ECE 4971- Group 2 - MarioKart Bike Design Phase 2

Reed Hester, Chase Griffin, Leah Faulkner
ECE Students, Tennessee Technological University
Cookeville, TN, United States of America
lrhester42@tntech.edu
cagriffin42@tntech.edu
lmfaulkner42@tntech.edu

***Abstract* — This document expands further on the design described in phase one and thoroughly explains the inner working of each subsystem. Each subsystem includes an artifact that follows the given specifications and constraints.**

Keywords --- Capstone Design, gaming, emulation, exercise, sensors, magnetics, engineering, motors

I. INTRODUCTION

The MarioKart Exercise bike system is a device with the ultimate goal of offering an engaging and practical way to combine the benefits of physical activity with the fun and excitement of playing a video game. In order to achieve such a goal, a design that can communicate the user's motion into utilizable control data is required. It must also offer a similarly fluid experience in controlling the game as one would find with a typical game controller. To be viable for the average consumer, the design must be modular enough to be applicable to a standard bicycle design and non-invasive enough to the original frame to be easily assemblable. Above all, it should also maintain the safety and comfort of the user during operation as well.

The system has been broken down within this document into several functional subsystems, each with a specific purpose relevant to the goals previously outlined. The diagram below represents the conceptual structure of the project as a whole.

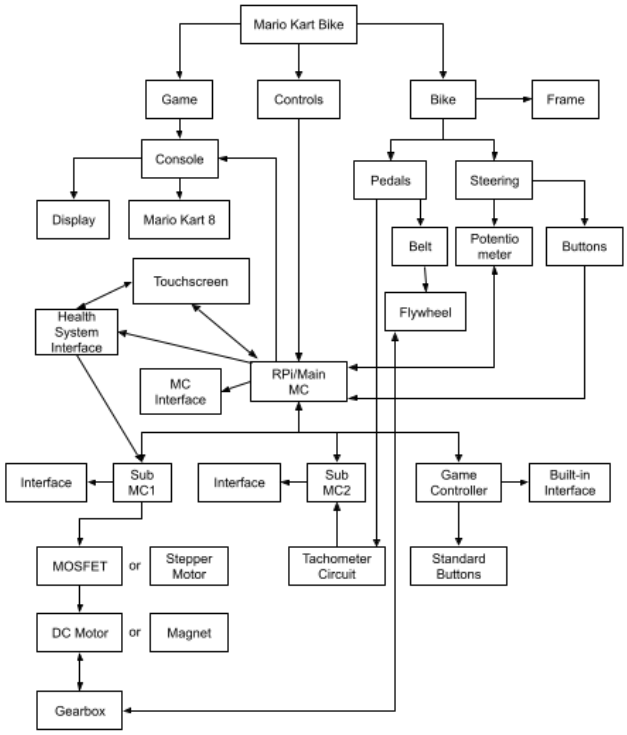

Fig. 1. Block Diagram of Mario Kart Bike System

As shown in the block diagram, the system will ideally have two versions: a DC Motor resistance system and a traditional magnetic resistance system. The rest of the subsystems on these bike designs will be exactly the same, with the only experimental variable being the method of resistance generation. A conceptual diagram with the location of each general connection between subsystems is shown below.

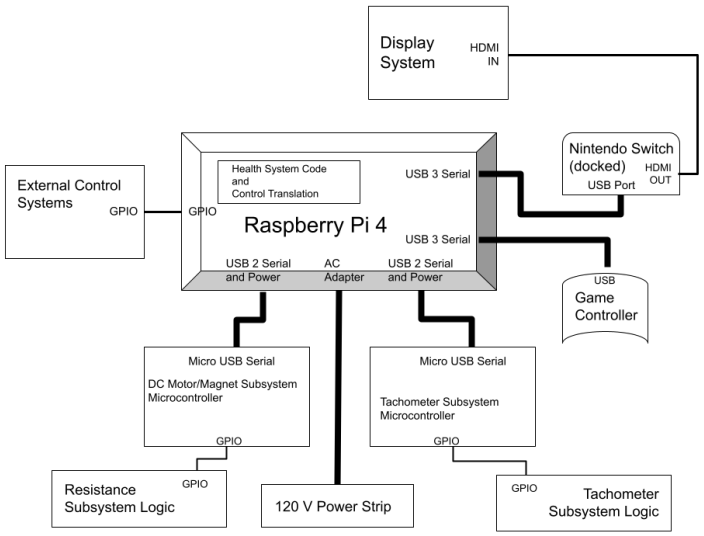

Fig. 2. Conceptual System Diagram

II. GAME SYSTEM

A. PC Subsystem

i) Running the Game

In order to run the game software itself, the project requires either a video game console or some form of emulation (i.e. running the game through emulation software on a personal computer). The decision was made to use the Nintendo Switch console to run the game, due to potential copyright issues pertaining to emulation. Additionally, the Nintendo Switch provides access to the latest version of MarioKart, which will allow users to enjoy the most up-to-date experience of the franchise.

ii) Displaying the Game

In terms of video and audio output, the Nintendo Switch console that is running the game will be connected to two PC Monitors via an HDMI splitter. This will allow two players to view the same game session on two different bikes with

separate monitors. Therefore, the players will be able to play together in real time as they normally would in-game. In addition, there will be a touchscreen display that serves as both a monitor for the health system and a selection menu for certain game and bike functions. The monitors and HDMI cables have already been freely obtained. The only component on the bill of materials for this system will be the splitter.

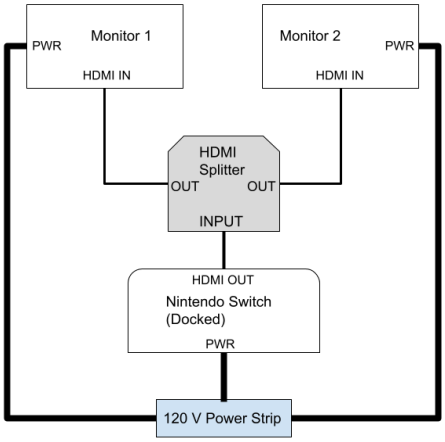

Fig. 3. Display Schematic

TABLE I. DISPLAY BOM

Part #	Description	Manufacturer	#	Cost per Item	Total Cost
B092VJXGJ6	1 to 2 HDMI Splitter	ZI YOUREN	1	$10.99	$10.99
60C8MAR1WW	23.8-inch FHD LED Backlit LCD Monitor	Lenovo	1	$240.00	$240.00
VE248Q	24" Full HD HDMI LED Monitor	ASUS	1	$217.77	$217.77

III. CONTROLS SYSTEM

The purpose of the controls system is to provide a means for the player to use their bicycle to play MarioKart. The system must interpret various mechanical inputs, like pedaling and steering, while also providing feedback to the player in the form of dynamic pedaling resistance. It must also contain all of the necessary control options to cover each in-game action like a standard game controller would. The system must also be modular for ease of replacing components.

A. Main Microcontroller

The main microcontroller in this design needs to cover both the control of data to and from the subsystems and the translation of signals into usable control data. Additionally, certain peripheral components for other game control purposes will be connected directly to the main microcontroller and interpreted according to their associated function (e.g. initial resistance selection). These desired functions necessitate ample GPIO and serial data connections, as well as the ability to power said subsystems and components. In order to keep the design as compact and modular as possible, each of these constraints should be met by a single microcontroller system controlling a series of sub-microcontrollers with less complex and more specific duties. Each GPIO pin can supply up to 16 mA of current, with a maximum possible draw of 30 mA. Each of the sensors we will be using require less than this amount for proper operation. The 5V power rails are able to supply power directly from the connected power source. The main microcontroller selected to fulfill the general needs of the design and serve as the overarching manager for each subsystem controller is the Raspberry Pi 4 Model B: 4 GB memory version.

