VR Hand Simulator

ECE 445 Design Document

Team 67 - Alex Brannick and Daryl Drake (brannic2, dadrake3)

TA: Dongwei Shi

1 Introduction

1.1 Objective

In the past decade, we have seen a new type of technology reach customers worldwide, and that technology is Virtual Reality. After the acquisition of Oculus VR for $2 billion by Facebook [1] larger companies began focusing their attention on developing their own VR technology. Shortly after companies such as HTC and Sony began making their own VR technology, bringing their own VR headsets to the market.[2] In particular, these VR headsets have been largely involved in the game industry. By making the headsets available to the typical consumer, it caught the attention of game companies and this brought out new games for these platforms. While the quantity and quality of these VR games has been increasing, the platforms that they are played on have stayed relatively the same. These types of games strive to create a more immersive experience for the player, and VR headsets have lacked to keep up with this increased creativity. In an article posted by the LA Times, there has been a significant decline in the amount of investment towards the VR industry recently. In 2018 alone, funding for VR startups fell 46% from 2017 [3]. Many of the investors' concerns were focused on the fact that the VR market had not reached a large enough mass of consumers.

Our goal is to help revive the VR industry by creating a more immersive environment for all VR headsets. We plan on doing this by creating an attachment for all headsets called the "VR Hand Simulator". With this attachment, it will allow for users to perform complex interactions within their virtual environment using their hands as the controllers. Haptic feedback bracelets will also be used to make the experience more immersive by providing gamers with a sense of touch within Virtual Reality. Our device will be a plug in for the Unity game engine, allowing any indie developer or large company to create games for these headsets using our device.

1.2 Background

In 2018, the VR industry saw a large decline in investment and a decline in adoption of VR platforms. While the quantity and quality of VR games has risen, the demand for these games has not reached as high of a level as was expected. With our product, we would combat this decline in sales by making virtual reality more immersive than ever before. Adding hand simulation and haptic feedback using our product provides a new avenue for developers to make games. This could be anything from playing a virtual piano to building a castle out of blocks.

Furthermore, by making our product into a plug-in for Unity, we give small indie developers and large companies a chance to develop their game to use our product.

1.3 High-Level Requirements

- The device must be able to accurately capture and simulate the movement of our hands in Virtual Reality at speeds of around 10-15 frames per second.
- The device needs to have a plug-in for the Unity game engine, where nearly 45% of all games are developed. [4]
- The device has to be easily useable with any VR headset, specifically the big 3 gaming VR headsets. (Oculus Rift, HTC Vive, Sony Playstation VR) This requires it to fit within a 3.9 in x 6.3 in rectangle space.

2 Design

The VR hand simulator will require a few main components to work properly. The device will consist of a headset and two haptic feedback bracelets. Two RGB cameras on the headset will capture video of the users hands during gameplay. All of the image data captured from the cameras will then be fed to a raspberry pi compute module. After the raspberry pi compute module has received the frames, it quickly sends the data to the Neural Net compute stick that is our tensorflow accelerator. Once the frame data has pinpointed the location of our hands and identified their joints, the data is passed back to the raspberry pi. This data is finally ready to be used and is sent over bluetooth or a serial connection to the game. We will create a Unity plug-in to interpret this data so that developers can then make VR games in Unity that use our product. They will be able to populate the users hands within the game as well as send haptic feedback to the headset, where it will transmit these codes to the bracelets over an IR connection. Finally, we will need some sort of power supply in the form of a battery or cable to power our headset and a battery for powering each bracelet.

Block Design

Headset Block Diagram

The chart below shows the flow of data for our design. As video data is received from our cameras, we will pass this data to the Raspberry Pi to be sent to the compute stick. From here, the results go back to the Raspberry Pi to be sent over a serial connection to our game. Any IR codes sent from the game travel through the serial connection as well to the Raspberry Pi and are sent by the IR transmitter. The power supply will be used to power the various pins on the compute module

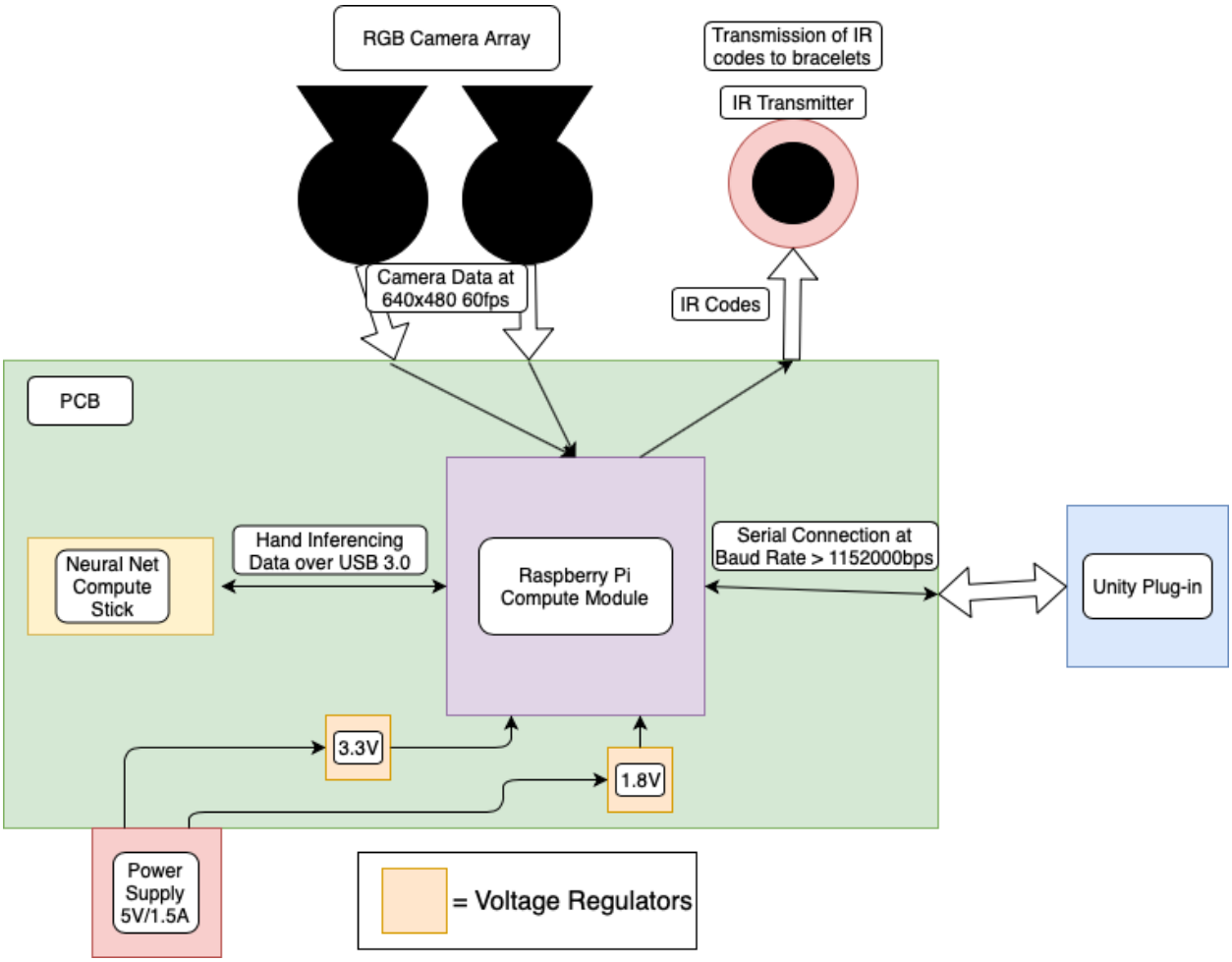

Figure 1. This model shows at a high level where the data is flowing from as it is taken in from the cameras and ultimately passed to the Unity Plug-in.

