Team Name: MoJa
Team Member's Names: Aayush Agarwal, Omar Syed, Josh Antonson, Mikhail Kutsovsky
Date: 5/13/2015
Project Title: RideThru

TABLE OF CONTENTS

0. Abstract

This project brings the latest developments in virtual reality gaming to the world of exercising. An exercise bike was fitted with a PCB that connects an RPM sensor, a fan, and a handlebar game controller to a computer that runs a virtual reality simulation on an Oculus Rift. This way a user can bike normally in real life, and see their actions translated into movements in the virtual reality. The experience is as if they were actually moving in the virtual reality, instead of biking on a stationary exercise bike. The RPM sensor is made from a reed switch and magnet which is placed on the rotating wheel of the bike. Everytime the magnet passes the reed switch, one rotation is counted. This is combined with the handlebars which are a PlayStation 2 controller that allows the user to be able to turn in virtual reality. The fan simulates wind which is done by reading the RPM of the bike and receiving signals from the game itself. These three components all connect to a single PCB which then connects to a computer and acts like a controller for the virtual reality simulation. On the software side, a cross platform driver and a sample game were created to experience the bike. The driver reads the information sent by the bike and maps it to game controls allowing the bike to be used with any kind of Oculus Rift game. A sample game was specifically created for the bike to showcase the functionality and also communicate back to the bike, which other games that are not designed for the bike cannot do. By doing this, we enhance traditional stationary workout methods by making them more immersive and fun while also bringing developments into virtual reality gaming by adding a physical component in addition to the preexisting visual component that the Oculus Rift provides. This also makes video games healthier by keeping people active while playing the game.

1. Project description

Many people prefer to bike outside but often due to time or weather constraints they choose to use indoor, stationary bikes. The problems with stationary bikes are very well known and the chief complaint is usually how mundane the workout is compared to a bike ride outdoors. People often try to combat this by watching TV or listening to music, but with today's technology it is possible for people to experience their ideal workout while still indoors on a stationary bike.

Utilizing a virtual reality headset, Oculus Rift, we created an embedded system that connects the exercise bike to the virtual reality displayed by the Oculus Rift. This creates an immersive workout experience that is even more exciting than an outdoor bike ride. The Oculus Rift can be used to immerse the user into many different types of environments ranging from bike trails, different cities, and even video games. Furthermore we can overlay the virtual reality scene with the user's workout information like resistance level, speed, and distance traveled. A user exercises with the indoor bike as they normally would but have an experience as if they were biking through their favorite scenery or playing favorite video game in first person.

We modified an existing exercise bike to communicate with an Oculus Rift and respond to the actions within the virtual environment of the Oculus Rift. In order to accomplish this, we augmented the bicycle with sensors and hardware to control movement in the virtual reality. There is a software component as well in creating the virtual reality landscape and letting movement be controlled by peripheral devices. This allows the user to experience both a visually and physically realistic experience due to real-virtual parallels such as turning, and wind from a fan during accelerations. Further modifications can also include dynamic resistance in response to incline changes in the virtual reality. This exercise bike is ideal for making people want to workout more and have more fun doing so than any other workout method. With rising national obesity rates and higher than ever demand for video games, RideThru aims to help people have the most fun while also exercising.

2. Design Requirements

Figure 1

We are using the bike in Figure 1 as our initial starting point. We added sensors to various parts of the bike to make it ready to interface with Unity and the Oculus Rift. This bike by

itself is not outfitted with any sensors, and has a manual resistance knob and a fixed handlebar. We will add the following usability through sensors and hardware to the bicycle. These are ranked in order of priority, with the sensors to determine RPM and rotating handlebars for turning being our top priorities. Creating a mechanical, motor controlled system to dynamically change the resistance of the bike is one of our stretch goals.

1. Pedal sensors to determine speed (RPM) and to estimate distance biked/calories burned
2. Rotating Handlebars for turning (similar to a real bike)
3. Adjustable-speed fan for a more interactive feeling
4. Motor Controlled Resistance
5. Bodily Sensors (i.e. heart-rate, etc.)