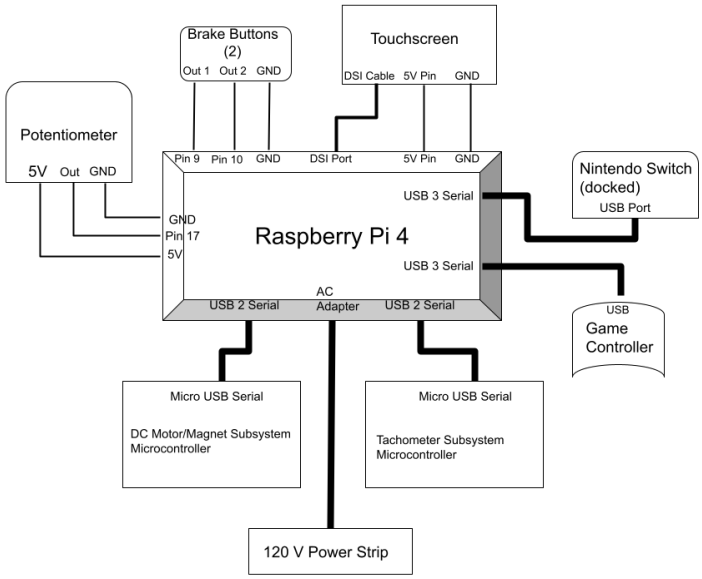

Fig 4. Raspberry Pi I/O Diagram

TABLE II. MAIN MICROCONTROLLER BOM

Part #	Description	Manufacturer	#	Cost per Item	Total Cost
RP4B4GB	Raspberry Pi 4 Model B - 4 GB RAM	Raspberry Pi	2	$55.00	$110.00

This version of Raspberry Pi (RPi) possesses two USB-2 and two USB-3 ports which will allow it to connect to both sub-microcontrollers through two separate data lines using mini-USB to USB Type A cables. This way, sensor information obtained from each subsystem and current game state information can be transmitted through serial data, and

each sub-microcontroller can be powered from those same connections. Additionally, the RPi contains a 40 pin GPIO header with two 3.3 V and two 5.0 V power pins, 4 dedicated ground pins, and several custom function pins that will be used to manage both the external buttons and the touchscreen device which all require direct connection to the RPi. It also contains an AC adapter which allows it to be powered from a standard wall outlet.

Control translation will take place directly on the RPi as well. Because it will be receiving player activity data from each subsystem, the RPi's primary purpose is to decipher the incoming signals and turn them into functional control information to be sent to the Nintendo Switch console and used for gameplay. There are several open-source, RPi-compatible libraries in existence that aid in the conversion of external data into control data (such as "NSGadget_Pi" which uses an RPi and other external electronic devices to impersonate a Nintendo Switch controller), and the RPi community has a large resource base of other programs with very similar functions that will help simplify the control operations as well.

B. Resistance Subsystem

i) Sub-microcontroller

The sub-microcontroller selected for both the DC motor and the magnetic resistance system is the Arduino Nano. This microcontroller contains two true analog outputs which are capable of outputting a voltage ranging from 0 to 5 V. Additionally, it contains 8 dedicated analog inputs that will be used to obtain feedback voltage and current values from the DC motor circuit. The fidelity of the digital-to-analog converter can be set to either 8 or 10 bits within the software. At the maximum fidelity of 10 bits, the user can adjust between 2^10 or 1024 partitions of voltage. Each partition modifies the voltage by approximately 4.883 mV. Though the analog input from the hall sensor will only reach approximately 0.7 V, an amplifier will be used to step this range up to 4 V, which leaves 819 partitions worth of fidelity. This should give the board an adequate degree of sensitivity for controlling the DC motor resistance system precisely enough to simulate a dynamic range of resistance settings that is both realistic and challenging to the user.

ii) DC Motor Resistance

The DC motor will serve as the main form of resistance for one of the bikes. Before choosing a motor to fit in this system, it's important to know what kind of load it will be receiving. Based on research, it has been found that the average person pedals a bike at 50-90 RPM with an applied torque of 30 ft-lbs. Since DC motors are built for the opposite (high speed and low torque), a gearbox will be needed to transfer the input from the pedals into what the motor can handle.

Using AutomationDirect's website, the team has found a gearbox and motor combo that will meet the required specifications.. The gearbox is a 20:1 speed reducer, but if it is run from the output shaft, it becomes a 1:20 speed increaser. The output shaft can handle 345 in-lbs (28.8 ft-lbs) at 88 RPM. Although 28.8 ft-lbs is slightly less than 30 ft-lbs, this gearbox will work because it's rated for that torque at the higher end of a cyclist's RPM. As a cyclist increases their RPM, their torque will decrease.

The chosen motor is a 0.33 HP permanent magnet direct current (PMDC) motor that is rated to run at 1800 RPM. Since a 1:20 ratio was chosen for the gearbox, if the rider of the bike pedals at 90 RPM (an average person's maximum effort) then they will match the rated RPM of the motor. The motor runs at 180 V with a peak current of 1.75 A, meaning the maximum power is around 315 W. The motor is sold in a 90 V variant as well, but the 180 V model was chosen because it has a lower peak current with the same power output. Average to healthy cyclists can produce anywhere from 100-300 W when pedaling, so this motor is rated to handle whoever drives it.

The reason a PMDC motor was chosen is because they have a constant current (at high RPM) and a variable voltage when used as a generator. The voltage has a linear relationship with the input RPM, while the current remains virtually constant once a certain RPM is reached. The speed increase from the gearbox ensures that the current will be in that constant range. The motor and gearbox are represented visually in the *DC Motor Control* section below.

iii) DC Motor Control

The purpose of the circuitry around the DC motor is first of all, to control the resistance across the positive and negative terminals, and second of all, to measure the voltage and current being generated by the motor during operation. In short, the resistance across the terminals will be controlled by an n-channel MOSFET, the voltage will be measured by an op amp configured as a diff amp, and the current will be measured by a hall sensor.

The point of measuring the voltage and current at all times is so that the performance of the MOSFET can be monitored. As the MOSFET is under operation, it will heat up. As the MOSFET heats up, it will become less conductive. To mend that, it will need a higher gate voltage. By monitoring the voltage and current being generated by the motor, the magnitude of that adjustment can be determined.

The MOSFET will serve as the main control of the connection between the positive and negative leads of the motor. Transistors in general are typically known as switching devices, where moving from a low state to a high state is the top priority. But for this use case, the MOSFET will be operated in its linear or ohmic region. Between the transition from low to high, there is a range of gate voltages that keep the MOSFET in a partially open/closed state. The goal here is to control the MOSFET with a microcontroller's analog output pin. That means the range of gate voltages for the ohmic region should be less than 5 V. The motor will be generating up to 180 V and 1.75 A during peak operation, so the MOSFET should be rated for such high parameters. The chosen MOSFET is rated for well over this use case, and the

ohmic region is between a gate voltage of 3-5 V. The specific package being used is the TO 247.

The protection diode included in the circuit is meant to be a safe-guard. Under normal operation, the current will be coming out of the positive terminal of the motor. However, if the rider were to pedal the bike backwards, the polarity of the motor would be flipped. That means the current would flow in the other direction. To prevent damage to other components in the circuit (mainly the microcontroller), a diode needs to be added. If the current comes out of the positive terminal like it should be, then the diode will see no current. If the current comes out of the other terminal of the motor, then the diode will be fully conductive and keep all of the current in a loop with the motor. The chosen diode is rated for 400 V and 2 A with a forward voltage of 1.1 V. The MOSFET and diode make up the bare minimum requirements for controlling the DC motor. They are represented in Figure 5.

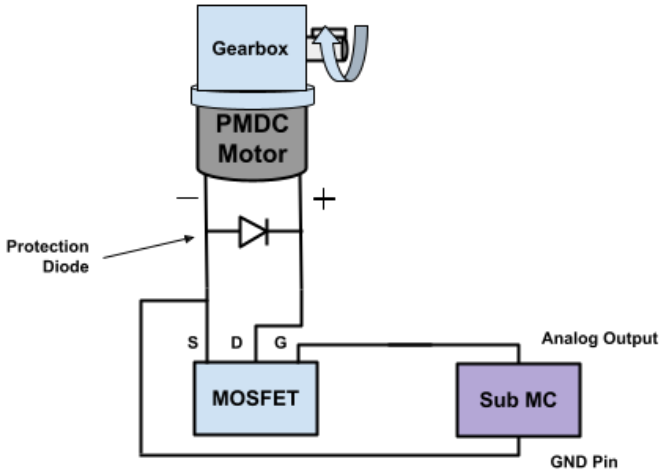

Fig. 5. MOSFET and Diode Circuit

The sub-microcontroller being used to control the MOSFET and collect the measurements of voltage and current is the Arduino Nano. This board has two true analog 5V output pins and several analog to digital input pins. The analog-to-digital (a2d) conversion happens on the board itself, and it has a configurable resolution. In the code, the programmer can set the fidelity to 8 or 10 bits. The team has chosen to use a 10 bit fidelity. 10 bits means 1,024 individual partitions. The pins can read signals up to 5V, which means there are 5/1,024 = 4.883 mV per partition. That's also 204.8 partitions per one volt. This will be a useful consideration because if the measurement devices can output a signal close to 5V, they can take advantage of the full resolution.

