Mobile Steering Wheel

Xiao Jin, Yuxuan Zhu, Qiaoan Shen

Department of Electrical and Computer Engineering,
Carnegie Mellon University

***Abstract*—Our design is a wireless video game controller system capable of detecting gyroscopic input, dedicated to function with BeamNG.Drive. For driving simulation games such as BeamNG.Drive, having a steering wheel controller helps tremendously with the level of immersion users can get. However, such steering wheel controllers tend to cost more than 10 times the cost of a generic gaming controller. Our wireless steering wheel controller is designed to expand on the abilities of generic controllers, by adding 360 degree angle sensing to simulate turning a steering wheel, a 5 inch LCD display for vehicle information, and a 3D printed enclosure that resembles real steering wheels in race cars. The whole package weighs around 400 grams, around the average of what smart phones weigh nowadays.**

***Index Terms*—Design, controller, driving, gaming, racing simulator, steering wheel, wireless**

I. Introduction

Simulated Racing, or simply Sim Racing, is a form of entertainment involving the simulation of driving or racing cars on computer games. Over the years, enthusiasts have evolved from using keyboard control, to dedicated game controllers, and now to full custom-made cockpits with 1:1 replication of real cockpits. Such sim racing cockpits often cost more than a few thousand USD, making it an extremely high bar of entry, and for this reason sim racing has been a very niche hobby. However, during the COVID-19 lockdown, an influx of newcomers surprised the sim racing community. As active members of this community, we were thrilled to see so many people interested in joining sim racing. It was unfortunate, however, for enthusiasts to learn that many people were turned away by the high cost of controller devices such as steering wheels and pedal sets. There simply isn't a controller on the market that combines the affordability of generic game controllers, and the immersive experience from an enthusiast-grade custom sim racing cockpit.

There is a void in the market for a cheap sim racing controller with immersive gaming experience, and our project is going to fill this void. As users of sim racing controllers ourselves, we brainstormed some use cases for such a product. The product will be used to control one of the most popular sim racing games, called BeamNG.Drive. It will resemble a steering wheel in a Formula One race car to introduce some level of immersion. It must be portable, in the sense that users can just stuff it in their backpack and take it on a trip. This means that such a product will not have force-feedback functionality, one of the reasons why some steering wheel controllers are very expensive. For people who are just trying out sim racing, however, force-feedback is not a necessity. This means that our product can be a free-floating, steering wheel shaped controller. Accurate steering input, on the other hand, is vital to sim racing. Traditionally when casually playing racing games, people simply use the arrow keys on keyboards to control acceleration, braking and turning. In order to achieve a high level of immersion in sim racing, controllers need to have analog acceleration, braking and steering inputs. Our product will have analog steering angle sensing capabilities, to simulate the effect of turning a steering wheel, even though there is no fixed axle the wheel should be attached to (steering column). Our controller will also support wireless connection to PCs, making it an ultimate package for maximizing portability.

Although portability is one of the biggest features of our controller, it is still important to retain some level of immersion. After all, it was the "simulated" aspect of sim racing that attracted people who used to play arcade style racing games. Compared with the XBOX Wireless Controller, which has a joystick for analog steering input, our product adds a more intuitive way of sensing the degree of tilt, making users' hand movement similar to turning the steering wheel in a real car. Our controller will also have a small form factor LCD display facing the user, so the user can choose to display some vital information about the car they are driving, just like how a real car's instrument cluster works. Compared with a more hardcore steering wheel controller such as the Thrustmaster T300RS, a very popular steering wheel controller setup amongst enthusiasts, our controller eliminates the accelerator and brake pedals, while still offering analog triggers for the same function. Our controller also does not have a heavy base that needs to be clamped onto a desk, unlike the Thrustmaster T300RS, and that means the complicated force-feedback function is also gone.

Overall, our controller eliminates some high-end features of existing steering wheel controllers, while retaining the core functionalities without breaking the bank, in a portable package that users can carry around in a backpack. The goal is to significantly lower the level of commitment, both in terms of money and space, required to get started with sim racing, and to attract more people into the wonderful world of sim racing.