These sensors (1, 2, 3) link to a microcontroller which acts as a middle layer interface between the bike and Unity/Oculus Rift. The microcontroller, as part of a PCB, serves to achieve the following functionality:

- The speed at which a user pedals (RPM) is sent from the pedal sensor to the microcontroller and then to Unity. As a result, the speed of the user's character in the Oculus Rift environment will dynamically change to correspond to the speed that the user is pedaling on the bike.
- Read data from the handlebars and relay this information to Unity in order to control the game. The microcontroller is be connected to the Unity software via a driver written in Processing. As a result, the direction the user's character is moving in the Oculus Rift environment will dynamically update to correspond to the direction the user is pointing the handlebars on the stationary bike.
- The microcontroller can receive information from Unity regarding the state of the game within the Oculus Rift environment in order to reflect those changes to the biker. It controls a DC motor variable-speed fan that blows on the user depending on the acceleration for the user in the game. For example, if the user is going down a large hill in the game, then the fan will be blowing at maximum speed to mimic the real-life feel of riding downhill. As part of our stretch goal, we would like for the microcontroller to change the resistance of the stationary bike correspondingly to the game environment in a similar fashion to the fan example mentioned prior. For example, if the user is going uphill in the game, then the resistance on the stationary bike will increase, and vice-versa if they are going downhill. This will be implemented in later stages of our product.
- The microcontroller will also be potentially recording information from bodily sensors regarding the user and they will be displayed in the Unity game environment once the session is over, such as heart-rate. We will also display an estimate of calories burned, total distance biked, etc. in the Oculus Rift. This too will be added in later development stages.