To measure voltage, an op amp will be connected across the positive and negative terminals of the motor. However, the 180 V being generated is too dangerous for most op amps and for the microcontroller. Therefore, the op amp needs to have a resistor network surrounding it that both protects its own inputs and configures the op amp into a differential amp. A network of four resistors can accomplish both tasks. By connecting the op amp in a negative feedback loop, it can be configured to have a gain less than one. That will be very important in stepping down the motor's 180 V to 5 V or less so that the microcontroller can understand the measurement and be kept safe. Aiming for 5 V or less also ensures that the op amp will be taking advantage of the full resolution of 1024 partitions. It is also important that the inverting and non-inverting terminals of the op amp don't see too much voltage or current. With some circuit analysis, these specs can be met. By setting R1=R2 and R3=R4, the process becomes even simpler. Through nodal analysis, the following equations were generated:

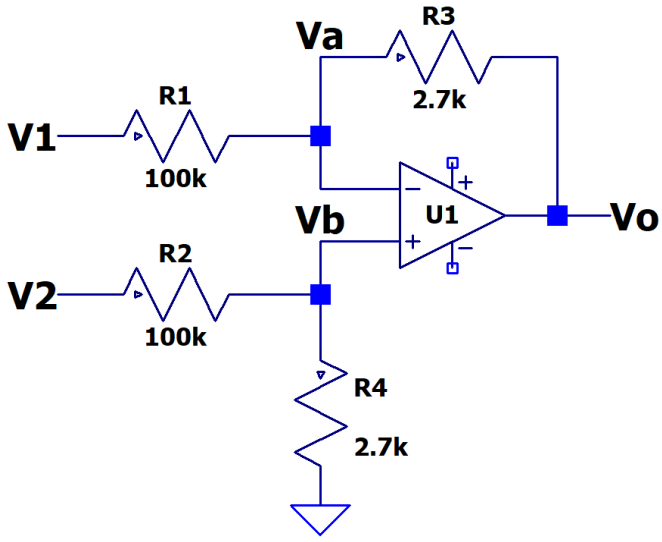

Fig. 6. Op Amp Voltage Measurement Circuit

$$V_a = \frac{V_O}{\frac{R_3}{R_1}+1} \quad (1)$$

$$V_a = \frac{V_2}{\frac{R_2}{R_4}+1} \quad (2)$$

$$V_O = \frac{R_3}{R_1}\left(V_2 - V_1\right) \quad (3)$$

$$I_1 = \frac{V_1 - V_a}{R_1} \quad (4)$$

$$I_2 = \frac{V_2 - V_b}{R_2} \quad (5)$$

The measurement will be taken by connecting the positive terminal of the motor to V2 and the negative to V1. The most voltage V2 will see is 180 V, and the least V1 will see is 0 V. Vo needs to be 5 V or less, meaning the ratio of R3 and R1 can be solved for. It comes out to 0.0278. Additionally, in order to keep the current low, all of the resistors need to be in at least the kiloOhm range. I've chosen to make R3 and R4 2.7 kOhms and R1 and R2 100 kOhms. With those chosen, that means:

$$V_a = 4.869\,V \quad (6)$$

$$V_b = 4.732\,V \quad (7)$$

$$I_1 = 48.69\ \mu A \quad (8)$$

$$I_2 = 1.753\ mA \quad (9)$$

With those specs in mind, an op amp can be chosen. The team has picked the LM 741 op amp, which can handle up to 30 V at each terminal.

To measure current, the team has chosen to use a hall effect sensor. It will have to be able to measure up to 1.75 A and potentially handle high voltage. The chosen chip is meant to have the full load of current injected into it. It is built with a layer of insulation between the input pins and the rest of the internal hardware. The hall effect takes place through this layer of insulation. Only the magnetic field will travel through the insulation. Then, the chip will output a voltage proportional to the current it is receiving. The chip will output 400 mV per amp it reads. This will leave the output voltage well under 5 V for the microcontroller. It is an 8 pin chip and pins 1-4 act as the IN and OUTs for the current to travel through. The purpose of including 2 IN pins and 2 OUT pins is so that the current can be split and a smaller trace can be used on the PCB the chip is mounted to. In particular, the A4 model should be used because it can read a minimum of 125 mA and a maximum of 12 A.

Since the chip outputs 400 mV per amp it measures, it will output a peak of 700 mV because the peak current of the motor is 1.75 A. To ensure that the resolution of the measurement is usable, the output of the hall sensor needs to be amplified. Since the resolution of the arduino's a2d conversion is 204.8 partitions per volt, if the signal is left unaltered, only 143.4 partitions will be available. The advertised peak current is 1.75 A, but just to be safe, it should be assumed that it could get higher. That means the chip's 0.7 V max should be amplified to around 4 V for safety. To accomplish this, another LM741 amplifier circuit will be implemented. The op amp will be configured as a non-inverting amplifier, which looks like this:

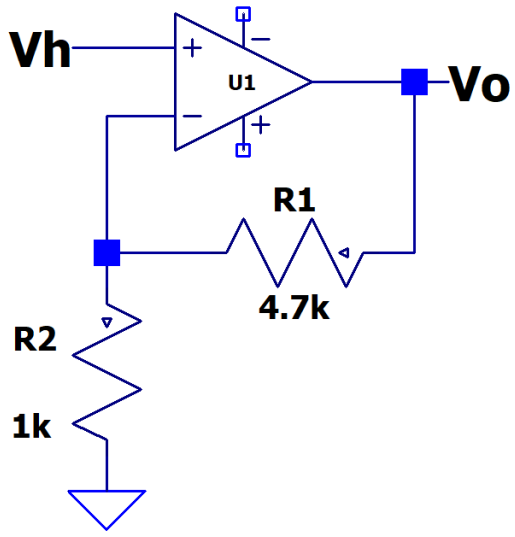

Fig. 7. Hall Sensor Preamp Circuit

$$V_O = V_h\left(\frac{R_1}{R_2} + 1\right) \quad (10)$$

$$\frac{R_1}{R_2} = \frac{V_O}{V_h} - 1 \quad (11)$$

$$I_O = \frac{V_O - V_h}{R_1} \quad (12)$$

Vh is the signal from the hall sensor, and Vo will be what is sent to the microcontroller. Vh has a typical maximum of 0.7 V, and to be safe, the op amp should output 4 V. That means the ratio of R1 and R2 should equal 4.714. If the maximum output is around 4 V, that means the resolution is 819.2 partitions. That's 80% of the maximum resolution.

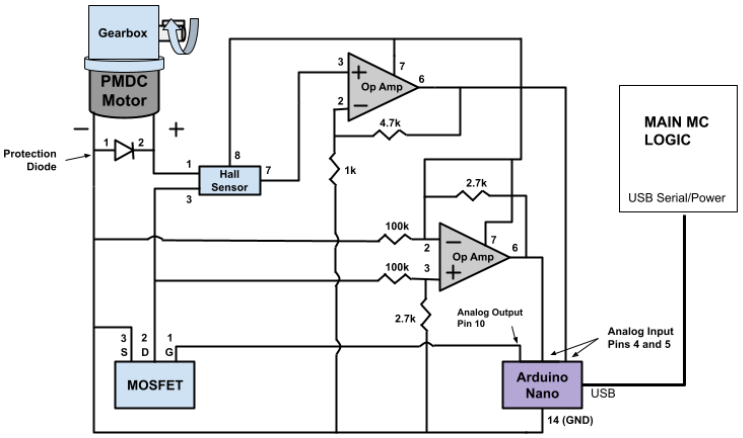

Fig. 8. Complete DC Motor Resistance Control Circuit

With all the components combined, the final circuit is shown above.

iv) PCB Design

The PCB must serve as a solid structure to mount the components of the circuit to and act as a conducting path for the appropriate connections. Sections of trace that are directly connected to the motor must be at least 40 mil in order to handle the max current of 1.75 Amps. There must be solder holes in places where a wire must be connected to the board. No connectors are necessary because the connections are meant to be permanent. The diagrams below show the layout of the PCB and highlight the trace connections.

