OV2640 image sensor and to drive the LED matrix at finish line
SPI	Serial, synchronous, full-duplex	Very fast, good for high-speed peripherals, no addressing needed	More wires (4+), no built-in device addressing	Used for high-speed communication between Pixy2 camera and finish line ESP32
PWM	Signal modulation	Great for precise analog-like control using digital pins	Not a communication protocol; limited data capability	Used to control motor speed and direction in the motor driver on each car's PCB
USB	Serial or parallel (host/device)	Standardized, high bandwidth, plug-and-play, supports power	Complex protocol stack, host-device hierarchy required	Used via USB-to-UART converter to program the ESP32 embedded on each car's PCB
UART	Serial, asynchronous, point-to-point	Simple, reliable, widely supported, long-distance capable	One-to-one only, no clock line (timing critical)	Used by USB-to-UART converter for programming ESP32 microcontrollers

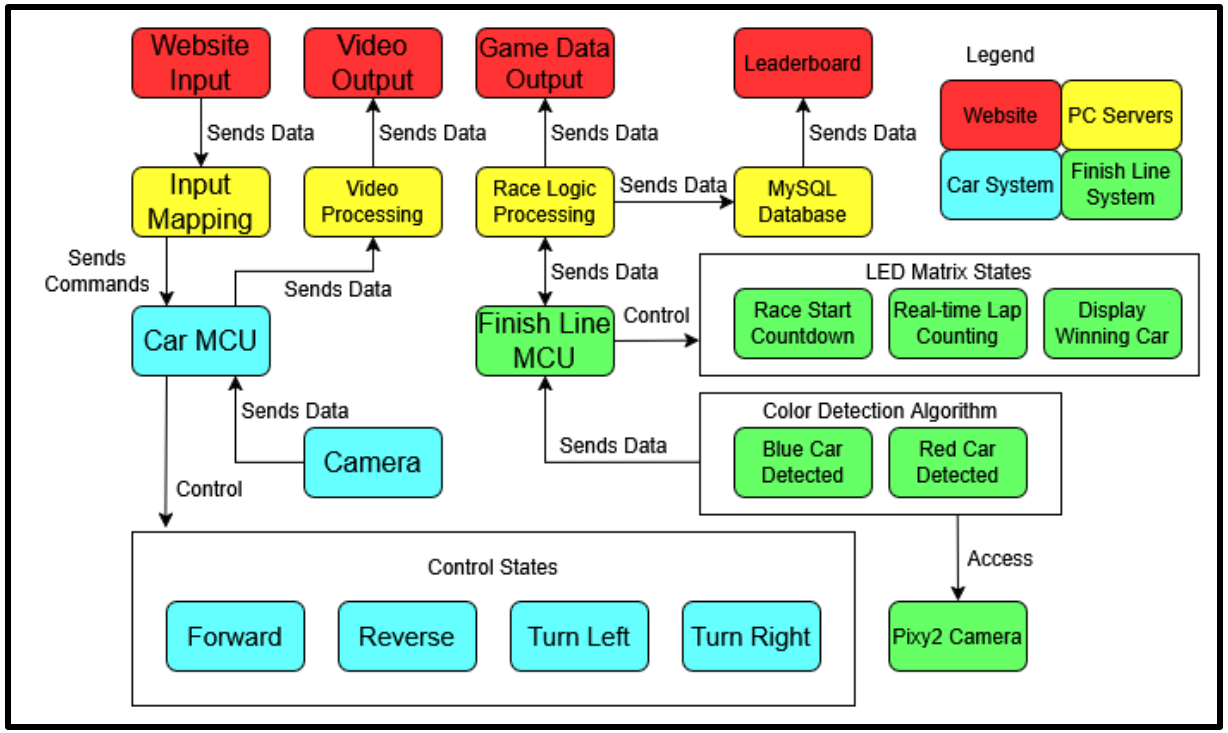
- **I2C** is used to communicate with the OV2640 image sensor on each car for capturing video and to send display data from the finish line ESP32 to the LED matrix. Its two-wire design makes it ideal for low-speed, short-range communication with multiple devices.
- **SPI** is used between the Pixy2 camera and the finish line ESP32 to enable fast, reliable color detection through high-speed data transfer. This allows the system to quickly identify car colors as they cross the finish line.
- **PWM** is used to control the motor drivers in each car's PCB, adjusting the speed and direction of the motors through varying pulse widths. It allows smooth and precise control of vehicle movement using standard digital output pins.
- **USB** is used indirectly through a USB-to-UART converter to program the embedded ESP32 microcontroller in each car. This provides a convenient and standardized interface between a computer and the ESP32 development environment.
- **UART** is the underlying protocol used by the USB-to-UART converter to transfer program data to the ESP32. It allows for serial, asynchronous communication with minimal wiring and is ideal for programming or debugging microcontrollers.



