University of Staffordshire

Beyond Buttons: Rethinking Control

Alex Wood – 22013321

Game Development Project

University of Staffordshire

Contents

Introduction

Problem

The problem this project aims to solve is figuring out how controllers and different input systems can impact immersion.

Evolution of Project

During this project, it has naturally grown and shifted in how it was carried out, as such the method of tackling the project has evolved with it.

The initial proposed method of solving the problem set out was creating numerous control methods and getting people to play the same game using these different methods, however after some idea feedback and minor testing it was decided that this project, whilst effective, would not form a fun experience and as such engagement with the project would suffer.

To remedy this the solution was changed to a clearer alternative.

Proposed Solution

The proposed solution to the problem is by constructing a controller with multiple different input methods,

Testing Methodology

This project will be tested by allowing people to playtest throughout its development then answering questions proposed in a pre and post playtest form

These playtesting sessions will take place in numerous different environments as that will allow me to get a wide variety of different people to give their feedback, allowing for a wider range of audiences removing bias from the results.

Narrative

Bomb Defusal

The original narrative for the first game, is a bomb defusal, where the player would have a tool in real life, which helps them defuse a virtual bomb in the game.

This physical controller would have lights and sounds, and a receipt printer which would give instructions from the game on how to defuse the bomb, taking a lot of inspiration from games like “Keep Talking and Nobody Explodes”.

Astronaut Guidance

Following a meeting with my FYP supervisor, we figured it would be best if I changed the games core narrative arc to make it more brand marketable, and the change helps incorporate other components which originally were going to be in separate games.

The new narrative is as follows:

> You are an engineer for an experimental space exploration company, recently your team launched an interplanetary explorer, into space. During a simple space walk they were hit by a bit of space debris, and they lost the ability to control their jetpack.
>
> In a panicked frenzy they call space control for help, but you are the only one with the kit to help; use your B.E.A.C.O.N (Briefcase Emergency Assistance for Cosmic Operations and Navigation) to help guide the astronaut back to the shuttle before they run out of oxygen or fuel.

This narrative shift towards this idea incorporates the pre-existing briefcase that I have made, with very little code changes at this point, but it also includes the joystick, which before this shift I was going to make its own game.

Research

Plugins

SerialCOM

(VideoFeedback, n.d.)

The SerialCOM plugin for Unreal Engine allows for serial communication, it adds a few functions such as opening and closing serial ports, read and writing data to them as well as some functions for converting data types such as bytes to floats.

In the project I use this plugin extensively as it is the easiest way to communicate with the Arduino, for any functions that are used from the plugin in screenshots, they will be marked with a red comment with the tag "@REF – SerialCOM".

RawInput

RawInput is a very useful plugin that allows me to interpret DirectInput as a gamepad controller. This is very useful as it means I can get whatever devices I want and use them for the game irrespective of hardware so long as it has DInput encoding.

Throughout the project I use a "Logitech 3D Extreme Pro" which the input mapping of is handled through the plugin.

Components

Ranging Sensors

Ultrasonic Ranging Sensor

An ultrasonic ranging sensor is a distance measurement tool which uses sound to measure. It fires an ultrasonic wave of sound and waits for it to return, then it bases the distance based off of how long it was waiting.

This type of sensor is cheap, easy to use and pretty reliable for small applications.

This sensor has drawbacks though in that sound waves are easy to manipulate and not super reliable when using it against textured surfaces as the sound waves can be distorted into other directions.

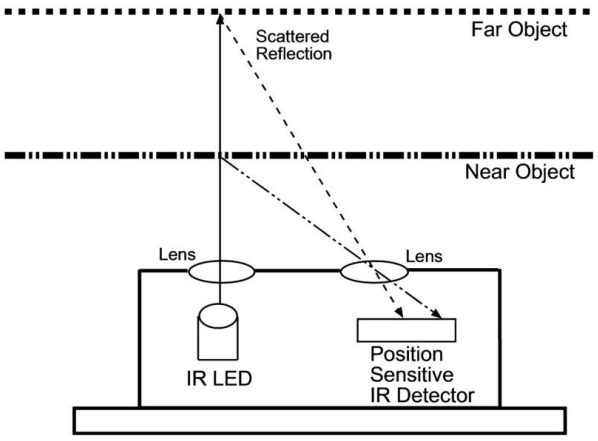
Infrared Distance Sensor

An infrared distance sensor is an alternative method of distance calculation, it uses infrared light and measures angle of return to gage distance. As shown in the diagram below, the IR LED fires a beam of infrared light, and when it is reflected into the secondary lens, the position sensitive IR detector depending on where the beam is received, determines how far the beam must have travelled.

Benefits of this system are that it is better for complex objects as light waves being faster than sound waves can get more accurate readings.

Downsides of this type of sensor is that it can be more expensive than the ultrasonic, and it is unsuitable for larger distances as the position sensitive IR detector can only support up to 80cm traditionally.

"IR distance sensors work through the principle of triangulation; measuring distance based on the angle of the reflected beam" - (SeeedStudio, 2024)



Comparison

Type	Ultrasonic	IR	LIDAR	ToF
Suitability for Long Range Sensing	No	No	Yes	Yes
High reading frequency	No	No	Yes	Yes
Cost	Low	Low	High	Moderate
Suitability to use for complex objects	No	Yes	Yes	Yes
Sensitive to external conditions	Yes	No	No	No
3D imaging compatible	No	No	Yes	Yes

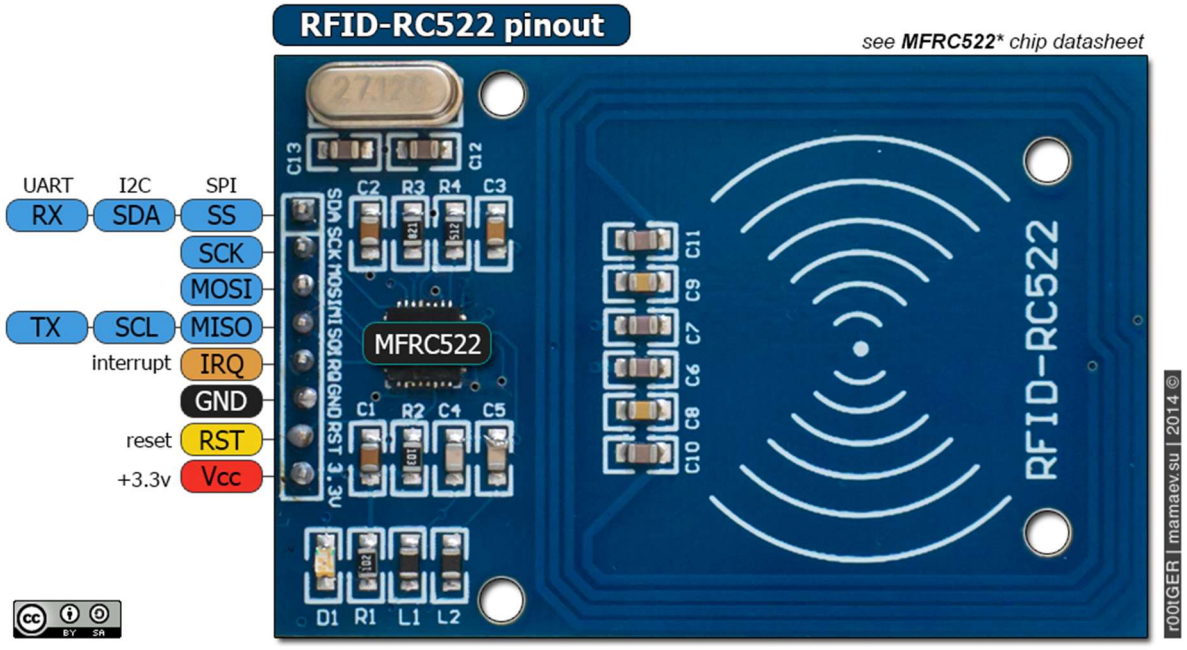
(SeeedStudio, 2024)

NFC

Introduction

Near Field Communication (NFC) is used to communicate small bytes of data over short distances using two components which aren't necessarily connected in a circuit. NFC uses small silicon wafers embedded into the circuitry to alter an electromagnetic signal to send data.

