

# DIY Adaptive Gaming

Game Control for People with Upper Limb  
Disabilities

**Dominic Matthews**

A thesis presented for the degree of  
Bachelor of Creative Technology

**UNIVERSITY  
OF TWENTE.**

Supervisor: Edwin Dertien  
Critical Observer: Pep Canyelles Pèricas  
University of Twente  
Netherlands  
10/02/2024

# Abstract

People with upper-limb disabilities have limited access to gaming as a hobby, as they are unable to make use of conventional game controllers. While there are existing solutions, they have do not make use of a “menu switching” system and as such, still require a user to actuate all the inputs of a conventional controller. This is often not possible for users suffering from the aforementioned upper-limb disabilities. Additionally, these solutions often do not allow users to easily customise the inputs used, so that they can adapt them for different games.

This project is a continuation of the work of C. Omtzigt and E. Dertien, and focuses on the creation of a “menu switching” system that allows users to bind all the standard inputs of a conventional controller to two buttons and a joystick. Additionally, this includes the creation of tooling to allow users to quickly and easily rebind inputs, in order to allow them to play different games, or adapt play styles.

Based on the research, the use of an Arduino Leonardo in combination with the XInput library allows the simple creation of a spoofed, HID-compliant Xbox controller. Furthermore, it can be concluded that the use of .csv file to contain the settings for each layer of the menu systems effectively allows user to adapt the inputs as needed.

# Acknowledgements

I would like to thank both Edwin Dertien and the larger Ability Tech team of the University of Twente for their support and feedback throughout the thesis.

# Contents

<b>Abstract</b>	<b>1</b>
<b>Acknowledgments</b>	<b>2</b>
<b>List of Acronyms</b>	<b>5</b>
<b>1 Introduction</b>	<b>6</b>
1.1 Context and Relevance . . . . .	6
1.2 Research Questions . . . . .	7
1.3 Outline . . . . .	7
<b>2 Background Research</b>	<b>8</b>
2.1 Methodology . . . . .	8
2.2 Relevance of this review . . . . .	8
2.3 Modification of Existing Hardware . . . . .	8
2.4 Proprietary Solutions . . . . .	9
2.5 Conclusion . . . . .	10
<b>3 Design &amp; Implementation</b>	<b>12</b>
3.1 Design Concept . . . . .	12
3.2 Menu Layer Switching . . . . .	12
3.3 Button Remapping Tooling . . . . .	14
3.3.1 Iteration 1 . . . . .	14
3.3.2 Iteration 2 . . . . .	14
3.3.3 Iteration 3 . . . . .	15
3.3.4 Approach . . . . .	16
3.3.5 Microcontroller . . . . .	16
3.3.6 SD Card Reader . . . . .	16
3.4 Requirements . . . . .	16
3.4.1 System Pipeline . . . . .	16
3.4.2 Functional Requirements . . . . .	16
3.5 Hardware . . . . .	17
3.5.1 Microcontroller . . . . .	17
3.5.2 Other Electrical Components . . . . .	17
3.6 Software . . . . .	18
3.6.1 Arduino IDE . . . . .	18
3.7 Libraries . . . . .	18

3.8	Design Process . . . . .	18
3.8.1	XInput . . . . .	19
3.8.2	HID Library . . . . .	19
3.8.3	Adafruit NeoPixel . . . . .	20
3.8.4	SD Library . . . . .	20
3.8.5	CSV Parser . . . . .	20
3.9	Prototype Setup . . . . .	20
<b>4</b>	<b>Evaluation Method &amp; Results</b>	<b>23</b>
4.1	Methodology . . . . .	23
4.2	Validation Interview . . . . .	24
4.3	Results . . . . .	25
<b>5</b>	<b>Discussion</b>	<b>28</b>
5.1	Limitations . . . . .	29
<b>6</b>	<b>Conclusion</b>	<b>30</b>
6.0.1	Future Research . . . . .	31
	<b>APPENDIX A - Prototype Code</b>	<b>32</b>
	<b>APPENDIX B - CSV Editing Instructions</b>	<b>39</b>
	<b>APPENDIX C - Information Letter</b>	<b>40</b>
	<b>APPENDIX D - User Instructions</b>	<b>41</b>
	<b>APPENDIX E - User Test Interviews</b>	<b>42</b>
6.1	User 1 . . . . .	42
6.1.1	Pre-Test Interview . . . . .	42
6.1.2	Post-Test Interview . . . . .	42
6.2	User 2 . . . . .	43
6.2.1	Pre-Test Interview . . . . .	43
6.2.2	Post-Test Interview . . . . .	44
6.3	User 3 . . . . .	45
6.3.1	Pre-Test Interview . . . . .	45
6.3.2	Post-Test Interview . . . . .	45
	<b>APPENDIX F - User Test Interviews</b>	<b>47</b>
	<b>APPENDIX G - Arduino Leonardo Pinout</b>	<b>48</b>
	<b>Bibliography</b>	<b>50</b>

