Mario Kar

Caitlyn Fong, Enrique Gomez, Nicolas Keck

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— Mario Kar is a novel teleoperation system featuring a wearable glove Kontroller and a physical RC Kar, designed to enhance driver immersion for racing and driving enthusiasts. The system achieves a highly intuitive experience by translating natural hand gestures, measured by an Inertial Measurement Unit (IMU), into precise vehicle commands. To complete the feedback loop, real-time haptic feedback is relayed from the Kar to the glove, allowing the user to feel the remote environment. The implementation is built upon custom-designed PCBs for both the Kontroller and Kar , which integrate all necessary sensors and motor Kontrollers for a seamless, low-latency control system.**

***Index Terms*—PCB, STM32, ESP32, RC, wearable device, haptic feedback, IMU, Kalman Filter**

I. Introduction

Kart racing or driving games are one of the most popular subgenres of video games that trade realistic driving simulation for a more accessible, arcade-style experience. These games typically feature go-kart-style vehicles on fantastical tracks and emphasize chaotic, item-based gameplay. The most prominent example is Nintendo's Mario Kart franchise, which has become a global phenomenon, selling nearly 200 million copies worldwide.

In Mario Kart, players typically use small controllers like the Joy-Con, which often compromise ergonomics and intuitive control for versatility. Many of these controllers are often small, have imprecise analog sticks, and persistent technical flaws like "stick drift". While functional, these design choices can create a disconnect between a player's intentions and the on-screen action, sacrificing the comfort and precision that these games demand.

To address these limitations, Mario Kar draws from established research in human-robot interaction and virtual reality. Studies consistently show that gesture-based controls are among the most intuitive and immersive methods for steering a vehicle. This approach is highly flexible, accommodating different users' hand sizes and control preferences. Furthermore, research demonstrates that integrating haptic feedback, using vibrations to simulate road texture or collisions, significantly improves a user's perception and situational awareness of the remote environment. [1]

A key advantage of this approach is the reduction of cognitive load, which is the mental effort required to operate a system. Traditional teleoperation often demands the operator's full attention, as complex or unintuitive controls can overwhelm working memory and increase the risk of error. In contrast, a more "hands-free" gesture system allows for more natural control, freeing up mental resources and leading to better performance and environmental recall. [2]

Mario Kar is a haptic-gesture Kontroller and physical RC Kar that aims to translate these principles into a real-world driving experience. By using hand orientation as the primary steering input, we are leveraging the most natural controller available: the human hand. The system will provide real-time vibrotactile feedback to the user, allowing them to feel the Kar 's interaction with its environment. The goal is to eliminate the abstract layer of joysticks and buttons, delivering a seamless, responsive, and deeply immersive experience to the gesture driver seeking a more intuitive sense of control, and the spectator driver who enjoys the thrill of watching the race.

II. Use-Case Requirements

The use-case requirements for our product can be divided between requirements for the Kar and the Kontroller. For example, if we want to provide the best driving experience for our user, we want the Kontroller to be as unobtrusive as possible. For that reason, two of our use-case requirements are that the Kontroller must be wearable to enable touchless control of the Kar, and that the Kontroller is lightweight (~60-80g) so that it is not burdensome for the user to wear. We also want to ensure that the user has a seamless connection to the Kar, ensuring an uninterrupted driving experience. To this end, another use case requirement is that we implement an intuitive mapping between user gestures and Kar motion; our end user will expect immersive driving, which will not be achieved if the control scheme is highly obtuse. Reliable wireless communication up to 10m from the user to the Kar is also a use-case requirement, if the user was tethered to the Kar the control experience would be too obtrusive, and if the communication was unreliable, the immersiveness of the experience for the user would be shattered. However, wireless communication introduces latency into our system, and hence we require that our end-to-end communication latency be <50ms again to ensure an immersive experience for our end user. To bolster the immersive experience provided by the system, we are also requiring a haptic feedback system to be implemented which can provide the user feedback for turns and collisions. This will allow the Mario Kar system to provide even greater immersion than a traditional controller, as our system provides both haptic feedback and hand gesture control. Finally, on the topic of safety, we are requiring that both the Kar and the Kontroller are able to be put into a reset state with a simple user input and that the maximum latency before all components of the system are in the reset state is within 50ms of the original input.

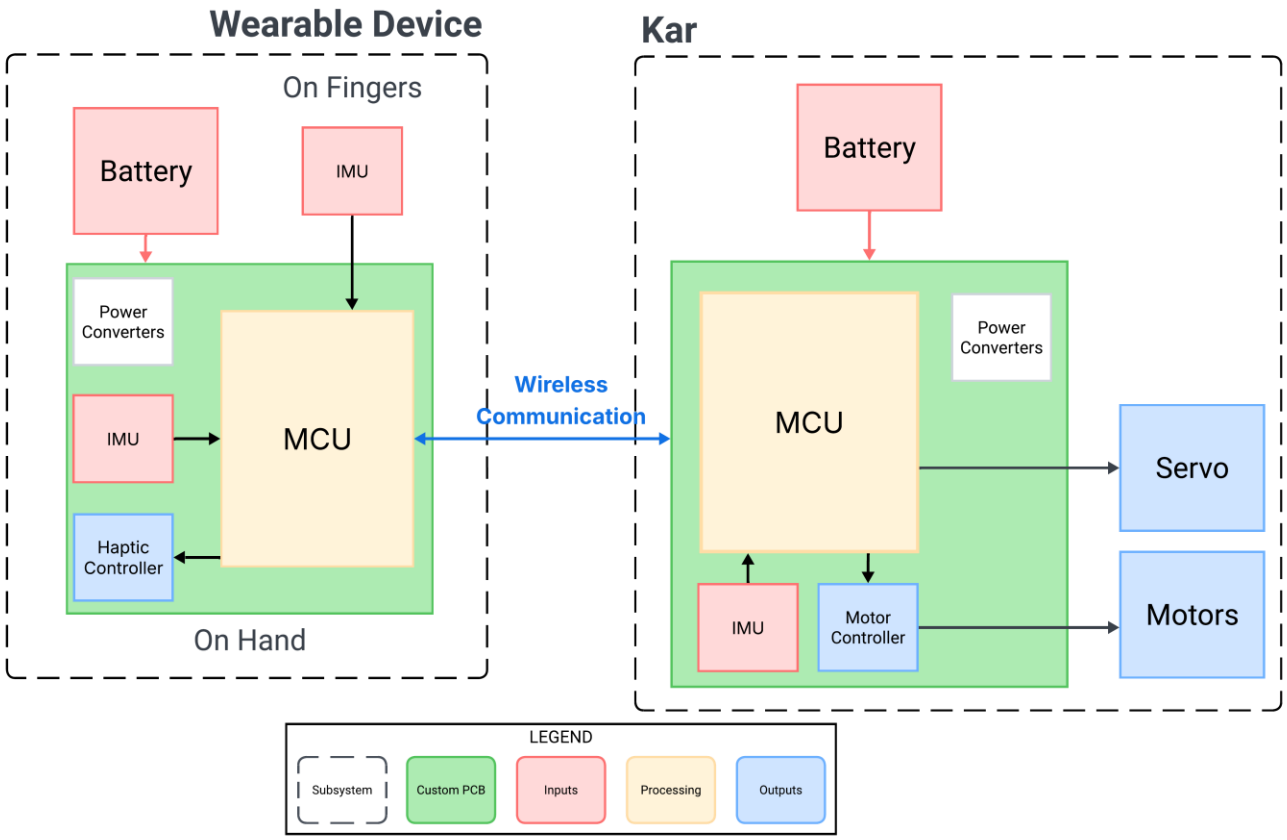

FIGURE 1: HIGH-LEVEL BLOCK DIAGRAM

A. Components in the Overall System

Mario Kar consists of two main subsystems: the wearable device and the Kar. These subsystems communicate through a wireless connection. Each subsystem processes sensor measurements as inputs and produces outputs accordingly. The overall system block diagram is shown in Figure 1. All hardware components, except for the custom printed circuit boards (PCBs), will be purchased off the shelf. The PCBs will be custom-designed and fabricated specifically for this project.