II. Use-Case Requirements

1) The controller should be able to detect the controller tilt angle in 360 degrees when users rotate the controller. 2) The controller should be able to last for 8 hours in order to create a smooth gameplay without having users to charge frequently. 3) The controller should have 14 buttons and 2 analog inputs to function similarly like XBOX and PS5 controllers on the market. The buttons on the controller can be programmed to simulate various functionalities on a real steering wheel. 4) The delay between the time when the user presses a button or makes a rotation on the controller and the time when the car moves should be less than 20ms. Similar controllers also have

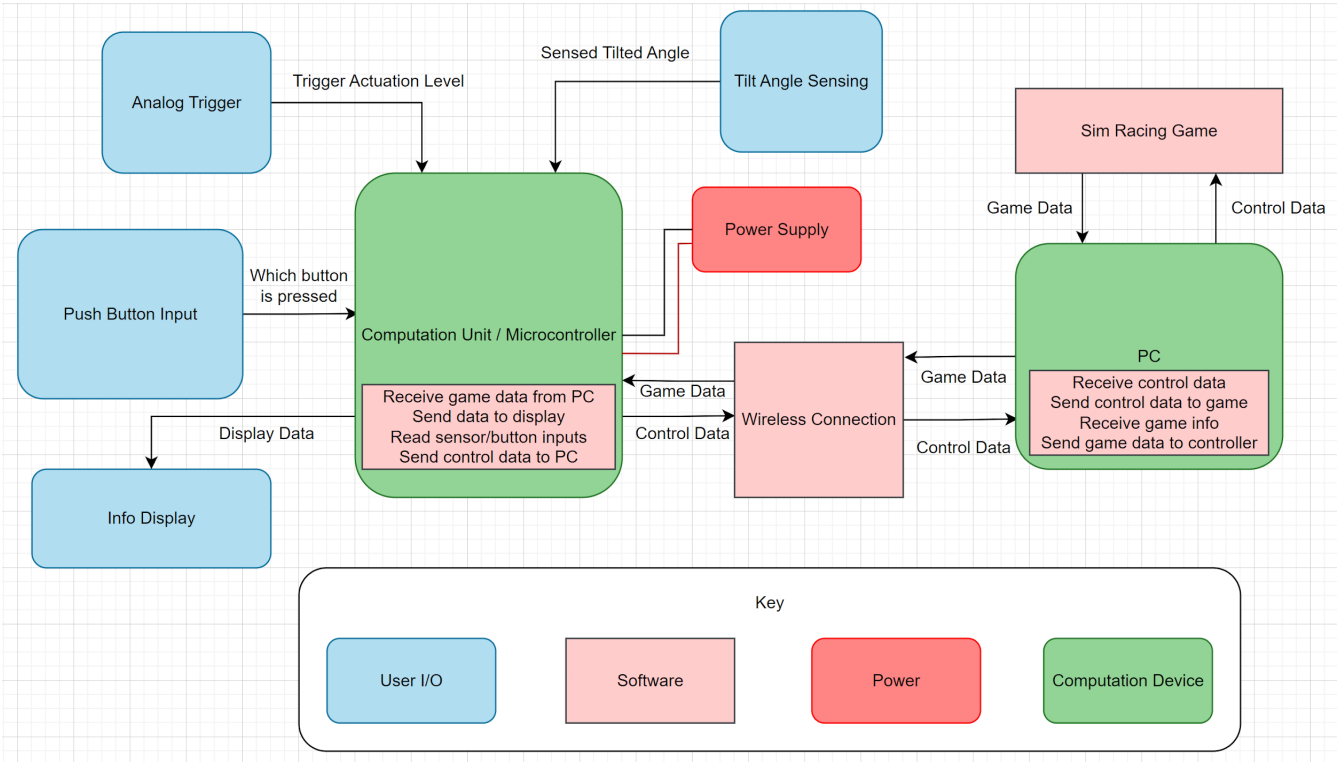

Fig. 1. System Architecture Block Diagram

input delay under this metric because otherwise the game will lag too much and won't be playable. 5) The weight of the controller should be less than 400g. A controller that weighs more than 400g will cause strain on the user's arms and hands as they hold it for a long time in the air. Many popular game controllers also weigh less than 400g. 6) An LCD display is required to present car information such as speed and acceleration on the controller. The delay between the time the game outputs the data and the time the controller receives the data should be less than 20ms. 7) When the controller is parallel to the ground, the car should move in a straight line. When the controller rotates to a certain angle, the steering wheel inside the game should also rotate to a specific angle. The relationship between the tilt angle detected and the angle of the steering in the game should follow a predetermined equation. 8) When pressing the accelerator analog button, the car is expected to accelerate at a speed predetermined by the car type. When pressing the brake analog trigger, the car in the game is expected to apply brake proportional to the amount of trigger being depressed. The exact time is also dependent on the car type used in the game.