3. Architecture

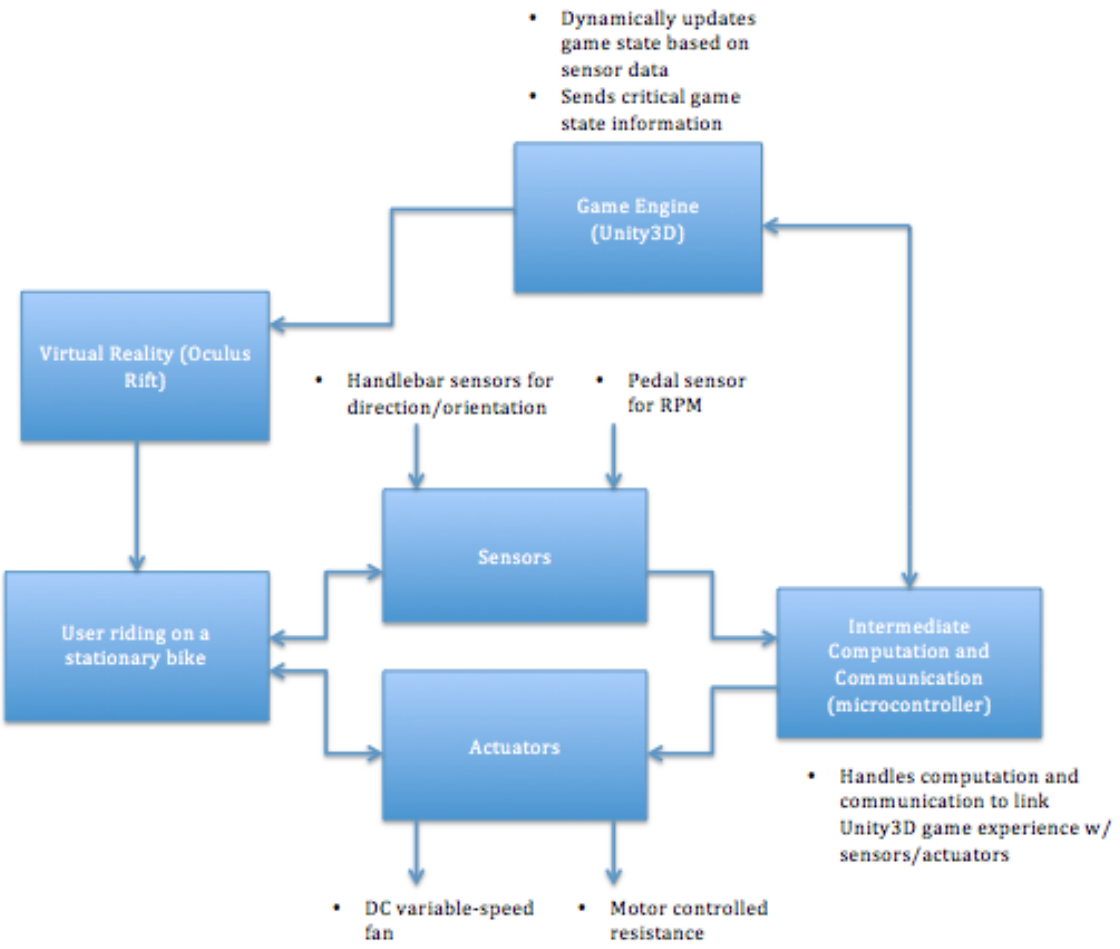

Figure 2

4. Design Trade Studies

4.1 Oculus Rift Connected to PC with HDMI vs WiFi

The standard way of connecting the Oculus Rift to the PC is via a USB and HDMI connection. However, there are a few people who have used a Raspberry Pi and USB Wifi Dongle in order to communicate between the Oculus Rift and PC via a Wifi connection. While this would be ideal for the potential commerciability of our device, the video feed has latency of 100-150 ms. This is a complete deal breaker for us, as our system depends on real-time responsiveness between the user physically biking and their experience in the virtual environment. Because of this, and the fact that physically connecting the Oculus Rift to the PC via USB and HDMI connections is merely a minor inconvenience for the user, we did not pursue the Wifi option.

4.2 Video Game Controller vs Handlebar Controller

An important design design that we needed to make was how we were going to allow the user to control the motion of the bike. Since stationary bikes don't have handlebars that can

move, we need to add our own handlebars to the system. We narrowed our options down to two choices: bike handlebars with gyroscope and accelerometer sensors or video game handlebar controllers. In order to implement the bike-style handlebars, we would need to have gyroscope and accelerometers on the handlebars which would send their data to our microcontroller. The microcontroller would then send this data to Unity, which would need to convert the data into updating the direction of the user in-game. This adds another layer of computation that we decided was unnecessary, since in using a video game handlebar controller all of that computation is done automatically.

4.3 Game With Resistance Control vs Without

Our system has the potential to map in-game difficulty to real-word difficulty, i.e. resistance. In order to do so, we would need to automatically control the resistance of the bike using a motor. If a user started biking uphill in the game environment, then we would need to increase the resistance of the bike, and vice-versa if they were biking downhill. While this would be ideal for our system, there isn't a standard way of controlling resistance on stationary bikes. Some models have the resistance controlled by pushing buttons on the display, while other models have the resistance controlled by pushing up/down or twisting a knob. In the stationary bike that we are be using, it is the latter. As a result, creating a mechanical system, using a motor and sensors, to automatically adjust and measure resistance is a very difficult task, which is why we decided to make it a stretch goal. Since automatically updating resistance corresponding to in-game activity is not critical to the functionality of our system, but rather an added, "coolness-factor" feature, we are making this our stretch goal, if time permits, and decided to not automatically alter the resistance of the bike.

4.4 Hall Effect vs Reed Switches

When deciding how we wanted to implement the RPM calculation we wanted to use a low power solution. This led us directly to using a magnetic sensor to determine the frequency at which the user is pedaling. When using magnetic sensors with PCBs, the two main devices used are reed switches or hall effect switches. The breaking point for hall effect sensors for us was the fact that in order to actually retrieve output of sensor, we'd need to include a signal amplification component to our design. Even though hall effect sensors are cheaper than reed sensors, the cost of the additional components required to truly use the hall effect sensor effectively makes the hall effect sensor the more costly route. Also, because reed sensors have no wearing parts, we can use the sensor at 5V @ 10mA, which is not too power expensive on the PCB.