The chip I have experimented during this project is the RFID-RC522.



This small chip allows easy read and write to an NFC chip.

In the game I am using the RC522 to read the Unique Identifier of the NFC cards so that the player can use their student IDs (or an alternative provided NFC card if they prefer) to interact with the game.

Experimenting

Since I had never used an NFC chip before, I wasn't familiar with the particular code required to operate it, as such I used example code from different libraries to get it together.

The library I ended up using (MFRC522) - (Balboa, 2025) which contained code I needed to operate the chip and that example code.

Read NUID Example Code

Reading information found in this example code I managed to extrapolate the useful functions and adapt them to my code, specific functions like isNewCardPresent() & ReadCardSerial() are used to determine whether or not a card can be read, GetType() being used to determine the type of chip present (this is important as things like animal microchips and bank cards operate on different frequencies that readers like this cannot access).

Final Code

```
void RFID_Read() {
  if (rfid.PICC_IsNewCardPresent())  // RFID Card Detected
  {
    if (rfid.PICC_ReadCardSerial()) {  // Can RFID be read

      // Check is the PICC of Classic MIFARE type
      MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
      if (!(piccType != MFRC522::PICC_TYPE_MIFARE_MINI && piccType !=
MFRC522::PICC_TYPE_MIFARE_1K && piccType != MFRC522::PICC_TYPE_MIFARE_4K))
{  // Is RFID Type a readable one (usually yes)

        //Print out detected RFID UID
        Serial.print("R-");
        printByteBufferAsHex(rfid.uid.uidByte, rfid.uid.size);

      } else {  // RFID Type was not a readable one
        Serial.println(F("RFID_ERROR: NOT MIFARE TYPE"));
      }
      rfid.PICC_HaltA();       // Stops attempting to read the data
      rfid.PCD_StopCrypto1();  // Stops binding to previously read card
    }
  }
}
```

Receipt Printer

Data Formatting

The receipt printer only has 1 data pin, and as such it can be difficult to have lots of different options, as such the printers usually use a parameter formatting system (like the code I wrote below for the Arduino) this means that sending a character like “@” wont print an “@” however it will tell the printer that the next byte sent could dictate whether or not the text following is bold, underlined etcetera

So, to print something, an example command could be “@8KHello World!”

These command keys are figured out using the spec of the printer usually, and resources like Adafruit have libraries for common printers.

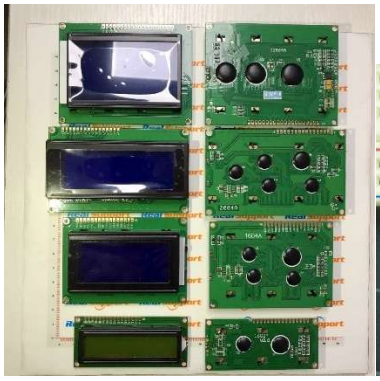
Displays

LCD

LCDs are a simple easy way to display things to the player. There are a few types worth mentioning and talking about.

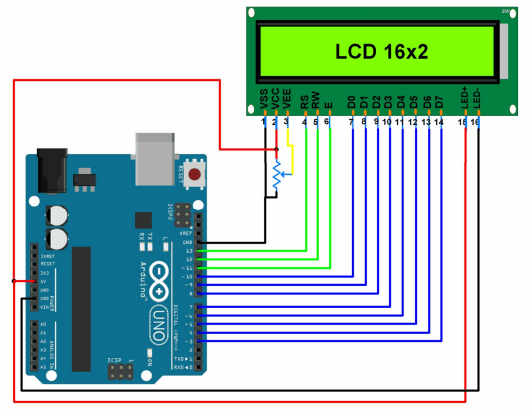
LCD Screens

LCD Screens are the easiest way to display alphanumeric characters to a player. They come in many different formats, usually being either 16x2 or 20x4 in number of characters they can display and having a backlight with a few different colour options.



The issue with these generally is they have a lot of pins as they work using parallel data processing where pins 7 to 15 are loaded with bits, and then a clock signal processes these and converts the 8-digit binary number (byte) into a characteristic, whether it be a letter, number, formatting code etcetera.

This issue can be easily solved by using what is known as a i2c backpack.



I2C Backpack

The I2C backpack (Inter-Integrated Circuit) is a component which mounts to the back of a lcd screen and converts serial information into parallel. So if I wanted to use the byte {01101110}, without it I would need 8 separate pins connected, but with the backpack connected, I simply need a data and a clock pin, and on the data pin I just send {0} then a clock signal, then a {1}, another clock pulse, etcetera, this does have the downside of being 8x slower however when the board is running at 115200 baud (115200 electronic processes per second) it is not at all noticeable; however it does mean that instead of the 16 pins an LCD usually requires, I can just use 4 {Power, Ground, Data, Clock}

Connections

Dupont Pliers

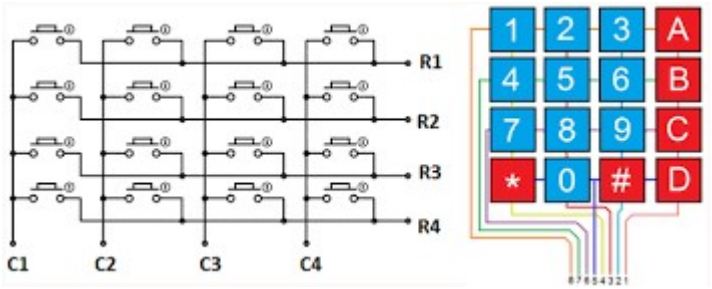
Dupont Pliers are a way of taking raw wire and creating pins on it that can be put into a circuit board or breadboard, this is super useful as it allows prototyping without the need for soldering (which very much benefitted me).

Inputs

Matrix Keypads

Matrix keypads are a way of having a lot of button inputs without the need to connect a wire for each one individually, as it uses a matrix pattern. This is

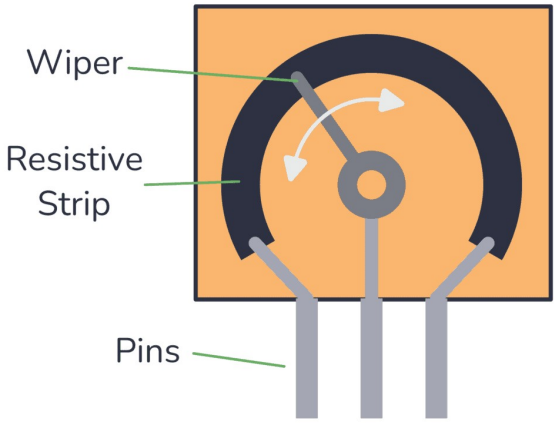
done by sending various amounts of voltage through either the column pins or the row pins, and then whenever a voltage is detected on the other side, this is measured, i.e. using the below diagram, if C1 is supplied with 3.3v and C2 is supplied with 2.5v if R3 reads as 3.3v then button 7 was pushed, however if R1 is reading as 2.5v then button 2 was pushed.



Potentiometers

Potentiometers are a great way to get a variable input from a circuit, as they have a dial which can be rotated to get an input from 0, the voltage supplied (interpreted as a 255 (full byte) in the programming) these work by using a slider which goes across a resistive track.

This results in the electrons moving from the input pin, and travelling along the resistive track, and then whenever they reach the wiper pin travelling down and from here it is able to be measured, as moving the wiper means the electrons have to travel further along the resistive track.

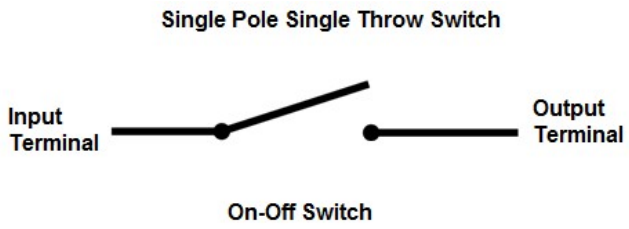


Key Switches

Key switches are a simple way to add input and have it different from just a standard switch. They have just two wires, for power and ground, however if

the power line is connected to a data pin on an Arduino, using an {INPUT_PULLUP} when initialising the pin, it can be used as an input.

