VR Glove Project

by

Akhila Liyanage (26451743)
Md Nadim Ahmed (26583321)

Supervised by
James Saunderson

Department of Electrical and Computer Systems Engineering
Monash University
June 1, 2019

1 Significant Contributions

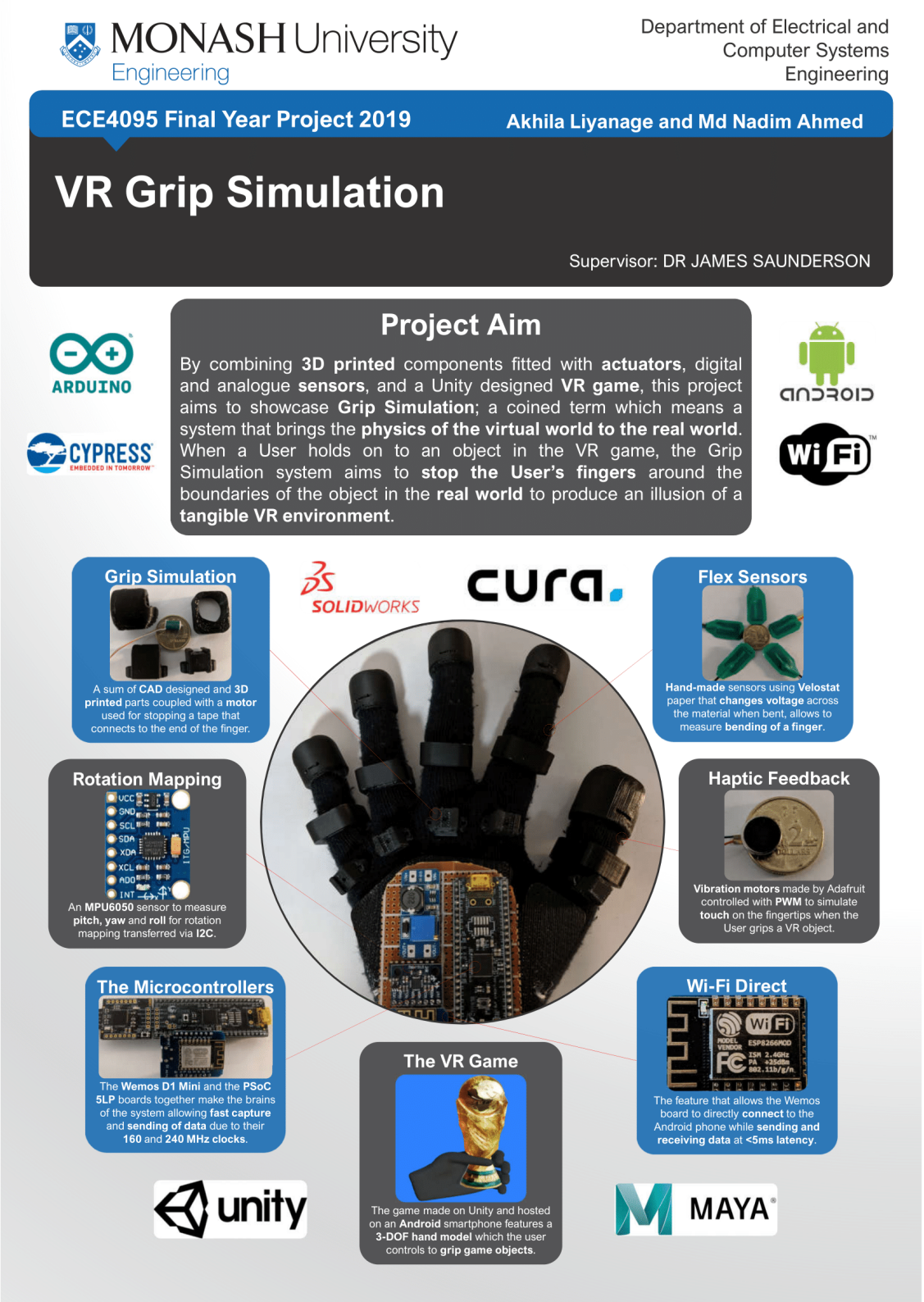
1. Designed a simple, cheap, compact and real time method to measure the amount of rotation when fingers are bent.
2. Designed a simple, cheap, compact and real time method to stop the user's finger (in the real world) when they are holding an object (in the virtual world).
3. Built a simple VR Game which interfaces with the Glove we built so that user can interact with the VR environment.

2 Acknowledgements

Over the course of our year, we received help from lots of different people for our project. Hence, we would like to thank the following people for their support, help and guidance -

- Our supervisor, James Saunderson for his continued assistance and guidance throughout the year.
- All the members of the Monash ECSE CAD CAM centre for assistance with the 3D printing for the project.
- All contributors to all the open source projects we used during the year, for example, Circuit Diagram, ESP8266 chips and its documentation page etc.
- All the various people on-line who helped in websites such as stackoverflow, YouTube etc.
- All the researches who worked on the papers we based a lot of our research and/or design on.

3 Project Poster



4 Executive Summary

This document details the VR Grip Simulation Glove, a Virtual Reality game controller that a user wears on their hand giving them full control over a VR environment. Grip Simulation is a coined term in this project which means a system that physically stops the User's fingers in the real world when they interact with a VR object in the virtual world. This is achieved with the use of actuators placed in CAD designed components to stop the movement of the User's fingers by forming a braking mechanism.

This controller was constructed as an affordable, portable, and compact solution so Users of any background can easily interact and experience it. The game runs on an Android smartphone placed in any inexpensive VR headset. The glove contains two microcontrollers; Wemos D1 Mini with Wi-Fi connectivity and the highly capable PSoC 5LP which undergoes a large portion of the computations required. These microcontrollers send and receive data to the VR Game environment quickly and wirelessly via Wi-Fi. The game was built on Unity, a game engine that is highly compatible with our requirements. The Glove also contains three other systems on top the Grip Simulation system. These are; Haptic feedback, Finger Bend sensing, and Orientation Tracking. Haptic feedback provides the sensation of touch on the User's fingertips when they touch a VR object. This is done by using small haptic motors placed under each of the fingers. Finger Bend sensing can independently track the bending of each of the User's knuckles with a total of 10 Bend sensors placed on the left hand. Finally, Orientation Tracking uses a Gyroscope and Accelerometer to measure rotation of the User's hand.

These components combine together in this VR Glove with the aim of providing the User with an immersive VR experience at an affordable cost. This document explains the process of solving each of the above mentioned systems and overcoming many of the challenges that were faced. Whilst there is plenty of room for improvement, this document shows the potential of the novel system constructed during the course of project. All files for this project can be found in [28].

Contents

List of Figures

List of Tables

5 Introduction

Figure 1: An example of VR Glove [46]

Figure 2: An example of VR Glove [26]

The main aim of this project is to build a glove that lets the user interact with a VR environment. The glove will contain a microcontroller, which controls sensors and actuators. This microcontroller will send sensor data to the VR Game environment, which will allow the Game to map a 3D model of hand with the movements of the real world glove. This glove should also send haptic feedback by giving the user the sensation of touch. In addition, our main focus will be implementing Grip Simulation. Grip Simulation is a term we coined, it means when the user's fingers are stopped by the glove in the real-world when the 3D hand model is holding an object in the virtual world. Figure 1 and figure 2 shows 2 examples of gloves, made by companies and researches to also allow users to interact with the VR environment.

6 Overview

6.1 Background

Virtual reality (VR) is an experience taking place within simulated and immersive environments that can be similar to or completely different from the real world. Applications of virtual reality can include platform for video games and educational purposes.

Due to increase in computing power of home devices and the dropping prices for VR headsets, the interest in VR technology has never been higher. Due to this rise, we expect the number of games specifically made for VR and also the application of VR in education to skyrocket in the next few years. For this reason, we decided it would be great to build a glove that can act as a controller for future VR applications.

6.2 Current Implementations

Figure 3: Full system used for Grip Simulation [26]

Many of the other projects have focused on tracking the motion of the user's hand and some have focused on sending the sense of touch to the user's hands. However, our focus, as mentioned before, is the Grip Simulation. Although this is not a new concept and as seen in figure 2, some companies and researchers are already working on it. However, as seen in figure 3, these systems are very bulky, expensive and difficult to manage. Hence, our aim in this project was to design a simple, compact, portable and cheap implementation of Grip Simulation.

6.3 Project Sub-systems and Glossary

In our implementation we are using the Wemos (an Arduino based microcontroller) as the master microcontroller and the PSoC as the slave microcontroller. The Game is built in Unity and the target platform is Android.

Important Clarifications For the rest of this report, when we refer to the Glove, we are referring to the glove we are building to interact with the VR environment. This Glove consists of the glove worn by the user including all the hardware components such as microntrollers, sensors and actuators etc. When we refer to the VR Game or simply Game, we are referring the test VR environment that the user is interacting with. When we refer to mobile phone or mobile device we are referring to the Android phone on which the VR Game is running.

This is a brief breakdown of our features and sub-systems -

Orientation Tracking Measuring the triple axes rotation of the Glove and mapping this movement onto the 3D hand model in the Game environment.

Finger Bend Tracking Measuring the rotation of the individual fingers of the user and mapping this data to the fingers of the 3D hand model.

Haptic Feedback When the 3D model touches an object in the VR environment, the user should feel a sensation of touch.

Grip Simulation When the user holds an object in the virtual world, the actuators in the Glove should prevent the user from moving past the boundaries of the object.

Position Mapping Measuring the triple axes position of the Glove and mapping that data to the 3D hand model.

7 Detailed Discussion and Test Results

7.1 Microcontrollers

7.1.1 Requirements

Large number of I/O pins For each finger of the Glove, a set of sensors and actuators are used to allow the user to interact with the VR environment. This resulted in a lot of pins being used by the microcontroller. By end of the project, 37 I/O were required for the microcontroller (15 for the Grip Simulation, 10 for Finger Bend Sensors, 5 for Haptic Feedback and 2 for Orientation Tracking). More details about the pin usage can be found in pages 10, 20, 14, 35 and 31.

5 PWM output pins 5 Pulse Width Modulation (PWM) pins are required for the Haptic Feedback sub-system. More details can be found in page 31.

Built-in ADC A fast Analogue-to-Digital Converter (ADC) is required for the Finger Bending Tracking sub-system. More details can be found in page 20.

10 Analogue Input Pins 10 Analogue Input Pins are required for the Finger Bending Tracking sub-system. More details can be found in page 20.

WiFi Capability Wifi Capability is required for data communication with the Game. More details can be found in page 10.

I2C Functionality I2C Functionality is required for use of the sensor used for Orientation Tracking. More details can be found in page 14.

Familiarity with Platform We had divided the work for the project into the different sub-systems and each team member was responsible for that sub-system. An example of a sub-system is Finger Bend Tracking or Grip Simulation. Each sub-system required heavy use of the microcontroller. Hence having a microcontroller we are both familiar with will significantly speeds up our work-flow.

Ease of use and Flexibility As mentioned before, both of us had to use the microcontroller extensively for the project. For this reason, if it is easy to use and flexible, it will significantly speed up our work-flow. Some desirable features include flexibility in pin assignments, supports Object Oriented Programming (since this helps with code organisation) etc.

7.1.2 Microcontrollers Considered

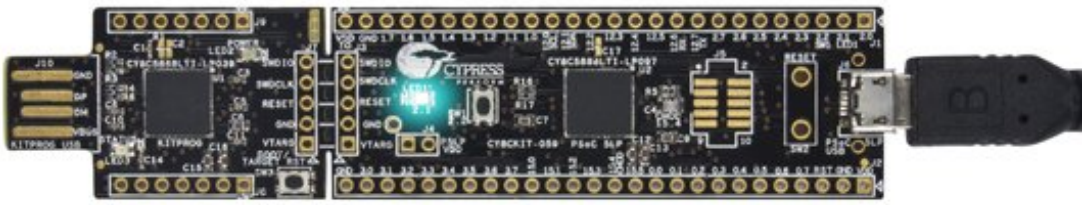


Figure 4: PSoC Board [19]

Figure 5: ESP8266 WiFi Chip [27]

Programmable System on Chip (PSoC) is the first microcontroller that was considered. This is an industry standard product that is very robust. We have learned to use this microcontroller fairly extensively in previous units. It also has all the hardware features that we have specified in our requirements except for the WiFi compatibility. We had attempted to compensate for this using the ESP8266 WiFi chip (as shown in figure 5). This chip is fairly popular amongst hobbyists and was compatible with Arduino devices. However, PSoC is not as popular as other microcontroller boards and hence not a lot of sensors are made to be compatible with it. After a lot of experimentation, we could not get the system working and hence we attempted to use other microcontroller boards as shown below.

Figure 6: Node MCU Board [32]

Node MCU is an open source microcontroller. It has a built-in ESP8266 chip and hence, has built-in WiFi capability. Node MCU is very popular on the market since you can program the device in the Arduino IDE.

This makes code organisation much easier since it supports C++ as opposed to PSoC which supports only C. It also supports sensors compatible with Arduino boards, which makes looking for sensors for it on-line significantly easier than the PSoC. It meets all the hardware requirements for our ideal board except that it does not have enough I/O pins. Even with multiplexers, there will not be enough pins on the board to run the full system in the end. The next sub-section discusses the final system in which we combine the 2 boards to solve this issue.

7.1.3 Final Design

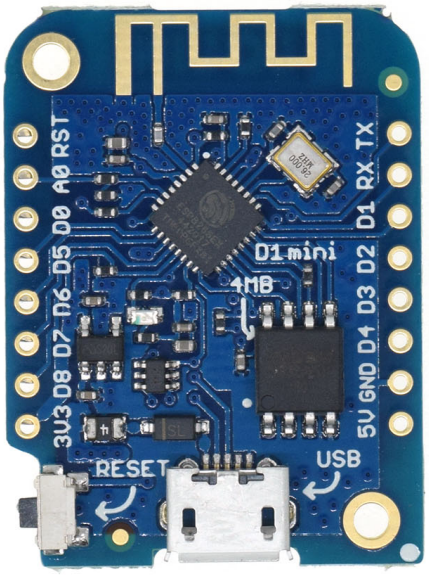


Figure 7: Wemos Board [20]

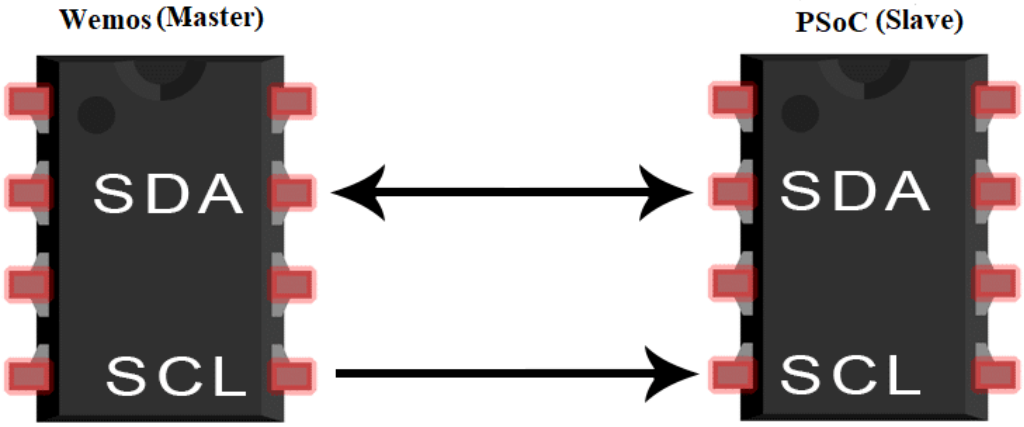


Figure 8: I2C Connection Between PSoC and Wemos boards [14]

The final design involves using a Master Slave I2C System. The I2C Bridge is used to send data between the 2 boards at 100 kbps. Figure 8 shows this connection. The Slave is a PSoC (as shown in figure 4) and the Master device is a Wemos (as shown in figure 7). A Wemos board is just a Node MCU board with a better form factor and fewer pins. The fewer pins on the Wemos isn't a problem since we have the PSoC as a Slave device.

The Wemos is in charge of all the WiFi communication with the VR Game. It is also in charge of calculating the data for the Orientation Tracking (as shown in 14) and sending that data to the Game. When the Wemos receives any data for the Haptic Feedback or Grip Simulation, that data is sent to the PSoC since it is in control of the actuators (as shown in 31 and 35). The PSoC is in charge of read data from the circuit for the Finger Bend Tracking (as explained in 20). The PSoC is polled by the Wemos at regular intervals for the Finger Bend data. Once this data is received by the Wemos, it is transferred to the VR Game. More information on how the sub-systems are connected are found in their individual sub-sections.

7.2 Game Development Platform

7.2.1 Requirements

VR Functionality Since the plan is to make an immersive experience for the user, VR would be the best platform for that. Hence, we need to look for a platform that already supports VR since it is beyond the scope of this project to implement it ourselves.

Easy and Free to develop on Game Development is not our expertise. The main focus of the project is to build the Glove. Hence, we do not need to spend too much time, money or resources making the actual game.

Cheap hardware The budget limitations on the project limits the kinds of the headsets we can afford. Additionally, since there are 2 project members, we would like to have one set of hardware for each member to work on individually. The cheap hardware is also important for the next requirement.

Widely Accessible We want the Glove so that it can be used by as many people as possible and not just hardcore VR enthusiasts. Hence, we would like to target a platform that is easily accessible for a lot of people.

7.2.2 Platforms Considered

Figure 9: High End VR Headsets [44]

Figure 9 shows the most popular high-end VR headsets that are on the market right now. Although these are powerful platforms to develop a VR application, they tend to cost around 400 and 900 AUD [4][6][8]. As mentioned before, we want the Game to be widely accessible and therefore, this factor makes them infeasible for this project.

For this reason, we selected our target platform to be Android. Since Android is the most popular mobile device operating system, it would make the project accessible for a large number of people. Additionally, beyond being able to render a VR environment, the processing requirements for the project is not that high. We just need to be able to render a human hand and all of its joints. Since it is an Android device it makes it easy to use WiFi Mobile Hotspot (which is what we are using for data communication with the Glove).

Figure 10: Google Cardboard [7]

Figure 11: VR Box [31]

In addition to an Android phone with the minimum requirements for rendering a VR environment, the user also requires a VR headset. These types of headsets are just holders for the phones so the user can strap

the phones onto their face. This headset is not required to run the Game but is useful to give the user the intended experience. For the simplicity of the VR headset, there are numerous ones available on the market. The most common one is Google Cardboard. Another one that could be used is the VR Box as shown in figure 11 (which is what are using to test and demonstrate the Game). But this is completely upto the user.

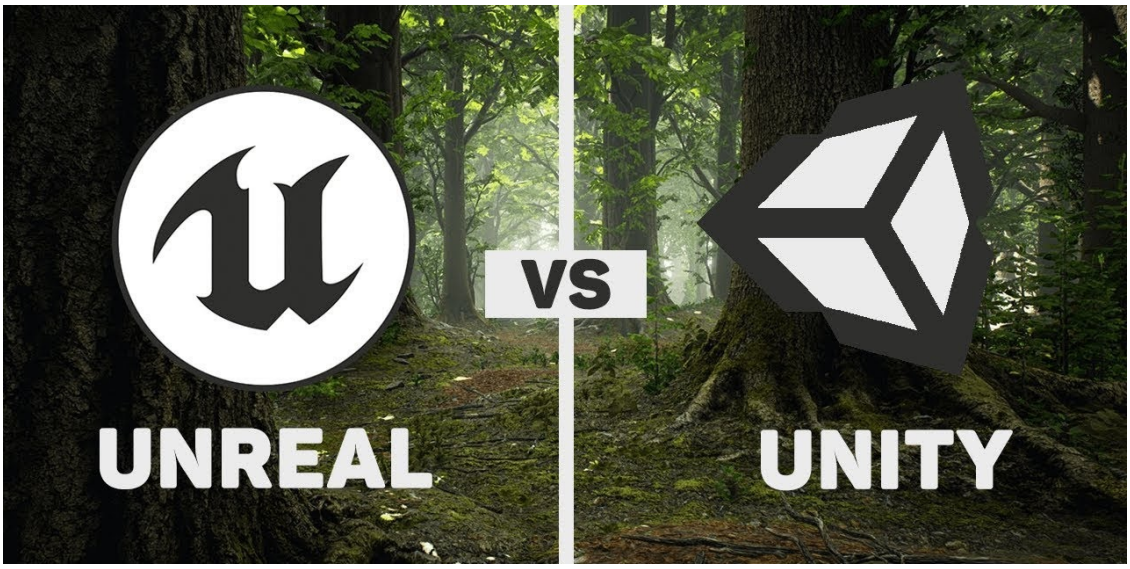


Figure 12: Unity and Unreal Engines [51]

Game Engines for development We considered both Unreal and Unity for use in building the VR Game we needed to test out the Glove. Both platforms are free-to-use and can support the Android VR platform for development built-in. However, Unreal has a much higher learning curve than Unity for beginner game developers. Since neither of us has no experience with Unreal or Unity, we decided to go with Unity as our Game Development Platform.

7.2.3 3D Model of Hand

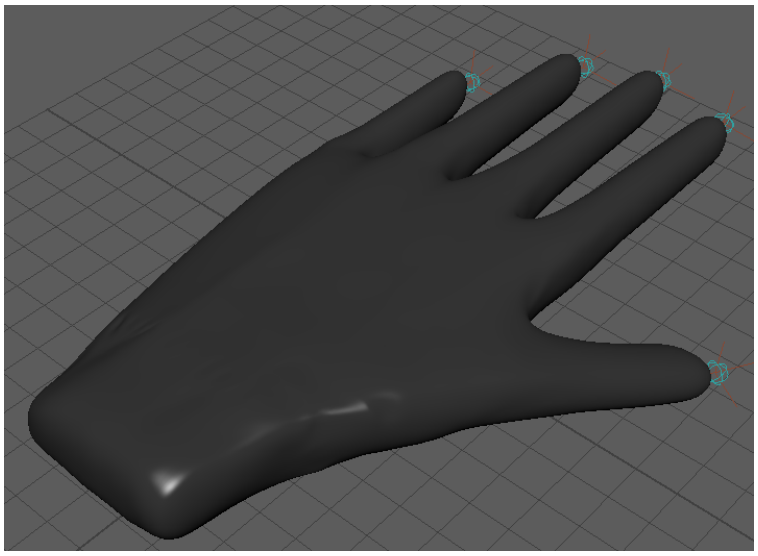


Figure 13: 3D Hand Model

Figure 13 shows the 3D hand model we built for the VR Game environment. The measurement used for the 3D model was based on the proportions of an average adult hand. This model was created from scratch in Maya, which is a popular and powerful 3D modelling program.