4.5 Unity vs Unreal Engine 4

To implement the Oculus Rift visuals, we needed to select a game engine that can render virtual reality landscapes. The most popular options are Unity and Unreal Engine 4 since both of these options are officially supported by Oculus. We have planned to go forward with Unity for three reasons: cost, system requirements and ease of use. In terms of cost, Unity is best option. They provide a free version that has Oculus integration built in. Unreal Engine on the other hand only provides a professional version that costs $19 per month plus 5% of revenue after the first $3,000 per game per calendar quarter. The Oculus device already takes up a large portion of our budget so we want to reduce costs as much as possible in all other

aspects of the project. The system requirements to run Unreal Engine 4 exceed the capabilities of the laptops on our team. Unreal Engine 4 requires a 2.5 GHz or faster CPU and a NVIDIA GeForce 470 GTX or AMD Radeon 6870 HD graphics card. The free version of Unity simply requires a CPU with SSE2 instruction support and a graphics card with DX9 (shader model 2.0) capabilities. The website says generally everything made since 2004 should work. While general reviews said that Unreal Engine 4 still works on laptops with lower specifications, albeit slightly slower, Unity fully works so it is the most efficient choice. For ease of use, Unity also seemed to be the better choice especially as we are a small team with first time developers. Unreal Engine 4 is a professional level tool and offers higher quality features but Unity is easier to get started with.

5. System Description and Depiction

Figure 3

5.1 Reed Switch Sensor on Wheel Frame

We used a reed switch to calculate the RPM of the pedals. It functions through two pieces of metal that touch together under the presence of a magnetic field creating a switch. By attaching the switch to the frame holding the spin wheel, and then a magnet to the wheel itself, we are able to determine the speed the wheel is turning from the number of HIGH signals that pass through the switch. Each one corresponds to a single rotation of the wheel. When the switch is open it will have 0V passing through it, and on the rotation when the magnets passes the reed switch, a 5V analog signal will pass through it. This is picked up by the microcontroller via an interrupt handler.

5.2 Handlebars

We are using a video game handlebar controller in order to allow the user to control their movement in the game. The handlebar controller connects with the microcontroller via the SPI communication protocol. This handlebar controller also has buttons on it which allow selection, menu navigating and also buttons for in-game interaction (braking).

5.3 DC Variable-Speed Fan

We are using a small 12V DC variable-speed fan. The microcontroller uses PWM in order to control the fan's speed. By increasing the duty-cycle of the PWM signal, the speed of the motor will increase and the fan will blow harder. By decreasing the duty-cycle of the PWM signal, the speed of the motor will decrease and the fan will not blow as hard. The speed of the fan directly corresponds to the rate at which the user is moving in the game, which corresponds to their averaged RPM over the prior 10 seconds. In addition, if the user hits a speed boost in the game the 12V DC fan blows at full speed.

5.5 Microcontroller (PCB)

The microcontroller has an interrupt handler to detect input from the Reed Switch sensor on the wheel frame. When, the reed switch is open (meaning the wheel is in the middle of a rotation), the microcontroller receives a low signal (0V). When the reed switch is closed (meaning the wheel just completed a rotation), the microcontroller will receive a high signal (5V), which triggers the interrupt handler. We count the number of interrupts every second. It then forwards the number of interrupts, every second, to Processing via the FTDI connection with the computer.

The microcontroller then sends an appropriate PWM signal to the DC variable-speed fan in order to change the speed of the fan in order to correspond with the current RPM status of the user.

The microcontroller receives button and joystick input information from the PS2 handlebar. The communication between the microcontroller and the PS2 handlebar happens over the SPI interface. The PS2 handlebar has six wires that are used: attention, ground, 5V, clock, command (MOSI), and data (MISO).

5.6 Processing:

Processing is a programming language and development environment. It is very useful since it is cross-platform allowing the bike and PCB to work with Oculus Rift enabled games on any operating system. We created an application in Processing that acts as a driver to enable the communication between the microcontroller and Unity. It listens for data being sent (via USB) from the microcontroller regarding RPM, and handlebar orientation which it then relays to Unity by simulating keypresses on the laptop. Most racing games are controlled by arrow keys or W,A,S,D so the Processing driver reads information from the bike and translates it to those key presses. Also, Processing writes data serially (via USB) to the microcontroller. Processing receives data from OSC (Unity) via network ports.

