Game System:

The game system consists of all the parts required to run and display MarioKart itself. The game version chosen for the project build is MarioKart 8 on the Nintendo Switch console. An HDMI-capable monitor is used for the display and sound. The Nintendo Switch dock station is used to output to the screen.

Interface:

The interface for the game system consists of a few key points. The Switch should be placed into the powered dock so that the outgoing HDMI port is connected to the console datastream. An HDMI cable will connect the dock to the monitor so that audio and video data are passed to it. The monitor then will display the game such that the user can play comfortably. The Nintendo Switch bluetooth capability must be working and enabled so that it can connect to the control system. In order to initially connect to the simulated RPi controller, the Switch must be placed into the “Change Grip/Order” menu in the controller connection settings. Once the RPi controller is connected, you can leave that menu via the touchscreen of the Switch (“back” button) or via the touchscreen GUI options. Using another controller or the onboard JoyCon controllers for navigation can cause issues with the connection of the simulated controller and is unadvised. Once the game is selected and running, and a race and character options have been selected, you can use the bike to play the race as intended. If disconnections occur during gameplay, the Switch should automatically pause the game and bring up the connection menu. Do not use the JoyCons or the Switch touchscreen to exit this menu; instead, simply re-run the RPi controller program and it will reconnect at this menu as well. Once reconnected, play can resume.

Power:

This system is simply powered by the Nintendo Switch’s wall wart adapter. This adapter goes straight into a 120V receptacle in any wall. This is to simulate that the user may be keeping their switch in a TV cabinet or gaming center, so it’s best to not interfere with that existing power connection.

Control System:

The control system consists of several features with specific operational necessities. The core components include the main microcontroller (Raspberry Pi) and the python program that serves to translate, package, and transmit the compiled control data, the sub microcontrollers (Arduino Nanos) that govern the external sensors and controls for the acceleration data and motor resistance setting, and all of the external buttons, sensors, and other input devices themselves that serve to collect the individual pieces of control data for each subsystem.

Interface:

RPi:

The RPi contains the modified open-source Github-hosted “Joycontrol” program that translates all of the external control system data coming in from the user’s operations into control data that appears to the Switch console as having come from a standard Pro controller. It is a Python 3, event-driven, object-oriented program that collects controller input data from user input via the terminal, and, through a complicated simulation of a controller MCU and Switch control data packaging, converts said input into controller output. This program was modified such that it can work with gpio signals, I2C data, and serial data over USB. It has only been tested on the Raspian GNU/Linux 11 (bullseye) OS. To begin the program, ensure all external connections are made to the RPi (exact pins and ports are defined in the README of the program), check that RPi bluetooth is enabled, and then change into the “joycontrol” directory and open the terminal. To run the software, use the following command:

sudo python3 run_controller_cli.py PRO_CONTROLLER

This command will execute the driver file and should begin attempting connection to the Switch. Ensure the Switch is powered on with bluetooth enabled and the “Change Grip/Order” menu open. The program should recognize the Switch automatically and attempt to connect. Upon successful connection, the terminal will appear to pause waiting for some kind of input. The connection should appear on the Switch controller menu, and the bike/program will be ready for use. If you wish to see the exact control data being sent in real time, simply press the “enter” key once in the terminal and the data should begin appearing rapidly. If any errors occur during connection, the best course of action is just to end the program and attempt reconnection. (Frequently noted issues are described in the README).

Tachometer & Nano 1:

The tachometer function is achieved through the operation of the KY-032 Obstacle Avoidance Sensor and the Arduino program that calculates RPM through a specific equation.The KY-032 must be placed such that the IR LED and IR-sensitive photodiode are facing the rotating surface (in this case the tire) and the LED brightness must be tuned via the on-board potentiometer such that the measuring distance is low enough to only detect reflection off the reflective partitions on the tire (or other rotating apparatus). It should be powered and grounded via GPIO connection to the Arduino Nano, and the output should also

be connected via another GPIO pin. The device will return a logical low output when an object is detected. It is the Nano program's responsibility to perform the RPM calculation from these signals. The Nano contains a function that calculates the RPM based on the number of partitions detected in a certain time interval:

$$RPM = \frac{\#\ of\ events\ \times samples\ per\ min}{\#\ of\ object\ partitions}$$

The result is obtained repeatedly as the user pedals. In order to pass this data on to the RPi and allow it to be used in-game, make sure the Nano containing this program is connected via USB to the RPi running the translation program. Then, it will begin sending serial data containing only the integer RPM value at 9600 baud rate (utf-8) over the serial connection.

Motor Control & Nano 2:

The motor control is done via a relay component and another Arduino Nano. The relay passthrough pins must be connected to the leads of the motor, and the relay must be by default open (and therefore not shorting the leads of the motor). Then, 3.3 V power and ground from the Nano should be connected to the relay control lines. Lastly, connect GPIO pin D6 to the input signal of the relay to control the switching. Most importantly before operating the system, isolate the motor leads and relay connection and do not allow contact with it as the motor generates high voltage and current in certain situations. The relay is operated by the Arduino Nano program. It begins in the open position, and the program runs a function that causes a random amount of delay between 10 and 45 seconds. After the delay, the relay is switched on for approximately 20 milliseconds, and then switched off. This cycle repeats throughout the race. The Nano is connected to the RPi via USB for power purposes. It originally used the serial data line to receive information from the RPi, but this is no longer utilized.