Communication Network Diagram

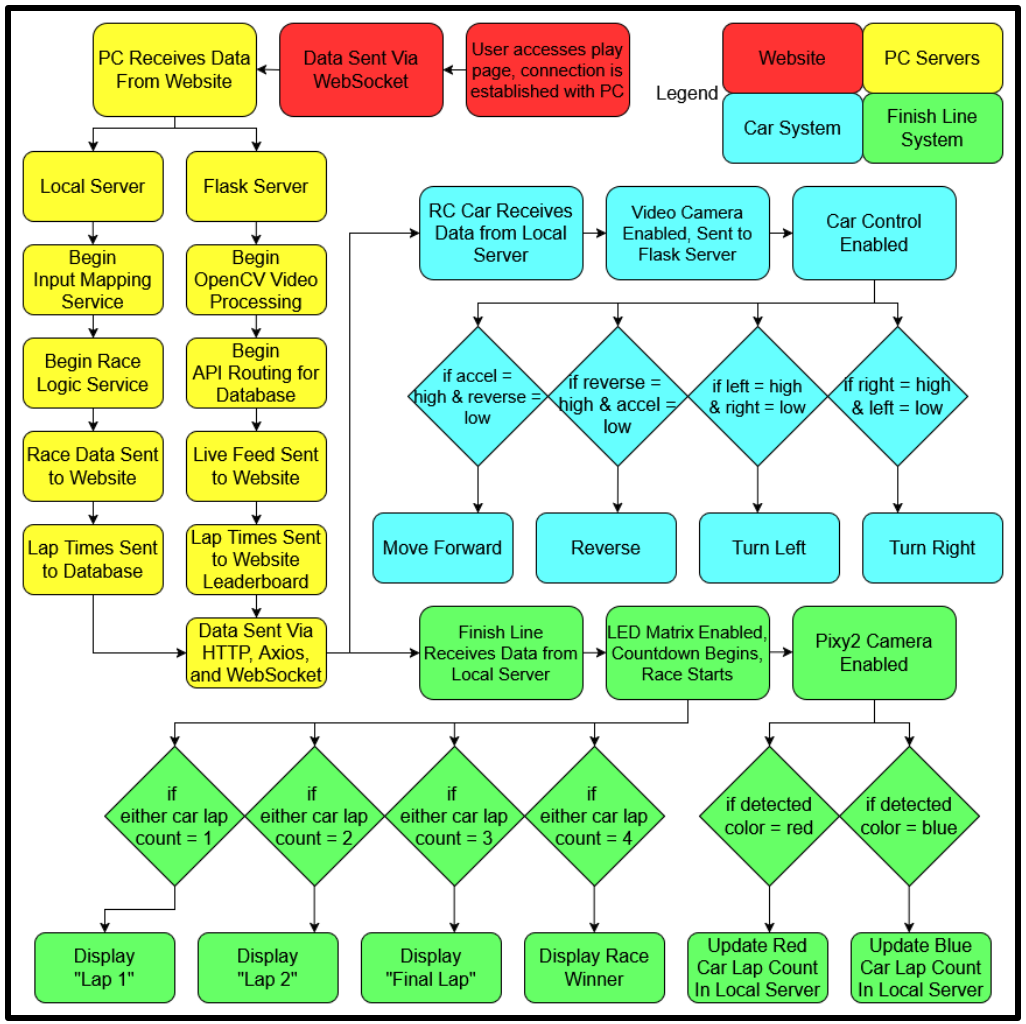
RACE System Software Design

Software Functionality



- The software system is built as a web-based RC car racing platform controlled through a browser.
- It includes a hosted website with three pages: Home, Play, and Leaderboard.
- Users connect through the Play page, are automatically assigned a car, and control it in real time using keyboard inputs.
- A Node.js local server manages car control, lap tracking, database interaction, and race logic.
- A Flask server handles live video streaming from the ESP32 camera mounted on the car.
- A MySQL database stores lap time records, which are displayed on the Leaderboard pag
- WebSocket and HTTP communication enabl real-time interactions between users, server and hardware.