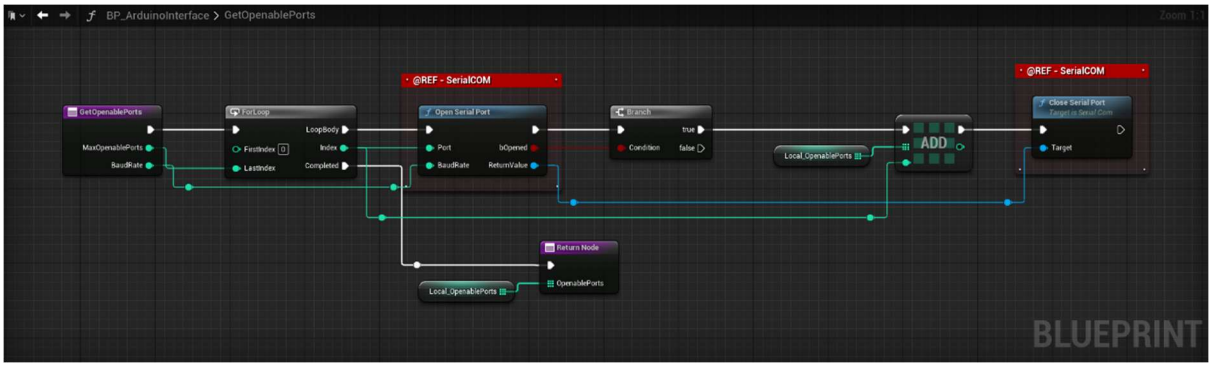
The switches are SPST (Single Pole Single Throw) which means that there is only 1 in for data and 1 out for data.



Unreal Development

Arduino Interface

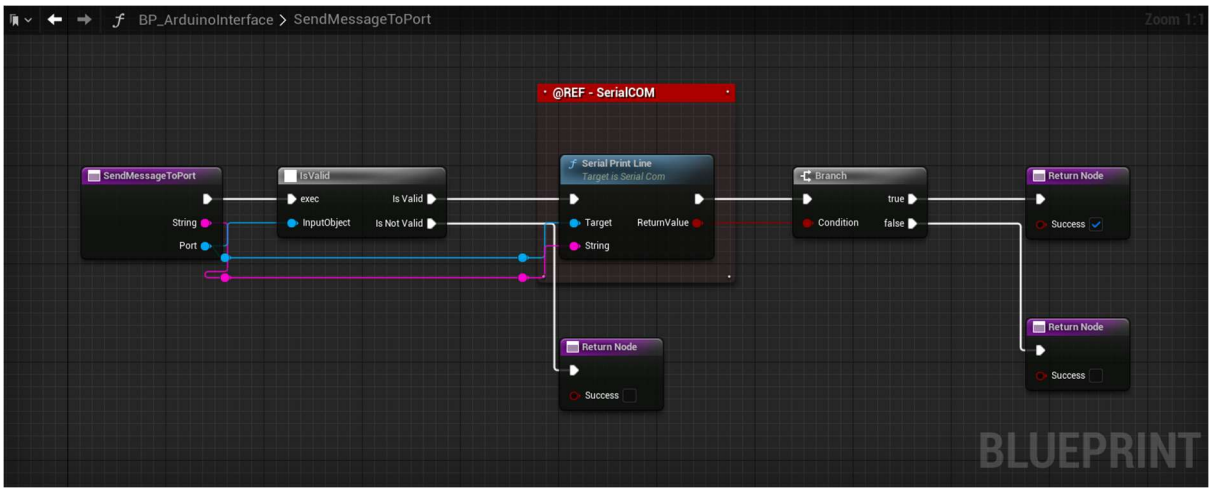
GetOpenablePorts()



The function above uses the SerialCOM plugin to check which ports can be opened, this works by on a for each loop attempting to open them, if they can be opened, that port ID is added to an array and then closed.

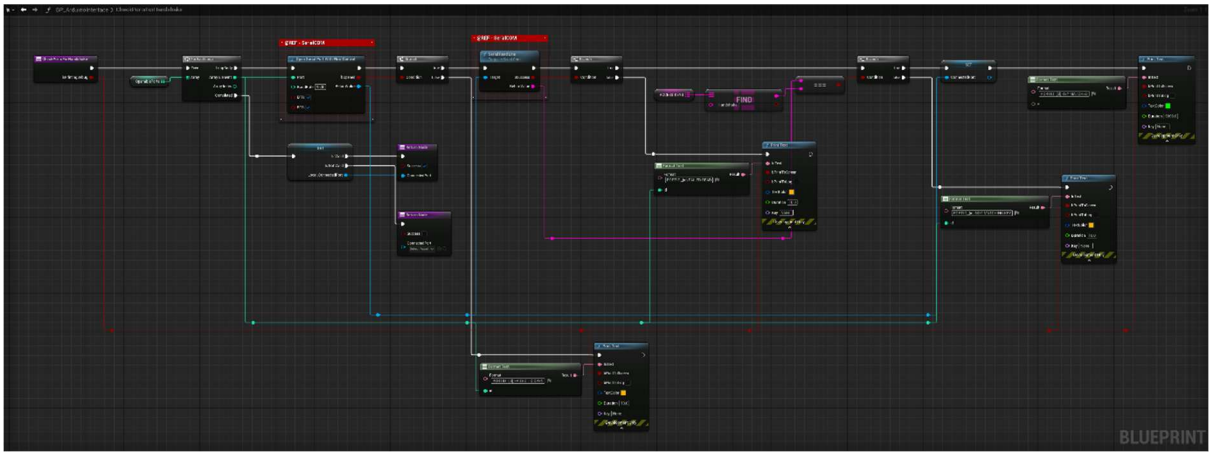
At the end of the function the array of ports that can be opened is returned.

SendMessageToPort()



The above function is a general use function that first checks if the port given is valid and active, then it attempts to write a line to it. If either of these things fail, then the function returns a false success Boolean.

CheckPortsForHandshake()

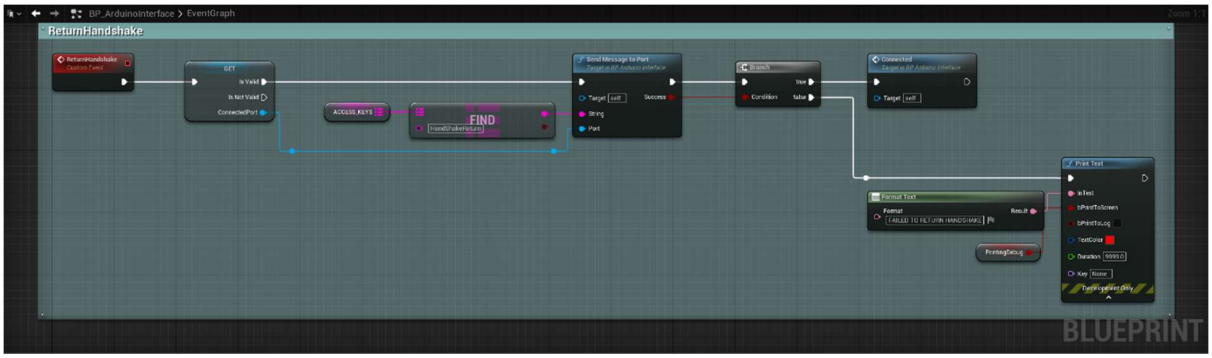


The above function takes the existing array of port id's that can be opened, and then re-opens them, and attempts to read whatever the port has put in the serial.

If the function manages to open the port, and read the data in it, it then checks if this matches the stored key, this is mostly a redundant check as other systems that are using the serial ports would not allow their data to be read without an initialiser and would lock to themselves however this added check just makes sure no data is lost during connection.

