**Beyond Buttons: Rethinking Control**

Alex Wood – 22013321

Game Development Project



**Table of Contents**

[Research 4](#_Toc190703506)

[Components 4](#_Toc190703507)

[Ranging Sensors 4](#_Toc190703508)

[NFC 5](#_Toc190703509)

[Receipt Printer 7](#_Toc190703510)

[Displays 7](#_Toc190703511)

[Connections 7](#_Toc190703512)

[Inputs 8](#_Toc190703513)

[Unreal Development 8](#_Toc190703514)

[Plugins 8](#_Toc190703515)

[SerialCOM 8](#_Toc190703516)

[RawInput 8](#_Toc190703517)

[Arduino Interface 8](#_Toc190703518)

[GetOpenablePorts() 8](#_Toc190703519)

[SendMessageToPort() 9](#_Toc190703520)

[CheckPortsForHandshake() 9](#_Toc190703521)

[ReturnHandshake() 10](#_Toc190703522)

[Close&Clean() 10](#_Toc190703523)

[Joystick Development 10](#_Toc190703524)

[Input Mapping 11](#_Toc190703525)

[Bad Linking / Disconnecting 11](#_Toc190703526)

[Arduino Development 11](#_Toc190703527)

[Unreal Interface 11](#_Toc190703528)

[Handshake Loop 11](#_Toc190703529)

[Peripheral Interfacing 12](#_Toc190703530)

[Printer 12](#_Toc190703531)

[Physical Representation 15](#_Toc190703532)

[Version 1 15](#_Toc190703533)

[Version 2 15](#_Toc190703534)

[Narrative 15](#_Toc190703535)

[Bomb Defusal 15](#_Toc190703536)

[Astronaut Guidance 16](#_Toc190703537)

[Testing and Feedback 16](#_Toc190703538)

[Feedback Declaration 16](#_Toc190703539)

[Quantitative Data 18](#_Toc190703540)

[Ratings over Time 19](#_Toc190703541)

[Component Immersion 19](#_Toc190703542)

[Implementation in modern control systems 20](#_Toc190703543)

[Qualitative Data 21](#_Toc190703544)

[Bugs 21](#_Toc190703545)

[Ethics 21](#_Toc190703546)

[Finance 21](#_Toc190703547)

[Production Costs Over Development 21](#_Toc190703548)

[References 21](#_Toc190703549)

# Research

## Components

### Ranging Sensors

#### Ultrasonic Ranging Sensor

An ultrasonic ranging sensor is a distance measurement tool which uses sound to measure. It fires a 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)

Diagram of a sensor with different angles

Description automatically generated with medium confidence

#### Laser Distance Sensor (LiDAR)

#### Time Of Flight Sensor (ToF)

#### 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 |

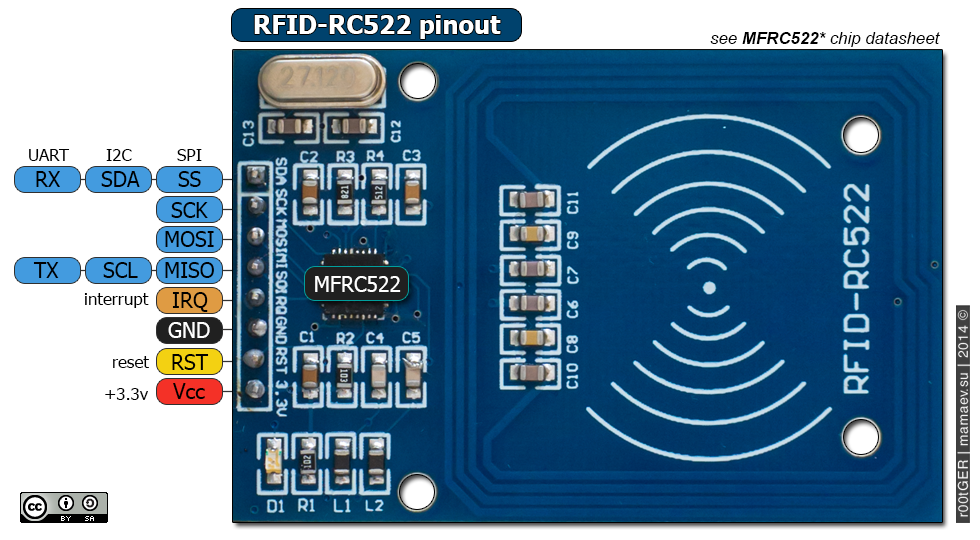
(SeeedStudio, 2024)

### 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 a 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 a 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 a 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.



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



### Receipt Printer

#### Power

#### Data Formatting

### Displays

#### LCD

#### 7 Segment Display

#### Bit Matrix’s

##### 8x8 Grid

##### Bar

### Connections

#### Dupont Pliers

### Inputs

#### Matrix Keypads

#### Potentiometers

# Unreal Development

## 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

## Arduino Interface

### GetOpenablePorts()

A screenshot of a computer

Description automatically generated

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()

A screenshot of a computer

Description automatically generated

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()

A computer screen shot of a computer program

Description automatically generated

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()

A screenshot of a computer program

Description automatically generated

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 its 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()

A screenshot of a computer

Description automatically generated

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

### Bad Linking / Disconnecting

# Arduino Development

## Unreal Interface

### Handshake Loop





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)





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



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.



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



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 cant 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

# 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 colin, 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.

# 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.

A screenshot of a graph

AI-generated content may be incorrect.

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.

A collage of a person wearing headphones

AI-generated content may be incorrect.

## 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

A graph with blue and orange lines

AI-generated content may be incorrect.

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

A graph of blue bars

AI-generated content may be incorrect.

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

A graph of different colored bars

AI-generated content may be incorrect.

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

# Ethics

# Finance

## Production Costs Over Development

|  |  |
| --- | --- |
| Item | Cost |
| Replacement Laptop | £650 |
| Arduino Mega | £18 |
| Receipt Printer | £25 |
| RC522 | £4.50 |
| Freenove Project Board | £50 |
| Dupont Crimp Connector & Kit | £30 |
| LCD-2004 i2c | £10 |
| 3D Printing | £0 |
| Briefcase | £0 |
| Nanuk 910 Case | £80 |
| Total: | £ 787.50 |

# 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].