5.7 Unity

Unity is a cross-platform game creation engine. The Unity engine provides a platform to create immersive 2D and 3D interactive content. We used Unity to control all aspects of the game environment, to render the 3D visual content for the Oculus Rift, and to communicate with our microcontroller to send/receive critical information. We created a 3D environment in Unity, program the gameplay, and made scripts that dynamically send/receive data from Processor. To create the virtual reality we followed a tutorial that built the terrain and scenery with a track. We then imported Mario Kart style assets to put a vehicle in the terrain. The physics engine associated with the kart had to be modified to work with the terrain which was made separately. In addition, we will create simple environments from scratch that relate more closely to a setting associate with outdoor biking, such as a trail system in a forest preserve.

Unity sends data to the Processing driver through Open Sound Control (OSC). Open Sound Control is a protocol for communication among computers, sound synthesizers, and other multimedia devices that is optimized for modern networking technology. In Unity, we created functions that utilize OSC to be able to broadcast UDP data to the Processing driver. The communication between OSC and Processor is done via network ports.

5.8 Oculus Rift

The Oculus Rift is a virtual-reality headset which allows the user to be fully immersed in the game/virtual-reality environment. It provides an immersive 3D experience and has a wide field of view, 110 degrees diagonally. The Oculus Rift Development Kit 2, which we are using, uses a low persistence OLED display to successfully eliminates motion blur and jitter, which are the two biggest contributors to motion sickness. The Oculus Rift is connected to a PC via USB and HDMI and directly communicates with the Unity engine.

6. Project management

6.1 Schedule

Date	Oculus/Unity Dev	Sensors	System Communication
2/23	• Set up development studio on computer(s) • Follow tutorial(s) online to learn how to setup necessary Unity environment	• Bring Reed sensor (RPM Sensor) to lab and determine ranges at which it still operates (distance, frequency) • Calibrate handlebar data with video game keys • Calibrate IR sensors for resistance • Calibrate bi-directional motor to turn in either direction	• implement interface for handlebar, RPM, fan, resistance sensor data to talk to unity client • Configure PCB to read input from Reed Sensor • Configure PCB to read input from IR sensor • Configure PCB to send signal to bi-directional motor
3/9	• Implement a responsive Unity environment	• Install PCB to bike • Build RPM sensor from the	• Configure PCB to relay RPM information to the

	integrated with the handlebar and RPM sensor interface • Implement testing for Unity environment using laptop keyboard	reed switches • Wire RPM sensor to PCB	laptop • Configure Unity Processing to send information to PCB • Configure PCB to receive information from laptop
3/23	• Implement the video game into the Unity environment • Integrate handlebar and RPM data into video game/Unity environment	• Test RPM sensor to ensure accuracy • Add resistance IR sensor • Add resistance motor	• Configure PCB to relay IR information to the laptop • Test communication of handlebars to unity
4/6	• Fix bugs with unity environment and handle edge cases	• Add fan to bike	• Configure PCB to send signal to control fan
4/22	• Test and adjust for bugs	• Test and adjust for bugs	• Test and adjust for bugs

6.2 Team Member Responsibilities

Misha:

- Motor-controlled resistance + IR sensor
- Controlling speed of DC variable-speed fan
- Inputting sensor data to microcontroller

Aayush:

- Calculating RPM via Reed Switch
- Inputting sensor data to microcontroller
- Calibrating Unity with microcontroller and handlebar data

Josh:

- Unity, OSC, Processing communication
- Microcontroller communication with Processing via serial USB

Omar:

- Creating the game environment in Unity
- Interfacing Unity with Oculus Rift
- Creating driver in Processing to read and write data to and from microcontroller

All:

- PCB design w/ microcontroller + circuitry for reed switch, fan, motor and ir sensor for resistance