B. Wearable Device System

The wearable device continuously collects data from two inertial measurement units (IMUs) within the subsystem. The first IMU, mounted on the user's fingers, will capture forward and backward linear motion. The second IMU, mounted on the user's hand, will detect rotational movements for wheel control. Both IMUs will use the same chip, with one integrated directly onto the custom PCB and the other mounted on a purchased breakout board.

The custom PCB will include the microcontroller unit (MCU), the second IMU, a haptic motor controller, and supporting power conversion circuitry. It will also supply power to the IMU breakout board, which will return sensor data to the MCU. The MCU will process and filter the IMU data to extract meaningful motion signals before transmitting the relevant information wirelessly to the MCU on the Kar.

C. Kar System

The Kar subsystem includes a custom PCB and external actuators. The PCB will feature an MCU, an on-board IMU for measuring the Kar's acceleration and rotation, a motor controller for driving the external motors, and an interface to a servo motor used for steering. Additional power regulation components will manage different voltage levels required across the system.

The MCU on the Kar will communicate wirelessly with the MCU on the wearable device. It will also process IMU data locally to interpret the Kar's motion and send corresponding feedback signals to the haptic motor controller on the wearable device, enabling responsive haptic feedback for the user.

IV. Design Requirements

Just as with use-case requirements, design requirements can be split between Kar and Kontroller requirements. For the Kontroller, we specified in the use-case section that the overall control system must weigh less than 80g. To this end, two of our downstream design requirements are that the controller batteries must have high power density to minimize battery weight, and that our Kontroller will consist of a custom PCB in order to allow for more precise mass control. Downstream of our Kontroller wearability requirements, we derive that the Kontroller PCB shall be <3in^2 and that the Kontroller

electronics shall be unobtrusive to the user when wearing the Kontroller glove; these combined contribute to ensuring that the Kontroller is neither unwieldy nor heavy, streamlining the control experience for the user. In order to meet our intuitive control mapping use-case requirement, we derive requirements for a direct correlation between finger tilt angle and Kar velocity, and hand rotation and Kar turning angle. These correlation requirements inform extra level 2 design requirements regarding accurate recording of both the angle of finger tilt, and the angle of hand rotation. Both of these must be recorded to within ±1° for proper mapping of gestures to Kar movement.

In terms of requirements which overlap between the Kar and Kontroller, we first derive the need for our Kar and Kontroller microcontrollers to natively support wireless protocols. Depending on what microcontrollers we select, this may be satisfied in the form of our microcontroller having built-in antennas, but it is also possible that it will be required to fit external antennas to these microcontrollers such that they may support wireless communication. For meeting our 50ms latency use-case requirement, we need to ensure that all of our processing steps in our critical path have very low latency; from this we derive that each processing step on the critical path shall have <5ms of latency each. The requirement of providing haptic feedback for the user leads to many design requirements. Firstly, it implies that we must communicate collision and turn data from the Kar to the Kontroller, meaning that the Kar-Kontroller communication must be two-way. Additionally, it means that we must collect collision and turn data in some way, which will be handled by including an IMU and magnetometer on the Kar. Finally, it implies that we must include a suitable haptic controller on the Kontroller to close the loop with the user.

In the case of safety, we must implement several design features in order to properly protect the user. To begin, seeing as both the Kontroller and the Kar must allow the user to provide a reset input, we know that our design must include a reset button or switch for both the Kar and Kontroller. Additionally, we are requiring that the Kontroller includes a set of gestures which are also capable of setting the Kar into the idle state, to provide extra safety options to the user. Because of the additional requirement that the idle state be reached from within 50ms of triggering the safety mode, it is also necessary that we implement the reset logic such that it bypasses the normal driving logic of the Kar; it does not make sense to prioritize driving fidelity over safety.

V. Design Trade Studies

A. Communication Protocol Tradeoffs

Communication between the Kar and the Kontroller must be low-latency for the user to perceive the interaction and movement as "smooth," and it must also satisfy a range requirement so the user can control the Kar freely on a track.

Because long cables would be impractical for controlling long-range movement, we considered 2 primary forms of wireless communication that are most commonly supported in IoT microcontrollers – BLE (Bluetooth Low Energy) and Wi-Fi 802.11n. As noted in Table I, Wi-Fi can typically support much higher ranges +100 m and can support data rates of up to 1 Gbps. On the other hand, BLE's range is limited (usually around 10 m), and has a much lower data rate up to 2 Mbps. While these Wi-Fi specs seem more desirable compared to BLE, we only need to transfer a few bytes, at a maximum of 100 times per second. For example, the Kontroller needs to send 3D rotational velocity and linear acceleration data to the Kar for movement. Assume the 3D coordinates are 24 bytes total (4 bytes for each axis for both velocity and acceleration). Then the data rate cost for the payload of our data sent wirelessly would be less than 2.5 Kbps. Therefore, the much higher data rate that Wi-Fi offers would be unnecessary. Furthermore, Wi-Fi uses significantly more power than BLE. Since low weight and sufficient battery life is a crucial design requirement, this makes BLE more desirable and practical. Additionally, as we'll discuss in a later section, BLE is already supported out-of-the-box on the microcontrollers that we are planning to use.

Next, we have to consider how we are going to communicate with the sensors on both the Kar and the Kontroller. We will be using IMU sensors to measure rotational velocity and linear acceleration. Therefore, we need some sort of embedded communication protocol. 3 of the most common protocols include UART, I2C, and SPI. Most IMUs that we researched, which we discuss in a later section, don't support UART communication. Instead, most support either I2C and SPI. Because we're using multiple of the same IMU sensor on the Kontroller, it wouldn't make sense to use I2C because they share the same slave address. This leaves SPI as our only communication protocol for the IMU sensor. However, the Kontroller contains a haptic feedback motor which only supports I2C communication. Therefore, it makes sense to use both I2C and SPI for wired communication in our design. Table I below demonstrates some other tradeoffs. The protocols selected for our design are shaded green.

TABLE I. COMMUNICATION PROTOCOL COMPARISONS

	Communication Protocols				
	Bluetooth 5 (BLE)	***Wi-Fi (802.11n)***	***I2C***	***SPI***	***UART***
Wireless	Yes	Yes	No	No	No
Latency	5-20 ms	1-10 ms	< 1 µs	< 1 µs	1-5 µs
Range	< 100 m	100+ m	< 1 m	< 1 m	< 1-2 m
Power	Low	High	Low / Moderate	Low	Low
# of sensors compatible	—	—	1 (Haptic)	3 (IMU sensors)	0

B. Microcontroller Design Tradeoffs

When researching microcontrollers, we mainly kept the following metrics in mind: cores and performance, wireless communication support, memory storage, power profile, and prior team experience. We primarily focused on 4 microcontrollers: STM32WB55, ESP32-S3, NordicnRF52840,

and Raspberry Pi 4. We specifically researched these because they're known for their IoT capabilities.

For our Kar, we are prioritizing a low-power profile, because it should ideally run for a couple of hours. The STM32WB55 has a variety of low-power modes that make handling energy efficiency easier. The Nordic nRF52840 also offers a low-power profile, while offering multiple wireless communication protocols. However, the STM32 has a wider base of general-purpose support and documentation, and most of the team members have had experience writing code for it. Therefore the STM32 becomes a clear choice for the Kar here.

For the Kontroller, we are prioritizing compute capability and P2P server stability for BLE communication. Although the Raspberry Pi family microcontrollers have more cores than the other options and also support Wi-Fi and BLE, it has a significantly high power-profile. Additionally, the standalone Raspberry Pi SoC is not available for purchase, so the standard board's form factor would make it unfit for use on the Kontroller. The XIAO ESP32-S3 offers a relatively power-profile, and it has a dual-core 240 MHz processor with on-chip Wi-Fi and BLE. It also has extensive online support and documentation, and a small form factor, making it the clear microcontroller choice for our Kontroller design. More tradeoffs are covered in Table II below.