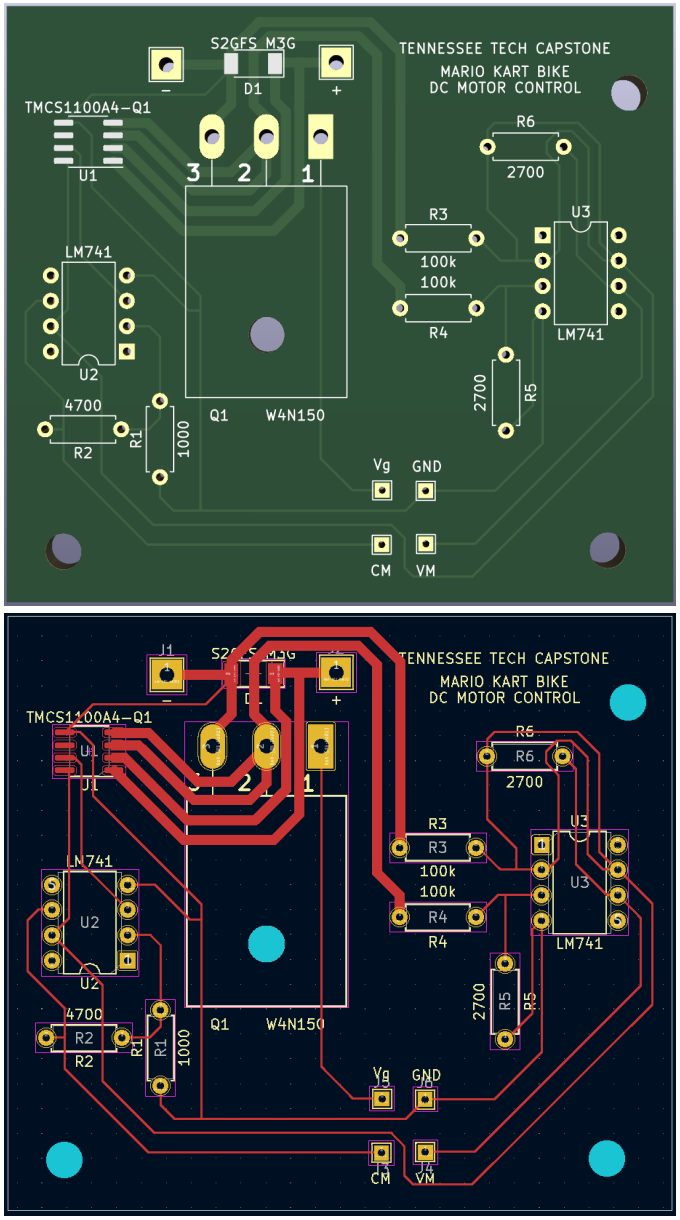

Fig. 9. PCB Design for the DC Motor Circuit

TABLE III. DC MOTOR CONTROLLER BOM

Part #	Description	Manufacturer	#	Cost per Item	Total Cost
A000005	Arduino Nano	Arduino	1	$20.70	$20.70
TMCS1100A4QDRQ1	Automotive Hall Effect Current Sensor	Texas Instruments	3	$3.76	$11.28
S2GFSM3G	Power Diode	Taiwan Semiconductor	3	$4.34	$13.02
STW4N150	Power MOSFET	STMicroelectronics	3	$8.01	$24.03
LM741	General Purpose Op Amp	Texas Instruments	6	$0.91	$5.46
CMF5550R000FKEK70	Carbon Film Resistor (Various Values)	Vishay Dale	18	$0.50	$9.00
N/A	Custom PCB (Set of 5)	JLCPCB	1	$2.00	$2.00

v) Magnetic Resistance

The magnetic system will serve as the main form of resistance for a second bike setup. The magnet must be electronically adjustable such that the distance between it and a flywheel can be modulated, and it must be able to provide resistance similar to what would be seen on various terrain inclines outdoors.

The most efficient method determined for use in the current bike design is to utilize a magnetic resistance component that is shipped along with the rear wheel frame, which will be shown in more detail later in the report. As for the magnetic component, it is essentially a magnet and flywheel working in tandem to provide resistance. It operates using eddy currents, the same principle used in designing auto-belay systems for climbers and repellers. There is a cable attached to the magnet that, when tensioned, moves the magnet closer to the flywheel. The closer these two get to each other, the greater the resistance.

vi) Magnet Control

The controller for the magnet must be able to electronically adjust the position of the magnet based on player preference and in-game factors. The current plan is to use a stepper motor to induce tension on the cable rather than the manual adjustment it ships with.

TABLE IV. MAGNET CONTROLLER BOM

Part #	Description	Manufacturer	#	Cost per Item	Total Cost
A000005	Arduino Nano	Arduino	1	$20.70	$20.70
TBD	TBD	TBD	TBD	TBD	TBD

C. Tachometer Subsystem

i) Sub-microcontroller

The sub-microcontroller selected for the tachometer subsystem is also the Arduino Nano. This microcontroller is required to convert the signals from the tachometer component into RPM in order for that information to accurately serve as the acceleration control in-game. It also possesses the minimal GPIO needed to operate the tachometer through the power, ground, and data pins found on the board. Additionally, the acceleration signal will be pulse width modulated upon leaving the Nano, not by using the PWM pins, but rather through a calculation within the program to allow uniform communication through the USB serial data lines, in increasing frequency based proportionally on increasing RPM. This will allow for a smoother acceleration control experience that more closely resembles the real-life process of gradually gaining speed when accelerating on a bicycle.

ii) Tachometer Board

For the tachometer component, this design necessitates the ability to optically measure the rotational speed of either the pedaling from the bike user or the rotation of some other mechanism on the bike. It also needs to return some sort of signal indicating the measured RPM or otherwise a signal that can be interpreted as an RPM value through some calculation made outside of or on the selected device..

The component that will serve as the optical tachometer designated for use in this subsystem is the KY-032 infrared obstacle avoidance sensor. This component contains an IR LED and an IR-sensitive photodiode that is capable of detecting reflected light from the LED off of some sort of reflective surface. The output signal returned by the device can follow both TTL and CMOS logic, and it sends a low level if an obstacle is being detected or a high level if no obstacle is being detected. The device contains two potentiometers, one that controls the flashing frequency of the LED through the NE555 timing chip, and the other that controls the brightness of the LED which allows for indirect control of the distance of detection by the photodiode. The board contains 4 pins, one for ground, one for power, one for the logical output of the system, and one for enabling the device. The enable pin is unused as long as the jumper on the board is kept in place, therefore it is left open. The control circuit for this subsystem is shown below.

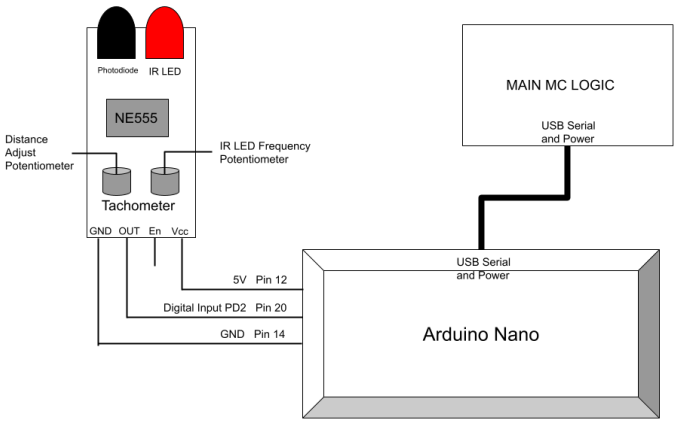

Fig. 10. Tachometer Control Circuit

The device's original intended function by the designer is to detect the IR light that the LED emits being reflected off of an object and subsequently signaling that an object has been detected in front of the device with a low logic signal. However, the true purpose of this device in this design is for use as an optical tachometer to measure the RPM of a spinning gear. This can be achieved by connecting it to the Arduino Nano and feeding it the detection data. Although this device is typically used to simply detect when an object is in front of it, the Arduino can be programmed to calculate RPM from the measurement. As previously stated, the component returns a logical low (~0 V) when the photodiode picks up reflected light from an object in front of it, and for a spinning object such as the pedals of the bike or the flywheel, it detects this signal multiple times with each pass of the object. The RPM can be determined from this by a simple calculation:

RPM = ((# of events * time interval of measurement) / (# of objects in rotation))

"# of events" is how many times the component has signaled a detected reflection, "time interval of measurement" is how many times a minute the RPM has been sampled, and "# of objects in rotation" represents the number of objects that pass in front of the detector that are part of the same spinning apparatus (e.g. there 2 pedals on the bike, but only one spinning apparatus).