Bracelet Block Diagram

Each bracelet will follow the block diagram below. The IR Receiver will take in IR codes from the headset and pass this data into the Adafruit Trinket M0. Here the signals will be decoded and they will determine how much to vibrate the motor or change the led. One led will be connected to a pwm pin for basic feedback from the bracelet while another pwm pin will be used to vary the voltage being supplied to the motor from the battery.

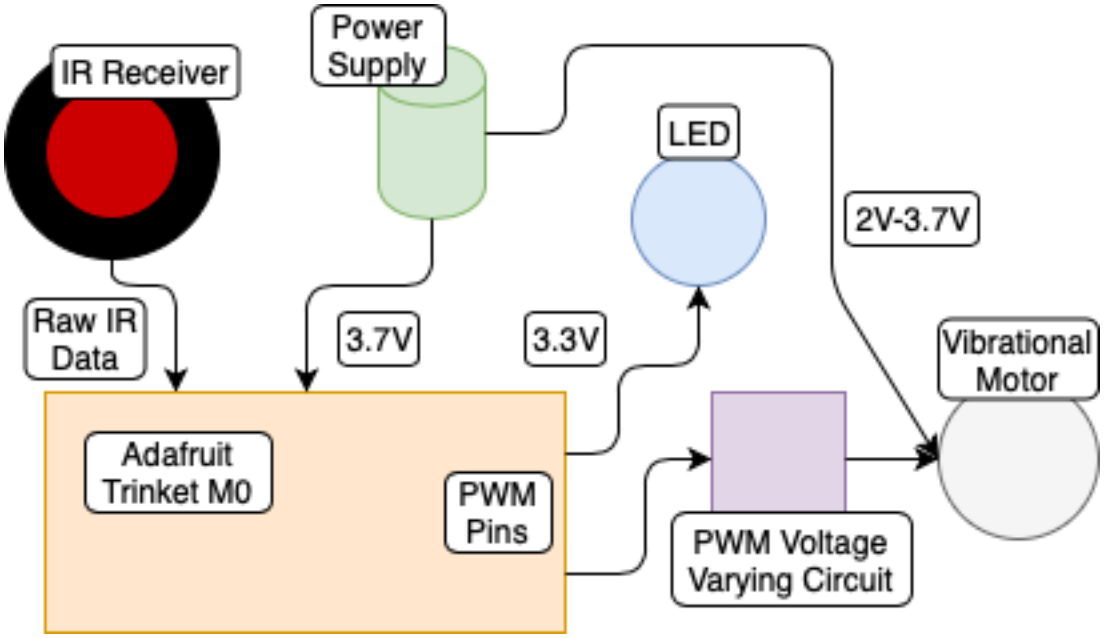

Figure 2. This figure shows the diagram of our bracelet. The IR data from the receiver will be passed to the Adafruit micro controller to change vibrational speed and give feedback from the led.

Physical Design

The image below is a mock up of what we plan our final design to look like. It is pictured on top of a generic VR headset, and would be mounted above the visor for a user. The PCB for the bracelet would be mounted to some type of silicon bracelet with a pouch for the battery.

