

# Digital Whiteboard

Sarthak Das

# Contents

The Problem	6
<i>Computational Methods</i>	8
<i>Stakeholders</i>	9
Client Interview	10
Stakeholder Questionnaires	12
Research	14
<i>Existing Technologies</i>	15
Infrared Scan Whiteboard	15
Electromagnetic Pen-based Board	16
Resistive Touch Board	17
<i>Existing Solutions</i>	17
Tablet Pen Recognition	17
Moveable Smart Board	18
<i>Calibration Techniques</i>	19
<i>Evaluation and Summary of Reseach</i>	21
Solution	22
<i>Specification</i>	24
<i>Limitations</i>	26
<i>Requirements</i>	27

<i>Success Criteria</i>	28
<b>Design</b>	<b>29</b>
<i>Program Structure</i>	31
GUI	31
Calibration	32
Sensor Interfacing	33
Class Diagram	34
<i>UI and UX design</i>	35
Initial Drawings	36
User Feedback and Testing	47
Smooth Line Algorithm	50
<i>Homography (Calibration Algorithm)</i>	55
<i>Data Flow Diagrams</i>	57
Wii Remote Data Handler	58
Initialiser	59
Homography Calibration	64
<i>Validation</i>	71
<i>Data Dictionary and Variable Table</i>	73
Main Code	73
Wiiremote	75
Homography	76
	2

<i>Post Development Testing</i>	80
Stakeholder Testing	81
Reliability Testing	81
<b>Development</b>	<b>84</b>
<i>Sensor Interfacing</i>	85
Establishing Wii Remote Connection	85
Data Transmission and Capturing	86
Final In-Development of Sensor Interfacing	90
Sensor In-Development Testing	92
<i>Homography</i>	99
Translation Matrix Generation	100
Final In-Development Translation Matrix Generation Testing	106
Co-ordinate Converter	109
Final In-Development Testing for Co-ordinate Converter	112
<i>GUI and UX</i>	114
Setting up of PyGame and the GUI	115
GUI Components and Pages	121
Final In-Development GUI Components Testing	125
Page Layout	128
<i>Email Validation</i>	140
Obtaining Characters Individually in the Email	141

Final In-Development Email Validation Testing	150
<i>Final Code Development</i>	<i>151</i>
Prototype 1	152
Prototype 2	155
Prototype 3	157
<i>Further Development of Prototype</i>	<i>171</i>
Looking at Example Code and Documentation	171
Attempting to Empty Memory Buffer	172
Changing Main Loop Run-time	172
<b>Testing</b>	<b>173</b>
<i>Functionality Testing</i>	<i>174</i>
Testing Against Problem Described	175
Range and Physical Limitations	177
<i>Usability Testing</i>	<i>180</i>
Questionnaire Survey	180
Focus Group Survey	183
Drawing Usability Testing	184
<i>Robustness Testing</i>	<i>187</i>
Different Linux Distros Testing	188
Testing on different amounts of RAM	189
Project Budget	191

Testing on Different Screen Sizes	191
Calibration Robustness Testing	192
<b>Evaluation</b>	<b>195</b>
<i>Reflection on Functionality Testing</i>	<i>196</i>
Success Criteria	197
Addressing Unmet Success Criteria	200
<i>Reflection on Usability Testing</i>	<i>201</i>
Success Criteria	201
Addressing Unmet Success Criteria	203
<i>Current Limitations</i>	<i>204</i>
<i>Maintenance</i>	<i>205</i>
<i>Further Development</i>	<i>206</i>

# The Problem

There is a gap between communication within collaborative work and the modern, high tech world we live in. Brainstorming and idea sharing is an essential part of daily life in companies and in schools, and there is no cheap, flexible solution that address this.

At work employees must often work as a team in order to come up with ideas or write reports. With current limiting solutions, this is made difficult. The most commonly used equipment is a whiteboard, where each member gets a pen and writes on a large board often mounted permanently on a wall.

In schools, students can be grouped to work on a project but don't have the resources to do so efficiently. Also, group discussion and mind mapping in class can be a struggle, as teachers often break the class down into smaller groups and later re-join the class discussion, which wastes valuable time. Also these smaller groups are unable to present their ideas due to the limitations of pen-and-paper brainstorming, due to the increase in paper costs and the reduction in budget for education.

Currently smartboards or interactive boards are extremely expensive and on average retail at prices above \$1000 causing many users to be disincentivised from buying such products, not only this but it leaves a large portion of the world's market untapped especially in 2<sup>nd</sup> and 3<sup>rd</sup> world countries. Using new technology almost 80% of the current whiteboard's technology can be replicated for a fraction of the cost.

This problem can be solved using abstraction using which would use existing hardware and electronics that are commercially available so that all of the electronics do not have to be custom designed and tested, this will save time and cost as this reduces the possible testing requirements and the things that are able to change creating a simpler, more streamlined solution that will be quicker to develop.

The main specification for the project should be that the computer should be sense the location of the users pen and then be able to accurately plot that point on the computer screen. This would be so that the user can write and that exact point will be recognised by the computer. The latency of the recognition must be minimal so this would offer a smooth user experience with minimal lag, creating the effect of a real pen.

Throughout the development the main key components can be developed in sections such as:

- Sensing the Location of the Pen
- Homography of pen and display calibration
- Wireless transmission of data from sensor to computer
- Plotting of the point on the display
- UX/UI design
- Integration with other software's and plugins to share

This would allow for clear set targets to be identified and easier error tracking during development and testing. This would also allow parallel development as each section can be developed independently of each other.

Logic comes in very handy in this project as there would have to be a sequence of steps that would have to be followed and use complex features such as error handling



and data handling from the sensors. This would be done to concurrently handle multiple things at once. Parallel processing would also have to be used in order for the sensor to sense the pens location and simultaneously plot the pens location of the display using the GUI.

## Computational Methods

This problem is best suited to a computer as it is able to permanently save data and files that can be used for a later date when needed and makes transportation significantly more easier with the development of SSD and smaller portable storage devices.

Abstraction should be used in this project to keep the software as streamlined and efficient as possible, removing extra features that the users may rarely use, reducing the coding load and hence the development cost but also making the overall application size smaller so it can be possible be less resource heavy and therefore operate on more devices. This should also eventually end up with a more minimalistic UI causing a better UX, as most of the useless features would have been removed.

From an overarching view and predicating ahead of how the software could be structured, is that it will need to somehow capture the users pen movements using a type of sensor to be determined later, and that should be able to filter out other touches i.e. palm rejection.

The problem will be solved using a decomposition, breaking down each part into individual sections and developing them independently, this will make debugging and error location easier, not only this but in the future if changes were to be made then the code structure would allow for this to occur by only changing that module. Each section will be viewed holistically when joining each part together as all the parts would have individually been pre-tested.

Concurrency may have to be used as there may require parallel processing of receiving data while plotting it and translating it, requiring the use of a buffer, stack, queue in order for the software to operate accurately.

## Stakeholders

Even though the end users target is teachers in 3<sup>rd</sup> world countries and businesses in such locations, the equivalent of those will be researched and looked at as people towards this demographic were not easy to get in constant contact with. Their equivalents will be researched as they generally have the same needs and wants.

During meetings there is a sharing of ideas from multiple people often on linearly (i.e without any order and contentiously) where constant changes where being made and some being altered. These companies tend to run on a tight budget and therefore have to stick to only necessary items. These companies tend to use bundled suites such as Microsoft Office or Apple Work, for all interactions and planning of meetings and sharing documents. Their brainstorming/ meetings generally last for about 20-60 minutes and contain a combination of looking at reports, drawing quick diagrams and creating 'kanban' lists.

In school many teachers have to use multiple different types of media such as whiteboard, presentations and pen and paper which causes major problems when trying to quickly move classrooms or organisation. They often teach for 50-60 minutes in a classroom then have to move to a different room, causing them to lose all their notes they have written on the board. The current technology of the boards also does not allow 'unlimited writing space' which causes them to rub out or erase previous work, causing the students to write quickly and unable to refer to work previously written.

Overall they would like a portable system that is accurate and easy to use and set up, from the education systems point of view they want a cheap system that is easy to install and maintain since most classrooms are generally small and old and often have to be retrofitted with new technology such as projectors, more sockets and other such technologies. This has also led to them asking for it to be small so that it does not take up significant space in the room.

## Client Interview

To figure out what the teachers' needs and requirements are, an interview is to be conducted for a more in-depth knowledge. In this scenario, a computer science teacher will be asked a series of questions that target around how they teach and their use of the whiteboard.

### Questions Reasoning and Answers

- (1) Currently what do you use to teach? What are your main resources you use and software?

This question is done to figure out any general features that should be included and any software's that it should be compatible with. This also helps to understand the teachers' workflow.

Currently he uses his laptop, to show the main bulk of the notes that are projected up on the board using a projector, while he used the side whiteboard for any additional notes and hand-drawn diagrams he has to draw. The main software that he uses is PowerPoint, Word and sometimes sketch book for certain digital diagrams he has to draw or photos he wants to annotate.

- (2) What do you use your whiteboard for mainly?

This was to figure out what features they may need in the whiteboard

The whiteboard is currently used for class discussions where people are able to come to the board and write what they thought the answers are or add to the list. He also uses it to sketch graphs or diagrams that are very likely to change.

(3) How do you currently share your notes that are on the whiteboard?

This is done to see what current technologies they use and the flaws and advantages that can be analysed later

Currently he has to either take a photo of it on his phone save it to drop box and then email it to all the students, or if they are higher up in the school he asks them to take a picture of it on their phone.

(4) What are any current frustrations you have with the whiteboard?

This is to see if they have any major problems they have self-identified and want fixing in the product and the reasoning in the future

He has to keep rubbing off current work whenever he has to move classrooms even though he had to use the same resources to there should be a way to save the work. He has also come into a common problem with people having their whiteboard pens stolen and them running out causing him to keep on buying pens and replacing them.

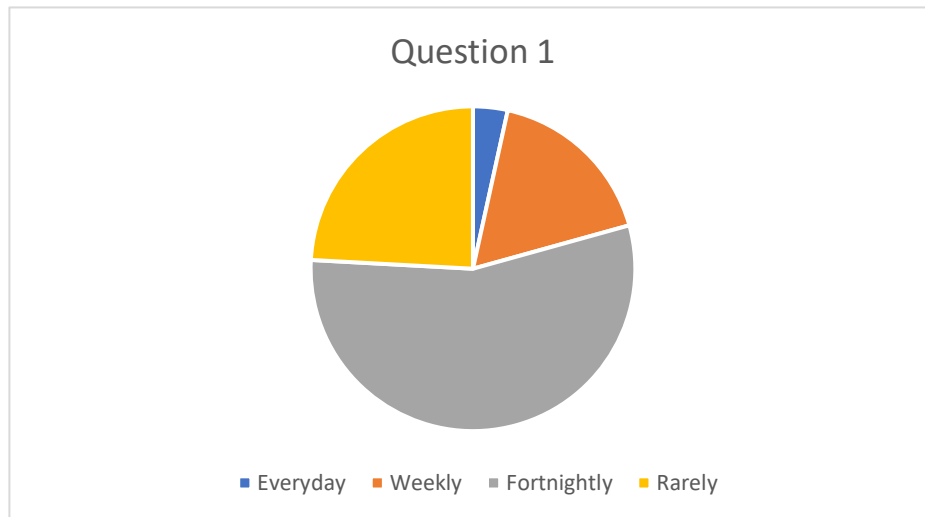
(5) Are there any features you would like to see in a product?

This is to find out any compulsory features that must be in the specification

He would like multiple people to use the whiteboard simultaneously, and for it to be able to handle multiple colours at once. He would also like a cheap system and one that is portable.

# Stakeholder Questionnaires

To gain some more knowledge on the general idea of the project a questionnaire was given out throughout the school for teachers to answer, so that the data can be analysed and be used to



influence the final product.

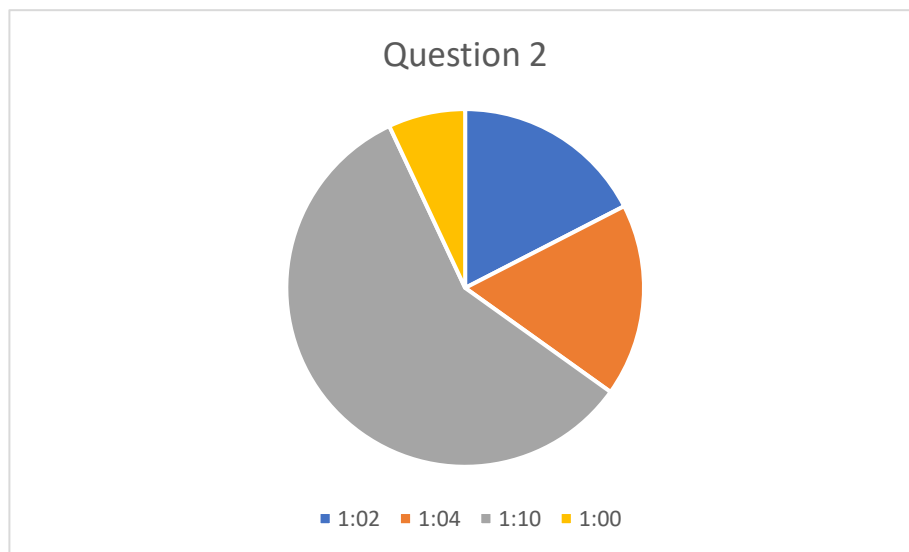
## Questions Reasoning and Answers

- How often do you use the interactive whiteboard?

This was asked to find out the frequency of use and can help estimate the usage to cost ratio.

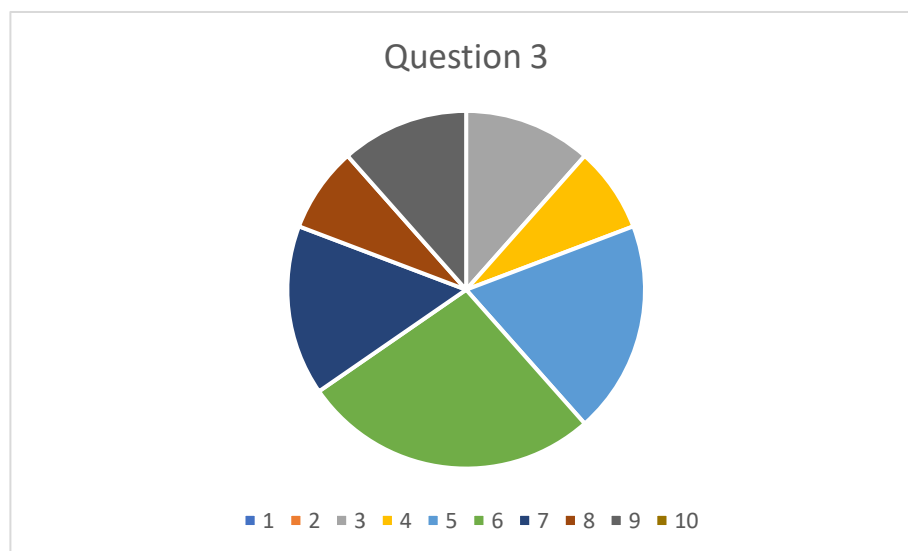
- What would you say is the student-teacher usage of the whiteboard?

This was to see how essential it was to be able to use multiple pens and ways to interact with the board.



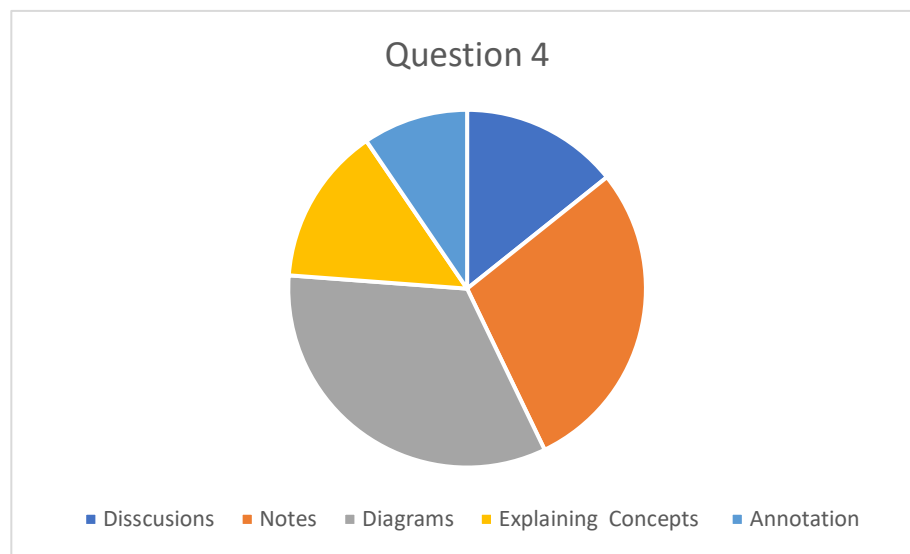
- On a scale of 1 – 10 how essential is your interactive whiteboard in your teaching kit?

This is to see if there is a wanted market for this and how desperate they are to have one.



- Do you have use the interactive whiteboard for your teaching purposes?

This is to see and get a feel on what features they may need in order for them to teach



Overall as per what that data shows is that interactive whiteboards not often used but by the teachers who use them they are used very often and the way they are used is diverse and multipurpose.

# Research

After identifying the general trends and needs of the client the specific research into the product can begin.

The research's main point is to identify ways to tackle the problem being faced by the users. This will be done by looking at the different technologies that are currently available in the market for the users to consume and use. Not only this but since the project is to do with financial differences as highlighted previously, market size growth and share will also be looked at in order to see if the project is commercially viable and able to withstand the competition in the market.

Sensor technology will also play an important role as this has to provide a key role in the software as without accurate recognition the calibration and writing experience will not be sufficient. The sensors used will also impact the algorithms used and software complexity, not only this but it undermines a major cost and retail value of the product.

## Existing Technologies

Through this project other existing technologies were looked at to see how they tackled a similar problem that this project also aims to solve. Not only this but since the solution also requires some research on sensors and technologies this will also have to be researched based on the cost, accuracy, reliability, and size.

### Infrared Scan Whiteboard

This is a large interactive display that can connect to a computer and a projector. This often requires a large space as the board has to be mounted to a wall or a moveable floor stand. This technology uses 3 or 4 sensors mounted to the edges of the board that



when the pen is pressed against the surface it causes the IR diode to light up, this infra-red hot spot can be then located by each of the sensors that have a x and y co-ordinate system. Using this data, the computer can then process this and tri-angulate the original point on the board. This technology is incredibly flexible, and the surface can be made out of almost any material, often dry-erase boards that allow people to use traditional whiteboard pens as well when they don't necessarily have a computer with them.

This technology is often cheap as it only requires 3-4 sensors that often are in the price range of \$50-\$100 often retailing for around the starting price of \$700 pounds. This technology often however is not very accurate and rare to see due to its size and inaccuracy of it, as it is prone to environment noise of other IR sources.

## Electromagnetic Pen-based Board

This technology generally consists of a array of wires that are hidden behind a solid board that is able to interact with the nib of the pen due to its magnetic interference. This is then able to identify the x and y co-ordinates of the pen. The pen usually is passive meaning that it does not require a power source. This technology is similar to those used in graphic tablets by digital artists and designers due to its accurate sensing abilities and it being able to sense pressure, that can allow for other capabilities to be developed. This technology is moderately expensive as it requires a huge arrays of wires, but the pen is often cheap due to its design.

# Resistive Touch Board

This technique involves a simple pointing device, with a common resistive system that covers the interactive areas surface. When the pointing device makes connection with the membrane it causes pressure which causes contact with the conducting back plate, this can then be electronically registered to find its co-ordinate. This technology is great as it does not require a special device and can be operated with a finger, causing it to be popular with teachers and classroom usage, however this means that there is the large board that required to be fitted and permanently installed in the room. This is generally very expensive but yields the best results and is the most accurate.

## Existing Solutions

This part focuses particularly on current solutions that are out there that are commercially in development and being sold around the globe and can be used to identify their strengths and weaknesses.

# Tablet Pen Recognition

This solution uses resistive touch method to track where the pen is and can offer features such as palm rejection. The prices can range from 50-200, depending on the

accuracy and depth (y-axis) sensitivity. The more expensive ones have a screen mirroring the computers screen which allows for users to accurately see where the current pen location is. This also allows the users to sit at a desk further away from the actual computer being controlled, say if it was being projected high up on to a large wall, such as in a auditorium or a large lecture theatre where reaching the board would be near impossible. This solution also offers the portability aspect as it can be placed inside a backpack and be transported from work to home or otherwise

Specifications : Wacom Cintiq Pro 24

Display Size	Resolution	Advanced Control	Productivity Boosters	Ergonomics	Compatibility
 59.94 cm / 23.6 in	 4K: Ultra HD (3840x2160)	 8192 levels pen pressure	 Time-saving on-screen settings, ExpressKeys, Touch Ring on included ExpressKey Remote	 Built-in stand (5° w/o legs or 20° w/ legs). Optional Wacom Ergo stand with rotation	 Windows Mac

Product type	Creative Pen Display 24
Model number	2 models: DTK-2420 creative pen display DTH-2420 creative pen display touch
Display size	677 x 394 x 47 mm (26.65 x 15.5 x 1.9 in)
Display weight	7.2 kg (15.87 lbs) without optional stand
Screen Size (Measured Diagonally)	59.94 cm / 23.6 in

[See more](#)



## Moveable Smart Board

The boards often either use the electromagnetic pen based or resistive touch. This solution is often more expensive ranging into the 500-2000-pound range. This solution targets more commercial and large scale usage. This solution consists of a large screen generally LCD or likewise with a sensor for pen tracking built inside the display. This is usually attached to a set of wheels so that it can be wheeled around different rooms or places or areas where it is needed or used. This is especially useful since it can easily be stored inside a storeroom if the product is not needed anymore at certain times.

# Calibration Techniques



## SMART Board® 7000 series

Inspire greatness

SMART Board 7000 series is 25 years in the making, and all new. The 7000 series is the digital heart of your interactive classroom experience, connecting SMART Boards and software, devices and vibrant imaginations. It helps students discover talents and passions, and teachers to promote student achievement in exciting new ways.

[Download brochure](#)

[Watch the video](#)

[Purchasing information](#)



FEATURING iQ embedded computing



FREE learning software



EXCLUSIVE SMART Ink® software



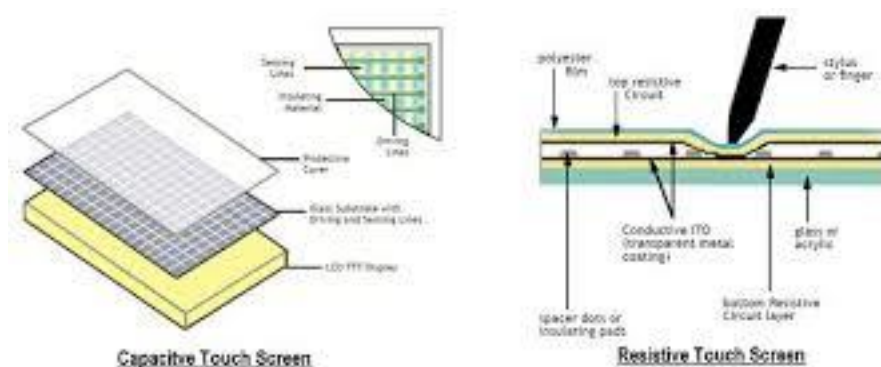
EXCLUSIVE Simultaneous Tool Differentiation



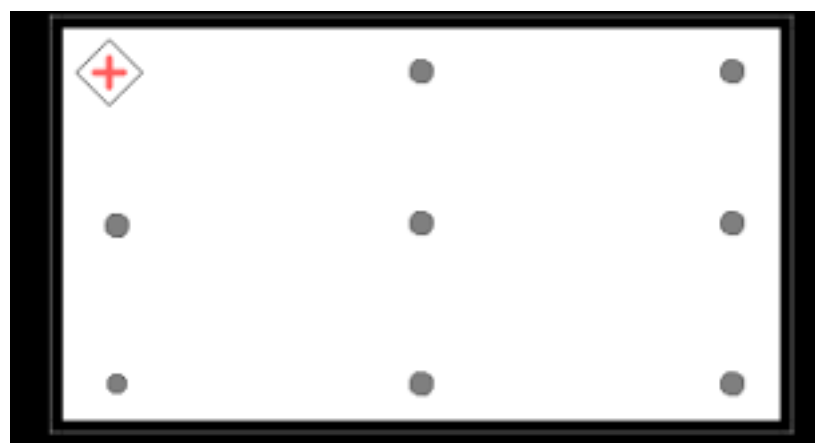
ENERGY STAR® certified

There are different calibration techniques are used depending on the type of exact application the engineers and designers were targeting and the technologies they used.

Resistive touch does not require any calibration as there is no need. This will ensure that once set there will always be a higher precision and accuracy as there will be no human error to be accounted for.



IR and Electromagnetic boards require calibration after every movement to make sure both systems are aligned and work effectively. What this requires is for the user to touch the board in certain place that was requested by the program. The amount of touches it requests for can vary from 4-8 with the more number of points making it more accurate as this would allow for reduction in human error and resultantly a better calibration.



# Evaluation and Summary of Research

From the stakeholder research and existing technologies and solutions a basic component selection can be made of hardware and software features.

A school would require a cheap solution as their main priority as they would not like to invest heavily on a product that is likely not to be used often. Therefore, this would have to rule out technologies that would be difficult and expensive to implement such as the electromagnetic technology and puts IR technology as its favour.

A product that is portable will also give it a huge market advantage as both solutions that are commercially viable are portable to some degree, there for to rival those products the solution that is being developed should also be. This was further backed up by teachers saying they move classrooms a lot and being able to carry their equipment around with them is better than it being set up permanently in one room.

Most of the solutions have basic features such as changing pen sizes and colours in-order for the product to work. Therefore, this could be considered core functionality as it is a common trait between all the existing solutions.

Overall the IR technology seems like the best fit for the conditions stated above as it is the cheapest technology and can also allow for some degree of transportation.

# Solution

The solution will include hardware and software, the software will recognize the input from the hardware and control the computer.

The idea is to have a infrared sensor with Bluetooth chip which will be used to send signals to the computer. The infrared sensor will sense when the infrared pen is turned on and will record the coordinate of the point this will then be sent to the computer via Bluetooth where it will be plotted on the application for the user to see. This idea was taken when looking at existing products as this was significantly cheaper than the alternatives and also allowed for a portable system compared to the other researched solutions.

The sensor will know the boundaries on the grid as when the application will be loaded up the user will be asked to put the pointer on all four corners. This will allow the computer and sensor to create a grid based on the four extremes.

The infrared pen will consist of a battery, push-switch and infrared emitting diode. When the button is pressed this will complete the circuit turning on the diode and allowing the infrared sensor to track its coordinates.

Overall the connection between the laptop and the sensor will be done using python as this language offers the flexibility due to its prebuilt modules and functionality that allow it to interact with the hardware. Python also has to have a GUI which would have to be developed to allow the user to interact with the program which can be developed with libraries such as Tkinter or PyQt.

During the development each stage or section will be developed individually of each other in order for the program to be modular interim of features and allows for an interactive development to be used compared to a linear one.

From the sensors point of view a wii remote will be used as this equipment is cheap to get hold of and has all of the inbuilt functions and hardware that is required to interface with the computer, such as the IR sensor and the Bluetooth chip that can provide the interfacing as well as the prebuilt circuitry and power distribution. This was taken user consideration during abstraction as this would simplify the development process and there would no need to design this from scratch.

The Wii remotes sensor also has the capability of tracking 4 points simultaneously, which should be more than adequate for multiple people to interact with the system at the same time making it ideal for usage in the classroom and can be seen using the data that people like the discussion usage of the whiteboard.

The solution chose the cheapest route the most often as reflecting upon the fact that teachers don't rate this piece of kit as essential for their teaching and this means that a device that costs \$3,000 is simply not worth the cost, where the technology from the wii remote and the IR technology may not offer the same amount of precision as other methods but will allow for 80% of the features to be produced with a fraction of the cost, making it more viable for a potential solution for 3<sup>rd</sup> world country users who can't afford to pay that steep of a price.



# Specification

This is a list of things that must be included in the designing of the software that can be used as a reference sheet during the development of this product.

Feature	Source of Feature	Essential	Description	Explanation
Addition of images to screen	Stakeholders	No	They can add images that can be written on to the canvas	This will allow them to annotate the picture with labels
Homography Calibration	Research of other solutions	Yes	This will allow for easier alignment from the users perspective	This can allow for the sensor to be placed anywhere and calibrated to the projector/screen
Multi-pen interaction	Stakeholders	Yes	Allows for multiple pens to be used at once	This will give it the flexibility of multiple users using the software at once

Multi-colour selection	Stakeholders	No	Allows for others colours to be chosen	So different blocks of writing can be
Handwriting Recognition	Own idea	No	Allows for written text to be identified and converted to typed text	Makes writing on the board easier to read
Laptop Connection to Wii remote	General	Yes	Able to retrieve data wirelessly from the sensor to track pens location	This allows the basic components of the system to work
File exporting/ Sharing ability	Stakeholders	No	To send their files to other students and teachers	This allows for the file to be removed from its application and sent
File saving	Stakeholders/ General	Yes	To be able to come back at a later time and make edits to a file	This allows the file to be worked on over a greater non-continuous period of time

For this project to be considered a success as a prototype model or as a proof of concept. It is to have the basic function of receiving data from the sensor and being able to accurately plot this on the GUI interface with the ability for it to be calibrated using Homograph. However, to make it a working or semi-usable by the stakeholders it should

have the ability to save the file and select multiple colours and be used with multiple pens.

## Limitations

The limitations mostly come from the economical and electronic sections in this project, with the accuracy the sensor and reliability having a direct consequence on the range the sensor from the board and the different environments it can be used in. Another factor that could impact the development is the time scale and existing knowledge that would greatly impact how far the project goes, especially in terms of the GUI and shareability features.

The project may encounter some limitations due to the hardware capability of the raspberry pi which it will be developed on, due to its small RAM size and slow virtual memory accessibility. This will possibly create bottlenecks that have buffer limitations, making the code have to be written in a certain way to make it more efficient and less dependent on the buffer.

Since the project is dependent on the sensor that will be bought in, the accuracy and limitations of which may impact the usability of the software. The exact extent of to which this will impact will have to be tested and evaluated in the software.

# Requirements

In order the program to be run and execute successfully certain conditions must be met such as:

- A Linux based operating system, as this is free and often used in 3<sup>rd</sup> world countries to save cost.
- The system must have Wi-Fi capabilities and Bluetooth capabilities in order to share files and to receive data from the sensor.
- Have a minimum of one IR touch pen in order for the sensor to track the pens location and plot it on the screen.
- Have python installed with the following dependent libraries: cwiid, pygame and back numpy which are the core libraries that are going to be used.
- The user will also have to have a Wii remote as this is the core external hardware required for the software to work.