Software Design



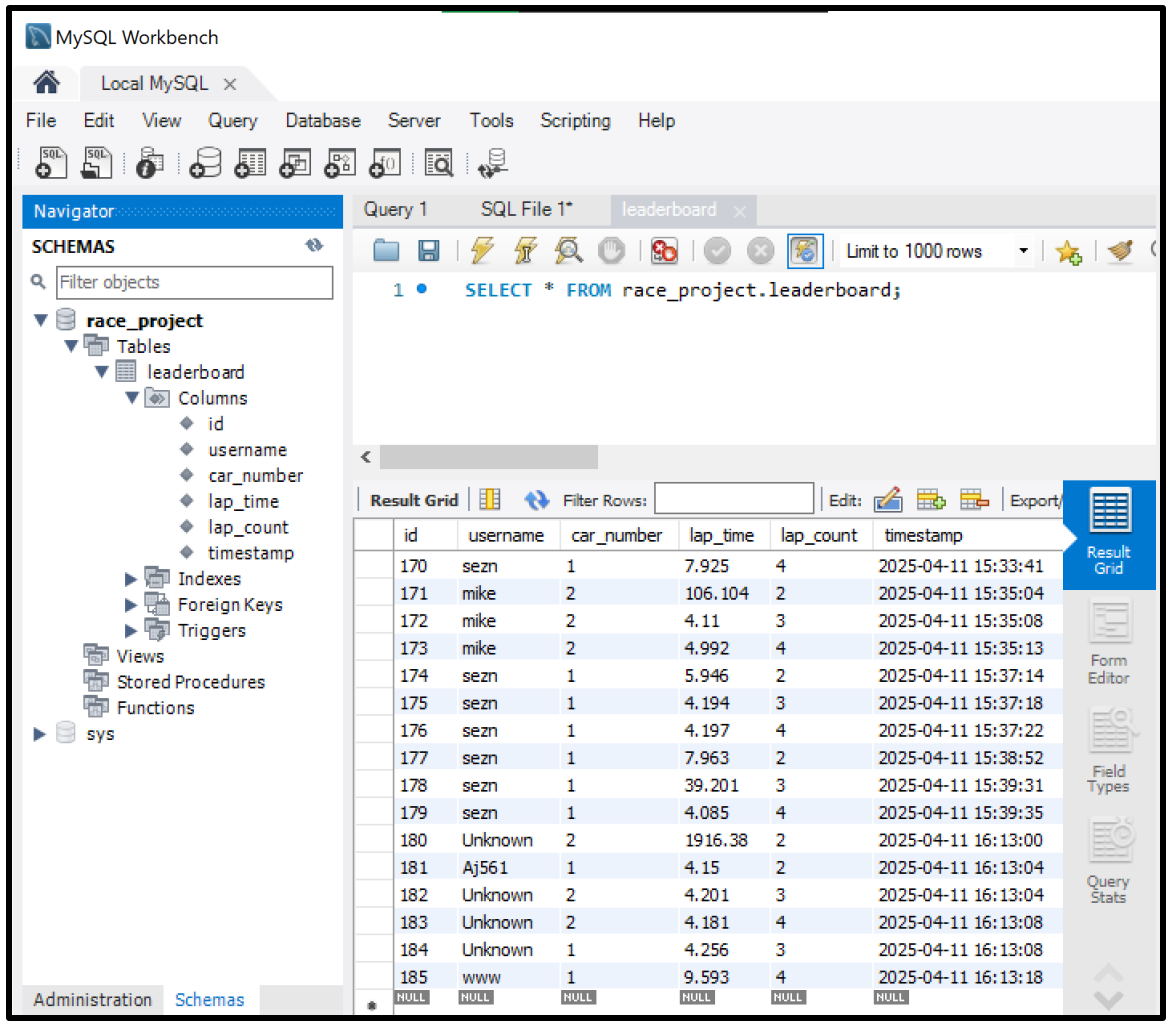
Local Server

- The local server is written in Node.js and runs on port 5001.
- It handles WebSocket communication with the website and receives control commands from each user.
- It forwards those commands to the ESP32 cars using HTTP requests.
- It tracks lap counts, lap times, and determines the race winner.
- The server inserts lap time records into the MySQL leaderboard table after each completed lap.
- It also broadcasts lap updates and the race winner to all connected clients.

Flask Server

- The Flask server is written in Python and runs on port 5000.
- It streams video from the ESP32 camera by repeatedly fetching images via HTTP.
- The images are decoded using OpenCV and streamed as MJPEG to the play page.
- Only the play page uses the video stream; the home and leaderboard pages do not.