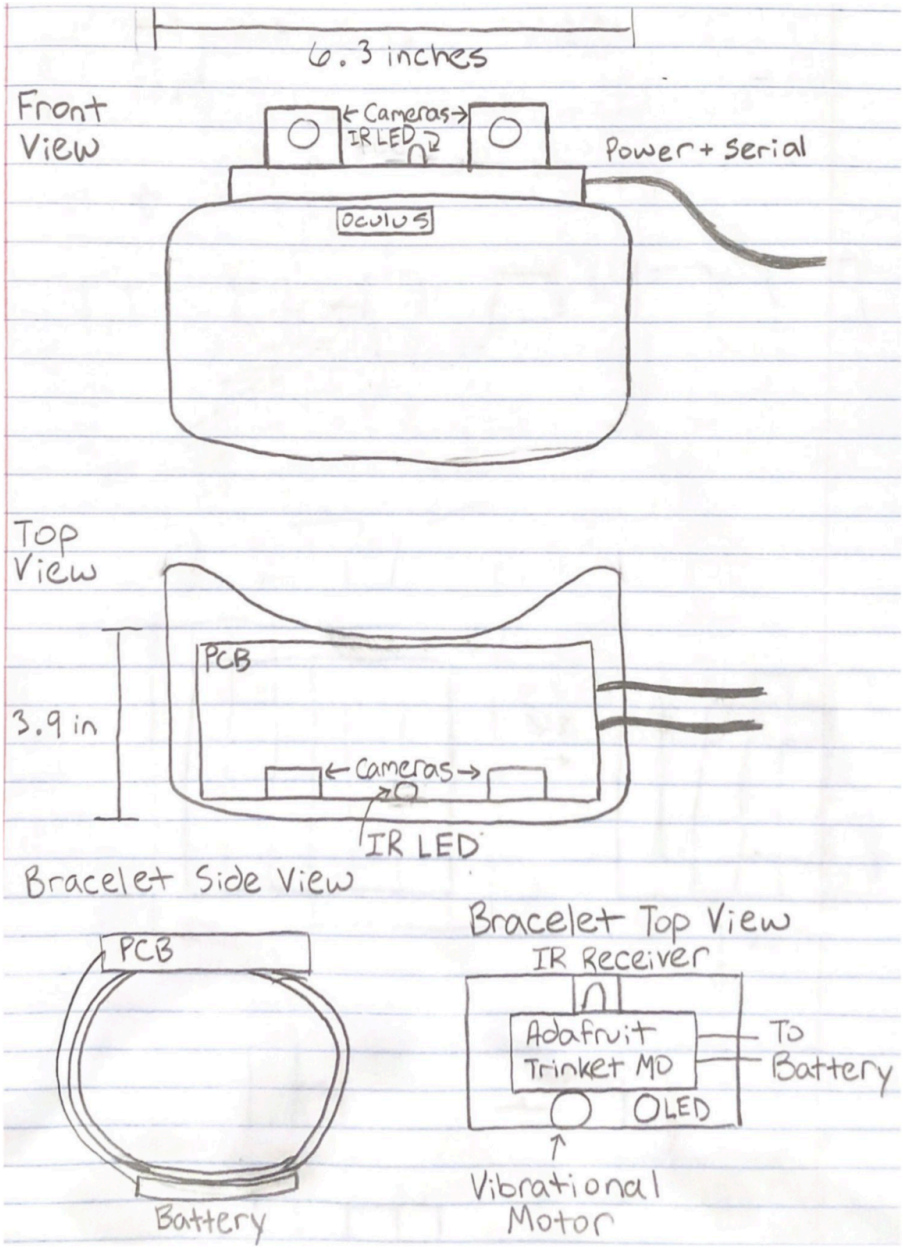

Figure 3. Above is the view of the Oculus Rift with our device. We plan on attaching our device to the top of the visor, and it will need to fit within a dimension of about 6.3x3.9 inches. A simple mock up of the bracelet is shown as well.

2.1 Headset Modules

RGB Camera Array

Our design requires that we receive video data from two cameras attached to the top of a VR headset. The camera we have chosen for our device is the Raspberry Pi camera board, which uses the CSI-3 camera ports on the Raspberry Pi compute module to transfer video data at 720p/60 or even 640x480p60.

Requirements	Verification
1. Be able to capture frames at speeds of 720p/60 and 640x480p60/90 2. Be able to handle two cameras at same speeds with no loss in quality	1. Connect a single camera to the Raspberry Pi and write a simple python script to check that it is properly receiving data at 60fps and 90fps for 640p and 720p respectively. Next, display the output from the camera on the monitor to check for latency issues between when the camera data is received and when it was captured. 2. Now hook up the second camera to the second CSI-3 port and run all tests from part 1 on this camera individually. After it has been verified to work, run all tests from part 1 with both cameras simultaneously. Verify that there is no loss in production.

CSI-3 Protocol and Port Layout

CSI, or camera serial interface, is a type of bidirectional protocol used for video and image transmission. [8] It can achieve bit rates of up to 5/5.8 Gbps depending on the clock speed, but when used with the Raspberry Pi Compute Module it can reach a maximum of 1 Gbps on both camera ports. Figure 4 (pictured below) shows the pin layouts on the Raspberry Pi that are used for the two camera interfaces. The device is built with a 2-lane and 4-lane CSI camera interface. The data buses are located from pins 135-162. Outside of the data pins, CSI-3 protocol requires an I2C connection between the camera and the compute module. In Figure 5 (below), there are some basic schematics for the CSI ports on the development board for the Raspberry Pi. These illustrate where the I2C pins are located and how the development board connects the compute module to the camera ports.

GND	133		134	GND
CAM1_DP3	135		136	CAM0_DP0
CAM1_DN3	137		138	CAM0_DN0
GND	139		140	GND
CAM1_DP2	141		142	CAM0_CP
CAM1_DN2	143		144	CAM0_CN
GND	145		146	GND
CAM1_CP	147		148	CAM0_DP1
CAM1_CN	149		150	CAM0_DN1
GND	151		152	GND
CAM1_DP1	153		154	NC
CAM1_DN1	155		156	NC
GND	157		158	NC
CAM1_DP0	159		160	NC
CAM1_DN0	161		162	NC
GND	163		164	GND

Figure 4. (Above) The pin layouts for the 4-lane and 2-lane CSI camera interfaces on the Raspberry Pi compute module. These pins are connected through the development board to CSI sockets, but will eventually be directly inserted into our PCB.

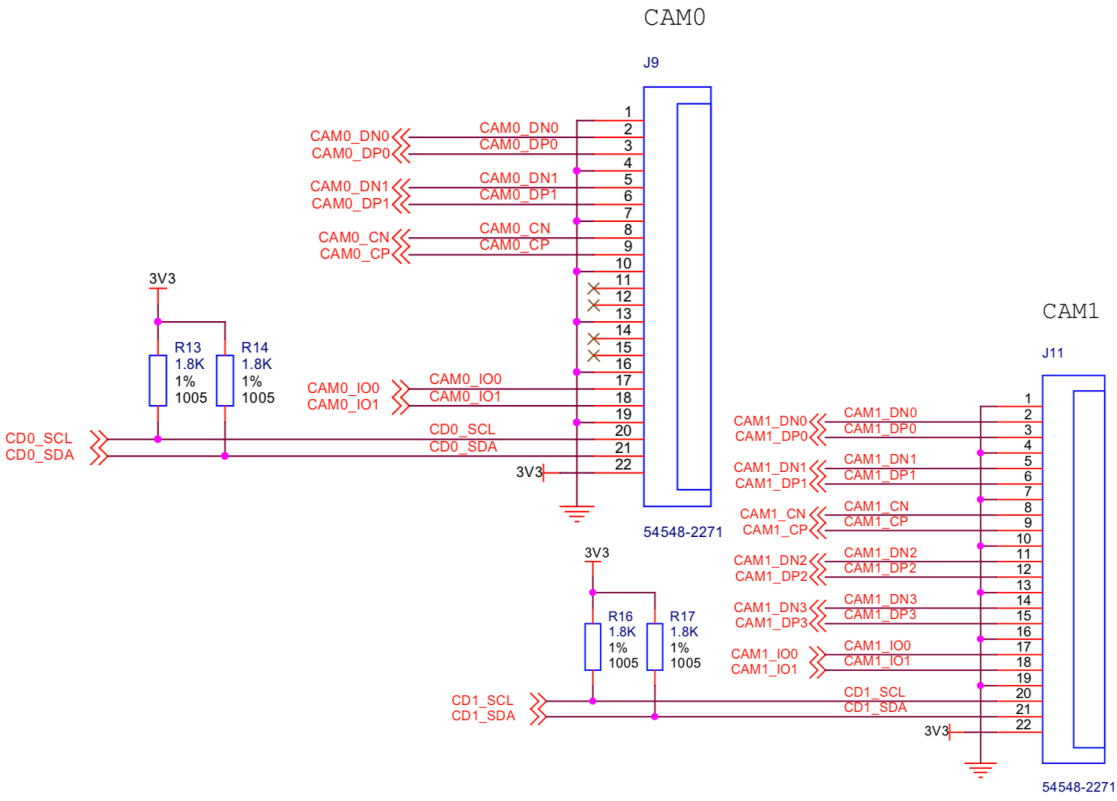

Figure 5. (Above) These are some of the schematics for the Raspberry Pi development board. They show the pins on the compute module that are directly connected to the CSI camera ports on the board.