III. Architecture and/or Principle of Operation

Our controller will consist of several sensors for getting user input, an LCD screen for displaying key data from the game to the user, a computation device to translate sensor data to game controller output that is readable by a computer program, some wireless functionality for the communication between the PC and the controller, and a battery system for cordless power. (See Fig. 1)

For the hardware side, our controller will be getting user input from a range of sensors, including push buttons, analog triggers and most importantly for us, the tilt sensor. When the user pushes a button on the steering wheel, the computation unit will recognize this event as a button push. When the user depresses the analog triggers to a certain percent of travel, the computation unit will record that amount of actuation. When the user tilts the steering wheel to a certain degree to simulate steering in a real car, the tilt sensor will recognize the degree of tilt, and pass that data to the computation unit. The computation unit will also be responsible for deciding the content for display on the LCD screen. A battery system will be responsible for providing power to the sensors, the screen and the computation unit.

For the software side, a program will run on the controller to consolidate all of the data from buttons and sensors. The program will establish a remote connection with the PC that runs the BeamNG.Drive game in order to send all the data from the controller to the PC. Also, the program on the controller should be able to interpret the car information received from the PC so that it can control the LCD to display car information such as speed and acceleration. The program on the PC side will also establish a remote connection to the program running on the controller so that it can receive the button and sensor information. After receiving the raw information, the program on the PC is responsible for translating the raw button information into an input format such that the game can understand. After the input is successfully translated, there will be responses in the game. In addition, the program on the PC will also aggregate game data from the game in order to send them remotely back to the controller for LCD display.

IV. Design Requirements

Based on the user-case requirements, we will determine the design requirements as stated below.

The shell design should follow the requirement that it should provide 14 buttons, 2 analog inputs and a screen to provide functions similar to XBOX and PS5 controllers. We need to design holes with radius no more than 5mm on our shell to place those buttons on the shell. To satisfy the 400g weight limit, the 3D printing shell should have no more than 10% infill to minimize weight.

The sensing system should allow users to perform 360 rotation on our steering wheel, and also be sensitive enough to detect any change more than 0.1 degree. The gyroscope should also be capable of processing a rapid tilting over 360 degrees within half a second. The gyroscope system also needs to consider the effect of the shaking of the user's hands, it should filter out changes within 2.5 degrees in less than 50ms so that the game will not be affected. The Accelerator and brake pedal should also be included in the sensing system that should read the user input from 0% to 100% pushed, with accuracy at 0.1%.

The software system has two requirements about data transmission rate and latency. The latency should be no more than 20ms since it's a time that's negligible for users to feel the latency, hence we need to find our method to transmit data within this time range. We need a fast and well-supported communication standard, and Bluetooth is our choice based on the requirements above. The software system also should not transmit data more than 125kB/s so the communication will not be blocked by too large packages. Our software system should design a package format for transmitting data.

The hardware system should be able to connect all the components and make sure the microcontroller reads the correct data from sensors and be able to send packages to the computer. It should also use the RPi to control a LCD screen that has no more than 20ms latency so the effect of the latency is negligible to user's experience. To support an 8-hour battery life, we investigated the power consumption of RPi. RPi 4B, the RPi type we choose, usually drains the battery with a current around 600mA. The power consumption can be calculated by the formula $Capacity = I * Battery\ Life$ and here we calculate the data as $600mA * 8h = 4800mAH$. To satisfy the 8-hour battery life of our product, we determine that we need to use a battery no smaller than 5000mAH.