This component can be directly powered and controlled by the Arduino Nano with it's digital GPIO pins and 5 Volt power pins, and it can perform all of the necessary calculations within the microprocessor on-board. Additionally, in order to achieve a more realistic acceleration control experience, the Nano can be used for pulse width modulation of the outgoing control signal in a way that causes gradual increase in speed rather than an all or nothing trigger.

The photodiode on the sensor has certain mechanisms to prevent signals that aren't intended for use from being picked up. This requires certain parameters to induce a desired logic signal that are outlined in the description below. The IR-sensitive photodiode used on this component is the HS0038B. It possesses a bandpass filter, an integrator stage,

and automatic gain control to filter out noise or disturbances from the desired IR signal detection. In order to distinguish a true data signal from said noise/disturbances, the signal received must be of or near the carrier frequency of 38 KHz, which is possible to achieve from the flashing LED using the tuning potentiometer and an oscilloscope to monitor the exact frequency as it is changed. Additionally, the reflected signal must be received in bursts of 10 – 70 cycles for optimal performance with a non-reflective gap time in between of at least 14 cycles. The datasheet for the device does not give an exact rise/fall time for the logic signal found at the output pin, however, the times for each component of the device are available. The circuit schematic of the device is shown in Figure 11 [1].

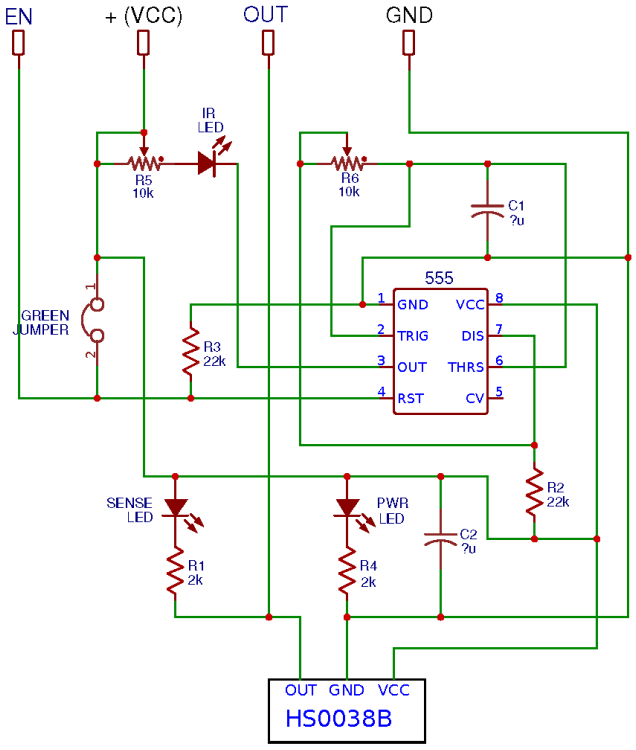

Fig. 11. Circuit Schematic of the Components on the KY-032 [1]

The only devices involved in the output logic signal are the HS0038B IR-Photodiode and the NE555 timing chip. The photodiode has a maximum rise/fall time of 100 nanoseconds, and the NE555 has a maximum rise/fall time of 300 nanoseconds, resulting in a total logic delay of 400 nanoseconds in a worst case scenario. At 38 KHz frequency, the LED will be flashing once every 26.316 microseconds. For 10 cycles to be achieved and including the initial fall time as the logic signal is set to low, the light must be reflected for at least 263.56 microseconds. The minimum gap time of 14 cycles will take up at least 368.824 microseconds of time, including the initial rise time as the logic signal is returned to high. Including these along with the dynamic response of the photodiode itself, this results in a minimum marginal total of 632.384 microseconds of operational time to achieve a sound logic signal. The object planned for use to measure RPM will be the gear of the bike itself with the holes and surface shrouded by a dark covering (black paper, tape, etc) and a white line on the covering to act as the reflective portion. The maximum estimated RPM for the gear when the bike is being used is approximately 90 RPM. This means there will be 1.5 rotations every second, or 0.6666 seconds per rotation. That equated to 666.66 milliseconds of time to work with in total for each rotation of the gear. Since only 632.384 microseconds (or 0.094%) of the rotation time is necessary for a functional logic signal, it will be simple to place a reflective strip of adequate width to both meet the minimum operational time needed and fit well onto the gear. Exact dimensions of the strip will be a minimum of 0.094% of the circumference of the gear in order to comply with this design. Figure 12 is a model of the design plan.

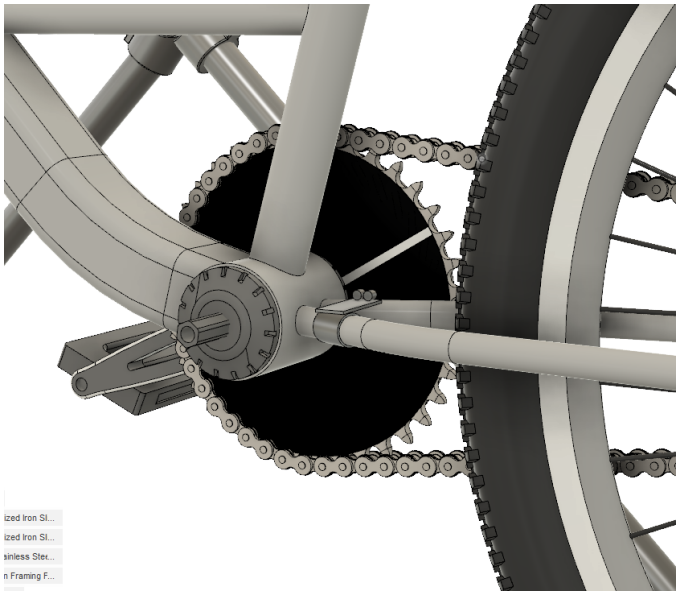

Fig. 12. Prototype CAD Model of the Tachometer Mounting Setup

The device is represented by the small gray plate and two cylinders in front of the pedal mount and is facing the gear. The gear has been shrouded with a black covering, and a white line, which will serve as the reflective patch for achieving the correct cycle count, has been placed on it. As the user pedals and the gear rotates, the white stripe will periodically pass in front of the device, causing the IR light to be reflected. It will be reflected for at least 10 – 70 cycles of flashing depending on the user's speed. This reflection will trigger the logic level of the device to low, and then a counter variable within the arduino program controlling it will increase. Once the white stripe is passed completely and at least 14 more cycles of flashing have passed, the light will no longer be reflected, and the signal will return to a logical high. This process will repeat until a certain amount of time has passed, and then the rpm calculation will be run with the result being sent to the RPi. This will continue again in its own cycle throughout the use of the bike.

TABLE V. TACHOMETER BOM

Part #	Description	Manufacturer	#	Cost per Item	Total Cost
A000005	Arduino Nano	Arduino	2	$20.70	$41.40
KY-032	IR Tachometer (Set of 3)	Keyes	1	$9.00	$9.00

D. Steering and External Controls

i) Directional Control

The control system design necessitates some way to convert the steering motion from the user's physical movements on the bike into directional movement and control data that can be used to correspondingly control the character in-game. The component that will serve this purpose must have a reasonable similarity to the steering function of a standard game controller. It must also be able to rotate with the bike at least 90 degrees in both directions for left and right turning. In order to obtain a precise and dynamic measurement of the steering direction, this design will include a potentiometer component that captures degrees of rotational motion from the user and returns that information to the Raspberry Pi to be interpreted as control data. The component selected for this system is a STEMMA 10k Ohm Rotary Potentiometer.