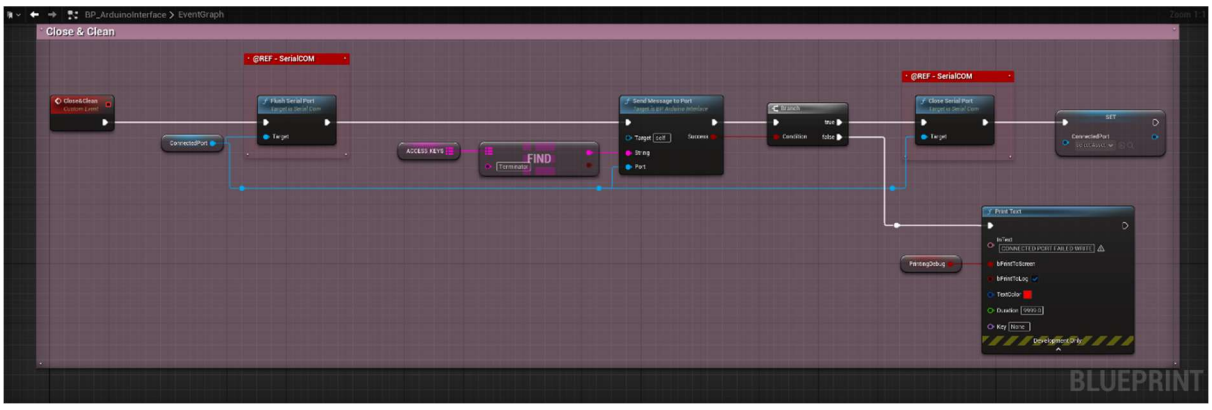
If the handshake key matches the expected handshake key, then the reference to the port is returned out of the function.

ReturnHandshake()



Once a port has been identified and it has returned the handshake key, then Unreal attempts to write a line to it, telling the Arduino it's been connected to. Since unreal has read the handshake key already it knows it is valid, however this is more so that the Arduino can start spitting out gameplay relevant code instead.

Close&Clean()



Since a port can only have a single listener at any given time, this function makes sure that whenever a port is open and saved, at the end of the game it is closed to make sure that next time Unreal Engine attempts to access it, it can be correctly accessed without thinking that something else is listening to its serial.

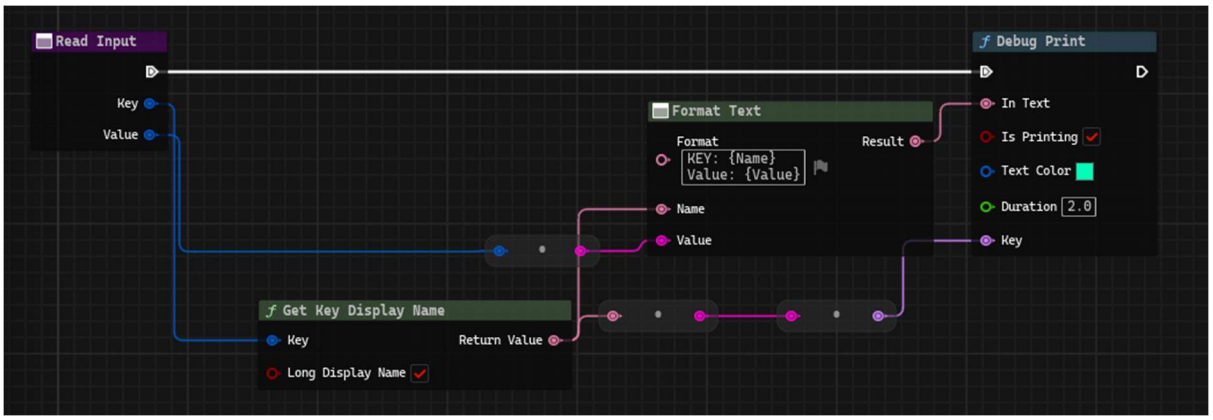
Joystick Development

Following the narrative shift away from the original idea where I was defusing a bomb, the new idea being an astronaut guidance system; the

joystick is used to move the astronaut from being stranded in space to inside the shuttle.

Input Mapping

As I'm using RawInput, a feature of the plugin is that it can support multiple controllers, however the downside of this is that the mappings that Windows interprets the controller as aren't always the same inside the plugin, as such I created a small function which detects which keys are being pressed and can spit out their output. This allows me to deduce which keys are being pressed at any given time.



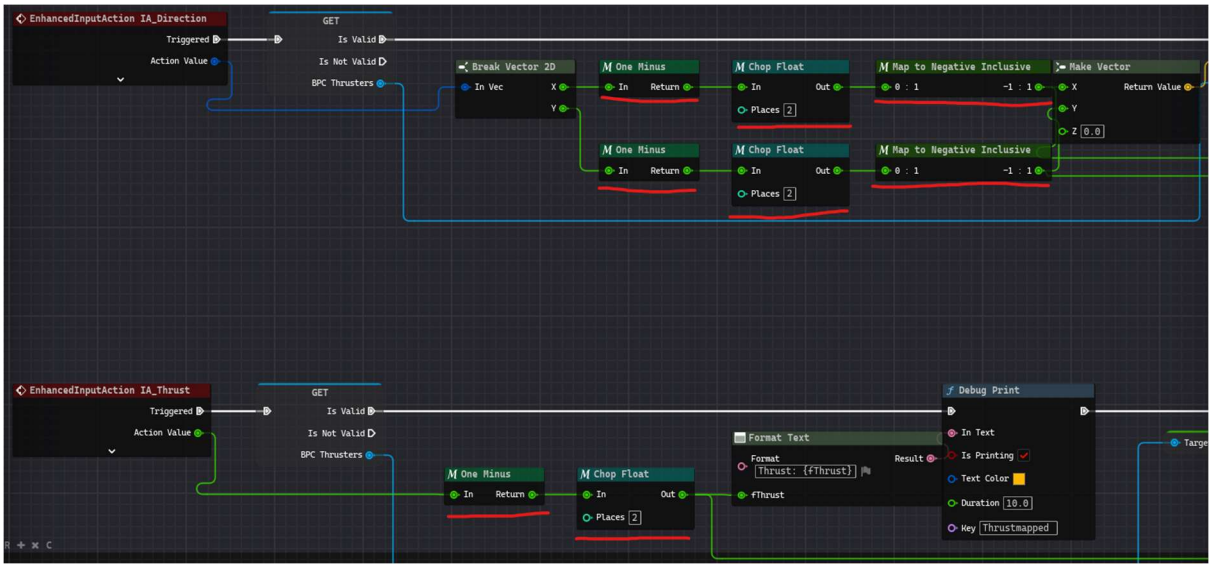
Bad Linking / Disconnecting

Unreal Engine Blueprint doesn't have great support for DirectInput based devices, as such if the controller is disconnected, then it just doesn't know, and as well if the device has any softwareIDs changed (found in registry editor) then this can cause the controller to just not be detectable.

I managed to overcome this by assigning the controller in the RawInput settings using its hardware product and vendor IDs.

Interpretation of Input

As games designed to use these controllers will have direct support for them embedded by using libraries from developers usually this is not something they have to worry about, however as the numbers come through as raw 0-255 from RawInput sometimes it cannot be the interpretation you would expect. For me at least I had to manually remap the inputs from {0 to 255} to {-128 to 127} this requires doing some bit magic as it uses twos complement however the functions underlined in red in the below image are the functions I created to do this.



Arduino Development

Unreal Interface

Handshake Loop

```
#define HANDSHAKEOUT "HANDSHAKEARD"    // What Arduino Sends for Unreal Engine to Recieve
#define HANDSHAKERETURN "HANDSHAKEUE"  // What Arduino is Expecting to recieve
#define TERMINATEKEY "TERMINATE"       // The String required to disconnect
```

```
while (!bCONNECTED) {
    if (Serial.available() > 0) {
      String _read = Serial.readString();
      _read.trim();
      if (_read == String(HANDSHAKERETURN) && bCONNECTED == false) {
        bCONNECTED = true;
        digitalWrite(HANDSHAKELED, HIGH);
      }
    }
  }
  HandleInput();
```

I need the Arduino to know when it is connected to unreal engine and vice versa, as such a simple handshake system is what I'm using. How this works is whilst the handshake hasn't been fulfilled, it listens to the serial port to see if "HANDSHAKERETURN" is interpreted, this is a definition key defined at the top of the header file, this is so that if it needs changing to something else then I don't have to go hunting through my code.