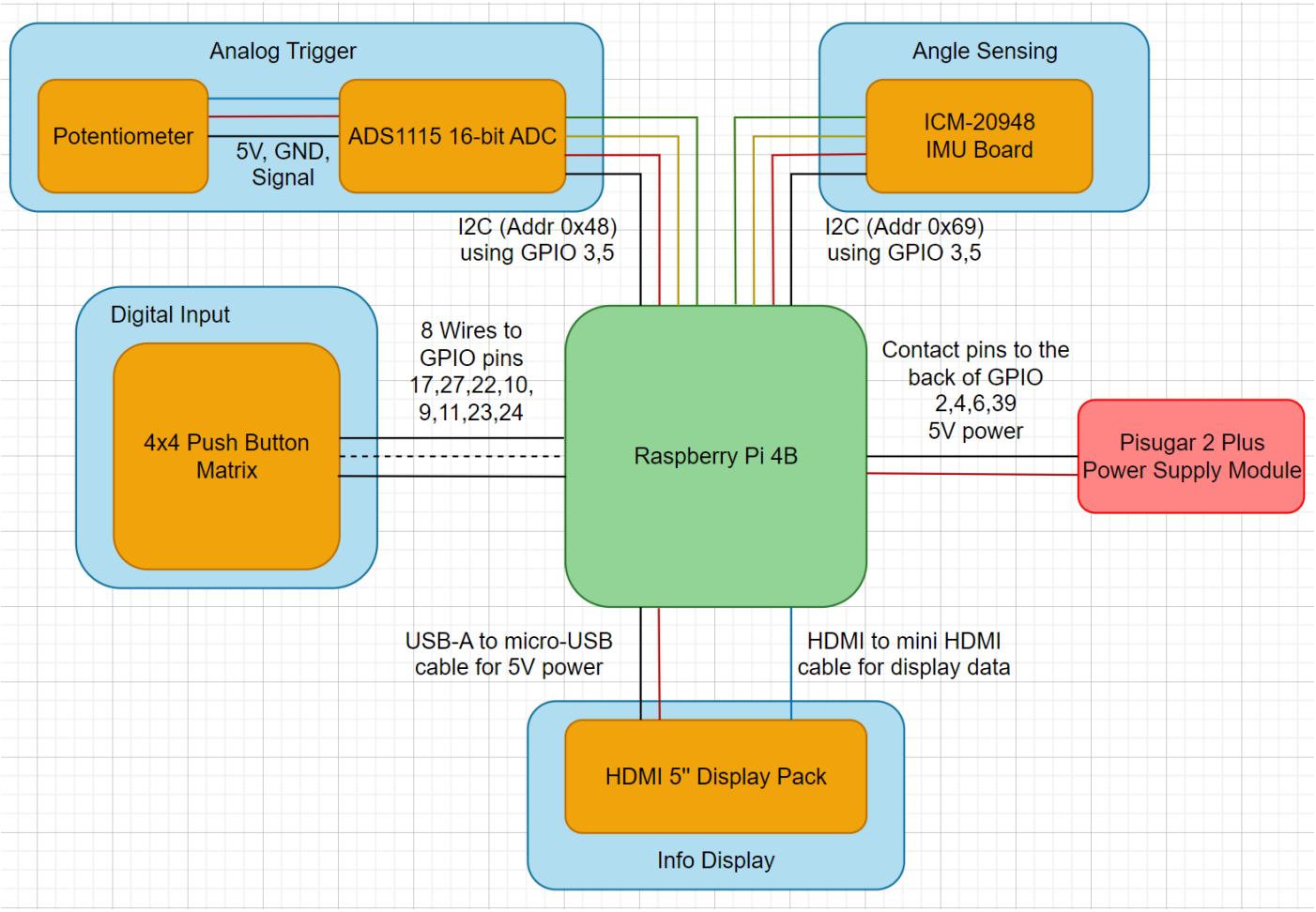

Fig. 2. Hardware Implementation Block Diagram

Subsystem	Use Case Target	Design requirements
Shell Design	14 buttons and 2 pedals; 400g weight limit	Holes radius less than 5mm and 10% infill for printed parts
Sensing system	360 degree rotation, acceleration and brake pedals	Gyroscope accuracy at 0.1 degree change, noise filter for less than 2.5 degrees change, pedals accuracy at 0.1%
Software System	20ms latency; Data rate limit	Bluetooth; < 125kB/s
Hardware System	8 hour battery life	> 5000mAH battery

V. System Implementation

A. Hardware Systems

See Fig. 2 for a detailed block diagram. We will be using a Raspberry Pi 4B as the computing device on the controller. All the sensors will communicate with the RPi, and the RPi will handle display output and Bluetooth communication as well. The IMU-20948 board will be responsible for sensing 3-axis gyroscope data, and communicating with RPi through I2C protocol on address 0x69 as specified in the datasheet. The analog trigger will consist of a 10K Ohm potentiometer and a ADS1115 16-bit ADC. The ADC is extremely important here, since RPi does not have a dedicated analog input, and is notorious for poor handling of analog signals. the ADC will communicate with RPi using I2C on address 0x48. Python library smbus2 will be used to read data from I2C registers. There will be 16 momentary push buttons wired in a 4x4 matrix. The 4 rows will each connect to GPIO pins 17, 27, 22 and 10, while the 4 columns will connect to pins 9, 11, 23 and 24. The Python package RPi.GPIO will be used to read button matrix pin states, and update the buttons being pressed. An HDMI 5" LCD display will be connected to the RPi through HDMI port and USB cable for 5V power. For power supply for the whole system, we chose Pisugar 2 Plus Power Management board. It connects to the back of the RPi via metal contacts, and does not occupy any GPIO pins. With that said, it does offer I2C communication capabilities, if we decide to implement that. With I2C, Pisugar will be able to communicate with RPi regarding low power sleep state, automatic power on, and so on.