TABLE II. MICROCONTROLLER SELECTION

	Microcontrollers			
	STM32WB55	***ESP32-S3***	***Nordic nRF52840***	***Raspberry Pi 4***
Core & Performance	Cortex-M4 @ 64 MHz, Cortex-M0 @32 MHz	Xtensa dual-core @ 240 MHz	Cortex-M4F @ 64 MHz	Quad-Core Cortex-A72 @ 1.5 GHz
Wireless	BLE 5.0	WiFi + BLE	BLE 5.0, Thread, Zigbee	WiFi + BLE
Memory Storage	1 MB Flash / 256 KB RAM	16 MB MB Flash / 512 KB RAM	1 MB Flash / 256 KB RAM	1-8 GB RAM + SD card
Power Profile	Ultra-low	Low-moderate	Ultra-low	High
Prior Team Experience	High	High	Medium	Medium

C. IMU Selection Tradeoffs

There are 3 commonly used IMUs that we researched for our design. Since we all had heard about the MPU-6050 before, we looked into that one first. However, we quickly found out that it has been considered obsolete and should not be used in new designs. The MPU-6050 is no longer in production by the original manufacturer, TDK InvenSense, and is listed as "obsolete" by suppliers like DigiKey.

Two other popular alternatives we found were the ICM-42670 and the BMI330. While the BMI330 has a higher max data rate, it was released much later (2025) than the ICM-42670 (2021). Additionally, the ICM-42670 has had more revisions to its data sheet, and there is more support and discussions about its use in forums. Therefore, we selected the ICM-42670 for its reliability and improved documentation. More tradeoffs can be found in Table III.

TABLE III. IMU SELECTION

	IMUs		
	MPU-6050 [4]	***ICM-42670 [5]***	***BMI330 [6]***
Cost	$6.99	$3.66	$4.50
Protocols	I2C	I2C, SPI	I2C, I3C, SPI
Max Date Rate	8kHz	1.6kHz	6.4kHz

D. Motor Driver Tradeoffs

The COTS car we selected to form the basis of our Kar subsystem uses a brushed DC motor for driving. In order to allow for control of the motor, we need to select our own brushed DC motor controller for inclusion on our PCB.

As can be seen in the table below, we considered 3 primary options for our motor drivers. Two options are from Texas Instruments, and one is made by Analog Devices. When it comes to output current, all ranges for all devices are acceptable, as all currents are much larger than we would need to drive our motor for the Kar. In terms of input voltage, all ranges are also acceptable, although the 6.5V input for the DRV8871 would require a larger battery voltage than the others. The major differentiator between the different drivers is that the DRV8242 is the only one which supports a SPI interface. This makes it a very desirable choice over the other drivers, as we already have a SPI bus for communicating with our IMU on the Kar; the fact that it is the cheapest option out of the 3 makes it the obvious best choice for our project.

TABLE IV. MOTOR DRIVER SELECTION

	Motor Drivers		
	DRV8242-Q1 [7]	***MAX22212 [8]***	***DRV8871 [9]***
Cost	$3.25	$4.41	$3.82
Interface	HW pins, or SPI (5V logic level)	HW pins	HW pins (Manual H-bridge signals)
Motor voltage	4.5V-35V	4.5-36V	6.5V-45V
Max Output current	6A	7.6A	3.6A

E. Level Shifter Tradeoffs

The one disadvantage of selecting the DRV8242 as our motor driver is that it only supports a minimum of 5V for logic level outputs. As our chosen microcontroller supports 3.3V logic levels, this necessitates the use of a level shifter to interface between the motor controller and the microcontroller. Although this is only really necessary on the MISO bus wire, as the DRV8242 is at minimum capable of interpreting 3.3V logic inputs, just not generating 3.3V logic outputs.

As can be seen below, the two options we considered are a 4-lane bidirectional level shifter and a single-lane monodirectional level shifter. The primary advantage of the single-lane level shifter is that it only needs to be connected to the output voltage level. This is convenient as it vastly reduces routing complexity. However, the TXB0104 has the advantage

of being more cost-efficient for the 4-lanes compared to implementing the SN74 four times. In the end, due to the DRV8242 only technically requiring the level shifter on the MISO wire, we decided to go with the SN74 as it decreases our routing complexity and is slightly cheaper for the single chip required.

TABLE V. LEVEL SHIFTER SELECTION

	Level Shifters	
	SN74LV1T34 [10]	***TXB0104PWR [11]***
Cost	$0.27	$0.91
Lanes	1	4
Bidirectional?	No	Yes
Supply Voltage	Single supply 1.6V-5.5V (Output voltage)	Dual supply 1.2V-3.6V and 1.65V-5.5V

F. Haptic Motor Driver Selection

The haptic motor driver is one of the few components where we decided to select a COTS solution for our system rather than selecting a discrete IC. This was done due to the team's lack of familiarity with haptic systems, and the limited complexity of our haptic system. Given our use-case and design requirements, we simply need to scale the intensity of our haptics with certain driving patterns. This requirement lends itself well to a COTS plug and play solution, which is exactly what the Adafruit DRV2605L breakout board provides. Not only does the board have a small form factor, which is useful considering our small board area requirement, but it interfaces directly with our ESP32 over I2C. Additionally, the boards can be acquired for free from Ideate and other on-campus sources, which helps drive down our project costs.

G. Battery Selection Tradeoffs

Lithium batteries are widely used in embedded systems because they offer high energy density, enabling smaller and lighter devices. They also provide a longer lifespan (higher cycle life), low self-discharge rates, and require virtually no maintenance.

Since we purchased an off-the-shelf Kar, we followed the manufacturer's battery recommendations and selected 18650 rechargeable lithium-ion batteries. These cells are commonly used in RC cars due to their cost-effectiveness, high energy density, and well-established safety record. Given our design requirement for the Kar to operate for approximately three hours, we determined that using four 18650 lithium-ion batteries in series and parallel combinations would provide the ideal balance between runtime and reliability. There were no limitations when it came to selecting the Kar's batteries given there was no weight requirement for the Kar. For the battery capacity of the Kar, we need at least a battery capacity calculated by the math below:

$$Avg.Current\ Draw, I = 3000mA \quad (1)$$
$$Min.Runtime = 3\ hours \quad (2)$$
$$Min.Capacity\ = I \times t \quad (3)$$
$$= 3000mA \times 3hours = 9000\text{mAh} \quad (4)$$

For the glove Kontroller, we had more flexibility in battery selection. The ESP32-S3 and the custom PCB components require a minimum operating voltage of 3.7V, so we opted for a lithium-based power source. While lithium-ion batteries typically offer higher energy density, we chose lithium-polymer (LiPo) batteries instead due to their lighter weight and flexible form factor, which better suit a wearable design. Because comfort and usability are prioritized over long battery life, we intentionally selected LiPo batteries with lower capacities to reduce the glove's overall weight. Although this requires swapping out batteries more frequently during extended use, it is a necessary trade-off to ensure the Kontroller remains lightweight and comfortable for the user. The minimum battery capacity for the glove Kontroller is calculated below:

$$Avg.Current\ Draw, I = 500mA \quad (1)$$
$$Min.Runtime = 1\ hours \quad (2)$$
$$Min.Capacity\ = I \times t \quad (3)$$
$$= 500mA \times 1hour = 500\text{mAh} \quad (4)$$

H. Power System Tradeoffs

Traditionally, there are two primary options for voltage regulators: linear regulators and buck regulators. The main tradeoff between the two is that linear regulators provide higher efficiency at low currents and low voltage drops, but buck regulators are vastly more efficient the higher the required voltage drop and the higher the current the system needs to support. Additionally, linear regulators provide lower voltage supply noise compared to the buck regulators, due to the lack of a switching element; this lower noise can be useful for powering digital components such as microcontrollers, as they generally have lower noise tolerance than other components.

Considering our component selections, we are required to select a 3.7V-3.3V regulator for our Kontroller PCB and we need to regulate 8.4V down to both 5V and 3.3V for our Kar PCB. Given that our voltage drop and load current are relatively low for our Kontroller, it makes sense just to implement a linear regulator for the small voltage regulation step needed. However, for our Kar PCB the drop from 8.4V to 5V and especially down to 3.3V is far too large to efficiently perform with a linear regulator. On the other hand, it would be preferred to supply all of our digital components at 3.3V with low-noise supply voltages. To solve both constraints, we can use a cascaded power system design where we regulate 8.4V down to 5V using a buck regulator and then regulate 5V down to 3.3V using a linear regulator. This essentially allows us to "get the best of both worlds," and solves our power supply design problem.