After the handshake loop has been completed, the function "HandleInput()" is run on the loop, this function interprets anything unreal engine might send towards the Arduino.

Peripheral Interfacing

Printer

Initialisation

(AdaFruit, 2024)

```
//Class Setups
SoftwareSerial PrinterSerial(RX_PIN, TX_PIN);  // Declare SoftwareSerial obj first
Adafruit_Thermal printer(&PrinterSerial);      // Pass addr to printer constructor
```

```
void RawInitPrinter() {
  PrinterSerial.write(27);  // ESC
  PrinterSerial.write(64);  // @ - Initialize printer
  RawPrintSettings();
}

void RawPrintSettings(int HeatingDots, int FeedTime, int HeatTime) {
  PrinterSerial.write(27);           // ESC
  PrinterSerial.write(55);           // 7 - Print settings
  PrinterSerial.write(HeatingDots);  // Heating dots (7 - max heat)
  PrinterSerial.write(FeedTime);     // Print and feed time (100 * 10 microseconds)
  PrinterSerial.write(HeatTime);     // Heating time (20 * 10 microseconds)
}
```

The printer needs to be initialised using a serial port, so I create one using the Software Serial library, after this I initialised the object of the printer on the Adafruit class, this gives me access to a lot of the functions native to the library.

RawInitPrinter() sends information to the printer through the serial port using ascii characters, these characters are deciphered from the datasheet from the printer.

RawPrintSettings() does a similar thing, writing raw data to the printer, the reason I'm doing this as opposed to using the standard setup functions native to the Adafruit library, as there are some differences between different types of printers, the Adafruit library is designed to work with their own, so using their function prints out garbage characters on the printer.

Data Formatting

```
/*!
* @brief Prints to connected printer with formatting options
* @param Format 6 bit byte dictating styles, consult function for style guide
* @param text text to print
*   FORMATTING STYLES : = 0-63
*   1 - justification - 0 = Left / 1 = Center
*   2 - bold - 0 = Off / 1 = On
*   4 - size - 0 = Small / 1 = Large
*   8 - underline - 0 = Off / 1 = On
*   16 - inverse - 0 = Off / 1 = On
*   32 - double height 0 = Off / 1 = On
*/
void PrintOnPrinter(byte Format, String text, bool AutoFeed = true);
```

When using the PrintOnPrinter() function I wanted to make a system where I can send small codes over and print using them, reason being the Arduino isn't the fastest device so smart data packaging is a good idea, for different formatting options they are usually on or off, as such packaging them into a 6 bit byte (a bit for each style option) seemed like a smart idea,

So, if a person wanted to send a code where it was bold, inverse and double height, instead of sending a long-winded code specifying all of those instructions, they can send just the number 50, which is 010011 in binary, with a 1 being active for each style option.

```
  case 'p':                                          // printer commands

    _read.remove(0, 1);                              // removes command key

    byte formatting = byte(_read.substring(0, 2).toInt());  // first two characters after p command dictate formatting using bit shifting - 0-63

    String text = _read.substring(2);

    PrintOnPrinter(formatting, text, true);

    break;
```

Including the above style byte options, printing codes are simply formed, a 'p' indicating the following message is a print function, the next two characters in the string are the style options (00 – 63) and then any text that follows is to be printed, as such a code received that looks like "p39Hello World!" would print "Hello World!" using a centre aligned bold font, at double height.

Printing

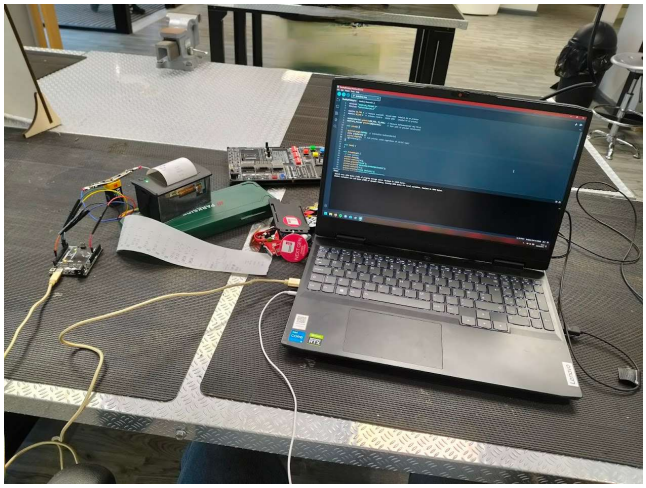
```
void PrintOnPrinter(byte Format, String text, bool AutoFeed) {
  printer.reset();  // reset previous instructions

  if (bitRead(Format, 0)) {  // 1 - justification - 0 = Left / 1 = Center
    printer.justify('C');
  } else {
    printer.justify('L');
  }
  if (bitRead(Format, 1)) {  // 2 - bold - 0 = Off / 1 = On
    printer.boldOn();
  } else {
    printer.boldOff();
  }
  if (bitRead(Format, 2)) {  // 4 - size - 0 = Small / 1 = Large
    printer.setSize('L');
  } else {
    printer.setSize('S');
  };
  if (bitRead(Format, 3)) {  // 8 - underline - 0 = Off / 1 = On
    printer.underlineOn();
  } else {
    printer.underlineOff();
  }
  if (bitRead(Format, 4)) {  // 16 - inverse - 0 = Off / 1 = On
    printer.inverseOn();
  } else {
    printer.inverseOff();
  }
  if (bitRead(Format, 5)) {  // 32 - double height 0 = Off / 1 = On
    printer.doubleHeightOn();
  } else {
    printer.doubleHeightOff();
  };

  printer.println(text);
  if (AutoFeed) { printer.feed(1); }
}
```

The above code is what translates those instructions into the print instructions that the printer can understand, and then ‘AutoFeed’ just ejects

another line out, so it isn't kept inside of the printer itself where it can't be seen.



Power

The receipt printer attached requires power to operate, as such it has a place to attach a 9v battery in the back, I did this and plugged the TTL connection into the Arduino which in turn was connected to my laptop, however what I didn't think about was whether or not the receipt printer had any surge protection, which I have now found out that it doesn't. I found this out by changing out the 9v battery and it sending the 5v from my laptops USB connector + the 9v from the battery back into the laptop and short circuiting my laptops graphics card permanently rendering it essentially useless (☹️)

Moving forward from this I have now plugged the extra power the receipt printer requires into the Arduino itself which works a lot better because that actually has surge protection.

Physical Representation

Version 1

To sell the narrative I wanted to make the Arduino have a housing, this both protects it and looks a lot better. I figured that a briefcase would be a great way to store it as it is rigid, spacious and easy to carry.

I sourced a fantastic briefcase, and mounted all of the components inside using a concerning amount of duct tape, later on I will laser cut some wood to make a better looking mounting but for the development it looks pretty

good already, inside is the Freenove Projects board which features a lot of the useful components I will be using for the project, as well as the receipt printer, as development continues I may mount more devices inside.

Version 2

Version 2 of the controller is functionally similar except instead of all being on an integrated board inside of a briefcase, it is now all wired together as separate components, mounted into a laser cut board of wood, and mounted inside of a purpose made case.

This new case has been so much better as not only does it really sell the narrative that I am going for, it also is a lot more stable and less fragile than the briefcase.

This version includes 4 key switches, a 20x4 LCD, a keypad & an RFID reader.

Finance

Production Costs Over Development

Because of the nature of the project, it has incurred some costs through its development, I feel its important to showcase these.

Item	Cost
Replacement Laptop	£650
Arduino Mega	£18
Key Switches	£5
Receipt Printer	£25
RC522	£4.50
Keypad	£3
Freenove Projects Board	£50
Dupont Crimp Connector & Kit	£30
LCD-2004 i2c	£10
V1 Briefcase	£0
Nanuk 910 Case	£80
Soldering Kit	£15
Laser Cutting	£7.50
Total:	**£898.00**

Marketing

Because of the nature of this project, it is not the kind of thing that can be made commercially available however this is not the end of its life, throughout the project I have been talking to people from companies about the project. These include but are not limited to:

SpecialEffect; a company which creates custom controllers for people with disabilities.