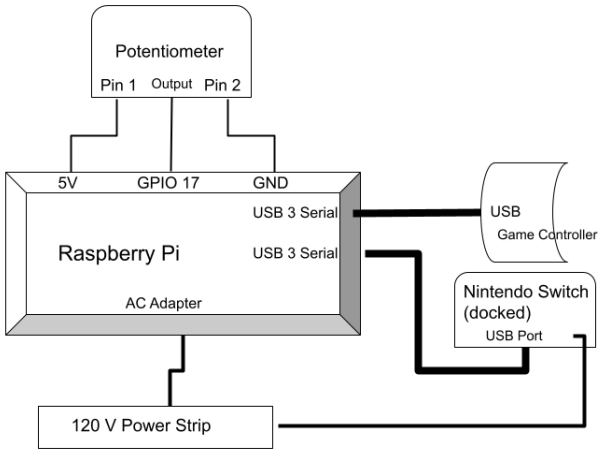

Fig. 13. Steering Component Schematic

This component functions like a standard potentiometer (i.e. variable resistance) where the turning of the dial increases or decreases the amount of current passing through it. The dial has a total rotational range of 270 degrees and will be mounted so that the midpoint at 135 degrees aligns with the center of the bike. Although the user would typically be turning no more than 90 degrees in each direction, the extra room will prevent accidental overturning from breaking the component. This potentiometer component is already contained on a board with connecting wires for each pin. It contains 3 total pins, one for power, one for ground, and the last one for the output signal. The power and ground signals can be directly connected to a 5V and ground pin on the RPi, and the output will go to an analog input GPIO pin. The analog input on the RPi has 10 bits of fidelity, which provides 1024 partitions, similar to the Arduino Nano. Since the user will only be turning 180 degrees total for gameplay (or ⅔ of the total arc), there are only 682 partitions that can be utilized. So, each partition represents 0.26 degrees of turning. This is comparable to the turning fidelity of the accelerometer on a Nintendo Switch controller. The partitions will be split in half, with 341 designated for each direction. The further to each side the user turns, the more partitions are counted by the receiving analog input pin on the RPi. This signal can be translated as a correspondingly increasing or decreasing turn signal in-game. In terms of the physical properties of the translation of bike motion into control data, this design is outlined in detail in the steering portion of the bike system below.

ii) Handlebar Buttons

The MarioKart game has several in-game control options other than the basic steering and acceleration operations. Throwing items at other players and drifting are part of the player's arsenal, meaning that this design needs some external controls to provide users with the same options as a standard controller. The design will include the attachment of clamping aluminum alloy bike brake levers. The levers will be detachable such that the system remains modular and can be applied to a variety of bike designs. The chosen brake levers are actually meant for E-Bike brakes, so they have a built-in switch. Using a realistic braking system simplifies the drifting and item throwing actions in-game and makes them easy to access while controlling the rest of the bike system. The type of switch used in those E-Bike levers is a "normally open" switch (connected/closed when the button is pressed). More specifically, the buttons will be pressed when the user applies adequate pressure to the levers, causing the circuit to connect. The resulting signal will be sent to the Raspberry Pi as control data for the corresponding button function through digital GPIO pins as shown in the diagram.

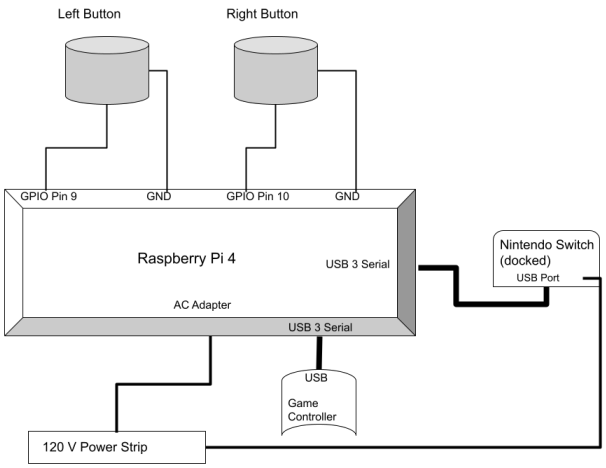

Fig. 14. Handlebar Buttons Schematic Diagram

The brake levers themselves look like any other standard bike brake lever, except they have two wires coming out of the back. The placement of these brake levers is shown in Fig. 16 on the bike CAD model.

iii) Touchscreen

An RPi-compatible 7 inch wide touchscreen tablet device will be employed as a secondary display. It will be connected through a dedicated ribbon cable directly to the RPi and will serve as the medium for customizable difficulty selection options as well as the method of interaction for the health monitoring system. The connections for this system are outlined in the diagram for the main microcontroller.

E. Game Controller Subsystem

The game controller will need to be present in order to preserve menu navigation for the user. The bike parts will only replace a couple of buttons on the controller, meaning that using the console otherwise would be impossible. The game controller is also likely going to be the communication device used to send the inputs to the console. The RPi libraries will send control data that is fed through the game controller and to the console. Therefore a standard console gamepad will be required.

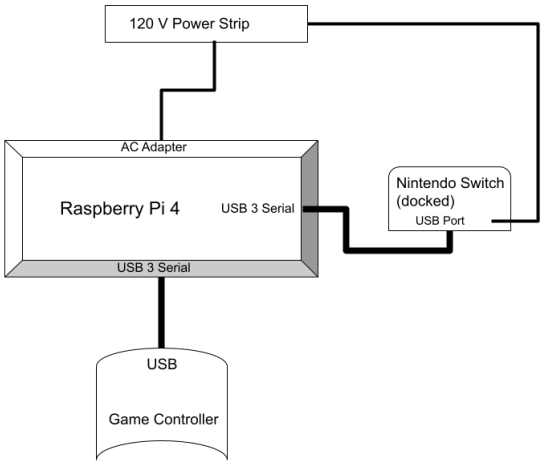

Fig. 15. Game Controller Schematic

F. Power

Much of the system will be powered through 5V pins on the various microcontrollers. The DC Motor and Magnet resistance circuits, tachometer board, steering component, handlebar buttons, and touch screen will all be powered this way as they require GPIO pin connection as the power source. The microcontrollers for each subsystem will be powered from the USB connections they share with the RPi. The console and monitors will be powered from a standard 120V wall outlet via a power strip located inside the microcontroller box.. The RPi will also be powered this way as it has a dedicated AC adapter on board. There are no batteries or external power circuits necessary for this design.

TABLE VI. STEERING/EXTERNAL CONTROLS BOM

Part #	Description	Manufacturer	#	Cost per Item	Total Cost
4493	10k Potentiometer	STEMMA	2	$3.95	$7.90
B07DR9S5C9	E-Bike Brake Levers with Switches (Set of 2)	Dioche	2	$21.99	$43.98
SC0025	Raspberry Pi 7” Touchscreen LCD Display	Raspberry Pi	2	$74.95	$149.90
B081ZQKXN4	Power Strip with 2 Outlets and 3 USBs	One Beat	1	$10.19	$10.19

IV. BIKE SYSTEM

A. Pedals Subsystem

This subsystem must accomplish the task of translating the rotational movement generated by the rider pedaling the bike into a controllable resistance via the magnet or DC motor systems. The main constraint is that very little modification should be made to the bike so that the user may still use their bike on the road.

This subsystem is already taken care of for the magnet bike variant. The magnet/flywheel housing comes with a roller that rests against the rear tire and is also what transfers the resistance from the magnet to the tire.

For the DC motor bike, the team has designed a support that will hold the motor/gearbox assembly. This support will act as the means through which the mechanical motion of the pedals is transferred to the DC motor. The details of this support are explored later in the "Frame Subsystem" section.

B. Steering Subsystem

This subsystem is the other main input that the rider should be able to use to steer in-game. This subsystem must allow for full range of motion of the bike's handlebars while also translating the rotational movement into game data. It must also be modular and removable for ease of use and easy installation. There are two components to this subsystem: the physical framework that allows the bike itself to turn properly as the user intends and the electronic component that will obtain that physical turning data and convert it into digital data to represent the steering control in-game. The digital component, already covered in the controls subsection, will of course be the potentiometer system. The physical frame component has its own set of constraints that will be met to maintain safety and functionality.

The first constraint of the physical frame will be met via the frame suspending the bike. This frame prevents the front tire from contacting the ground and still allows for full range of motion of the handlebars and will be detailed further in the frame subsystem section. The second constraint requires that the potentiometer be mounted onto the bike and connected to the turn shaft in such a way that the steering motions of the user translate to a corresponding physical manipulation of the encoder rod and thus a steering response by the in-game character. For this, a system of gears including a tall one mounted to the rod of the potentiometer meshed with a thinner one mounted to the turn shaft of the bike will be utilized. The gears will be placed such that the turning at the potentiometer rod is the result of the inverse of the user's direction, but this will be corrected within the translation code. The potentiometer itself will be held by a clamp placed on the front, static portion of the bike frame that encompasses the turn shaft. This design prevents any manipulation of the potentiometer save for the rotating rod, and it keeps the entire apparatus out of the way of the user. A model of this setup is shown below.