- The user must have python 2.7 installed and able to run programs written in this software coding language.

The reason for the software to be only based on Linux because there is the use of the external library, which is only available for Linux based operating systems such as Ubuntu and Raspian this does not work on MacOS even though it is partly based on Linux. This should not encounter much of a problem as most poorer nations would use Linux based laptops as they are cheaper and can operate on lower specification machines and PCs.

The IR pen must be there as that is what the sensor is able to track to know where the user is writing. This however will require batteries likely 2x AA in order to power the LEDs, the pen must be of a minimum intensity and size in order for the sensor to pick up the location of the pen and plot it on the software.

## Success Criteria

To make sure the software is successful and has met the specification a range of tests will have to be carried out in order for it to be deemed successful in its purpose.

Code	Criteria	Category	Measured by (Test Method)	Justification
SC1	Data can be received from sensor	Core Functionality	Check Box	This is to ensure that the connection is possible between sensor and pc
SC2	GUI is usable and intuitive	Usability	Observing test subjects interaction with the software	This can then be seen where they

				have difficulty and how to adapt it
SC3	Completed under budget of 300	Financial		To make sure that the project is commercially viable
SC4	Robust in a wide variety of environment	Core Functionality	Random and Iterative Black box testing	This makes sure the software will work whatever the scenario.
SC5	File Sharing is compatible with different email servers	Usability	Black box testing	This is so whatever email they use the email sharing function will work

# Design

Currently, smartboards are very expensive and are not used by a wide variety of teachers, making it hard to justify the purchase of such equipment. But over 80% of the functionality a smart board can be achieved using only a fraction of the cost making it more of a justifiable purchase, this will also allow the equipment and project to be sold to a wider variety of consumers such as teachers in third world countries who struggle to afford such equipment previously. Another key factor is that this system has to be portable and quick to set up.

The solution will include hardware and software, the software will recognise the input from the hardware and control the computer.

The idea is to have a infrared sensor with Bluetooth chip which will be used to send signals to the computer. The infrared sensor will sense when the infrared pen is turned on and will record the coordinate of the point this will then be sent to the

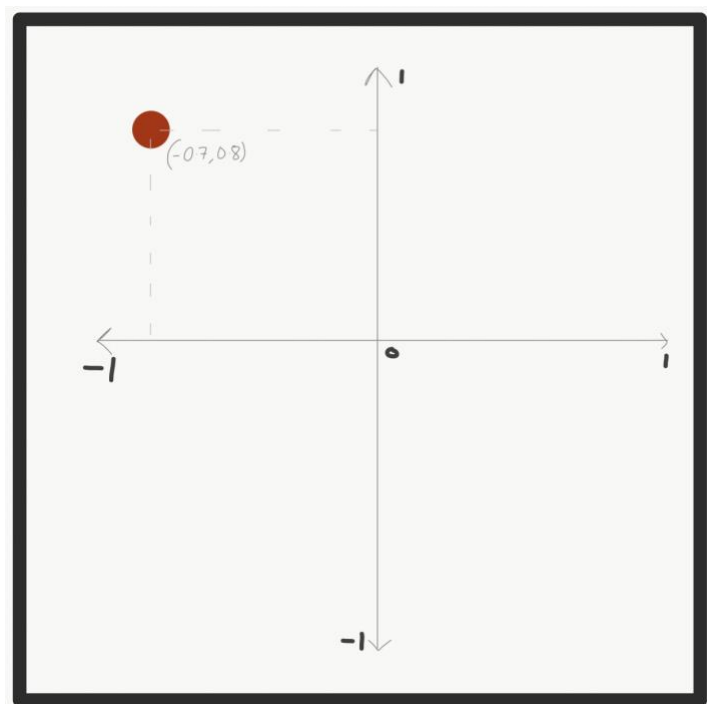


computer via bluetooth where it will be plotted on the application for the user to see.

The sensor will know the boundaries on the grid as when the application will be loaded up the user will be asked to put the pointer on all four corners. This will allow the computer to and sensor to create a grid based on the four extremes, using homography algorithms and equations as well.

The infrared pen will consist of a battery, push-switch and infrared emitting diode. When the button is press this will complete the circuit turning on the diode and allowing the infrared sensor to track its coordinates.

The infrared sensors will be able to track 4 pens at once this will allow multiple to write on the board at once.

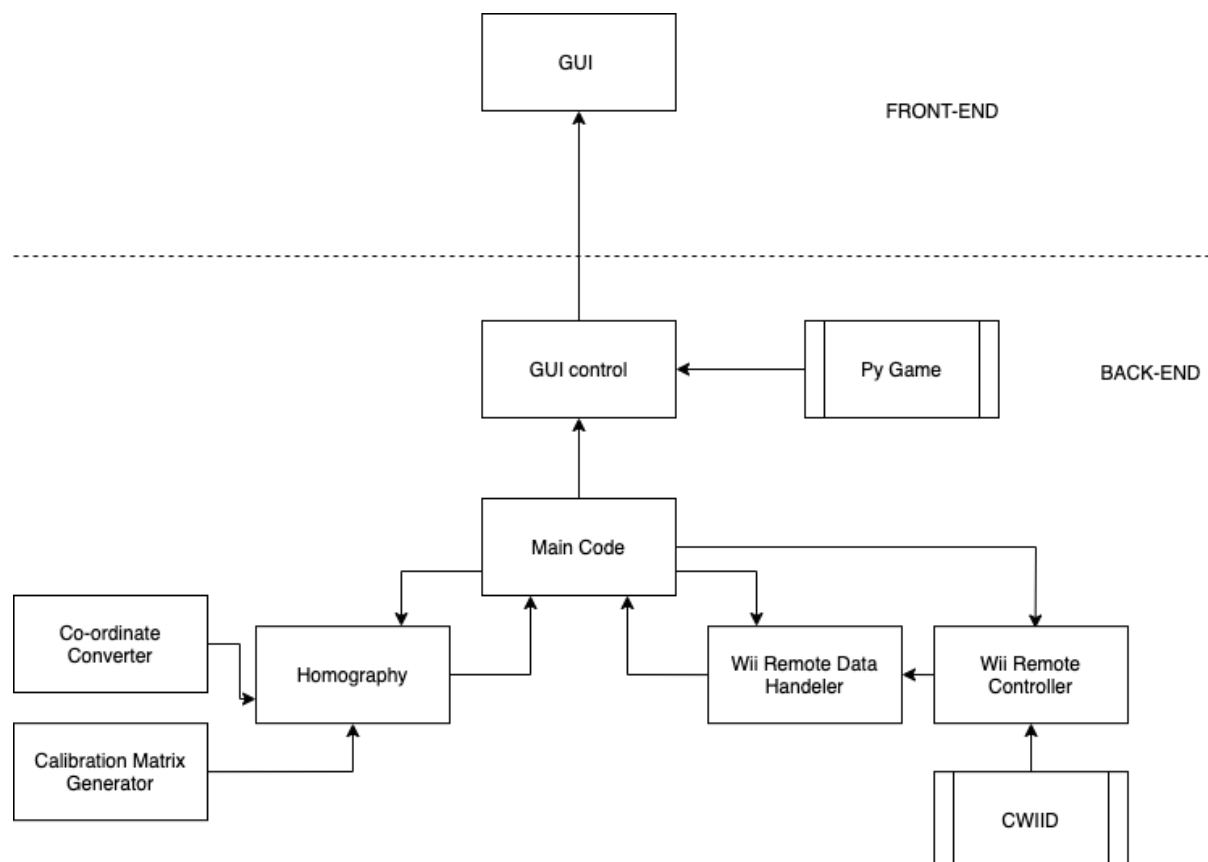


## Program Structure

The program will be developed in 3 main separate stages split due to its function that each section carries out or the main libraries that it uses.

### GUI

This section would mainly deal with the front end that the user will interact with, this is where the UI and UX design will have to be considered to allow for the best user



interaction allowing them to be productive.



The overall layout will be simple and straight forward, with a minimalistic design possibly using flat UI colours. Since this is simple to create and easy to use and interact with.

The GUI main interaction will have 3 main 'pages' or 'section':

- Connection page – this is to determine and show the user if the connection to the Wii remote was successful and how to do it.
- Calibration page – this is the page that will show icons on all four corners allowing for the four extremities of the screen to be found
- Canvas page – this is where the user can draw on the whiteboard and pick what colours and size of pen they want.

## Calibration

Since the projector (the screen that displays what the user sees) uses one set of co-ordinate system, and the Wii remote sensor uses another and they don't always have to align properly. This means that there has to be a mathematical way to deal with the alignment, which comes under the calibration.

Overall this section of the program should be responsible for:

- Creating the transformation matrix from the extremities of both co-ordinate systems

- When given one co-ordinate transferring that to the other systems co-ordinate system

## Sensor Interfacing

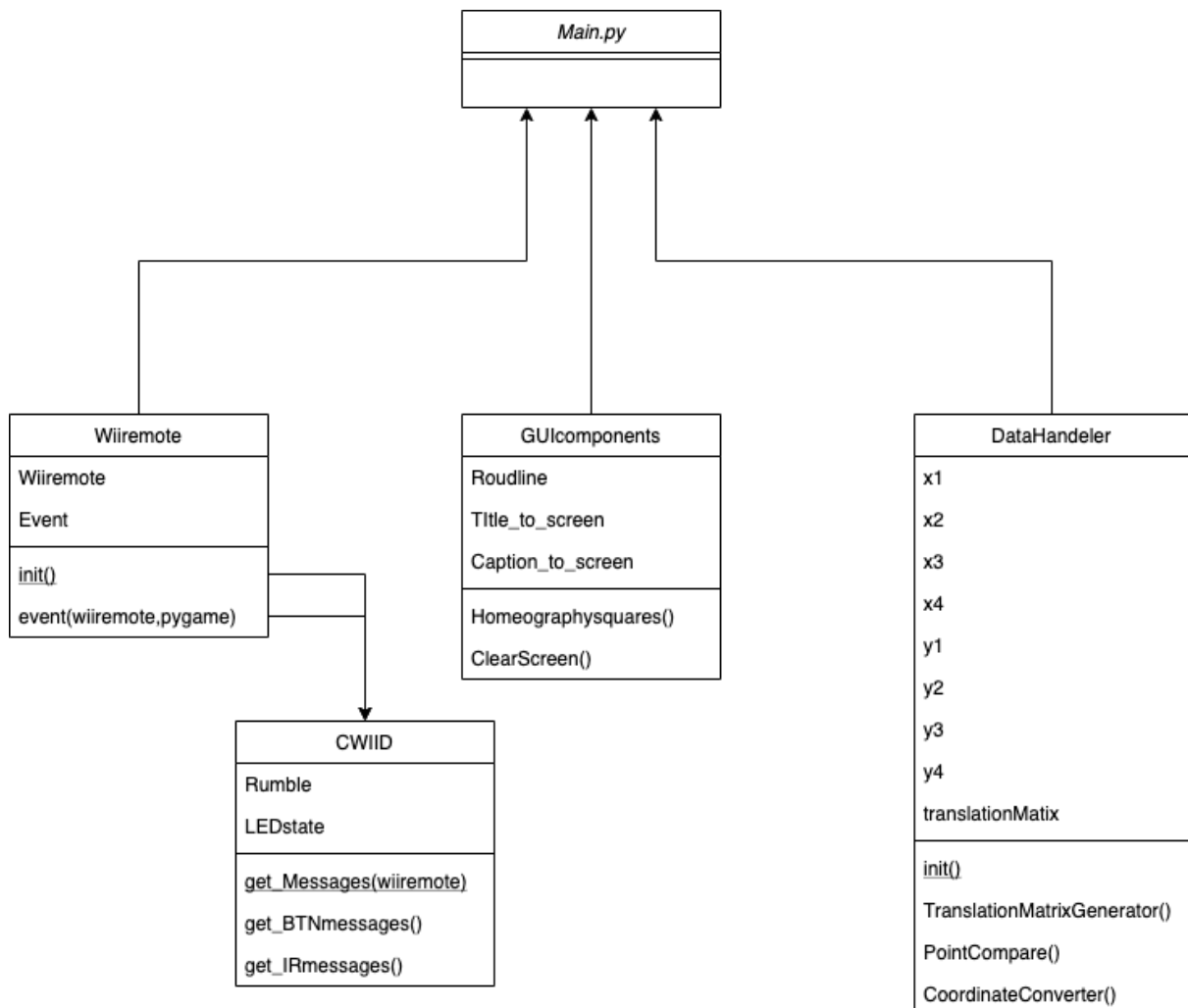
This part is responsible for the Wii remotes data handling and communication. This also includes basic data filtration to only allow useful data to be passed through, to prevent excessive data being passed around the system making it able to be more streamlined and efficient.

This part of the system will also include the section of interrupts from the sensor for example if any of the following happen:

- The Wii remote gets disconnected
- A new IR point has been found on the sensor

This will cause the program to stop what it's doing and handle this data first before continuing on what it was doing before, this is done to minimise lag when drawing or writing on the system as a whole.

# Class Diagram



This is the class diagram of what the code will look like. The main codes purpose is to act as a middleman between each of the other modules and classes, calling each objects methods and functions when needed.

The code is separated into different objects to make it easier to maintain as each module can be changed independently and should not effect the other modules as long as the input and output stays the same. This also allows the program to scale up in the future if it was to be developed commercially.

# UI and UX design

This is how the programs front end will look like and how it was developed.

When designing the front there were key features that had to be considered with the design to make sure it was usable for everyone, with a variety of teachers and their technical ability and also students who may have certain disabilities such as colour blindness, and vision impairment.

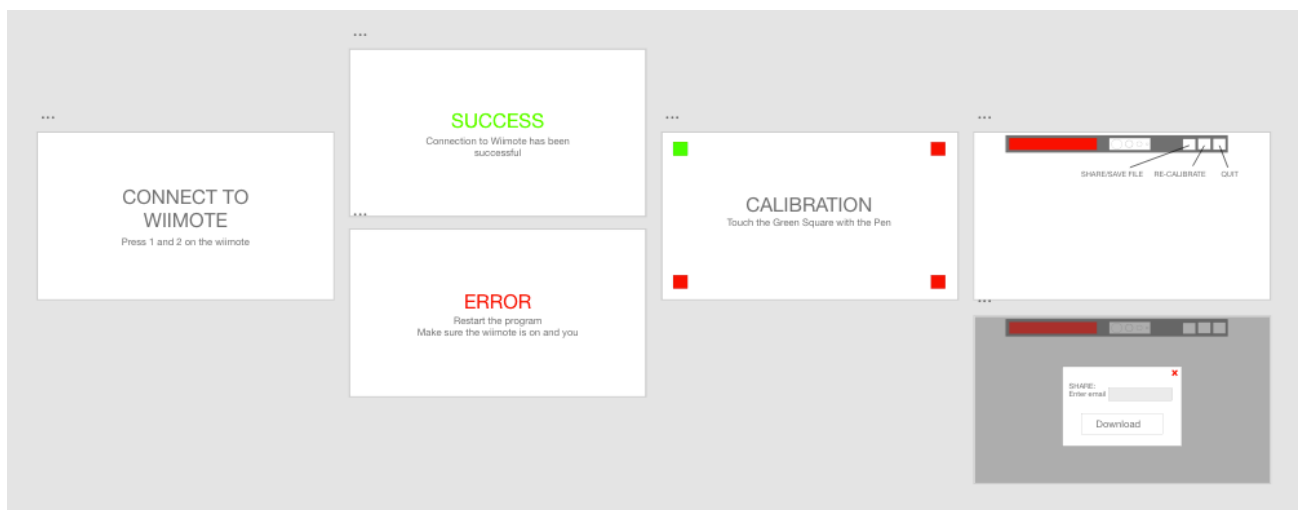
Key aspects that have to be considered when designing a frontend for the users:

- Accessibility of toolbar due to height
  - This is because the projector could be very tall making it hard for people who are vertically challenged to reach the top of the projected screen.
- Accuracy of the pen
  - Since the pen is not the most accurate and is prone to some interference, the buttons should be very large that even if there was slight discrepancies the correct size would be chosen.
- Vision Impairment
  - When the icons are used some of the colours used in them may not be visible to all people, especially those with vision impairments such as colour blindness and maybe partial blindness. This means that the icons must be large and easily recognisable and also be able to be seen from some distance. The colour palette must not include similar colours, making few but easily recognisable colours so that people can easily distinguish between each one.
- Responsive Design

- The use on the laptop/computer/projector screens can come in different sized and aspect ratios so the system must be able to adapt to each one of those and still provide all functionality

## Initial Drawings

The initial UI design used the feedback and research and incorporated the UX



design principles. The mock-ups were made using Adobe XD to easily create these visual templates that can be used to show potential clients. The start of the application is modelled on the left side and progresses towards the right side.

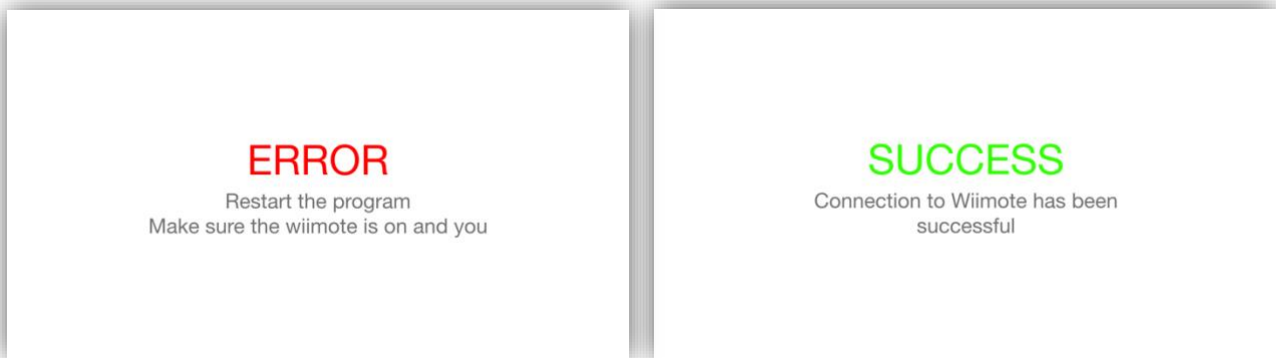
During the modelling a simplistic, minimalist design has been chosen, to make the UI as easy to use as possible. The only main colours that have been used is green and red to signify a positive or negative outcome, in line with other design standards. Each 'Title Page' has a sub header to describe what has happened to the user to help them understand the system more and know what to do.

# CONNECT TO WIIMOTE

Press 1 and 2 on the wiimote

## Start page

This is the first page/ UI the user is presented with. It has the large title that can easily be visible to the user and a sub-title containing the instructions of what the user should do to progress. There is no other texts or options in order for the software design to remain minimalistic and simple, so even less computer-literate people can also use the software.



## Connection Status

These are the connection status pages and used to indicate if the Wii remote and has been successfully linked to the computer successfully. Both pages will be displayed for a brief 2 seconds as they provide no other function other than informing the user and do not contain significant text or information. The titles are written in respective colours with 'red' for error and 'green' for success and this follows standard convention.

The Error page will lead to the software restarting so the user can re-attempt to connect to the wii remote.

The Success page will progress to the calibration page.

The style of these pages follows similar UI language to ones from the previous and following.



### Calibration Page

The calibration page has the title centred in the middle and the action that the user has to perform underneath it. There are 4 boxes of dimensions 10px by 10px. These dimensions have been chosen as there is enough area for the user to see the box from afar but small enough for the user to tap the centre of the box accurately.

After the user taps the green box it turns red and the next box in a clockwise direction turns green. This is then iterated through till all the boxes are red again.



### Calibration page Algorithm for Spot Detection and Box Activation

```
Plot top-right square red

Plot top-left square red

Plot bottom-right square red

Plot bottom left square red

Plot corresponding Main and Subordinate Text

For squares in list_of_Squares:

    Make square green

    while not IR:

        IRdata = get data

        if IRdata != None:

            IR_coord = IRdata

            IR = True

            IR_list.append(IR_coord)

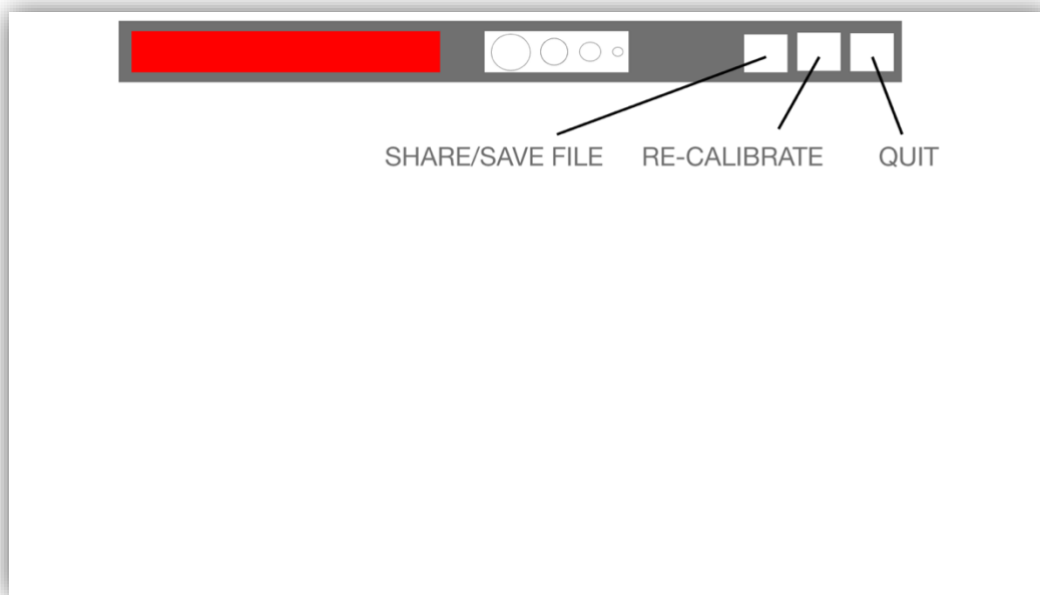
    Make square red

    Sleep(1)

Print(IR_list)
```

The code above is the pseudocode of how the box changing colour algorithm would work. First the GUI would have to be displayed with all the present squares and the text, containing the instructions.

Then the code will go through the list of all the boxes and turn each one green and take



the IR reading of that. And then save it to an list that can be used later. Then it will turn that square red and move on to the next one.

The time.sleep function is there so that if the IR pen is held for longer it wont register the same point for two squares.

At the end the Ir list has to be saved and passed on the homography at a later stage.

## Main Page

This is the page where the user will spend most of the time in the application, as this is where the interaction occurs with the software. On this page there is a tool bar at the top which contains the:

- Colour Picker
- Size Picker
- File Share/Save
- Re-Calibrate
- Off /Quit button

This is located at the top to allow for the user to utilize the most space in terms of writing and content.

The colour picker will have a select prechosen colours in the range of 10-15 which the user can select from. This will be easier than a colour wheel as selecting the same exact colour again will be more difficult and may cause frustration. This is similar with the pen size picker

## Algorithm for Taskbar

```
Method Setup_of_ToolBar ():
```

```
    Draw rectangle of x=750 and y=50 with the centre of (30,500)
```

```
    Draw colour_box()
```

```
    Draw Pen_sizes()
```

```
    Draw Buttons()
```

This code sets the code up for plotting and initiating the toolbar by plotting all the points.

```
Location = point_inside(point)
```

```
If location == colour:
```

```
    If location.sub() == red:
```

```
        Colour = red
```

```
    elif location.sub() == blue:
```

```
        Colour = blue
```

```
.....
```

```
If location == pensize:
```

```
    Pensize = location.sub()
```

This is to check if the taskbar is clicked.

This is for the class point inside to check where the pen is:

```
Class point_inside(point):  
  
    Method constructor:  
  
        If point is in range (50,100) to (1000 – 50,0):  
  
            If point in range pensize:  
  
                Return "pensize"  
  
            If point in range color:  
  
                Return "colour"  
  
        Else:  
  
            Return "canvas"
```

This class simply checks which part of the screen the point has been detected. Between (50,100) and (1000 – 75,0) is where the task bar is present so any point in this square will be a point where the tool bar is pressed.

The constants of the sub areas in the toolbox such as the pensize area and the colour area are saved previously as global. This should then be checked if the point is also inside those two sub categories.

If the pen does not come under the toolbox the only other location it could be located is the canvas where the user is able to draw the lines and diagrams this simply returns the point canvas which can then be converted and plotted.

```
Method sub():
```

```
    If pensize:
```

```
        If point is in range (...) to (...):
```

```
            Return pensize + 1
```

```
        Elif (...) to (...):
```

```
            Return pensize – 1
```

```
    If colour:
```

```
        If point is in range (...) to (...):
```

```
            Return "red"
```

```
        If point is in range (...) to (...):
```

```
            Return "blue"
```

```
        .....(Repeated for how many colours are needed).....
```

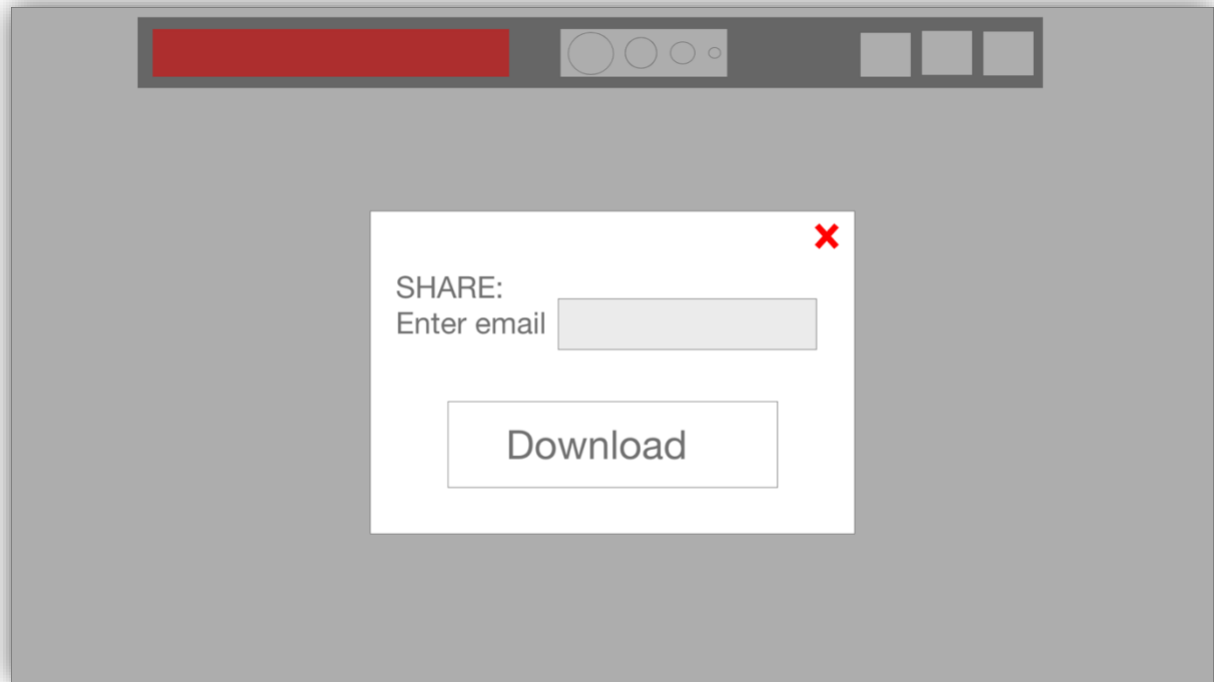
This is the method called sub under the class location, this feedback the exact thing the pen is on top of, as the pen could be on the toolbar but did not click in a valid area.

This simple checks if the point is on the pen size making it larger or making it slower and then returns this to the previous algorithm to make that change to the exact variable.

This is also the same for changing the pen colour, however this code is repetitive simply checking if the pen and the colour is was in contact with and then returning this colour.

For the other options in the toolbar they will be icons to represent each of the actions, as this is will make it more universal to understand, across languages and will also allow for a cleaner more compact UI and user experience.

This is the modal that will popup if the share button is clicked on the tool bar



from the main page.

This modal will have 2 way to interact with it the email address input box and the, download button. There is no other text or buttons to keep a simple and clean User Interface.

When the modal is active the rest of the page (background) will be dimmed and turned 'grey' this will allow for the users focus to be forced and placed onto the centre of the screen making it obvious for then to interact with the foreground and the background is currently disabled.

In order for the user to go back to what they were doing, the canvas mode, they should click the red cross button that signifies close to remove the modal and activate the background and continue using the software.

# User Feedback and Testing

To evaluate the effectiveness and the usefulness of the design of the UI and UX to the user, a test was conducted to measure the effectiveness and see if any changes were to be made.

A group of potential users were gathered, in this case teachers from the school. The selection was made in order to have a wide range of potential users, with teachers varying in:

- Height
- Computer literacy ability
- Specialties (Area of their degrees)
- Ages
- Dexterity (Left handed or Right handed)

This group would be able to simulate the group of potential users when the product is finished and see how they interact with it. This can then be observed and studied on how they interact and alterations and changes to the UI and UX can be changed.

The test will be conducted by taking the identified user into a classroom where the software will be loaded up on the connection screen, they will have access to a mouse and keyboard to allow the user to interact with the mock-up in that way if they wish and feel as if it is a desirable way, and also a Wii remote.

The users will not told any specific instructions, other than progress through the software and attempt to get it working. This will be observed and approximately timed, this can then be compared to time of which they were estimated to have taken, from this the data should show us if there are places where the users struggled to complete certain tasks.



The main task or steps they have to accomplish are:

- Press 1 + 2 on the wii remote
- Successfully follow the steps for calibration
- Change pen colour
- Change pen size
- Re calibrate
- Share a file
- Download a file
- Close the program

Since there is no working code they will just have to click the general area or location/ button and the observer (me) will tell them they are successful and can progress, while observing.

Following this a post-discussion/ interview will be conducted where the user will be asked if they faced any difficulties and if they had any changes they would want made to the software UI layout if it was up to them.

## Results

After the test there were a couple of things noticed that lead to changes in the UI:

The starting pages were relatively easy and almost all users successfully were able to find and press buttons 1 + 2 on the wii remote, so no changes were to be made there. Also all users were able to read the text clearly and efficiently and did not highlight any difficulties.

The calibration page, most users understood what had to be done, and successfully taped the green square when it appeared in the correct order. A couple of problems that were noticed was that some users, mostly the shorter ones struggled to reach the top part of the board, but were still able to manage it by tip-toeing. However due to how the calibration works (discussed later on) there is little to no way around this problem.