More details More details on how the Game and the Glove interact will be explained in the Full System subsection (page 50).

7.3 Data Communication Between Glove and Game

7.3.1 Requirements

For the user to interact with the VR environment in real time, data needs to be sent and received continuously between the Glove and the Game. To provide a good user the experience we had the following requirements for the communication between the Glove and Game: -

High Speed Communication For the Game to provide a good user experience, it needs to run at the standard 30 fps (30 frames per second). The data transfer speed should be fast enough so that the Game and the Glove have enough time to process their data and also meet the standard frames per second.

Wireless Communication The distance between the Game device and the Glove is unlikely to be more than 2 meters. Although wired communication between 2 meters is easy to achieve, we still decided to go with Wireless Data Transfer. This gives the user greater freedom of movement and likely to improve the user experience.

Simple and Flexible The project has a lot of moving parts and not a lot of time should be spent on the communication. The data transfer should be simple to use, capable of working in different ranges and capable of dealing with different packet sizes on the fly.

Cross-platform We also wanted the communication to be usable for any platform. This is so that in the case of the future research into this project, it can be easily migrated to a different platform.

7.3.2 Protocols Considered and Experimental Results

Figure 14: Bluetooth Vs WiFi [30]

The most common example for protocol used to send data between a mobile device and a mircocontroller is either WiFi or Bluetooth. Both comes different advantages and disadvantages. Both Bluetooth and WiFi compatible with most Android mobile phones. Since we could not find a reliable source for the data transmission speeds for these protocols, we decided to test out the modules ourselves.

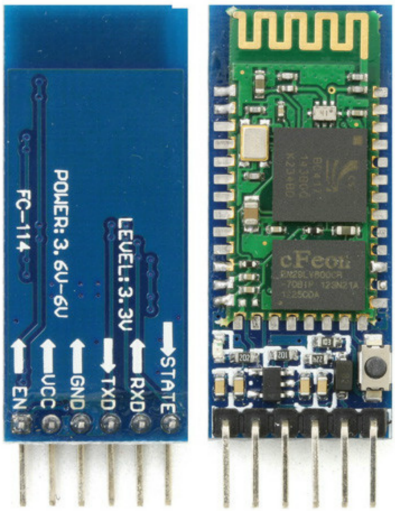


Figure 15: Bluetooth Module HC-05 [37]

The Bluetooth Module HC-06 (as shown in figure 15) is very popular with hobbyists due to its ease of use and compatibility with most microcontrollers that are available on the market. Bluetooth is also a Peer-to-Peer (P2P) communication protocol. Since the distance between the Glove and the Android device is unlikely to change significantly (i.e. not more 2 meters) during a game session, Bluetooth will provide a relatively consistent/predictable data transfer speed. However, after performing some tests, we found that for sending a standard packet it takes on average 30ms. Since we are aiming for a frame rate of 30, it is not acceptable for use in our project.

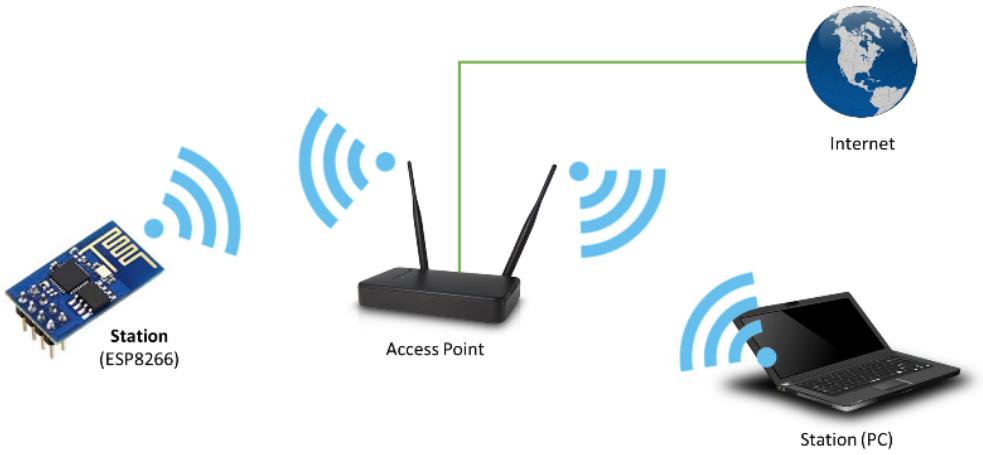


Figure 16: Station Mode [35]

In terms of compatibility with mobile devices, WiFi seems to rank the highest. However, WiFi is not a P2P protocol. The standard method (also known as Station Mode as shown in figure 16) for sending data between a mircocontroller and Android device requires a router as a middleman. Despite this, the data transfer speed from this set-up is between 5ms and 10ms, which is significantly faster than Bluetooth's transfer speed. However, since the network isn't P2P the data transfer can vary depending on the state of the router and the distance between the router and the Android device. After further research we found other protocols that build on WiFi but does not face some of the problems we just discussed. The 2 alternative protocols we considered were WiFi Direct and Software Access Point (SoftAp).

Figure 17: WiFi-Direct [22]

WiFi Direct allows us to create an Ad Hoc Network, which allows any device to connect to any other device in a P2P fashion [62]. This avoids a lot of the problems we discussed with Station Mode WiFi data transfer. However, WiFi Direct is relatively new when this project was being worked on. Although many mobile devices support it, we could not find any chips or microcontroller which support WiFi Direct. Chips such as ESP826 claim that support WiFi Direct [58] but they actually mean they support SoftAp which is not the same thing as we are about to discuss.

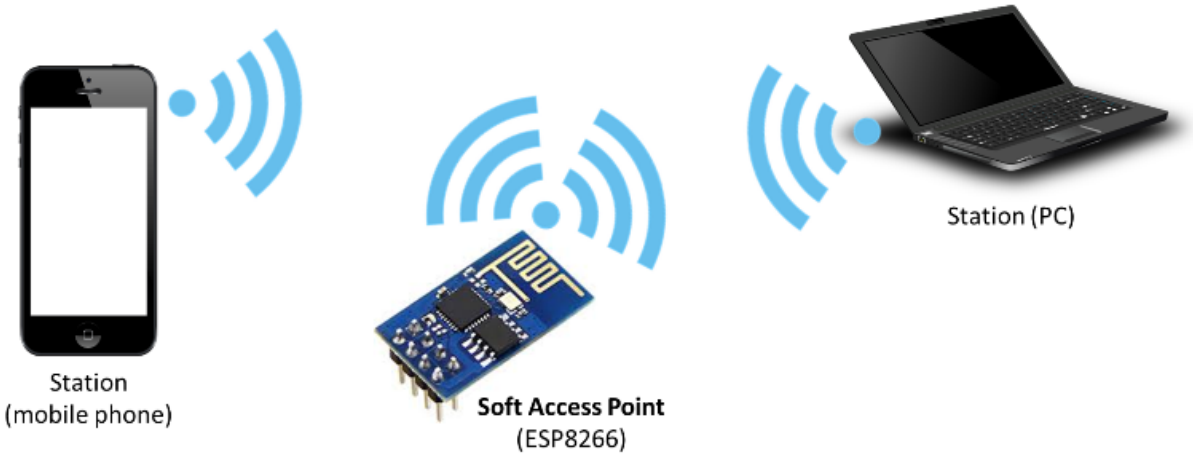


Figure 18: Software Access Point (SoftAp) Mode [35]

Software Access Points (SoftAp) is essential using your microcontroller as Mobile Hotspot. After testing this set-up, we found the data transfer delay to be less than 2ms, which is the fastest set-up we have tried so far. Unfortunately, there are certain limitations, one of them is higher energy consumption by the circuit.

Another major problem is that the device can only connect to one router at a time. Hence, when the mobile device connects to the SoftAp network they can no longer access the Internet.

One solution to the lack of the Internet problem is connecting the WiFi chip to a router that connects to the Internet. So, the mobile device should be using the WiFi chip to connect to the Internet. This again increases the energy consumption by the circuit and adds unnecessary complexity to the WiFi chip's functions. There 2 reasons why the lack of Internet connection on the device connecting to the WiFi chip is a problem as listed below -

Causes a poor user experience Too much of modern life is dependant on the Internet for the user to be willing to be cut-off from it.

Limits future potential for the Game We would this project to be built on in the future by other students/researchers. If the mobile device can't access the Internet puts a lot of limitations in what the Game can be used for in the future.

Causes problems for development work-flow To make the development process faster, during the debugging stage, we are using out laptops to build and run the Game instead of a mobile device. This is because the it makes the building process significantly faster and the Unity engine gives access to a lot of features that Game Development easier if the the Game is being run on the computer. Hence, if we go with the SoftAp set-up we cannot use the Internet while developing. This will result in a significant slow down of our work-flow.

The final set-up we are about to discuss is the one we ended up going with it. This involves use the mobile phone as WiFi hotspot instead of using the WiFi chip as the hotspot. This does allow us to use the Internet on the Game device during the game session, without additional complexity. It is also easier to implement that the SoftAp system. However, this moves the energy consumption burden on the mobile device. The mobile device will already have high energy consumption regardless due the VR Game. But it is still better than the other methods we considered even though this is significant drain on the battery. The details that have been discussed so far have been summarised in table 1

Table 1: Summarised Information for Data Communication

Criteria	**Bluetooth**	**Station Mode**	**SoftAp**	**Mobile Hotspot**
Data Transfer Delay (in ms)	30	5-10	2	2
Data Transfer Speed Consistent?	Consistent	Consistent	Inconsistent	Consistent
Difficulty to Implement?	Low	Medium	Medium	Low
Internet Available on Game Device?	Yes	Yes	No (yes if willing to do the extra work)	Yes
Energy Consumption	Low	Low	High on Circuit	High on Game Device

7.4 Orientation Tracking

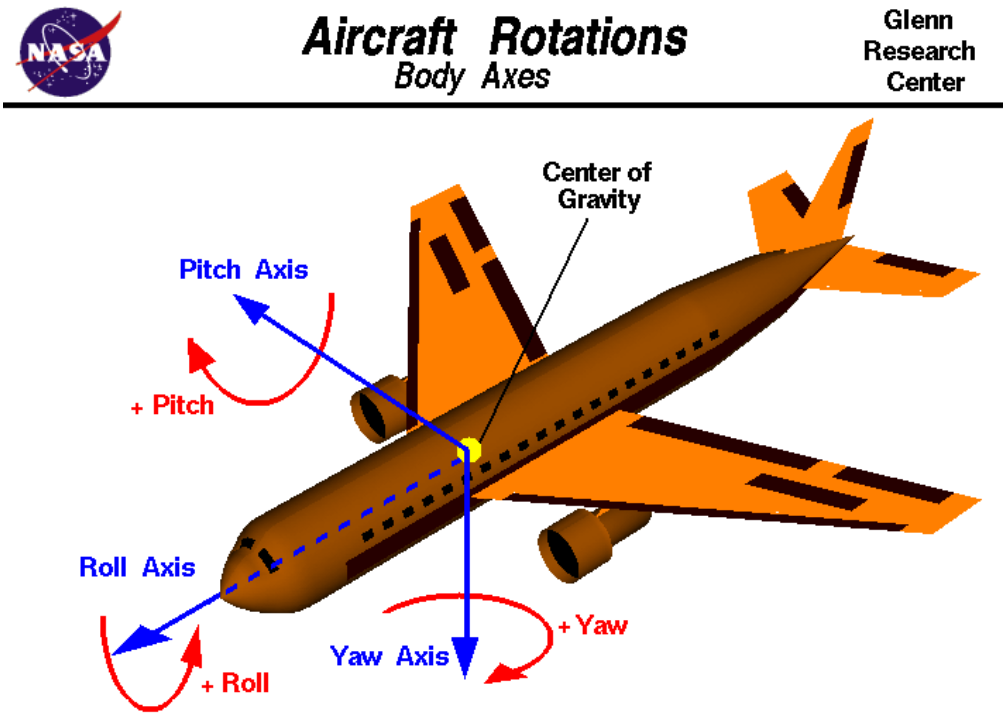


Figure 19: Orientation Axes [2]

7.4.1 Requirements

Calculate Pitch, Roll and Yaw Pitch, roll and yaw are the rotation of an object in the x, y and z axes, respectively (as shown in figure 19). These rotations need to be calculated for the user's hand. After these calculations, the data can be periodically transferred to the VR Game. Using this data, the 3D Model of the Glove in the VR environment can be moved in order to map the Orientation.

High Speed Calculation The intended experience is to give the user the illusion of the 3D model of a hand in the VR environment to be their own hand. For this reason, the orientation needs to be mapped in real time. The delay between the movement on the user's real hand and the virtual hand should not be noticeable for the user to realise. For this reason, the calculations need to be fast. Since the intended frame rate of the game is 30. We are aiming for the calculation to be take significantly lower to take into account the data transfer speed (more details can be found in 10) and the time taken to render the Virtual World and the 3D hand model.

Accuracy For the intended experience, the difference between the real hand and the virtual hand should not be noticeable. Hence, we are aiming for the calculated error to be less than 1°.

Should not accumulate drift The calculated pitch, roll and yaw angles should remain accuracy over time. If the orientation mapping sub-system accumulates errors over time, it would break the intended illusion to the user and result in a poor user experience.

7.4.2 Approaches Considered

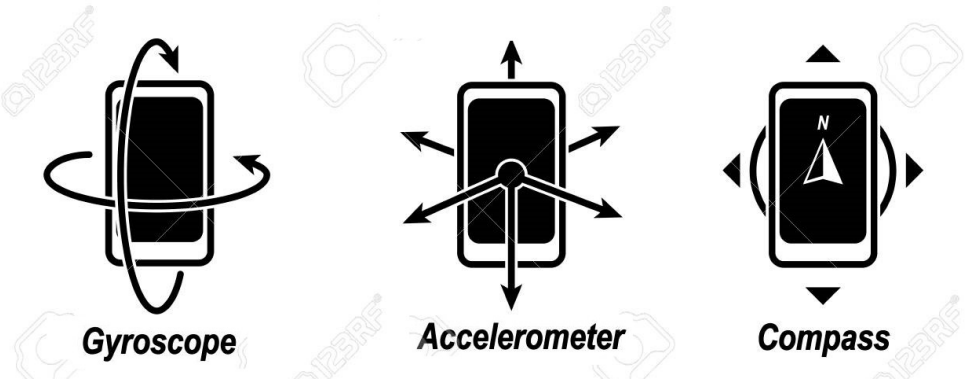


Figure 20: Orientation Sensors (in phones) [42]

Figure 20 shows the 3 sensors that were considered for the calculating the orientation of the user's hand in real time. These sensors are commonly used in mobile devices to calculate orientation and other positional data for a variety of applications. In this sub-section, we discuss how these sensors can be used to measure orientation values (i.e. pitch, roll and yaw). These sensors are very popular amongst hobbyists. Hence, the sensor boards that are available in the market come with built-in Analogue-to-Digital Converters. Since these types of boards were accurate and fast enough for our application, we used these off-the-shelf ones instead building them ourselves. For this reason, the inner workings of these sensors will not be discussed. The discussions will be centred around how well the data measured by these sensors can be used to meet the requirements of the application.

Gyroscope Gyroscope are the most popular set of sensors for applications involving measuring orientation. The data from the Gyroscope is the orientation of the sensor, in all 3 axes, measured in degrees/seconds ($^\circ/s$).

In order to convert the Gyroscope data into pitch, roll and yaw, it has to integrated with time. The pitch angle of the Glove can be calculated by the following formula -

$$Pitch = Pitch + GyroscopeX \times timeInterval; \tag{1}$$

where,
$Pitch$ = persistent variable that shows the current pitch of the sensor.
$GyroscopeX$ = orientation in the axis measured in degrees/second ($^\circ/s$).
$timeInterval$ = is the time period between each Gyroscope Sensor sample.

This procedure can be repeated for roll and yaw values. The calculation seems to work best when the addition is done at fixed time intervals. However, the sensor will face the following problems -

Accumulates drift over time The major problem with this sensor is that the calculated orientation values tend to drift over time. This is due to the error from the sensor readings that are accumulated over time.

Only relative orientation is calculated At the beginning of the Game session, the value of the orientation variables are set to 0. Hence, the calculated orientation will always to be relative to the original position of the Glove at the beginning of the Game session instead of the absolute position. An easy fix of this problem is to force the user to keep their hand to be perfectly level at the beginning of the game. Otherwise, the mapped orientation in the VR environment will not be correct and will break the intended illusion.

Accelerometer The accelerometers provide us with the linear acceleration experienced by the sensor. By using the formula below, the pitch and roll angles of the sensor can be calculated -

$$Pitch = \arctan(\frac{-Ax}{\sqrt{Ay^2 + Az^2}}) \tag{2}$$

$$Roll = \arctan(\frac{Ay}{Az}) \tag{3}$$

where,
$Pitch$ = absolute pitch that is calculated
$Roll$ = absolute roll that is calculated
Ax = linear acceleration in the x-direction that is calculated by the accelerometer
Ay = linear acceleration in the y-direction that is calculated by the accelerometer
Az = linear acceleration in the z-direction that is calculated by the accelerometer

The derivation of equation (2) and equation (3) can found in [65]. However, this method runs into the following problems as described below -

Does not calculate Yaw Yaw cannot be calculated by using the accelerometer values and hence, 3 degrees of freedom cannot be achieved by the VR Glove.

Values are not stable The Pitch and Roll calculated by the data tends not be stable. This might be cause by the force exerted by the user's hand.

Digital Compass/Hall effect Sensor A digital compass can be used to measure the magnetic field in the x,y, and z-direction. Using this information, the orientation of the Glove can be calculated using the Earth's magnetic field as a reference point. However, this can cause the following issues -

Interference from other devices Lots of electrical devices emit Electromagnetic Interference. This might cause errors in the compass measurements which need to be filtered out or the circuit needs to be shielded from external sources of interference. The interference can be particularly high in place such as the Electrical Labs.

Interference from other components in the circuit The circuit itself uses 2 sets of 5 motors for the Haptic Feedback and Grip Simulation (as shown in pages 31 and 35). The electromagnetic interference from the motors will cause problems for the compass measurements.

Magnetometer are also being used for Position Mapping We had intended to use a magnetometer and an AC magnetic field source as a base station for position mapping. More details can be found in page 44. Although due to time constraints we were unable to get it working, it is something we would be added on to the Glove if future work is done on the project.

7.4.3 Final Design and Chosen Sensor

Our final design involved using both a gyroscope and an accelerometer for calculating the orientation. With the data gathered from both sensors, we have hoped to mitigate the limitations of both sensors. We used an off-the-shelf sensor (i.e. MPU-6050 as shown in figure 21). This sensor has a built-in 3-axes gyroscope and accelerometer. The board's built-in Analogue-to-Digital Converter (ADC) convert the analogue data from the sensors to usable by the microcontroller.

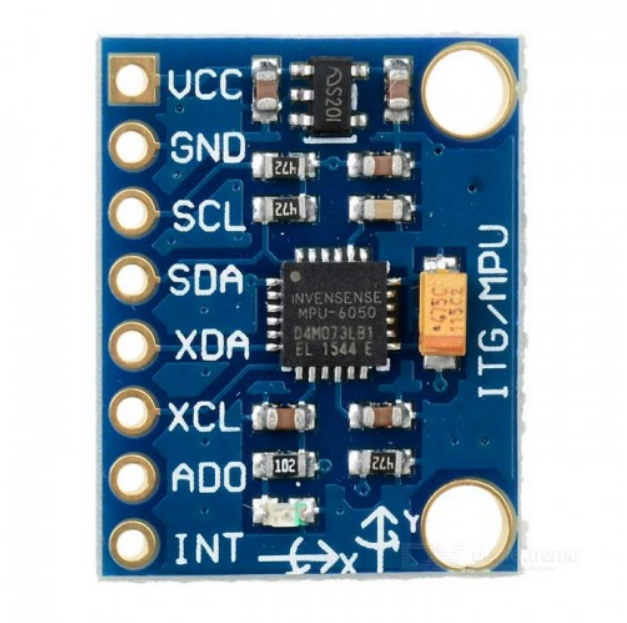


Figure 21: MPU-6050 Sensor [25]

The data processed by the ADCs is sent to the microcontroller via I2C. In the I2C circuit, the MPU6050 acts as the slave device while the microcontroller acts as the master. The microntroller master chosen is the Wemos as opposed to the PSoC (more details can be found in 4). This is because since the Wemos is already the master to the PSoC, adding another slave is easier than building a multi-master I2C system. In addition, since the Wemos is responsible for sending the data to the Game via WiFi anyway, it will be faster for the orientation data to be calculated by the Wemos Master. Figure 22 shows the I2C circuit between the Wemos and the MPU6050.

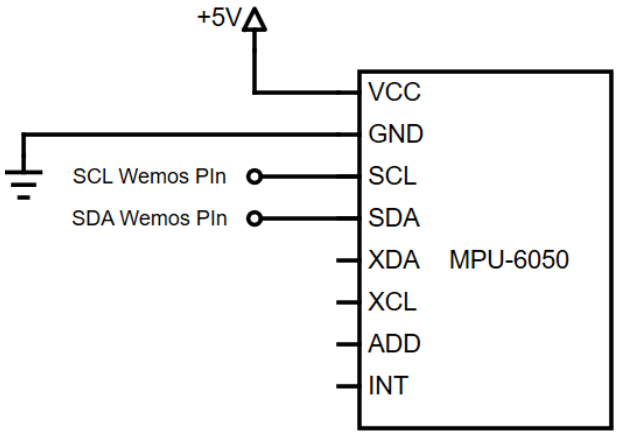


Figure 22: MPU-6050 Circuit Connection

Alpha Beta filter The Alpha Beta filter is used to combine the data from the accelerometer and gyroscope to calculate more accurate orientation data. An alpha beta filter (also called alpha-beta filter, f-g filter or g-h filter) is a simplified form of observer for estimation, data smoothing and control applications. It is related to Kalman Filters but is a more simplified model [61]. Reminder that this technique is only applied to the calculation of the pitch and roll and not yaw.