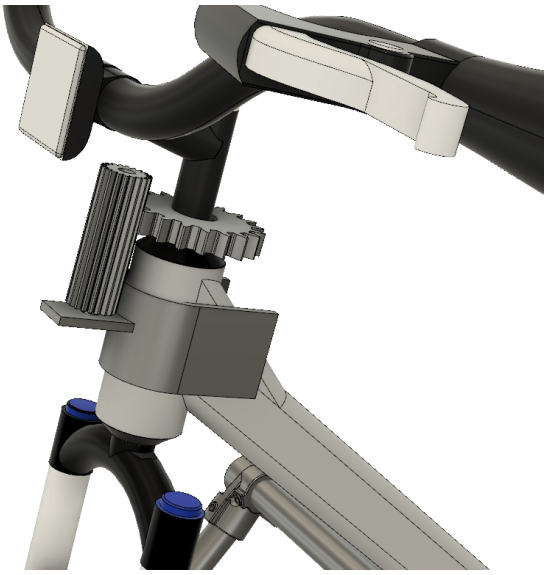

Fig. 16. Prototype CAD Model of the Potentiometer Mounting

C. Frame Subsystem

The frame of the bike must suspend the front and rear wheels off the floor. It must also hold the display that the game will be played on. The frame should be height-adjustable so that various kinds and sizes of bicycle may be attached. The frame should also be foldable or easily disassembled for storage. It should be rigid and wide so that the bike can't be tipped over from any angle. This is to ensure rider safety.

For the purpose of lifting the bike off the ground, the team has chosen to use a hybrid frame. The back wheel will be suspended by a bike trainer, and the rest of the bike will be held up by a frame designed by the team. The bike trainer serves the dual purpose of lifting the back wheel and providing resistance to it for the magnet bike variant. The chosen bike trainer comes with a manually adjustable magnetic roller, but the adjustment will be automated and controlled by a sub-microcontroller. The rest of the bicycle will be supported by a frame built in-house. The plan is to build it out of one inch steel pipe and connect the pipes using slip-on fittings. The display will be mounted to a vertical pipe at the front of the frame using a VESA pole mount. A concept model of this is shown in the next Figure.

Fig. 17. Prototype CAD Model of the Bicycle Supporting Frame

The bike trainer attaches using a special pin included in the packaging. This pin replaces the one on the bicycle and allows the bike to be mounted using a single clamp. The bike attaches to the pipe frame via a clamp that is screwed tight. The clamp is intended to allow bikes of various sizes and constructions to be mounted to the frame. It will be angle-adjustable and have a rubber interior for extra grip. The required fittings are a slip-on tee, two three-way 90° slip-on elbows, four 45° slip-on wyes, two 90° slip-on elbows, and a wye connector. The triangles on each side are to prevent the bike from falling forward or backward and side to side. The frame is wide enough to allow full range of motion for the front wheel to steer. The clamp is meant to be mounted in a position on the bike that doesn't obstruct the motion of the pedals either. The bicycle, bike trainer, pipe fittings, VESA mount, and display are models that the team acquired online. The bicycle was made by user Omkar Narkar on GrabCAD. The bike trainer was made by user Muhammad Furqan on GrabCAD. The display was made by user LeoDJ on GrabCAD. The pipe fittings and VESA mount were acquired from McMaster-Carr's website.

The motor support mentioned previously is meant to hold the motor/gearbox assembly in a way that allows it to act as a replacement resistance system for the bike trainer stand. The stand comes with a magnetic resistance system that presses against the rear tire. The idea here is that the motor support will mimic the attachment of the existing system. Below is a figure of the proposed motor support.

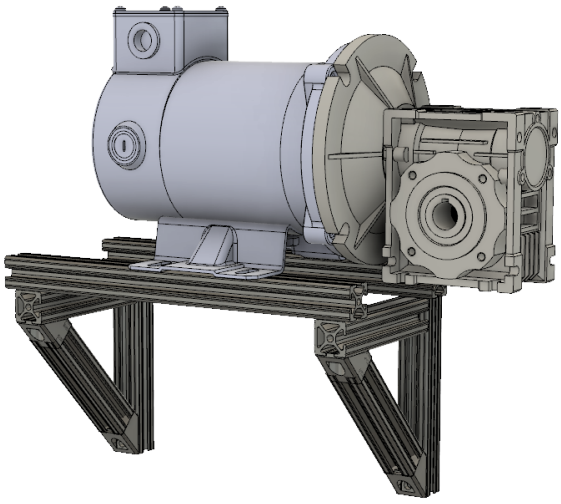

Fig. 18. Proposed DC Motor/Gearbox Support

The motor will mount to T-slotted framing rails that allow it to slide back and forth. This is needed so that the roller attached to the gearbox can be pressed against various tire sizes. The rails have two 45 degree braces underneath that will serve to support the weight of the motor. They will be attached to the rails using concealed brackets meant for fastening T-slotted framing rails together. This assembly will be attached to two mounting plates on the bike trainer stand. One plate has a vertical slot, and the other a horizontal slot. This allows for angle adjustment. The plates are pictured in Figure 19.

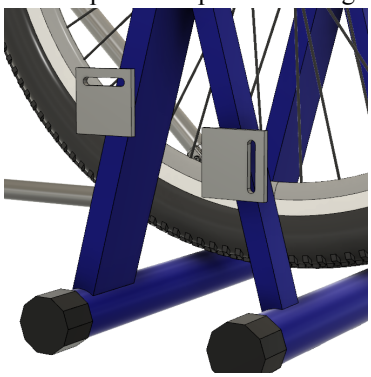

Fig. 19. Motor Support Mounting Plates

TABLE VII. BIKE FRAME BOM

Part #	Description	Manufacturer	#	Cost per Item	Total Cost
MTPM-P33-1M18	PMDC Motor	Automation Direct	1	$186.00	$186.00
WGA-40M-020-H1	20:1 Worm Gearbox	Automation Direct	1	$140.00	$140.00
4936T3	1" OD	McMaster-Carr	3	$85.09	$255.27

45	galvanized steel framing rail (10 feet)				
4698T31	90 degree elbow connector for 1" rail	McMaster-Carr	4	$10.52	$42.08
4698T96	Adjustable angle wye connector for 1" rail	McMaster-Carr	8	$18.16	$145.28
4698T86	Adjustable angle cross connector for 1" rail	McMaster-Carr	2	$26.92	$53.84
4698T73	Adjustable angle elbow connector for 1" rail	McMaster-Carr	4	$22.84	$91.36
NB-SC6-BS12	Tie down cradle for 1” tubing (set of 6)	BIKEGEAR	1	$49.50	$49.50
72457	Zefal "Cristophe" Mountain Bicycle Half Toe Clip S/M (CHOOSE S/M SIZE)	Zefal	2	$11.90	$23.80
8541751174	Universal Monitor Pole Mount Bracket with VESA 75/100	Loutytuo	2	$37.89	$75.78
B089NKPW39	DC Motor Bike Trainer	Unisky	1	$89.99	$89.99
B07PYHLTSX	Magnet Bike Trainer	Alpcour	1	$159.95	$159.95
47065T411	Single Four Slot Rail, Silver, 1" High x 1" Wide, Solid, 1’ Length	McMaster-Carr	5	$8.04	$40.20
47065T331	Diagonal Brace Ends for 1" High Single Rail T-Slotted Framing (Set of 2)	McMaster-Carr	1	$20.07	$20.07
47065T	Silver Corner	McMaster-Carr	4	$7.01	$28.04
236	Bracket, 1" Long for 1" High Rail T-Slotted Framing				
47065T736	Silver Gusset Bracket, 2" Long for 1" High Rail T-Slotted Framing	McMaster-Carr	2	$13.45	$26.90
8975K85	Multipurpose 6061 Aluminum 3/16" Thick x 3" Wide x 6” Long	McMaster-Carr	1	$4.90	$4.90

As for the mounting of all of the computing components to the bike frame, a box has been designed. The box houses the Raspberry Pi, two Arduino Nanos, the DC motor/magnet control circuit box, and the power strip. The team decided to put the control circuits in their own boxes due to the high voltage and current being used in the circuit. This smaller box will be velcro taped to the inside of the larger box for ease of removal. The power strip will also be velcro taped to the inner wall of the larger box. The two Nanos will be attached to the floor of the box using M1.6 standoffs and screws. The Raspberry Pi display has been integrated into the lid of the box, sitting perfectly flush. The Raspberry Pi mounts to the underside of the display. The walls of the box are talls enough so that the Raspberry Pi does not come into contact with the other components inside. The box has a large slot at the bottom for all of the cables to feed through. Below is a figure of the lid and the box with estimates of the component placement inside. The box will be attached to the front pole of the frame, just below the monitor, using zip ties or velcro straps. The box will be 3D printed as well.