6.3 Budget

- Oculus Rift Dev Kit 2 = ($350)
  https://www.oculus.com/dk2/
- Stationary Bike = ($120)
  Previously purchased off of craigslist
- Reed Switch + Magnet = ($1.95) + ($1.50)
  https://www.sparkfun.com/products/10601

https://www.sparkfun.com/products/8643

- Video Game Motorbike Controller = ($29.95 + $24.62 shipping)
  http://www.ebay.com/itm/NEW-MOTORCYCLE-JETSKII-ATV-HANDLEBAR-CONTROLLER-FOR-PLAYSTATION-2-PS2-/310322617545
- PS2 to USB adaptor for controller = ($5.99 or $3.40)
  http://www.amazon.com/Gen-Dual-PlayStation-Controller-Adapter-2/dp/B000F6BGXY http://www.amazon.com/PS2-Playstation-Controller-Adapter-Converter-2/dp/B000YMQGWU/ref=pd_sim_vg_3?ie=UTF8&refRID=1HHVCHAVDQH2Z59SEX9P
- Microcontroller (MCP2200) = ($2.12)
  http://www.mouser.com/ProductDetail/Microchip-Technology/MCP2200-I-SO/?qs=kI7nsUnms46fN8L9VZlJCA==
- USB connector = ($1.25)
  https://www.sparkfun.com/products/9011
  https://www.sparkfun.com/products/139 ($1.25)
- DC Brushless Fan = ($4.95 or $5.95)
  https://www.sparkfun.com/products/11718
  https://www.sparkfun.com/products/9649
  https://www.sparkfun.com/products/9648
- IR Sensor + Connector = ($13.95 + $1.50)
  https://www.sparkfun.com/products/242
  https://www.sparkfun.com/products/12728
  https://www.sparkfun.com/products/8733
- DC Bi-Directional Motor = ($16.95)
  https://www.sparkfun.com/products/10846
- **Estimated Total = $576.95**

6.4 Risks

Some of the main design risks we have identified thus far are:

6.4.1 Reed Switch Interference by Metal Flywheel

In order to mitigate this risk, we will try to find the best place on the bike to place sensor such that the sensor is secure as well as effective in identifying when the magnet has passed it. If we are unable to find a satisfactory location, we will default to a system that relies on a RFID signal to implement frequency.

6.4.2 Resistance Motor Unable to Turn

Resistance Motor may be unable to turn the knob because of high friction with the way the resistance is designed on the spin bike. This may result in us not being able to control the resistance and have that be a setting the user manually has to adjust. As this is a stretch goal, if

time permits we will try to use alternative methods to implement the turning of the difficulty knob, else we will try to focus on the other parts of the project.

6.4.3 Latency of Application

Users will expect a smooth experience while using our system. In order to reduce the risk of having an unplayable game because our system takes too long to respond to events, we will try to ensure that the latency between the microcontroller and the laptop is always under 30ms

6.4.4 Creating an Immersive Environment in Unity

None of the team members have experience developing environments in Unity. It may turn out that we cannot port existing video games to be played with the Oculus Rift. If this happens we will stick to creating landscapes that are designed in Unity and made specifically for the Oculus Rift.

6.4.5 Reed Switch Unable to Activate

We face the risk of being unable to locate a spot on the bike which accurately depicts a rotation and withstands the speed at which the user may be pedaling. The best spots seem to be the pedals or the disc that rotates from the pedals. If the magnet is too far from the reed switch or it falls off due to the user pedalling too fast, the reed switch may not be activated. This will prevent the bike from communicating the RPM to the virtual reality, so it is critical to have this working. To mitigate this risk, we will purchase a more powerful magnet or we will try to find a reed switch that works better in stress conditions. Furthermore, we can test on both the pedals and the disc to figure out which is the optimal location.

8. Conclusions

i. We learned a lot during the course of bringing RideThru to life. What we learned falls into two categories, relating to technical and project-management. On the technical side, none of our team had any real experience in making a PCB or has taken very advanced classes relating to circuit design. As a result, designing and then bringing up our PCB was very challenging for us, but also gave all of us a chance to learn a great deal about it. The PCB was the most critical piece of our project as it was the link between the physical sensors on the bike and the game experience in Unity, so we spent a lot of time in ensuring that our design was exactly how we wanted it to be. We also learned a great deal about how to manage the project across 4 different people. We definitely underestimated how hard it would be to work cohesively as a group. Initially our plan was to split up the work on the project, which made a lot of sense. Our project was really broken up into three parts: the physical sensors on the bike, the PCB design and bringing up, and the work on the game side in Unity. We split up the work across the four of us and it worked pretty well, but we ran into some issues once we tried to link everything together. In hindsight, since everyone had a small piece to work on we ran into issues since we hadn't really looked into the specifics of actually linking everything together. In addition, we definitely underestimated the time that we would spend debugging.