VI. System Implementation

A. Custom Glove PCB

As described in our design requirements, our Kontroller PCB will consist of a custom design created by the team. The schematic and the 2D layout of the completed Glove PCB can be referenced in the appendix. The PCB is two-layer and has dimensions of 1.7 x 1.6 inches, which meets our size design requirement. As outlined in our system diagram also seen in

the appendix, the PCB contains the relevant layout for the ESP32 on a Xiao breakout board, the 3.7V-3.3V linear regulator, one of our Kontroller IMUs, the battery connector for connecting to the 1S LiPo battery, the Adafruit DRV2605L breakout board, and a connector for connecting to the finger IMU we will be using for detecting finger tilt. The board also contains other supporting components such as a reset switch, accompanying resistors and capacitors for power management, and a variety of indicator LEDs for displaying power and Kontroller calibration status.

B. Kontroller IMUs and ESP32 Filtering

On the Kontroller we will start with straightforward FIR filtering to clean the raw accelerometer and gyroscope streams before any gesture logic. Sampling at 200 Hz, each axis will pass through a short, linear-phase low-pass FIR that preserves the shape of hand motion while attenuating sensor and quantization noise. As an initial target, a 12–24 tap Hamming-windowed design with a 15–20 Hz cutoff balances attenuation and latency; at 200 Hz this yields a group delay of roughly 30–60 ms, which we will compensate by timestamping samples at acquisition and accounting for the fixed delay in the control loop. A light high-pass on acceleration can be added for step or tremor rejection if needed, and a narrow notch can suppress any persistent mechanical resonance discovered during testing. Before filtering, we will remove gyro bias estimated at startup during a brief still window, and we will clamp obvious outliers to protect the FIR from transient spikes. The ESP32 will run the filters in a small ring buffer per axis to keep memory bounded, and we will validate the design with frequency sweeps and step responses so the chosen cutoff does not blunt intentional, fast wrist motions.

If the FIR stage does not deliver the needed stability, we will promote to a Kalman-based estimator that models gyro bias explicitly and fuses rate and tilt information. For low compute cost we will begin with a per-axis linear Kalman filter during gestures that are well approximated by small angles, with a state comprising angle and gyro bias and measurements coming from the filtered accelerometer tilt. If we later need full 3D attitude, we will switch to an AHRS-style extended Kalman filter operating on a unit quaternion and a three-axis bias state. In both cases the process and measurement noise will be tuned from logged data, using Allan variance to set reasonable gyro random-walk and bias-instability terms and using stationary segments to re-tighten bias. Zero-velocity or low-acceleration detections will act as soft corrections that slowly pull the estimate back when motion stops, which helps long-term drift without making the system feel sticky during rapid gestures. This staged approach lets us start with predictable latency and simple tuning, then add model-based fusion only if the data shows we need it.

C. Custom Kar PCB

While our COTS Kar platform comes with its own PCB, this PCB is primarily designed for interfacing with a Raspberry Pi or Jetson for controlling the Kar. Seeing as we have decided to use an STM32 instead, a custom PCB is needed to mount the Nucleo board and interface with the Kar platform's servo motor for steering and the primary BDC motor for driving. The top-level schematic and the 2D layout of the completed Kar PCB can be found in the appendix, as well as the system block diagram describing the overall PCB.

As seen in the system diagram, the PCB contains holders for the 18650 Lion batteries that we are using for powering the Kar platform, the 8.4V-5V buck regulator, the 5V-3.3V linear regulator, the DRV8242-Q1 motor controller IC, the SN74LV1T34DBVR level shifter required for communication between the STM32 and the DRV8242, the pin headers needed for the STM32WB55 nucleo board to mount on the PCB, and the ICM-42670-P IMU. The board also contains a 20-pin connector for connecting a JTAG debugger to the STM32, as it we will be unable to connect to the built-in STLINK IC on the nucleo board when we are providing it with 3.3V, a reset button for putting the Kar in the idle reset state as described in our use-case requirements, the 3-pin header connector required for sending PWM signals to control the Kar platform steering servo, a power switch for turning the Kar on and off, and a power indicator LED for informing the user of power status. The board is shaped and has the same mounting holes as the original Kar platform PCB, and has the added benefit of containing Mario's friendly visage on the top silkscreen.

D. Kar IMU and Haptic Feedback

We will use the ICM-42670 as the kart IMU, connected to the STM32 over SPI (mode 0) with DMA for burst transfers. The IMU FIFO will be enabled with a watermark interrupt on INT1 so the MCU services samples in batches rather than polling. Initial settings will target ±8 g on the accelerometer and ±1000 dps on the gyroscope, with an output data rate near 200 Hz and the on-chip digital low-pass filter set to remove high-frequency vibration from the drivetrain. On boot, the firmware will perform a brief stationary calibration to estimate gyro bias and accelerometer offset, then refine bias opportunistically during detected rest periods. Each sample will be timestamped on the STM32 using a monotonic timer to support precise downsampling and alignment with control loops and BLE packets. On-device processing was covered in section B.

For the haptic motor, we will drive an ERM vibration motor using the DRV2605L over I²C at 400 kHz. At startup the driver will run its built-in calibration to set compensation and overdrive parameters for the attached motor, then enter internal trigger mode. The application will expose a small set of patterns mapped to system events (for example: link acquired, control fault, over-tilt warning). For short cues we will use the DRV2605L waveform library; for variable-intensity feedback we will use real-time playback when appropriate. Each command will include an application-level sequence number so the firmware can ignore stale triggers and coalesce rapid event bursts into a single cue. The haptic subsystem will be managed by a simple queue with "latest-wins" behavior to keep latency predictable.

Electrical and timing details will be handled to ensure reliable operation. The I²C bus will use proper pull-ups sized for 400 kHz, and the motor supply will be budgeted for the ERM's peak current as specified by the driver. The firmware will gate haptic playback on link state and battery level, and it

will enforce maximum duty cycles to avoid overheating. If the DRV2605L INT pin is available, the MCU will use it to detect end-of-pattern and faults; otherwise, completion will be polled via the GO bit with a short watchdog timeout. This design provides deterministic actuation with compact commands and clear interaction rules, while keeping power and CPU cost low.

E. STM32 and ESP32 Bluetooth Connection

The critical latency stack is given in figure 2 on the left.

We will use a point-to-point (P2P) BLE connection that carries a custom GATT service for control and telemetry. The STM32WB55 will operate as the GATT server and advertise as a peripheral; the ESP32-S3 will act as the GATT client and initiate the connection. The service will expose three characteristics: Control RX for steering and throttle using Write Without Response, Telemetry TX using Notify, and an optional Haptics channel for acknowledgements and buzz commands. On link-up, the client will enable notifications, request an MTU of 247 bytes, negotiate Data Length Extension, and prefer the LE 2M PHY. Target parameters are a 15–30 ms connection interval, slave latency 0–4, and a 500–1000 ms supervision timeout. Control messages will be short (~20 bytes) and sent once per connection event; telemetry will use multiple notifications per event up to the Kontroller's packets-per-event limit. Under these settings, expected end-to-end control latency is roughly one to two connection intervals.

Reliability and observability will be handled at the application layer. Each packet will include a sequence number and a 32-bit timestamp to support loss detection, duplicate suppression, and simple round-trip timing. The client will keep a small transmit queue and always send the most recent control sample to avoid backlog. Devices will use bonded static addresses for fast reconnection and GATT caching, and the client will filter for the server's address to avoid unintended peers. The stack will monitor RSSI and fall back to the 1M PHY if link quality drops below about −85 dBm. A simple link state machine (Idle, Connecting, Configured, Active) will govern recovery: on rising round-trip time or a missed supervision event, the system will pause telemetry, attempt a parameter update, and reconnect if needed. This P2P GATT design aims to provide low, predictable control latency on battery power with clear behavior under degraded radio conditions.