Raspberry Pi Compute Module

The Raspberry Pi compute module is the main processor of our headset. It will be responsible for receiving the image data from the headset and passing it to the compute stick for inferencing. Besides passing this data around, it needs to prepare the data to send. This formatted data must be sent over a serial connection to then be interpreted and used by the Unity plugin. Data may also be sent back in the form of IR codes from the plug in, where it will then send these codes to the IR transmitter for the bracelets. This portion of our headset will be the most complex, and it will require a PCB to be designed that can handle the routing of data between the Raspberry Pi and its peripherals. We will go into the design of our PCB in more detail below. At the start of the project, we will focus on designing the headset on the development board before moving to the PCB.

Requirements	Verification
1. Handle image data at 720p60 and 640x480p60/90 from two different cameras 2. Send IR codes over the IR transmitter to the bracelets 3. Send and received data from the Unity plug in over a bluetooth or serial connection	1. This part requires the same unit testing from the R&V of the camera array 2. Individually send each IR code through the Raspberry Pi to the transmitter and verify that each command is received by the bracelets correctly 3. Verify that all IR codes can be sent from the plug in to the Pi and compare the hand data from the Pi with the data that has been received by the plug in

Raspberry Pi Compute Module Pin Layout

The pin layout for the Raspberry Pi compute module is listed on their website, and it details the 200 pins of the device before it is connected to the development board. We used this in order to begin working on a custom PCB for the final iteration of our design.

We will need approximately 39 pins on the Raspberry Pi compute module to connect it to each module in our design. The two CSI-3 camera ports use 24 of the pins (135-163) to connect to the two RGB Cameras. Another 4 pins are needed for the two voltage inputs on the Raspberry Pi, and those are in the pin ranges of 183-196. In order to communicate with the Neural compute stick, we will also need 4 pins for a female USB2.0 Type A port. For the IR LED transmitter, another 3 pins will be used. Finally, the data needs to be transferred over a serial connection so the game can interpret this data. This will require another USB2.0 Type A or Micro USB port,

which would require 4 pins for either configuration. Simple pin layouts for the USB ports and and the IR transmitter can be found below.

CM1	CM3-Lite	CM3	PIN	PIN	CM3	CM3-Lite	CM1
GND			1	2	EMMC_DISABLE_N		
GPIO0			3	4	NC	SDX_VDD	NC
GPIO1			5	6	NC	SDX_VDD	NC
GND			7	8	GND		NC
GPIO2			9	10	NC	SDX_CLK	NC
GPIO3			11	12	NC	SDX_CMD	NC
GND			13	14	GND		NC
GPIO4			15	16	NC	SDX_D0	NC
GPIO5			17	18	NC	SDX_D1	NC
GND			19	20	GND		NC
GPIO6			21	22	NC	SDX_D2	NC
GPIO7			23	24	NC	SDX_D3	NC
GND			25	26	GND		
GPIO8			27	28	GPIO28		
GPIO9			29	30	GPIO29		
GND			31	32	GND		
GPIO10			33	34	GPIO30		
GPIO11			35	36	GPIO31		
GND			37	38	GND		
GPIO0-27_VDD			39	40	GPIO0-27_VDD		
			KEY				
GPIO28-45_VDD			41	42	GPIO28-45_VDD		
GND			43	44	GND		
GPIO12			45	46	GPIO32		
GPIO13			47	48	GPIO33		
GND			49	50	GND		
GPIO14			51	52	GPIO34		
GPIO15			53	54	GPIO35		
GND			55	56	GND		
GPIO16			57	58	GPIO36		
GPIO17			59	60	GPIO37		
GND			61	62	GND		
GPIO18			63	64	GPIO38		
GPIO19			65	66	GPIO39		
GND			67	68	GND		
GPIO20			69	70	GPIO40		
GPIO21			71	72	GPIO41		
GND			73	74	GND		
GPIO22			75	76	GPIO42		
GPIO23			77	78	GPIO43		
GND			79	80	GND		
GPIO24			81	82	GPIO44		
GPIO25			83	84	GPIO45		
GND			85	86	GND		
GPIO26			87	88	HDMI_HPD_N_1V8		GPIO46_1V8
GPIO27			89	90	EMMC_EN_N_1V8		GPIO47_1V8
GND			91	92	GND		
DSI0_DN1			93	94	DSI1_DP0		
DSI0_DP1			95	96	DSI1_DN0		
GND			97	98	GND		
DSI0_DN0			99	100	DSI1_CP		
DSI0_DP0			101	102	DSI1_CN		
GND			103	104	GND		
DSI0_CN			105	106	DSI1_DP3		
DSI0_CP			107	108	DSI1_DN3		
GND			109	110	GND		
HDMI_CLK_N			111	112	DSI1_DP2		
HDMI_CLK_P			113	114	DSI1_DN2		
GND			115	116	GND		
HDMI_D0_N			117	118	DSI1_DP1		
HDMI_D0_P			119	120	DSI1_DN1		
GND			121	122	GND		
HDMI_D1_N			123	124	NC		
HDMI_D1_P			125	126	NC		
GND			127	128	NC		
HDMI_D2_N			129	130	NC		
HDMI_D2_P			131	132	NC		
GND			133	134	GND		
CAM1_DP3			135	136	CAM0_DP0		
CAM1_DN3			137	138	CAM0_DN0		
GND			139	140	GND		
CAM1_DP2			141	142	CAM0_CP		
CAM1_DN2			143	144	CAM0_CN		
GND			145	146	GND		
CAM1_CP			147	148	CAM0_DP1		
CAM1_CN			149	150	CAM0_DN1		
GND			151	152	GND		
CAM1_DP1			153	154	NC		
CAM1_DN1			155	156	NC		
GND			157	158	NC		
CAM1_DP0			159	160	NC		
CAM1_DN0			161	162	NC		
GND			163	164	GND		
USB_DP			165	166	TVDAC		
USB_DM			167	168	USB_OTGID		
GND			169	170	GND		
HDMI_CEC			171	172	VC_TRST_N		
HDMI_SDA			173	174	VC_TDI		
HDMI_SCL			175	176	VC_TMS		
RUN			177	178	VC_TDO		
VDD_CORE (DO NOT CONNECT)			179	180	VC_TCK		
GND			181	182	GND		
1V8			183	184	1V8		
1V8			185	186	1V8		
GND			187	188	GND		
VDAC			189	190	VDAC		
3V3			191	192	3V3		
3V3			193	194	3V3		
GND			195	196	GND		
VBAT			197	198	VBAT		
VBAT			199	200	VBAT		

Figure 5. (Above) These are the pin assignments for the Raspberry Pi compute module. All pins described in this part of the Design Document can be found on this table.

USB ports

In any type of USB2.0 port, it requires at least 4 pins. These include a Voltage pin, Ground pin, a Data-in pin, and a Data-out pin. The Voltage pin requires 5v, so we can attach these pins to any output voltage supply from the board. As for the data pins, these can be connected to any one of the GPIO pins from the board. This is true for all ports that we will need on the board.