Steering:

The steering is managed via a gear system, a potentiometer, an analog to digital converter (ADC), and the RPi program. The gear system is set up in a 1-to-1 ratio with one gear around the turnshaft of the bike and the other attached to the knob of the potentiometer. This works to translate the motion of the user turning the bike into electronic data. The potentiometer returns a variable voltage in analog form as it is turned. The output of the potentiometer is connected to the first analog channel of the ADC chip. The ADC chip applies a ⅔ gain to the incoming signal to give us the full range of voltages from the potentiometer in a more condensed form. It then returns a digital signal from 0 - 27648 via I2C

indicating the position of the potentiometer. This is received by the RPi and converted into analog stick directional data through a logarithmic mapping. Because of the gear system, the output of the potentiometer is actually the reverse of the turning motion of the user, but this is easily inverted in the program. Also, the measurable range has been limited to a maximum of 90 degrees in either direction as any further angle is impractical for the bike user. In terms of connections, the gears must mesh together such that the potentiometer turns exactly when the bike handles do. If the handles are turned too far, it may break the potentiometer knob at the ends of its range, so this must be avoided. The potentiometer power and ground pins must be connected to RPi 5V power and ground respectively. The same goes for the ADC chip's power and ground pins. Additionally, the I2C data is done over the SDA and SCL pins on both the ADC and the RPi.

RPi Touchscreen:

The RPi touchscreen contains a GUI which allows the user to interact with the MET health data program. There is a keypad that allows for age and weight data entry and timer buttons that allow the user to start and stop the time during a race. After the timer is stopped, the caloric burn value is calculated and then displayed to the user. There is also a basic D-Pad and A/B buttons for in-game menu navigation. The pin connections for this are as follows: D Up - 27 to 23, D Down - 22 to 24, D Left - 5 to 25, D Right - 6 to 8, A - 13 to 7, B - 19 to 12. The data and power between the touchscreen and the RPi is connected via the ribbon cable and the DSI display port on the board.

External Buttons:

There are two handle bar buttons that serve as the "ZL" and "ZR" buttons found on a Pro controller. They are activated when pressed back like a standard bike brake lever. The buttons are electronically connected to the RPi via one GPIO pin and one ground pin each. Pin 7 connects the left button and pin 11 connects the right button. Once they are signaled, the program translates that signal into the corresponding controller button and sends it to the game.

Power:

The control system is powered by a single power strip connected to one plug on a 120V wall receptacle. This power strip has two 120V plugs and two 5V USB plugs. The RPi is plugged into one 120V plug, and the other plug is used by the display. The fans are powered by one of the 5V USB plugs on the power strip and they are connected to a USB splitter. The current draw of the fans is 0.25 A each, and a USB

can output up to 0.9 A. That means that using a splitter still supplies both fans with ample current.

The RPi serves as a power distribution of sorts. It draws 260 mA in an idle state and even more when there are devices connected to its USB ports. It has 4 5V USB ports on it, which the two Arduino Nanos are connected to. Each of the Nanos draws 280 mA. Each of the MCs has power and GPIO pins which supply power to the respective sensors. The KY-032 module draws a max of 20 mA at 3.3V. The potentiometer draws a maximum of 0.5 mA.

Bike System:

The bike system contains all of the parts necessary to give the stationary bike mechanical functionality. This includes the support frame, handlebar lever buttons, steering, pedaling, and transmission of motion to the resistance system.

Interface:

The bike system is housed on the steel pipe frame. Its purpose is to support the rider and house the other physical equipment and sensors. It also supports the bicycle provided by the user. Simply attach the bike to the frame using the rubber strap on the bike mounting cup. It may be easier to mount the bike with the front wheel removed, but that depends on the type and size of the bike.

The sensors are meant to be mounted to the bike itself. The potentiometer goes at the front and receives input from the 1:1 gear system attached to the bike's steering shaft. The shaft has a gear that has been split in half and rejoined with screws, while the potentiometer has an equally sized gear pressure-fit onto the knob. An L-shaped bracket goes underneath the potentiometer, where the ADC chip is housed as well. The L-bracket is attached to the bike using hose clamps. The intention is that each of these pieces is removable so that the bicycle can be used as a normal bicycle in the future. The tachometer is mounted directly to the frame of the bike next to the rear tire using a C-clamp. It is mounted so that the IR LED points towards the partitions on the tire. The use of a C-clamp also makes this sensor removable. The handlebar buttons act as the drifting and item-throwing controls. They are E-bike brake levers, so they mount flawlessly to bicycle handlebars.

The rear tire goes on the bike trainer stand. This stand comes with a roller that allows for the transition of motion directly to the resistance system. The bike mounts to this stand using two bolts that press against the rear tire shaft and suspend it in the air.

All of the wiring for the sensors is zip-tied to the frame for a secure route to the control box.

Power:

The bike system draws no power as it is a purely mechanical subsystem.