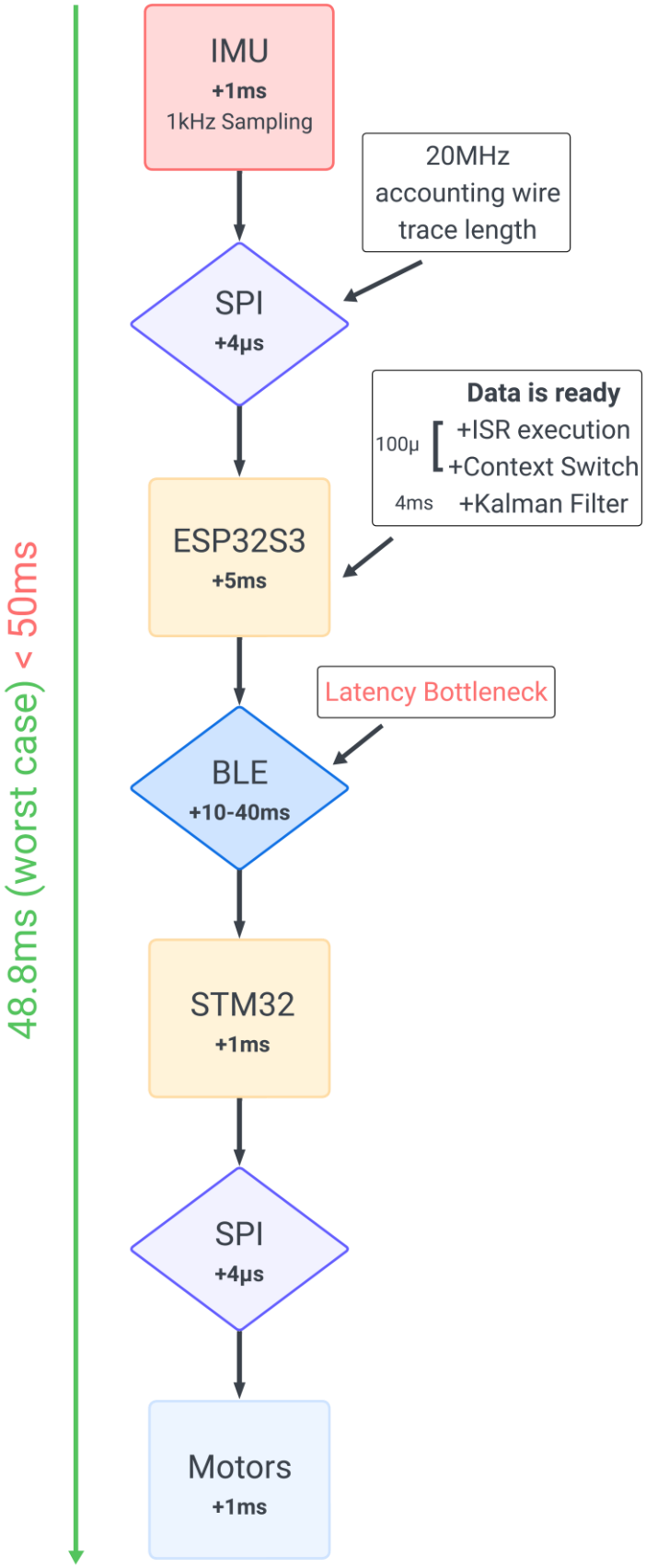

FIGURE 2: LATENCY STACK

VII. Test, Verification and Validation

To evaluate the effectiveness of our design and verify that it meets both the design and use-case requirements, we will employ a combination of user testing, quantitative measurements, and hardware validation. Each test is designed to directly assess how well the system performs relative to its specifications, while also ensuring that the final implementation delivers a safe, intuitive, and immersive experience for the user.

A. Tests for End-to-End Latency Measurement

To measure end-to-end latency, we will quantify the total system delay between the ESP32's command output and the STM32's corresponding response to verify the <50 ms use-case requirement. The ESP32 will send a GPIO set command, and the STM32 will output a corresponding GPIO signal upon processing the command. Using an oscilloscope, we will measure the time difference between the ESP32's GPIO pin going high and the STM32's output pin going high. This delay represents the total communication and processing latency. The design will be considered successful if the average latency remains below 50 ms, with individual processing stages, such as transmission and actuation, each taking less than 5 ms.

We will conduct 10 individual tests, where each test must result in an average end-to-end latency less than 50 ms.

B. Tests for Control Scheme Viability

To evaluate how intuitive and responsive the gesture-based control system is, we will assemble a simple driving course in a classroom environment using chairs or tables as obstacles. Ten users will attempt to navigate the course using the wearable Kontroller. Performance will be measured by recording the number of collisions or contacts with obstacles. Users who complete the course without hitting any obstacles achieve a 100% success rate, while each additional collision reduces the success rate by 10%. In addition to this quantitative metric, participants will complete a Likert scale survey (1–5) rating how intuitive the control scheme feels:

- 1 – Not intuitive at all
- 2 – Barely intuitive
- 3 – Somewhat intuitive but difficult to control
- 4 – Intuitive and controllable
- 5 – Very intuitive and easy to control

An average success rate greater than 90% and average score close to 5 will indicate that the design successfully meets the intuitive control use-case requirement.

C. Tests for Kontroller Weight, Wear, and Comfort

The weight, wear, and comfort of the Kontroller will be evaluated during the same user testing sessions. The same group of ten users will rate how light and comfortable the wearable Kontroller feels on two Likert scales from 1 to 5. The first Likert scale will grade the user's comfort level while wearing the device:

- 1 – Very uncomfortable
- 2 – Uncomfortable
- 3 – Not uncomfortable but not comfortable
- 4 – comfortable
- 5 – Very comfortable

On another Likert scale, the user will be able to provide their feedback on the weight of the Kontroller:

- 1 – Too heavy
- 2 – Somewhat heavy
- 3 – Could become heavy after 30 minutes
- 4 – Somewhat light
- 5 – Very light

To meet our design goal, the Kontroller must achieve an average user rating of at least 4/5 on both Likert scales, confirming that the device satisfies the wearability and comfort use-case requirements.

D. Tests for Haptic Feedback Clarity

Ten users will again operate the system under various driving conditions and rate how accurately the vibration feedback corresponds to their driving actions. For example, users should feel stronger haptic feedback when accelerating or colliding with an obstacle compared to when moving slowly or turning gently. Some conditions that will be tested during user driving include:

- Acceleration/deacceleration
- Slow driving/nominal driving
- Turns
- Crashes / heavy breaking

Success in this test will be determined by how well users perceive that the haptic signals reflect real-time driving dynamics through a Likert scale.

- 1 – Haptic feedback unclear or makes no sense
- 2 – Haptic feedback is somewhat clear
- 3 – Clear haptic feedback

Ratings of only 3 will validate that the feedback system enhances immersion and meets the use-case requirement for clear, informative haptic cues.

E. Tests for PCB Functionality

Prior to fabrication, the PCB design will pass an Electrical Rule Check (ERC) and Design Rule Check (DRC) in Altium to ensure that all electrical connections and manufacturing constraints are satisfied. Simulated performance tests within Altium will further validate that power distribution and signal integrity meet design expectations. Once fabricated, functionality will be confirmed during hardware bring-up, where the PCB's actual performance will be compared to the expected behavior observed during prototyping. Successful operation at this stage will demonstrate that the PCB design meets its technical specifications and supports reliable system integration.

VIII. Project Management

A. Schedule

A Gantt chart outlining our team's schedule more precisely can be found in Figure 4.

The goal is to finish the MVP by the end of October, leaving enough slack time in November to pursue some of our reach goals, which include integrating a camera onto the Kar and building a race track for demo day.

Important milestones throughout the MVP period include

finishing the schematic and layout of the custom PCB, setting up Bluetooth Low Energy communication between the glove Kontroller MCU and the Kar MCU, interpreting/translating IMU data from the Kontroller into Kar movement, and interpreting/translating IMU data from the Kar into haptic feedback. Integration will be completed as subsystems are completed.

B. Team Member Responsibilities

Caitlyn Fong: Primarily responsible for IMU raw-value interpretation and filter implementation. Responsibilities also include integrating the haptic motor controller. Co-responsible for working on the Kar motor control and PID system with Enrique.

Enrique Gomez: Primarily responsible for setting up Bluetooth communication between the STM32 and ESP32 and setting up FreeRTOS on both microcontrollers. Responsibilities also include writing coverage tests for safety analysis. Co-responsible for working on the Kar motor control and PID system with Caitlyn.

Nicolas Keck: Primarily responsible for the hardware stack from designing the custom PCB schematic, layout and assembling the PCB. Responsibilities also include prototyping for hardware verification.