An alpha beta filter assumes that the state we are trying to calculate (in this case pitch or roll) can estimated by a model involving two internal states (in this case the accelerometer data and the gyroscope data). In this model it is assumed that the first state (the pitch or roll calculated by the accelerometer data) can estimated by integrating the second state (the gyroscope data) over time [63]. The filter takes 2 constants called alpha and beta respectively which is used to correct the gyroscope and accelerometer data. Hence, the final result from the filter can estimated using the following equations -

$$Pitch_{out} = \alpha \times Pitch_{accelerometer} + \beta \times Pitch_{gyroscope} \tag{4}$$

where,
$Pitch_{out}$ = the calculated pitch value after filtering
$Pitch_{accelerometer}$ = pitch calculated using accelerometer data
$Pitch_{gyroscope}$ = pitch calculated using gyroscope data
α = constant used to correct accelerometer data
β = constant used to correct gyroscope data

For convergence and stability, the values of alpha and beta must meet the following inequalities -

$$0 < \alpha < 1 \tag{5}$$

$$0 < \beta \leq 2 \tag{6}$$

$$0 < 4 - 2\alpha - \beta \tag{7}$$

For noise suppression, beta needs to meet the following inequality -

$$0 < \beta < 1 \tag{8}$$

The above equations can be gathered from [63] and [61]. The values for alpha and beta are obtained through experimentation. Since the accelerometer data is more noisy when compared with the gyroscope data. For this reason, a smaller value for alpha is chosen when compared to the value for beta. After researching on-line and experimentation of our own, we finalised on the following values for alpha and beta (more information about the experiments can be found in the following sub-sections) -

$$\alpha = 0.02 \tag{9}$$

$$\beta = 0.98 \tag{10}$$

7.4.4 Methodology and Experimental Results

We use regular set squares and protractors (found in most geometry box sets) to determine the accuracy of the Orientation measurement and calculation. During testing, the Wemos and MPU-6050 are connected to a breadboard. The breadboard is then inclined on a plane as seen in the figure 23.

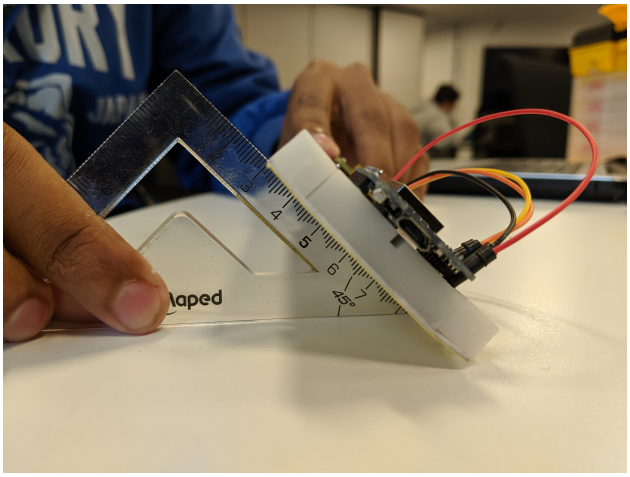


Figure 23: Orientation Test Example

In figure 23 the breadboard is inclined on the 45° slope of the set square. The MPU-6050 isn't exactly level with breadboard, so we use the protractor to measure the incline of the sensor. After this done, the Orientation values (i.e. pitch, roll and yaw) calculated by the Wemos are displayed to the screen roughly every second. The incline is placed at different directions relative to the MPU-6050 as seen in figure 24 to see the change on the different calculated values. In this way, the accuracy of each orientation value can be determined. We also use another set square to create a slope of 30° or 60°.

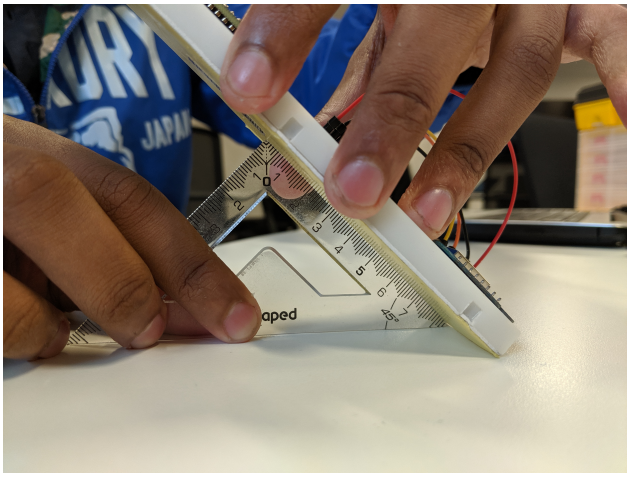


Figure 24: Orientation Test Example2

After many of these types of experiments we found that the error tolerance of the orientation data is less than 1°, which meets our accuracy requirement. This meets the error requirement for different speeds of rotation changes. In addition, the calculation is done at 250 Hz and hence, meets our speed requirements for the Game.

7.5 Finger Bend Tracking

7.5.1 Understanding the Sensor Behaviour

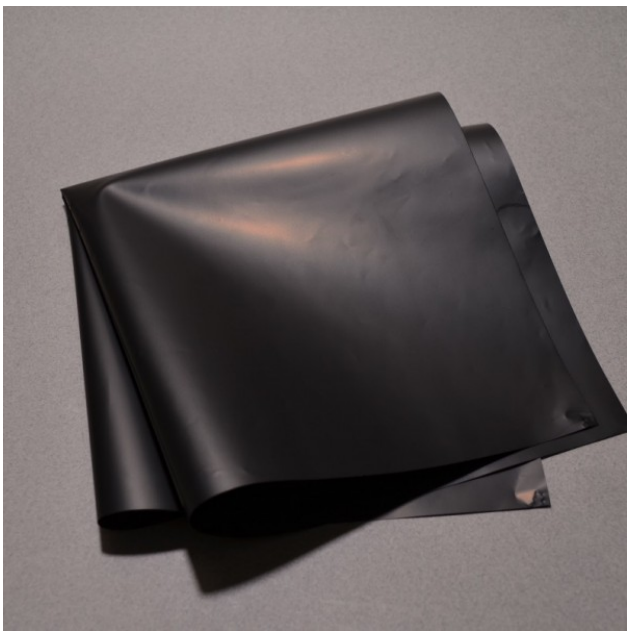


Figure 25: Velostat Sheet [12]

Several methods exist that allows to measure the bending of fingers, both digital and analogue. One method is to use a camera to capture movement of the User's hand and interpret each movement using computer vision. However, since the approach of this project was to create a completely portable VR solution, this method would not meet the requirements. Hence, a sensor built directly into the Glove was considered. A popular off-the-shelf sensor for measuring bend is the Flex Sensor by Spectra Symbol [67] as shown in figure 26. The patented technology of this sensor claims to be comprised of a conductive substrate that changes resistance when deflected/bent [29]. The sensor by Spectra Symbol is available in two sizes; 2.2 and 4.5 inches. A typical application of the sensor is by connecting it in a voltage divider as the R_1 resistor coupled with a unity gain buffer [67]. Given the size of the sensor and the fact that it is a professionally manufactured product which has likely undergone rigorous testing, the Flex Sensor seemed like an ideal solution for this project. Unfortunately, the biggest limitation of the product is its price. As of writing, the current price of the 2.2" sensor is 15.16 AUD per unit on Core Electronics [16]. A single sensor can only measure a single point of bend on the hand. As the design requirement of this project is measuring bend at two points per finger totalling ten points for the hand, ten Flex Sensors would need to be purchased. With a total of 151.6 AUD, the VR Glove is no longer an affordable solution. However, with further research into bend sensing, another interesting material with similar properties was found and used in the final construction of the Glove.

Velostat Velostat or Linqstat is a pressure-sensitive conductive material that changes resistance when deflected/bent [13] similar to the Flex Sensor. One difference though is that Flex Sensors increase resistance when bent whereas Velostat decreases resistance [17]. Velostat is composed of a carbon-impregnated polyolefin material and is typically used as static protection for electronic/chemical products [13]. A convincing argument for the use of Velostat instead of the Flex Sensor is the price. As of writing, a 28 cm $\times$ 28 cm sheet of Velostat as shown in figure 27 is priced at just 7.50 AUD on Core Electronics [17]. For the purpose of this project, a fraction of a single sheet of Velostat is required to fulfil the design requirements. While this makes Velostat an affordable solution, however, the challenge remained of constructing the ten sensors for the Glove.

Creating the sensors at home has both its pros and cons. It would be possible to create the sensors exactly according to the design specifications and limitations of the Glove, particularly in terms of the available space. On the other hand, a fair amount of experimentation is required before achieving a workable solution, if such a solution exists in the first place. Fortunately, the sheet of Velostat purchased was more than sufficient for testing out multiple prototypes of the sensor.

Prototype 1 The first build was quick and simple. Since there is little space available above the knuckles of the hand, we cut a 30 mm $\times$ 20 mm piece of the Velostat sheet and connected cables to either end of the sheet as seen in figure 26. To give a better idea as to how this forms a resistor, refer to figure 27 which shows the build from its side. Essentially, the current will now flow through the thickness of the material. This shape seemed reasonable enough to fit over the knuckles and long enough to bend up to 90° while still being in-line with the finger. This variable resistor was then measured for resistance using a multimeter. When laid flat or when no bend is applied, the resistance was recorded at approximately 30.5 kΩ. When bent to 90°, the resistance was recorded at approximately 15.8 kΩ. This gives a range of 14.7 kΩ. The reduction in resistance when bent is consistent with the properties of the material as laid out by core electronics [17].

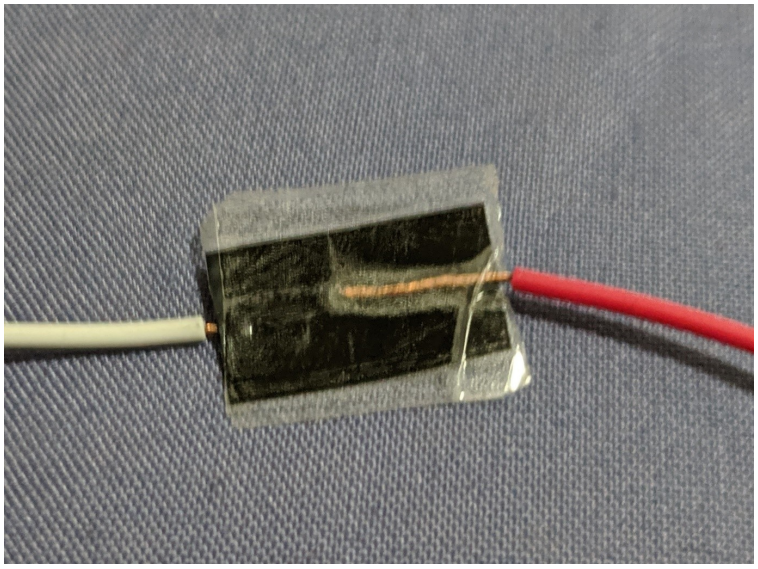


Figure 26: Bend Sensor Prototype 1 build

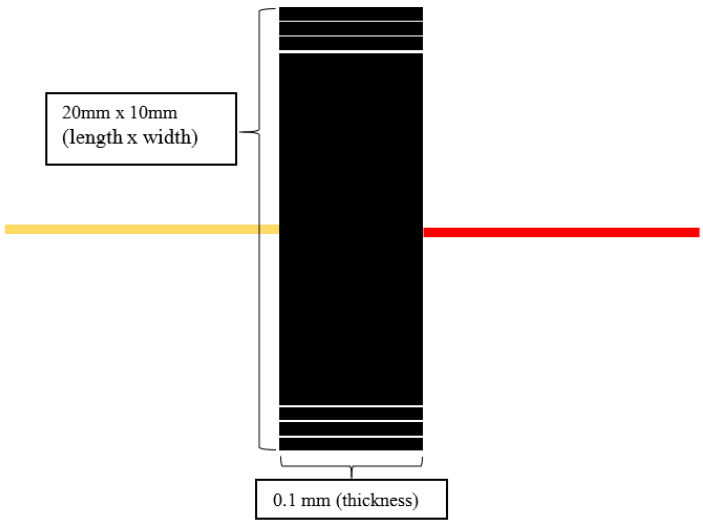


Figure 27: Bend Sensor Prototype 1 diagram

Next, the thermal properties of the material were tested. For our project, we do not want any amount of heat dissipating from the components which can be deemed hazardous. Therefore, the Velostat should not emit heat of this level when in operation. This was tested by checking at what maximum voltage a single rectangular cut-out of Velostat can safely operate. Theoretically, a resistance of 30 kΩ should only draw 0.4 mA when supplied with a voltage of 12 V. Fortunately, during the test when powered on for 30 seconds, the Velostat cut-out did not emit hazardous amounts of heat.

Finally, the sensitivity of this Prototype build was tested. We did this by connecting the build in a voltage divider configuration as the R_1 resistor. R_2 was selected as 500 Ω. The voltage across R_2 was measured using an Arduino Uno board through one of its analogue pins. The Arduino IDE has a useful Serial plotter tool that automatically plots incoming Rx data from an analogue channel with labelled axis. During the test, the Velostat sheet was first started off laid flat at 0°. Then, the sheet was gradually bent over time up to 90°. Figure 28 shows the result from Serial plotter.

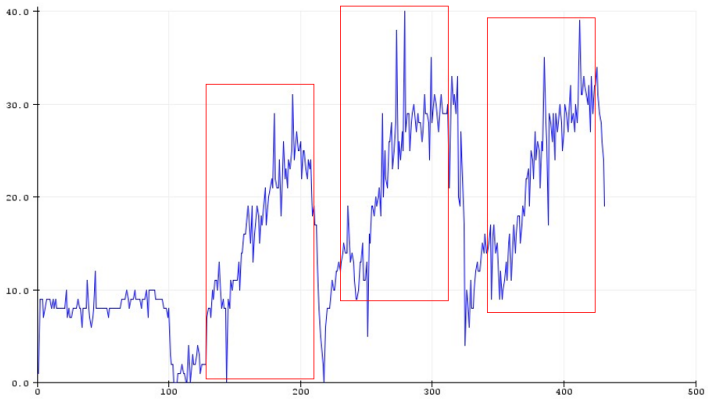


Figure 28: Prototype 2 sensitivity plot

The y axis shows the analogue read data as the raw count with a range of 1024. Each of the curves indicated by the red boxes is the Velostat being bent from 0° to 90° . What is quickly apparent is the maximum value the build could reach was 40 and the range was only 30. This is a sensitivity of approximately just 0.3 counts per degree. For our project, this value is far too low to be feasible and so we decided to make a different build.

Prototype 2 As current flows through the thickness of the material, increasing this thickness should increase the resistance through the material according to Ohm's Law. This is given by the equation $R = \frac{\rho \times L}{A}$, where R is the resistance of the material, ρ is the resistivity of the material, L is the length, and A is the cross-sectional area. Here, L is the thickness of the Velostat resistor used in our project and is proportional to Resistance. For Prototype 2, we placed three cut-outs of the Velostat sheet with the same dimensions sandwiched together and measured its properties. As expected, the resistance of this build when laid flat was 41.1 kΩ. More importantly however, the resistance when bent to 90° was 8.48 kΩ. This gives a range of 32.62 kΩ, more than double that of the previous Prototype. The sensitivity test was conducted in the same method as the previous test. Figure 29 shows the plot of this test.

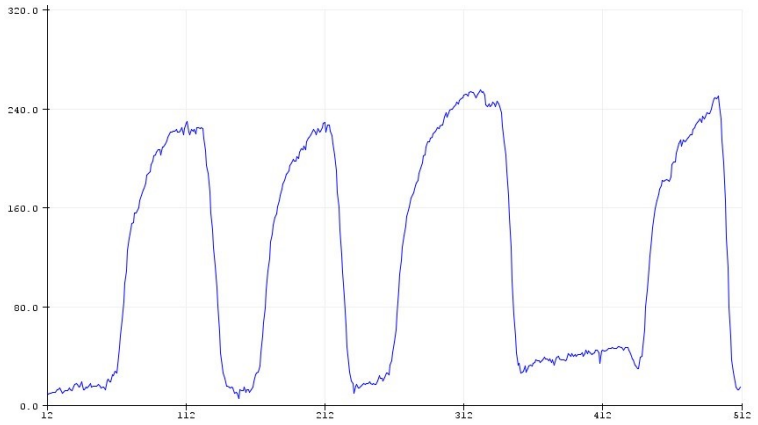


Figure 29: Prototype 2 sensitivity plot

Similarly, the y axis shows the analogue read data as the raw count with a range of 1024. Each of the curves is the Velostat being bent from 0° to 90°. This time, the build had a maximum value of 245 and a range of approximately 230. This is a sensitivity of approximately 2.6 counts per degree. This is a much more feasible sensitivity value and so we decided to continue the use of at least three layers of Velostat cut-outs for all five fingers in future builds and tests.

Prototype 3 This build involved finding a suitable container for the Velostat and the copper wires. The structure would need to be flexible whilst securely holding the Velostat. It would also need to be small enough to reside on the limited space of the Glove. For the previous builds, regular adhesive clear tape was used. However, with regular bending of the sensor, this tape started to give way and affect the performance of the sensor. For this prototype, we decided to test out electrical tape. Electrical tape is a good insulator and has the added bonus of being very adhesive. It is also just as flexible as clear tape. Figure 30 shows this prototype, the connection wires are now placed parallel to each other but still remain on opposite sides of the Velostat layers. Figure 31 shows this more clearly from its side.

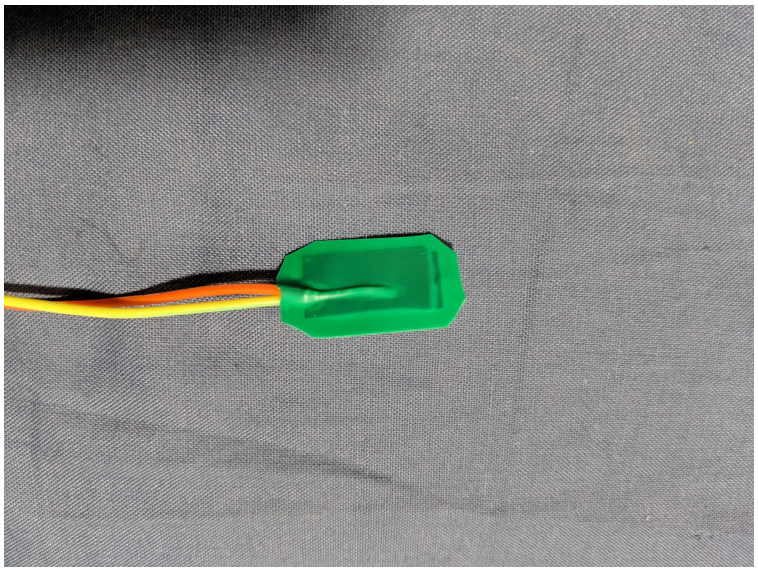


Figure 30: Bend Sensor Prototype 3 build

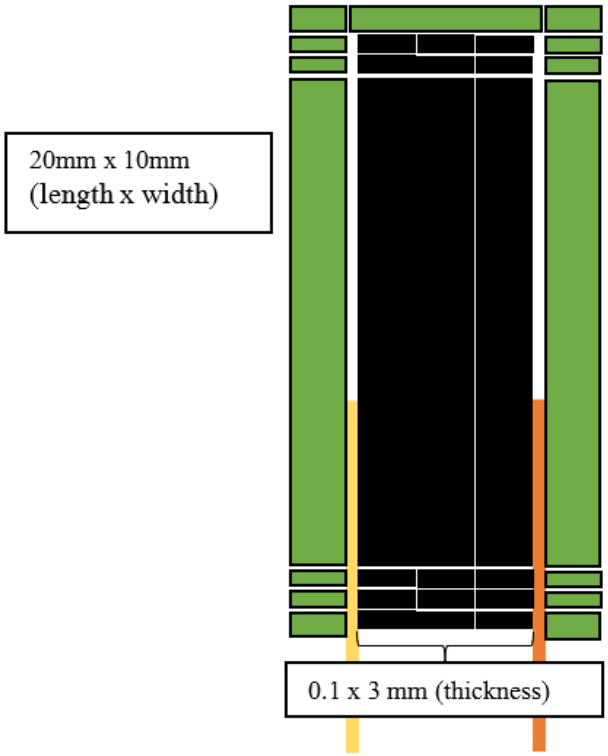


Figure 31: Prototype 3 Diagram

The sensitivity test yielded the same results as Prototype 2 as expected and thermal properties of the sensor was within same limits.

7.5.2 Circuit Diagrams

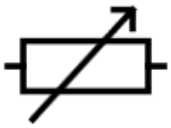


Figure 32: Variable Resistor Symbol

Figure 32 shows the circuit symbol we have chosen to represent the flex sensor we built using velostat.

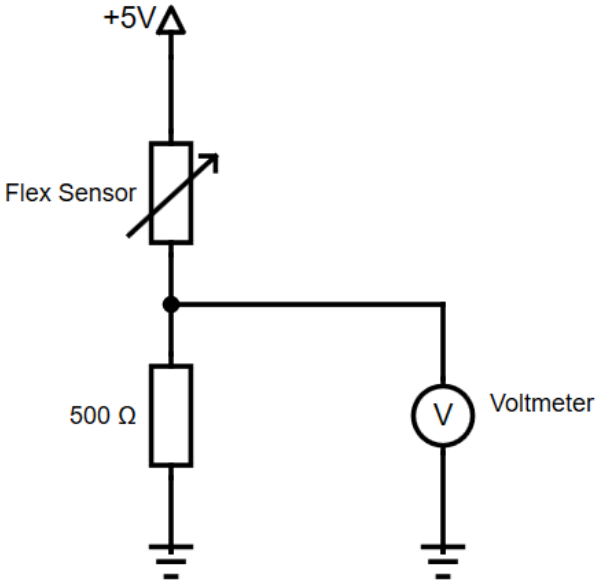


Figure 33: Voltmeter Measurement of a Flex Sensor Circuit

As mentioned before, the resistance of the flex sensor decreases when the user's finger bends. Figure 33 shows how this property of the flex sensor can be measured using a voltmeter. The flex sensor is connected in series with a 500 Ω resistor. When the resistance of the flex sensor decreases, the voltage measured by the voltmeter (which is the voltage across the 500 Ω resistor) increases.

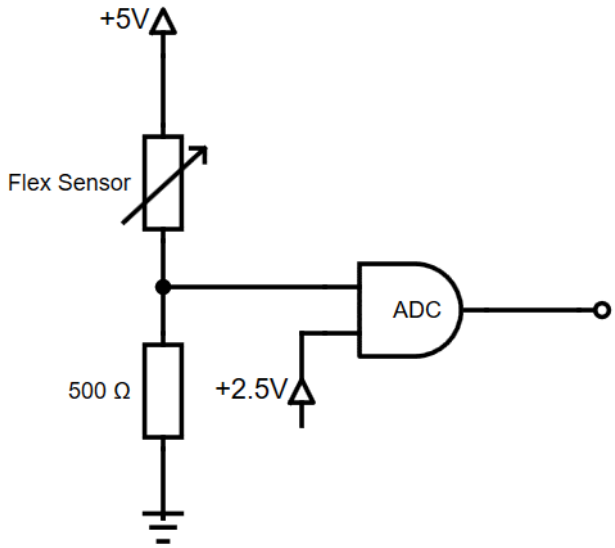


Figure 34: ADC Measurement of Flex Sensor Circuit

Figure 34 shows voltage of the 500Ω resistor is measured by an analogue-to-digital converter (ADC). The ADC is run at a 12-bit resolution and uses 2.5V as a reference. This is because with the flex sensor we have constructed the voltage across the 500Ω resistor never goes below 2.5V. Having 2.5V as a reference allows us to take higher resolution measurements without getting a more powerful/expensive ADC.