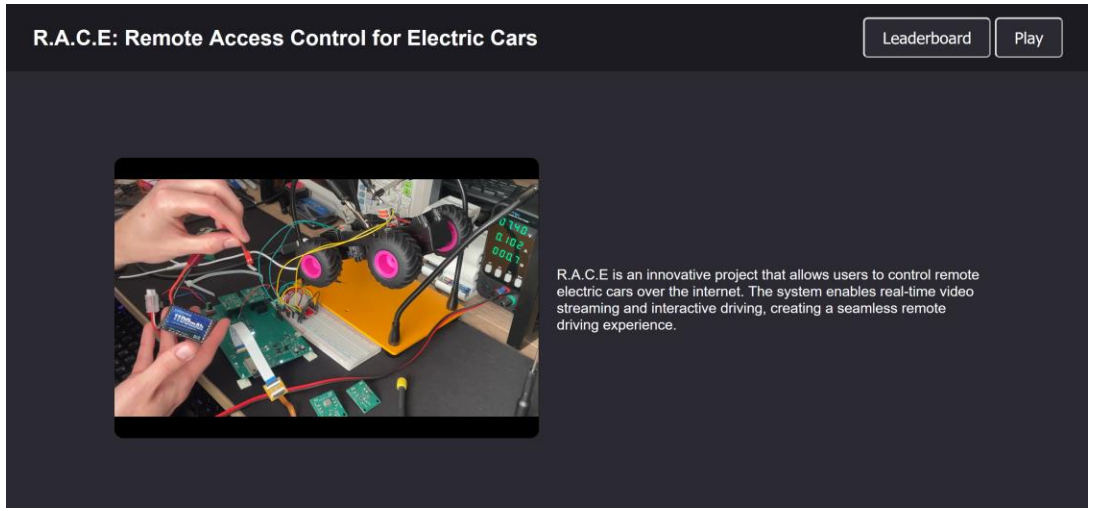
Database



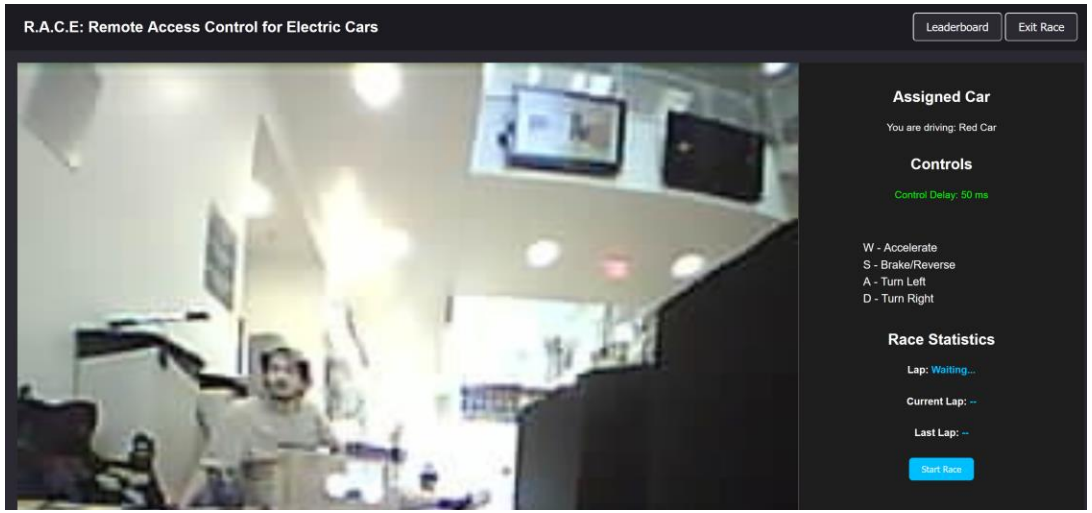
id	username	car_number	lap_time	lap_count	timestamp
170	sezn	1	7.925	4	2025-04-11 15:33:41
171	mike	2	106.104	2	2025-04-11 15:35:04
172	mike	2	4.11	3	2025-04-11 15:35:08
173	mike	2	4.992	4	2025-04-11 15:35:13
174	sezn	1	5.946	2	2025-04-11 15:37:14
175	sezn	1	4.194	3	2025-04-11 15:37:18
176	sezn	1	4.197	4	2025-04-11 15:37:22
177	sezn	1	7.963	2	2025-04-11 15:38:52
178	sezn	1	39.201	3	2025-04-11 15:39:31
179	sezn	1	4.085	4	2025-04-11 15:39:35
180	Unknown	2	1916.38	2	2025-04-11 16:13:00
181	Aj561	1	4.15	2	2025-04-11 16:13:04
182	Unknown	2	4.201	3	2025-04-11 16:13:04
183	Unknown	2	4.181	4	2025-04-11 16:13:08
184	Unknown	1	4.256	3	2025-04-11 16:13:08
185	www	1	9.593	4	2025-04-11 16:13:18
NULL	NULL	NULL	NULL	NULL	NULL

- The database is managed through MySQL Workbench.
- It contains a single table called leaderboard for storing lap time data.
- Each record includes an auto-incrementing id, the username of the player, and the car_number (1 for Red, 2 for Blue).
- It also stores the lap_time in seconds, the lap_count (1–4), and a timestamp of when the lap was completed.
- The leaderboard is sorted by lap_time in ascending order to show the top 50 fastest laps.
- The data is accessed via a Node.js API endpoint at /api/leaderboard.
- New lap records are inserted into the database by the local server each time a car completes a lap.

Website Design



Home Page



Play Page

Username	Car Color	Lap Time	Lap	Timestamp
sean	Red	3.32s	2	4/11/2025, 12:57:20 PM
sean	Red	4.30s	3	4/11/2025, 12:57:24 PM
sean	Red	5.85s	2	4/11/2025, 12:58:15 PM
sean	Red	6.19s	4	4/11/2025, 12:58:29 PM
sean	Red	7.39s	3	4/11/2025, 12:58:23 PM
sean	Red	12.86s	4	4/11/2025, 1:00:03 PM
sean	Red	13.68s	3	4/11/2025, 12:59:50 PM
sean	Red	15.14s	2	4/11/2025, 12:59:36 PM