C. Bill of Materials and Budget

Our budget is $600. We have currently spent around $300 of our budget We are currently at about $300 spend of the budget, leaving us Please refer to Appendix A Table 7 regarding the bill of materials for our MVP.

D. TechSpark Use Plan

The individual subsystems will be built and tested individually; however final assembly will be done at TechSpark. Remaining surface-mounted components that need to be assembled and soldered on the PCB will be done in the PCB labs in TechSpark using the ProtoFlow S Reflow oven. Through-hole components will be manually soldered by hand using a soldering iron.

E. Risk Mitigation Plans

One of the primary risks in the Mario Kar system is ensuring the safety of the Kar during operation. Since the Kar moves autonomously in response to user gestures, it is critical to prevent unintended motion or collisions. A key failure mode could occur if the wearable Kontroller loses connection due to battery depletion or if the BLE link goes out of range. In such cases, the Kar may continue its last-received command, potentially causing unsafe movement. To mitigate this, the Kar's MCU will implement a software timeout mechanism. If no valid control signal is received within a specified period, the MCU will automatically transition the Kar into a safe idle state, stopping all motor activity.

Another potential risk involves unreliable or noisy IMU data, particularly from the IMU mounted on the fingers. Because finger motion is often subtle and subject to small vibrations or external disturbances, the IMU readings may be inconsistent, leading to inaccurate movement detection. To address this, the system design includes support for flex sensors as an alternative or complementary sensing method. The glove Kontroller's PCB includes additional 6-pin connectors specifically designed for integrating flex sensors, allowing the system to measure finger bending more robustly.

Power consumption on the wearable glove presents another significant risk. Continuous IMU data collection and wireless communication can quickly drain the battery, potentially shortening operational time. To mitigate this, the system can offload IMU data filtering and computation to the STM32 MCU on the Kar side whenever possible.

Finally, high processing latency poses a risk to responsiveness and overall user experience. Latency can arise from multiple sources, including sensor sampling delays, filtering operations, wireless transmission, and actuation response. To mitigate this, the design will focus on minimizing latency in controllable stages of the pipeline. This includes optimizing filtering algorithms for efficiency, using interrupt-driven communication instead of polling where feasible, and ensuring minimal overhead in data transmission between the wearable and the Kar.

IX. Related Work

Several existing projects and products share conceptual similarities with the Mario Kar system, particularly in the areas of gesture-controlled robotics, wearable motion interfaces, and wireless haptic feedback systems.

One relevant example is Nintendo's Mario Kart Live: Home Circuit, which also combines physical cars with digital control. [3] However, that system relies on a handheld gaming controller and camera-based tracking, whereas our project emphasizes intuitive, touchless gesture control through wearable sensors and real-time motion mapping.

In the research and hobbyist domain, several gesture-controlled robotic cars exist that use accelerometer or IMU-based control via Arduino or Raspberry Pi platforms. These typically interpret tilt or rotation from a handheld IMU to steer and accelerate a robot. However, these implementations generally lack multi-sensor fusion, haptic feedback, and wearable integration.

X. Summary

Mario Kar combines a wearable gesture-based Kontroller and a wirelessly connected Kar to create an immersive, touchless driving experience. The Kontroller interprets the hand and finger movements through IMUs, while the Kar translates these commands into motion. The IMU data on the Kar is processed into haptic feedback that is transmitted to the glove to enable users to feel collisions and turns through vibration.

The design emphasizes user experience, safety, and reliability for users through the lightweight ergonomic Kontroller that ensures comfort while the intuitive gesture-to-

motion mapping minimizes learning time.

Key implementation challenges include maintaining low-latency and reliable BLE communication, as wireless interference and synchronization between MCUs could affect responsiveness. Another challenge is ensuring IMU precision while minimizing power consumption on the glove, which can be mitigated by shifting some data processing to the Kar's STM32 MCU.

In summary, Mario Kar delivers an innovative blend of wearable motion sensing, wireless control, and haptic interaction, offering gesturer and spectator drivers a safe, responsive, and immersive driving experience.

Glossary of Acronyms

BDC – Brushless DC (Motor)
BLE – Bluetooth Low Energy
COTS – Commercial Off the Shelf
DC – Direct Current
DMA – Direct Memory Access
FIR – Finite Impulse Response
FIFO – First-in First-out
IMU – International Measurement Unit
I2C – Inter-Integrated Circuit
MCU – Microcontroller Unit
MQTT – Message Queuing Telemetry Transport
PCB – Printed Circuit Board
OBD – On-Board Diagnostics
RC – Remote Control
RPi – Raspberry Pi
SPI – Serial Peripheral Interface

References

[1] https://ai.stanford.edu/~conti/papers/SPIE00-CONTI.pdf
[2] https://www.roboticsproceedings.org/rss07/p27.pdf
[3] https://www.nintendo.com/us/store/products/mario-kart-live-home-circuit-mario-set/?srsltid=AfmBOorC9l-gbHMPrttC-9k83zFtBYVZ_gh-myAGa2lyfU9aci5JDj1K
[4] IEEE, *IEEE Author Center: Author tools*, Accessed on Jan 17, 2022, [Online]. Available: https://newauthors.ieeeauthorcenter.ieee.org/author-tools/
[5] https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf
[6] https://invensense.tdk.com/wp-content/uploads/2021/07/ds-000451_icm-42670-p-datasheet.pdf
[7] https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/7174/828_bst-bmi330-ds000.pdf
[8] https://www.ti.com/lit/ds/symlink/drv8242-q1.pdf?ts=1712092039219&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252Fes-mx%252FDRV8242-Q1
[9] https://www.analog.com/media/en/technical-documentation/data-sheets/max22212.pdf
[10] https://www.ti.com/lit/ds/symlink/drv8871-q1.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1760148132621&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Fdrv8871-q1
[11] https://www.ti.com/lit/ds/symlink/sn74lv1t34-q1.pdf
[12] https://www.ti.com/lit/ds/symlink/txb0104.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1760106893363&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Ftxb0104