# List of Acronyms

<b>DMD</b>	Duchenne Muscular Dystrophy
<b>GP</b>	Graduation Project
<b>LED</b>	Light Emitting Diode
<b>USB</b>	Universal Serial Bus
<b>D-Pad</b>	Directional Pad
<b>IDE</b>	Integrated Development Environment
<b>HID</b>	Human Interface Device
<b>IMU</b>	Inertial Measurement Unit
<b>sEMG</b>	Surface Electromyography
<b>GADF</b>	Game Accessibility Development Framework
<b>MLS</b>	Menu Layer Switching
<b>BRT</b>	Button Remapping Tooling
<b>CSV</b>	Comma Separated Values
<b>CAD</b>	Computer Assisted Design
<b>GUI</b>	Graphical User Interface
<b>PCB</b>	Printed Circuit Board

# Chapter 1

## Introduction

People suffering from upper limb disabilities such as Duchenne Muscular Dystrophy (DMD)[1][2] are unable to participate in the hobby of gaming. They are unable to use conventional controllers for a variety of reasons, which effectively excludes them from participating in a widespread hobby. In order to enable them, as well as people in a similar situation, to pursue the hobby of gaming an adaptive controller system is needed. Once this system is fully realised, both the hardware and software components will be made available on an open-source platform.

Due to the end goal of the project, the button-mapping layout of the controller system needs to be flexible. To this end, a “menu switching” software system is required, as well as the creation of an HID compliant controller utilising an Arduino Leonardo in addition to joysticks and buttons.

### 1.1 Context and Relevance

User experience while gaming is of the utmost importance, given that gaming is, at its core, a recreational activity. The problem is that many people suffering from disabilities cannot engage in this activity, particularly when they are suffering from diseases such as DMD. This is due to the fact that they often struggle to move their hands into position to hold the controller, as well as being unable to move their fingers to comfortably hold the controller while being able to actuate all necessary buttons.

Given that game consoles require the use of specific chips within controllers to verify them, the focus of this project is gaming on PC, specifically Windows. This focus allows more flexibility, as the controller needs only be HID compliant in order to work, as well as providing a platform to edit the key configuration.

Additionally, in working on the thesis, a new partner in the development of accessible technology emerged, namely the Ability Tech team at the university. Their proposal was to also include the option for the control pad of a wheelchair to be usable as the input for the controller.

## 1.2 Research Questions

The aim of this project is to design an adaptive controller that consists of only 3 physical buttons and a joystick, but which has the functionality of a conventional game controller through a menu switching system. The button menu's must also be able to be configured by people with limited technical knowledge. Therefore the goal of this research is to answer the following main question:

**RQ:** How can a modular adaptive controller system for people with disabilities be designed and realised?

As well as the sub research questions:

**SubRQ1:** How can this system be made available online as open source hardware and software?

**SubRQ2:** Does a fixed button to switch menus work best?

## 1.3 Outline

This report will begin with a review of existing literature, as well as an evaluation of the solutions provided therein. This will be followed by an explanation of the methodological approach taken in order to answer the research questions and test the prototype. Accordingly, it will be followed by a description and evaluation of the user test results. Finally, the report will end in a conclusion which contains an evaluation of the approach taken, as well as recommendations for future research.



# Chapter 2

## Background Research

The aim of the literature review is to determine the state of the art of adaptive game controllers. The literature review consists of two equally important parts. The first part is an analysis of solutions based on the modification of existing hardware, while the second part analyses the development of proprietary solutions. Finally, the review ends with a conclusion.

### 2.1 Methodology

The presented literature was found using mainly Google Scholar, but also Science Direct. The literature was selected based on relevancy to the subject matter, as well as the research question. Keywords for the found literature include: Adaptive Controller, Gaming, Disability, Duchenne Muscular Dystrophy, Control System, Accessibility, and Custom.

### 2.2 Relevance of this review

Gaming is a widespread hobby that is enjoyed by people worldwide. However, people suffering from muscular diseases such as Duchenne Muscular Dystrophy (DMD), upper body weakness, or upper limb deficiencies have no access to this form of recreation [3]. This injustice is not perpetuated through any form of active exclusion, but rather a lack of accessibility to tools that aid inclusion. It is important not only to raise awareness of this injustice, but to effectively remedy it through the development of an adaptive game controller.