Leaderboard Page

Hardware & Software Testing

Internet-Controlled Car Testing

Live Video Streaming

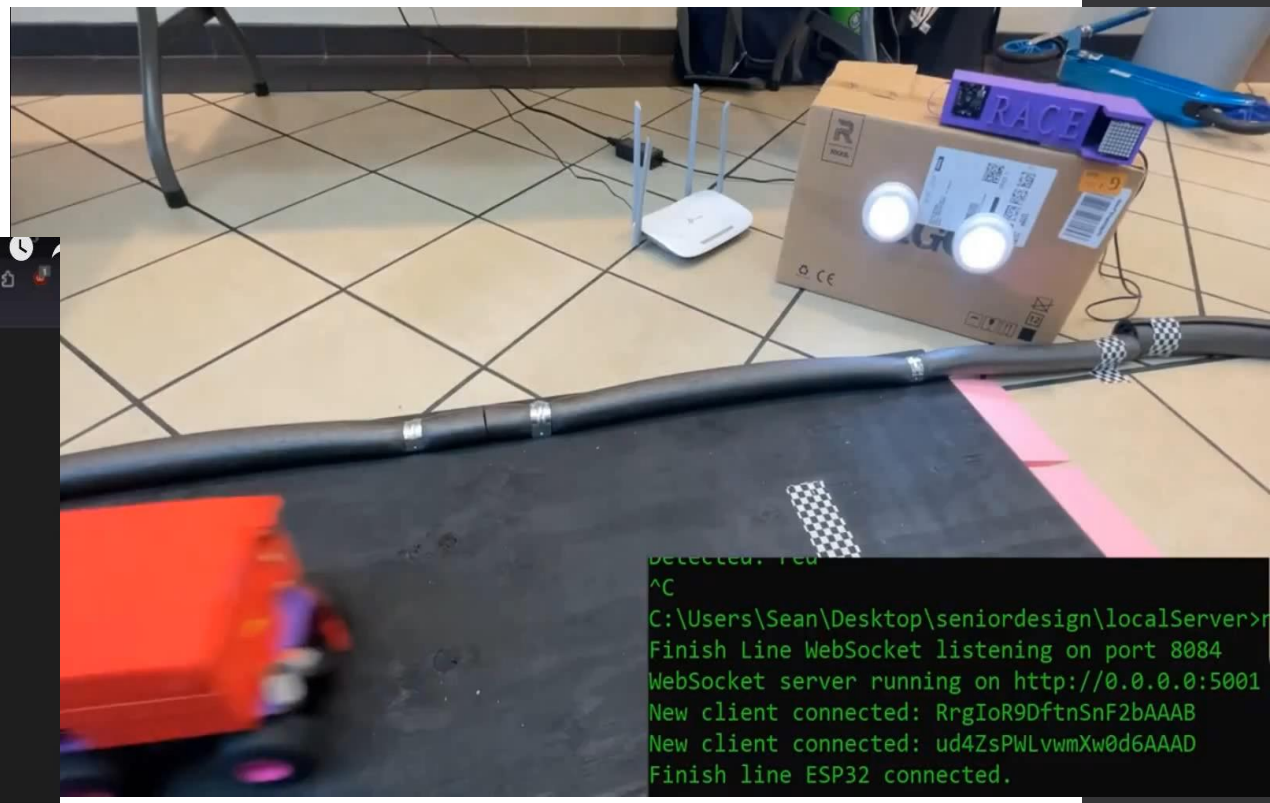
- A Flask server runs alongside the control server to handle video.
- The ESP32-CAM captures video frames and sends them over HTTP.
- Flask hosts the stream, which users can view live in their browser.