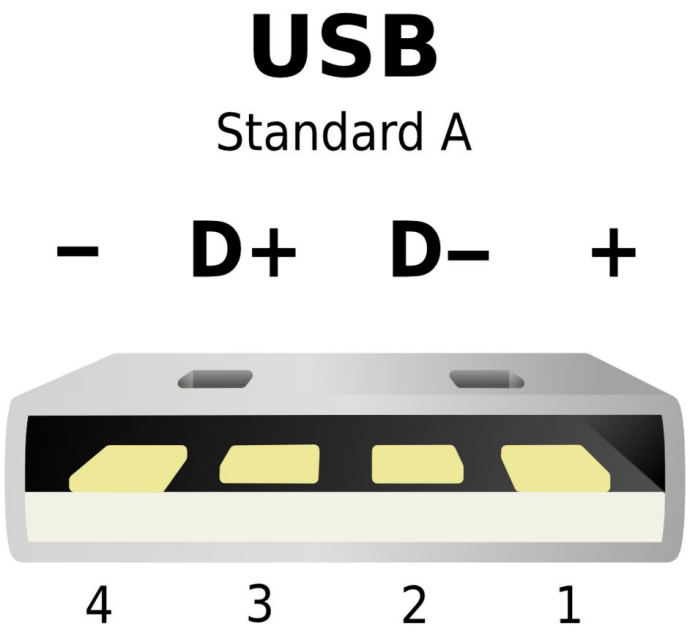

Figure 6. This diagram represents the pin layout for a simple USB2.0 type A port. Four pins are needed for use with each port.

Neural Net Compute Stick

The Neural Net compute stick is a small piece of hardware used for accelerating neural networks locally. It connects to your device over a USB3.0 type A port and works well with the Raspberry Pi. In our design, it will help speed up the inferencing done on the video frame data we receive form the cameras so we are able to inference more frames per second.

Requirements	Verification
1. The compute stick can transfer data between the Pi and itself without corruption 2. Reduces inferencing time down to <3ms.	1. Verify that the compute stick is properly connected to the Raspberry Pi by checking all wires have been properly attached and soldered for the USB port. 2. Use a timer to measure the time from when the frame data is sent to the compute stick and when the inferencing is completed.

Power Supply

In order to power our headset, we will need a power supply to support our FPGA board and Raspberry Pi compute module. The power supply will be hooked up through a single input on the PCB that routes power to each individual module to reduce the number of cables in our design. We plan on using a micro-usb cable plugged into a battery or wall outlet to power the device. In a later iteration of our design, we intend on using a Lithium-Ion battery mounted on the headset to provide a cable free device. This was ruled out of the first design however for the added weight and complexity to the headset.

The Raspberry Pi compute module requires 3.3V and 1.8V at a minimum of 250mA to work. In order to supply our device with the correct voltage and current, we will use a simple usb-cable that can provide a 5V/1A power supply to our device. This will get us to the required power and voltage for the Pi, with voltage regulators used to bring the voltage supply to the right target value for each input. In terms of current, USB 2.0 standards can handle up to 900mA while USB 3.0 is able to reach current of up to 1.5A, which will provide more than enough power for our Pi. One issue that we may encounter is from a large surge of power blowing out one of the pins on the chips. We may need to add a diode to ground connection along our wires to the voltage pins to prevent any surge from effecting the chips.

Requirements	Verification
1. All modules on the PCB are adequately powered.	1. Use a voltmeter to verify that 3.3V and 1.8V is being supplied to our Raspberry Pi compute module from the PCB.

PCB Design

In order for our device to have a low profile, we decided to forgo using any type of development board in our final design for a more slim look. The PCB that we will be designing will be used as a median between the Raspberry Pi and all of the other modules by creating connections from the cameras, IR transmitter, and compute stick to our Pi. In our first iteration of the PCB, we will need to account for connections between the IR transmitter, the Raspberry Pi, the RGB cameras, the USB port for the compute stick, the serial cable, and a power supply. We will need to look over the RGB camera documentation as well as the CSI-3 pin layout and protocol to make sure that we properly connect the two camera array to our Raspberry Pi compute module. Our PCB will also be needed to create female USB ports for connection with the Neural compute stick, Unity Plug-in, and power supply. The Raspberry Pi need two types of input voltages, so this will lead to a total of two connections with two voltage regulators. After reading into powering these devices, we may also need to use diodes to help prevent any power surges from destroying the chip.

Requirements	Verification
1. PCB must be small and needs to fit within the dimensions of all VR headset visors 2. Adequately power the entire headset from a single input	1. Verify that the final PCB design is within the dimensions of 160mm x 100mm (about 6.3in x3.9in), which keeps the PCB within the limits of all VR headset dimensions 2. After verifying proper voltage is being applied to each device, use a digital multimeter to check the input voltage at each part of our circuit. (Ex: 3.3V needed across the power supply pins for the Raspberry Pi)

IR Transmitter

In order to send the IR codes to our bracelets, we will need a transmitter on the headset. After the Raspberry Pi has gotten the codes from the plug in, it will send them over this transmitter to the bracelets. We will need an IR led, a transistor, and a resistor to be able to connect the IR led to the Raspberry Pi. This will allow us to flash the LED with the pwm output of a GPIO pin on the Pi in order to send our signal.

Requirements	Verification
1. Eight different codes must be able to be transmitter from the Raspberry Pi to the bracelets	1. Properly connect the IR LED to the Raspberry Pi. Next run a simple program to test that each individual code can be sent and is received by either bracelet

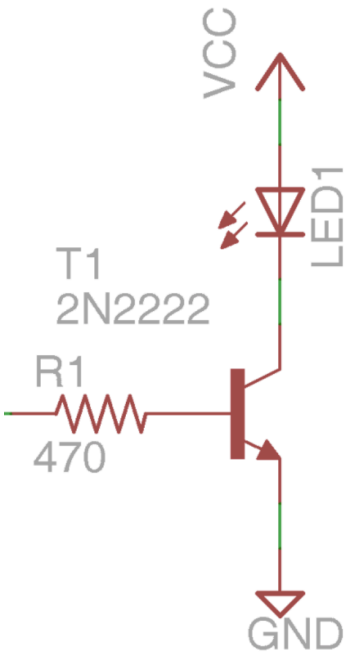

Figure 7. The IR LED will be powered directly from a rail on the compute module. The voltage across it will be varied however by a pwm output from a GPIO pin on the Raspberry Pi compute module.

Unity Game Engine

The final part of our project will require us to interface our headset with the Unity Game Engine. We will need to transmit the data of our hand location and positioning to the game, where this data will allow a user to control the game. In order to test our device we will also need to create some type of demo, or obstacle course to check how we handle basic hand movements. By completing these steps, it will allow developers to use our plug-in for their VR games on any platform.