In this way, the change in voltage across the resistor becomes useful information that can be processed by the microcontroller. After this has been processed by the mircocontroller, it can be sent to the VR Game environment. This processed data is then used by the VR Game environment to apply the appropriate rotations to the 3D model of the user's hand. More details can be found between pages 7 and 10.

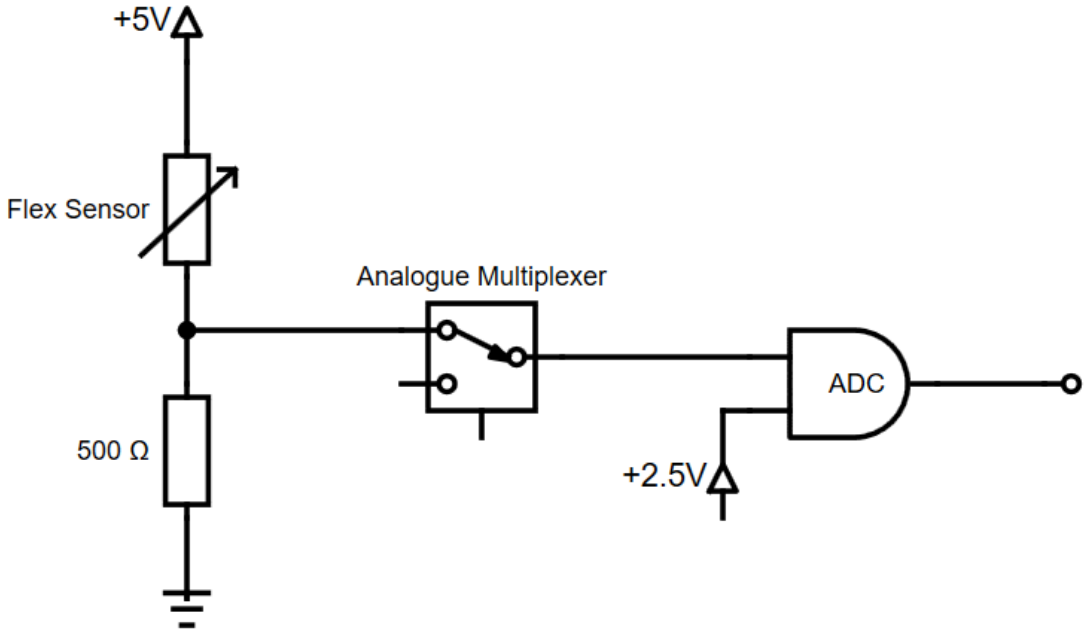


Figure 35: Flex Sensor Circuit Multiplexed

ADCs are fairly expensive resource on most circuits. Although we considered using a dedicated ADC for each flex sensor (in total 10), we decided to go with the general rule of thumb which is to simply multiplex the input pins. This is because it would add a significant financial cost on the building of the Glove. (Note: figure 35 shows only one flex sensor circuit being multiplexed on a 2:1 multiplexer for the sake of simplicity).

However, we ran into certain issues when trying to get data from 10 flex sensor circuits. We would find that when output on just on flex sensor should be changing the output from all the other flex sensor circuits are also changing. This happened even though each flex sensor circuit was tested individually. This behaviour only happened when there were multiple sensors outputs being multiplexed. Even the multiplexer was tested for problems but that wasn't the issue. The problematic behaviour would go away if we delay around 10ms between each measurement. If each 10 of the flex sensor circuits needed 10ms to get correct measurements, the Game will not able to map the user's finger movements effectively enough and would result in a poor user experience. After a while we realised that the problem was caused by the capacitor in the ADC. The ADC uses a capacitor to 'record' the voltage that is input to the ADC. Since there was not enough current input to the ADC, the internal capacitor was not charging fast enough. To solve this problem, we would have to find to way to increase the current input to the ADC without changing the voltage output.

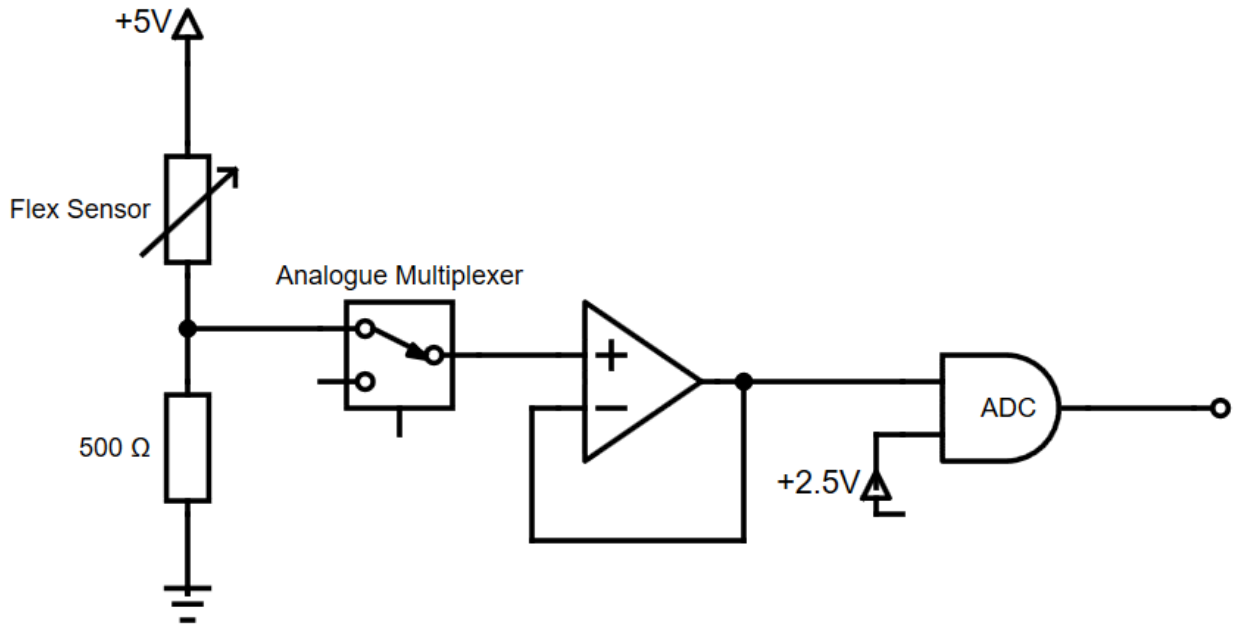


Figure 36: Flex Sensor Circuit Current Amplified

Figure 36 provides a solution to the problem we just discussed. An operational amplifier (op-amp) is placed between the Multiplexer output and the ADC input. The op-amp is set in a voltage follower configuration. This configuration results in the output of the op-amp to have the same voltage as the input voltage. However, since the op-amp has a low impedance output, there is more current flowing into the input of the ADC and hence, more current for the internal capacitor of the ADC. This resulted in an acceptable speed for the flex sensor measurement. More details about the final measurement speed can found in the following sub-section.

Since we wanted to the final circuit to be compact enough to be placed on the user's hand, we wanted to reduce the number of circuit components being used. The PSoC has the analogue multiplexer, op-amp and ADC built-in and makes it very easy to control. The final circuit (external to the PSoC) for the flex sensor can found in figure 37

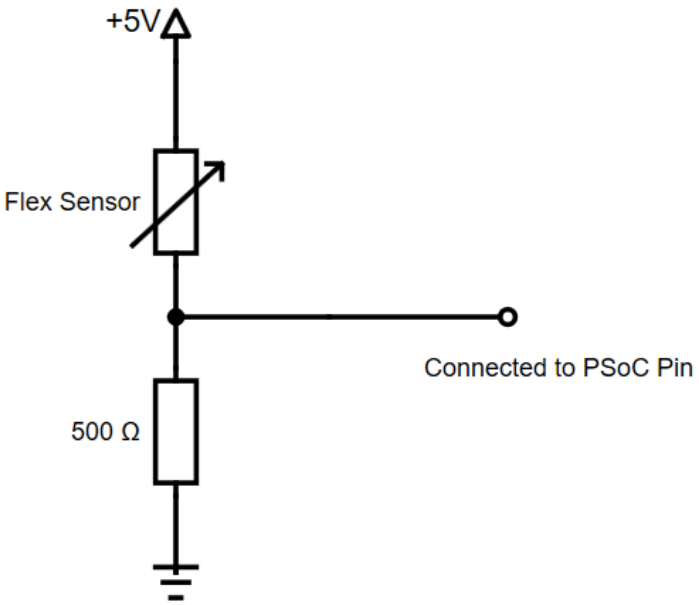


Figure 37: Final Flex Sensor Circuit

7.5.3 Methodology and Experimental Results

10 flex sensors are used by the 5 fingers that are tracked. These sensors are placed on the back of the user's hand at the various important finger joints. Using the data from the flex sensors, the rotation of the finger joints can be estimated. The location of the flex sensors on the user's hand is shown in the figure 38.

Figure 38: Flex Sensor Locations on User's hand

As seen in figure 38, we are not attaching the flex sensors on the top joints on each finger. This is because by measuring the rotation of the lower joints the rotation of the top joints can be easily predicted, within reasonable accuracy. This enables us to use 10 flex sensors to track the rotation of the fingers as opposed to using 15. This helps us reduce the analogue pins that are required and reduces the load on the ADC. Due to the lower load on the ADC, it also enables us to send data to the Game more frequently.

As explained in the microcontrollers sub-section (page 4), we are using a PSoC and Wemos. The Finger Bend data is calculated by the PSoC and sent to the Wemos so that the Wemos can send it to the Game. As soon as the PSoC is turned on it starts converting the flex sensor data using the ADC and saves the value in an array. As soon as the value is saved, the PSoC starts chooses a different flex sensor to repeat the process. Since the PSoC is a slave device to the Wemos for simplicity, at regular intervals, the Wemos requests the Finger Bend data from the PSoC. Then the PSoC sends back the data to the Wemos. This regular request is achieved through by using a timer interrupt in the Wemos. After some experimentation we found that the lowest period for this timer interrupt is 8 ms. This period is a result of the time taken to calculate the data from all the bend sensors and the data transfer delay. Even with the additional time taken to send this data to the Game device via WiFi, the delay is short enough for target frame rate of the VR Game.

Our finger bend tracking method also has some limitations as described below -

Adduction of the fingers cannot be measured From our current set-up of the flex sensors we can only measure the flexion and extension of the fingers as shown in figure 39. However, we cannot measure the lateral motion of the fingers (i.e. adduction) as shown in figure 40. This can be particularly limiting when it comes to measuring the orientation of the thumb.

Flex Sensors are not robust As mentioned in the previous sub-section, we have built the flex sensors ourselves and hence, there are very robust. This is because we have used regular tape to connect the components together. Details can on how to make the flex sensor more robust is discussed in the future works section (page 53).

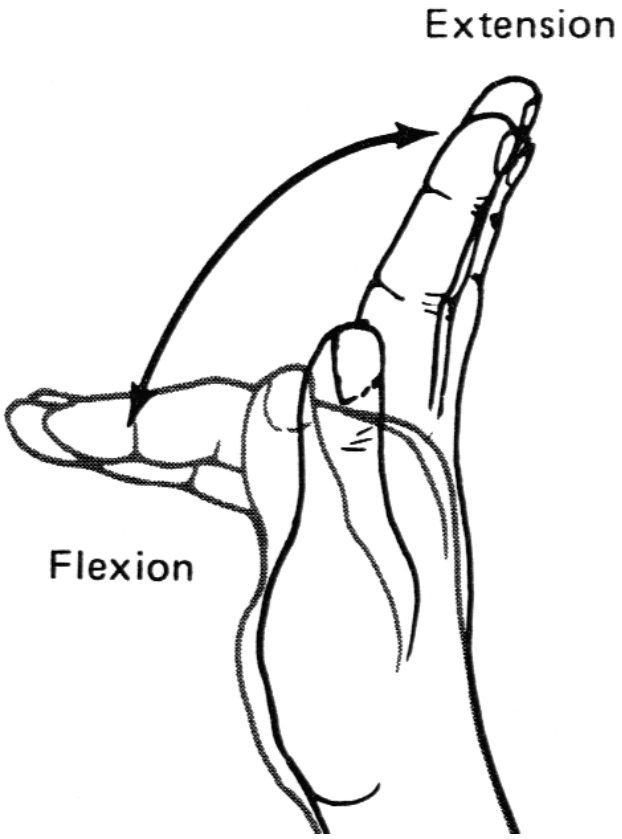


Figure 39: Vibration Motor [1]

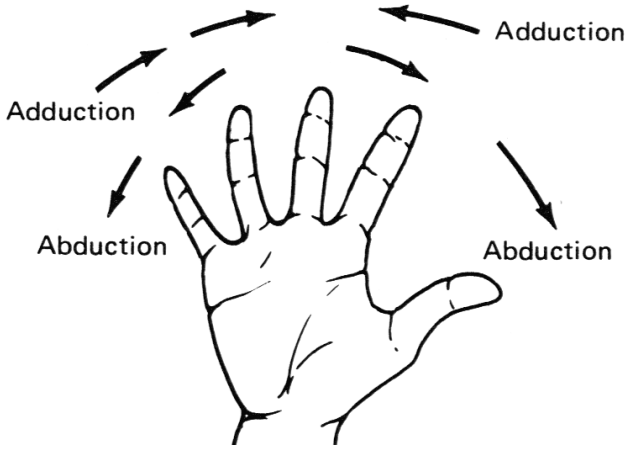


Figure 40: Vibration Motor [1]

7.6 Haptic Feedback

7.6.1 About the Actuator

Figure 41: Vibration Motor [45]

Figure 41 shows the vibration motor we chose for the project. These are used to provide Haptic Feedback to the user's hands whenever they touch something in the VR environment. These motors are typically used in mobile phones are providing vibrations for alarms and such. They are also used by Arduino hobbyists in their projects due to the low cost. The actuator is fairly simple to use, and it just turns on whenever a voltage is applied across the red and blue wires. The level of vibration provided by the motors depends on the voltage across the circuit and/or a duty cycle of Pulse Width Modulation (PWM) signal. [45]

Due to financial and time constraints of the project we chose to use only 5 Vibration Motors for the project. Each of them is attached to the finger tips of the Glove. This results in the user only the getting the sensation of touch on the fingertips instead of on their entire hand, in the ideal case. Since the Grip Simulation was the main highlight of the project and previous Final Year Projects have already worked on haptic feedback [50], we chose not to spend too much time on it and settle for something simple and functional, even though it is very limited.

7.6.2 Circuit Diagrams

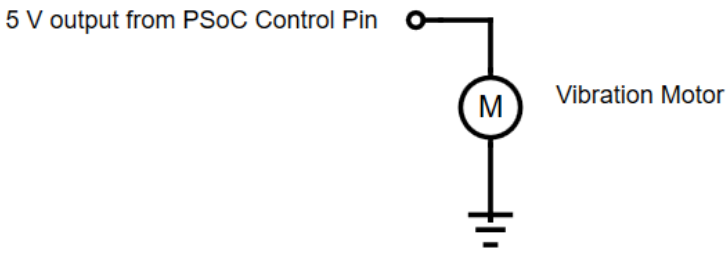


Figure 42: Basic Haptic Motor Control Circuit

Figure 42 shows the most basic circuit for using the vibration motors. The PSoC pin connected to the positive end (red wire) of the motor and the negative end (blue wire) is grounded. We decided the control the level of vibration of the motor using PWM outputs from the PSoC. This is because this gives more fine tuned control of the vibration from the motors than changing the voltage. Changing the duty cycle of a PWM signal, especially in a PSoC which is built-in easy-to-use PWM modules, is significantly easier to change than the voltage across the motor. This also allows the vibration level to be changed dynamically during the running of the game. This means that the vibration level can be changed depending on the type of object that is being touched in the VR environment.

However, the above circuit runs into certain problems. The most significant being that of high current draw. At 100 percent duty cycle, 5 V output, the vibration motor draws 100 mA of current from the circuit [45]. When 5 of these motors are used by the PSoC, it could potentially need to supply up to 500 mA. This is significantly above the current usage limits of the PSoC and high likely to get damaged. Hence, the circuit in figure 43 is adopted to make the circuit safer for the PSoC.

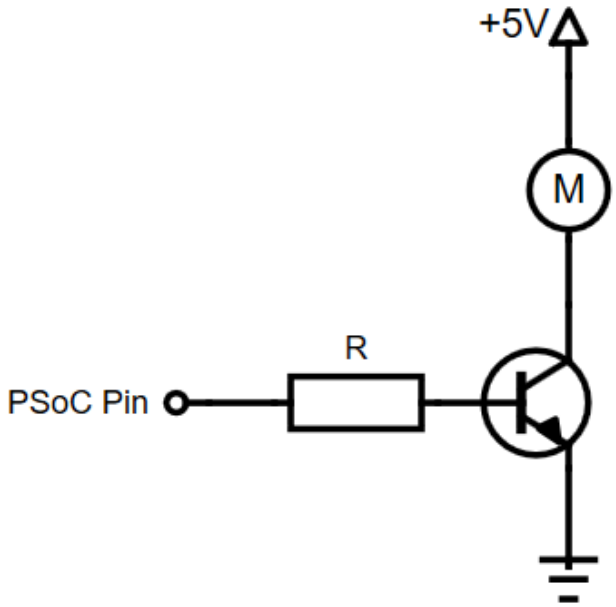


Figure 43: Haptic Motor Circuit with NPN Transistor

Figure 43 shows a simple NPN transistor being used to control the vibration motor. The vibration motor is placed on the collector of the transistor and the emitter is grounded. The base of the transistor is connected to a resistor R and the PSoC Pin, which has been assigned to control the motor output. Resistor R (typically around 4.7 kΩ) prevents high current draw from the PSoC pin. Whenever the output from the PSoC pin is high (which means it supply 5V as opposed to 0V which means low), the transistor switches on and the current passed through the motor. In this way, the motor is still controlled by the PWM signal that is outputted by the PSoC without the high current draw as seen by the circuit in figure 42

Since this circuit needs to be fitted on a human hand, we tried to reduce the number of circuit components used. On this circuit we decided to use a transistor chip (which contains upto 8 transistors in one chip) as opposed to 5 copies of the one shown in figure 43. The chip we choose was the ULN2803a. This is a popular and cheap transistor chip and used a wide variety of applications. The circuit for this chip can be found in figure 44.

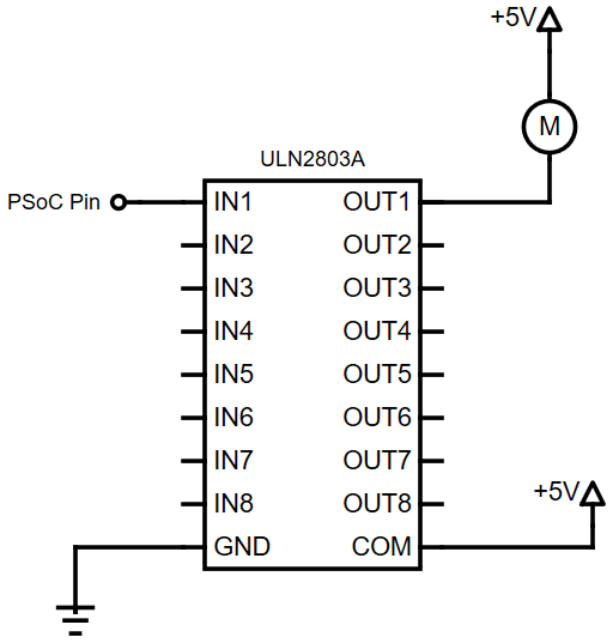


Figure 44: Haptic Motor Circuit with Transistor Chip

Figure 44 shows the ULN2803a being used for controlling one vibration motor. The remaining 4 are not being shown for the sake of simplicity but can be easily accommodated for. The COM and GND pins on the chip are used to supply power and ground the chip respectively. Between each IN and OUT pin of the chip is a transistor circuit as shown in figure 45.

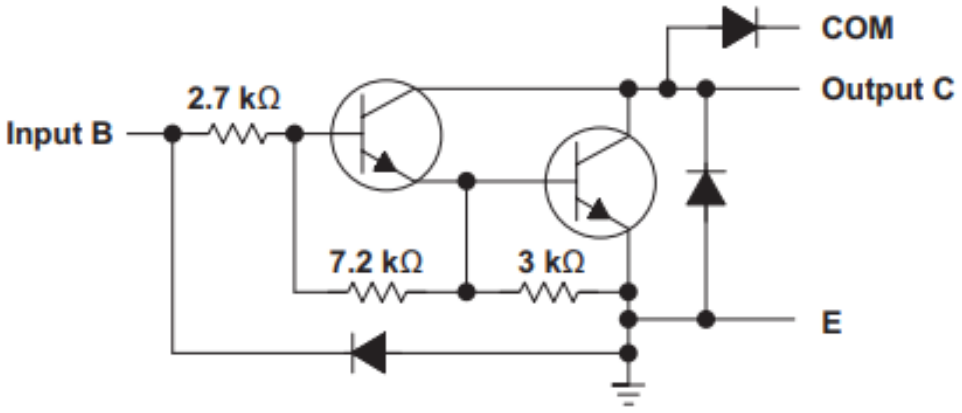


Figure 45: Transistor Circuit Inside ULN2803a [68]

The extra components shown in this figure prevents to the circuit in the case of numerous edge cases and faulty wiring, which are discussed in the chip's datasheet [68].

7.6.3 Test Results

As mentioned before, the Haptic Feedback vibration motor is controlled by a PWM signal sent out from the PSoC. The PSoC board has built-in PWM modules. The PWM module used has a resolution of 8-bit and and run by a clock of 100 kHz. Faster clocks were attempted by the produced signal was too fast to

be registered by the vibration motor. Given the configuration of the PWM module, figure 46, figure 47 and figure 48 shows the PWM signal being outputted at different compare values.