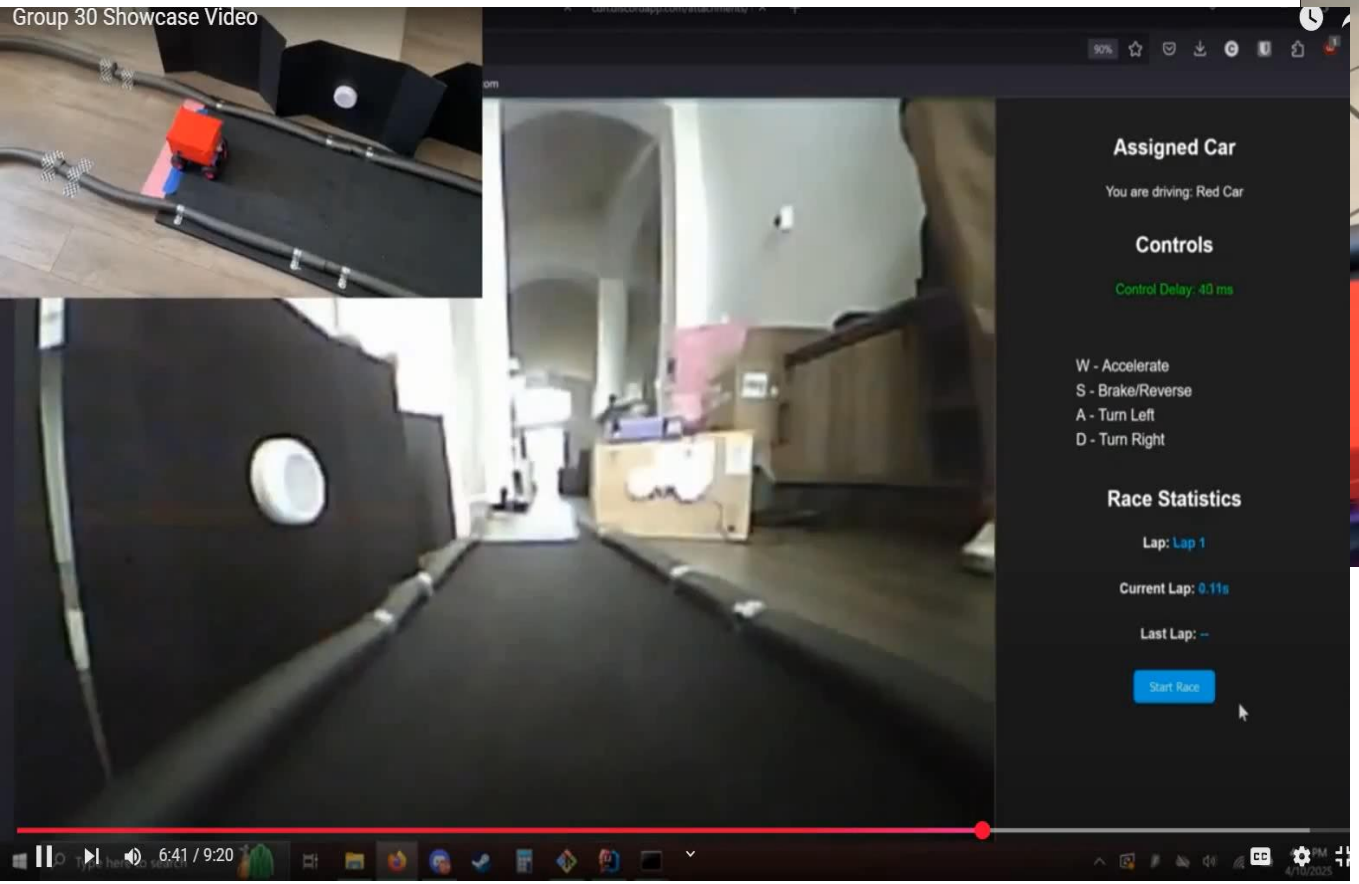
Control System Overview

- Key presses/releases are sent to our Node.js server via WebSockets.
- The server interprets input and sends one of six commands to the ESP32:
  - **move_forward**, **move_backward**, **stop**, **left**, **right**, **stop_turn**.

```
Command for 192.168.1.140: move_forward
ESP32 192.168.1.140 Response: Moving Forward
Command for 192.168.1.140: stop
ESP32 192.168.1.140 Response: Stopping
Command for 192.168.1.140: turn_left
ESP32 192.168.1.140 Response: Turning Left
Command for 192.168.1.140: stop_turn
ESP32 192.168.1.140 Response: Stopping Turn
```