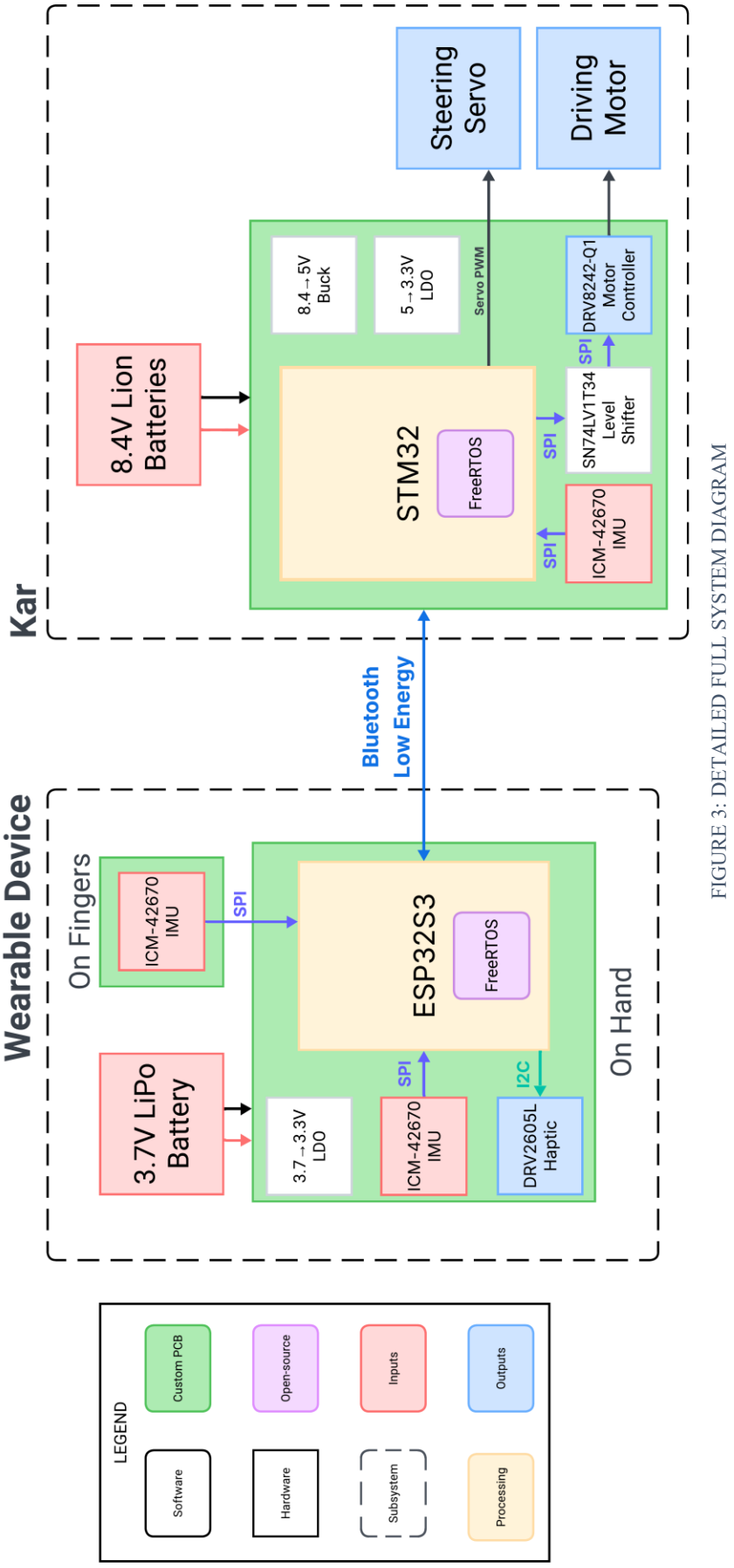

FIGURE 3: DETAILED FULL SYSTEM DIAGRAM

WBS NUMBER	TASK TITLE	TASK OWNER	PCT OF TASK COMPLETE	September					October				November				December
				08/31-09/06	09/07-09/13	09/14-09/20	09/21-09/27	09/28-10/04	10/05-10/11	10/12-10/18	10/19-10/25	10/26-11/01	11/02-11/08	11/09-11/15	11/16-11/22	11/23-11/29	11/30-12/06
1	**Logistics and Setup**																
	Website Setup	Everyone	100%														
	Slack	Everyone	100%														
	Proposal Presentation	Everyone	100%														
	Design Presentation	Everyone	100%														
	Interim Demo	Everyone	0%														
	Final Presentation	Everyone	0%														
	Poster PDF	Everyone	0%														
	Final Video	Everyone	0%														
2	**Project Planning**																
	Abstract	Everyone	100%														
	Order Materials	Everyone	80%														
	Finalize Design	Everyone	100%														
3	**Project Definition**																
	Custom PCB Schematic	Nick	100%														
	Custom PCB Layout	Nick	100%														
	Breadboarding Components	Nick	50%														
	FreeRTOS Setup Research	Caitlyn & Enrique	100%														
	FreeRTOS Setup for ESP32/STM32	Caitlyn & Enrique	100%														
	IMU raw-value interpretation	Caitlyn	70%														
	IMU FIR experimentation	Caitlyn	50%														
	IMU Kalman filter experimentation	Caitlyn	20%														
	Bluetooth research and prototyping	Enrique	100%														
	Bluetooth communication implementation	Enrique	50%														
	Haptic controller	Caitlyn	0%														
	PCB Bringup	Nick	0%														
	PCB Design Fixes	Nick	0%														
	PID system for motor control	Caitlyn & Enrique	0%														
	Final PCB Bringup	Nick	0%														
	Final Software Fixes	Enrique	0%														
	Construct Track (Slack)	Caitlyn & Nick	0%														
	Integrated Camera (Slack)	Enrique	0%														
4	**Project Testing / Monitoring**																
	Prototype hardware verification	Nick	0%														
	Coverage Tests	Enrique	0%														
	Safety Tests	Everyone	0%														
	User Testing	Everyone	0%														

FIGURE 4: GANTT CHART

TABLE VII: BILL OF MATERIALS

Description	Model #	Manufacturer	Quantity	Cost @	Total
XIAO ESP32S3	113991114	Seeed Studio	2	7.99	15.98
NUCLEO-WB55RG	STM32WB55RG	STMicroelectronics	1	35.51	35.51
Donkey Car	PiRacer Pro AI Kit Acce	Waveshare	1	179.99	179.99
ST-LINK/V2	ST-LINK/V2	STMicroelectronics	1	22.29	22.29
3.7V 600mAh Rechargeable Lithium Polymer Battery	YDL2018032959	YDL	1	24.98	24.98
18650 Rechargeable Battery	9900	Ltrrysoa Zhuhai Xutai Commodity Co., Ltd	4	2.98	11.92
				Grand Total	**$290.67**