The National Videogame Museum: an exhibition of videogames, numerous of which have custom controllers and entire exhibits.

EGX-LeftField Collection; a collection of experimental games which are showcased in one of the Uks largest games expo.

Be-Inspired; a company at Staffs which fund the development of startup businesses in the UK

All of these companies had useful insight into the project and gave guidance on how to take it further, but this project has already been to big events and has plans to go even further.

Ethics

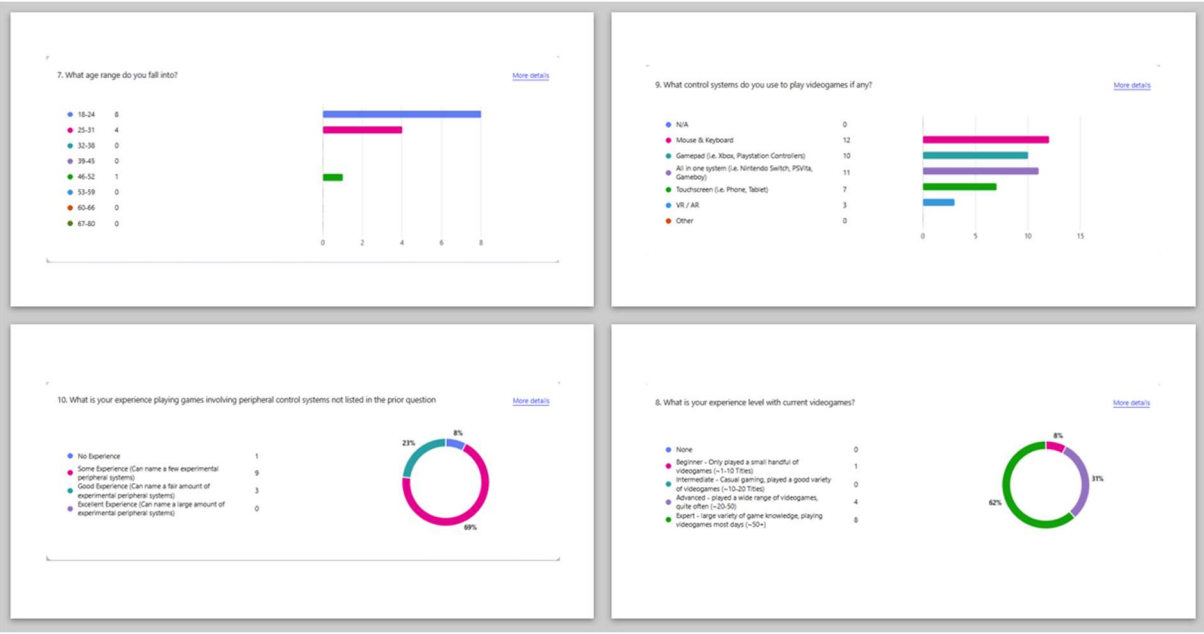
As with all research projects the ethics of such need to be considered, things like safety first and foremost, but also things like data protection. All of these have been carefully written up and presented to the ethics board and approved.

Testing and Feedback

Feedback Declaration

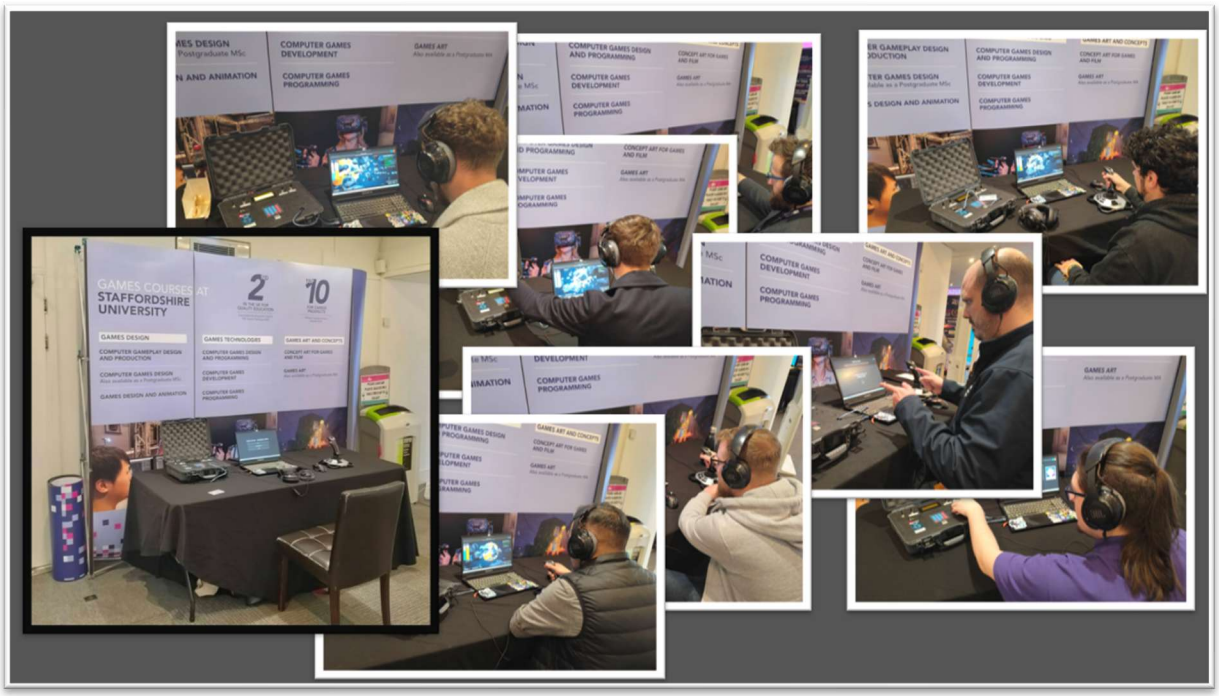
Because of the nature of the project, I wanted to make sure that the data for my project wasn't biased, as such I aimed to test with a wide area of demographics to try and remove this, I did my testing in two main areas.

The first being in university classrooms, this data was collected on Microsoft forms in controlled environments, as such this data was mainly games students.



I was quite pleased with the array of people I managed to get to test the game from this first batch, as I had a range of people who played with different control schemes, however not a lot of them had much experience with experimental system, which I feel was a benefit to the project.

The second batch of testers was at the “Let’s Talk Games Conference” hosted by the university in Leamington spa on the 11th of Feb 2025; because of the nature and busy-ness of the event it was difficult to get participants to fill in ethics forms as such this data was collected in person, by asking for feedback verbally. However, during this testing, I managed to get people from industry to test my game and give their feedback; people from companies such as SpecialEffect, Ubisoft Leamington, Lighthouse Games & SnappyGurus.

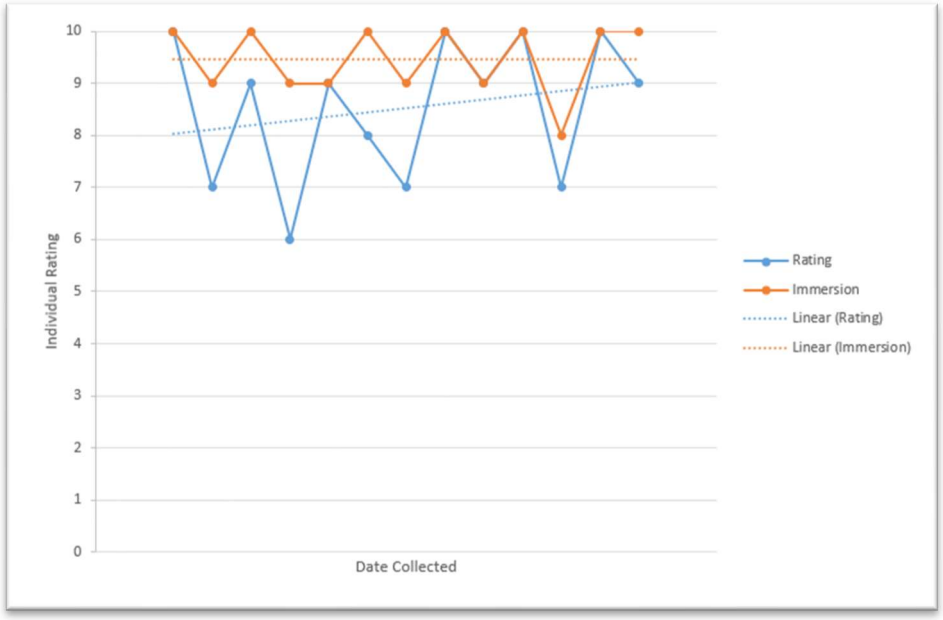


Quantitative Data

During the feedback form I asked a series of qualitative and quantitative questions so I would not only get a good gauge of the overall progress of the project over time, but also, I would get a good list of notes that I can take to further the project.