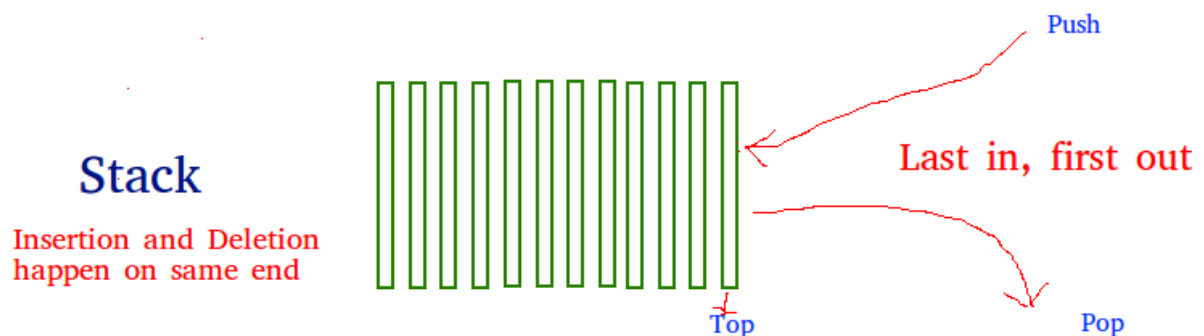
In the canvas page or the main page, similar to the problem above teachers struggled to reach the top, so in the next design/UI design the tool bar will be moved to the side to allow even short teachers to reach it, making it more comfortable to use. All teachers where able to change colours and pen sizes, the same is said for sharing and re-calibrating and closing the software, so no changes are needed to be made there.

# Smooth Line Algorithm

Due to the sensor being used having a certain refresh rate ie, a fixed sample rate or baud rate, it will take readings at certain times in fixed intervals, but the movement of the pen will be quicker than the baud rate so that it won't be able to pick up every point that the pen crosses over. This may cause a spotty line to be drawn.

Hence, an algorithm is needed to fill in the gaps between the spots to produce a smooth line. But during this between writing the pen could be briefly turned off, so we need to check if no IR point is found.

## Stack Algorithm



The stack will have a maximum depth of 2 items as only 2 points are needed to draw a smooth line. Stacks use the first in last out so that when the time is needed to generate the two points can be popped out and then the smooth line algorithm can be implemented.

```

Class Queue:

    Method addItem:

        If Queue.length is not 2:

            Queue = Queue.append(Point)

    Method Pop:

        Temp = Head

        Head = Head + 1

        Return Queue[Temp]

    Method Wipe:

        Queue[head] = None

        Queue[head+1] = None

```

As seen above is the algorithm for the stack of putting an item on top of it and removing an item. There is also a method called Wipe which is used to get rid of the points that are currently stored in the queue if the IR pen is turned off. This method will only be run if None is detected by the IR sensor

```

Get point

If point = None:

    Queue.wipe()

Else:

    Queue.additem(point)

If Queue.length() = 2:

    Smoothline(Queue)

```

This will be run once the queue is full so the two points can be popped off and then be used to draw the smooth line between them.

## Smooth Line Algorithm

One the two points are popped off the stack these two values will then be passed into the function SmoothLine() which will find all the intermediate points between the two points given using mathematical equations of finding the gradient and all the sub points it has to plot between each one.

```
Method SmoothLine(point1, point2):

    #Calculates the change in y and x

    dx = point1(x) - point2(x)

    dy = point1(y) - point2(y)

    positiveX = sqrt(dx*dx)

    positiveY = sqrt(dy*dy)

    #Calculates the distance between the two points

    distance = max(positiveX, positiveY)

    #Calculates the sub-points between the two points

    For subpoint in range(distance):

        x = int(point[0] + float(subpoint) / distance * positiveX)

        y = int(point[1] + float(subpoint) / distance * positiveY)

        plot(x, y)
```

As we can see the first section of code take the two points and first calculates the distance between the two points, it then uses a for loop to loop through each sub integer points between them and generates an intermediate point which is then passed onto the GUI circle plot function.

## During-Development Testing

Test for	Input	Validation rule used	Explanation	Justification	Success Criteria Code to fulfil
Responsive Design	Resizing Screen Size	Visual Check	This is to make sure that the software will work on a variety of screen sizes and then	This is so that when users use the software no matter their setup that their program will run and work as efficiently and effectively as another machine in terms of UX	SC2
Font Compatibility	Font Families and Saved Fonts	Data Check	This is to make sure that the software can still function if the user has changed default system fonts and	This is to ensure that what ever font the user has installed on their system if it is different from the system default or they have removed the system default font the	SC2

				instruction will still be shown	
--	--	--	--	------------------------------------	--

For the GUIs testing and development there will be simple tests that can affect the way it displays to the user, that could affect the UI and UX.

Since this module has no input or output there is no real in depth testing required for this section.

In terms of variable changes and things that can vary is very little as the display has a set UI and design, the only things that cause the variables to change is the hardware or systems software settings, such as resolution/aspect ratio/ display type and size. This will make it harder for doing a logical search through to find any errors. Hence a simpler quicker but more random method will be used, by running the code with different monitors/projectors/displays to see if any discrepancies appear that requires the code to be tweaked in order for the UI to stay exactly the same though out all of the devices, and runtime environments.

## Homography (Calibration Algorithm)

Homography is used to make sure that both cameras (projector and IR sensor) share the same co-ordinate system as they grids they have may not always correspond to each other as the Wii remote has the flexibility to be placed anywhere.

This algorithm was used to make sure that the co-ordinate for example (1,100) would reference the exact same point of another co-ordinate of the other camera. This would allow for the point picked up by the IR sensor to reference the same point on the projector so where that pen is placed the dot will be in the same location. The way each of the co-ordinates are converted is using a mathematical equation that handles with the conversion (referenced later in the report).

For this system to work the 4 extremities of the IR sensor (ie its total view finder) must cover the projector area (area where the text will be written). This is because if it was the otherwise the other way around the pen will be out of the frame and its location will be impossible to be determined. Hence the statement that the overall area of the IR sensor covered is greater than the projector area covered.



Due to the design of the sensor that it can be placed anywhere the projectors grid from the point of view of the IR camera will have distortion and various translations (assuming there is not lens distortion so a pinhole camera is used). This can cause the projectors view to appear in perspective, such as the diagram below.

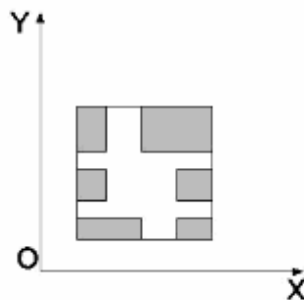
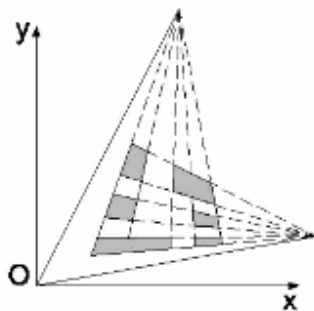
Where the biggest x and y axis is the grid of the IR sensor and the pattern is the projector/screen display.

Using the formulae below:

$$\begin{vmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -y_1X_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -y_2X_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3X_3 & -y_3X_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4X_4 & -y_4X_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1Y_1 & -y_1Y_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2Y_2 & -y_2Y_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3Y_3 & -y_3Y_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4Y_4 & -y_4Y_4 \end{vmatrix} \cdot \begin{vmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{vmatrix} = \begin{vmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{vmatrix}$$

Where all the lowercase x and y refer to the IR camera and the uppercase X and Y refer to the projector.

All of the co-ordinated would be known as the extremities of the IR sensor would



default to the all 4 corners that the sensor data can give. The extremities of the projector can be determined by making the user touch the pen at all 4 points of the projector, allowing for this data to be determined.

Following this the equation can be rearranged to solve for a,b,c,d,e,f,g,h giving the equation of

$$X = \frac{ax + by + c}{gx + hy + 1} \quad Y = \frac{dx + ey + f}{gx + hy + 1} \quad \text{the}$$

transformation of the girds where each letter represents:

- a - fixed scale factor in X direction with scale Y unchanged.
- b - scale factor in X direction proportional to Y distance from origin.
- c - origin translation in X direction.
- d - scale factor in Y direction proportional to X distance from origin.
- e - fixed scale factor in Y direction with scale X unchanged.
- f - origin translation in Y direction.
- g - proportional scale factors X and Y in function of X.
- h - proportional scale factors X and Y in function of Y.

Once these variables are found the equations:

Can be used to translate co-ordinated where values x and y are given by the IR sensor, and a,b,c,d,e,f,g,h are known from the previous equation, the values of X and Y which reference the projector co-ordinate system can be used.

## Data Flow Diagrams

The program will consist of 3 main objects that will be responsible for the functioning of the software.

- GUI (Module simplification for easier referencing)
- Wiimote Data (Bluetooth data handling)
- Homography Calibration
- Main code (Contains the logical algorithmic data flow)

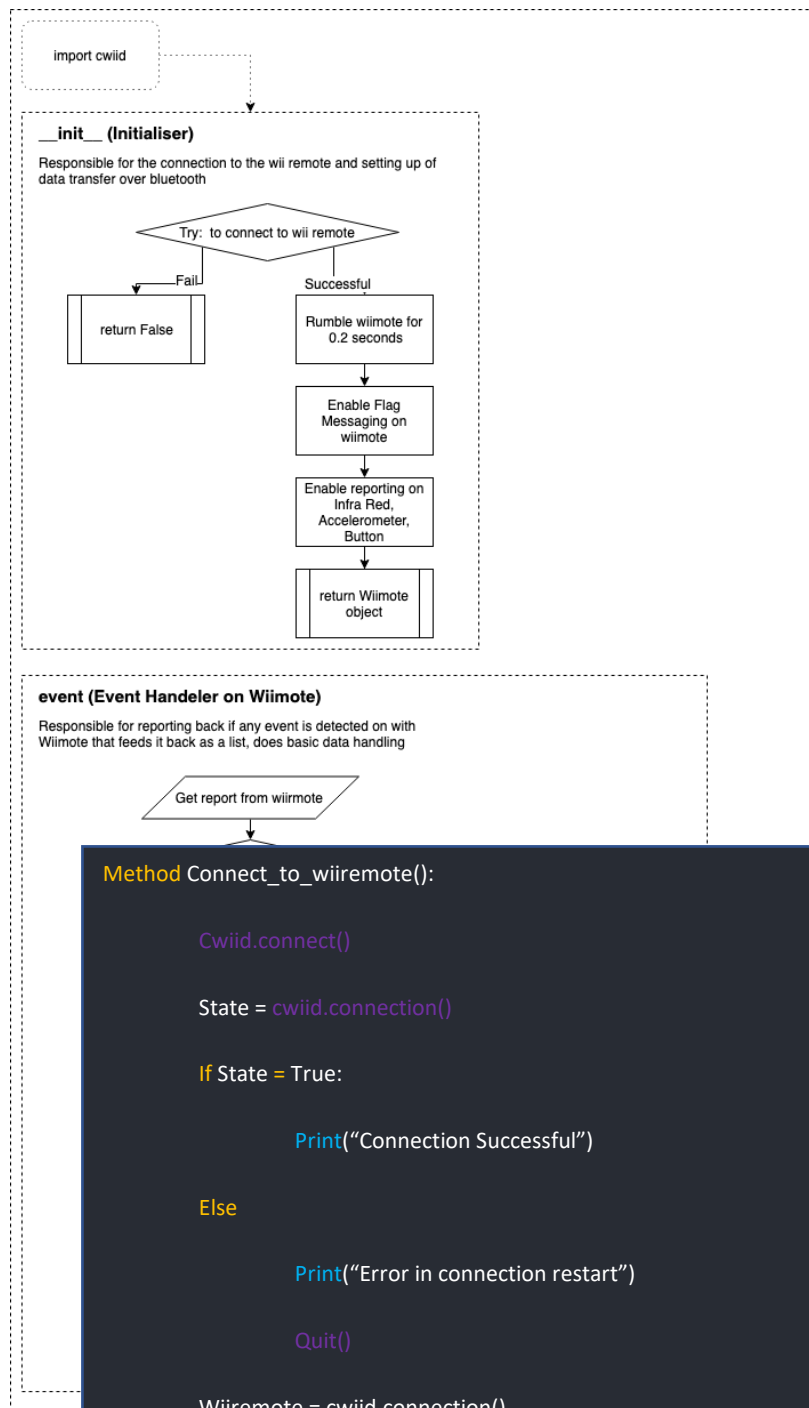
Each module will be responsible for its tasks as stated above and previously and will be used by referencing by the main code to carry out individual tasks. This will allow the code to be sectioned, making it easier for new code to be added and debugging if there was an error in the code.

## Wii Remote Data Handler

This module is developed on top of the library called cwiid which is a pre-programmed library that handles the connection and Bluetooth socket and data stream from the wii remote.

## Initialiser

This is responsible for the starting the connection between the wiiremote. The cwiid library throws an error if the connection failed causing the use of the try and except clauses. When successful the wiiremote rumbles to make it easier for the user that the connection has happened, the program then sets up the reporting mode for the wiiremote and returns the object to be used later.



## Event Handler

This part of the code is responsible for organising the data from the wiiremote and checking if there is any interaction from it. Its main things its checking for is IR spots to see if the pen was detected and where it was, and the home button was pressed to signify a termination of the program.

## Initialiser Algorithm

```
Method Connect_to_wiiremote():

    Cwiid.connect()

    State = cwiid.connection()

    If State = True:

        Print("Connection Successful")

    Else

        Print("Error in connection restart")

        Quit()

    Wiiremote = cwiid.connection()

    Set up reporting and error handling code

    Enable IR camera and Button Event Handelters

    Return wiiremtote
```

This is the code algorithm for the module which allows the program to connect to the Wii remote.

## Event Handler Algorithm

```
Method Event_Handler(wiiremote):  
  
    Get Bluetooth Data  
  
    For data in message:  
  
        If data.button.home_button is pressed:  
  
            Print("Quting")  
  
            Quit()
```

This is the algorithm to show to retrieve data from the wiiremote and filter it out and convert the data types and the layout into a way to program can understand it.

## During-Development Testing

Test for	Input	Validation rule used	Explanation	Justification	Success Criteria Code to fulfil
Connection with background noise from Wii remotes/ Bluetooth devices	Bluetooth Connections	Connection successful and stream of data  Erroneous Check	This makes sure that the connection is secure and filter out white noise	This is to make sure that the data can be streamed without it cutting out or causing data streaming/ connection difficulties for a more real world test and usage	SC4 SC1
Range (Distance) of Connection	Bluetooth Connection	Connection successful and stream of data  Range Check	So that Bluetooth and sufficient range to cover the screen but still have a connection to the computer	This is so that the Bluetooth range can be tested to make sure it works over a large enough range so it can be placed in a comfortable position.	SC4 SC1
IR sensor reliability and strength	Sensor Test	Check for range of the IR	This is to check if the sensor is able	This is to make sure that even at a far distance it is	SC4 SC1

		sensor to spot IR points  Erroneous Check  Range Check	to see an IR point at a distance that is suitable	able to sense a IR point and pick up the strongest one if there is multiple points of varying strengths.	
--	--	--	--	--	--

During development testing for this module of code is mainly to check that there is a data connection between the wiiremote and the computer. Mainly the tests of this module have to come from reliability of the data such as:

- How is accurate is the positioning of the IR point
- Distance of communication
- Accuracy of accelerometer

In terms of input and output checking there is very little to be done here as this will be providing the input to the rest of the code.



# Homography Calibration

Following the equations designed and explained previously, pseudo code was written in order to map out variables and matrixes in python for the computation. There are two parts for this, the first part being the equation for getting the translation matrix from two sets of 4 points and the second being the conversion of one point to another point on the other grid.

This functions can assume all data input has been sanitized and checked before.

## Obtaining the IR maximum Co-ordinates

```
quad1 = np.array([[0, 0], [100, 0],[100,100],[0,100]])

#maps the array to individual variables
x1 = quad1[0,0]
y1 = quad1[0,1]

x2 = quad1[1,0]
y2 = quad1[1,1]

x3 = quad1[2,0]
y3 = quad1[2,1]

x4 = quad1[3,0]
y4 = quad1[3,1]
```

Using numpy and python, we can obtain the 4 coordinates in as a two dimensional array, with the first coordinate being in the top left then top right then bottom right then bottom left in a clock wise direction.

Using this the 2 dimensional array called quad 1, which is passed into the function. The 2 dimensional array is broken up into individual variables. Which can then be referenced more easily later on even if this makes the code less compact.

### Obtaining the Projector Co-ordinates

```
#Co-ordinates that are constant from the projector (display maximum)
#           TL      TR      BR      BL
quad2 = np.array([[100, 100], [200, 100],[200,200],[100,200]])

#maps the array to individual variables
X1 = quad2[0,0]
Y1 = quad2[0,1]

X2 = quad2[1,0]
Y2 = quad2[1,1]

X3 = quad2[2,0]
Y3 = quad2[2,1]

X4 = quad2[3,0]
Y4 = quad2[3,1]
```

This co-ordinates follow a similar structure as the previous section of code. The co-ordinates stored in a 2 dimensional array in a clockwise fashion.

Quad 2 is constant and is the same as the window size, and will not change, therefore can be stored inside the function and does not have to be passed in.

Quad 2 is also broken up into individual co-ordinates with a capital letter to show that it is from the Projector co-ordinate system rather than the IR co-ordinate system which uses lower case letters.

## Matrix Mathematical Algorithm

```
matrix1 = np.array([[x1, y1, 1, 0, 0, 0, -x1 * X1, -y1 * X1],
                    [x2, y2, 1, 0, 0, 0, -x2 * X2, -y2 * X2],
                    [x3, y3, 1, 0, 0, 0, -x3 * X3, -y3 * X3],
                    [x4, y4, 1, 0, 0, 0, -x4 * X4, -y4 * X4],
                    [0, 0, 0, x1, y1, 1, -x1 * Y1, -y1 * Y1],
                    [0, 0, 0, x2, y2, 1, -x2 * Y2, -y2 * Y2],
                    [0, 0, 0, x3, y3, 1, -x3 * Y3, -y3 * Y3],
                    [0, 0, 0, x4, y4, 1, -x4 * Y4, -y4 * Y4],
                    ])

matrix2 = np.array([[X1],
                    [X2],
                    [X3],
                    [X4],
                    [Y1],
                    [Y2],
                    [Y3],
                    [Y4]])
```

This section of code maps all of the individual co-ordinates from the projector co-ordinate system (quad2) and the IR co-ordinate system (quad1) into the matrix that can be used later on for the math to form the algorithm.

The co-ordinates could be directly referenced by each was saved as an individual variable as this was easier to debug if there was an error in the code and improves maintainability is any code needed to be changed.

```
mat4 = np.linalg.inv(matrix1)
matrix3 = np.matmul(mat4, matrix2)
```

The final section of code rearranges the mathematical equation stated above by inverting matrix1 into mat4.

Then as the final step matrix2 and mat4 are multiplied together to form matrix 3 which is the translation matrix that can be used to convert any point between the two co-ordinate systems.

Matrix3 is then returned back to the main section of code and that can be passed onto the second stage of homography.

**Translation Matrix**

```
translationParameters = np.array([[1],
                                   [0],
                                   [0],
                                   [100],
                                   [1],
                                   [0],
                                   [0],
                                   [0]])

#Saves the vector translation parameters to variable constants
a = translationParameters[0,0]
b = translationParameters[1,0]
c = translationParameters[2,0]
d = translationParameters[3,0]
e = translationParameters[4,0]
f = translationParameters[5,0]
g = translationParameters[6,0]
h = translationParameters[7,0]
```

This takes the matrix generated in the previous section of code which is stored in a one by 8 array,

This is then translated into individual variable from the matrix as this will be easier to reference later on the code. The variables chosen for the individual addressing of each number is the same as the variables in the equation mentioned above.

TranslationParameters would be passed into the function so the constants currently stored in it would change each time the program is run, due to the change of positioning of the IR camera.

### Translation Matrix

```
#the co-ordinate that needs to be converted
inputCoordinate = np.array([[2,54]])

#saves the co-ordinates to seperate variables
x = inputCoordinate[0,0]
y = inputCoordinate[0,1]
```

This array which is two dimensional stores the co-ordinate which would have been received by the IR sensor co-ordinate system that needs to be converted using the translation parameters to a co-ordinate in the projector system.

The co-ordinate that will be passed into the function will be then be broken down into 2 individual variables to make it easier to reference later on. This uses lowercase letters as this co-ordinate is referenced on the IR system which follows the coding style on the previous function for generating the translation matrix.

### Conversion function and equation mapping

```
#used Homography calculation to convert co-ordinate

X = ((a * x) + (b * y) + c) / ((g * x) + (h * y) + 1)
```

This takes the translation matrix and the inputted co-ordinate and using the formulae states previously it converts it to find X and Y which are the capital variables as they reference the Projector screen. These individual variables are then grouped together in an array called outputCoordinate which is then returned by the function so that the point can be placed on the screen.

### During-Development Testing

Test for	Input	Validation rule used	Explanation	Justification	Success Criteria Code to fulfil
Generation of Matrix 1	Extremities of the Ir co-ordinate system inputted by the user	Correct matrix is formed and printed out	This is to make sure that the matrixes are generated properly and accurately	This is so that the homography calibration works properly, otherwise there may be a shift right or inverse if the code is written wrong	
Generation of Matrix 2	Extremities of Users Display	Correct matrix is formed and printed out			
Matrix Calculation is accurate and correct	Matrix 1 and Matrix 2	Calculations will be cross checked by another algorithms answers		This is needed to make sure that the calculations are correct in order for the calibration of the two systems to be correct	

During these tests, the data will be manually inputted for during development testing purposes as to the wiiremote getting the data. Most of the answers will have to be cross checked with hand worked out matrixes.

# Validation

The only form of validation required in this part of the email, that is required to be typed in by the user to share the document. The main way to check if an email is valid is:

- Check for an “@” symbol in the string
- Check that there is at least 1 “.” in the string
- Make sure that there is no other symbols In the string such as “@£\$%^&\*()”
- Check that there is characters before the “@” symbol
- Check that there is a “.” after the “@” symbol
- Check that there are characters after the “@” symbol
- Check that there are characters before the “.” symbol
- Check that there are characters after the “.” symbol

These tests are used to validate that the email entered is correct and makes syntactical sense to the computer. This will allow for any typing errors or spelling errors to be picked up by the client side before sending the data off to the server, reducing the server load of handling the emails.

This was used to allow the user to quickly and immediately correct any mistake they may have made when initially typing it in. This makes the user experience more efficient as if the error was processed by the server, the error would take significantly longer to flag the error then notify the user and then let the user fix the mistake. This will be more time consuming and therefore cause an overall worse experience for the user. Hence the use of the client-side validation and testing to ensure that the email entered is correct



## Email Validation Algorithm

```
Method email_validation(email):  
  
    For character in email:  
  
        If character == "@":  
            At = email.position(character)  
  
        If character == ".":  
            Dot = email.position(character)  
  
    For character in the range between (0 and at):  
        Check if there are alpha-numeric characters present  
  
    For characters in the range between (at and dot):  
        Check if there are alpha-numeric characters present  
  
    For characters in the range between (dot and last character):  
        Check is there are alpha-numeric characters present  
  
    If all statements are true:  
        Return "email is valid"  
  
    Else:
```

# Data Dictionary and Variable Table

In the following table will be the key variables in the table broken up, with its data type, and its usage and what it stores. Each table will refer to a different class/module of the program.

## Main Code

Relies on Libraries and Modules:

- Wiiremote
- Pygame
- GUIcomponents
- Time
- Sys

Identifier	Data Type/ Structure	Description	Justification
Pygame	Object	Used to control all GUI elements	Required to use pygame
appDisplay	Object	Used as a canvas to show elements of the UI	Required for pygame

gameExit	Boolean	Shows that the app is in use	Used to generate an infinite loop for GUI
calibrated	Boolean	Shows if the IR camera is calibrated to the projector	Used to run a subloop to ensure calibration
Title_font	Object	Contains the font text settings	So that the title aesthetics and data does not have to be entered manually each time
Caption_font	Object	Contains the font text settings	So that the title aesthetics and data does not have to be entered manually each time
Wii	Object	Contains the object to communicate via Bluetooth	Required for cwiid
IR	Boolean	Shows if an IR spot has been found	To run a separate loop to check for what those co-ordinates are
IRdata	Truple	Contains the co-ordintes of the IR spots	So that the truple can be iterated through and each

			spot can be deciphered
--	--	--	---------------------------

## Wiiremote

Relies on Libraries and Modules:

- Cwiid
- Time
- Pygame

### Init (Function)

Identifier	Data Type / Structure	Description	Justification
Wiimote	Object	Used to create a Bluetooth socket for the communication	Used to communicate with the wiiremote

### Event (Function)

Relies on these inputs:

- Wiiremote

- Pygame

Identifier	Data Type / Structure	Description	Justification
Messages	Array	Stores one packet stream of the data from the wiiremote	This can be used later to analyse data
Sources	Truple	Contains the data of the IR spots	So that it can be analysed later and interated through
Found	Boolean	Shows if there is a IR point that meets the minimum threshold is found	So it knows the location of the spot that is found
Position	Array	Shows the co-ordinate that the point is at	So that it can be displayed on the screen

## Homography

Relies on Libraries and Modules:

- Numpy

### Translation Parameters

Relies on these inputs:

- iRcameraCoordinates

Identifier	Data Type / Structure	Description	Justification
iRcameraCoordinates	Np.array	Used to store the 4 co-ordinates of the projector in the IR camera system	So it can be using in the equation later on
x1	Integer	Top left X co-ordinate	IR co-ordinate
y1	Integer	Top left Y co-ordinate	IR co-ordinate
x2	Integer	Top right X co-ordinate	IR co-ordinate
y2	Integer	Top right Y co-ordinate	IR co-ordinate
x3	Integer	Bottom right X co-ordinate	IR co-ordinate
y3	Integer	Bottom right Y co-ordinate	IR co-ordinate
x4	Integer	Bottom left X co-ordinate	IR co-ordinate
y4	integer	Bottom left Y co-ordinate	IR co-ordinate
ProjectorCoordinates	Np.array	Used to store the 4 co-ordinates of the projector	So it can be using in the equation later on

X1	Integer	Top left X co-ordinate	Projector co-ordinate
Y1	Integer	Top left Y co-ordinate	Projector co-ordinate
X2	Integer	Top right X co-ordinate	Projector co-ordinate
Y2	Integer	Top right Y co-ordinate	Projector co-ordinate
X3	Integer	Bottom right X co-ordinate	Projector co-ordinate
Y3	Integer	Bottom right Y co-ordinate	Projector co-ordinate
x4	Integer	Bottom left X co-ordinate	Projector co-ordinate
y4	integer	Bottom left Y co-ordinate	Projector co-ordinate
Matrix1	Np.array	Stores all the values as a matrix	This is so it can be used by the equation
Matrix 2	Np.array	Stores all the values as a matrix	This is so it can be used by the equation
Matrix2inverse	Np.array	Stores the inverse of matrix 2	
TranslationParameters	Np.array	Stores the translation parameters that are generated using the equation	This is so it can be used to convert points between the two co-ordinate systems

## Coordinate Converter

Relies on these inputs:

- input co-ordinate
- Transplation parameters

Identifier	Data Type / Structure	Description	Justification
A	Float	fixed scale factor in X direction with scale Y unchanged	
B	Float	scale factor in X direction proportional to Y distance from origin	
C	Float	origin translation in X direction	



D	Float	scale factor in Y direction proportional to X distance from origin	
E	Float	fixed scale factor in Y direction with scale X unchanged	
F	Float	origin translation in Y direction	
G	Float	proportional scale factors X and Y in function of X	
H	Float	proportional scale factors X and Y in function of Y	

## Post Development Testing

After the final development of the program and when it is finished, certain tests will be carried out on top of the during development tests, to ensure that the software runs according to the specification and design brief.

These tests are for a holistic view and test of the code not to test individual sections or modules but more of a real-world test, to find errors that may not be found in normal data testing.

There are two scenarios or models used to test this code to ensure it is successfully met:

# Stakeholder Testing

Similar to the test done to the UI interface. This will be when a series of potential users will be given the software and given a set of tasks to complete. The user will then be observed if they can complete these tasks successfully and to see how quickly they can accomplish any tasks.

During this as well one of the tasks should be on let the user try breaking the system, by ways they seem appropriate e.g spamming the keyboard or clicking the same button multiple times. If they manage to break the software then the errors will be noted and how they did it, so that it can be recreated in a more controlled environment next time in order for the problem to be diagnosed and solved.

After the task have been attempted and completed a follow up interview will have taken place where questions about the user experience will be asked to see what they found difficult and what features they would want in future versions in order to make it more user friendly.

# Reliability Testing

This will be to test if my code is robust. This will be carried out over a longer period of time to see if the hardware still function properly with the code. Tests user this section includes:

- Duration Testing
  - This is to test that the software and code will work properly even if the software is being used for a long period of time for example 6 hours straight, to see if any errors appear. To simulate real world usage.

- Environment Testing
  - This is similar to the tests carried out for the Wii remote module but rather than using it to determine benchmarks and how far it can be pushed these will be done with more real world scenarios. Taking the software and using it in a dark room, or a very small room for example. To emulate how the software will be used when it is deployed.
- Computer Hardware Testing
  - This is to see what computer hardware setting it requires to run efficiently and where the threshold point is. Factors that will be changed is the RAM amount, CPU architecture, Caches size, Bluetooth chips and protocols.

Test for	Input	Validation rule used	Explanation	Justification	Success Criteria Code to fulfil
Making sure the data point are being plotted accurately on the screen	IR point	Making sure where the pen is the plotted point is in the same position	This is to make sure that the homogrphay and calibration is accurate	This is so that there is accurate plotting making the usage of the application better and more usable otherwise, it will make the software look buggy and inaccurate	
Program does not randomly crash	Overloading the system with spam data of IR points	Making sure the System can filter out false points  (Stability Testing)	This is to make sure the system is reliable	In order for a system to be used in a wider context, it needs to be reliable and usable for the user, as the user cant use the software if it keeps on crashing	

# Development

In the development stage the code will be made using a top down approach where the classes, data structures and variable names are created. And then a bottom up approach will be taken where the code is added and each of the classes functions and procedures are added accordingly.

The language chosen was Python as it was the language I was most familiar with. This was also an ideal language as it has the ability to be object orientated and procedurally typed, giving that level of flexibility and the option to switch between the two depending on the code and the exact structure. Python is an interpreted language which allows for prototyping to be done more efficiently and quickly as it could potentially end up being a large program and having to compile the code each time will be strenuous with the waiting for compiling while trying to develop it.

Each section as stated previously with the exception of the GUI section will be developed individually as classes, procedures and functions. This is so that these classes can be independently developed and tested, enabling error finding to be made easier as the syntax or logic error can be narrowed down to the specific class. This will also make it better for it to be developed in the future as a fully functional project as this will make it easier for another developer or team of developers to develop the code as it would be layout in classes and procedures

# Sensor Interfacing

This section is to do with the interfacing with the Wii remote to obtain its data and filter it out till we are able to obtain the IR data. The Wii remote has to be capable of 2 way communication. The main way of communication will be using Bluetooth, a prebuilt library called cwiid can help establish this connection.