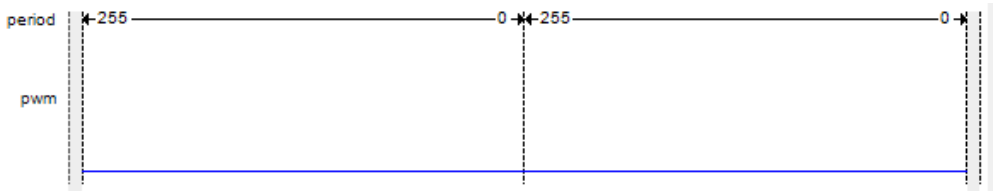


Figure 46: PWM Signal at Compare Value = 0

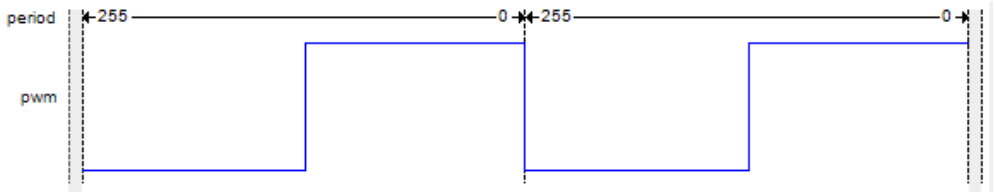


Figure 47: PWM Signal at Compare Value = 127

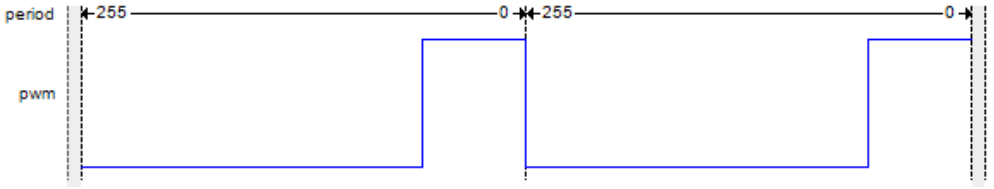


Figure 48: PWM Signal at Compare Value = 60

After some tests we found that a minimum compare value of 30 is required to turn on the vibration motor on at all. Below a value of 30, there is not vibration at all. After some experimentation, we picked 60 to be a good value to simulating touch. Future work involving a more subtle feedback system is discussed in the Future Works section (page 53).

7.7 Grip Simulation

7.7.1 Mechanical Description and 3D Modelling

This system is composed of multiple 3D printed parts specifically designed to fit the glove along with actuators and movable tapes. The goal was to design the parts as lightweight and miniature in size as possible, so as to provide a good level of comfort for the User while using the Glove. This brought forth several challenges to overcome during the design and testing process. These challenges and their respective outcomes are listed below for each of the 3D components and actuators. Included in each discussion are previous components that we tested but were not implemented in the final Glove due to various shortcomings. The complete system fitted to one finger is shown in figure 49.

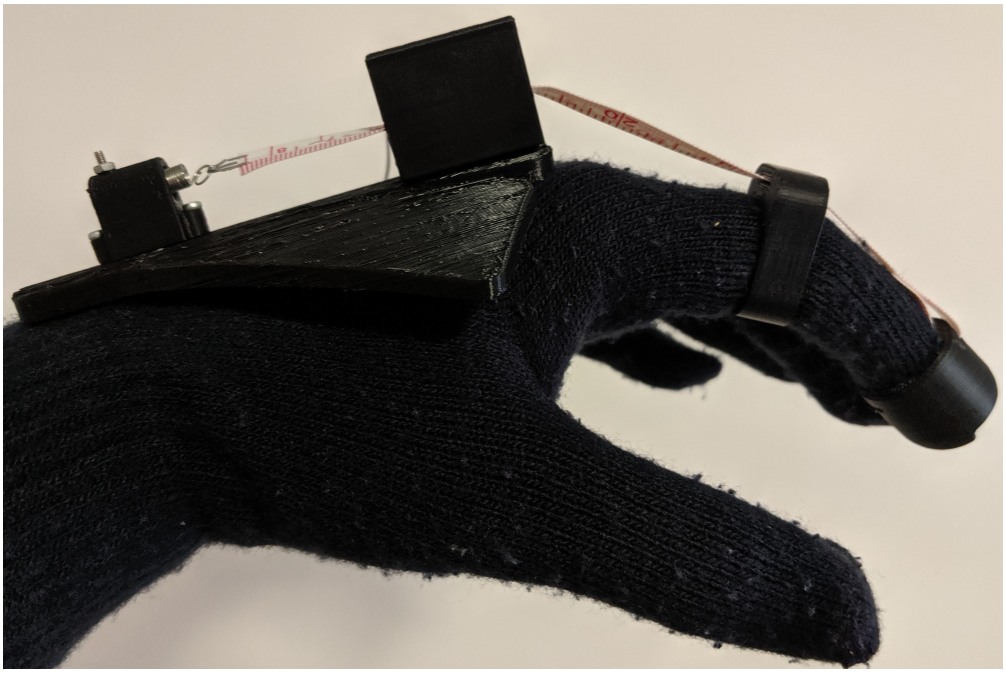


Figure 49: Grip Simulation system on one finger

All components were designed on Solidworks 2017 [38], the printing software used was Ultimaker Cura 4.0 [40], and the 3D printer used was Creality Ender 3 [18].

Base The largest component of the glove is the Base as pictured below in figure 50. The Base is fixed in place over the Glove so as to house most of the components including the control and power systems. We designed it around the dimensions of an adult male hand with the aim of occupying as much space as possible. The Base also has a natural bend on the vertical axis similar to that of a human hand. A small section of the Base protrudes on to the User's wrist. This is to allow for the Base to be securely fastened to the User's arm by using a wrist band to prevent the Glove from slipping off during operation.

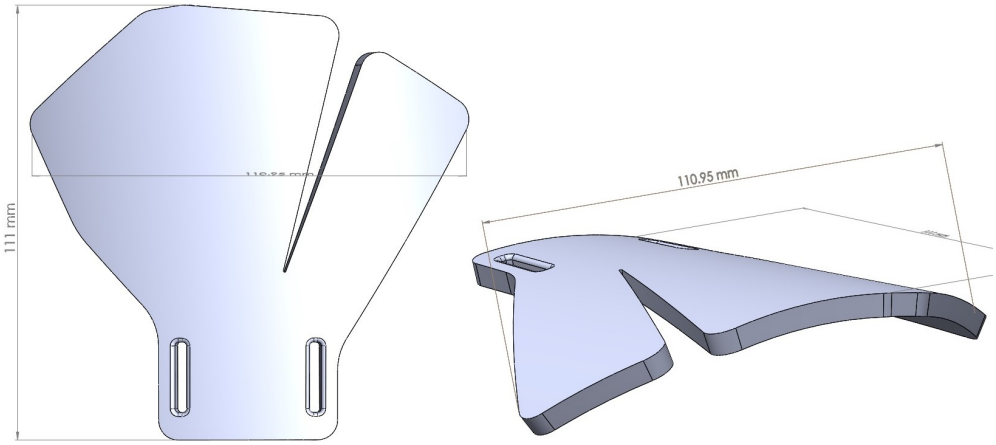


Figure 50: The top view (left) and side view [right] of the Base modelled in Solidworks

All other 3D components were designed around the available space on the Base, with the aim of fitting all of the components which make up the Grip Simulation system and the electronics lie flat on the Base.

However, this design was later found to be impractical as the Grip Simulation components occupied more of the available space than anticipated. On top of this, the use of two microcontrollers including the larger form size PSoC, did not leave enough space for every component to be placed on the Base. Hence, for the final design the electronics were placed on a second layer above the Grip Simulation components. M2 sized screws and bolts are used for fastening the components and the second layer's legs on to the Base. The Grip Simulation system was also cut down from five working fingers to two (fore-finger and thumb) to further reduce the weight on the User's hand and to minimise unnecessary work, as demonstrating the system with two fingers in its prototype stage is enough to demonstrate a working idea and implementation similar in principal to demonstrating all five fingers.

While all the other 3D printed components were printed with PLA plastic, the Base was printed with TPU flexible plastic. This plastic has the unique property of being very flexible and also being stronger than PLA. This was decided as a good approach since all other components will put a large amount of stress on the Base. Being flexible allows the Glove to naturally bend according to the User's hand as well.

Spring Holder Grip Simulation works by moving a tape back and forth over the User's fingers as the User manoeuvres their hand. When the hand is relaxed, the tape needs to retract back to the rest position. The simplest method in which this can be done is by fitting an extension spring to the back of the tape which will automatically retract back when the hand is relaxed. This is the approach that we implemented in this project. For the spring to retract back to a fixed position, the Spring Holder as seen in figure 51 was designed to both house and hold the spring in a fixed position. The Spring Holder is secured to the base using M2 screws and the spring sits in place with another M2 screw. The spring is placed a height of 1cm above the Base.

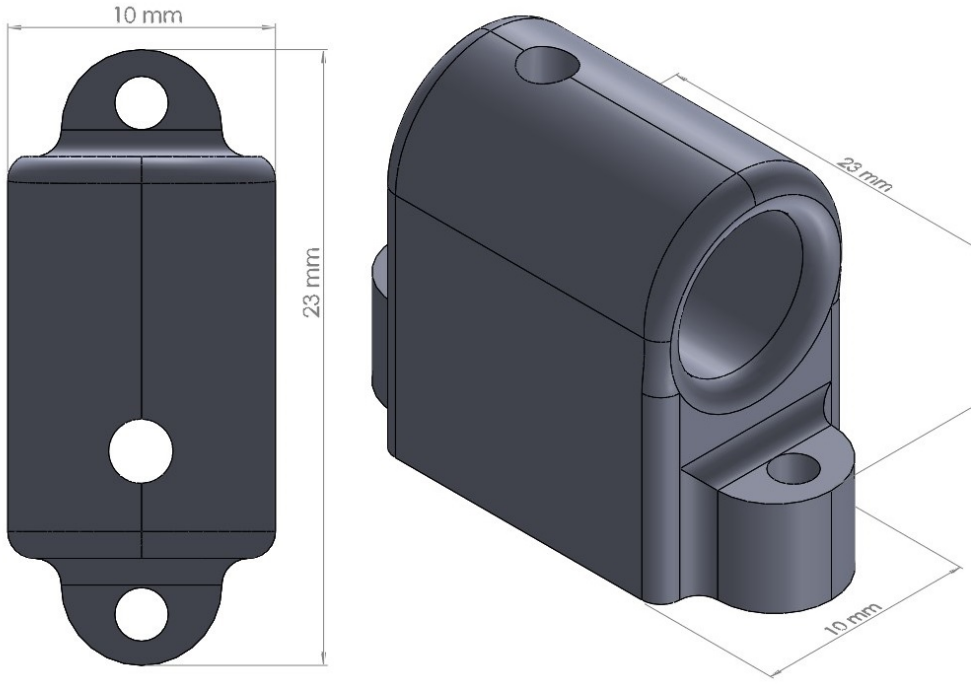


Figure 51: The top view (left) and side view [right] of the Spring Holder modelled in Solidworks

The Braking System - Motor Holder, Motor, and Hook To physically stop the User's hand around the boundaries of a virtual object when held requires a form of braking system to stop the tape from moving as the User attempts to further curl their fingers "through" the object. In a real-time setting, it isn't possible to exactly predict when the User will attempt to hold on to an object nor at which velocity, they make this attempt. Hence, the system must be implemented in such a way that it will react with virtually no delay as

soon as the User grips an object. The braking system was designed for this purpose in order to react within a fraction of a second of the User gripping an object in the VR game.

This system consists of three parts; Holder, Motor, and Hook as shown in figure 52, figure 53 and figure 54 respectively. Figure 55 shows all the components connected together to form the braking system. The Holder is a 3D printed part that houses both the Motor and Hook and is secured to the Base using M2 screws. It consists of three openings; to the left for placing the Motor and Hook inside the Holder, and two slits of 1mm height on the back and front for the tape to horizontally pass through.

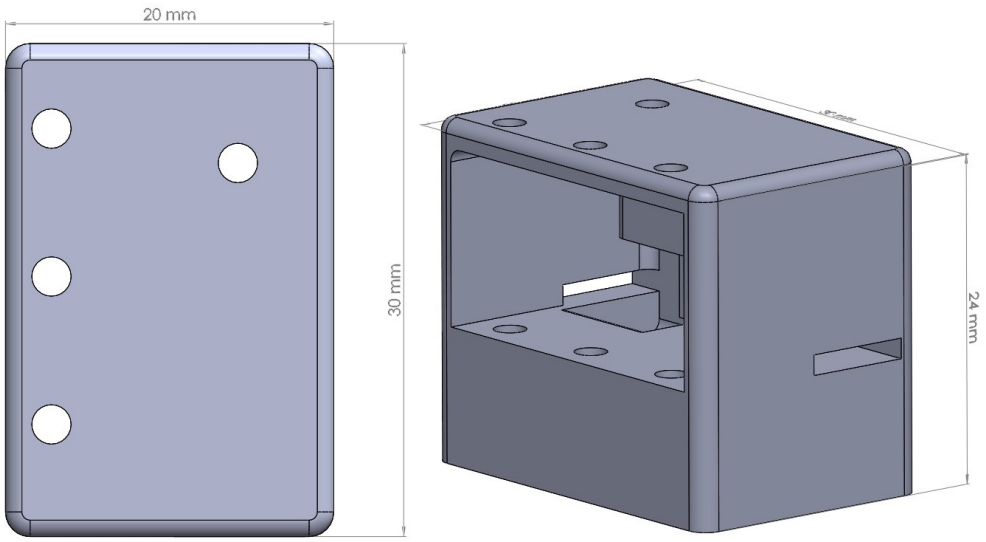


Figure 52: The top view (left) and side view [right) of the Spring Motor Holder modelled in Solidworks

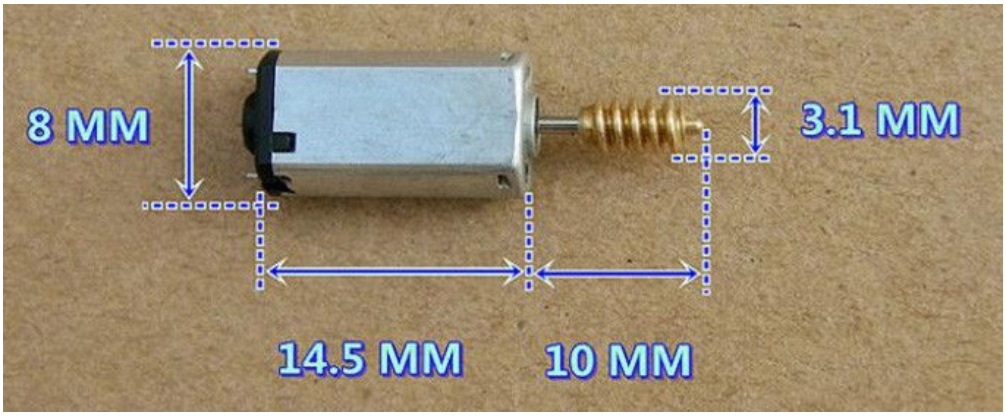


Figure 53: Mitsumi K20 Motor

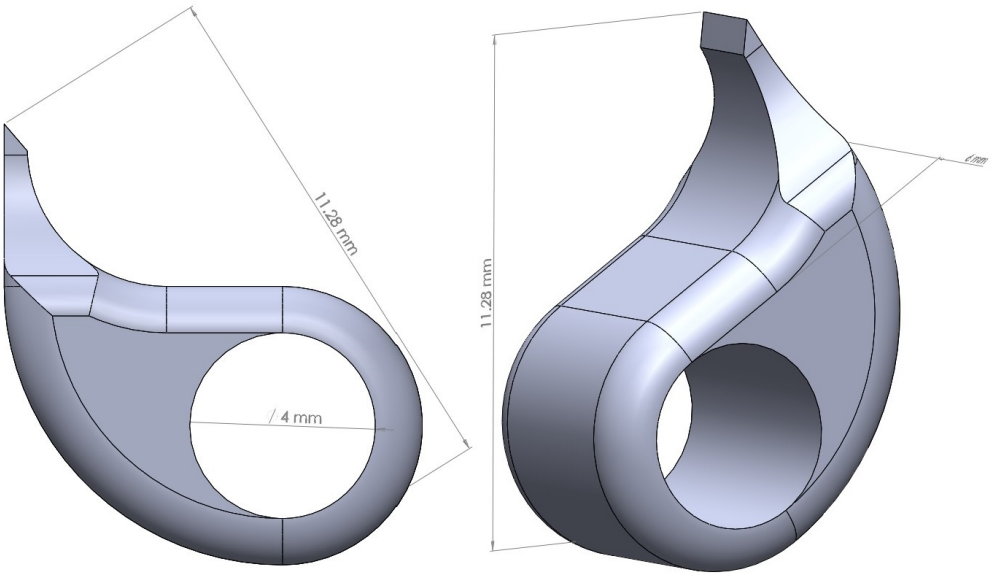


Figure 54: The top view (left) and side view [right) of the Spring Hook modelled in Solidworks

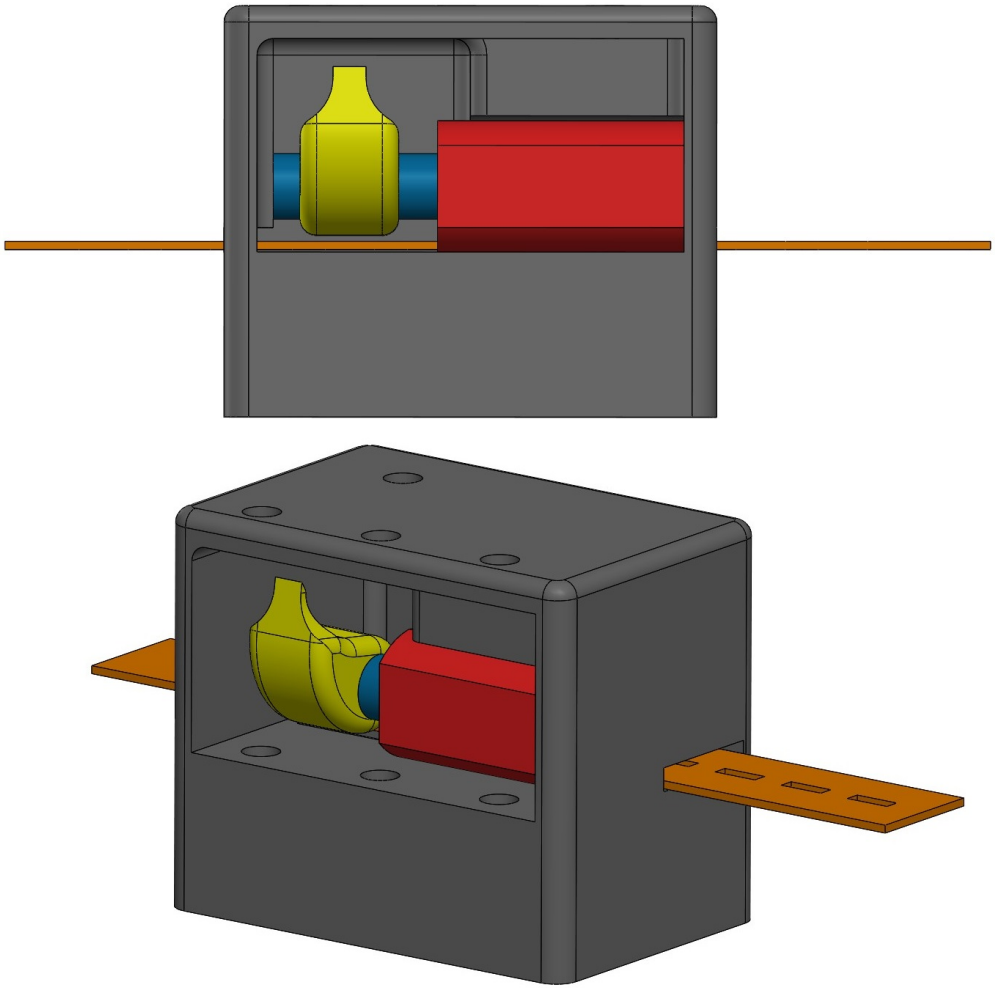


Figure 55: The braking system; the motor (red), motor shaft (blue), hook (yellow), holder (grey) and the tape (orange)

The Motor is a Mitsumi K20 dc motor achieving 33000rpm at 5V and consuming 60mA [24]. At 33000rpm the Motor could achieve a single revolution in 1.82ms. For the purposes of this project, the Motor is required to revolve a quarter revolution towards the moving tape and back from the tape when required. Hence, in the best-case scenario, the Motor would be able to respond to hook and unhook commands in 1.82/4 or 0.45ms, a time frame more than sufficient for the purposes of this project.

Finally, the Hook is a small 3D printed component that is fixed on to the Motor's shaft. The point of the hook was designed with the aim of being durable enough to withstand the force of the User curling their fingers. It was designed with a 2 mm width endpoint and printed with 100% infill. When the User grips a VR object, the Motor is powered on and the Hook revolves towards the tape. It attempts to push through one of the holes cut in the tape thereby stopping the tape from further moving outwards. While the User stays holding on to the VR object, the Motor is continuously powered with the Hook constantly pushing downwards towards the Base. Figure 56 shows an illustration of the forces acting on the tape when the User first grips an object at time=0 and the Hook latching on to the tape at time=0.45 ms. When the User releases the object, the Motor is powered in reverse, and the Hook moves in the opposite direction until it is no longer latching on to the tape. The Motor is powered off after 3 seconds of the unhook command to reduce the overall power consumption and wear over time of the Motor.

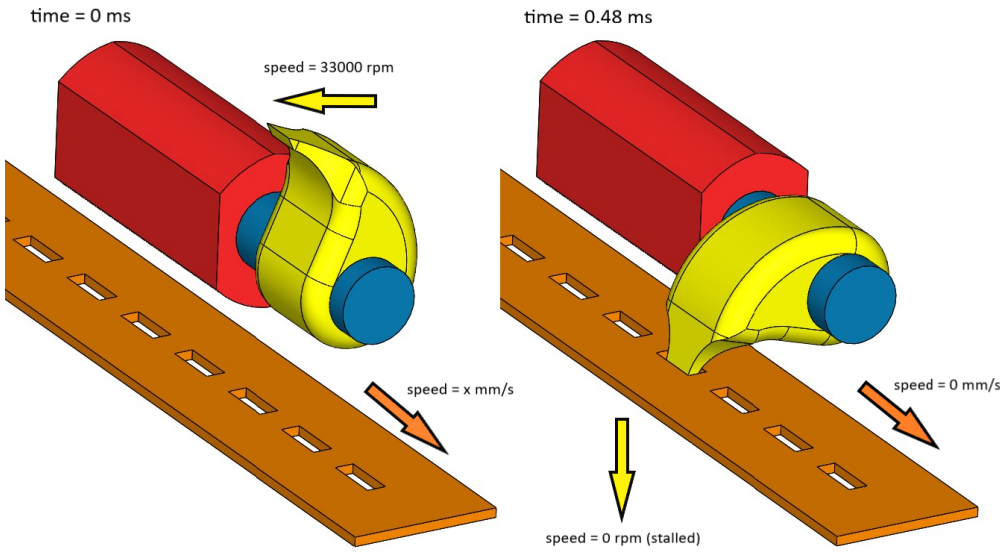


Figure 56: The forces acting on the bodies with the colour of the arrow corresponding to the body

Prior to the above method of the braking system, and earlier method involved the use of an electromagnet and magnetic rod to stop the moving tape. This method attempts to achieve braking by firing the magnetic rod through one of the holes of the moving tape when the electromagnet is powered on. This implementation including its own Holder is shown in figure 57. For the most part, this electromagnet was able to achieve braking when needed similar to the Motor. The main disadvantage of this method, however, was overheating.