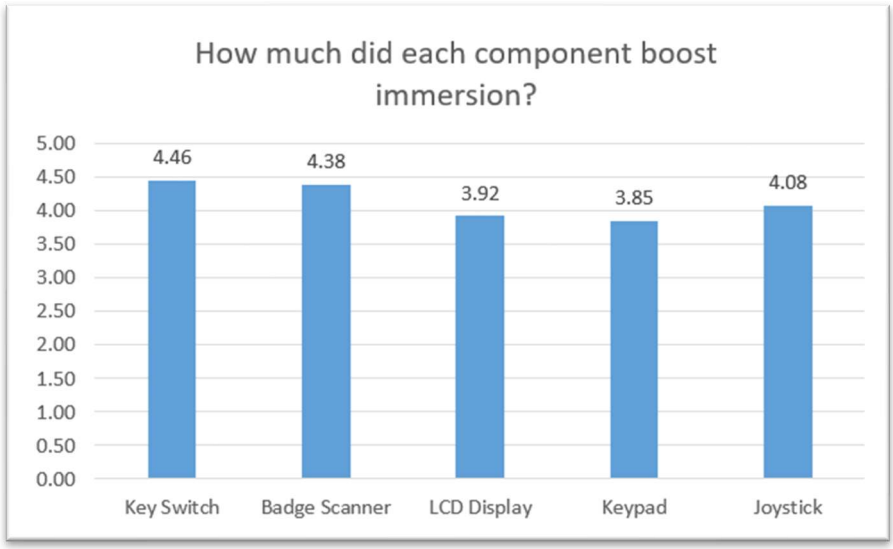
The first question I asked was rating the project on a whole from multiple different perspectives. The reason for this question is as a progress gauge, aiming for this statistic to improve as the project progresses as feedback was taken throughout the entire duration.

Ratings over Time



Above is a graph showing the data collected regarding how people rate the game on a general level as well as rating the immersion, I am quite happy with this response as on a whole the level of immersion has remained incredibly high whilst the general rating of the whole experiment has improved as time has gone on. This type of data is useful to the project as it goes to show that the changes made during development have been good and useful.

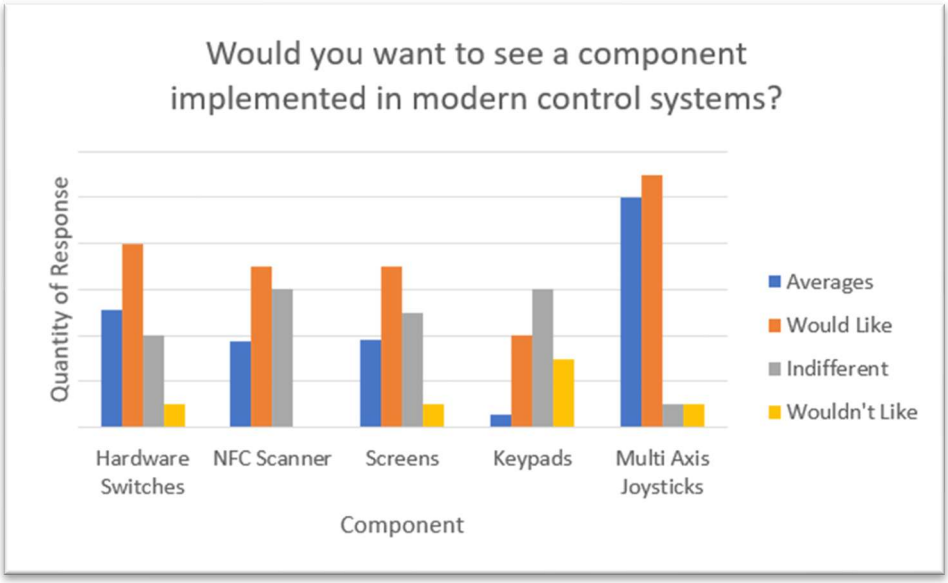
Component Immersion



The above question asked how much each component boosted immersion, and asked for a rating out of 5, I asked this question to see if I can extrapolate what it is that makes a component boost immersion. Participants tended to enjoy the key switch most, ranking it the highest, I suspect this is because not only is it a very tactile input mechanism, it is also a mechanism which is not common, so given the narrative it makes sense to be there, and looking at the trends from that data I feel this is a good assumption to make, with the badge scanner coming in second place, a component which is not commonly used, is very tactile & makes sense given the narrative.

This data is quite useful as I can see what people enjoy and in future development of the game, I can include these components in the game more and have them more involved in the game.

Implementation in modern control systems



In the form I asked how much people thought different components would fare in modern control systems, the above data shows this, the blue bars represent the average response taking into consideration those who disagree with the component being in systems, people who are indifferent and those who would like them there. As the graph shows people quite enjoyed the multi-axis joystick and want it implemented in control schemes, in this question “multi-axis joystick” refers to the joystick used in the test, a joystick which has the traditional 2D axis movement but also rotation.

This data should not be used to say which components are better in this test as this is deterministic of games currently in creation and production. As having a keypad embedded into a controller might not work for a lot of currently available games. However, hardware switches, NFC scanners and screens could be used quite well.

Qualitative Data

Bugs

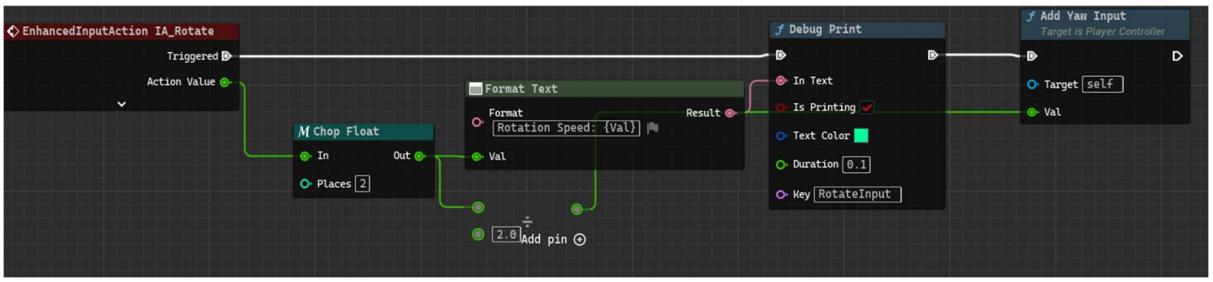
During development bugs are more than likely to occur, one method of mitigation I used was including in the form a section for finding any bugs, if the user did, I encouraged them to write them down. I asked both for gameplay and control system

ID	Name	Responses
1	anonymous	No
2	anonymous	No
3	anonymous	Unsure if this is a bug, but nothing appeared to happen when the O2 meter was depleted
4	anonymous	No
5	anonymous	No
6	anonymous	No
7	anonymous	No
8	anonymous	No
9	anonymous	Camera clipping inside the station, just have the spring arm collisioin on
10	anonymous	No
11	anonymous	No
12	anonymous	No
13	anonymous	No

ID	Name	Responses
1	anonymous	No
2	anonymous	No
3	anonymous	No
4	anonymous	Struggled to turn left as the joystick did not register a full turn
5	anonymous	No
6	anonymous	the joystick would'nt go all the way to the left.
7	anonymous	No
8	anonymous	just the hardware issue on the joystick
9	anonymous	No
10	anonymous	joystick left rotation was broken
11	anonymous	I turned off the thruster key but it didn't turn off the thrusters
12	anonymous	No
13	anonymous	No

The only gameplay issues, camera clipping and o2 were easy fixes, however the bug about the joystick's left turn was more of a problem because as far as I was aware this was an issue with the joystick itself.