## Establishing Wii Remote Connection

This was first done to achieve what would be the first steppingstone of connecting to the Wii remote. This was done to ensure that the Bluetooth connection works as intended to and is able to connect to the Wii remote.

```
import cwiid
print "Press 1 + 2 on the wiiremote"
#attempts to connect to the wiimote
wiimote = cwiid.Wiimote()
print wiimote
|
```

Currently since the GUI section will be developed later, therefore print statements will be temporarily used to indicate to me/ the user/ the testers of the instructions via the terminal prompt.

## Testing

```
Press 1 + 2 on the wiiremote  
<cwiid.Wiimote object at 0x76a535d8>
```

The instruction line was printed successfully and the object was generated when the Wii remote was connected showing that a Bluetooth connection has been established between the two systems.

## Data Transmission and Capturing

In order for the remote to send data to the computing the computer need it to be able to prompt it for data. Also the Wii remote has certain outputs such as a rumble mode and LEDs, which can be used as output and a UI from the Wii remote to signify connection and battery percentage.

This code was developed from the code in the previous section:

```
#Sets up reporting mode  
wiimote.enable(cwiid.FLAG_MESG_IFC)  
wiimote.rpt_mode = cwiid.RPT_ACC | cwiid.RPT_BTN | cwiid.RPT_IR  
  
wiimote.rumble = 1  
time.sleep(0.2)  
wiimote.rumble = 0
```

Initially the Wii remote has to be put into a reporting mode as this can be used so that the Wii remote only records and transmits some data there are 3 settings for this Accelerometer, Buttons, IR. While all 3 may not be used in the final code this will be kept this was as to testing purposes and so that we can understand the Wii remotes data better.

The second part was the rumble section what will make the remote rumble. When wiimote.rumble is set to 1 it starts and when set to 0 it stops. A time.sleep function was also added to make it vibrate for a set period of time.

## Testing

No output of the code changed from previously as expected, which is successful. The Wii remote did vibrate for .2 of a second which is also a success as this was expected from the code.

```
while True:
    #Gets the data output of the wiiremote
    messages = wiimote.get_mesg()
    print messages
```

Next an infinite loop was created using the while True statement so that this section of the code will always be run.

The variable messages will store the current slice of data the Wii remote has, and then output in the print statement.

## Testing

```
[(1, 0), (2, (113, 110, 139)), (3, [None, None, None, None])]
[(2, (117, 114, 144)), (3, [None, None, None, None])]
[(2, (116, 111, 141)), (3, [None, None, None, None])]
[(2, (116, 111, 141)), (3, [None, None, None, None])]
[(2, (116, 112, 143)), (3, [None, None, None, None])]
[(2, (115, 112, 142)), (3, [None, None, None, None])]
[(2, (115, 111, 140)), (3, [None, None, None, None])]
[(1, 4), (2, (114, 112, 142)), (3, [None, None, None, None])]
[(1, 12), (2, (117, 109, 137)), (3, [None, None, None, None])]
```



This is the slice of the data that was received by the computer. Since there was success full print messages shows that there is successful data transmission. Looking at one data slice.

```
[(1, 12), (2, (117, 109, 137)), (3, [None, None, None, None])]
```

We can see that the first bit refers to the button that is being pressed, the second with the accelerometer data (X, Y, Z), and the third is the IR data. This data is being sent and stored as a tuple. This data can be filtered out using if statements to only get the data that is needed at that particular time.

```
for mesg in messages:

    #Detects whether HOME isheld down and if they are it quits the program
    if mesg[0] == cwiid.MESG_BTN:
        if mesg[1] & cwiid.BTN_HOME:
            print "Ending Program"
            quit()

    #Detects whether there is any visable IR points
    elif mesg[0] == cwiid.MESG_IR:
        sources = mesg[1]
        found = False
        #Loops through all the found IR points
        for spot in sources:
            if spot:
                #Converts the grouped points into individual points
                position = spot["pos"]
                found = True
        if found:
            #Return the co-ordinate of the point as a tuple
            print position
```

This was the filtering used to remove all the excess data such as the other button presses which will be redundant as they don't have to be used. Only the home button was used as an escape function to close the application.

The second elif statement is used to first only get the IR points and then check if there are any present IR dots. If there are then it proceeds to loop through the found spots and individually selects them and prints out the co-ordinates as a tuple on the terminal prompt.

## Testing

```
(338, 724)
(337, 722)
(485, 602)
(490, 601)
(500, 570)
(499, 571)
(518, 471)
(518, 472)
(504, 515)
(471, 519)
(471, 512)
(493, 597)
(495, 596)
(495, 597)
(513, 535)
(513, 538)
(499, 600)
```

This shows the point at which it found the IR spot and the co-ordinate it had on the grid. Using the data we can see it has a maximum value of 1000 and a minimum value of 0. This makes it fairly precise for what our application requires however still has a possibility of it not working. But this step is considered a success as IR data is being received from the Wii remote.

# Final In-Development of Sensor Interfacing

As each of the individual sub sections of this section have been developed, it now has to be reorganised into its final structure. Where the large section of code is broken up into its predefined sections as previously stated in its design section:

- The connection to the Wii remote
- The event retriever of the current data slice

```
def init():

    print "Press 1+2 on the Wiimote now"
    #attempts to connect to the wii remote
    try:
        wiimote = cwiid.Wiimote()
    except:
        #returns a False or None object to signify a unsuccessful conection
        return False

    #Rumble to indicate a connection
    wiimote.rumble = 1
    print "Connection established"
    time.sleep(0.2)
    wiimote.rumble = 0

    #Sets up reporting mode
    wiimote.enable(cwiid.FLAG_MESG_IFC)
    wiimote.rpt_mode = cwiid.RPT_ACC | cwiid.RPT_BTN | cwiid.RPT_IR

    return wiimote
```

This is the init function that attempts to connect to the wii mote. This attempts to connect within the try statements and if successful saves the object to the variable Wii mote. If there is not a successful connection then it returns the object False where the connection was a failure.

This then rumbles the wii remote so the user can feel/see/hear that the connection is successful. It then sets the wii remote up in the correct reporting mode for it to obtain the required data.

```
def event(wiimote,pygame):  
  
    #gets the slice of data from the wiiremote  
    messages = wiimote.get_mesg()  
    #itterates through the messages  
    for mesg in messages:  
  
        #Detects whether HOME is held down and if they are it quits the program  
        if mesg[0] == cwiid.MESG_BTN:  
            if mesg[1] & cwiid.BTN_HOME:  
                print "Ending Program"  
  
        #Detects whether there is any visable IR points  
        elif mesg[0] == cwiid.MESG_IR:  
            sources = mesg[1]  
            found = False  
            #Loops through all the found IR points  
            for spot in sources:  
                if spot:  
                    #Converts the grouped points into individual points  
                    position = spot["pos"]  
                    found = True  
            if found:  
                #Return the co-ordinate of the point as a tuple  
                return position
```

This was the second function in this class. This was needed so that it would fetch the data slice of the current IR co-ordinates and returns the position of it.

This also has a section of code on the if the home button is pressed then it will terminate the software and quit out of the application.

# Sensor In-Development Testing

From the Development Testing in the Design section, a series of test will be carried out to make sure that this section and module of code works efficiently and correctly. The tests will be carried out are:

- Testing of Connection with Various background noise and interference
- Range of Detection (Of connection and IR detection)
- IR sensor and reliability

## Testing of Connection with Various background noise and Interference

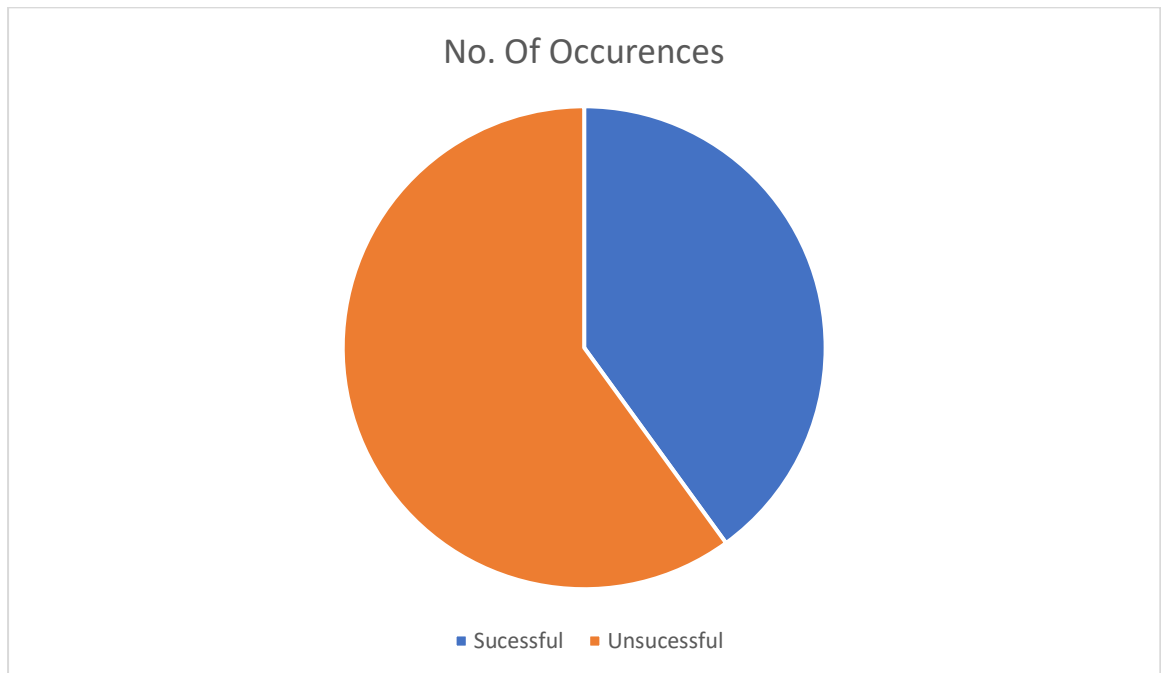
This test was simulated by running the connection code with various Bluetooth devices either:

- Trying to connect to the wii remote
- Sitting Idle with Bluetooth receiving and transmitting on

Different Bluetooth devices was used, such as iPhones, Android Phones, Laptops, Smart watches. These devices were selected to simulate the real environment as these devices would be commonly found in a school and/or similar environment, giving the most realistic test environment.

The code was then run 50 times with each type of Bluetooth connection. Making overall 100 times run. It was done this many times as to obtain clear and accurate picture of each scenarios and remove and erroneous or random errors that may have appeared in while the tests where run causing unexpected results.

### Devices Trying to connect to the Wii Remote

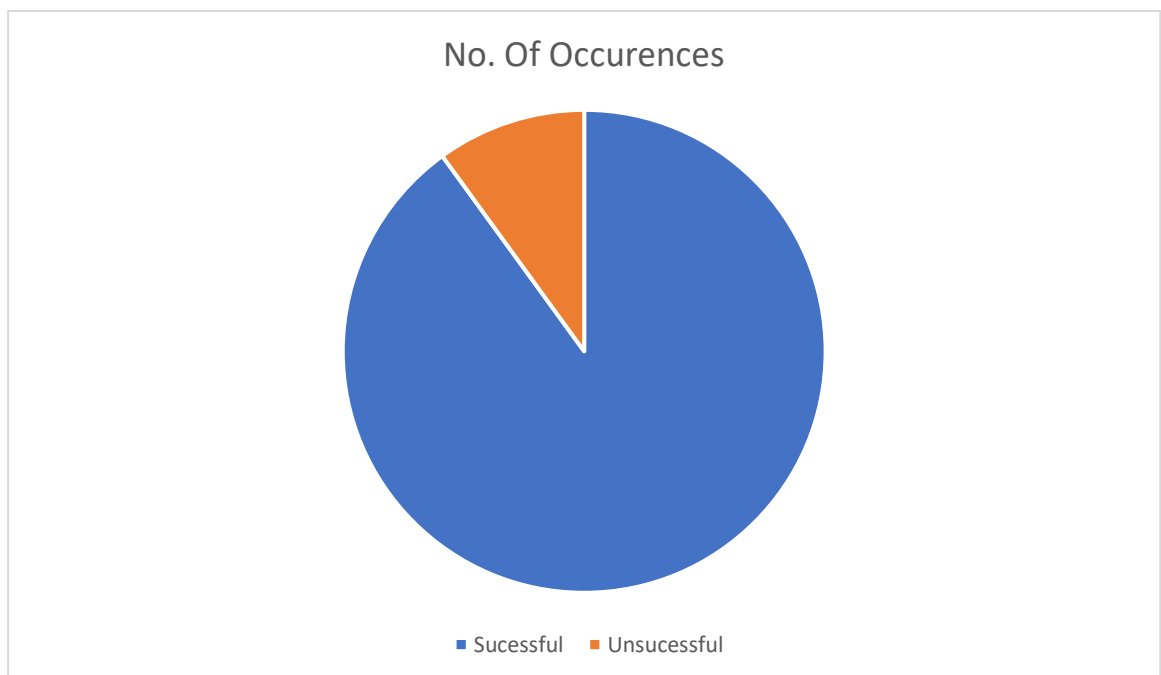


From this we can see that overall the connection was more unsuccessful then successful, which was not the result that we were hoping for.

This was due to the fact that the wii remote would end up connecting to another laptop or device causing it being unable to connect to the raspberry pi on which the code was running.

This should be noted that in order to increase connection success rate that other Bluetooth devices shouldn't not try to simultaneously connect to the wii remote, as this will cause errors.

### Sitting Idle with Bluetooth receiving and transmitting On



With idle devices around the wii remote the success rate of connection increased dramatically. Only 5 of the 50 connections failed. This gives a success chance of 90% of connection which is satisfactory and successful enough according to the specification of SC1 and SC4.

This means that the code and software will work in normal classroom scenarios with idle and on Bluetooth devices being in the vicinity of the room.

## Range of Detection (Of connection and IR detection)

This is to test the extremities of the wii remote, by pushing it too its limits to see what are the maximum ranges it can cope with. The main varying factor is the distance from the Bluetooth receiver for the connection, and distance from the IR point for the IR detection.

There are two tests that have to be carried out under this heading:

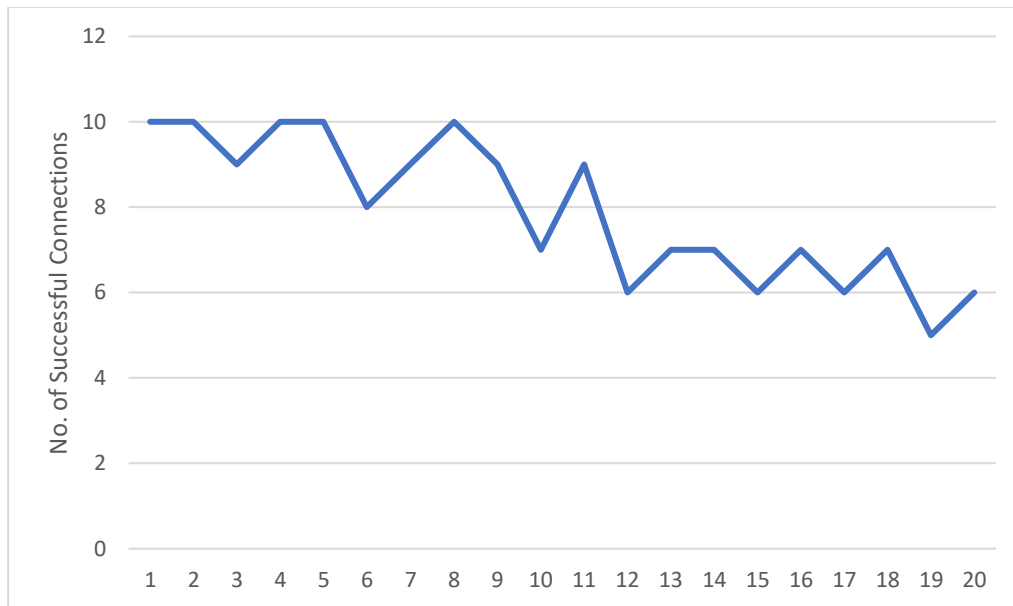
- Range of Connection
- Range of Detection of an IR point

### Range of Connection

The independent variable that will be changed and measured is distance varying from 1m to 20m. As 20m is the 75% percentile of a large classroom/ workshop and is across the room. This means that if it is able to cope up with 20m it is able to work with in all classrooms. The measured variable will be a Boolean of if it has connected to not.

At each distance of 1m approximately the wii remote will try to be connected at least 10 times. Obtaining around 200 data points.





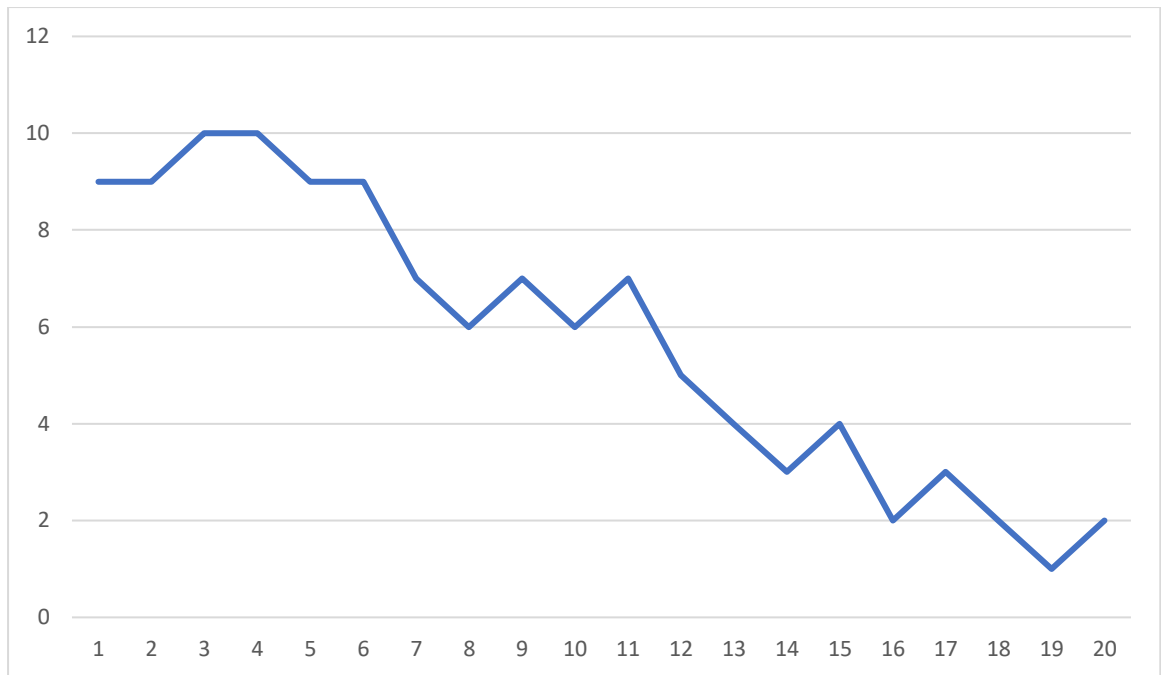
Overall from the data it can be seen to be tending towards zero and the distance increases, however the gradient is very shallow and should likely have a 50% connection chance at 23-22 meters. There is a lot of fluctuation in the results, which can be account by the multiple other variable that cannot be controlled, causing this level of randomness.

Overall this test can be counted as a success as this more then fulfils the range of 15m and can even work at a range of 20m with some degree of success. However these results may vary a lot due to different environments and settings and the type of Bluetooth receiver used.

### **Range of Detection of IR Points**

Similar to the previous experiment the distance is changes and the point at which the IR point is kept the same and the intensity of the IR point.

This also was done 10 times a different ranges in distances of 1m from 1m to 10m.



From the graph above we can see that the point is able to sense the IR point constantly from around 5m with almost 90-100% accuracy rate. This however decreases to an acceptable standard to 11 – 12 meters, where it as approximately a 75% chance of detection. Any further out from 12 m is unacceptable and cant be used for the program.

From this data we can gather that if we wanted the wii remote to be kept in the 15m range the intensity of the IR spot would have to be increase intensity by grouping multiple IR LEDs together to form a brighter spot or use one LED with a greater intensity and voltage so that it can be picked up by the sensor from a further distance.

### IR Sensor and Reliability

In this test it is to test the sensitivity of the IR sensor and how it is able to pick up IR points in different scenarios:

The scenarios are modelled like this:

- Under direct Sunlight

- With other IR sources nearby (candles, and remotes)
- A room with mirrors and other reflective materials

These scenarios are taken of what could be in a classroom and amplified to get erroneous testing data, and to test the limits of what the sensor can achieve.

### **Under Direct Sunlight**

Under these conditions the wii remote did face some difficulty when it was pointing directly at the sun, and would pick the sun as an IR point. Causing it not to be working as accurately, when used facing away from the sun it worked perfectly yielding similar results to previous tests. Overall as long as it is not used outdoors and facing towards the sun it is able to cope with this scenario.

### **With other IR Sources Nearby**

This test was also successful as the wii remote only picked up the IR LEDs emission of IR as the candles produced every weak IR sources. However when there were active remotes their LEDs showed up in the program. This makes the system unusable if there are strong IR sources near by but passive sources such as candles pose no interference.

### **A room with mirrors and reflective materials**

This was done to see if the sensor would pick up the IR point even when it was reflected off a surface. The test showed that the sensor would pick it up if it is close to the sensor, however the further away it is the less probability of it of picking up the reflected IR point, probably due to the decrease in intensity. However what must be notes is that a whiteboard surface did reflect enough IR light to cause issues sometimes.

# Homography

This section is the deals with the data handling of the two-coordinate systems and dealing with the matrixes. This main section has 2 subsections:

One has to be able to generate the translation matrix by taking in a 2 set of 4 points creating a 2 2D array. This can then be used to generate the translation matrix using the mathematical algorithm.

The other function will take in a co-ordinate and using the translation matrix generated before.

Both of these modules are dependent on the numpy library which will help with the mathematical component of the development as this has in built matrix manipulation, reducing the total development time and complexity of development.

# Translation Matrix Generation

```
4 #Edge co-ordinates respective to the IR camera (Variable)
5 #
6 #           TL           TR           BR           BL
7 iRcameraCoordinates = np.array([[50, 0], [100, 0], [100,100], [0,100]])
8
9 #maps the array to individual variables
10 x1 = iRcameraCoordinates[0,0]
11 y1 = iRcameraCoordinates[0,1]
12
13 x2 = iRcameraCoordinates[1,0]
14 y2 = iRcameraCoordinates[1,1]
15
16 x3 = iRcameraCoordinates[2,0]
17 y3 = iRcameraCoordinates[2,1]
18
19 x4 = iRcameraCoordinates[3,0]
20 y4 = iRcameraCoordinates[3,1]
21
```

This section is the module takes the inputs of the two co-ordinate systems as integers and then using the algorithm designed and mentioned in design it creates the matrix that can be used to convert different points between the two systems, using the other module in this class.

```
23 #Edge co-ordinates respective to the Projector(Constant)
24 projectorCoordinates = np.array([[0, 0], [100, 0], [100,100], [0,100]])
25
26 X1 = projectorCoordinates[0,0]
27 Y1 = projectorCoordinates[0,1]
28
29 X2 = projectorCoordinates[1,0]
30 Y2 = projectorCoordinates[1,1]
31
32 X3 = projectorCoordinates[2,0]
33 Y3 = projectorCoordinates[2,1]
34
35 X4 = projectorCoordinates[3,0]
36 Y4 = projectorCoordinates[3,1]
```

Initial set up of generic variables:

This allows for the inputted array IRcameraCoordinates to be taken out the array and into individual variables so it will be easier to work with later on. The IRcameraCoordinates will be a variable and passed on depending on the pens location.

```
► Special Variables
01 X1 = {int64} 0
01 X2 = {int64} 100
01 X3 = {int64} 100
01 X4 = {int64} 0
01 Y1 = {int64} 0
01 Y2 = {int64} 0
01 Y3 = {int64} 100
01 Y4 = {int64} 100
► iRcameraCoordinates = {ndarray} [[ 50  0]\n [100  0]\n [100 100]\n [ 0 100]] ...View as Array
► projectorCoordinates = {ndarray} [[ 0  0]\n [100  0]\n [100 100]\n [ 0 100]] ...View as Array
01 x1 = {int64} 50
01 x2 = {int64} 100
01 x3 = {int64} 100
01 x4 = {int64} 0
01 y1 = {int64} 0
01 y2 = {int64} 0
01 y3 = {int64} 100
01 y4 = {int64} 100
```

This is the grid for the display and will be fixed and nonchanging, as this will be full screen of a square shape. This can be simplified by putting it directly into the X and Y variables but kept as an array initially so it can be changed easily for testing purposes.

## Testing

As you can see above, of the two trace tables using PyCharm and Jupiter Notebooks functions to see assigned variables. We can see that this section of code works as there all variables have been assigned correctly.

Next using the variables above it was put into a matrix using numpy.array

								0
0								0
1								100
2								100
3								0
4								0
5								0
6								100
7								100

	0	1	2	3	4	5	6	7
0	50	0	1	0	0	0	0	0
1	100	0	1	0	0	0	-10000	0
2	100	100	1	0	0	0	-10000	-10000
3	0	100	1	0	0	0	0	0
4	0	0	0	50	0	1	0	0
5	0	0	0	100	0	1	0	0
6	0	0	0	100	100	1	-10000	-10000
7	0	0	0	0	100	1	0	-10000

Above we can see the two matrixes being created and generated. This is so that it can later be used for the final math.

## Testing

```

39 matrix1 = np.array([[x1, y1, 1, 0, 0, 0, - x1 * X1, - y1 * X1],
40                     [x2, y2, 1, 0, 0, 0, - x2 * X2, - y2 * X2],
41                     [x3, y3, 1, 0, 0, 0, - x3 * X3, - y3 * X3],
42                     [x4, y4, 1, 0, 0, 0, - x4 * X4, - y4 * X4],
43                     [0, 0, 0, x1, y1, 1, - x1 * Y1, - y1 * Y1],
44                     [0, 0, 0, x2, y2, 1, - x2 * Y2, - y2 * Y2],
45                     [0, 0, 0, x3, y3, 1, - x3 * Y3, - y3 * Y3],
46                     [0, 0, 0, x4, y4, 1, - x4 * Y4, - y4 * Y4],
47                     ])
48
49 matrix2 = np.array([[X1],
50                     [X2],
51                     [X3],
52                     [X4],
53                     [Y1],
54                     [Y2],
55                     [Y3],
56                     [Y4]])

```

Using SciView we can see each array has been generated effectively and

	0
0	100
1	-999900
2	-1989900
3	10100
4	0
5	0
6	-2000000
7	-1000000

correctly with the correct variables in the correct place. So the generation of both matrixes are correct.

Then using a NumPy matrix multiplication we can multiply the two matrixes

```
60 matrix1inverse = np.linalg.inv(matrix1)
61 translationParameters = np.matmul(matrix1inverse, matrix2)
62 print(translationParameters)
```

together.

```
60 translationParameters = np.matmul(matrix2, matrix2)
61 print(translationParameters)
```

Testing

Looking at the final translation matrix something has gone wrong as -999900 is a very small number which would make so sense under the context of what that number represents as stated in the design section.

After reviewing the code it turns out that the function does not inverse the matrix beforehand so another line has to be added to make sure that matrix1 is inverted and this is called matrix1inverse as per logical reasoning.



## Testing

This matrix is the correct matrix expected, hence showing that all the previous steps were concluded effectively.

	0
0	2.0
1	1.0
2	-100.0
3	0.0
4	2.0
5	0.0
6	0.0
7	0.009999999999999998

```

17 def translationParameterGenerator(p1,p2,p3,p4):
18
19     x1 = p1[0]
20     y1 = p1[1]
21     x2 = p2[0]
22     y2 = p2[1]
23     x3 = p3[0]
24     y3 = p3[1]
25     x4 = p4[0]
26     y4 = p4[1]
27
28     #Edge co-ordinates respective to the Projector(Constant)
29     projectorCoordinates = np.array([[0, 0], [1000, 0],[1000,1000],[0,1000]])
30
31     X1 = projectorCoordinates[0,0]
32     Y1 = projectorCoordinates[0,1]
33     X2 = projectorCoordinates[1,0]
34     Y2 = projectorCoordinates[1,1]
35     X3 = projectorCoordinates[2,0]
36     Y3 = projectorCoordinates[2,1]
37     X4 = projectorCoordinates[3,0]
38     Y4 = projectorCoordinates[3,1]
39
40     matrix1 = np.array([[x1, y1, 1, 0, 0, 0, - x1 * X1, - y1 * X1],
41                         [x2, y2, 1, 0, 0, 0, - x2 * X2, - y2 * X2],
42                         [x3, y3, 1, 0, 0, 0, - x3 * X3, - y3 * X3],
43                         [x4, y4, 1, 0, 0, 0, - x4 * X4, - y4 * X4],
44                         [0, 0, 0, x1, y1, 1, - x1 * Y1, - y1 * Y1],
45                         [0, 0, 0, x2, y2, 1, - x2 * Y2, - y2 * Y2],
46                         [0, 0, 0, x3, y3, 1, - x3 * Y3, - y3 * Y3],
47                         [0, 0, 0, x4, y4, 1, - x4 * Y4, - y4 * Y4],
48                         ])
49
50     matrix2 = np.array([[X1],
51                         [X2],
52                         [X3],
53                         [X4],
54                         [Y1],
55                         [Y2],
56                         [Y3],
57                         [Y4]])
58
59
60     matrix2inverse = np.linalg.inv(matrix1)
61     translationParameters = np.matmul(matrix2inverse, matrix2)
62     return translationParameters
63

```

The completed code is shown above.

# Final In-Development Translation Matrix Generation Testing

As stated previously this module will be tested by passing possible values in and seeing how it copes. No erroneous testing will be conducted such as passing strings or non-integers as all data will be sanitized before has in the data handle module.