Requirements	Verification
1. Properly pass data to Unity plugin 2. Recreate hand movements within Unity game engine demo environment	1. Make sure that we are passing the formatted data from the headset at a baud rate quick enough for the data (most likely >115200 bps). Verify that the data is accurate by comparing the data sent from the Raspberry Pi to what we are receiving in the Unity game engine. 2. Create a Unity game demo with basic hand movements. To verify the joint segmentation is working, check that all 10 fingers can move independently as well as each of their joints. Now check that the depth works by moving your hands towards various targets within the game at 3 depth levels at varying speeds. Finally, check the latency of the movement of the hands by setting a timer in the game. Have a test where the game prompts you to move your hands and times the length at which you take to respond. We are expected responsiveness to be < 0.1 seconds.

Data Transfer Protocol

In order to send the formatted hand data from our headset, we need to devise some type of data protocol for Unity to interact with our device. Using a serial connection, we would be able to take advantage of a python library named PySerial to handle the passing of data. This would allow us to achieve baud rates of up to 115200 bps and guarantees we will get our data across without loss. Packets sent to Unity will contain positional hand data that is marked with a timestamp and id to provide more information for the developer. Since we are dealing with a compute module Raspberry Pi and not a development board, we will need to hook up a serial bus from the device in order to connect. The figure below shows the pin layout where a serial connection can be established.

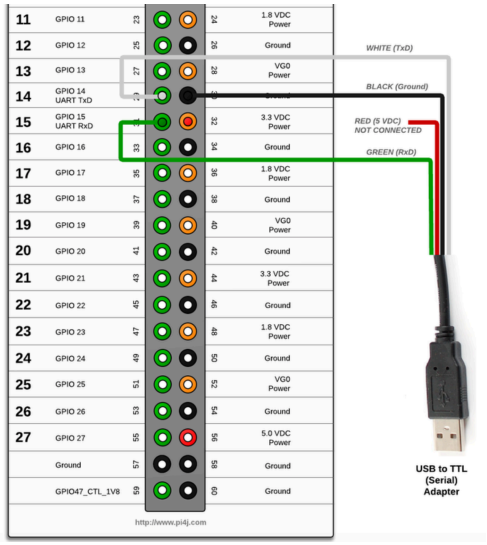

Figure 8. The image shows the pins required to create a TTL to USB adapter. [9]

2.2 Bracelet Design

IR Receiver

The IR Receiver is needed to receive commands from the device. We chose to use the TSOP38238 supplied on Adafruit, as it was a small size and can handle most TV level remote commands. To use it, we would need to connect pin 2 to ground, pin 3 to the voltage supply (2.7-5.5V), and pin 1 to a gpio on the microcontroller.

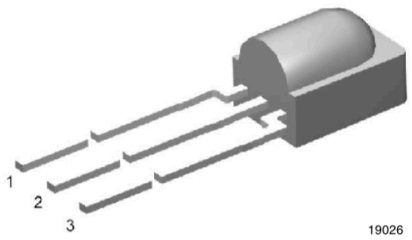

Figure 9. TSOP32238 IR Receiver Image

Requirements	Verification
1. Receiver is able to read all IR codes from the headset and relay them to the microcontroller	1. This can be checked by sending all codes from the headset to the bracelet and verifying the codes are correct on the microcontroller.

Adafruit Trinket M0

The Adafruit Trinket M0 is a quarter-sized microcontroller that we will be using for our bracelet design. This device will be used for decoding the codes from the IR receiver to control the LED and vibrational motor. It has five GPIO pads, and we will use pads 0 and 3 for pwm output to the LED and motor. To receive analog input from the IR receiver, we will use pad 2.

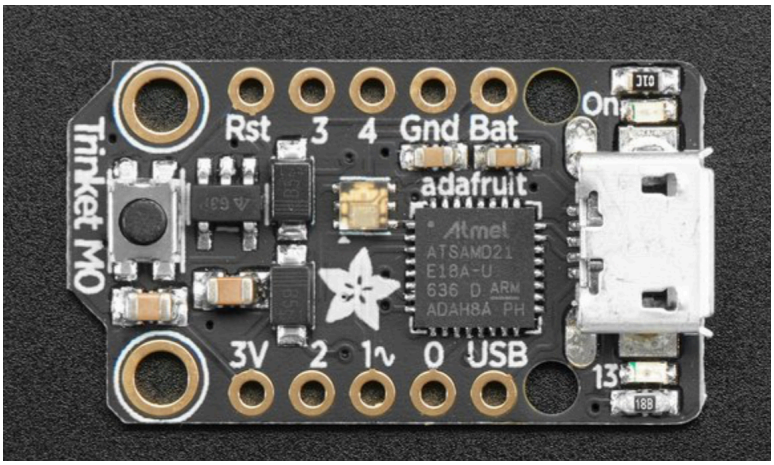

Figure 10. This is the image of the Adafruit Trinket M0.

Requirements	Verification
1. Correctly reads IR codes from IR receiver 2. Can control the motor at three different speeds 3. Can control the LED to display codes	1. Verify all four codes are received and correctly read from the headset 2. Check that the 2V, 3V, and 3.7V outputs can be reached for the motor from the pwm output pin 3. Make sure that the pwm output for the led can flash the LED whenever a code is read from the receiver

