**Beyond Buttons: Rethinking Control**

Alex Wood – 22013321

Game Development Project



**Table of Contents**

[Research 4](#_Toc182575691)

[Components 4](#_Toc182575692)

[Ranging Sensors 4](#_Toc182575693)

[NFC 6](#_Toc182575694)

[Receipt Printer 6](#_Toc182575695)

[Displays 6](#_Toc182575696)

[Unreal Development 6](#_Toc182575697)

[Plugins 6](#_Toc182575698)

[SerialCOM 6](#_Toc182575699)

[RawInput 6](#_Toc182575700)

[Arduino Interface 7](#_Toc182575701)

[GetOpenablePorts() 7](#_Toc182575702)

[SendMessageToPort() 7](#_Toc182575703)

[CheckPortsForHandshake() 8](#_Toc182575704)

[ReturnHandshake() 8](#_Toc182575705)

[Close&Clean() 9](#_Toc182575706)

[Joystick Development 9](#_Toc182575707)

[Input Mapping 9](#_Toc182575708)

[Bad Linking / Disconnecting 9](#_Toc182575709)

[Arduino Development 9](#_Toc182575710)

[Unreal Interface 9](#_Toc182575711)

[Handshake Loop 9](#_Toc182575712)

[Peripheral Interfacing 10](#_Toc182575713)

[Printer 10](#_Toc182575714)

[Briefcase / Physical Representation 14](#_Toc182575715)

[Narrative 14](#_Toc182575716)

[Bomb Defusal 14](#_Toc182575717)

[Astronaut Guidance 14](#_Toc182575718)

[Qualitative Data 14](#_Toc182575719)

[References 15](#_Toc182575720)

# 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

### Receipt Printer

#### Power

#### Data Formatting

### Displays

#### LCD

#### 7 Segment Display

#### Bit Matrix’s

##### 8x8 Grid

##### Bar

# 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.

## Briefcase / Physical Representation

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.

# 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.

# Ethics

## 

# References

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

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].