The input will be entered, and the expected output will be calculated using an online homography calculator and will be compared to the value generated with the

Input	Expected Output	Output	Pass/ Fail
[[ 0 0] [100 0] [100 100] [ 0 100]]	[[1.] [0.] [0.] [0.] [1.] [0.] [0.] [0.]]	[[1.] [0.] [0.] [0.] [1.] [0.] [0.] [0.]]	PASS
[[ 50 50] [100 50] [100 100] [ 50 100]]	[[ 2.0000000e+00] [ 0.0000000e+00] [-1.0000000e+02] [ 0.0000000e+00] [ 2.0000000e+00] [-1.0000000e+02] [ 6.9388939e-18] [ 0.0000000e+00]]	[[ 2.0000000e+00] [ 0.0000000e+00] [-1.0000000e+02] [ 0.0000000e+00] [ 2.0000000e+00] [-1.0000000e+02] [ 6.9388939e-18] [ 0.0000000e+00]]	PASS
[[10 10] [90 10] [90 90] [10 90]]	[[ 1.25000000e+00] [ 1.75640752e-17] [-1.25000000e+01] [-1.92747053e-18] [ 1.25000000e+00] [-1.25000000e+01] [-3.46944695e-18] [ 0.00000000e+00]]	[[ 1.25000000e+00] [ 1.75640752e-17] [-1.25000000e+01] [-1.92747053e-18] [ 1.25000000e+00] [-1.25000000e+01] [-3.46944695e-18] [ 0.00000000e+00]]	PASS
[[ 30 50] [ 50 100] [ 60 30] [ 10 50]]	[[ -1.20274161e-15] [-2.80536913e+00] [ 1.40268456e+02] [ 1.47651007e+00] [-5.90604027e-01] [-1.47651007e+01] [ 6.26398210e-03] [-2.71588367e-02]]	[[ -1.20274161e-15] [-2.80536913e+00] [ 1.40268456e+02] [ 1.47651007e+00] [-5.90604027e-01] [-1.47651007e+01] [ 6.26398210e-03] [-2.71588367e-02]]	PASS
[[60 90] [10 0] [10 30] [50 20]]	[[ -1.53621068e+00] [ 2.19458669e-01] [ 7.24213606e+01] [-3.81858083e+00] [ 2.12143380e+00] [ 3.81858083e+01] [-4.29407462e-02] [ 2.19458669e-03]]	[[ -1.53621068e+00] [ 2.19458669e-01] [ 7.24213606e+01] [-3.81858083e+00] [ 2.12143380e+00] [ 3.81858083e+01] [-4.29407462e-02] [ 2.19458669e-03]]	PASS
[[ 10 10] [ 0 0] [ 50 50] [100 100]]	n/a	LinAlgErrorTraceback (most recent call la st) in <module>() ----> 1 matrixlinver se = np.linalg.inv(m atrix1) 2 translationP arameters = np.matmu l(matrixlinverse, ma trix2) 3 print(transl ationParameters)LinA lgError: Singular ma trix	FAIL

[[ 10 0] [ 0 0] [ 50 60] [100 90]]	[[-1.00000000e+01] [ 1.00000000e+01] [ 1.00000000e+02] [-6.66133815e-16] [ 3.33333333e+00] [ 0.00000000e+00] [ 2.00000000e-02] [ 1.38777878e-17]]	[[-1.00000000e+01] [ 1.00000000e+01] [ 1.00000000e+02] [-6.66133815e-16] [ 3.33333333e+00] [ 0.00000000e+00] [ 2.00000000e-02] [ 1.38777878e-17]]	PASS
[[50 50] [50 50] [50 50] [50 50]]	n/a	LinAlgErrorTraceback (most recent call last) in <module>() ----> 1 matrixlinverse = np.linalg.inv(matrix1) 2 translationParameters = np.matmul(matrixlinverse, matrix2) 3 print(translationParameters)LinAlgError: Singular matrix	FAIL

## Evaluative Comment of Test Results

From the testing table we can see that it passed most tests but unfortunately failed some. After more investigation it was realised that if all four points are passed in and 3 of them can be joint using a straight line it causes an error with the mathematical equations, hence resulting in the LingAlgError to be generated and presented.

However this has be overlooked and ignored and the code overall can be considered a success as the if this was the case the program would not run and would be borderline unusable. This is also since the likelihood of the IR camera being projected in such away over the grid will be rare and borderline impossible physically and due to human error making it even harder.

Overall this section of code can be considered a success due to the following reasons and can be viewed as more then satisfactory.

In future the code could contain some basic level of error handling and reporting to the user via the GUI to reposition the wiiremote and recalibrate.

# Co-ordinate Converter

This module will use the inputted IR point on that grid and the translation matrix generated before, as inputs and then generate another point that references the projector system, but the same physical point.

[illegible]

This code above simply generates the translation matrix using manually inputted values. Currently these values are being used because this means that both grid systems are matching exactly and there should be no change in the numbers so it is easier to

```
15 #Saves the vector translation parameters to variable constants
16 a = translationParameters[0,0]
17 b = translationParameters[1,0]
18 c = translationParameters[2,0]
19 d = translationParameters[3,0]
20 e = translationParameters[4,0]
21 f = translationParameters[5,0]
22 g = translationParameters[6,0]
23 h = translationParameters[7,0]
```

compare.

This then takes the matrix and breaks it up into individual variables so it is easier to work with. These variables have the same names as the variables in the equation stated in the design section. So it is easier to compare them and know what each one does.

```

26     #the co-ordinate that needs to be converted
27     inputCoordinate = np.array([[1,100]])
28
29     #saves the co-ordinates to seperate variables
30     x = inputCoordinate[0,0]
31     y = inputCoordinate[0,1]
32

```

Then the input co-ordinate was generated which is the co-ordinate from the IR grid that needs to be converted to the Projector grid.

Likewise this was also broken down into x and y co-ordinates, so it will be easier to program later on.

```

34     #used Homography calculation to convert co-ordinate
35     X = ((a * x) + (b * y) + c )/((g * x) + (h * y) + 1)
36     Y = ((d * x) + (e * y) + f )/((g * x) + (h * y) + 1)
37

```

Then the X and Y values are calculated using this formulae that was shown in the design section. They are calculated separately.

## Testing

```

► ■ Special Variables
  X = {int64} 1
  Y = {int64} 100
  a = {int64} 1
  b = {int64} 0
  c = {int64} 0
  d = {int64} 0
  e = {int64} 1
  f = {int64} 0
  g = {int64} 0
  h = {int64} 0
  ► inputCoordinate = {ndarray} [[ 1 100]] ...View as Array
  ► outputCoordinate = {ndarray} [ 1 100] ...View as Array
  ► translationParameters = {ndarray} [[1]\n [0]\n [0]\n [0]\n [1]\n [0]\n [0]\n [0]] ...View as Array
  x = {int64} 1
  y = {int64} 100

```

This trace table shows the all of the variables current values and as we can see the X and Y co-ordinates are the same as x and y which mean there is no logical or arithmetical issues in the code that have been identified so far.

```

63
64 def coordinateConverter(coord,translationParameters):
65     a = translationParameters[0,0]
66     b = translationParameters[1,0]
67     c = translationParameters[2,0]
68     d = translationParameters[3,0]
69     e = translationParameters[4,0]
70     f = translationParameters[5,0]
71     g = translationParameters[6,0]
72     h = translationParameters[7,0]
73
74     x = coord[0]
75     y = coord[1]
76
77     X = ((a * x) + (b * y) + c )/((g * x) + (h * y) + 1)
78     Y = ((d * x) + (e * y) + f )/((g * x) + (h * y) + 1)
79
80     outputCoordinate = np.array([X,Y])
81
82     return list(outputCoordinate)
83

```

This is the final code above.



# Final In-Development Testing for Co-ordinate Converter

Likewise with the other module in this section, all data will be sanitized before hand so all valid data will be entered for tests. However unlike the other test all values will be calculated by hand and using a calculator as no online inbuild calculator could be found that works.

Input (Translation Matrix)	Input (Co-ordinate)	Expected Output	Output	Pass/ Fail
[ 1.75640752e-17], [- 1.25000000e+01], [-1.92747053e-18], [ 1.25000000e+00], [- 1.25000000e+01], [-3.46944695e-18], [ 0.00000000e+00]]	[[50 50]]	[50. 50.]	[50. 50.]	PASS
[[0], [1], [2], [3], [4], [0], [0], [0]]	[[50 50]]	[ 52 350]	[ 52 350]	PASS
[[3], [16], [2], [5], [7], [7], [9], [13]]	[[100 100]]	[0 0]	[0 0]	PASS

[[1], [0], [0], [0], [0], [0], [0], [0]]	[[100 100]]	[100 0]	[100 0]	PASS
[[1], [10], [0], [50], [0], [100], [0], [100]]	[[500 100]]	[0 2]	[0 2]	PASS
[[0], [1], [5], [20], [4], [10], [0], [0]]	[[0 0]]	[ 5 10]	[ 5 10]	PASS
[[04], [12], [52], [2], [40], [0], [3], [50]]	[[ 2 54]]	[0 0]	[0 0]	PASS

### Evaluative Comment on Test Results

All of the outcomes were expected and exact as they had been calculated by hand. This module can be considered very successful. There was no need to test strings and all data will be sanitised beforehand to prevent errors from occurring inside the module.

# GUI and UX

This section is for the development for the front end of the application. This will look at all of the possible interaction with the computer from the user end. This will also include the GUI of the calibration page but the homography stages will not be built in yet, as these two stages were programmed parallelly.

The main stages of this section are:

Creating a window for where GUI elements can be placed of a particular size, and giving it default properties such as for it to be minimised and for the “X” button on the top right to work, and close/terminate the app. While also setting up the main loop of the program of which it will iterate over.

The next stage is to develop the real UI with the background and positioning all the elements of the text and images around it. This will constitute of the face of the application.

The next stage is optimisation of screen updates, to increase frame rate and make the program more memory efficient. This part will also include building the links between each page and combining each of the components together to create the mock up designed in the design section.

# Setting up of PyGame and the GUI

This part is where I will initialise all of the instances of the GUI so that pygame and operate effectively on the computer and can be manipulated on what we require.

```
import pygame

#Initialises pygame library
pygame.init()

#Defines the dimentions of the screen
appDisplay = pygame.display.set_mode((1000,1000))

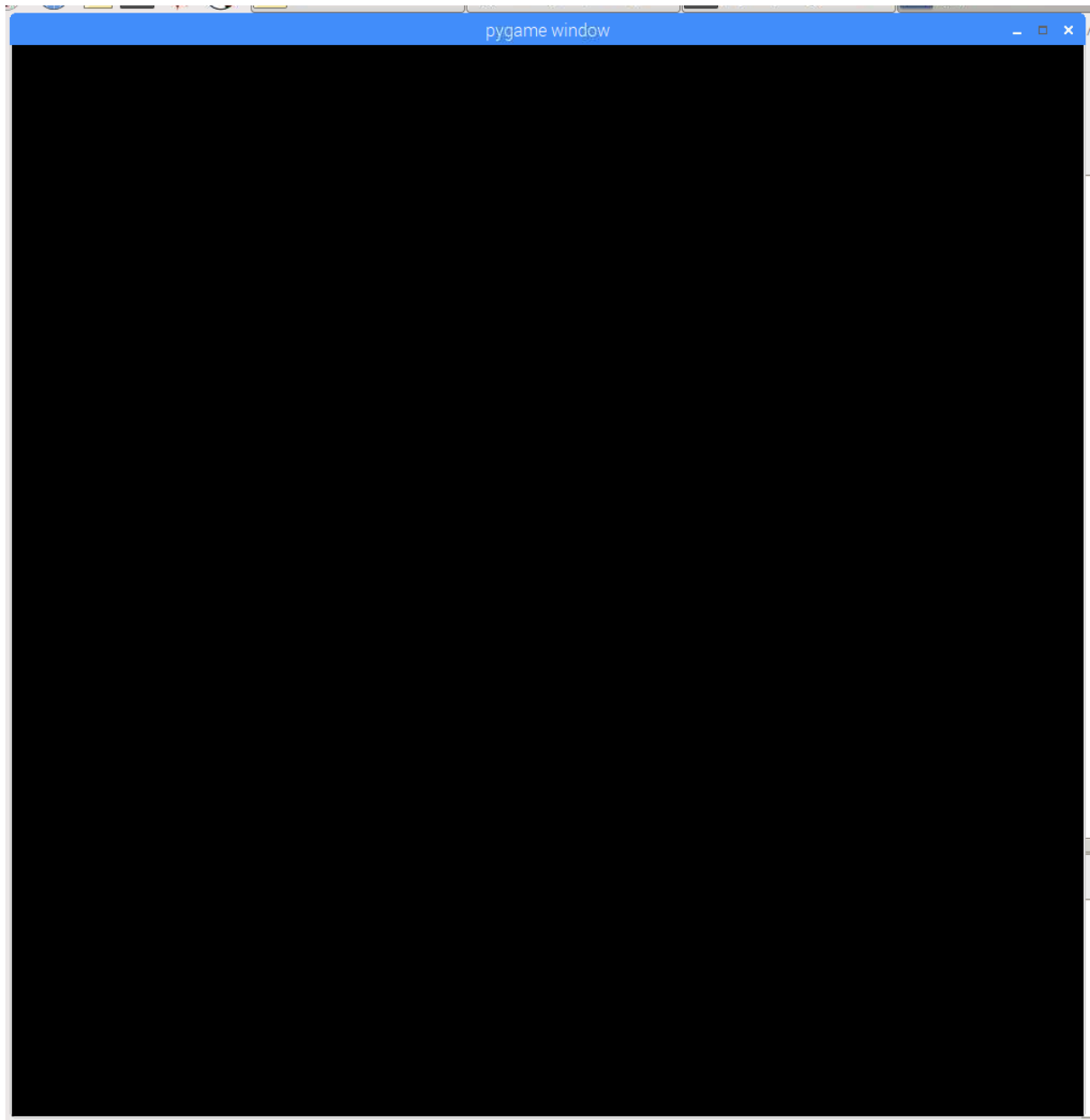
#Updates the user screen to show the application
pygame.display.update()
```

This imports the pygame library and initialises it, essentially creating all of the lower level structures that are considered that backbone of the GUI that are autogenerated.

The appDisplay is the canvas of the screen and the mode we set it to is a square of dimensions of 1000 by 1000.

Lastly the display.update() pushes the changes made to the users screen.

## Testing



As you can see this application window has been generated successfully and is of the correct dimensions

```
import pygame

#Initialises pygame library
pygame.init()

#Defines the dimentions of the screen
appDisplay = pygame.display.set_mode((1000,1000))

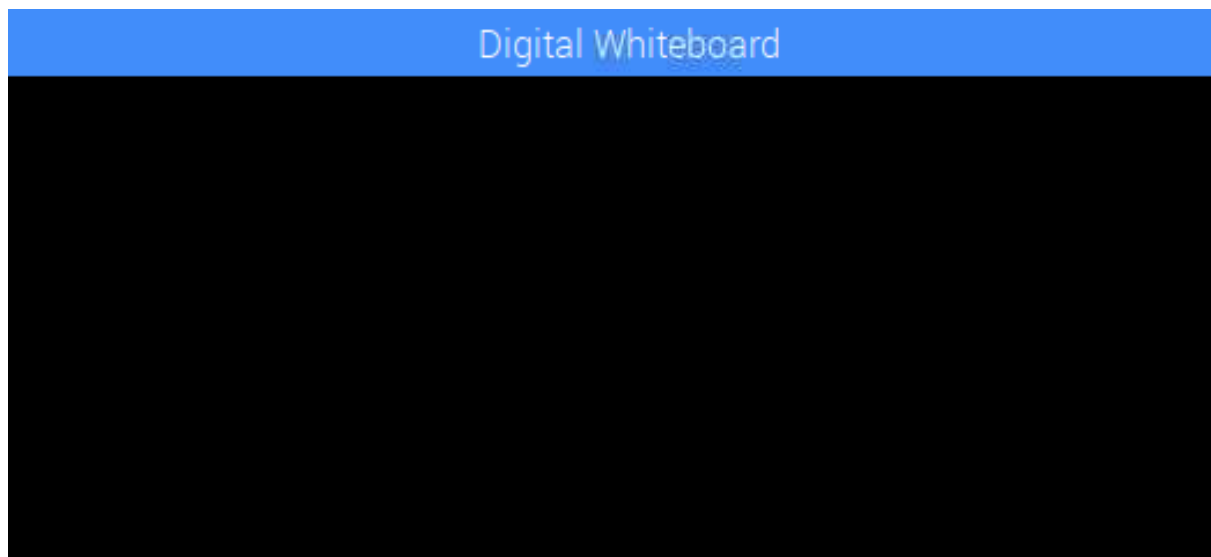
#Sets the display name
pygame.display.set_caption('Digital Whiteboard')

#Updates the user screen to show the application
pygame.display.update()
```

The code was then modified so that it would have the title of Digital Whiteboard on the top of the screen, as seen in the previous screenshot where pygame window is stated.

## Testing

As we can see from this section of the screen shot the caption name has changed and therefore this part of the code is considered a success.



```
#Condition for main loop
gameExit = False

#Clock for rate of refresh
clock = pygame.time.Clock()

#Main loop
while not gameExit:

    pygame.display.update()
    clock.tick(100)

#De-initialises pygame first before quitting best practice
pygame.quit()
quit()
```

This section was then added on that is the main loop of the program that will keep the program till something causes the program to terminate.

GameExit is the clause that needs to be changed to cause the program to safely terminate, first quitting pygame as per its documentation of best practices and then quitting the program its self.

Clock tick will make sure that is the program finishes one iteration of the loop very fast it will buffer for 100 milliseconds. This essentially creates a minimum runtime of one interaction of the code.

## Testing

Nothing changes visually from the last screen but screen stays up for an infinite period of time, otherwise previously appearing and disappearing in an instance.



```

1  import pygame
2  import time
3
4  #Initialises pygame library
5  pygame.init()
6
7  #Defines the dimentions of the screen
8  appDisplay = pygame.display.set_mode((1000,1000))
9
10 #Sets the display name
11 pygame.display.set_caption('Digital Whiteboard')
12
13 #Updates the user screen to show the application
14 pygame.display.update()
15
16 #Condition for main loop
17 gameExit = False
18
19 #Clock for rate of refresh
20 clock = pygame.time.Clock()
21
22 #Main loop
23 while not gameExit:
24
25     pygame.display.update()
26     clock.tick(100)
27
28 #De-initialises pygame first before quitting best practice
29 pygame.quit()
30 quit()

```

This is the code after the initial development and set up of Pygame and the environment.

# GUI Components and Pages

This is where all possible colours will be defined and created, any constants function that have to be created such as title and caption generation. Shapes such as circles, squares and triangles will also be programmed.

After all the constants are defined each page will be generated individually, with the respective code for each one.

```
white = (255,255,255)
black = (0,0,0)
red = (255,0,0)
green = (0,255,0)
```

This is all the main colours that are defined more can be added later in development or after the program is finished. The colour definition follows RGB convention with the first number denoting red value, second green and third blue value. Pygame does not allow for any opacity to be defined, hence not following RGBO convention.

```
def Title_to_screen(msg,color):  
    screen_text = Title_font.render(msg, True, color)  
    appDisplay.blit(screen_text, [500,500])  
  
Title_font = pygame.font.SysFont(None,50)
```

Title\_to\_screen function takes in two variables such as msg a string of what text needs to be placed on the screen, and the colour of the text. This will automatically place the text in the middle of the screen ([500,500]). The title of the font will be the systems default font and will be of size 50.

```
def Caption_to_screen(msg,color):  
    screen_text = Caption_font.render(msg, True, color)  
    appDisplay.blit(screen_text, [500,600])  
  
Caption_font = pygame.font.SysFont(None,25)
```

Caption\_to\_screen has the same effect of Title\_to\_screen but has a smaller size and will be located 100 pixels down from the centre and the Title.

```

#Given two points draws a smooth line between them
def roundline(srf, color, start, end, radius=1):
    #Calculates the gradient
    dx = end[0]-start[0]
    dy = end[1]-start[1]
    #Calculates the length of the line
    distance = max(abs(dx), abs(dy))
    #Makes smaller dots across the line to join the two points up
    for i in range(distance):
        x = int( start[0]+float(i)/distance*dx)
        y = int( start[1]+float(i)/distance*dy)
        pygame.draw.circle(srf, color, (x, y), radius)

```

Since there is a time delay between each reading of the IR point each point will have to some way of joining the two lines together to make it smoother producing a better out come as the camera refresh rate is too slow to produce smooth lines.

```

1  import pygame
2  import time
3
4  #Initialises pygame library
5  pygame.init()
6
7  #Defines the dimentions of the screen
8  appDisplay = pygame.display.set_mode((1000,1000))
9
10 #Sets the display name
11 pygame.display.set_caption('Digital Whiteboard')
12
13 #Updates the user screen to show the application
14 pygame.display.update()
15
16 #Condition for main loop
17 gameExit = False
18 #Clock for rate of refresh
19 clock = pygame.time.Clock()
20
21 white = (255,255,255)
22 black = (0,0,0)
23 red = (255,0,0)
24 green = (0,255,0)
25
26
27 #Given two points draws a smooth line between them
28 def roundline(srf, color, start, end, radius=1):
29     #Calculates the gradient
30     dx = end[0]-start[0]
31     dy = end[1]-start[1]
32     #Calculates the length of the line
33     distance = max(abs(dx), abs(dy))
34     #Makes smaller dots across the line to join the two points up
35     for i in range(distance):
36         x = int( start[0]+float(i)/distance*dx)
37         y = int( start[1]+float(i)/distance*dy)
38         pygame.draw.circle(srf, color, (x, y), radius)
39
40
41 def Title_to_screen(msg,color):
42     screen_text = Title_font.render(msg, True, color)
43     appDisplay.blit(screen_text, [500,500])
44
45 Title_font = pygame.font.SysFont(None,50)
46
47 def Caption_to_screen(msg,color):
48     screen_text = Caption_font.render(msg, True, color)
49     appDisplay.blit(screen_text, [500,600])
50
51 Caption_font = pygame.font.SysFont(None,25)
52
53 #Main loop
54 while not gameExit:
55
56     Title_to_screen("SAMPLE TITLE TEXT", black)
57     Caption_to_screen("sample caption text", red)
58
59     pygame.display.update()
60     clock.tick(100)
61
62 #De-initialises pygame first before quitting best practice
63 pygame.quit()
64 quit()
65

```

This is the completed code for this section. The highlighted coded is temporary and only for testing purposes.

## Final In-Development GUI Components Testing

What will be Done	Expected Output	Output	Pass/ Fail	Comment
Trying to find the maximum length of string Title_to_screen can take	The string will extend out of the page and disappear showing the central part of the text	The longest string it can take is approximately 40 characters depending on the size of the characters as the font is not monospaced	PASS	This is more then enough to fit titles along the screen.
Trying to find the maximum length of string Caption_to_Screen can take	The string will extend out of the page and disappear showing the central part of the text	The longest space it can take is approximately 100 characters and can vary greatly due to the font	PASS	This is more then enough to fit caption which are 1 sentence along the screen

Drawing a line with two inputs and checking if it fills up the two positions.	A line will be drawn of the same width between the two points.	Draws the line in-between the two given points	PASS	Successfully draws the line between the two of constant radius
Two Titles will be pushed to the screen at the same time	The old Title will disappear and be replaced with the new Title that has been pushed	Overlays both of the texts over each other	FAIL	It must be noted that the screen must be cleared before changing titles
Two Captions will be pushed to the screen at the same time	The old Caption will disappear and be replaced with the new Caption that has been pushed.	Overlays both of the captions over each other	FAIL	Same as above (pushing two tiles)
Two points will be passed of the same point	No visible change will occur on the screen	No visible change happens	PASS	
Two points will be passed that are out of	The line is not drawn	It still draws the line but	FAIL	The data for joining the

bounds for the screen		the ends of the line cannot be seen		lines together must be sanitized before hand
--------------------------	--	---	--	--

## Evaluative Comment on Test Results

From the results we can see above we can see that the lengths of the strings pushed was adequate as the maximum length for a title is approximately 15 character which is substantially under the 40 limit and leaves room for a margin and padding. This is similar to the caption as at the maximum there will only be 1 sentence being pushed, which is approximately 50 characters which is also below the 100 character limit. This data uses heavy approximations as the font used is not monospaced hence that depending on the letters and characters used this can cause the text to appear larger or smaller.

For the testing of pushing both data at the same time this can cause the text to overlap on top of the previous text that has been pushed. This was not expected and hence required the screen to be cleared, after some reseach this could be done by using `display.flip()` or `display.update()` to make the screen change to the most UpToDate state. This failure Is not one of major importance due to its ability to easily be fixed, using the addition of a singular line of code. This however must be noted of great importance in the later sections.

In terms of the generation of the smooth line between the two points, this is done successfully and would to do anything to significantly wrong, however one thing that must be pointed out is that it will join two points even if they are off the screen. This will result in the line spanning the entire screen. This will cause some basic background validation to check if the point is in the screen before it joins it up, which may have to be



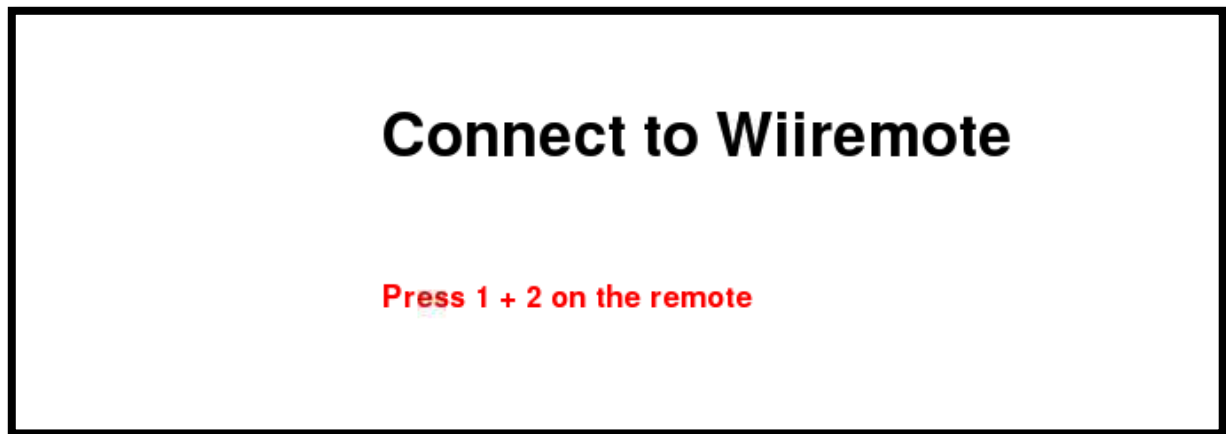
added but should not effect the overall performance of the code as the pen should never be turned on outside this region.

## Page Layout

Since the individual components are tested and developed taking those building blocks and combining them to produce the end result of what was wanted in the design section, that was tested there.

This will be making up the larger building blocks of each page that can be linked up using the main algorithm page.

```
#Initialises connection with the wiiremote  
appDisplay.fill(white)  
#Puts the title to the screen and the caption and then refreshes it  
Title_to_screen("Connect to Wiiremote",black)  
Caption_to_screen("Press 1 + 2 on the remote",red)
```



### Connecting to Wiiremote page

This page has a title of “Connect to Wii remote” in black with the caption “Press 1 + 2 on the remote” also in black. This is the first page the user will see.

Testing

This successfully pushes the correct information to the screen and looks like the mock-up generated in the design section so is successful.

### Connection Failure Page

This page is used to tell the user that the connection the Wii remote was not successful. This page only consists of a tile in red reading “Connection Not Successful”.

This page will only be visited for 1 second before the application quits if no Wii remote is detected.

```
appDisplay.fill(white)
#Puts the title to the screen and the caption and then refreshes it
Title_to_screen("Connection Not Successful",red)
pygame.display.update()
```

Testing



**Connection Not Sucessful**

This page is relatively simple and requires minimal code and appears successfully on screen.

## Connection Successful Page

This page is used to tell the user that the wii remote has been successfully liked to the computer and the main application is being loaded in. This will consist of a title in green reading "Success".

```
appDisplay.fill(white)
#Puts the title to the screen and the caption and then refreshes it
Title_to_screen("Success",green)
pygame.display.update()
```



## Testing

As seen above the text is successfully displayed in the correct colour and position.

## Calibration Page

This page works in collaboration with the translation matrix and is used to direct the user to put the pen in all four corners of the screen, that can then be picked up by the wii remote to be used to generate the calibration co-ordinates.

This will involve of coding a red and green square in all four corners of the screen and then instructing the user to touch the green square. Once that has been done that square will then turn red and then the next square will turn green. After all squared have been touched this page will be complete.

```
#Main loop
while not gameExit:

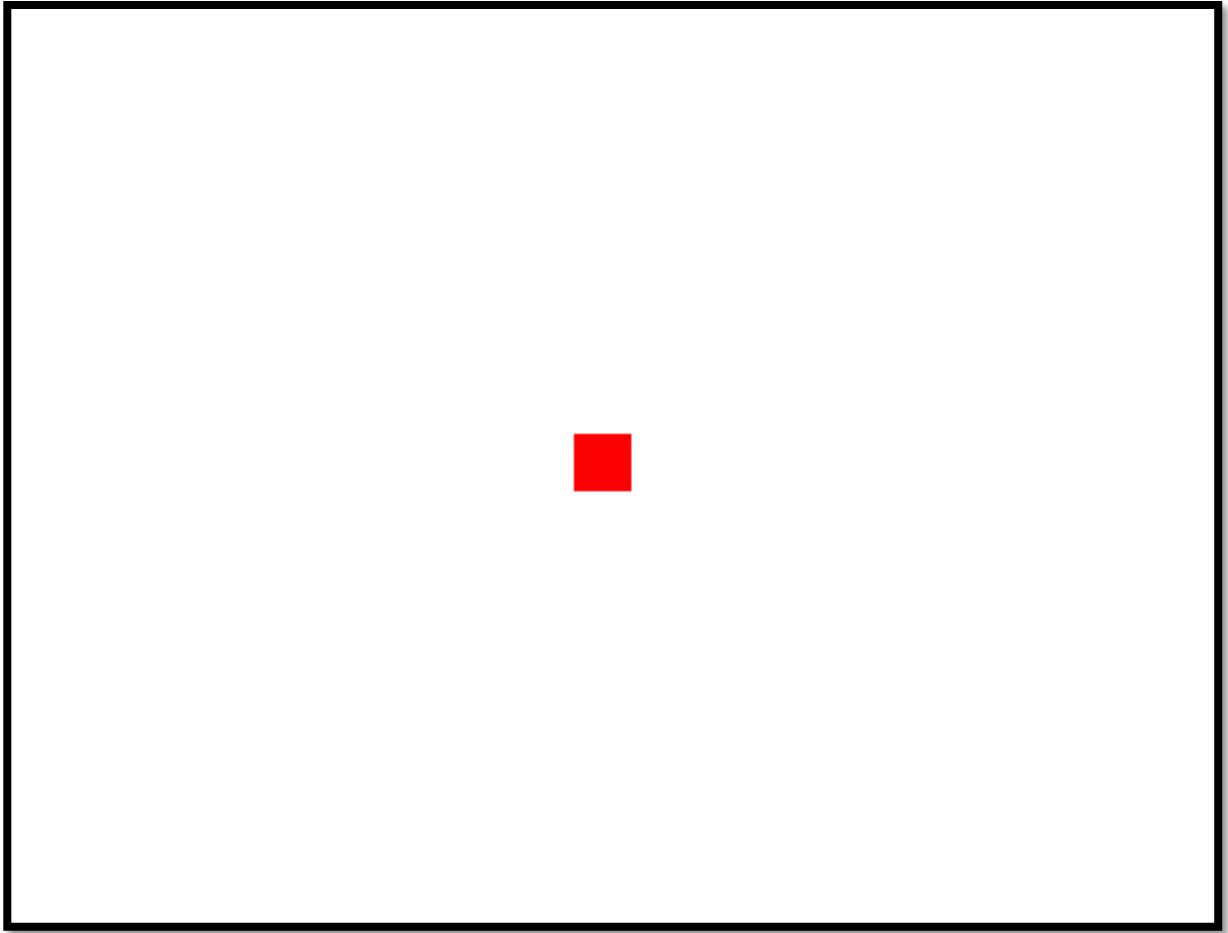
    appDisplay.fill(white)
    #Draws a red box in the center
    pygame.draw.rect(appDisplay,red, [500,500,25,25])

    pygame.display.update()
    clock.tick(100)
```

This code was added to the main loop of the code developed earlier under the GUI section when developing this section of code and all code following will be placed here.