Basic Schematic

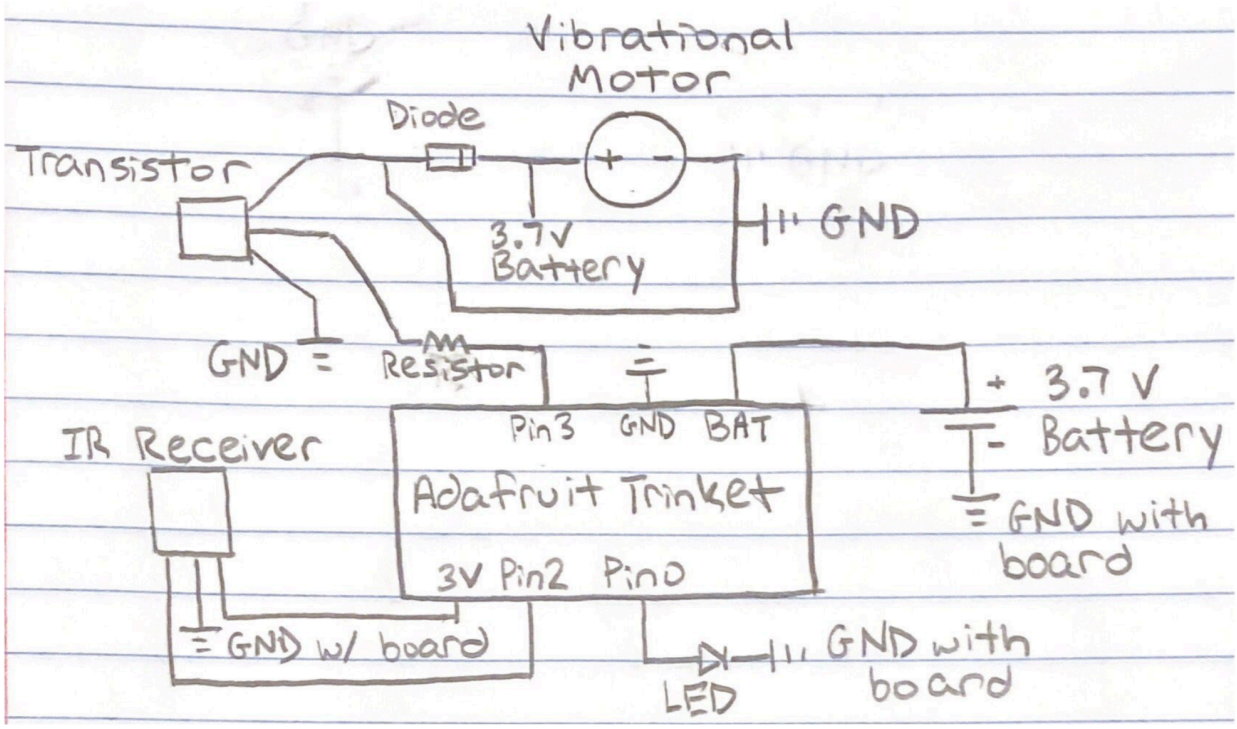

Figure 11. The basic circuit schematic for the bracelet. The PWM circuit is shown above for the vibrational motor along with all of the pin connections between the board and its peripherals.

LED

The LED we will be using is going to be a simple diode for debugging and communicating purposes. It will be connected directly to one of the Adafruit Trinket M0's pwm pads. Our plan is to have it flash for various tasks that the microcontroller maybe performing. We would flash it once at startup, and flash it twice every time a new code has been received.

Vibrational Motor/PWM Circuit

The motor we have chosen to use for our design is the vibrating mini motor disc from adafruit. It is designed to work with input voltages anywhere from 2V-5V, with larger vibrations at higher voltages. The device needs to be connected to a power supply and ground, so we will have our voltage supply come from the battery and use a pwm pin to vary the voltage. The pwm pin will be fed into a transistor that can reduce the voltage.

Requirements	Verification
1. Verify that the motor can receive 2V, 3V, and 3.7V from the microcontroller.	1. This verification step can be done using a voltmeter and some simple code from the microcontroller to test each voltage level

Power Supply

We will have a power supply come from a single batter pack that will be tucked into our bracelet. This should be sufficient in supplying power to the bracelet as our circuit will not draw much current. The battery we plan on using will be a LIPO that supplies 3.7V at 350mAh. Even if the vibrational motor were to vibrate at 5V, the battery would be able to sustain this vibration for an hour (5V draws at 100mA for the motor). Without constant vibrating, the bracelet should be able to have a battery life of a few hours. The battery can be connected to the BAT and GND pins on the Adafruit Trinket M0 to power the device.

Requirements	Verification
1. Power the microcontroller for at least 2 or more hours	1. Allow the bracelet to run while receiving haptic feedback codes somewhat frequently. Time the amount of time it takes before the battery on the cell dies.

Tolerance Analysis

For our tolerance analysis, we will focus on the power consumption of the bracelet. Batteries are measured in Amp hours, which represents how much current the battery could provide in a single hour before being completely drained. The battery we are using is a LIPO that supplies 3.7V at 350mAh. In order for our bracelets to meet our requirements, we want them to be able to run for at least 2 hours before they need to be recharged.

In order to properly determine the battery life of our bracelet, we will need to take into consideration the total amount of current draw from the circuit. However, this is not the only parameter to consider. Some of the other important characteristics of the battery include the cycle life and rate of discharge for the battery. [11] We will now go through each step to calculate the total battery life of the circuit.

From our bracelet, the only two parts of our circuit that are directly connected to the battery are the vibrational motor and the Adafruit Trinket. The vibrational motor was rated at 80mA draw at 4V while the Adafruit Trinket was found to draw about 15mA at most. [12] If we are only using the vibrational motor 25% of the time, that means that our current draw would really be about 20mA. Without accounting for cycle life or rate of discharge, we calculate that we will need a battery life (BL) of

$$\text{BL} = \text{Total current draw (A)} * \text{Hours needed (H)}$$
$$\text{BL} = (15\text{mA} + 20\text{mA}) * 2\text{hrs}$$
$$\text{BL} = 70\text{mAh}$$

While this is well under the current batteries rating of 350mAh, we still need to account for cycle life. Cycle life is how much of the battery can be used before it should be charged again. Most batteries shouldn't run past 80% of their charge, and ours is no different. After looking at the data sheet, we see that it is recommended to not go past 70% of its charge. Using this metric, we get

$$\text{BL}^* = \text{BL (mAh)} / \text{Cycle Life (\%)}$$
$$\text{BL}^* = (70\text{mAh}) / 0.7$$
$$\text{BL}^* = 100\text{mAh}$$

Finally, the rate of discharge is needed to determine the final battery life needed. With certain battery chemistries, it can lead to much fewer amp hours if you discharge them quickly.

However, we need our battery to last at least 2 hours. A typical rule of thumb is that a battery will only get about half of its rated amp hours if it were fully discharged in an hour. [12] Therefore, we can expect around 60% of the rated amp hours for our LiPO. Taking this into account, we get

$$BL^{**} = BL^{*} (mAh) / \text{Rate of Discharge } (\%)$$
$$BL^{**} = (100mAh) / 0.6$$
$$BL^{**} = 166.667mAh$$

After calculating the rate of discharge, we can see that we would need a battery life of around 167 mAh to be able to use the bracelet for 2 hours. Therefore, our battery with 350mAh will be able to handle the requirements of our circuit.

3 Cost and Schedule

Cost Analysis

Labor

We have estimated that work on this project would be quoted at about $40.50/hour. This was calculated using the Average Computer Engineering Salary for a graduate of the ECE department in 2014-2015. [5] We are expecting to work on this project an estimated 10 hours a week, and it should take us about 10 weeks to complete our prototype. With all of this being said, we have calculated the total cost for both of us below:

Total = 2 partners x $40.50/hour x 10 hours/week x 10 weeks = $8100

Parts

The total cost of the parts along with a breakdown for each one is listed below:

Parts	Cost	Quantity
Intel Movidius Neural Compute Stick	$74.99	1
Raspberry Pi Development Kit	$99.50	1
IR LED Diode	$1.05	2

RGB Camera	$29.95	2
PCB	$58 (PCB way)	5
Adafruit Trinket M0	$8.95	2
IR Receiver	$1.95	2
Vibrating Mini Motor Disc	$1.95	2
Lithium Ion Polymer Battery	$6.95	2
Bracelet PCB (PCB Way)	$3.10	5
Total	$337.19	—

Total Cost

The total cost of development for our prototype comes out to about $8100 + $337.19 = $8437.19. We came to this figure after viewing the average starting salary of Computer Engineering Graduates from the University of Illinois along with researching the parts we would need in our design. While these costs seem high, they are needed only for the prototype and would become much more affordable at a higher quantity.