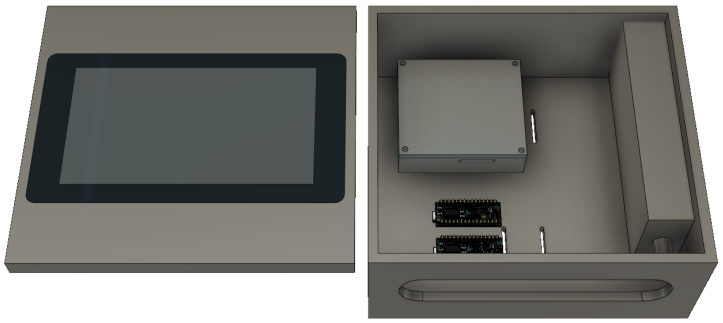

Fig 20. Computing Control Box

V. HEALTH SYSTEM

A. Calculating Calories: Permanent Magnet Version

In this subsystem, the RPi 7" touchscreen will receive the input of the user, then send the data to the Raspberry Pi via a Display Serial Interface (DSI) connector to be calculated internally and displayed on the touchscreen after the user ends the session. For the inputs, the customer must enter their age and weight (lbs), and then press start. The start button is the condition for the function *timeDuration()* to be enabled. When the user presses the stop button, the amount of time elapsed **(1)** will be calculated in this function and then the value returned will be stored in a variable and passed to *calculationMET()* to calculate calories **(2)** with a preset value for the Metabolic Equivalent Task (MET). With the stop button as a condition, the *display()* function will cause both timeElapsed and calories to be transmitted via the DSI connector to the touchscreen display.

Equation for calculating duration:

(1) timeElapsed = stopTime - startTime

Equation for calculating calories with MET:

(2) calories = T*60*MET*3.5*W

Where the values **T** = time (hours), **MET** = Metabolic Equivalent of the chosen task (a preset value), and **W** = weight (kg, converted from pounds to kilograms).

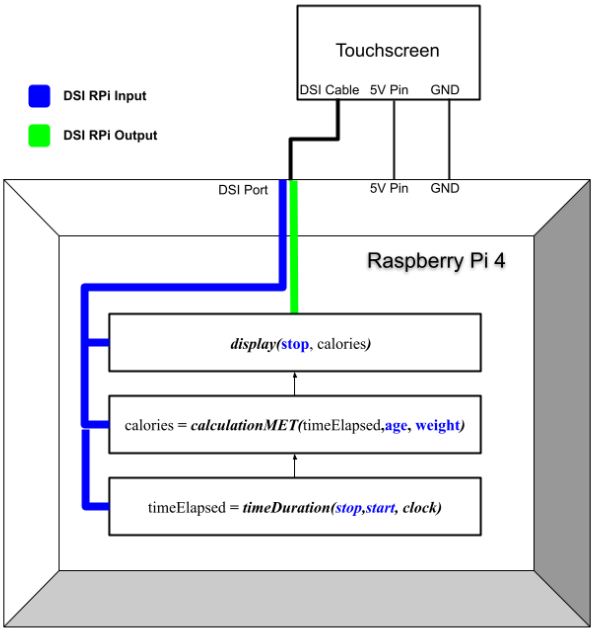

Fig. 21. Magnet Bike Health System

B. DC Motor Version

This subsystem should read the voltage and current from the Arduino Nano and use the values to calculate power and kilocalories, then output the time elapsed and kilocalories to the 7" display for the customer. The Nano has two analog values for inputs to the DC Motor control. The voltage has a range of 5 volts or 1024 in resolution for Analog to Digital Converter (ADC). The current has 4 volts, or the ADC resolution of 819. The following equation that can be used to convert from the ADC reading:

(1) ADC Resolution/System Voltage = ADC Reading/Measured Voltage

Since the given values would be the system voltage, ADC resolution, and the ADC reading, the measured voltage can be solved for. The same can apply for the current, which is represented by the system voltage of 4 volts.

Example Arduino Code to Demonstrate ADC Reading:

```
//Defining pin as input
pinMode(A0, INPUT);

//Read analog value of A0 and store in y
int y = analogRead(A0);

//Printing would show the value in ADC resolution
Serial.println(y)
```

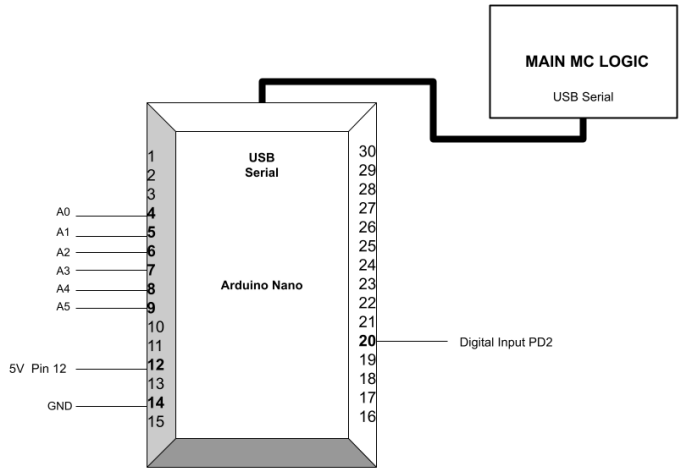

Fig 22. Motor Bike Health System Data Connections

In order to communicate with the Raspberry Pi (labeled as Main MC LOGIC), the Nano uses a serial USB port. There are libraries that help the Raspberry Pi identify the Nano and access its analog pins. In order to avoid a common issue of having to manually enter the Arduino's serial device name, there is an available "sketch" project for Arduino and a library

for RPi that has APIs (Application Programming Interface) in C and Python. The link for this solution follows:

https://www.uugear.com/uugear-rpi-arduino-solution/

In addition to the serial USB communication, the user input for start and stop via the DSI of the 7" RPi touchscreen will be input to the system. Once the user presses stop, the RPi function *caloriesPower()* should use the time computed from a PLL clock, the time elapsed, as well as the current and voltage transmitted from the Arduino Nano serial USB port to compute the power. The code structure flow can be demonstrated from the following equations:

(1) Power = Current * Voltage

The voltage and current will be measured every second in a loop in conjunction with the time elapsed to calculate power.

(2) Kilocalories/Hour = Power * 0.860421

The power will be used as a variable to a second equation to calculate kilocalories per hour.

(3) kcal = Kilocalories/Hour * timeElapsed

The third equation in the function will compute the total kilocalories burned in a session.

Once the user presses the stop button, the arguments to the *caloriesPower()* function will be the kilocalories and the time elapsed. These variables will be displayed via the DSI connector to the RPi 7" Touchscreen.

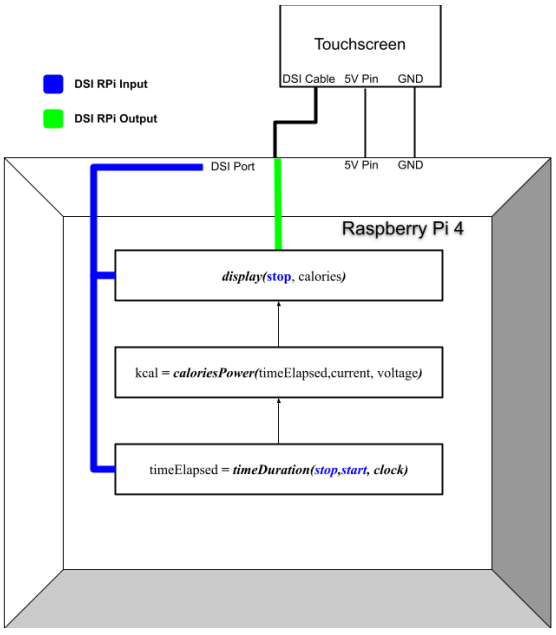

Fig 23. Motor Bike Health System

Example: Pseudo Code for Time Duration

```
timeDuration = functionTime(startVariable,stopVariable)

If (startButton)

{
      startVariable = I2C_data (at time stamp)
}

If(stopButton)

{
     stopVariable = I2C_data (at time stamp)
}

functionTime(start, stop)
{
      Time = stop - start
      (Return Time)
}

If(stopButton)
{
   (display timeDuration)
}
```

Fig 24. Pseudocode

VI. REFERENCES

[1] B. Ryerson, "IR sensor for obstacle avoidance KY-032 (AD-032)," *IR Sensor Obstacle Avoidance Keyes KY 032*, 21-May-2021. [Online]. Available: http://irsensor.wizecode.com/. [Accessed: 27-Jan-2022].