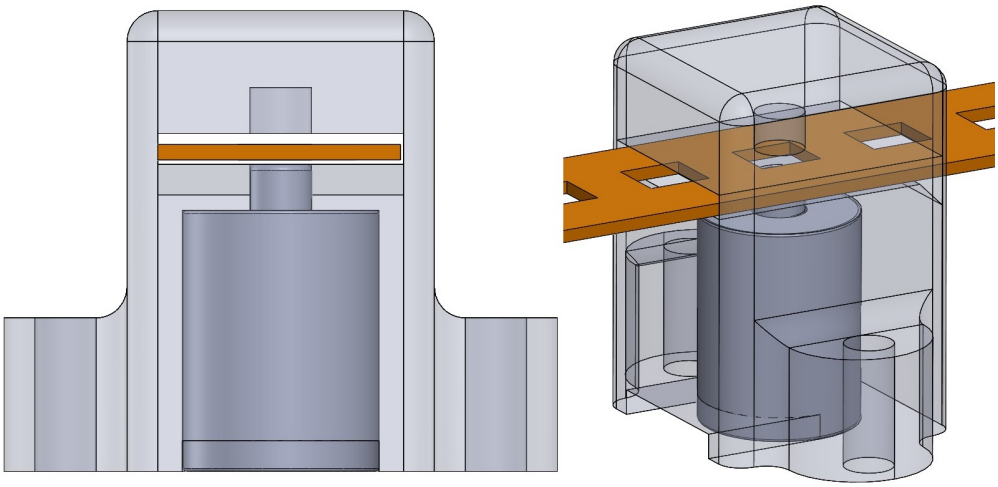


Figure 57: The front view (left) and side view [right) of the Electromagnet Holder (transparent) modelled in Solidworks (moving tape in orange and the electromagnet in grey)

When powered on, the electromagnet requires 1A at 5V supply to rapidly fire the rod outwards. When supplied with lower power, the rod does not move with a velocity fast enough to not perceive delay and often the User's hand would stop "within" the VR object when gripped. Nevertheless, whether the electromagnet is supplied enough power or less power, it would suffer from overheating when powered on for more than 1 second. Heating was expected as one of the side effects of an electromagnet is Ohmic heating [66]. What wasn't apparent was the amount of time taken to significantly heat the electromagnet when first powered on. The electromagnet would accumulate heat in such a small timeframe that continuous gripping and releasing

of objects in the game would overheat the component to the point that the plastic interior would deform. The hole in the electromagnet is measured at 0.5mm wider than the 2mm magnetic rod which allowed the rod to move freely within the electromagnet. However, due to overheating, the plastic would deform enough to reduce this gap making free movement of the rod impossible. The overheating and deformation of the component was enough to conclude this method is too hazardous for use in the project and so was replaced with the Motor implementation.

Middle Support It is difficult to make certain that the tape moving over the User's finger will remain perfectly inline with the finger at all time as the User curls and uncurls their fingers. It is likely that as the User curls their finger and/or grips a VR object, the tape will slip off the sides of the finger. This will result in a failed system. To eliminate this issue, the Middle Support is a 3D printed component that is placed through the User's finger and allows the tape to passthrough the top of the Support as seen in figure 58. Now, as the User curls/uncurls their fingers, the tape is held more securely above the finger making it less likely that the system would fail.

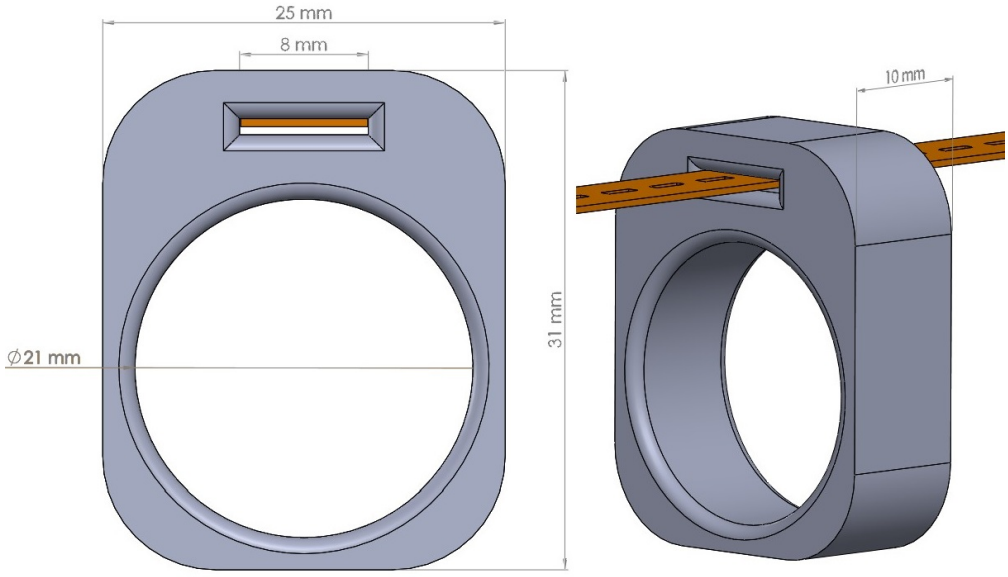


Figure 58: The front view (left) and side view [right] of the Middle Support modelled in Solidworks (moving tape in orange)

The biggest limitation of the Middle Support and by extension the Grip Simulation system itself is the ring size of the Support. The component used for this project was modelled around an adult male hand. This means that while the Grip Simulation should work just as well for other adults of similar hand size, it may not work for a majority of women and children whose hands are typically smaller or thinner in comparison. As a proof of concept however, this implementation is sufficient. A more universal fit is discussed in detail in the future works section.

Finger Mount The final component of the system is placed over the tip of the User's finger. The other end of the moving tape is attached to this component. To put the system in to retrospect, if the moving tape is considered the muscle, then the Spring Holder and Finger Mount make up the bones that the muscle is fixed between. It also houses the haptic feedback motor which is placed below the finger. The Mount including the haptic motor is shown in figure 59. Like the Middle Support component, the Finger Mount suffers the same limitation in design. A more universal fit for the Mount is discussed in future works.

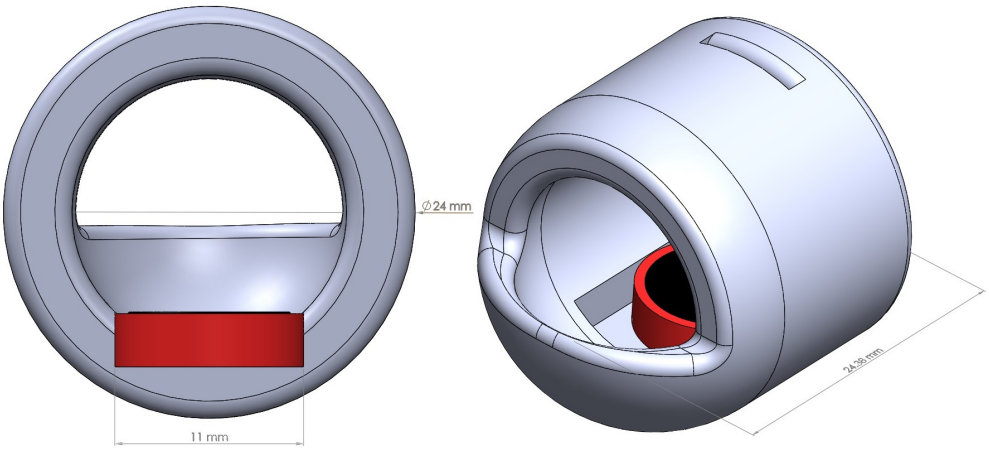


Figure 59: The back view (left) and side view [right) of the Finger Mount modelled in Solidworks (Haptic Motor in red)

7.7.2 Circuit Diagrams

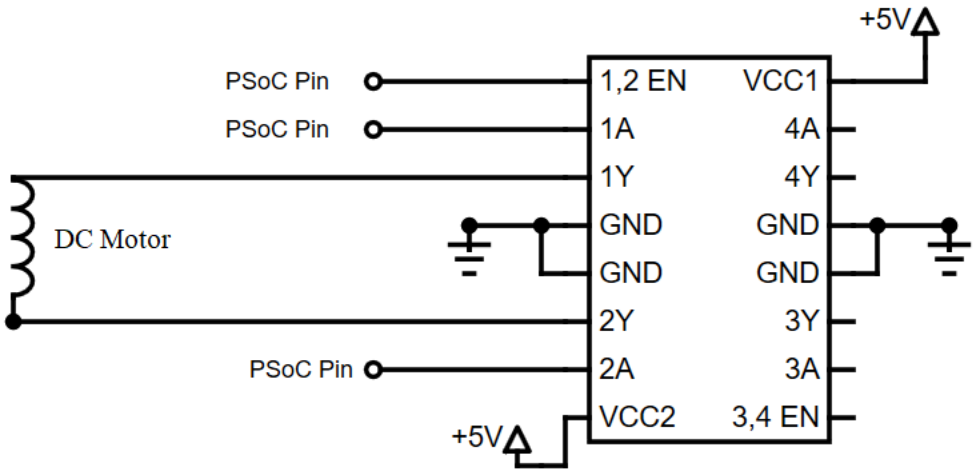


Figure 60: Grip Simulation Circuit (using Motors)

As seen in figure 60, an H-bridge is used to control the direction of current that passes through the motor. The direction of the current determines the direction of the rotation of the motor.

7.7.3 Final Design and Test Results

Due to time constraints, the Grip Simulation was only made for usage for 2 fingers. These fingers being the thumb and index finger. After testing we saw that the Grip Simulation is able to effectively stop the user's fingers at the correct positions. Possible avenues of improvement in this sub-system is discussed in the Future Works section (page 53).

7.8 Position Mapping

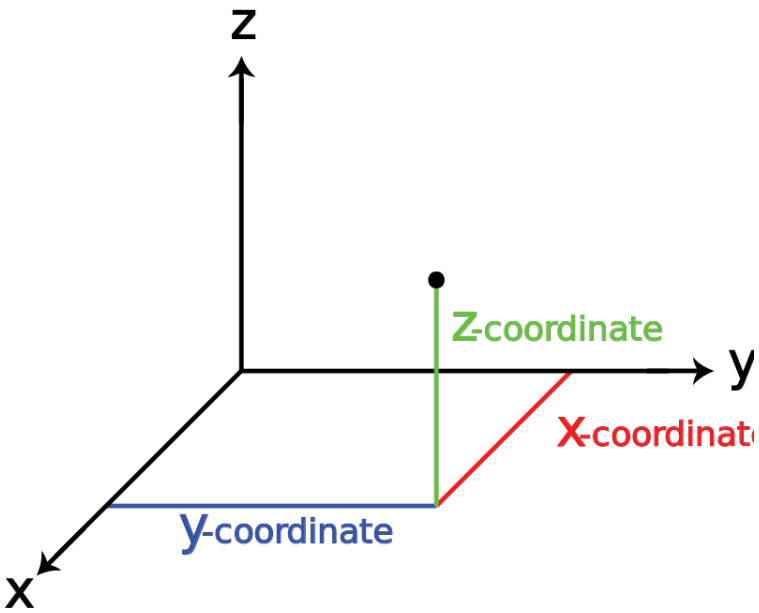


Figure 61: Position Axes [49]

7.8.1 Requirements

3D position measurement The Position measurement system must be able to measure the position of the Glove in 3D space. This means the system needs to output displacement of the Glove in x,y and z directions.

Accuracy The accuracy of the position measurement system needs to high enough to give the user the illusion of interacting with the VR environment. For our minimum requirements, we set the maximum tolerance for error to be less than 5 cm.

High Speed Measurement The position measurement system needs to be able to output data at high rate so the user's movements can be mapped in real time. Considering our target frame rate for the VR Game (which is 30), our target position measurement frequency should be at least 60 Hz, while still meeting the other requirements.

Range The user's hand is unlikely to move more than 1-2 m (in all 3 directions) during the gaming session. For this reason, the position measurement system should maintain all other requirements for between this range of movement. Maintaining these requirements for a bigger range is beyond the scope of the project.

7.8.2 Methodologies Considered

Accelerometers The simplest method of measurement the position of the Glove would be use the accelerometer data. We are already using the accelerometer data for Orientation Mapping (as seen in page 14).The accelerometer gives the acceleration of the Glove in 3D space. Integrating this data over time will give us the velocity of the Glove. Integrating this velocity data over time again we give us the displacement of the Glove. This can be seen in the following equations -

$$Velocity_X = \int_0^t Acceleration_X dt \tag{11}$$

$$Displacement_X = \int_0^t Velocity_X dt \tag{12}$$

where,
t = the time (in seconds) since the gaming session has started
$Acceleration_X$ = the acceleration of the Glove in the x-direction that is measured by the accelerometer at time t
$Velocity_X$ = the velocity of the Glove in the x-direction at time t
$Displacement_X$ = the relative displacement of the Glove at time t

However, this a major problem with this methodology. Real world sensors always have some error. Hence, when their values are integrated the calculated value accumulates drift (as seen in Orientation Tracking sub-section - page 14). . Hence, the calculated velocity accumulates drift. However, since for calculating displacement, the velocity is integrated, the displacement value accumulates drift even faster. When tested, the calculated displacement value, accumulated drift of several meters within a few seconds. Even after filtration it is possible to remove this drift completely.

Additionally, this method only allows us to measure displacement (relative the initial position of the Glove at the beginning of the Game session) as opposed to absolute position. Hence, using accelerometers (especially using low-cost accelerometers) it is possible to measure the position of the Glove within the required accuracy level.

Figure 62: Google Maps Logo [48]

GPS Global Positioning System (GPS) is another common method for getting position. The accuracy of GPS is not high enough for our application (GPS accuracy around 20-40 cm) and the measurement frequency would also very low (between 1-5 Hz) [53]. Additionally, GPS does not work very well in indoor conditions (which where the game will typically be played) [69].

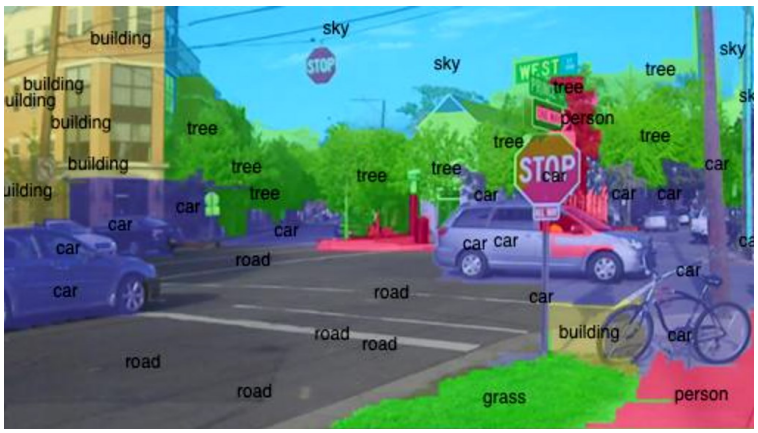


Figure 63: Example of Computer Vision [3]

Computer Vision Computer Vision is very popular method for position mapping. It is heavily used in current VR consoles and controllers. One possible method would be to use the camera on the mobile phone (that is running the VR Game) for computer vision (let's call this configuration **PhoneCamera**). Another method would to use an external camera that acts as a base station. This base station will capture the images of the Glove, process it to measure position using Computer Vision and send the data over WiFi to the mobile phone VR game (let's call this configuration **BaseStationCamera**). For these configurations we run into the following problems -

Poor performance in certain conditions Computer Vision applications do not tend work well in low light conditions. Although filters exist for these applications to make it easier, trying to get it work in low light conditions will increase the workload for the project.

Cannot track Glove if not within line of sight If the Glove is not in front of a camera, it will not possible for the computer vision application to determine its position. This can happen while the user is moving or an obstacle between the Glove and the camera.

Power Consumption problems for the Phone (PhoneCamera only) The mobile phone device already has a lot of processing happening with rendering the VR Game and acting as a WiFi Hotspot (more details can be found in Data Communication sub-section - page 10). Additional real time computer vision processing will cause a significant drop in the battery life of the mobile phone during a gaming session.

Additional Orientation Tracking (PhoneCamera only) The mobile phone will not be stationary during the gaming session. For this reason, if the phone's camera is used, we would need to also compensate for the phone's movements. This can be done using by performing Orientation Tracking in the 3-axes in real time using the phone's built-in gyroscope and accelerometer. However, this places an additional burden for coding and power consumption on the mobile phone.

Time constraints Neither of us have ever learned Computer Vision and/or worked on projects involving the subject. Since position mapping is not the most important aspect for our proof of concept Glove, it is not feasible for us to learn the technique from scratch and implement it well enough for our application.

Figure 64: Ultrasonic Sensor - HC-SR04 [33]

Ultrasound Ultrasound is any sound wave with a frequency above that of human hearing (above 20 kHz). Ultrasound is commonly used to proximity measurement and obstacle detection [59]. However, this methodology is very complicated to implement for our purposes since we need a detailed map of the environment for the program to be able to measure the position of the hand in real time.

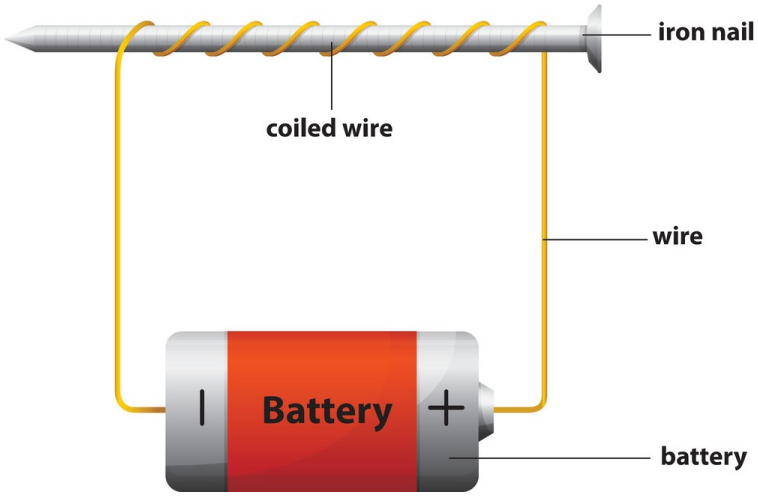


Figure 65: Simple Electromagnet [43]

Magnetometer Magnetometers also play an important role in many position and orientation mapping applications. This is achieved by using a magnetometer and a magnet that acts as a stationary base station. Since the magnet field measured by the magnetometer will change depending on the distance away from the magnet. In this way, the relative distance from the magnetometer can be measured. This allows to measure distance and orientation (and maybe replace or reinforce our current Orientation measurement system) in fast and simple way [64]. This will enable us to meet the requirements we had set out for the position measurement system.

However, one of the biggest problems with this method is electromagnetic interference from other devices and also the Earth's magnetic field. This can be solved by replacing the magnet in the base station with an electromagnet of specific frequency. An electromagnet can be created a looping a certain amount current carrying wire around an iron (non-magnetised) core, as seen in figure 65. This electromagnet can be made using an AC current of the specific frequency or a current at is pulsed at a specific frequency. The magnetometer data can filtered to calculate the magnetic field caused by the electromagnet base station. [54]

The distance can be measured using the formula below [52]-

$$r = \sqrt[3]{\frac{\mu_0 N I r_0^3}{4B}} \tag{13}$$

where,
r = the distance between the magnet and the magnetometer, measured in meters
μ_0 = permeability constant ($4\pi \times 10^{-7}$)
N = the number of turns on the electromagnet
I = the current passed in the electromagnet, measured in Amps
r_0 = the radius of the electromagnet (i.e. the distance between the magnetic core and the current carrying wire), measured in meters
B = magnetic flux density, measured by the magnetometer, is measured in Tesla

The position measurement system proposed is relatively simple to implement and avoids some of issues discussed so far such as the magnet does need to be within line of sight of the magnetometer for the measurement to work. Additionally, 2 extra base stations can be added to the current proposed system. Each base station electromagnet can have a different frequency. Using data from all three magnetic fields, we use triangulation to make the position measurement more accurate. [52]

7.8.3 Final Results

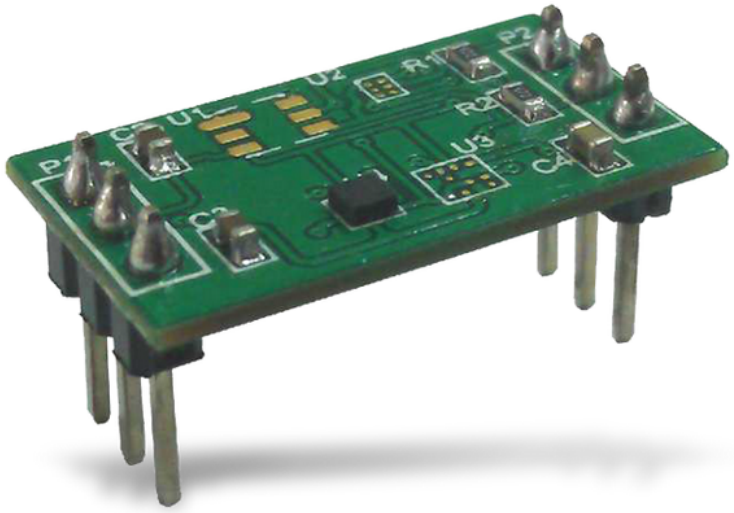


Figure 66: MMC3416PJ-B Magnetometer [21]

After the research we conducted, we decided to try out the position measurement using AC magnetic fields and magnetometers since it seemed the simplest to implement within our time constraints. We set the target frequency of the electromagnet, that acts as our base station, to be around 30-40 Hz. To measure a magnetic field of this frequency, the digital magnetometer we picked needed to be able measure the magnetic field at a higher frequency. This will enable us to measure and filter the magnetic field from the base station more accurately. However, most digital magnetometers don't sample at high frequencies. They typically measure upto around 40 Hz, which is not high enough. So, we picked the MMC3416PJ-B Magnetometer, as seen in figure 66. This sensor allows us to measure the magnetic field from 125 Hz to upto 800 Hz.