Part URLs:

Headset:

Intel NCSM2450.DK1 Movidius Neural Compute Stick:
https://www.amazon.com/Intel-NCSM2450-DK1-Movidius-Neural-Compute/dp/B076751BN8/ref=asc_df_B076751BN8/?tag=hyprod-20&linkCode=df0&hvadid=309807187084&hvpos=1o2&hvnetw=g&hvrand=15160679524891762428&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9022196&hvtargid=pla-492045282848&psc=1

Raspberry Pi Compute Module 3+ Dev Kit:
https://www.adafruit.com/product/4092?src=raspberrypi

IR LED Diode:
https://www.alliedelec.com/product/optek-tt-electronics-/op166a/70048885/?&mkwid=sjSorGz0k&pcrid=334950710611&pkw=&pmt=&gclid=CjwKCAiA2fjjBRAjEiwAuewS_TPDmwnKpS9NF98UW9ytfKFS67Mihc2ERQkGIZ0WwY2Y_6LwlaPp_BoCa_cQAvD_BwE&gclsrc=aw.ds

Raspberry Pi Camera Board v2 - 8 Megapixels:
https://www.adafruit.com/product/3099?src=raspberrypi

Bracelet:

TSOP38238 IR Receiver:
https://www.adafruit.com/product/157?gclid=CjwKCAiAqt7jBRAcEiwAof2uK1d-Zihbewwqe4hBGyGufDCSU9Qz0VgkV4cGRUL_Im4p9HiPXqmjnhoCowoQAvD_BwE

Adafruit Trinket M0:
https://www.adafruit.com/product/3500

Adafruit Vibrating Mini Motor Disc:
https://www.adafruit.com/product/1201?gclid=CjwKCAiAqt7jBRAcEiwAof2uK3LvvbFKIqwKgHNaywmqPvjn80dnbosf7CTSltzXOiHNRSl8ipaUnxoCW0EQAvD_BwE

Lithium Ion Polymer Battery - 3.7v 350mAh:
https://www.adafruit.com/product/2750

Schedule

Week	Alex	Daryl
2/23/19	Practice Design Review Presentation	Practice Design Review Presentation
3/2/19	Finalize PCB schematics for bracelets	Begin software development of bracelets
3/9/19	Finalize PCB schematics for headset Begin software development for rgb cameras, serial data transfer, and IR transmitter	Finalize software development of bracelets Begin training model for hand mapping and segmenting
3/16/19	Work on development of training model	Work on development of training model
3/23/19	Finalize raspberry pi routing code for rob cameras, serial data transfer, and IR transmitter	Work on development of training model
3/30/19	Finalize hand modeling software Begin work on Unity plugin	Finalize hand modeling software Begin work on Unity plugin

4/6/19	Build full prototype headset Finalize Unity plugin	Help build full prototype Begin design of Unity demo game
4/13/19	Test full prototype on Unity demo game	Test full prototype on Unity demo game
4/20/19	Begin presentation and final report	Iron out last minute bug fixes and clean up/comment all code
4/27/19	Practice Presentation, finalize report	Practice Presentation, finalize report

4 Safety and Ethics

With our proposed project, there are a few safety hazards that should be identified before advancing. In our we plan on eventually needing some sort of lithium-ion battery to charge the bracelets while they are running. Lithium-ion batteries have been known to occasionally fail, and this includes catching on fire or even exploding. Many of these failures come from the battery being exposed to a lot of heat at a full charge. [6] In our PCB design we will work on keeping the battery adequately cooled and add components to our circuit to prevent any unsafe discharge of power.

Our device is also meant to be an attachment to other VR headsets and a persons wrists, so it will be in close proximity to a users head and hands. If our device becomes to hot it can possibly harm the individual if it were to burn the user or overheat the circuit. We plan on preventing this by adequately cooling our circuit through adding heat sinks to warm areas and possibly adding a fan. There can also be a sensor to measure the heat of the device, which could act as a kill switch if it exceeds a certain temperature.

In the IEEE code of ethics, rule 9 states that we should “avoid injuring others, their property, reputation, or employment by false or malicious action”. [7] Since our device is considered an add-on or attachment for VR headsets, we would be forced to address this rule. While we would not be working directly for any of these VR companies, we are creating a product that could be affiliated with their brand and has a chance of causing injury to their reputation. We would work to address this issue by explicitly stating on our packaging or device that we are not in any way affiliated with the VR companies. Furthermore, if the product were to gain traction it would be necessary to contact the VR headset companies to work with them in creating an attachment to their platform that they agree with.

References

[1] Virtual reality. (2019, February 06). Retrieved from https://en.wikipedia.org/wiki/Virtual_reality

[2] Thier, D. (2014, March 21). Sony Announces 'Project Morpheus:' Virtual Reality Headset For PS4. Retrieved from https://www.forbes.com/sites/davidthier/2014/03/18/sony-announces-virtual-reality-headset-for-ps4/#73aab31d1639

[3] VR gets reality check with significant decline in investment. (2019, January 13). Retrieved from https://www.latimes.com/business/hollywood/la-fi-ct-virtual-reality-investment20190113-story.html

[4] Deals, T. (2016, March 24). This engine is dominating the gaming industry right now. Retrieved from https://thenextweb.com/gaming/2016/03/24/engine-dominating-gaming-industry-right-now/

[5] Services, E. I. (n.d.). Salary Averages. Retrieved from https://ece.illinois.edu/admissions/why-ece/salary-averages.asp

[6] BU-304a: Safety Concerns with Li-ion. (n.d.). Retrieved from https://batteryuniversity.com/learn/article/safety_concerns_with_li_ion

[7] IEEE Code of Ethics. (n.d.). Retrieved from https://www.ieee.org/about/corporate/governance/p7-8.html

[8] Camera Serial Interface. (2018, September 30). Retrieved from https://en.wikipedia.org/wiki/Camera_Serial_Interface

[9] Savage.home.automation. (n.d.). Retrieved from http://www.savagehomeautomation.com/projects/rs232-serial-on-raspberry-pi-compute-module.html

[10] Noronha, D. H., Salehpour, B., & Wilton, S. J. (2018, July 14). LeFlow: Enabling Flexible FPGA High-Level Synthesis of Tensorflow Deep Neural Networks. Retrieved from https://arxiv.org/pdf/1807.05317.pdf

[11] How to calculate battery run-time. (n.d.). Retrieved from https://www.powerstream.com/battery-capacity-calculations.html

[12] Cook, J. S. (2017, April 17). Optimizing Power Consumption for an Adafruit Pro Trinket. Retrieved from https://blog.hackster.io/optimizing-power-consumption-for-an-adafruit-pro-trinket-928f1e55a570