ii. Much of what we would do differently if we were given the opportunity to start over from scratch relates to what we learned working on this project. We definitely wouldn't split up the work as rigidly as we did, since this left people with expertise on their piece of the project, but did not know a ton about other pieces. As a result, we ran into some unforeseen issues once we tried to link everything up, mostly on the PCB side of things. This leads to the second change we would make. We designed our PCB with a lot of goals in mind, as it had to handle RPM information, fan PWM, handlebar inputs, force sensor input, and to control a motor. Accommodating all of these design goals was very challenging on the PCB. Since controlling the motor and the force sensor input were always stretch goals, we would not attempt to implement these next time. Adding these two the PCB only increased the difficulty and resulted in a few decisions on the PCB that didn't work, which we had to hack around.

iii. I think our first big step would be to define what we want RideThru to be as a product. Do we want it to be something that the user buys and attaches easily to his or her stationary bike? Or, do we want it to be built into a custom stationary bike that the user sells? We believe that we would shoot for the former. From this, our next course of action would be to figure out how we would design the system to easily interface with a stationary bike. In addition, we would need to work on creating a very well packaged system for the sensors and PCB. Also, we would like to test out our current version on other games, as we have designed it to be able to easy interface with any game that accepts keyboard inputs.

8. Related Work (Competition)

8.1 PaperDude VR

Globacore built a game using Unity, the Oculus Rift, a Kinect, and a KickR sensor to recreate the game paperboy. As the user pedals, their character in the game pedals and they can toss paper in the game by making a pretend throw motion in real-life, as interpreted by the Kinect. Our system is very similar in terms of overall concept to the PaperDude VR, but we have added multiple features that enhance the overall experience of virtual-reality biking. Our system allows the user to use handlebars to control the motion of their character within the virtual-reality environment, which is the big limitation of PaperDude VR. In PaperDude VR, you can only move straight ahead, so the overall experience is limited. By allowing the user to control the motion of their character, we open up the opportunity to create richer game environments for the user to explore and more closely mimic outdoor biking. Also, we have an automatically controlled fan, which mimics the acceleration that a user would feel outside. In addition, our system is compatible with user-created environments in Unity, so the user can ride a different virtual-reality experience potentially each time they bike.

8.2 Expresso

Expresso is a virtual system for stationary bikes, which allows the user to "ride" on real-world trails. They can see their character on a screen in front of them. They can control handlebars to move their character, and the resistance of the bike changes as they go uphill/downhill. The faster they pedal, the faster their character moves. This system has a lot of the functionality that we would ideally like to implement, apart from the automatically controlled fan. However, our system utilizes the Oculus Rift in order to create a fully immersive 3D

experience. At the end of the day, when using a system like Espresso you still constantly realize that you are on a stationary bike and are working out. Ideally, our system will allow the user to feel like they are biking in a different environment and will actually make stationary biking fun and enjoyable.

9. References

Hall Sensor vs Reed Sensor:
http://www.meder.com/sensor-vs-hall-effect.html
Connecting Oculus Rift to PC:
http://www.tomsguide.com/faq/id-2355028/connecting-oculus-rift.html
Arduino Bike Speedometer:
http://www.instructables.com/id/Arduino-Bike-Speedometer/?ALLSTEPS
Arduino + Fan + PWM:
http://www.electroschematics.com/9009/led-brightness-fan-speed-arduino/
http://www.electroschematics.com/9540/arduino-fan-speed-controlled-temperature/
Stationary Bike + RPM + Reed Switch:
http://urbanhonking.com/ideasfordozens/2009/11/21/stationary_bike_speedometer_wi/
PaperDude VR:
http://www.globacore.com/paperdude-vr/
Expresso:
http://expresso.com/Learn/Experience
Using Microchip MCP2200 for USB Serial Communication:
http://pwc.theclarkwebsite.com/mcp2200.php
Unity System Requirements
http://unity3d.com/unity/system-requirements
Unreal Engine 4 Requirements
https://www.unrealengine.com/faq