B. Software Systems

See Fig. 3 for a detailed block diagram. There will be three main programs developed for this game controller system. The first two programs will run on RPi and the second program will run on PC. All programs will run in the Python programming language and the code will be stored on Github for version control.

The program to tune sensors will transform data from analog output of pedals and gyroscopes to a more user-friendly output to send to the PC. Its main focus will be filtering out the noise from users' hand shaking and changing sensitivity to a more user friendly way. For the first target, a Gaussian filter will be applied on the input gyroscope readings based on the changes less than 2.5 degrees within 50ms.

Except for the filtering part, the tilt sensor will also treat user inputs at two different speeds. If the user turns the steering wheel rapidly over 90 degrees, the system should identify the user as making a sharp turn and will provide a faster response that jumps the output directly to the user's target angle. For example, if the user suddenly rotates the steering wheel to a place close to +180 degrees, it's very likely the user wants to turn right as much as possible, at this time the output will be set to +180 degrees immediately. On the other hand, if the user is slowly changing the degree in a long period of time, the steering wheel should consider the case that these changes may not be intended by the user and are due to the user's body movement. In this case, the transformed gyroscope output will be determined by the average angle in the past 50 ms to slow down the change.

One last part of the sensing work is to transform the output from analog pedals. When a user pushes the pedal over 95%, will tune the data output to 100% to smooth the user experience since the user is very likely just pushing the pedal with large enough force.

To establish Bluetooth communication between RPi and PC, Pybluez package will be used. The package provides useful functions for devices to find service and devices. Just like the traditional server and client communication program in socket, the package allows the program to act as a server and advertise itself to other devices. Other devices can act as clients such that they can discover the advertised devices. The initial plan is to set PC as the server and RPi as the client. Once the connection is established in two programs, the two will send and receive data in a while loop. The while loop in each program contains a series of sequential logistics that handles data reading from game or sensors and interpreting data.

On the RPi side, the logistics in the while loop will handle 3 core functionalities. The first will focus on reading data from RPi buttons and sensors. In this part, the program will use the Pigpio package which provides functionalities to read data from GPIO pins and communicate with devices through UART and I2C protocols. This part will mostly focus on reading all the required data without doing processing and interpretation. The second part will focus on transmission of data using the Pybluez package as mentioned above. It will first call a function that writes the button and sensor data remotely to the PC. Then it will call a function that reads the data from the PC which is the car data in the game. The third