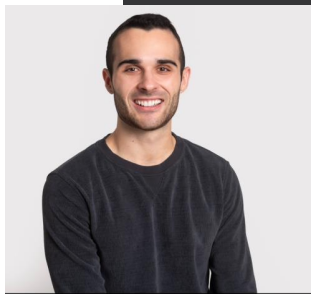
Finish Line System Testing





Challenges & Recommendations for Improvements

- Communication Network Challenges
  - Our project is not designed to be setup in public places, in-person demoing at UCF could only be accomplished through local connection.
- Car's Video Streaming Setup Challenges
  - Very difficult to implement with websockets. We were able to get it working after much trial and error.
- Finish Line Camera System Environment Challenges
  - Pixy 2 camera is VERY sensitive to lighting changes in natural environments, would recommend different camera product for more customization or different detection method for more reliable results in environments where natural lighting is out of our control.
- Improvements in Car's Mechanical Design
  - Recommend assistance from a mechanical engineering student to create a better chassis and structural composition for the cars, enabling smoother cars control.
- PCB Improvements
  - Recommend incorporating a battery monitoring system with a dedicated IC to measure both voltage and current, and to incorporate protection against undervoltage conditions.



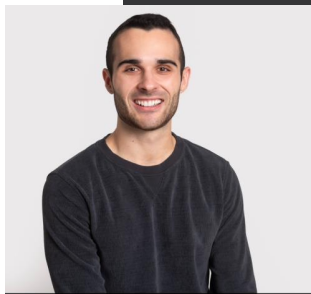
Administration & Mangement

Budget

Component	Part Description	Quantity	Cost
Cars System			
RC Toy-Grade Car	Car Chassis and DC Motors	2	$30
CMOS Image Sensor	OV2640 Image Sensor	2	$10
Battery Supply	7.4 V Li-Po Battery, 1100 mAh Capacity, with Charger and Adapters	2	$30
Car's Structure(s)	PCB Platform and Enclosure on Car Chassis	2	~$20
Cars PCB			
Microcontroller	ESP32-WROVER-E	2	~$6
Motor Driver	DRV8833 Dual Motor Driver	2	~$4
Antenna	(Included in ESP32)	2	$0
3.3 V Regulator	TPS561208	2	<$1
5 V Regulator	TPS563231	2	<$1
LDO Regulators	Torex XC6206 Series	4	<$2
UART-USB Bridge	CP2102N-A02-GQFN28	2	~$9
Camera Connector	Amphenol F32R-1A7H1-11024	2	<$2
On/Off Switch	500SSP1S2M2QEA Slider Switch	2	~$7
Fuse	Glass Cartridge Fuse	2	<$3
Circuit Boards	JLCPCB Fabricated	8	~$10
Other PCB Materials	Resistors, Capacitors, Inductors, Connectors, Transistors, Diodes, LEDs, etc.	2	~$20
Finish Line System & Racetrack			
Sensor/Camera	Pixy 2 Camera	1	$60
LED Matrix	Adafruit Bi-Color LED Square Pixel Matrix	1	$15
Freenove Control Board	V5 Rev4 WiFi, Arm Cortex-M4 Microcontroller with Onboard ESP32-S3	1	$20
Finish Line Enclosure	3D Printed Enclosure	1	~$10
Racetrack Materials	Baseboard, Boundaries, Duct Tape, Lighting, etc.	1	~$60
Combined Total Cost			~$322

Work Distribution

Team Member	Key Responsibilities
Sean Chaney	Software Design & Implementation
	Website Design & Implementation
	Server Management
	Communications Implementation
Michael Mallette	Cars Design & Assembly
	Motors Implementation & Video Transmission
	Communications Implementation
	Software Design & Implementation
Jesvany Colon	Finish Line Design & Assembly
	Finish Line Camera Implementation
	Racetrack Implementation
	Software Design & Implementation
Austin Silva	Project Management & Administration
	PCB Design & Implementation
	Power Supply Unit Implementation
	Cars Design & Assembly
	Support Finish Line Design & Assembly



Thank You!