This draws a red box in the centre of the screen and keeps it there.

## Testing



In the screen a red box has been placed in the centre, so this code for drawing the box of 25 by 25 px and then placing it in the centre at 500 and 500 has worked successfully.

The next step was to make 4 replicas of this box and place it all around the edges 50px around the edges, effectively giving the display a padding of 50px.

This there for meant that boxes had to be located at:

- (50,50)
- (50,1000-50)
- (1000-50,1000-50)

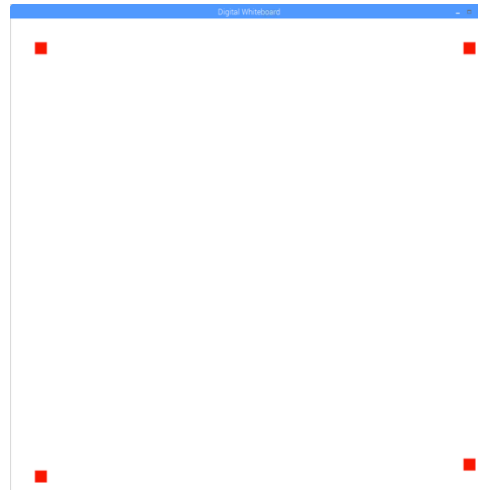
```
#Main loop
while not gameExit:

    appDisplay.fill(white)
    #Generation of 4 target markers
    pygame.draw.rect(appDisplay,red, [50,50,25,25])#top left
    pygame.draw.rect(appDisplay,red, [50,1000-50,25,25])#bottom left
    pygame.draw.rect(appDisplay,red, [1000-50,1000-75,25,25])#bottom right
    pygame.draw.rect(appDisplay,red, [1000-50,50,25,25])#top right

    pygame.display.update()
    clock.tick(100)
```

- (1000-50, 50)

Testing



This test failed as the squares are not lined up. This is possibly due to the squares being drawn from the bottom left corner



The co-ordinated were then readjusted to:

- (50,50)
- (50,1000-75)

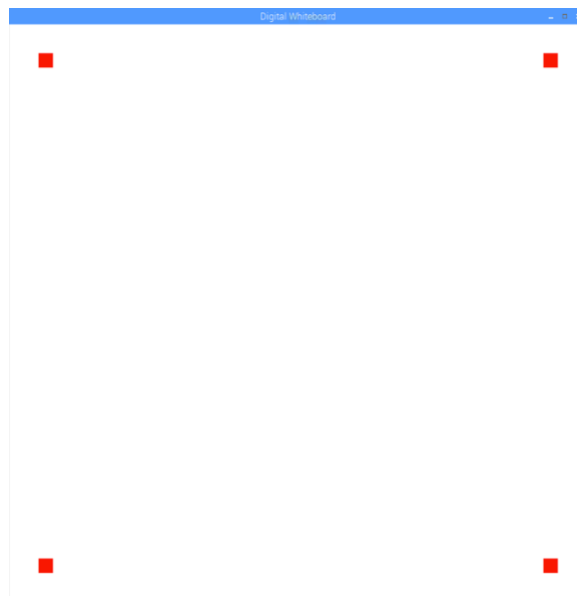
```
#Main loop
while not gameExit:

    appDisplay.fill(white)
    #Generation of 4 target markers
    pygame.draw.rect(appDisplay,red, [50,50,25,25])#top left
    pygame.draw.rect(appDisplay,red, [50,1000-75,25,25])#bottom left
    pygame.draw.rect(appDisplay,red, [1000-75,1000-75,25,25])#bottom right
    pygame.draw.rect(appDisplay,red, [1000-75,50,25,25])#top right

    pygame.display.update()
    clock.tick(100)
```

- (1000-75,1000-75)
- (1000-75,50)

Testing



As we can see that there are squares that have now been aligned correctly, to the edges.

```
#Main Loop
while not gameExit:

    appDisplay.fill(white)

    #Generation of 4 target markers
    TL = pygame.draw.rect(appDisplay,red, [50,50,25,25])#top left
    BL = pygame.draw.rect(appDisplay,red, [50,1000-75,25,25])#bottom left
    BR = pygame.draw.rect(appDisplay,red, [1000-75,1000-75,25,25])#bottom right
    TR = pygame.draw.rect(appDisplay,red, [1000-75,50,25,25])#top right

    listOfBoxes = [TL,BL,BR,TR]

    for box in listOfBoxes:

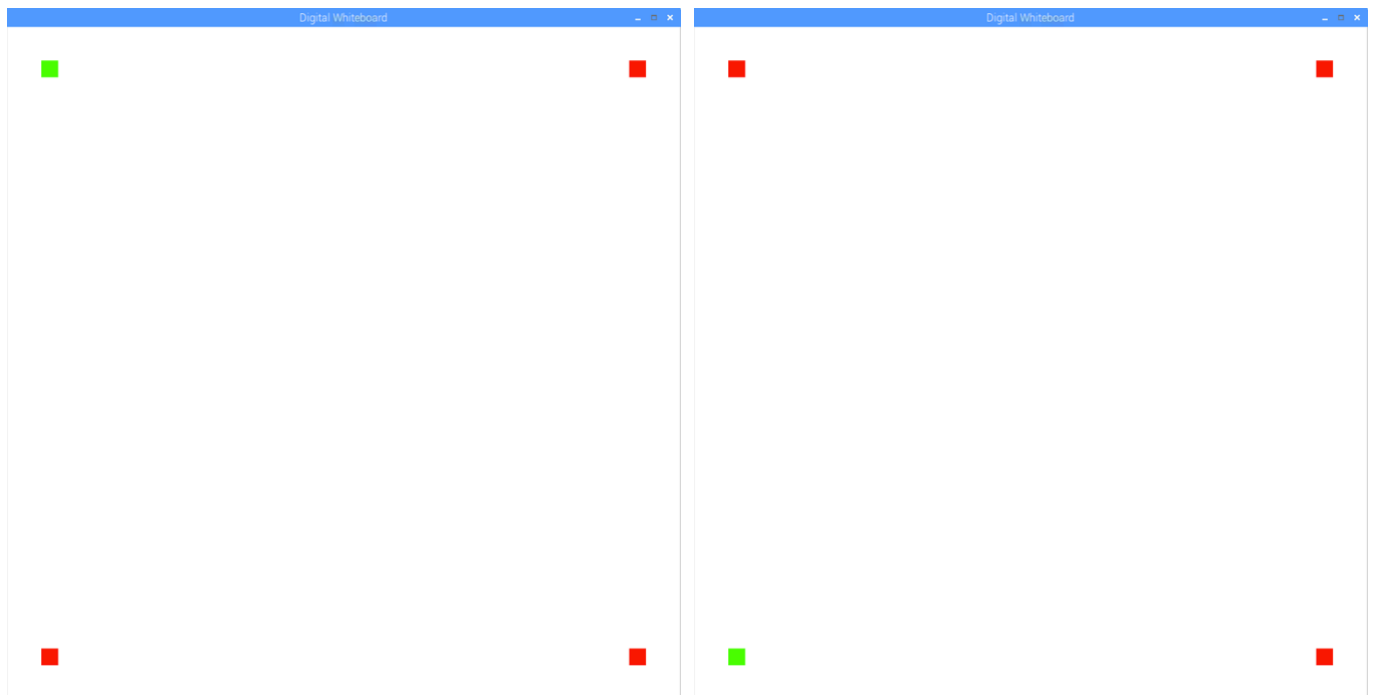
        #Makes box green
        box.colour(green)

        #Updates screen and then makes the program wait
        appDisplay.update()
        time.sleep(5)

        #Makes box green
        box.color(red)
```

Next each square was turned green to make sure the user knew which one to click. The order in which one would go green would be in a anti-clockwise order. In the following order top left, bottom left, bottom right, top right.

As the wiiremote IR detection has not been added in yet temporary code will be added with the time.sleep() to simulate the user interaction.



## Testing

From the code running above we can see two of the screen shots that makes the square green then makes it red and makes the next one green in a anti-clockwise order.

This section of the code and be viewed successful as it works successfully.

In the centre of the calibration screen the title “Calibration Mode” has to be displayed in black, with the caption of “Tap the green square with the pen” in black as well.

```
#Main Loop
while not gameExit:

    appDisplay.fill(white)

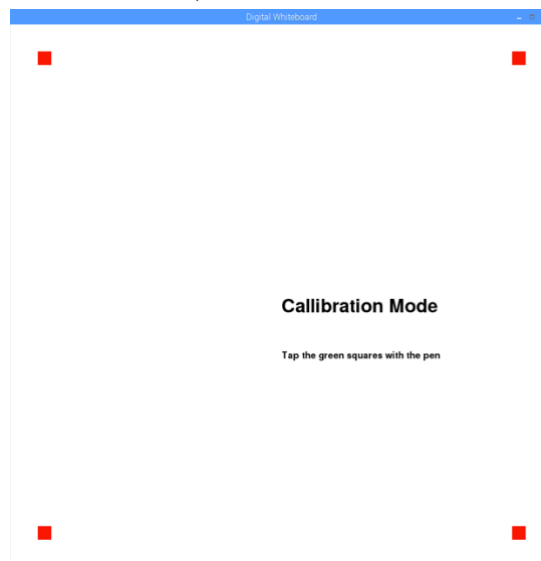
    #Generation of 4 target markers
    TL = pygame.draw.rect(appDisplay,red, [50,50,25,25])#top left
    BL = pygame.draw.rect(appDisplay,red, [50,1000-75,25,25])#bottom left
    BR = pygame.draw.rect(appDisplay,red, [1000-75,1000-75,25,25])#bottom right
    TR = pygame.draw.rect(appDisplay,red, [1000-75,50,25,25])#top right

    Title_to_screen("Calibration Mode",black)
    Caption_to_screen("Tap the green squares with the pen",black)
    pygame.display.update()
,
```

## Testing

As you can see the tile and caption have been successfully generated, with the correct colour and text.

## Email Validation



In this section of code this is used to check if the email is valid or not on the client side using regression and other methods discussed in the design section of validation. This will do basic checks for:

- Check for an "@" symbol in the string
- Check that there is at least 1 "." in the string
- Make sure that there are no other symbols in the string such as "@£\$%^&\*()"

- Check that there is characters before the “@” symbol
- Check that there is a “.” after the “@” symbol
- Check that there are characters after the “@” symbol
- Check that there are characters before the “.” symbol
- Check that there are characters after the “.” symbol

This is needed in the code to make sure the email is correct before emailing the file off the users or students in the class.

A simple pre-built email regression could be used, but this was not used as it would allow greater control when self-developing it and can have features such as only email students that are in school added on as a functionality later on in other prototypes.

## Obtaining Characters Individually in the Email

First the initial test email will be a valid email called [JohneeyAppleseed@email.com](mailto:JohneeyAppleseed@email.com).

```
1  
2 email = "JohneeyAppleseed@email.com"  
3
```

Then the next stage is to use a for loop to iterate through the string.

```
1
2 email = "JohneeyAppleseed@email.com"
3
4 #Character flags
5 at = False
6 dot = False
7
8 #Loops throught the email string
9 for character in email:
10     print(character)
11
```

## Testing

```
J  
o  
h  
n  
e  
e  
y  
A  
p  
p  
l  
e  
s  
e  
e  
d  
@  
e  
m  
a  
i  
l  
.  
c  
o  
m
```

As you can see each character was printed on a different line hence showing that it iterated through the string



The next step was to have checks to make sure that there was a “@” sign, and a “.” in the string given. This will be achieved using conditional if statements to set a flag to true if found.

```
1  email = "JohneeyAppleseed@email.com"
2
3
4  #Character flags
5  at = False
6  dot = False
7
8  #Loops throught the email string
9  for character in email:
10     #Checks if @ exisits
11     if character == "@":
12         at = True
13     #Checks if . exists
14     if character == ".":
15         dot = True
16
17  print(at, dot)
18
```

Testing

```
True True
```

The expected output was true as there is a “@” and “.” in the string email.

The code is then changed to get the position of the “@” and the “.” Using the function enumerate.

```
1  email = "JohneeyAppleseed@email.com"
2
3
4  # Character flags
5  at = False
6  dot = False
7
8  # Loops throught the email string
9  for position, character in enumerate(email):
10     # Checks if @ exisits
11     if character == "@":
12         at = position
13     # Checks if . exists
14     if character == ".":
15         dot = position
16
17  print(at, dot)
```

Testing

```
16 22
```

This was the expected result as this is the position at which the respective symbols exist.

Then the next check was added which was to make sure that between the positions of the start of the string and “@” sign there is at least one character.

The first step is to iterate and get the amount of characters of the first part of the email.

```
17 for character in range(0,at):  
18     print(character)
```

Testing

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15
```

The expected output was all of the numbers till 15 because that is the length of “JohneeyAppleseed” which was the outcome so it was successful.

```

18
19 # Loops though all ints of the first part of the email
20 for characterPosition in range(0,at):
21     # Checks if is a alpha-character
22     if ((email[characterPosition] >= 'a' and email[characterPosition] <= 'z')
23         or (email[characterPosition] >= 'A' and email[characterPosition] <= 'Z')):
24         a = a + 1

```

Then next stage was the check how many alpha characters exist before the “@” symbol.

The variable a is to count how many alpha-characters exist before the “@” symbol.

Testing

16

The expected result was 16 so this piece of code can be considered a success.

The next part was to loop over the last part of the string and do the same. To make sure there are letters in between the “@” and the “.”

```

26
27 # Loops through all ints in between @ and .
28 for characterPosition in range(at,dot):
29     # Checks if is a alpha-character
30     if ((email[characterPosition] >= 'a' and email[characterPosition] <= 'z')
31         or (email[characterPosition] >= 'A' and email[characterPosition] <= 'Z')):
32         b = b + 1
33
34 print(b)
35

```

Testing

5

The expected result was 5 therefore this code is correct and is working appropriately.

This was then repeated for the last part of the string to make sure there are characters after “.”.

Then finally the final code was completed to make sure that there was a, b and c where all greater than 0 then the email was valid.

```
42     if a > 0 and b > 0 and c > 0:
43         print("Valid Email")
44     else:
45         print("Invalid Email")
46
```

## Testing

```
Valid Email
```

The input email was valid so therefore the output should be valid which is the outcome so therefore this last section of the code is correct.

```

1
2 def emailValidation(email):
3     # Character flags
4     a = 0
5     b = 0
6     c = 0
7
8     # Loops through the email string
9     for position, character in enumerate(email):
10        # Checks if @ exists
11        if character == "@":
12            at = position
13        # Checks if . exists
14        if character == ".":
15            dot = position
16
17        # Loops through all ints of the first part of the email
18        for characterPosition in range(0,at):
19            # Checks if is a alpha-character
20            if ((email[characterPosition] >= 'a' and email[characterPosition] <= 'z')
21                or (email[characterPosition] >= 'A' and email[characterPosition] <= 'Z')):
22                a = a + 1
23
24        # Loops through all ints in between @ and .
25        for characterPosition in range(at,dot):
26            # Checks if is a alpha-character
27            if ((email[characterPosition] >= 'a' and email[characterPosition] <= 'z')
28                or (email[characterPosition] >= 'A' and email[characterPosition] <= 'Z')):
29                b = b + 1
30
31        # Loops through all ints in between . and last part
32        for characterPosition in range(dot,len(email)):
33            # Checks if is a alpha-character
34            if ((email[characterPosition] >= 'a' and email[characterPosition] <= 'z')
35                or (email[characterPosition] >= 'A' and email[characterPosition] <= 'Z')):
36                c = c + 1
37
38        if a > 0 and b > 0 and c > 0:
39            return True
40        else:
41            return False
42

```

This is the final code for the email validation.

# Final In-Development Email Validation Testing

Using the testing table in the design section there will be various different strings passed into the function to see how it reacts and if any of it does manage to break the system then is there a pattern that can be found and be fixed.

Input (Email)	Expected Output	Output	Pass/Fail
<a href="mailto:Sdas4@icloud.com">Sdas4@icloud.com</a>	True	True	PASS
Sdas4@icloud.	False	False	PASS
<a href="mailto:Sdas4@icloud....da">Sdas4@icloud....da</a>	False	True	FAIL
@£\$%^&*(	False	Traceback (most recent call last): File "/Users/sarthakdas/Desktop/Development/EmailValidation.py", line 44, in <module>print(emailValidation(email))File "/Users/sarthakdas/Desktop/Development/EmailValidation.py", line 25, in emailValidation for characterPosition in range(at,dot) UnboundLocalError: local variable 'dot' referenced before assignment	FAIL
Asdasd.asda@asd.com	True	True	PASS
.das..@@.asd	False	False	PASS
Sdas4@@dasd.com	False	True	FAIL

## Evaluative Comment on Test Results

This code was not very effective at checking if the emails were valid or not, but did a good enough job for most parts, of simple errors. One major error was found of that if the user did not enter a . or a @ it caused a traceback error causing the whole program to crash so a basic level of previous sanitisation is required to make sure that these characters are valid and in the string.

# Final Code Development

Each of the individual modules have been developed previously and independently, now it is time to develop the code that is able to unite these modules together as discussed and shown in the design section.

This is the section where various prototypes will be developed leading up the final one that introduces all of the modules. This will check that at least two of the modules work together and can be tested by the user after each prototype so the user can check each development iteration and cycle.

## Prototype 1

This will be getting the GUI of connection linked with the Wii remote connection module, so they can interact with each other and tell the user if it has failed or passes. This will not include the homography and the calibration GUI.

## Prototype 2

This is where the point plotting is added after the connection page so the user can draw lines however the lines will not line up with the point as there is no calibration.

## Prototype 3

This is where the smooth line algorithm and queue section will be implemented to provide smoother lines.

## Prototype 4

This is when the calibration section will be added to the code.



## Prototype 1

This is simply linking the GUI sections of the code with the connection modules

```
43  
44  #Initialises connction with the wiiremote  
45  appDisplay.fill(white)  
46  #Pushes title to screen for connection  
47  Title_to_screen("Connect to Wiiremote",black)  
48  Caption_to_screen("Press 1 + 2 on the remote",red)  
49  pygame.display.update()  
50  time.sleep(1)  
51  #Trys to connect to wiiremote  
52  wii = wiiremote.init()  
53
```

developed under the Wii control class.

First the connection and the “Press 1 + 2” page was joint up so the user knew what buttons to press.

This section of code of was placed outside of the main loop as this should only be run once.

### Testing

The code ran as expected displayed the test correctly and allowed the Wii remote to connect 1 seconds after the test was displayed.

The next section of code was to link up the successful and failed connection pages to the code. This will be done with a try function and a finally statement.

```
53
54 #Checks if there was a successful connection
55 try:
56     #Runs if it failed
57     if not wii:
58         appDisplay.fill(white)
59         Title_to_screen("Connection Not Successful",red)
60         pygame.display.update()
61         time.sleep(2)
62         pygame.quit()
63         quit()
64 #This will always run | when the connection was successful
65 finally:
66     appDisplay.fill(white)
67     Title_to_screen("Success",green)
68     pygame.display.update()
69     time.sleep(1)
70
```

If the connection has failed then the program will display the failed message and then terminate the program after 1 second.

If the connection was successful then the program will display the success message and pause at this prototype.

## Testing

The code successfully runs and only activates each section of code when it is meant to. This section of code works successfully.

## User Testing

After this prototype was complete this was passed onto possible users and clients and they were asked to test it. This was to see if they could find any flaws in the code or any usability issues that have not been considered during the in-development testing and the design section.

The users were simply asked to follow the instructions and were observed to see what they struggled with and asked at the end of the testing period what improvements could be made to the program at its current stage.

When observing the users all users found it easy to interact with the board and understand what it was asking. Some users took too long to press the 1 and 2 buttons on the Wii remote, as either they were not familiar with the product and had to look for the buttons or they placed the Wii remote away from themselves unable for them to press the buttons in time. However, these problems were only experienced during the first tests with the user, possibly because the users were not expecting that, but on the second run they were able to predict what was going to be asked of them and hence press the buttons in time.

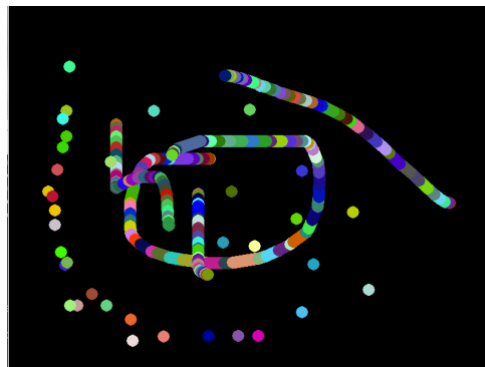
Overall after the observation almost all the users said that the software so far was easy to user and it conveyed the instructions simply and clearly. And that this prototype was complete and is able to be developed further.

```
66
67 #Main loop
68 while not gameExit:
69
70     #Gets the IR data at that time
71     pos = wiiremote.event(wii,pygame)
72     #Checks if there is a visible IR point
73     if pos != None:
74         #Plots a circle at that point
75         pygame.draw.circle(appDisplay, color, pos, 10)
76         #updates the screen
77         pygame.display.flip()
78
```

## Prototype

This prototype builds and will create the first the main loop.

This is when the IR gathered from the Wii remote and be directly plotted on the projectors grid without any translation or conversion.



2

on the previous one useable feature in

points will be

Testing

As you can see above lines were drawn successfully on the screen using the IR pen. Even though the line drawn was not aligned with the IR pen the general movement tracked was successful.

## User Testing

The first stage of the application which was the connection of the Wii remote was previously tested so no major problems with that section should be shown here, since both parts are independent of each other.

This user testing will focus on the users interaction with the pen and the movement and how quickly they write and how quickly it is plotted to the screen. The main parts that will be analysed here is the responsiveness of the line drawing and the reliability of the Wii remote

From the feedback gained from the user the software was responsive enough, there was a noticeable lag compared to other commercial products but most users found it justifiable considering the cost of each respective product.

The did get noticeably worse the longer the program was used however this will be overlooked for now as this may be resolved automatically in the future prototypes.

Overall the users said it was fine, not as great as they hoped but would be good enough for it to progress to the next prototype, most where unhappy with the lag, or did not fully understand the concept of no calibration at this stage.

```
69     #Homography calibration for projector-sensor
70     while not calibrated:
71         appDisplay.fill(white)
72         #Generation of 4 target markers
73         pygame.draw.rect(appDisplay,red, [50,50,25,25])#top left
74         pygame.draw.rect(appDisplay,red, [50,1000-75,25,25])#bottom left
75         pygame.draw.rect(appDisplay,red, [1000-75,1000-75,25,25])#bottom right
76         pygame.draw.rect(appDisplay,red, [1000-75,50,25,25])#top right
77         Title_to_screen("Callibration Mode",black)
78         Caption_to_screen("Tap the green squares with the pen",black)
79         pygame.display.update()
80
81         #Marker one callibration (Top-Left)
82         pygame.draw.rect(appDisplay,green, [50,50,25,25])
83         pygame.display.update()
84         #Checks for next IR spot
85         IR = False
86         while not IR:
87             #gets the IR point
88             IRdata= wiiremote.event(wii,pygame)
89             #Checks if it actually picked up a point
90             if IRdata != None:
91                 #saves it as a variable
92                 IR_coord1 = list(IRdata)
93                 IR = True
94
95         #Sleep to prevent long presses from being registered
96         time.sleep(2)
97
98         #Makes the square red
99         pygame.draw.rect(appDisplay,red, [50,50,25,25])
100        pygame.display.update()
101
102        #Marker two callibration (Bottom-Left)
103        pygame.draw.rect(appDisplay,green, [50,1000-75,25,25])
104        pygame.display.update()
105        #Checks for next IR spot
106        IR = False
107        while not IR:
108            IRdata = wiiremote.event(wii,pygame)
109            if IRdata != None:
110                IR_coord2 = list(IRdata)
111                IR = True
112
```

## Prototype 3

This prototype is built on top of the previous one, but has the calibration added to it with the homography and co-ordinate converter included. This prototype will be able to show off the main functionality of the product and should be able to draw the lines exactly as they are intended by the user.

```


153
154     print "======"
155     print IR_coord1
156     print IR_coord2
157     print IR_coord3
158     print IR_coord4
159     print "======"
160     translationParameters = datahandle.translationParameterGenerator(IR_coord1,IR_coord2,IR_coord3,IR_coord4)
161     print translationParameters
162     calibrated = True
163

```

The code above has the IR checking built in, using the while not IR to make sure an actual IR co-ordinate was received and not none, if it didn't find any point. Time.sleep(1) was used to prevent long presses on a singular point being registered for the two squares. This gives it a buffer for the user to move between the two squares and press on the next square. The code above shows for squares 1 and 2 but is the same for the rest of the squares and follows the same code structure.

After all the boxes have been pressed the IR points that have been saved are then printed out in the terminal and then the translationParameter function is called and passes in all those values. Once the matrix has been generated it is then printed to the screen so the user can see. This is temporary and only for testing purposes this section of code. In the real code nothing will be printed to the terminal as all communication will be conducted via the GUI to create a better UX.

## Testing



```
Mesg pipe overflow
```

Approximately 3 seconds into running the program, but varies in the range of 2-5, the program stops being responsive and is unable to be terminated from the Wii remote. This is after it prints to the screen "Mesg pipe overflow".

This is suspected to be caused by the pipeline build up and a stack overflow as it is not able to cope up with the constant flow of data quick enough. This caused the code to hang and the program to become unresponsive.



Therefor to eliminate this problem the buffer will have to be manually emptied after each time.sleep function being called.

This uses the library garbage collector to delete unused memory that is building up the stack.

```
94
95     #Sleep to prevent long presses from being registered
96     time.sleep(2)
97     #Emties the memory using garbage collector
98     gc.collect()
99
```

## Testing

No difference was made and the code still failed and became unresponsive. This solution was inadequate and made no effect on the running of the code.

Many other examples were tried that were suggested on stack over flow and other such websites and the cwiid library documentation was searched for possible ways of emptying the buffer.

## Testing

None of the various examples worked or created any major effect that allowed the program to run as intended.

Since this problem could not be solved this section of the code has to be re-designed and the logic of the code had to be re-thought. Essentially what was needed was some way to detect when the pen what pressed over one square and then over another.

But also, a way to eliminate a long press from being registered as pressing both the squares.

Initially a `time.sleep` was used creating a buffer to stop a long press from being registered but caused memory issues so all following solutions have to omit the `time.sleep` function entirely.

One solution that was thought of was that when moving from square 1 to square 2 only the Y-coordinate must change by more than 100. This eliminates long presses as they are held over the same vicinity of area, and when moving the pen between squares the pen should be turned off after a while so it will then be registered when it is turned on at the location of the next square.

Hence the next solution that will be tried will be to check if the X or Y co-ordinate is changed when comparing the two points.

When moving from:

- Square 1 to Square 2 results in a  $\Delta Y$
- Square 2 to Square 3 results in a  $\Delta X$
- Square 3 to Square 4 results in a  $\Delta Y$

Due to the change in design other modules have to be developed in order to support this.

One module would have to be called point compare which checks if there is a significant change in the X or Y co-ordinate axis.

Developing point change.

For  $\Delta Y$  co-ordinates:

```
2
3 def pointCompareY(set1, set2):
4     #Takes the two points that have to be compared and checks the Y co-ordinates
5
6     #Breaks the points up into individual variables
7     x1 = set1[0]
8     y1 = set1[1]
9     x2 = set2[0]
10    y2 = set2[1]
11
12    #Checks that the Y co-ordinates are at least 100 different
13    if ( y1 > y2 and (y1 - y2)>100 )or ( y2 > y1) and (y2 - y1)>100:
14        return True
15    else:
16        return False
```

For  $\Delta X$  co-coordinates:

```
18 def pointCompareX(set1, set2):
19     #Takes the two points that have to be compared and checks the X co-ordinates
20
21     #Breaks the points up into individual variables
22     x1 = set1[0]
23     y1 = set1[1]
24     x2 = set2[0]
25     y2 = set2[1]
26
27     #Checks that the X co-ordinates are at least 100 different
28     if ( x1 > x2 and (x1 - x2)>100 )or ( x2 > x1) and (x2 - x1)>100:
29         return True
30     else:
31         return False
32
```

## Testing

Random numbers were placed into the tables and the outputs were correct as expected returning True or False depending if the numbers vary enough.

For the main section of the code the time.sleep functions where removed and the point compare functions where inserted into the code.

```
113     #Checks for next IR spot
114     IR = False
115     while not IR:
116         #Gets the IR data
117         IRdata = wiiremote.event(wii,pygame)
118         #Checks that there was a visible IR point
119         if IRdata != None:
120             #Saves the co-ordinate
121             IR_coord2 = list(IRdata)
122             #Checks that the co-ordinate if diffrent
123             if datahandle.pointCompareY(IR_coord2, IR_coord1):
124                 IR = True
125
```

This code is repeated but changing which pointCompare function it calls if it was the X or Y one.