Despite this I managed to figure out a solution where I reinterpreted the value and saved it, without cutting off any "overflow" values



How was gameplay altered by the controller?

19. Can you explain how you feel the game experience was altered by the use of the EPS?

13 Responses

ID	Name	Responses
1	anonymous	Made the game more interesting
2	anonymous	I think a deeper tutorial might be useful as this is a no-usual control method
3	anonymous	Originally, I thought that the whole game was going to be controlled with the joystick, so finding that the BEACON case and its different inputs was also used was a pleasant surprise. Using the EPS added a great degree of physical interaction that enhanced the digital gameplay, and learning that different actions needed to be completed with different inputs such as the NFC scanner and key switches was very enjoyable. That being said, at times it was easy to forget that the BEACON case needed to be used, which may have been due to me not taking the time to completely understand what it did. Nonetheless, the EPS definitely added a hightened sense of immersion to the game.
4	anonymous	It felt a lot more exciting navigating the ship (despite it being harder) as the joystick was very fun to maneuver around with!
5	anonymous	It allowed a "real" level of interaction that helped me better understand and guide intuition towards the right controls.
6	anonymous	I felt that the controls added a lot to the experience.
7	anonymous	It added an extra layer of physicality to playing the game
8	anonymous	It really boosts overall immersion. Controlling in a zero gravity situation is boosted with the use of the joystick, alot more intuitive to use as opposed to a keyboard. The case is also a fun element to give really nice tactile feedback to the user when interacting with the game. I like the use of the keys especially to start up systems within the game.
9	anonymous	Very very cool the usage of the box for the experience. I just wish i had the chance to press more often the switches and button etc. For example once the card is scanned to log in it's not required anymore, same for the trusters switches. Add more tiny events where it's required to turn them on and off. Same for the num pad etc.

the feedback I gained from this question was arguably one of the most useful as it wasn't posed in a negative or positive way, but it allowed people to express their thoughts.

The general thought was that people enjoyed using the controller, it added to the theme and definitely made the game feel more immersive, however people wanted more interaction with said controller, as it is not used much after the first use.

Suggestions to improve?

in this question I asked what people thought would improve the game and there was a mixed bag of results, some people asked for more affordance in the way of tutorialisation and explanation, and some said they wanted the game to be longer with more to do.

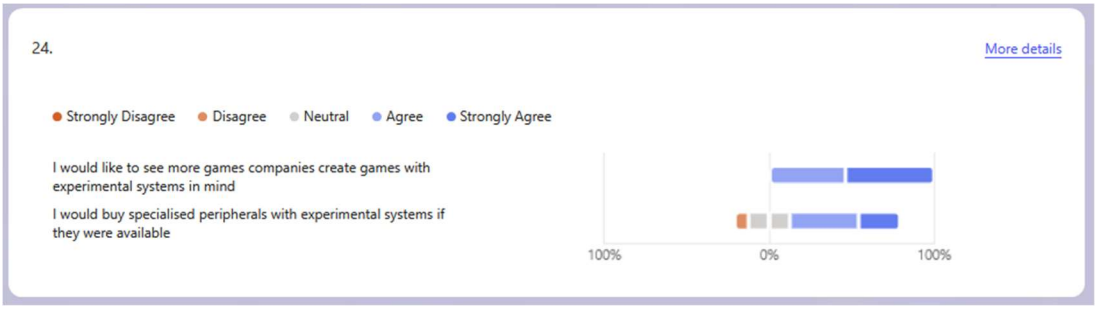
23. Do you have any suggestions to improve the game in the future

13 Responses

1	anonymous	to make it a little clearer how to use thruster mechanics
2	anonymous	I think more tutorialisation might be useful
3	anonymous	Perhaps give the player a small nudge to remember to look at the BEACON case and encourage them to take a moment to look at its inputs. The reminders do not need to be extreme, but they could be helpful so that they can figure out problems on their own.
4	anonymous	Maybe multiple keycards that need to be scanned for access to different peripherals The controls were cool but maybe a more slow / spacey feeling? (I kept hard turning!) Something more with the keys! Different notes and documents that can give the code in a unique way! More clear instructions!
5	anonymous	Maybe add an interior area, objects to avoid, maybe lasers?
6	anonymous	Make it easier to open the case.
7	anonymous	adjustments to player movement and maybe a way to remind the player of the code when they reach the door in case they forget
8	anonymous	The only thing i might add is maybe a way to interact with the keys some more. in a sense of right now you would turn them all on and leave them so its a one time use. Maybe something like a battery to manage so you cant have all the systems on all the time.
9	anonymous	more interactions\events with the briefcase. For example, have the door to require also the keycard again. Have an event that asks to turn off and on some of the switches. Because once done for the first time there are no other moments to use them again.
10	anonymous	Make the font slightly easier to read, the numbers were difficult to understand when first looking at them. More gameplay would be appreciated
11	anonymous	have more affordance for when the player is meant to be using the different controllers (beacon or control stick) and what they're meant to do with them. Also have something on the screen that says what the passcode was once the player gets it rather than having to remember.

Following this feedback I have successfully implemented tutorialisation and explanation to actions in the form of a objectives list, however expanding the game, there simply isn't time right now, as I wanted to ensure all features in are in to a high standard before expanding on them. I will expand upon this game ahead of GradEX though.

Project Feasibility in Industry



Even though this is technically quantitative, this data is ultimately the decider for the project's feasibility.

These results I'm extremely happy with as this is exactly what I wanted, people would love to see people make games using weird and experimental controllers, however they aren't as bothered about buying them individually which furthers their application for things like museums and expositions. Exactly where I want to take this project.

Conclusion

Overall, I believe this project was incredibly successful, it had a few mid project minor shifts in narrative and design however I am really happy with the end result.

Comparing this project to what is currently available in industry I believe there is a real market for this kind of thing, and whilst I don't believe that market is high scale commercial games production, I believe for things like museums, disability inclusion and interactive experiences this project and others like it have a real future.

Over the course of this project, I have learnt so much, from data formatting, how to use a special set of pliers, how ergonomics relate to immersion and professional testing standard.

I have already been accepted onto the Games Design MSc course and during it I will be forwarding this style of project.

Attributions

I would like to thank a few people for helping during this project as without their support this would never have been possible.

Adam Martin – Adam first introduced me to controller construction and since then I have received incredible amounts of support in the production of this project, and he has been vital to its success.

Daniel Williamson – Dan spent lots of time with me helping me learn electronics and how to do various non-games related tasks like using Dupont pliers and Soldering.

Mike Acosta – Mike Acosta ran the “Lets Talk Games” event where I had the opportunity to showcase my game, where I had my own table and got to speak to industry professionals about it.

Chris Headleand – Chris as the head of the games department has facilitated my development and learning whilst at staffs.

Andrew Forrester – Andy taught me how to use the laser cutting machines and worked with me in helping the physical controller look as polished as possible.

Matt Evans – Matt from the Henrion media centre sourced the new case for the project which helped it look significantly more professional

References

AdaFruit, 2024. *Github.* [Online]
Available at: https://github.com/adafruit/Adafruit-Thermal-Printer-Library
[Accessed 30 10 2024].

Balboa, M., 2025. *Github.* [Online]
Available at: https://github.com/miguelbalboa/rfid
[Accessed 1 1 2025].

SeeedStudio, 2024. *Seeed Studio Blog.* [Online]
Available at: https://www.seeedstudio.com/blog/2019/12/23/distance-sensors-types-and-selection-guide/

VideoFeedback, n.d. *GitHub.* [Online]
Available at:
https://github.com/videofeedback/Unreal_Engine_SerialCOM_Plugin
[Accessed 21 October 2024].