However, when we attempted to interface it with the Wemos board, we could not get it working. The sensor kept giving us overflow values for the magnetic field measurements even though there was magnetic field. Since this was not a popular sensor, we could not find any help for it on-line, besides with the datasheet, which does not come with sample code.

We came into this problem fairly late into the final semester for the project. It was too late to try, find and test a different sensor and still be able to implement the system. This was especially difficult since there many other features for the Glove that still needed to be implemented. Since the main focus of the project was the Grip Simulation and not the Position Mapping, we decided to scale back the project and cut this feature for the proof of concept prototype. This has also been addressed in out Future Works section of the report (page 53).

7.9 Power Supply

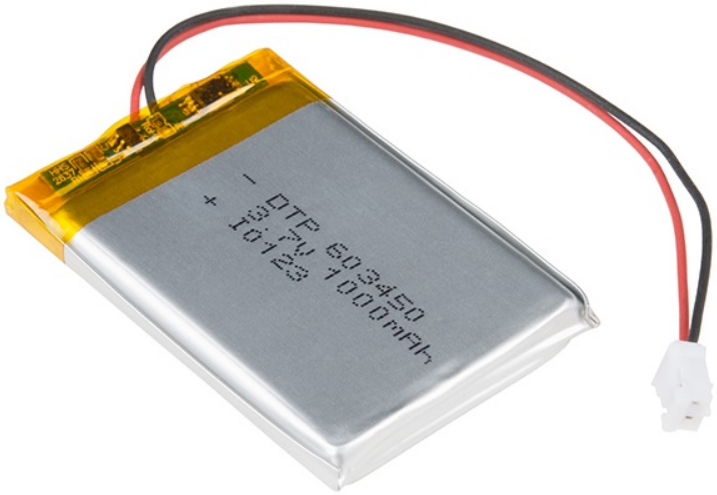


Figure 67: Lithium Ion Battery [39]

Figure 67 shows the 500 mAh (milli Amp hour) Lithium Ion Battery that we are using as the power source for the Glove. However, the voltage output for this battery is only 3.7V and (as seen in the previous circuit diagrams) the components in the Glove require a power supply of 5V. To mitigate this problem, we use a boost converter.

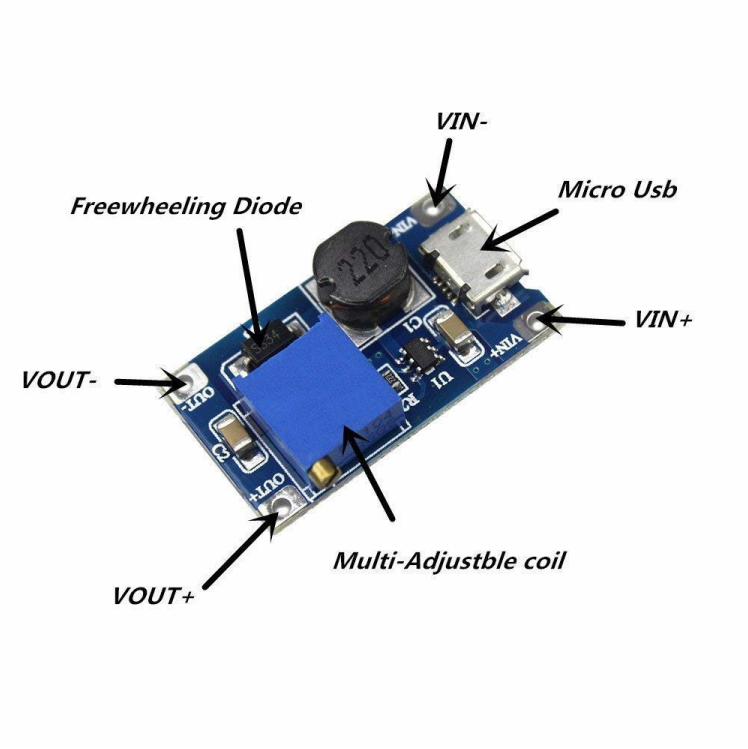


Figure 68: Boost Converter - MT3608 [23]

Figure 68 shows the boost converter that is used for the project. A boost converter is a circuit component that takes in a smaller input voltage (in this case 3.7V) and output a larger voltage (5V). The circuit diagram

for the converter is shown in figure 69

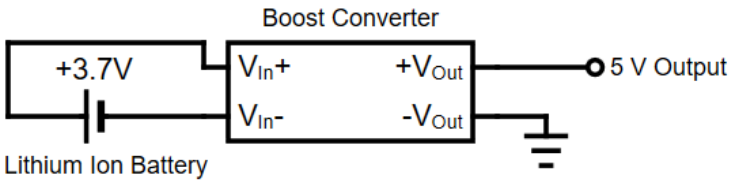


Figure 69: Power Supply Circuit with Boost Converter

7.10 Full System

7.10.1 Orientation Mapping

For the 3D hand model, we have turned off all the built-in Unity engine physics. This is so we have complete control over its movements using 3-axes vector transformations. The orientation sensors are sampled at 4 ms intervals (250 Hz). The orientation data is then filtered and calculated. Afterwards this data is sent to the Game using WiFi.

Once the data is received by the Game, a script that runs every frame, updates by the orientation of the 3D hand model. This is done by using the 3-axes rotation transformation, which is a built-in feature of the game. In this case, the entire 3D hand model is rotated in response to the orientation data.

7.10.2 Finger Bend Tracking

In our application, the ADC value measured by ADC values measured for the flex sensors can range between 0-200. At the beginning of the game session, we calibrate the ADC measurements against the initial values measured by the ADC. We are assuming at the beginning of the game session, the user's fingers are parallel to the palm. Hence, we are only measuring the relative finger bend rotation. Every 8 ms intervals, the PSoC sends the data to the Wemos and the Wemos sends the data to the Unity Game.

The maximum rotation a finger is always below 90°. Hence, these received data is mapped into degrees by multiplying it with some constant. This constant is slightly different for each finger joint.

When this data is received, the individual finger joints on the 3D hand model is rotated according to the received data. The rotation on the top finger joint for each finger interpolated using the data from the lower finger joints. In this way, real time finger bend tracking is achieved.

7.10.3 Haptic Feedback and Grip Simulation

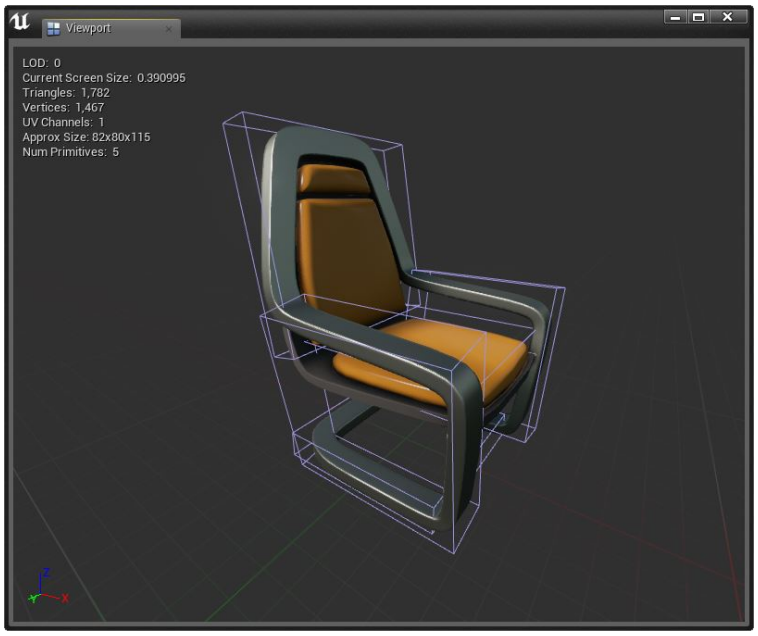


Figure 70: An example of Collision Box around a chair [41]

Collision Boxes are used to check collision between objects in a Game environment. An example of a collision box is the transparent box formed around the chair as seen in figure 70. By placing individual collision boxes at individual joints of the fingers and on the palm. Using this method we can find out when the individual parts of 3D hand model is interacting with another object. Using the information, we can find out when to active the haptic vibration motors and grip simulation. This data is sent to the Glove whenever the state changes (i.e. when the user is supposed to be touching an object etc.).

8 Conclusions

For our final year project, we tasked ourselves with constructing a simple, affordable, portable, and compact VR glove controller. This glove would highlight the novel implementation that we coined as Grip Simulation, which is the process that physically stops the User's fingers in the real world when they interact with a VR object in the virtual world. As a design requirement, this system should be able to provide an immersive experience for the User. The Grip Simulation system would be accompanied by other systems that would add several other controls for the User. These systems are; Haptic feedback, Finger Bend sensing, and Orientation Tracking.

In this report, we documented our implementation and results. We were able to showcase Haptic feedback, Finger Bend sensing, and Orientation Tracking to great success. As for Grip Simulation, after some setbacks we were able to construct a system that was quickly responsive to changes in the VR environment and mechanisms built with good durability. Though we limited this system to just two fingers, there is enough potential for this system such that it can be improved with time and resources.

9 Future Work

microcontrollers

Using SPI instead of I2C We are currently using an I2C Master Slave Circuit to send data between the Wemos and PSoC (more details can be found in page 4). By default, the clock speed for data transfer is 100 kHz [11]. This can be a potential area for bottleneck especially if more features are implemented in the current program. The I2C clock speed can support up to 3.4 MHz [11][56]. We suggest for future implementation Serial Peripheral Interface (SPI) is used instead since it supported by both microcontrollers. For the PSoC and Wemos the clock speed can be increased upto 16 MHz [57][10]. Due to the time constraints of the project and simplicity of I2C when compared to SPI, I2C was used in the current version.

Using a single microcontroller instead of 2 One of the major limitations of the project is that we ended up using 2 microcontrollers instead of 1. Although the data transfer speed can be increased, there are still other challenges. One of them is we have to maintain 2 separate code bases. We also need to write additional code whenever we need to send data between the PSoC and the VR Game. One possible way to solve the problem is to get the ESP8266 chip working with the PSoC board. Our personal recommendation for future improvement would be using PSoC 6 WiFi-BT Pioneer Kit (CY8CKIT-062-WiFi-BT). It has more enough GPIO pins, has built-in triple-axis gyroscope and accelerometers (which is used for Orientation Mapping as shown in page 14), and most importantly built-in WiFi capability. Additionally, it has removes the need for a boost converter since the board can be connected straight to a lithium ion battery [55] (more details can be found in page 49).

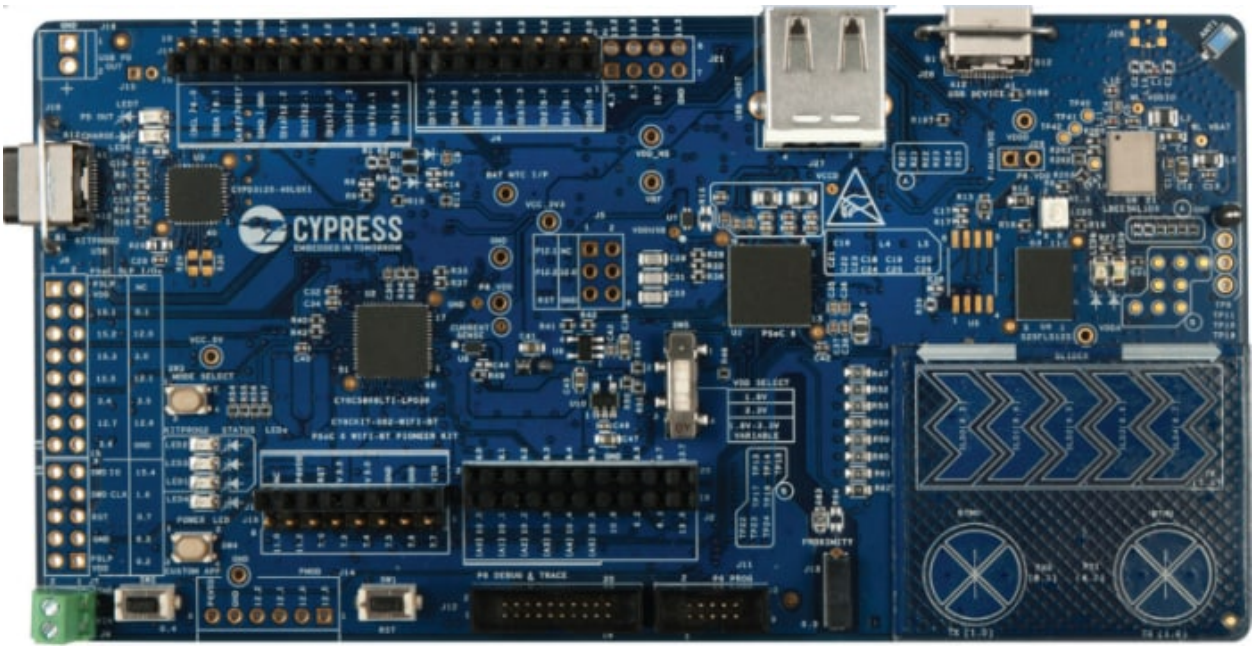


Figure 71: PSoC 6 WiFi-BT Pioneer Kit (CY8CKIT-062-WiFi-BT) [34]

VR Game

An actual game The Glove is made for primarily gaming purposes. The original plan was to build a small game to show off the Glove's capabilities. Due to time constraints we were unable to implement a simple game. Currently the Game side of the project is fairly limited with simple interaction of VR objects to act as a proof of concept for the Glove. In future, a small fully-fleshed out game can be created to showcase the Glove's features.

Orientation Tracking

Use MPU6000 instead of MPU6050 MPU6000 is a more powerful sensor than MPU6050. It is very commonly used by hobbyists as an Inertial Measurement Unit (IMU) in quad-copters. The major reason for this that instead using I2C for data transfer like the MPU6050, it uses SPI which is much faster. With this advantage it is possible to sample the gyroscope at 8 kHz as opposed to the 250 Hz which is used to sample the I2C in the current application. The MPU6000 also has a higher tolerance for vibrations [60]. The reason this sensor wasn't used was because I2C is much simpler to use than SPI and also, we meet the Orientation accuracy and update frequency requirements for our project. Although we did feel the high speed sampling was not necessary it still might be useful to have for high precision or heavy movement use of the VR Glove in future applications.

Absolute Yaw Estimation As mentioned in the Orientation Tracking sub-section (page 14), we could not calculate the absolute yaw of the Glove due to the limitations of the gyroscope and accelerometer. One of the most common ways to calculate absolute orientation is computer vision. The computer vision calculations can be done in the Game using the device's camera to track the hand. The challenges of using solely computer vision is discussed in the Position Mapping sub-section (page 44). One alternate is use computer vision only at the beginning of the game session to calculate the absolute orientation (hence, only need to face the camera at the Glove at the beginning). Since neither of us have any experience with computer vision implementation, we opted to not use it for small improvement as absolute yaw estimation.

Finger Bend Tracking

Make better flex sensor For finger bend tracking, we made our own flex sensors using velostat (more details can be found in its sub-section - page 20). We currently using tape to put the components of the flex sensor together. This is everyday tape and hence, a good, robust long-term solution for making a sensor. Instead of using regular tape, industrial strength tape or even laminating the components together can be done. This is to make the flex sensor more professional grade.

Better ADC An analogue-to-digital converter (ADC) is used in Finger Bend Tracking as explained in the Finger Bend Tracking sub-section (page 20). The latency and accuracy of this feature is largely dependent on the ADC. Hence, an ADC with a higher resolution (currently 12-bit) and faster processing time will helpful in using the Glove for high precision applications.

Haptic Feedback

Create more subtle feedback The current system is only calibrated to send only one kind of haptic feedback, i.e. the vibration motor will either be supplied a compare value of 0 or 60 for the PWM signal (more details can be found in the Haptic Feedback sub-section - page 31). This can be improved upon by allowing the user to interact with different types of objects in the game and more compare values can be calibrated for each type of object. This can be used to create a more subtle haptic feedback system and truly give the user the illusion of touching an object in the VR world.

Use more motors all over the user's hand Currently the vibration motors are only placed at the tips of the user's fingers. Hence, haptic feedback can only be sent to the tips of the fingers. More haptic feedback sensors can be placed at various areas on the hand such as between the joints on the fingers and a few on the palm. This will help create a more complete of illusion of touch for the user.

Reduce power consumption As just mentioned, we would like to add more vibration motors in the future. One possible limitation of doing this is that the power consumption of the glove circuit will increase significantly, since each motor consumes 100 mA, when at 100 percent duty cycle at 5 V (which is used currently to power the motor). However, at lower voltages it consumes lesser current (at, 4V current=80mA, 3V current=60mA, 2V current=40mA) [45]. Experimentation can be found to find the optimal voltage supply for the motor. If a voltage other than 5 volts is used, we will need to add a voltage regulator to the circuit.

Grip Simulation

Implement for all 5 fingers Currently the Grip Simulation is only implemented for only 2 fingers (that is the Index finger and the Thumb). The future work will be to fit all the grip simulation components for use in all 5 fingers.

A more compact design The size of the component used for Grip Simulation is very large to be able to fit for use in all 5 fingers. In future, they can be made more compact to reduce the size they take up on the Glove.

Position Mapping

Implement using Magnetometers The methodology described in [64], [54] and [52] is a good starting point for create a custom methodology for position mapping using magnetism that meets our requirements (as listed in page 44). Either the magnetometer we had selected can be used or use a different magnetometer that can take measurements at the appropriate frequency. If this method is implemented it will also increase the accuracy of the Orientation Tracking as discussed in [64].

Implement using Computer Vision Although from limited knowledge we found that Computer Vision would be not be good enough for position mapping in our system, it is still popular for these applications. Hence, more can be researched in how commercial applications are using these technologies and overcoming their limitations.

Overall System

Replace Veroboard circuit with PCBs Currently the whole system is implemented on veroboards. This is as compact and robust as PCBs.

Make a more compact and robust Glove A lot of how the system can be made more compact has already been discussed in the previous sub-sections. The more compact and robust design will create a better user experience for the user.

References

[1] "Access Physiotherapy the upper extremity: The elbow, forearm, wrist, and hand," https://accessphysiotherapy.mhmedical.com/content.aspx?bookid=965&sectionid=53599844, accessed: 2019-05-27.

[2] "Aircraft Rotations body axes," https://www.grc.nasa.gov/www/k-12/airplane/rotations.html, accessed: 2019-05-18.

[3] "Algorithmia introduction to computer vision," https://blog.algorithmia.com/introduction-to-computer-vision/, accessed: 2019-05-31.

[4] "Amazon htc htc virtual reality apparatus, vive," https://www.amazon.com.au/HTC-Virtual-Reality-Apparatus-NEO_Combo-C9/dp/B077CXF6CM/ref=sr_1_1?keywords=HTC+Vive&qid=1558274794&s=gateway&sr=8-1, accessed: 2019-05-19.

[5] "Amazon mediabridge usb 2.0 - micro-usb to usb cable (6 feet) - high-speed a male to micro b," https://www.amazon.com/Mediabridge-USB-2-0-High-Speed-30-004-06B/dp/B004GF8TIK, accessed: 2019-05-31.

[6] "Amazon oculus rift s pc-powered vr gaming headset," https://www.amazon.com.au/Oculus-Rift-PC-Powered-Gaming-Headset/dp/B07PTMKYS7/ref=sr_1_2?crid=20X13QJQTLRXQ&keywords=oculus+rift&qid=1558275782&s=gateway&sprefix=oculus+rift%2Caps%2C405&sr=8-2, accessed: 2019-05-19.

[7] "Amazon official google cardboard," https://www.amazon.com/Google-87002822-01-Official-Cardboard/dp/B01L92Z8D6, accessed: 2019-05-21.

[8] "Amazon playstation vr with camera and vr worlds game (v2)," https://www.amazon.com.au/PlayStation-VR-Camera-Worlds-Game/dp/B07B2QBQYP/ref=sr_1_1?crid=398QFJL2WB53M&keywords=playstation+vr&qid=1558402177&s=videogames&sprefix=playst%2Cvideogames%2C293&sr=1-1, accessed: 2019-05-21.

[9] "Arduino download the arduino ide," https://www.arduino.cc/en/Main/Software, accessed: 2019-05-31.

[10] "Arduino setclockdivider()," https://www.arduino.cc/en/Reference/SPISetClockDivider, accessed: 2019-05-24.

[11] "Arduino wire.setclock()," https://www.arduino.cc/en/Reference/WireSetClock, accessed: 2019-05-24.

[12] "BC Robotics pressure-sensitive conductive sheet," https://www.bc-robotics.com/shop/pressure-sensitive-conductive-sheet/, accessed: 2019-05-18.

[13] "Caplinq linqstat mvcf 50,000 ohms/sq 4 mil conductive film," https://www.caplinq.com/linqstat-mvcf-50000-ohms/sq-4-mil-mid-level-electrically-conductive-plastic-sheeting-mvcf-4s50k-series.html, accessed: 2019-06-01.

[14] "Circuit Basics basics of the i2c communication protocol," http://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/, accessed: 2019-05-18.

[15] "Circuit Diagram design and share diagrams using a wide range of components," https://www.circuit-diagram.org/, accessed: 2019-05-30.

[16] "Core Electronics flex sensor 2.2"," https://core-electronics.com.au/flex-sensor-2-2.html, accessed: 2019-06-01.

[17] "Core Electronics pressure-sensitive conductive sheet (velostat/linqstat)," https://core-electronics.com.au/pressure-sensitive-conductive-sheet-velostat-linqstat.html, accessed: 2019-06-01.

[18] "Creality creality ender3 3d printer," https://www.creality3d.cn/creality-ender3-3d-printer-p00244p1.html, accessed: 2019-05-27.

[19] "CY8CKIT-059 PSoC 5LP prototyping kit with onboard programmer and debugger," https://www.cypress.com/documentation/development-kitsboards/cy8ckit-059-psoc-5lp-prototyping-kit-onboard-programmer-and, accessed: 2019-05-18.

[20] "D1 mini a mini wifi board with 4mb flash based on esp-8266ex," https://wiki.wemos.cc, accessed: 2019-05-18.

[21] "Digi-Key mmc3416pj-b," https://www.digikey.com.au/product-detail/en/memsic-inc/MMC34160PJ-B/1267-1044-ND/5658592, accessed: 2019-05-18.

[22] "Digital Citizen simple questions: What is wifi direct? how does it work?" https://www.digitalcitizen.life/simple-questions-what-wifi-direct-how-does-it-work, accessed: 2019-05-18.