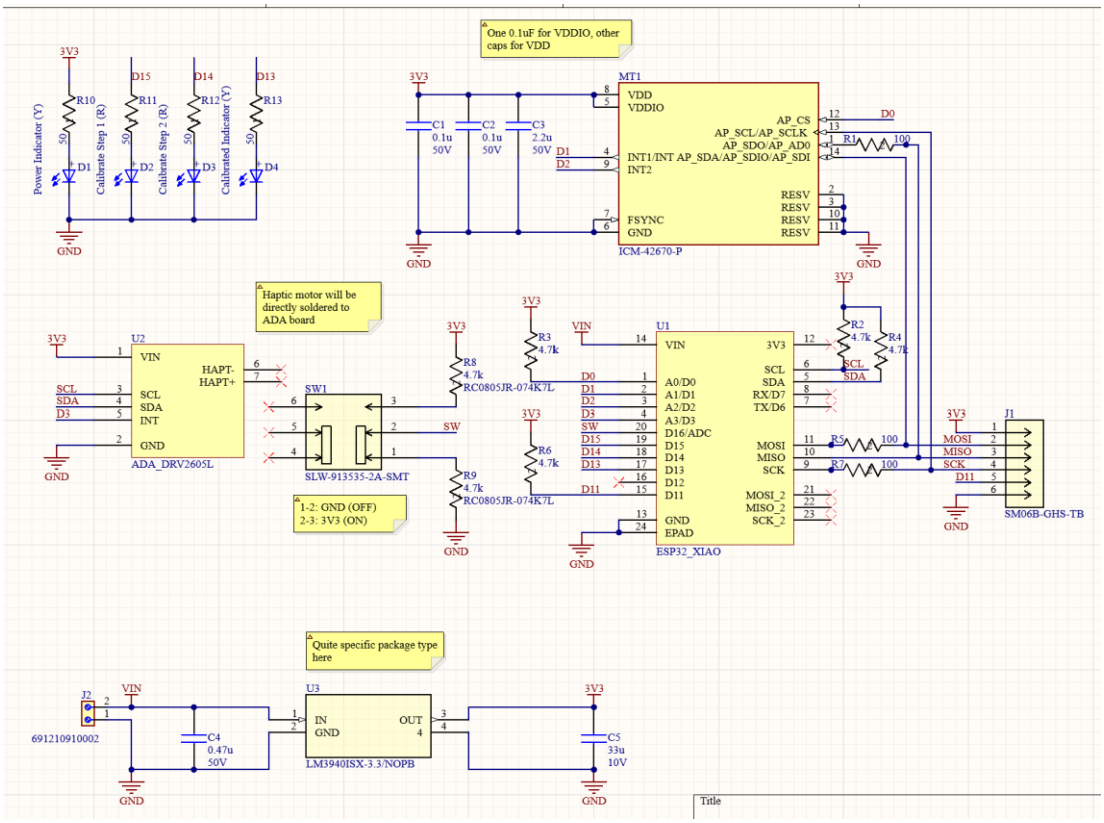

FIGURE 5: KONTROLLER SCHEMATIC

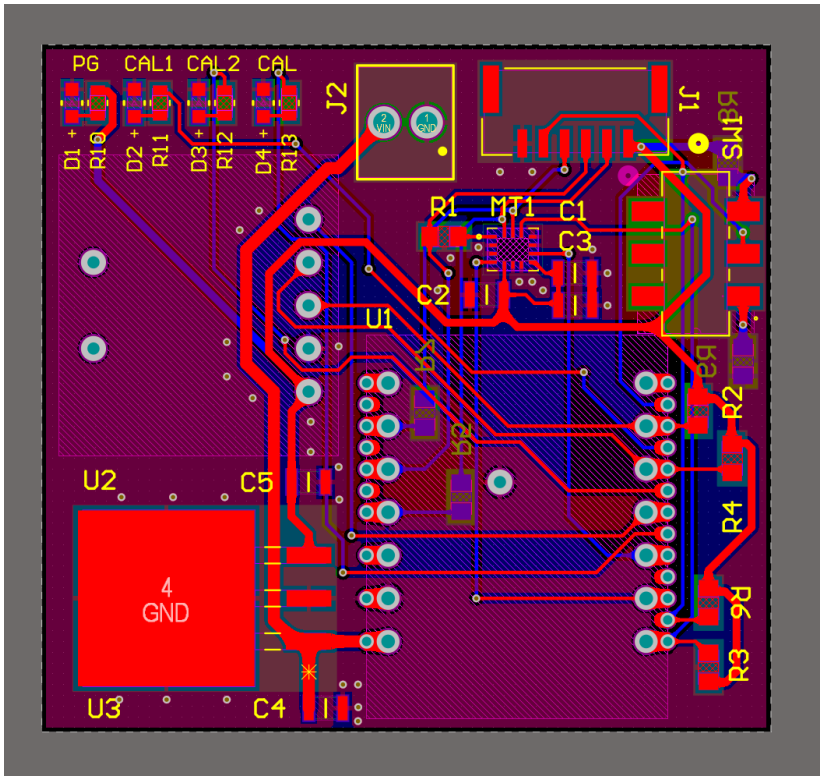

FIGURE 6: KONTROLLER LAYOUT

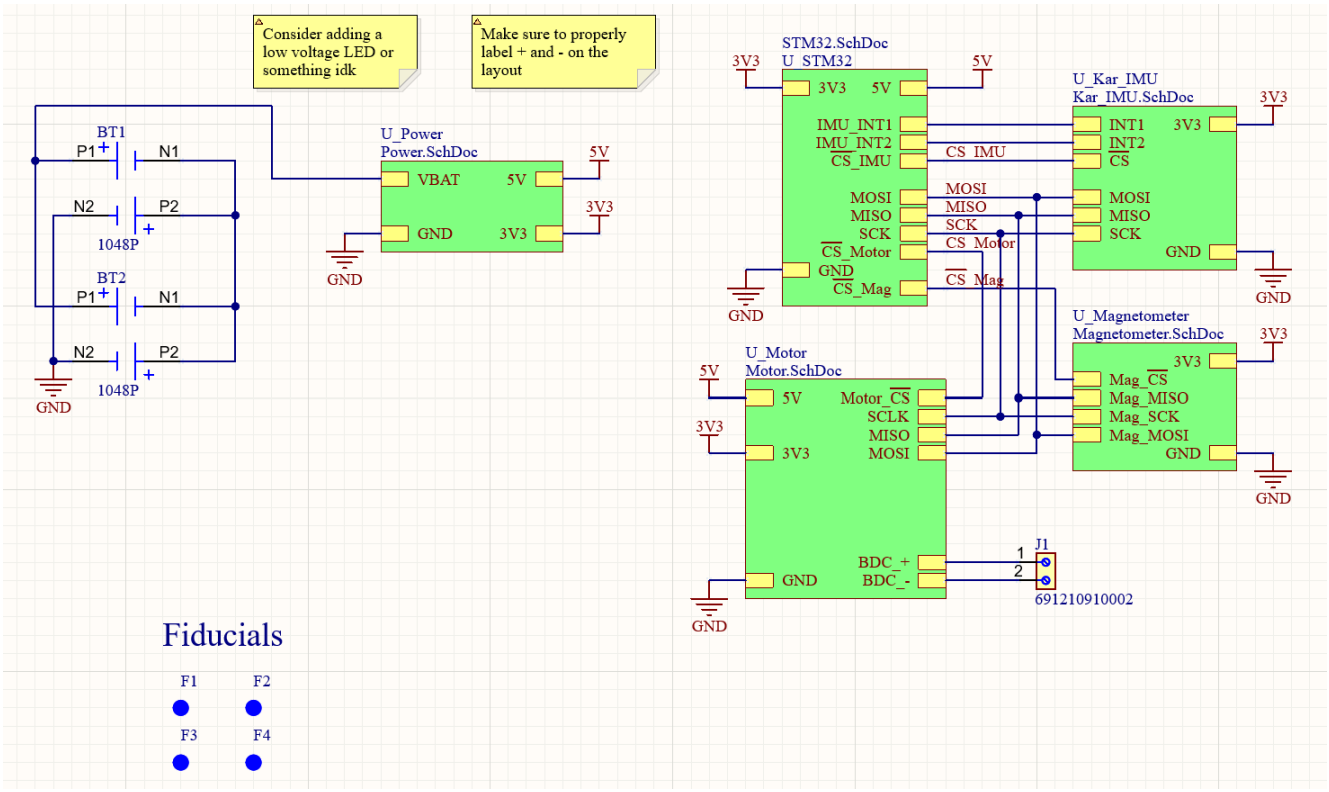

FIGURE 7: KAR SCHEMATIC

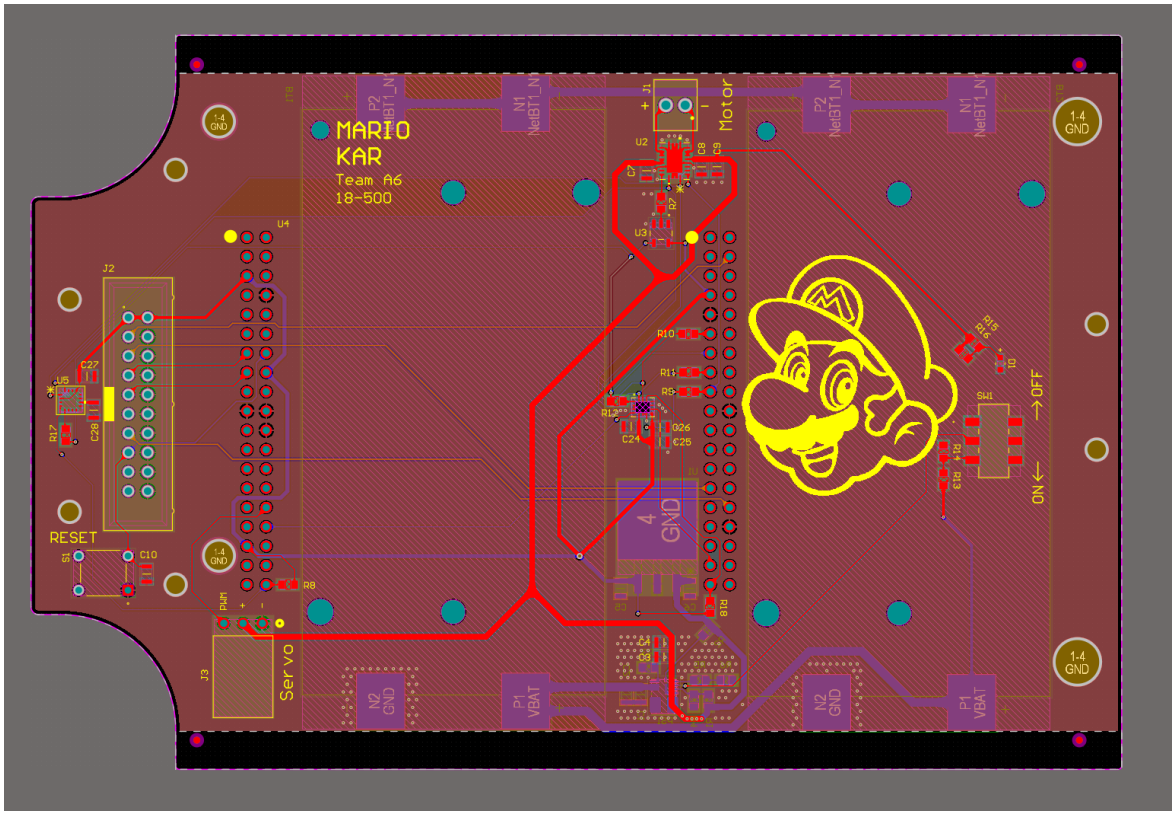

FIGURE 8: KAR LAYOUT