```

1  import wiiremote
2  import pygame
3  import GUIcomponents as GUI
4  import datahandle
5  import time
6  import sys
7  import random
8
9  #Initialises pygame library
10 pygame.init()
11 appDisplay = pygame.display.set_mode((1000,1000))
12 pygame.display.set_caption('Digital Whiteboard')
13 pygame.display.update()
14
15 #Constant declerations
16 def roundline(srf, color, start, end, radius=10):
17     dx = end[0]-start[0]
18     dy = end[1]-start[1]
19     distance = max(abs(dx), abs(dy))
20     for i in range(distance):
21         x = int( start[0]+float(i)/distance*dx)
22         y = int( start[1]+float(i)/distance*dy)
23         pygame.draw.circle(srf, color, (x, y), radius)
24
25 #Title to the screen
26 def Title_to_screen(msg,color):
27     screen_text = Title_font.render(msg, True, color)
28     appDisplay.blit(screen_text, [500,500])
29
30 #Caption to the screen
31 def Caption_to_screen(msg,color):
32     screen_text = Caption_font.render(msg, True, color)
33     appDisplay.blit(screen_text, [500,600])
34
35 gameExit = False
36 calibrated = False
37 clock = pygame.time.Clock()
38 Title_font = pygame.font.SysFont(None,50)
39 Caption_font = pygame.font.SysFont(None,25)
40
41 posPrev = None
42 pointLast = False
43
44 white = (255,255,255)
45 black = (0,0,0)
46 red = (255,0,0)
47 green = (0,255,0)
48

```

```

48
49 #Initialises connction with the wiiremote
50 appDisplay.fill(white)
51 #Puts the title to the screen and the cpation and then refreshes it
52 Title_to_screen("Connect to Wiiremote",black)
53 Caption_to_screen("Press 1 + 2 on the remote",red)
54
55 pygame.display.update()
56 time.sleep(1)
57 wii = wiiremote.init()
58 try:
59     if not wii:
60
61         appDisplay.fill(white)
62         #Puts the title to the screen and the cpation and then refreshes it
63         Title_to_screen("Connection Not Sucessful",red)
64         pygame.display.update()
65
66         time.sleep(2)
67         pygame.quit()
68         quit()
69 finally:
70
71     appDisplay.fill(white)
72     #Puts the title to the screen and the cpation and then refreshes it
73     Title_to_screen("Success",green)
74     pygame.display.update()
75
76
77     time.sleep(1)
78
79
80
81 #Main loop
82 while not gameExit:
83     #Homography calibration for projector-sensor
84     while not calibrated:
85         appDisplay.fill(white)
86         #Generation of 4 target markers
87         pygame.draw.rect(appDisplay,red, [50,50,25,25])#top left
88         pygame.draw.rect(appDisplay,red, [50,1000-75,25,25])#bottom left
89         pygame.draw.rect(appDisplay,red, [1000-75,1000-75,25,25])#bottom right
90         pygame.draw.rect(appDisplay,red, [1000-75,50,25,25])#top right
91         Title_to_screen("Callibration Mode",black)
92         Caption_to_screen("Tap the green squares with the pen",black)
93         pygame.display.update()
94

```

```

94
95     #Marker one callibration (Top-Left)
96     pygame.draw.rect(appDisplay,green, [50,50,25,25])
97     pygame.display.update()
98     #Checks for next IR spot
99     IR = False
100    while not IR:
101        #Gets the IR data
102        IRdata= wiiremote.event(wii,pygame)
103        #Checks that there was a visible IR point
104        if IRdata != None:
105            #Save the co-ordinate
106            IR_coord1 = list(IRdata)
107            IR = True
108
109    pygame.draw.rect(appDisplay,red, [50,50,25,25])
110    pygame.display.update()
111
112    #Marker two callibration (Bottom-Left)
113    pygame.draw.rect(appDisplay,green, [50,1000-75,25,25])
114    pygame.display.update()
115    #Checks for next IR spot
116    IR = False
117    while not IR:
118        #Gets the IR data
119        IRdata = wiiremote.event(wii,pygame)
120        #Checks that there was a visible IR point
121        if IRdata != None:
122            #Saves the co-ordinate
123            IR_coord2 = list(IRdata)
124            #Checks that the co-ordinate if diffrent
125            if datahandle.pointCompareY(IR_coord2, IR_coord1):
126                IR = True
127
128    pygame.draw.rect(appDisplay,red, [50,1000-75,25,25])
129    pygame.display.update()
130
131    #Marker three callibration (Bottom-Right)
132    pygame.draw.rect(appDisplay,green, [1000-75,1000-75,25,25])
133    pygame.display.update()
134    #Checks for next IR spot
135    IR = False
136    while not IR:
137        IRdata = wiiremote.event(wii,pygame)
138        if IRdata != None:
139            IR_coord3 = list(IRdata)
140            if datahandle.pointCompareX(IR_coord3, IR_coord2):
141                IR = True
142
143    pygame.draw.rect(appDisplay,red, [1000-75,1000-75,25,25])
144    pygame.display.update()
145

```



```

149
150     #Checks for next IR spot
151     IR = False
152     while not IR:
153         IRdata = wiiremote.event(wii,pygame)
154         if IRdata != None:
155             IR_coord4 = list(IRdata)
156             if datahandle.pointCompareY(IR_coord4, IR_coord3):
157                 IR = True
158
159         pygame.draw.rect(appDisplay,red, [1000-75,50,25,25])
160         pygame.display.update()
161         pygame.display.update()
162
163         appDisplay.fill(white)
164         Title_to_screen("Calibration Success",green)
165         pygame.display.update()
166
167         appDisplay.fill(white)
168         pygame.display.update()
169
170         #Prints the edge co-ordinates
171         print("=====")
172         print IR_coord1
173         print IR_coord2
174         print IR_coord3
175         print IR_coord4
176         print "====="
177         #gets the translationParameters
178         translationParameters = datahandle.translationParameterGenerator(IR_coord1,IR_coord2,IR_coord3,IR_coord4)
179         print translationParameters
180         calibrated = True
181
182     #gets the postion of the IR spot
183     pos = wiiremote.event(wii,pygame)
184
185     #Checks if there was a visible IR spot
186     if pos != None:
187         #Converts the point
188         pos = datahandle.coordinateConverter(pos,translationParameters)
189         color = black
190         print pos
191         #Plots the point
192         pygame.draw.circle(appDisplay, color, pos, 10)
193         pygame.display.update()
194         pointLast = True
195
196
197     #Gets all events e.g keyboard,mouse
198     for event in pygame.event.get():
199         #Checks if the 'X' button has been pressed then exits
200         if event.type == pygame.QUIT:
201             gameExit = True
202         #print(wiiremote.event(wii,pygame))

```