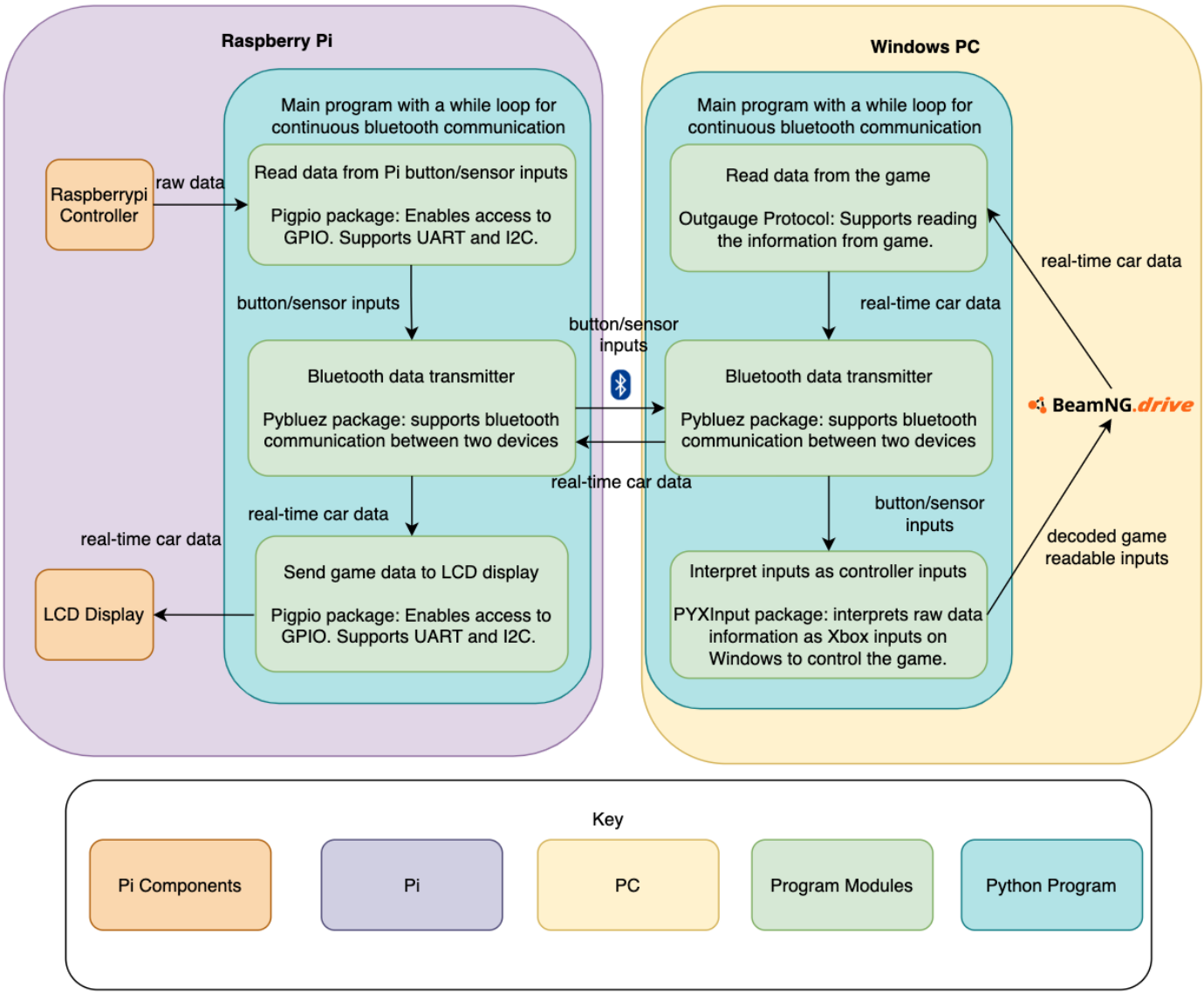

Fig. 3. Software Implementation Block Diagram

part will be reading the car data and output necessary information to the LCD display. This process also involves the Pigpio package which supports UART and I2C communication. Before outputting the information, the program will also do some processing and formatting of the data so that the data on the LCD display is readable.

On the PC side, the logistics in the while loop also handles 3 core functionalities. The first will focus on reading data from the game. This step involves setting the game to outgauge mode and binding the socket to port 4444 such that the program can receive the game data. The second part will focus on transmitting game data to the RPi side using Pybluez package. After sending data to the RPi, this part will read data from RPi to obtain the raw button and sensor data. The third will focus on decoding the data from RPi. It will implement a logic such that all of the inputs can be matched to Xbox input in order for the game to understand. In order to achieve this, PYXInput package which can simulate an Xbox controller connected to a Windows PC. The decoded inputs will be eventually called by functions in the package to control the game.

VI. Test, Verification and Validation

The project can be measured by two different perspectives, including hardware and software perspectives.

Testing on hardware systems aims to make sure the product has all its components connected properly. We need everything in the hardware field like sensors to work as expected to generate correct input and output. In addition to this, hardware systems should also pay attention to limits like cost and weight that are defined by our use case requirements for a cheap and portable mobile device, though this usually means some trade-offs on performance.

Testing on software systems aims to make sure the software meets the requirements from the user experience perspective. The software system should design an algorithm to tune the data from pedals and gyroscope to allow a smooth user

experience and a working communication protocol to transmit data from the controller to the computer inside the given time limit.

A. Tests for Hardware I/O

We will perform tests on I/O accuracy when hardware components are assembled. We will manipulate the steering wheel to a certain angle to simulate the user input to our device, and then check the readings from the gyroscope on the microcontroller to see if the result matches the correct user input. We will also send a sample package using the bluetooth part to the computer to see if that works as expected.

B. Tests for Hardware Connectivity

After receiving the manufactured PCB board, we will move all components from the breadboard and check if all components are working properly on our PCB board. We will check if all sensors are connected by pushing buttons and tilt the steering wheel to see if we receive output from sensors as expected.