[23] "eBay au mt3608 step up dc-dc switching power apply module boost converter for arduino," https://www.ebay.com.au/itm/AU-MT3608-Step-Up-DC-DC-Switching-Power-Apply-Module-Boost-Converter-for-Arduino/153114535863?hash=item23a65667b7:g:c1EAAOSwnn1bWXhP, accessed: 2019-05-27.

[24] "Ebay mitsumi micro k20 motor 6mm*8mm dc 3v-5v high speed motor worm gear screw shaft," https://www.ebay.com.au/itm/MITSUMI-Micro-K20-Motor-6mm-8mm-DC-3V-5V-High-Speed-Motor-Worm-Gear-Screw-Shaft-/113193483227, accessed: 2019-05-19.

[25] "elementz mpu6050 mpu-6050 gy-521 3 axis analog gyro sensor + 3 axis accelerometer module," https://www.elementzonline.com/mpu6050-gy-521-3-axis-analog-gyro-sensors-accelerometer-module, accessed: 2019-05-18.

[26] "engadget haptx promises to make your virtual hands feel like real ones," https://www.engadget.com/2017/11/20/haptx-gloves-vr/, accessed: 2019-05-31.

[27] "ESP8266 WiFi Module the esp8266 wifi module is a self contained soc with integrated tcp/ip protocol stack that can give any microcontroller access to your wifi network," https://grobotronics.com/esp8266-wifi-module.html?sl=en, accessed: 2019-05-18.

[28] "GitHub vr-glove-final-year-project-fyp-2019," https://github.com/mnahm5/VR-Glove-Final-Year-Project-FYP-2019, accessed: 2019-06-01.

[29] "Google Patents flexible potentiometer," https://patents.google.com/patent/US5583476A/en, accessed: 2019-06-01.

[30] "Is Bluetooth Faster than Wi-Fi?" https://www.howtogeek.com/191546/is-bluetooth-faster-than-wi-fi/, accessed: 2019-05-18.

[31] "Khora Virtual Reality vr box – iphone and android – low-end vr headset," https://khora-vr.com/product/vr-box/, accessed: 2019-05-21.

[32] "Lolin NodeMCU lolin nodemcu esp8266 cp2102 nodemcu wifi serial wireless module," https://www.amazon.in/Lolin-NodeMCU-ESP8266-CP2102-Wireless/dp/B010O1G1ES, accessed: 2019-05-18.

[33] "MakerLab Electronics ultrasonic sensor hc-sr04," https://blog.algorithmia.com/introduction-to-computer-vision/, accessed: 2019-05-31.

[34] "Mouser Electronics cypress semiconductor cy8ckit-062-wifi-bt psoc 6 wifi-bt pioneer kit," https://au.mouser.com/new/cypress-semiconductor/cypress-psoc6-wifi-bt-pioneer-kit/, accessed: 2019-05-24.

[35] "Node MCU Documentation wifi module," https://nodemcu.readthedocs.io/en/master/modules/wifi/, accessed: 2019-05-20.

[36] "Packet Sender award-winning free utility to send and receive network packets. tcp, udp, ssl," https://packetsender.com/, accessed: 2019-05-30.

[37] "Positron Technologies hc-05 wireless bluetooth," http://positrontech.in/eshop/product/hc-05-wireless-bluetooth/, accessed: 2019-05-27.

[38] "Solidworks solidworks and sw pdm system requirements," https://www.solidworks.com/sw/support/SystemRequirements.html, accessed: 2019-05-27.

[39] "Sparkfun lithium ion battery - 1ah," https://www.sparkfun.com/products/13813, accessed: 2019-05-26.

[40] "Ultimaker install ultimaker cura," https://ultimaker.com/en/resources/52833-install-ultimaker-cura, accessed: 2019-05-27.

[41] "Unreal Engine setting up collisions with static meshes," https://docs.unrealengine.com/en-US/Engine/Content/Types/StaticMeshes/HowTo/SettingCollision/index.html, accessed: 2019-05-31.

[42] "Vector accelerometer, gyroscope, compass, gps, light sensor, barometer. important phone functions. black icon," https://previews.123rf.com/images/jeler/jeler1709/jeler170900017/85477056-accelerometer-gyroscope-compass-gps-light-sensor-barometer-important-phone-functions-black-icon-.jpg, accessed: 2019-05-18.

[43] "Vector Stock simple electromagnet," https://www.vectorstock.com/royalty-free-vector/simple-electromagnet-vector-1856326, accessed: 2019-05-31.

[44] "Veer VR Blog high-end-vr-headsets," https://veer.tv/blog/the-most-popular-vr-headsets-in-2017-and-why-they-are-different-in-price/high-end-vr-headsets/, accessed: 2019-05-21.

[45] "Vibrating Mini Motor Disc," https://www.adafruit.com/product/1201, accessed: 2019-05-18.

[46] "VR Times full list of glove controllers for vr," https://virtualrealitytimes.com/2017/03/14/full-list-of-glove-controllers-for-vr/, accessed: 2019-06-01.

[47] "Wikipedia arduino ide," https://en.wikipedia.org/wiki/Arduino_IDE, accessed: 2019-05-31.

[48] "Wikipedia file:googlemaps logo.svg," https://upload.wikimedia.org/wikipedia/commons/thumb/4/48/GoogleMaps_logo.svg/1011px-GoogleMaps_logo.svg.png, accessed: 2019-05-31.

[49] "Wikipedia three-dimensional space," https://en.wikipedia.org/wiki/Three-dimensional_space, accessed: 2019-05-31.

[50] "YouTube - Peterjohn Walter finalyearprojectvideo," https://youtu.be/SyVdhz7PpnE, accessed: 2019-05-19.

[51] "YouTube - Sykoo unity vs unreal - design, graphics and performance," https://youtu.be/3SGb8-o-cO0, accessed: 2019-05-19.

[52] J. Blankenbach and A. Norrdine, "Position estimation using artificial generated magnetic fields," in *2010 International Conference on Indoor Positioning and Indoor Navigation*, Sep. 2010, pp. 1–5.

[53] J. Castellano, D. Casamichana, J. Calleja-Gonzalez, J. S. Roman, and S. M. Ostojic, "Reliability and Accuracy of 10 Hz GPS Devices for Short-Distance Exercise," *J Sports Sci Med*, vol. 10, no. 1, pp. 233–234, 2011, [PubMed Central:PMC3737891] [PubMed:19853507].

[54] K.-Y. Chen, S. N. Patel, and S. Keller, "Finexus: Tracking precise motions of multiple fingertips using magnetic sensing," 05 2016, pp. 1504–1514.

[55] *PSoC® 6 WiFi-BT Pioneer Kit Guide*, Cypress, April 2019.

[56] *I2C Master/Multi-Master/Slave*, Cypress PSoC Creater Component Datasheet, September 2014.

[57] *Serial Peripheral Interface (SPI) Master*, Cypress PSoC Creater Component Datasheet, November 2017.

[58] *ESP8266EX Datasheet*, Espressif Systems IOT Team, June 2015.

[59] Ho-Duck Kim, Sang-Wook Seo, In-hun Jang, and Kwee-Bo Sim, "Slam of mobile robot in the indoor environment with digital magnetic compass and ultrasonic sensors," in *2007 International Conference on Control, Automation and Systems*, Oct 2007, pp. 87–90.

[60] *MPU-6000 and MPU-6050 Product Specification*, InvenSense Inc, August 2013.

[61] D. J. Simon, "Optimal state estimation: Kalman, h-infinity, and nonlinear approaches," 01 2006.

[62] M. A. Khan, W. Chérif, F. Filali, and H. Ridha, "Wi-fi direct research - current status and future perspectives," *Journal of Network and Computer Applications*, vol. 93, 06 2017.

[63] J. H. Painter, D. Kerstetter, and S. Jowers, "Reconciling steady-state kalman and alpha-beta filter design," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, no. 6, pp. 986–991, Nov 1990.

[64] E. Paperno, I. Sasada, and E. Leonovich, "A new method for magnetic position and orientation tracking," *IEEE Transactions on Magnetics*, vol. 37, no. 4, pp. 1938–1940, July 2001.

[65] M. Pedley, "Tilt sensing using a three-axis accelerometer," *Freescale semiconductor application note*, vol. 1, pp. 2012–2013, 01 2013.

[66] J. Russell and R. Cohn, *Joule Heating*. Book on Demand, 2012. [Online]. Available: https://books.google.com.au/books?id=Jc6XMQEACAAJ

[67] *Flex Sensor*, Spectra Symbol, Jan 2014.

[68] *ULN2803A Darlington Transistor Arrays*, Texas Instruments, Feb 2017.

[69] Y. Xuan, R. Sengupta, and Y. Fallah, "Making indoor maps with portable accelerometer and magnetometer," in *2010 Ubiquitous Positioning Indoor Navigation and Location Based Service*, Oct 2010, pp. 1–7.

10 Appendices

10.1 Setting up the Wemos Development Environment

Figure 72: Micro-USB Cable [5]

The Wemos can be connected to the computer for programming and debugging, using a Micro-USB cable (as seen in figure 72. The drivers should be automatically installed as soon as the cable is connected to the PC, as long as the internet connection is available in the PC.

Figure 73: Arduino IDE [47]

The Wemos supports Arduino code. Hence, we used the Arduino IDE for writing the code and uploading the code to the Wemos. The IDE can be download from their website [9].

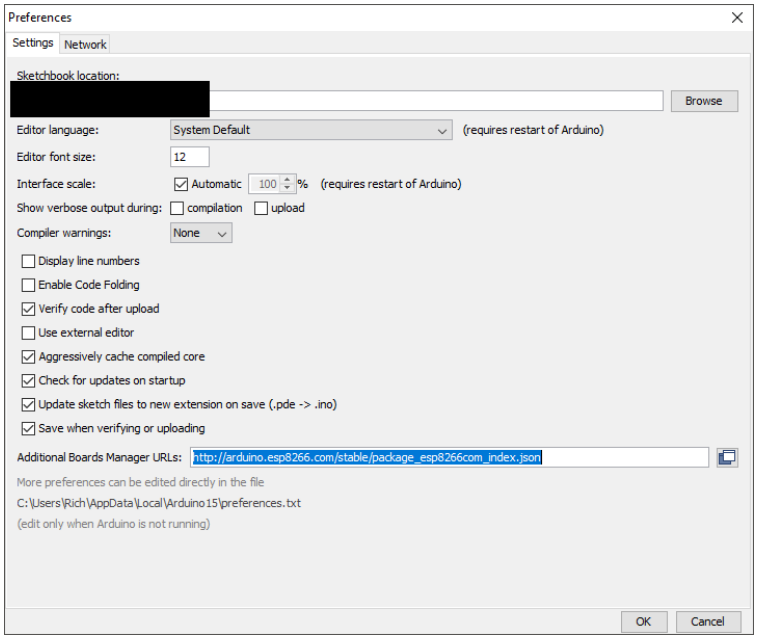


Figure 74: Arduino IDE Preferences

The Arduino IDE does not, by default, support ESP8266 boards like the Wemos. To allow us to compile and upload Wemos board, we need the following steps. We need to **File>Preferences** on the Arduino IDE menu. After this the Preferences Window will open as seen in figure 74. Towards the bottom of the window, on the **Additional Boards Manager URLs** field, we have type in the following link - http://arduino.esp8266.com/stable/package_esp8266com_index.json.

The next step involves opening the Board Manager window as shown in figure 75. This can be done through the Arduino IDE menu **Tools>Board>Boards Manager**. In the search window, we search the **esp8266** as shown in figure 75. The package that shows up needs to be installed for compiling and uploading code to the Wemos.

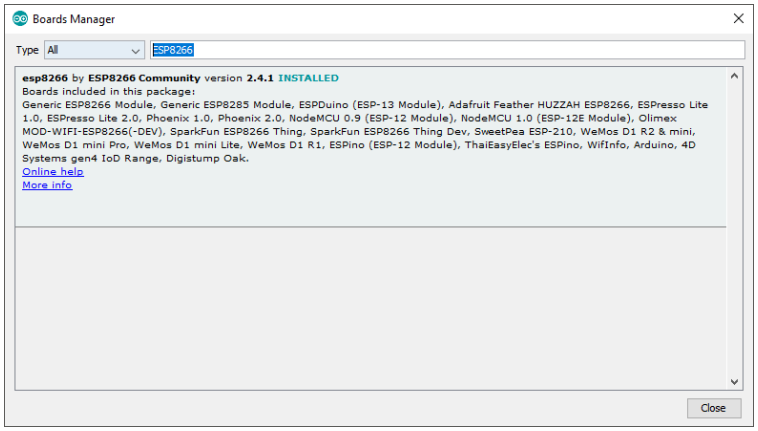


Figure 75: Arduino IDE Board Manager

After this set-up, we open the Wemos project by going to the **root/hardware/wemos-master/** folder and open the **wemos-master.ino** file. After opening the file, we can select the board through the IDE menu **Tools− >Boards** and pick the **LOLIN(WEMOS) D1 R2 & Mini** as seen in figure 76.

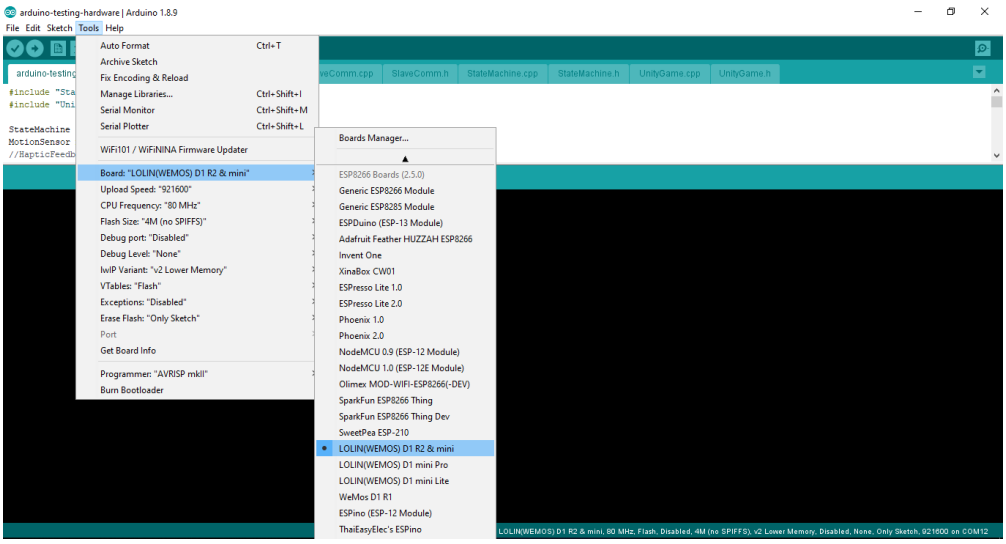


Figure 76: Arduino IDE Pick the Wemos Board

10.2 Packet Sender

Packet Sender was a tool used during the testing stages of the project. It was used to test the data transfer to and from the Wemos. The benefit that this tool provided was that the data communication of the Wemos can tested with the use of Unity Game. More details about the program can be found in [36].

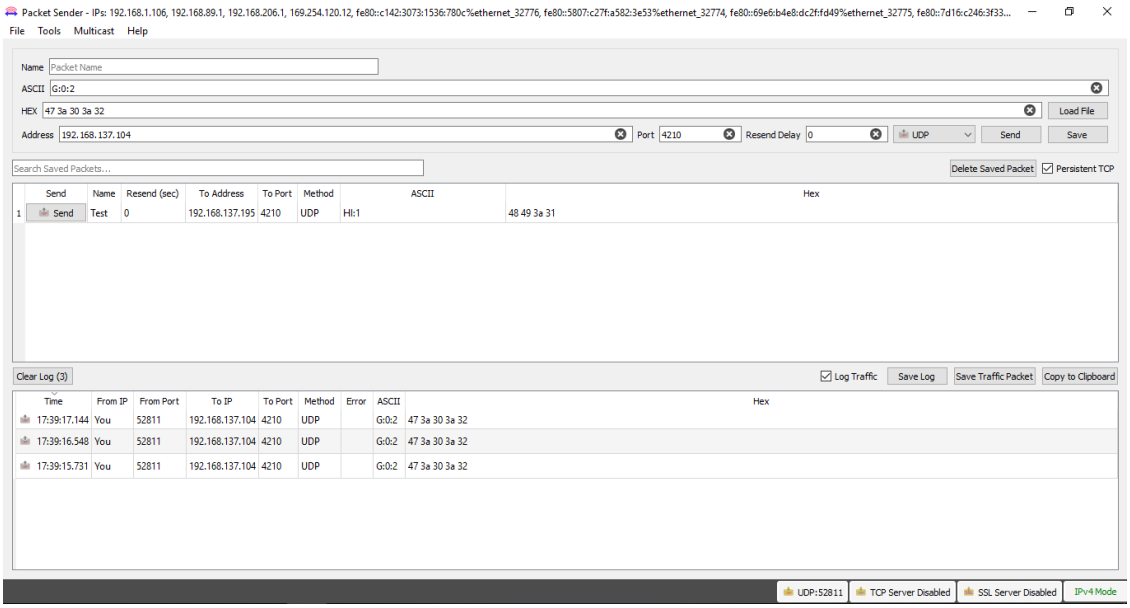


Figure 77: Packet Sender

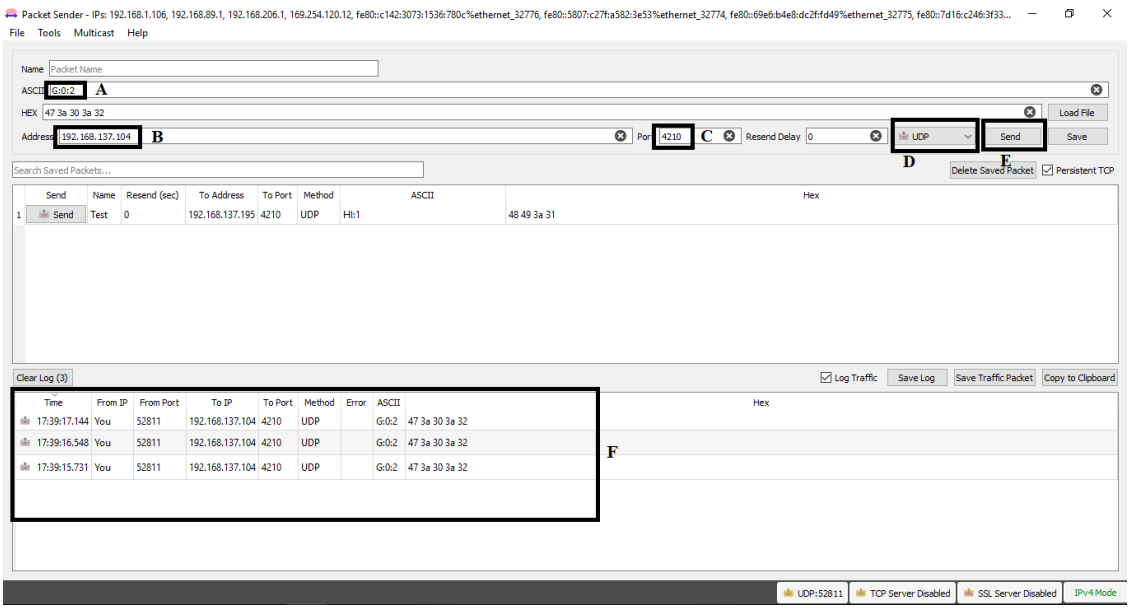


Figure 78: Packet Sender Highlighted for information

Figure 78 shows the Packet Sender tool in use. The important areas of the screen have been highlighted using boxes and labelled. The labels are explained below -

A The data to be sent

B The local ip address assigned to the Wemos

C The port number for data communication used by the Wemos

D The protocol for sending the data (in this case UDP)

E Send packet to the Wemos. To be pressed once the rest has been configured.

F The logs. Shows both send data to the Wemos and received data from the Wemos (if available).

10.3 Circuit Diagrams

The circuit diagrams used in this document can be made using an open source, easy-to-use program called Circuit Diagram. Figure 79 shows the program that was used. More details can be found in [15].

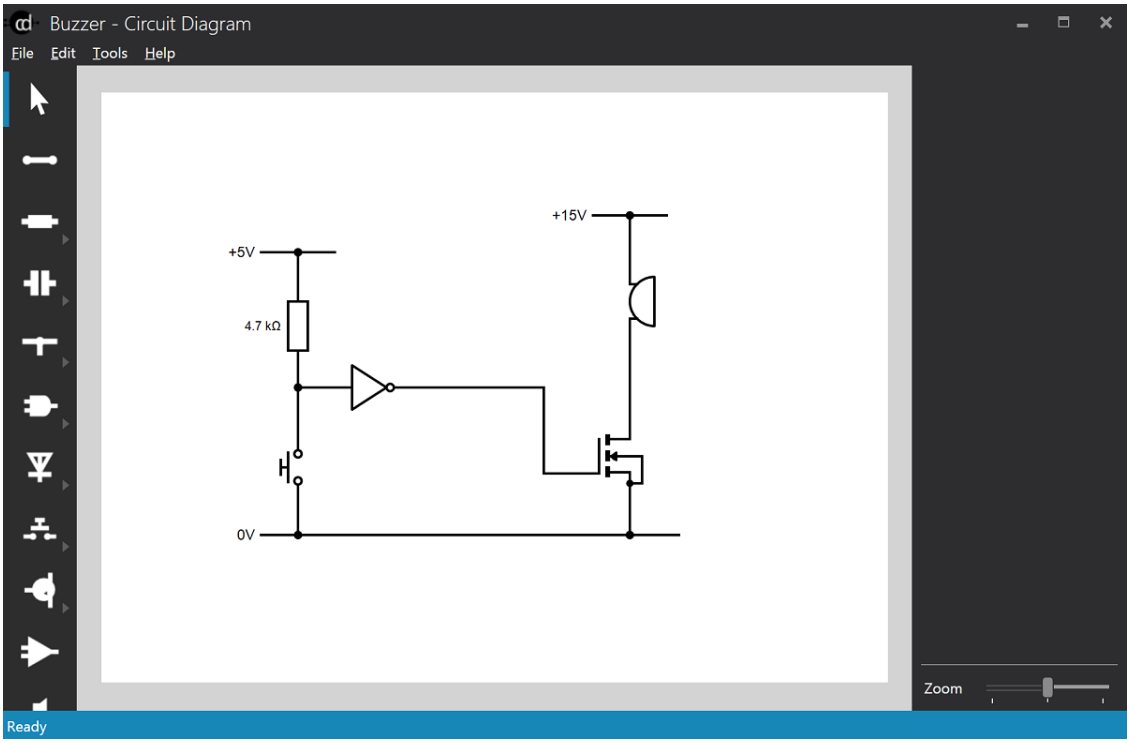


Figure 79: Circuit Diagram Desktop Tool [15]