```

196
197     #Gets all events e.g keyboard,mouse
198     for event in pygame.event.get():
199         #Checks if the 'X' button has been pressed then exits
200         if event.type == pygame.QUIT:
201             gameExit = True
202         #print(wiiremote.event(wii,pygame))
203
204     #Updates the app display
205     pygame.display.update()
206     clock.tick(2000)
207
208     Title_to_screen("Quit",red)
209     pygame.display.update()
210     time.sleep(2)
211
212
213     pygame.quit()
214     quit()
215

```

As you can see above is the main structure of the program that ties in all the elements that were previously developed and tested separately now work together. This is the code for the final prototype.

## Testing

Nothing visual has changed but the lines now match where the pen is being placed. And the homography works reliably without crashing or causing any errors.

## User Testing

This prototype shows off the main feature of the product that will be used by the users and will let them experience this for the first time. This example should draw lines exactly where the users place their pens if they are able to calibrate the software properly and draw is almost real time to their hand movements.

When given the Wii remote and the pen users were able successfully calibrate the system however the precision of some users meant they pressed the top of the squares for all the calibration or the bottom, causing the end result of the calibration to be slightly off, due to them not knowing where the centre was. This can be improved in future models by including a centreline in the squares. But even with this issue that was noticed most users were able to subconsciously edit their movements to link lines together.

The bigger issue that was faced however was the lag during the using of the software. The longer the software was kept running the worse the lag got eventually leading to 3 seconds of lag and growing. This caused the software to become very unresponsive and unusable within 30 seconds of using the program.

However, with in those 30 seconds of the product, all users where happy with the accuracy of the product and how well the software was able to calibrate itself. Not only this but they also commented on how easy the software was to set up and use.

# Further Development of Prototype

As seen above in the in-development testing the prototype faces issues in terms of generating a line in pseudo-Realtime, and started to develop major lag issues hindering its usability and function of the product. To find a solution to this and to find what was causing it a set of strategies was used to try to debug the issue.

## Looking at Example Code and Documentation

Using the internet mainly and websites, such as stack overflow and GitHub were searched for example code that use the library of cwiid. To see other people's who's code works and possibly reverse engineer how it works to develop a module on a way of solving the issue I was faces.

Documentation was also searched; however, this was extremely hard to find as the library cwiid was very outdated, last refreshed and pushed in 2012. Also, since this is based on proprietary hardware and protocol the library can be seen as a hack-around and possibly hence forth doesn't contain any formal documentation.

Overall from trying to use this method to solve the solution did not make much progress in improving the code and solving the issue that was causing the problem.

# Attempting to Empty Memory Buffer

Looking at other modules that can be brought into the software to force empty the buffer, in hopes that this means that the computer has less data to deal with resulting in a better run speed and improving usability.

Libraries such as sys and os were looked at and numerous solution and example codes were copied and tested and also altering current code so it can fit in with the change in design. But since the data was coming in through a library it was protected and kept private and there for making it unable for the external developed code to reach it and dump the memory registers.

This method was also no meaning full effect on the code and improving usability.

## Changing Main Loop Run-time

In pygame there is a line of code that determines the minimum runtime the main loop can have. This value is states in milliseconds and can be changed to force the computer to take longer to run that section of code.

The idea behind this was that the longer the main loop took to run would reduce the sample rate at which it probes the wii remote for data. This therefore should make it gather less data and hence make it easier for the computer to deal with the data.

When this was done with various values, what was observed was that while responsiveness increased the lines became increasingly spotty and broken. This however still did not solve the problem entirely but made slight improvements.

# Testing

This is the post-development testing that will be checked against the specification designed in the design section this will also be followed by the results table.

The main testing that will be carried out here is alpha and beta testing. For alpha, I the developer, will try to break the system using various methods, and see how it compares to the specification this will be more of a white box testing as the logic that the programs will be known allowing for this testing to occur and create more in-depth testing. Beta testing will involve taking a focus group or a group of potential clients and giving them the software and hardware and seeing if they are able to break the system or cause any unexpected scenarios to occur. This will be more of a form of white box testing as they will not be aware of the exact inner working of the product but know the general gist of what it should be capable of doing using its inputs and outputs.

The main use of this final testing is to check functionality, usability, robustness. Functionality will involve checking if the product is checked to what extent it is able to solve the problem stated in the research section. This means it will be seen if it is able to assist the teachers in their teaching while also being light weight easy to set up and economically viable. Usability will involve obtaining a wide range of people and seeing how they cope with the product such as people with disabilities of various forms. This will also be taken in the form of running the code on various other machines to make sure it still functions. Robustness, in part is a combination of the previous two, but focuses on how reliable it is under different scenarios, and how it copes in each of these.

# Functionality Testing

This testing looks at the software and product holistically and will be a combination of alpha and beta testing ie testing of the developer and the user. The end data will be laid out in a table to make easy reference when evaluating the product against its success criteria.

This be used to measure against a scale on how successful the product is in solving the problem stated before in the problem section.

The following tests will be conducted under this section:

- Testing each component of the problem against the solution
- Gathering data on the ranges and limitations of the final developed model

These tests will be able to develop some tangible data allowing it to be compared to other existing solutions and see how it is able to operate and what would be its commercial viability if and when launched into the market at its current state or with future improvements in certain areas.

Sine this project has a hardware aspect to it, this will also be able to provide more insightful data on what the capabilities of the hardware is in a non-controlled environment and what's those limitations are such as range and angle.

# Testing Against Problem Described

This test is used to test on how effectively it solves the problem that it was initially set out to solve. In short this could be described as a cheap and portable system that is able to create a digital smart board which allows the teacher to interact with software with a pen passed device to write or draw diagrams. The system must also be under the maximum cost of 300 pounds, with the lower the cost the better.

The main success criteria of this section is:

- Cost with development costs
- Portability
- Ability to track the pen
- Ease of set up

Each of these subcategories will be checked and evaluated in separate ways. In each way it will be done in the way that is most effective and will produce the most accurate and representable data of the program so it can be evaluated effectively later on.



Over all the program was tested to see if it was able to achieve what was promised, when the project was first described. The testing of this can be seen in the attached video.

First the Wii remote was checked if it was able to be connected to raspberry pi effectively and this was repeated several time similarly to the test when developing the module for connectivity, results mainly stayed the same as there for rate of success connections and range with other factors remaining constant.

In terms of its ability to recognise a IR point on the screen this also remained fairly constant and accurate with previous data.

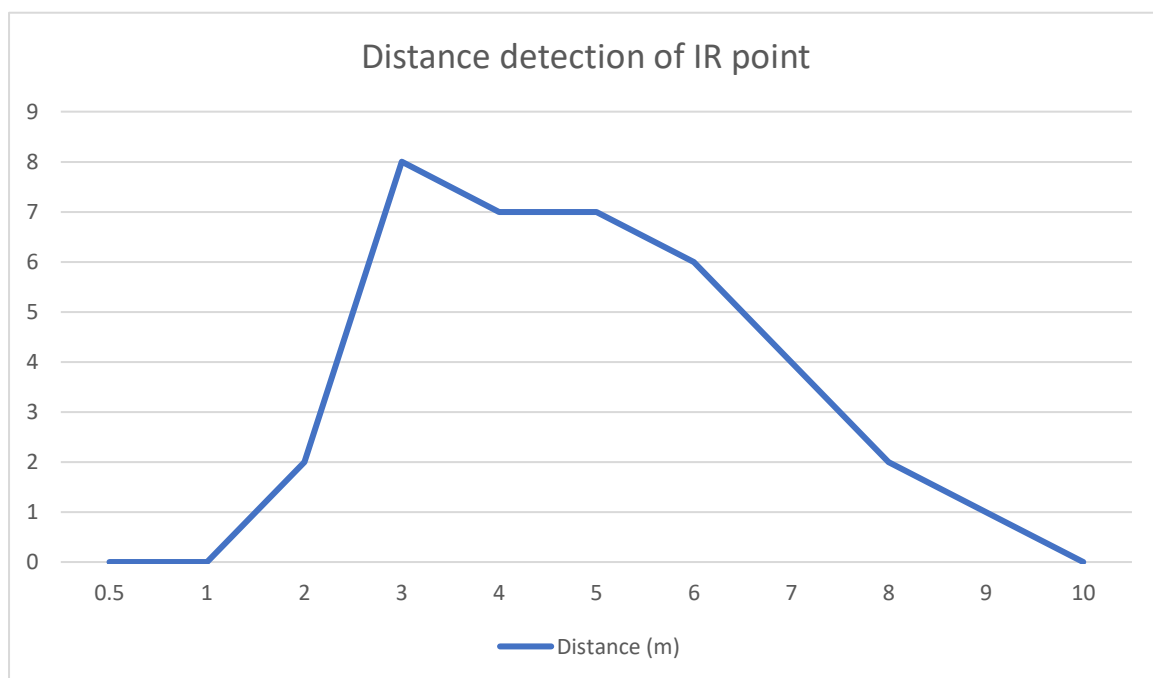
One thing to be noted that was a major issue was that as the program is used the lag between the movement of the pen due to the build up of the buffer got more noticeable the longer the program was running. This issue is most likely the biggest flaw of the program as this compromises the basic core functionality of it, making it unable to do the main task it is supposed to, however this problem could be combination of hardware and software issues.

This also failed other features such as shareability and ability to change colours and pen-sizes due to the fact that it became so disconnected that these features simply could not be implemented at this stage in the prototype currently. Causing it to fail in these respect immediately simply for not having the modules developed.

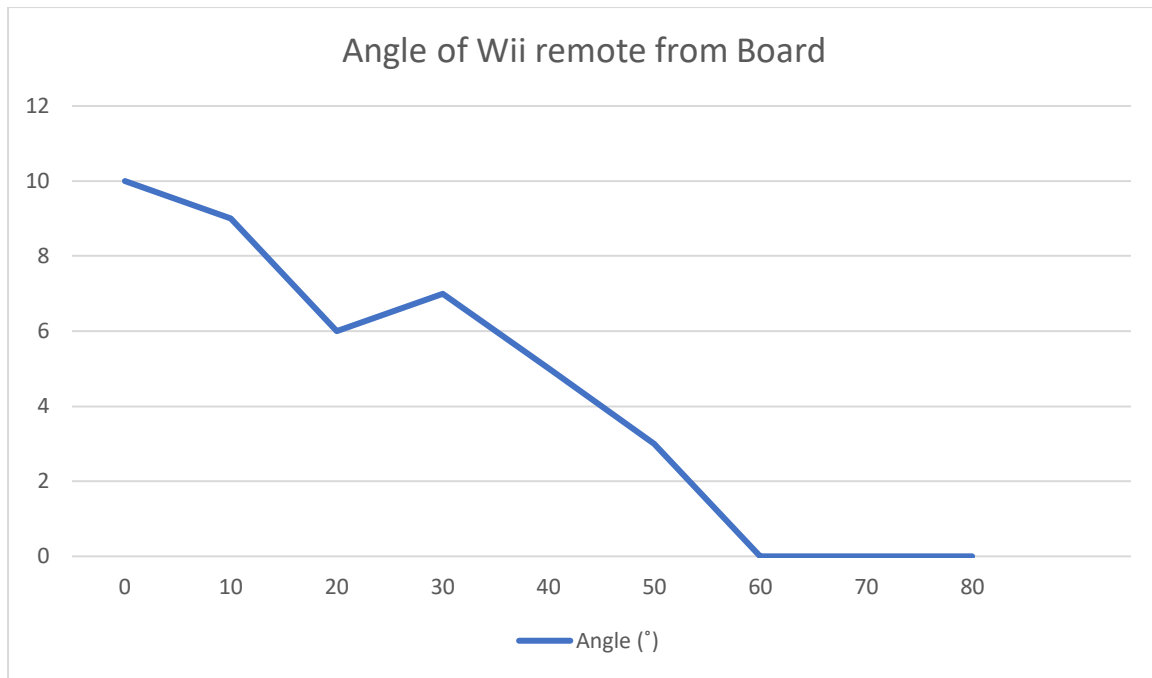
# Range and Physical Limitations

These following tests are used to find optimum and maximum set up for the software and hardware as each can cause limitations themselves from the Intensity of IR to the exact calibration algorithm used.

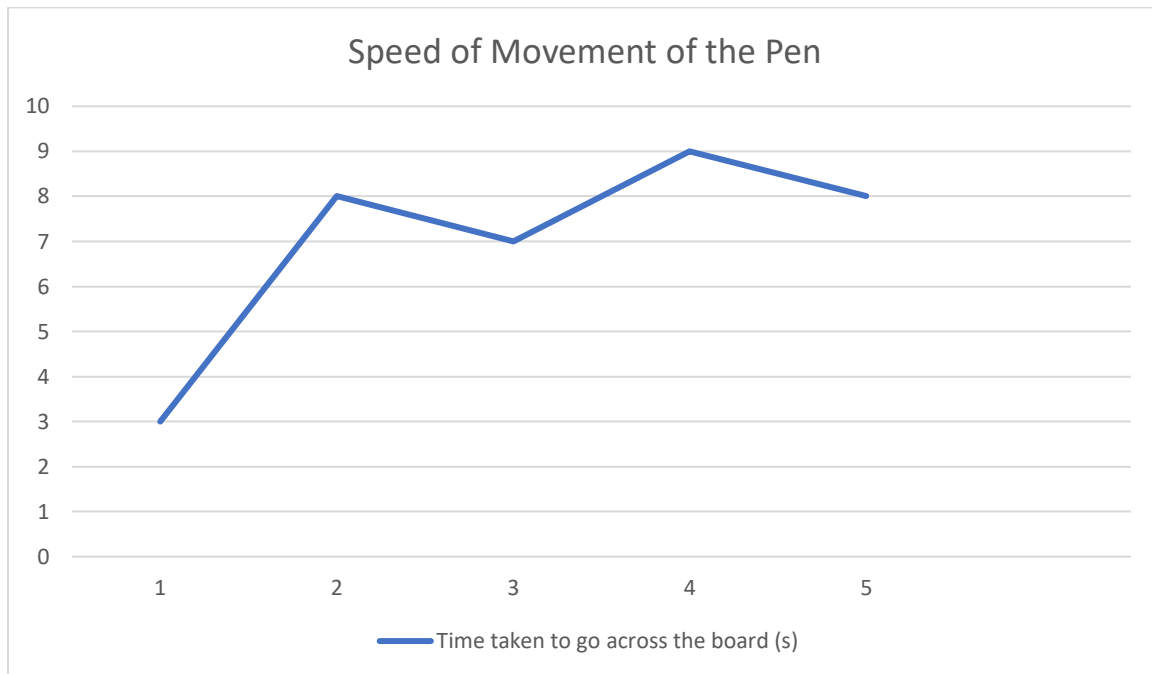
The different tests will be used to assess the range that the Wii remote is effective of tracking and picking up the IR point reliably. The maximum angle at which the Wii remote can be kept at, from the board. The angle at which the IR can be held at. The accuracy of tracking over a fast-moving pen. These will allow for a holistic view of quantitatively data that can be used to evaluate its successfulness at meeting the success criteria.



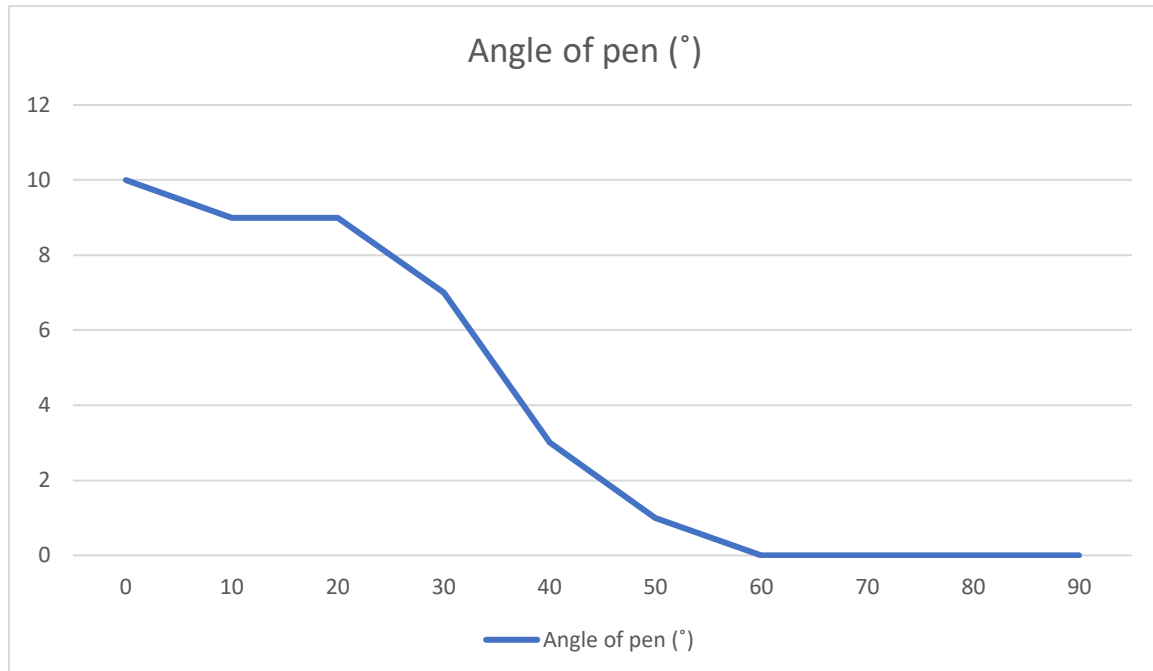
As the data shows above the system was useable at 0.5 and 1m away from the board but was effective when this distance grew. It was most effective at 3 – 4 meters and then slowly tapered down from there to 0 at 10 and the effective range is 3 – 6 meters.



This graph shows that it the pen works accurately enough till approximately 30° degrees from the centre. This also shows that the program will not work at any satisfactory above 50° and will not work at all above 60°.



This is measured by a lag rating that is observed. The slower the movement of the pen, the less noticeable the pen lag is to the user. Average pen moving speed is around 1.5-2 m/s which shows the rating is around 5-7, under normal usage.

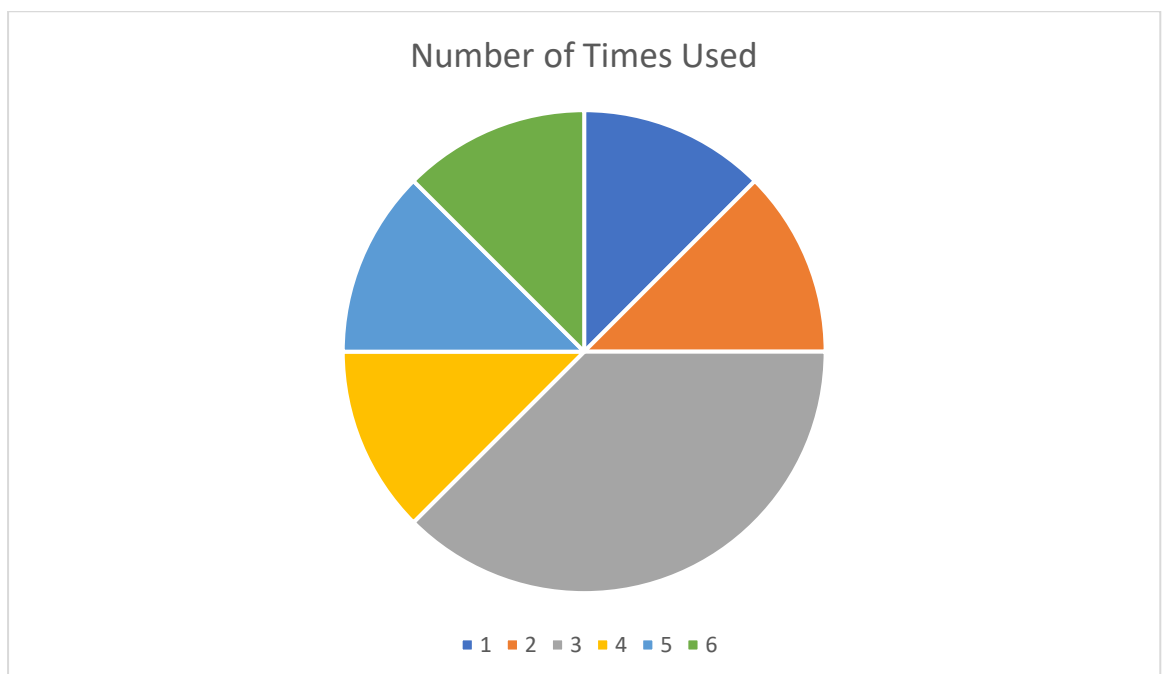


This shows that it was very effective till approximately 30° but ideally should be kept around 20° to the sensors centre for maximum effectiveness.

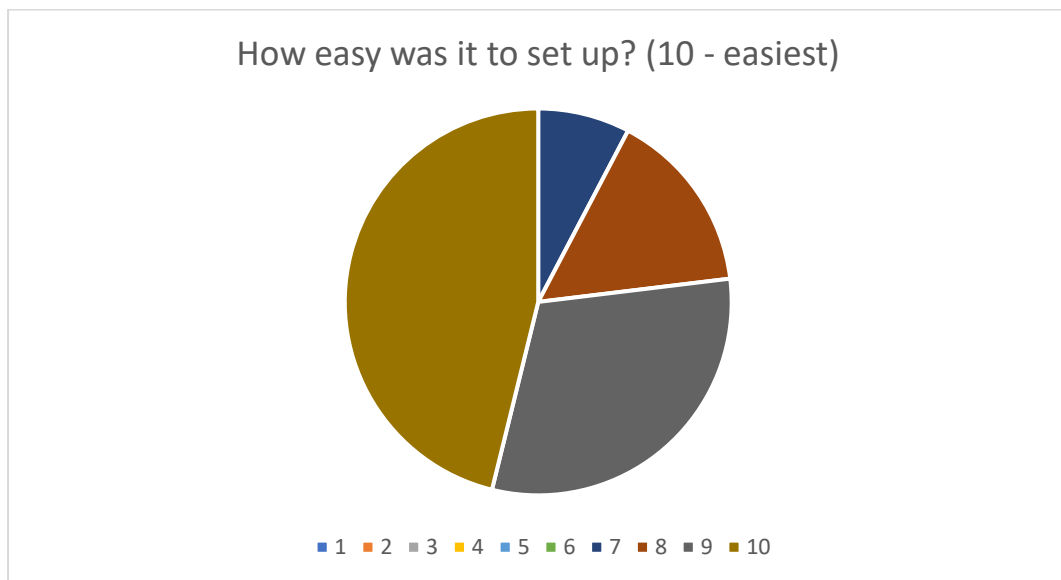
## Usability Testing

Following this the same test was repeated but with potential clients and users of the product, where they were each given a raspberry pi and the software loaded on it and a Wii remote and the IR pen that was created. They were then given a trail period of which they were able to use the system over a period of two days whenever they wanted and data was gathered on how they felt about the product and what their opinions where of it.

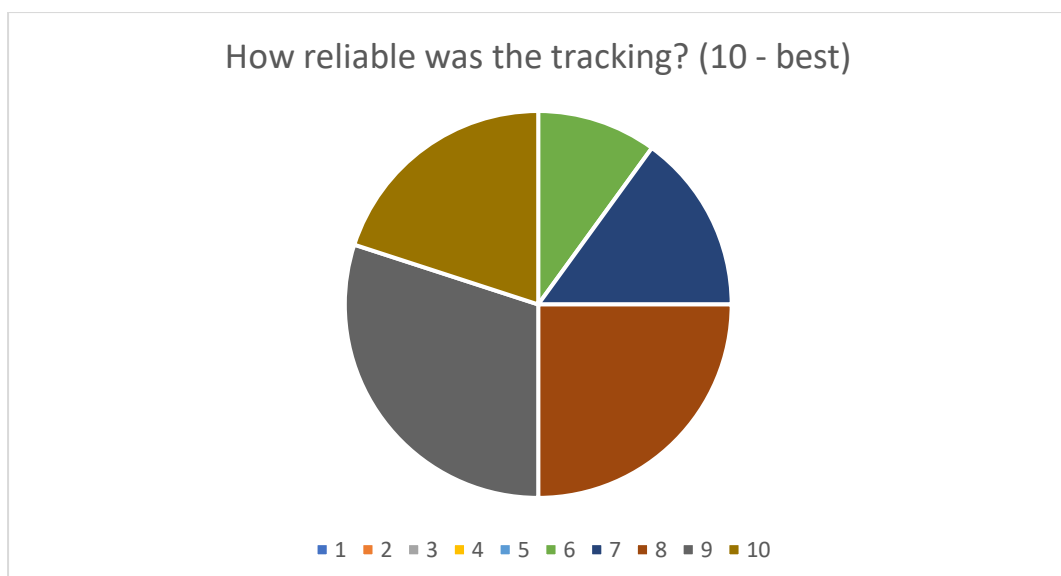
## Questionnaire Survey



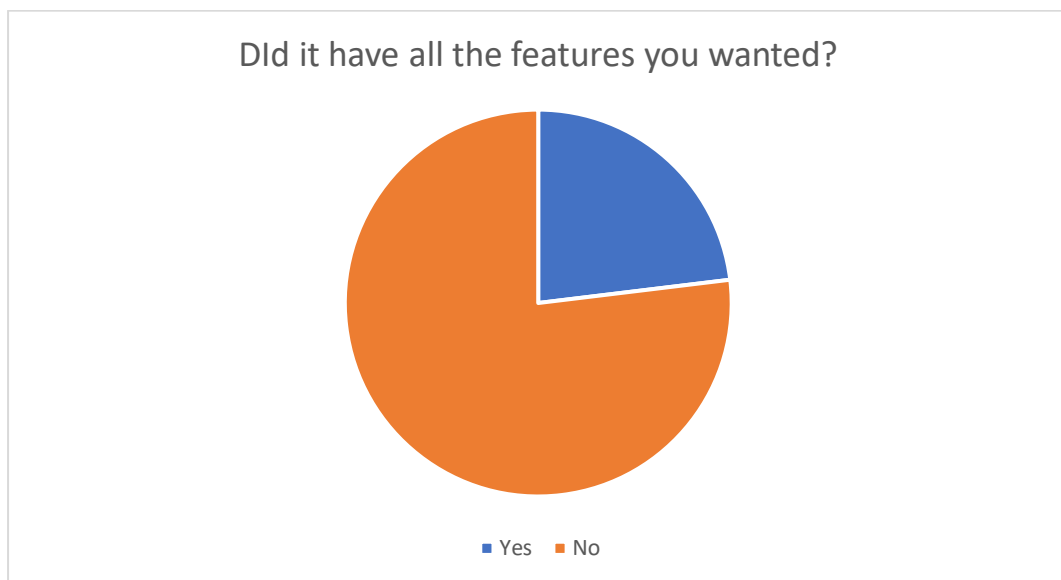
As you can see by the data above of how many times they were used this software in a day can show its success rate. Most of them used it 3 times with it tending to be on the lesser side of the number. This data was gathered after they had their 2 days of using the system in a normal school day which generally consisted of a maximum 12 opportunities to potentially use this tool in the lesson.



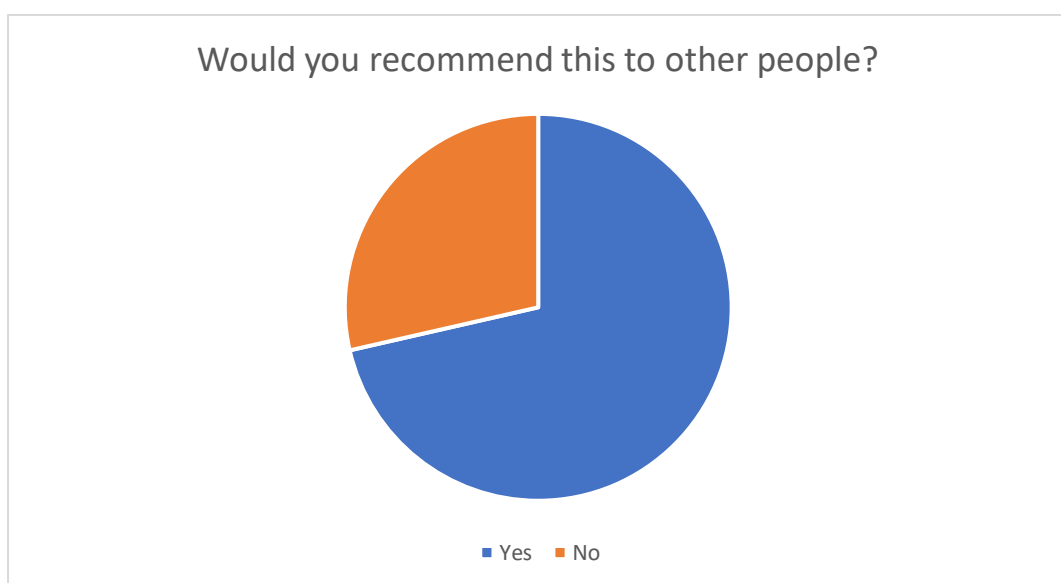
The results show that the almost everyone found it very easy to set up and move about while working. The most common result was 10 showing that it was extremely easy to set up and the data ranged between 7-10 which is towards the easier end of the scale. No one said that the system was difficult to set up.



This data was more mixed and had a wider range between 5-10 but was definably in the upper brackets. The most common score given was 9, which is almost perfect tracking.



From the data we can see almost 75% of people said it did not meet the features they required. This mean that the software was not effective for what the teachers wanted.



Most people with just under 75% responded yes to this question. Making most people wanting to recommend this product to other people to use/ buy.

## Focus Group Survey

These following question were asked with a group of users who have used the product over the two day period where longer questions will be asked and this was held in a focus group, discussion format where questions were asked and it went round the table where users could answer add or add on extra to another person's comments or answers.

How did it compare to other products you have used?

Overall the most contentious was that this product stood out greatly as it was cheap easy to use and was very portable, compared to other products. They said the tracking was on par but had some uses here and there in certain environments. They mainly found it tricky to find a place to mount the Wii remote in a position so that it could see all of the board.

If you could change anything about the product what would you change?

The main points brought up here where that they would have liked to see features such as colours, importing images and changing pen size and also the eraser. They also noted that they would have liked to have a great reduction in lag, and it made the software difficult to use at times. Some also expressed a desire for the canvas to be shared and emailed across to people or students.



What did you like about the product?

All users said the price was highly attractable and almost perfectly priced. They loved the user friendliness of the product of how easy it was to setup and use and that It required no training or instruction and almost work out the box every time. The portability aspect also was brought up by many people during this discussion.

## Drawing Usability Testing

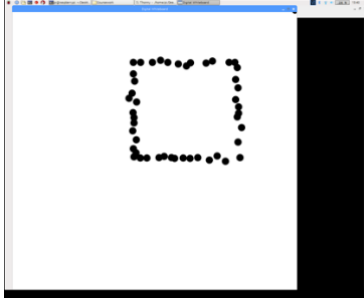
In this section various shapes will be drawn, containing the same set of inputs but with the smooth line algorithm on and off to see if this makes any difference to how the computer handles the data or reacts.

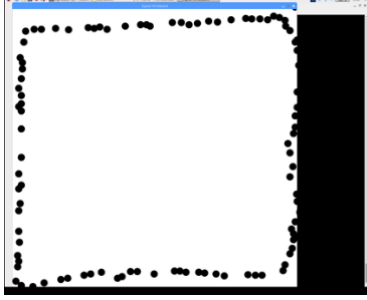
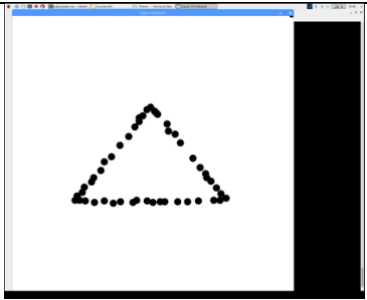
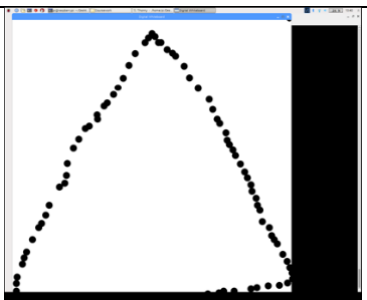
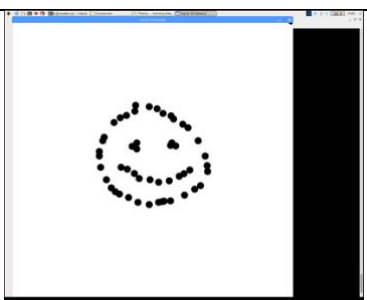
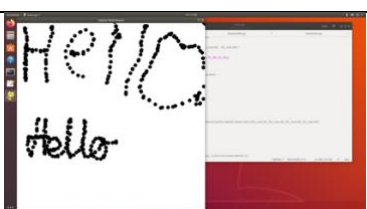
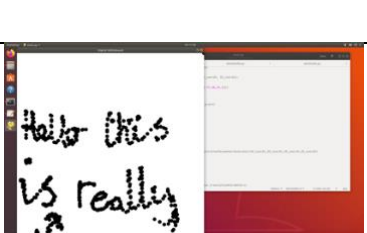
### Smooth Line Algorithm ON

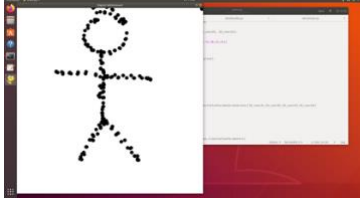
Input	Output	Proof	Pass/Fail
Drawing a square in the centre of the screen	Draws the square		
Drawing a square that covers the entire screen			
Drawing a Triangle in the centre of the screen			

Drawing a Triangle that covers the entire screen			
Drawing a smiley face			
Writing the words "Hello"			
Writing the a sentence of "Hello this is really fun"			
Drawing an image of two stickmen and a tree			
Filling the whole screen up with a line			

### Smooth Line Algorithm OFF

Input	Output	Proof	Pass/Fail
Drawing a square in the centre of the screen	Draws the square with good accuracy		PASS

Drawing a square that covers the entire screen	Draws the square but struggles with the very edges		PASS
Drawing a Triangle in the centre of the screen	Draws the triangle with good accuracy		PASS
Drawing a Triangle that covers the entire screen	Draws the triangle struggles on the edge that is near the bottom		PASS
Drawing a smiley face	Draws the smiley face with good accuracy		PASS
Writing the words "Hello"	Writes hello however letters are hard to differentiate		PARTIALLY
Writing the a sentence of "Hello this is really fun"	Writes the words but writing is difficult and handwriting		PARTIALLY

	ends up larger then normal		
Drawing an image of a stickman	Draws the stickman successfully		PASS
Filling the whole screen up with a line			

## Robustness Testing

This testing is seeing how reliable the system is, and how well it works on different platforms and operating systems. Multiple tests will be carried out under this section to make sure that the software will work in almost all scenarios.

The testing that will be conducted under this section are:

- Testing on different Linux Distributions.
- Testing on computers with different amounts of RAM
- Different screen sizes

# Different Linux Distros Testing

The program was developed and previously tested on a raspberry pi 3B+ and with the OS of Raspbian. This is a popular distro used with raspberry pis. The other distros that will be tested will be:

- Ubuntu
- Raspbian
- Kali Linux
- Fedora
- Debian
- CentOS

These distros were chosen as there are the most commonly used distributions in the world and have a large variety as they change between each one significantly enough that it should work with all other distributions that fit in between these distributions.

Distribution Name and Version	Pass / Fail	Comment
Ubuntu 14.04.3	PASS	Worked more smoothly compared to default OS was easy to install and run.
Raspbian 10.4.19.8.3.1.8.2	PASS	Worked well and reliably. This was used as the benchmark as it was developed on here.
Kali Linux 2019.4	PASS	Downloading modules was hard and pygame installing was difficult.

Fedora 32	PASS	Worked well, as was reliable as the rest nothing worth nothing.
Debian 10.2	PASS	Worked equivalently as Raspbian in setup and package downloading.
CentOS 8.1-1911	PASS	Worked well once installed but pygame had problems installing and required different set of instructions.

## Testing on different amounts of RAM

These tests were not very controlled as other factors were changes as well such as CPU and the architecture and the other hardware. One thing that was kept constant was the operating system which was Raspbian. The exact tests were fairly subjective as there was no real way of quantifying the improvement.

Amount of RAM	Run ability	Comment
0.5 (Raspberry Zero)	FAIL	The program was borderline unusable and became unresponsive after

		approximately 30 seconds of use
1 (Raspberry PI 3B)	FAIL	The program ran slightly better than the 0.5 RAM computer but was still unusable on a day to day scenario
2 (Raspberry PI 3B+)	PARTIALLY	This worked better than the previous two tests and was manageable, but was very irritating and difficult to use but still usable
4 (Desktop PC)	PASS	This worked to a useable latency speed and was running in almost real time.
8 (Desktop PC)	PASS	This was almost exactly the same as the previous test will slightly better results

The code worked well on all machines above 2 GB RAM which allowed for the program to be used effectively, the program ran well and properly on all computer above 4 GB RAM.

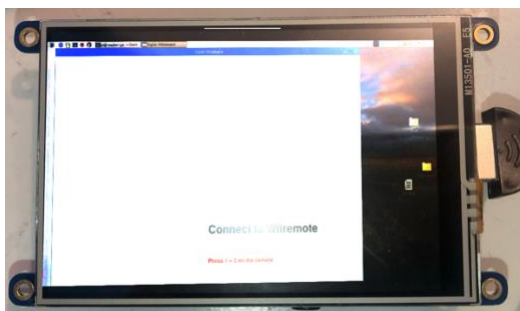
However, during the test other factors were also changed and not kept constant such as CPU used the clock speed, the cache size and the type of secondary storage used which is used as virtual memory. These factors may have played a bigger factor than the RAM on affecting the code's rentability

# Project Budget

This program runs effectively on the 4gb RAM which is a can be bought in the form of a raspberry pi 4 which retails for \$50 this plus IR pen which retails for \$2 and a wii remote which is approximately \$20-30. This makes the total cost of the of the product \$72-82 assuming they are have the other available resources such as keyboards, mouse and cables.

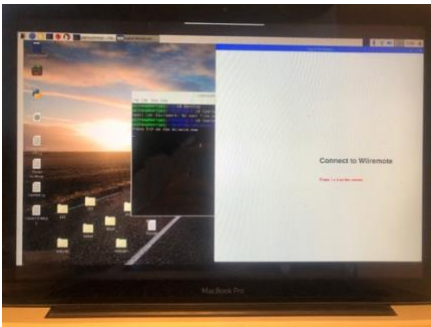
## Testing on Different Screen Sizes

Below you can see evidence and the outcomes of the program running on different screen sizes:





This is on a 3.5” touchscreen that mounts over the footprint of the raspberry pi.



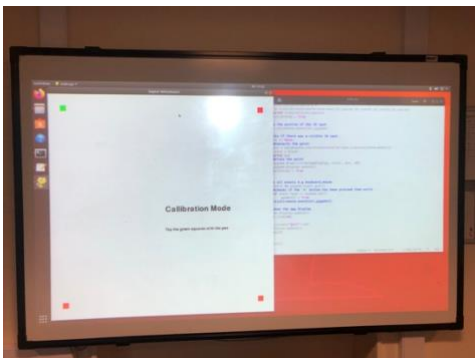
This is on a 13” retina display on a MacBook



This is it on a projector

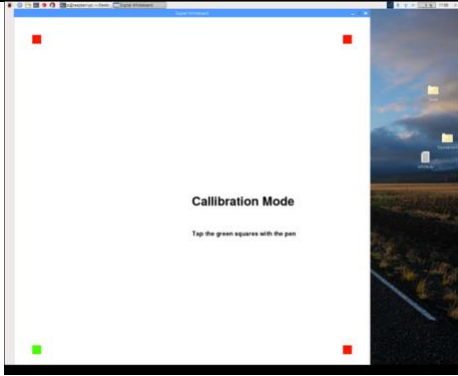
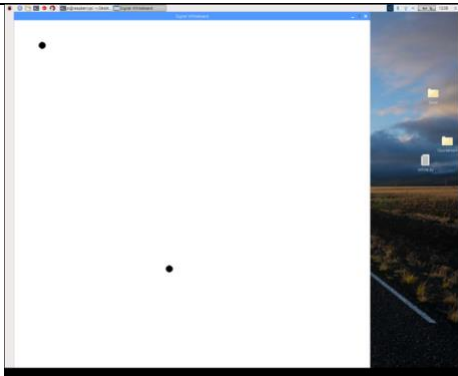
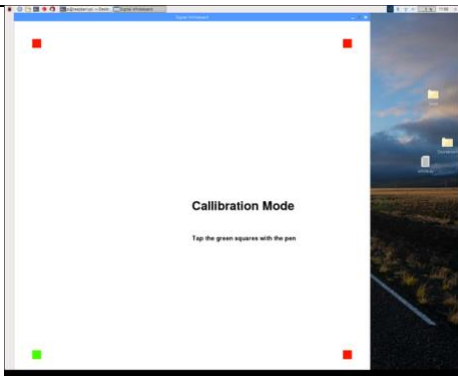
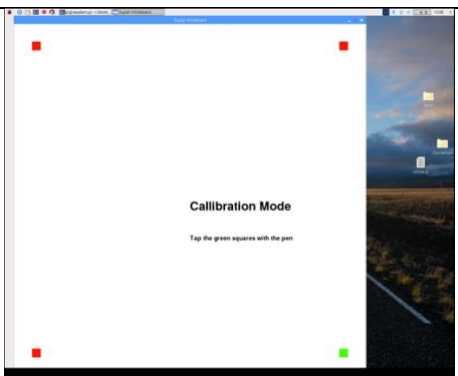
# Calibration Robustness Testing

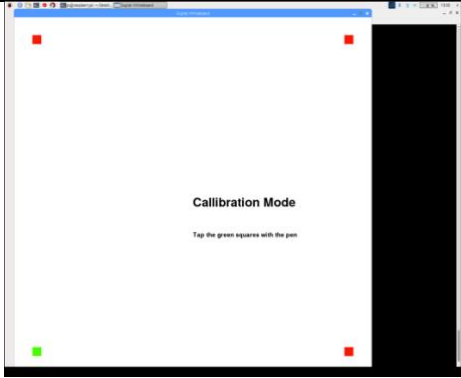
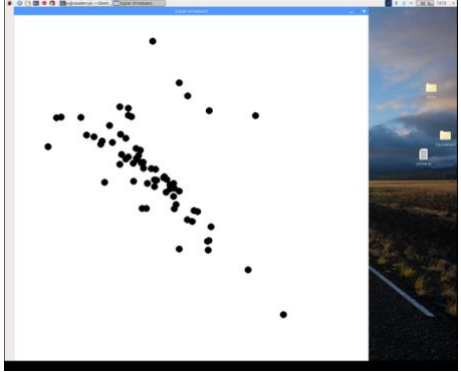
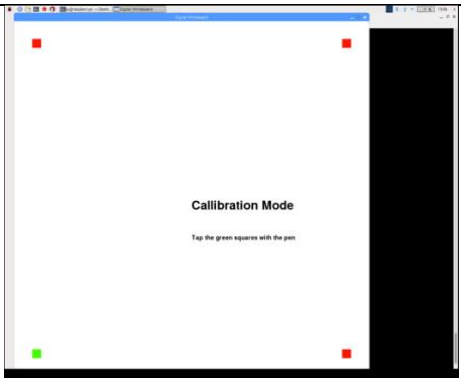
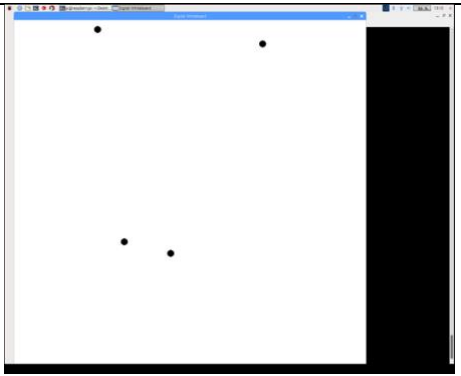
of

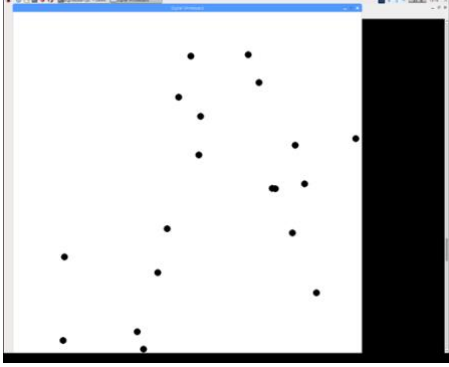


These calibration tests will be done to test how the computer reacts to non-simulated data, ie data received from the IR. The series tests will be conducted to see how the software reacts if the user does something unexpected or does something wrong.

Input	Output	Proof	Pass/Fail
-------	--------	-------	-----------

The same corner was pressed and held for approximately 5 seconds	No translation matrix was generated		PASS
The boxes were done in the opposite direction then what was instructed to do	<pre>[[ -4.89458513e+00] [ -2.81130633e+00] [  4.63439929e+03] [  3.25391946e+00] [  4.43180886e+00] [ -4.43669710e+03] [ -2.55999007e-03] [  1.83330910e-03]]</pre>		FAIL
The boxes were done in a “Z” direction	No translation matrix was generated		PASS
The boxes were done in a “X” direction	<p>No translation matrix was generated</p> <p>Recognised the first 2 points then stopped</p>		PARTIALLY

The same two points that are directly opposite each other were pressed twice	No translation matrix was generated		PASS
The same two points that are directly diagonally opposite each other are pressed twice	[[ -2.15115348e-01] [ -4.66083254e-01] [ 3.01520013e+02] [ -1.99852817e-01] [ -4.72580142e-01] [ 3.03787323e+02] [ -9.15957749e-04] [ -1.24361358e-03]]		FAIL
The centre of the screen was tapped 4 times	No translation matrix was generated		PASS
The points were 4 times on the X axis along the same axis, but spread even on the Y axis	[[ 6.61535589e-01] [ -3.79082191e+00] [ 2.25978327e+03] [ -2.48927310e+00] [ -5.03371014e-01] [ 1.84891621e+03] [ 3.88408979e-04] [ 4.55856345e-04]]		FAIL

The points were 4 times on the Y axis	<pre> [[-1.69967268e-01]  [-1.38806602e+00]  [ 7.98789502e+02]  [-1.54387510e+00]  [-1.93308731e+00]  [ 1.96078624e+03]  [-1.58936293e-04]  [-1.94513025e-03]] </pre>		FAIL
---	---	--	------

As seen by the tests above, none of them should have accepted the calibration in theory however as we can see the software did accept some of them when it shouldn't have.

Where a translation was successfully made the pen was moves around randomly around the screen, and in all of these tests the calibrations did not work practically and then lines that were being drawn were not matching where the pen was.

# Evaluation

Following the development and testing of the developed program (in-development and post development) the finished developed program can be compared to

what was needed to make a successful product, which was defined under the design section and success criteria.

Overall in this section this will evaluate how successful the program was and how effective it is at solving the problem defined in the previous section. The testing data developed previously will be used as a way to quantify and back up with evidence the reasoning for if it has met or not met the points mentioned.

Considering the final prototype the current limitations will be evaluated and stated from the test data. Showing what are the useable boundaries of this project and conditions are need for it to function effectively in.

Since the project is still in prototyping phase, if it were to be commercially developed or deployed certain changes would have to be made, in order for this to happen. This will be discussed of what the order of priority should be and what will have to be changed, in order to make this project go further.

## Reflection on Functionality Testing

Following the testing data developed for each section during development and after in post development the data and feedback from customers can be analysed and

evaluated. This section will look at the program holistically from the functionality aspect of the project. This check can it achieve what it set out to do and solve the problem stated effectively.

## Success Criteria

Success Criteria Code	Name	Proof	Comment	Pass/Fail
SC1	<b>Core Functionality</b>	Video, Range of IR detection, Wii remote connection, User feedback	This was successful on gathering the IR point and did this very accurately as long as the calibration was successful and accurately done. Connection to the Wii remote was very successful and almost had a 100% chance of connection at short distances away from the computer. The main problem was that the system became slow under more use and became unusable at times. The speed of tracking also lacked at time creating a spotty line, rather than a smooth one	Partially
	Is the program able to successfully get data of where the pen is	Pages 160, 161, 162, 164, 169		
SC2	<b>GUI</b>			Pass

	Is the GUI usable and works effectively on all devices	GUI testing, User Feedback, Video, Calibration Testing	The GUI was scalable and was easy to use according to user feedback on how they found the system. The GUI was said to be simple to interact with and the instruction was clear and straight to the point. The GUI plotted the pen and the calibration front-end was successful and was easy to follow and do. However, the GUI did have to have a way to select colour and pen size which was partially developed but not integrated due to constraints in other modules, which prevented this from being able to work efficiently.	
		Pages 170, 163, 164, 165, 166,		
SC3	<b>Budget</b>	Robustness	The total cost of the project came out to be at max \$82 which is well below the \$100 maximum and the retail cost of approx \$300. However all this data does	Pass
	Is the below the \$100 pound development limit	Testing, User Testing and Feedback		

		Page 163, 166	not consider future development costs marketing or developer costs, making this data slightly redundant. However this can still be expected to be significantly below competitors price. Which makes it successful of the project in terms of budget.	
SC4	<b>Environment</b>	Wii remote module testing, User Feedback and Survey	In all of the tests for the Wii remote testing module, it was successful in all reasonable simulated model of the classroom that the product can be used in. This was also shown as highly successful by the users when they tried the product for 2 days to see how it copes. However they brought up points of finding places where it is difficult to mount so the camera can see the IR pen and the board, but since this issue is not directly related to the software end this part of the project can be considered a success.	Pass
	Can it function in a normal environment at school	Pages 160, 161, 162, 84, 85, 86, 87, 88, 89		
SC5	<b>File Saving and Sharing</b>	User feedback and Survey	Although this module was developed with the email validation and designed it could	Fail



	Is the file able to be shared and saved for later use	Pages  137	not be implemented into the current prototype due to the fact that the system simply could not cope with that much amount of data processing in real time causing the crashes to occur. In the future this is a feature that is needed in order to fulfil what the users need as they brought this up as a feature they wanted.	
--	---	------------------	---	--

## Addressing Unmet Success Criteria

From the table above we can see that there are criteria that have not been met or have been partially met that need to be addressed in the next prototype.

Firstly, the main and biggest feature that could not be added was the save and sharing ability, but this was not to do with complications within this modules but rather that the computer was unable to handle so much data processing in real time. This must be considered in the next prototype where there may have to be a total rewriting of the design without the wii remote and the raspberry pi that are possibly sources of bottlenecks slowing down the program. Using an Arduino to handle the data processing would be more effective as it is specifically designed for embedded systems. And allow the data plotting and GUI interface to be handled by the hosts computer. This will however make the overall product price increase but also increase functionality of the product.

In terms of core functionality most of the features were met other then the ability to smoothly draw the line this again can be solved using a better hardware or possibly a

different algorithm for smoothing out the line, which has to be developed that is less CPU intensive and easier to process.

Other ways to improve the functionality is to implement threading and possibly multicore processing as this will allow for the computer to run the program more efficiently and cut down on lag time and make the software appear more real time.

## Reflection on Usability Testing

This section focuses on the front end of the application and how the software appears and is interacted by the user.

### Success Criteria

Name	Proof	Comment	Pass/ Fail
<b>Software is intuitive and easy to use</b>	User Observation	Overall, we could see that all the users could easily navigate through the software effectively, to get to the main user page. Some users initially struggled as they kept to Wii remote down as they were not able anticipate what to do next. But when asked to repeat they knew what was expected of them, so	Pass
Is the program able to be started and used by most users,			

without significant help		they were able to do it effectively. All users said that the software was easy to use and instructions were non-ambiguous making it successful to say that the program is easy to use and set up.	
<b>GUI</b>	User	This was to check if this program was useable by people who may face certain disabilities. Overall all colour-blind people could use the software easily as there was only 2 main colours red and green. Some short people faced problems touching the top part of the calibration but, this can't be fixed with code, as they have to manually readjust it, for their environment	Pass
Is the GUI usable by all users	Testing, User feedback		
<b>Did the users understand all the features?</b>	User Focus Group, User Feedback	This was only partially testable as most of the features were unable to be developed due to problems with hardware that caused limitations with the amount of processing. They were able to understand drawing and calibration successfully. Users pointed out that they wanted features such as changing colour and size of the pen and a way to clear the board. All users highlighted the need of a way to share the files or the sketches they have generated. Most of these features were already considered in the design section and partially developed just not integrated into the final prototype. Some features were however not considered initially that need to	Partially
Are they able to use all the features of the product successfully and did it meet what they wanted			

		be developed as per the user recommendations, such as recalibration.	
<b>Positioning of the Wii remote</b>	User feedback, User focus group, User observation	Most users were able to find a position that was able to fit then entire screen. Almost all users chose to go almost perpendicular to the screen and straight in front of it, even though it could cope with angles to a certain degree. Some did struggle on how to position and keep the remote in a proper position, that covers the whole screen, but after some trial and error on the users behalf they eventually got it into a position where they were able to use the program.	Pass
Did the IR camera have to ability to successfully cover the screen from the users convenience of placing it down			

## Addressing Unmet Success Criteria

Most of the criteria was met under this section, as it passed successfully. All of the main criteria were hit.

However, most of the testing on the users integration with the software could not be tested as majority of the additional features were not developed entirely and integrated into the prototype making it had to be holistically evaluated by the user.

This section would have to be retested heavily under future version as this will be a point of major focus as more features are added, while trying to keep the user interface minimalistic and clean and simple to use as most people from the feedback said they liked this element of design. Causing a good and successful user experience.

## Current Limitations

In the most advanced prototype, there are certain limitations with the product. These limitations stem from algorithms, technology used and the overall approach to design. In this section hardware limitations will also be discussed and evaluated on how this impacted the program.

From the data shown in testing, the limitations with the hardware is that the Wii remote cannot be kept further then 5 meters away from the board as this reduces the visibility strength for the Wii remote that causes inaccuracies in tracking when using the program. The Wii remote has to also be kept approximately 30° from the centre of the board to allow for the best visibility for the screen. The Wii remote can also be held at many angles however at a very specific rotational angle and distance from the screen it

will cause calibration error, however the likelihood of this occurring is near impossible and can hence be overlooked currently.

In terms of the IR pen this is a very simple device and the more powerful and intensive IR diodes used will result in better visibility and hence better tracking.

The program is also shown to run better the more ram that it has available.

In terms of software algorithms, the Wii remote currently can only track 1 IR point even though the Wii remote has the capability of tracking up to 4. It simply ignores the rest of the visible IR points.

The software also only works on Linux based operating systems therefore not working on MacOS and Windows which causes this software not to be accessible by some users who may want to use this product.

## Maintenance

For the future developments all parts of the code have been commented with the function they provide and has been fully documented in this report. Not only that but all key variables are also listed in a table stating their usage and data type they are.

In the future every change that is made should be documented and the date written on; who made the change; why the change was made. So, it is easier to maintain the code and know exactly who did what and why.

All code was developed in individual modules and classes that allow for each one to be independently changed and for it not to affect the other sections of code. Also when working in teams this can allow for multiple people to work on the same project at the same time, as each person can be assigned a class or a function that need to be developed.

When developing the current code and prototype, libraries that have been used have been noted and been included stating their purpose and version used. What must be considered is the use of the library `cwiid` which is built upon proprietary code and protocols developed by Nintendo and is seen as a hack or a work around. This module was last updated 7 years ago so support on future version and updates are very likely to not occur, not only this but documentation on this module is near non-existent, with only a couple of example scripts but nothing more advanced or organised then that.

## Further Development

If this product was to be developed commercially deployed or used in a wider context these issues have to be addressed to make it fully functioning and make it a better overall product.

The lag and responsiveness need to be minimised to make it smoother and more real time, to follow the users pen. This can either be done by software optimisation and using different algorithms or writing the code in a more low-level programming language to make it more efficient. Or this can also be solved by changing the hardware and connecting to the IR sensor directly by the wire reducing transition latency. This can

possibly be achieved by using an Arduino or a dedicated embedded system board that is designed to operate in real time.

In terms of software the system can be re designed to operate on other non-Linux based systems such as MacOS and Windows allowing for it to be used by more people. This may mean rebuilding the connection script and not using the library cwiid as this only works on Linux however the rest can be kept the same.

The software also needs to add features such as sharing and saving files as this was viewed as the most wanted feature that the software is currently missing. This would only be added after addressing the other issues first. Other features that have been requested for is changing pen size, colour, adding images. This will make it more appealing to more users making it have a larger user base.