C. Tests for Software Latency

We will use a slow motion camera to check the input latency by counting the frames for the game to react after the user pushes a button.We can calculate the latency in milliseconds by the formula:

$$Latency = \#\ of\ frames * 1000ms/fps$$

To pass the test, we need the latency to be less than 20ms so that the latency will be negligible to the user.

D. Tests for Gyroscope Tuning

The gyroscope should be tuned to provide a better user experience so that it should not be too sensitive to the shaking of the user's hand. We will shake the steering wheel by changing the direction less than 2.5 degrees multiple times in both directions within 50ms and the gyroscope should filter out these inputs as noise. To pass the test, we should not see any change on tuned gyroscope output after we perform the shake above.

E. Tests for Data Rate

We will measure the data rate transmitted to make sure it's under the hardware limit to ensure the performance. We will write a program to calculate the total size of data sent by the controller and received by the computer per second, and the test requires the transmitted data rate to be less than 125 kB/s.

F. Tests for Battery Life

We will fully charge the battery of the steering wheel. Then we connect the controller with the game and measure the time it runs out of the power as the controller continuously communicates with the game. The battery should last more than 8 hours to pass the test.

G. Tests for Weight

We will measure the total weight of our steering wheel by an electronic scale. To pass the test the controller should be less than 400g so the user will not feel tired holding it in their hands.

VII. Project Management

A. Schedule

The detailed schedule is shown on Fig. 4.

B. Team Member Responsibilities

Xiao Jin will be responsible for building the hardware system. He will connect components with the RPi microcontroller with a PCB board designed by himself.

Qiaoan Shen will be responsible for building the shell of the steering wheel and deal with sensors like gyroscope and analog pedals. He will build a 3D model for the product and create algorithms to tune sensors.

Yuxuan Zhu will be responsible for the communication with the PC game. He will focus on utilizing the API of the game and create fast and reliable connection between the controller and the game.

C. Bill of Materials and Budget

See Table I at the end of this report.

D. Risk Mitigation Plans

The biggest risk factor in our project is the tuning of the gyroscope output. We previously have had no experience with a gyroscope, and having to use an algorithm to consolidate potentially all 9 channels of data could be very challenging. In the event that a complicated algorithm does not work as expected, or if we run out of time, we will consider making compromises on the performance of the controller, for example, downgrade from sensing 3-axis movement to only sensing 1-axis movement.

VIII. Related Work

There have been enthusiasts designing custom steering wheels for sim racing systems. Their design works like a detachable "button box", with only changes made to the style of the steering wheel and the layout of the buttons on the wheel. The wheel base, which offers the steering angle sensing and force-feedback system, remains untouched. These custom wheels, while not functioning the same way as our controller, offer great inspiration when designing our own.

There are game controllers with built-in gyroscope control on the market. However, none of them are tailored towards racing games, in the sense that they do not feel like a "steering wheel", and that they offer significantly worse immersive experience than dedicated steering wheel controllers. Their implementation, however, gave us the idea that integrating gyroscope control into a portable game controller is feasible, and that the level of accuracy, with the state-of-the-art technology, is high enough for good performance.

IX. Summary

Our design details our goal of implementing a wireless video game controller system capable of detecting gyroscopic input, dedicated to function with BeamNG.Drive. The hardware on the RPi will be capable of sensing button presses from the user and gyroscope data. The software will be able to transmit these data to the PC side via bluetooth to control the game running on it.

The design will significantly benefit the people who are looking for a mobile controller that can have all the core functionalities that an expensive game steering wheel has without worrying about carrying the bulky steering wheel around. The targeted users will be familiar with the buttons on the mobile steering wheel as we adopt a design from the expensive game steering wheel as well as generic game controllers. For example, the screen that displays the car information is a popular feature on game steering wheels while the analog buttons that mimic accelerator and brakes are common in generic game controllers. The users will find using our mobile steering wheel with familiarity and ease since they now can carry the controller around.

The most challenging part of meeting the requirements will be bluetooth communication latency and gyroscope accuracy. A slow communication can result in user input not reflecting on the game on time. An inaccurate gyroscope reading will cause the car to be not controllable as the car might move even when the users are holding the steering wheel horizontally.

Glossary of Acronyms

RPi – Raspberry Pi

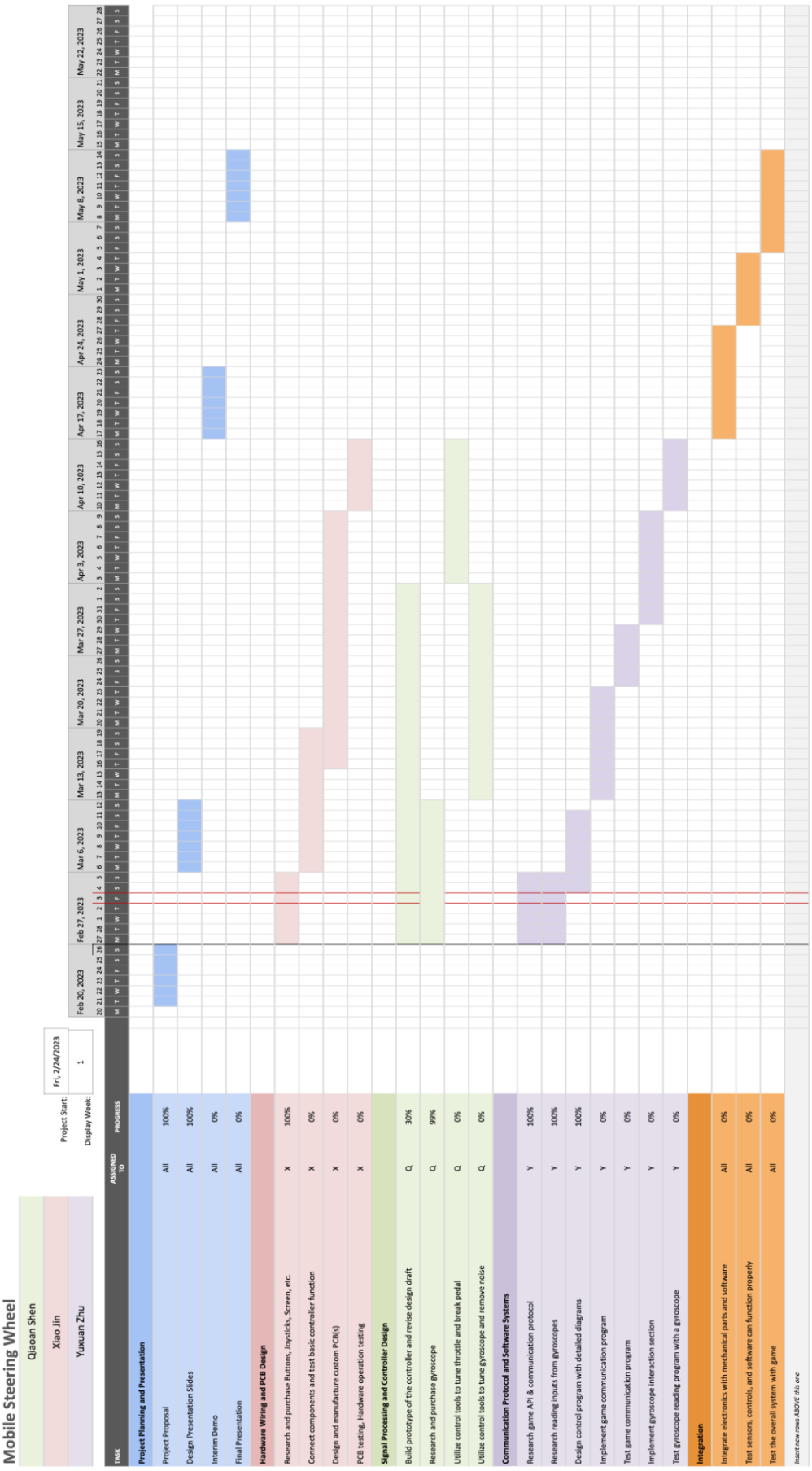

Fig. 4. Schedule example with milestones and team responsibilities

TABLE I. BILL OF MATERIALS

Description	Model	Manufacturer	Quantity	Cost	Total
Panel Mount Momentary Pushbutton	16mm	Adafruit	20	$0.95	$20.14
HDMI 5" Display Backpack	KD50G21-40NT-A1	Adafruit	1	$59.95	$63.55
SparkFun 9DoF IMU Breakout - ICM-20948	ICM-20948	SparkFun	2	$18.50	$39.22
Potentiometer with Built In Knob	10K Ohm	Adafruit	4	$1.25	$5.30
ADS1115 16-Bit ADC - 4 Channel	ADS1115	Adafruit	2	$14.95	$31.69
Pisugar 2 Plus: Battery for Raspberry Pi 3B/3B+/4B	Pisugar 2 Plus	Pisugar Kitchen	1	$49.99	$52.99