### 2.3 Modification of Existing Hardware

The creation of a proprietary adaptive game controller is an extensive and time-consuming process [4], and as such, an alternative has been found in modifying existing hardware. Iacopetti *et al.* [5] have argued that game controllers adapted for individual users are often modified commercialised controllers. Additionally, Hamadeh *et al.* from Colorado State University [6] found that existing adaptive controllers were “very expensive, and often specialised for those with no movement below the neck”. While both working with

clients suffering from muscular dystrophy, Hamadeh *et al.* [6], as well as Dertien of the University of Twente [7] took this approach. Modifying the existing hardware allowed them to make use of technology that the clients were familiar with, while also allowing ease-of-interfacing with the client’s consoles.

The physical properties of an adaptive controller are of great importance to their effectiveness. Hamadeh *et al.* [6] found that when working with users suffering from muscular dystrophy, it was vital to lower the physical resistance of the inputs as much as possible, as the atrophied hand muscles of the client would not allow them to actuate switches and buttons. Dertien [7] found that splitting a controller into two halves proved effective at allowing his client to comfortably hold the controller comfortably, and make use of the inputs on either side more effectively. An issue with the modification of existing hardware is that no such solution could be brought to market at a commercial scale, as this would violate intellectual property laws [5]. Additionally, given that each potential user has different needs, an ideal adaptive controller would need to be easily modifiable to incorporate different inputs, which would clash with the technical challenge of modifying a device that was not designed to be modified.

## 2.4 Proprietary Solutions

Proprietary solutions can be brought to market at a commercial scale, while these designs are often more widely adaptable than modified preexisting hardware. Furthermore, there is a larger diversity of proposed solutions compared to controllers made of modified existing hardware. One solution from Villar *et al.* [8] consisted of a custom fabric-adjacent PCB which allowed the simple connection of off-the-shelf sensors. It was however limited by its connection method to the computer, as well as the usage of proprietary software and emulators. Iacopetti *et al.* [5] proposed a more elegant solution consisting of a piece of middleware connected between a Playstation 2 and the controller. This solution allowed for the connection of sensors via 3.5mm jacks, and allowed for input customisation using some proprietary software. Omtzigt [4] also created a system designed around the use off-the-shelf sensors combined with middleware as an continuation of Dertien’s [7] design. All three of these approaches demonstrate that in order for a controller to be maximally customisable, and reach the largest user base, it must allow easy connection of off-the-shelf sensors. Additionally, their research indicates that a controller’s housing and button layout should be as customisable as possible.

Unconventional input methods feature heavily in proprietary controller designs, as they cater to users suffering from extensive physical disabilities. Fall *et al.* [9], Hassan *et al.* [10], and Muguro *et al.* [11] all propose the use of surface electromyography (sEMG) to read user input, as this input method relies solely on nerve signals. This makes sEMG applicable to a wide demographic of users who have little to no control of their muscles. All found sEMG to be useful in combination with other technology, such as a “Multimodal Body-Machine Interface” (BoMI) [9], Inertial Measurement Unit (IMU) [10], or an “equation estimation and a machine learning model” [11]. While undoubtedly effective, sEMG is comparatively more complex and costly to implement, as well as requiring a higher level of technical skill from the user to connect properly. Additionally, the need for sEMG to be coupled with other components hampers its effectiveness in a

highly customisable controller. The final issue with sEMG inclusive designs is that they tend to be designed for people suffering from a specific disability, which determines the secondary system that is used in combination with sEMG.

Eye and facial trackers are more niche inputs for proprietary controllers, given that they cater to a smaller subset of users. Vickers *et al.* [12] propose the use of an eye tracker for in-game locomotion, while Wang *et al.* [13] designed a system that uses both eye movement and facial expressions as inputs. While both systems proved effective, their high upfront costs, as well as the complexity of setup make them undesirable for this particular application. Further compounding this issue, eye and face trackers are more complicated to integrate into a controller than off-the-shelf sensors. Body trackers are more widely used for proprietary controllers, but are still somewhat niche. Scardovelli and Frère [14] designed a software framework that used limb position as an input for in-game actions. Similarly, Lin *et al.* [15] proposed the use of an eye tracker in combination with a data glove which translated measured position and grip strength into game inputs. Both systems proved effective at allowing users to play games without discomfort or embarrassment. However, despite the merits of both solutions, they are markedly more complicated and costly to integrate than standard sensors. Once again, the use of such inputs would require a high degree of technical knowledge from the user to implement. These issues raise the barrier of entry to these input methods to the point where they are no longer viable options. Large corporations have also begun to develop proprietary adaptive game controllers. Microsoft [16] and Sony [17] have both released more accessible controller designs, although with radically different approaches. Microsoft focuses on having inputs for every button on a normal controller, which can then be connected to a variety of proprietary sensors. Sony on the other hand developed a system which uses a joystick to select which button is currently in use. Both systems are effective for certain users, however the fact that both are expensive and only usable with their specific console is disheartening. Additionally, their limited customisability showcases that this is more about driving profit than helping people. There is not enough data on either system to truly gauge their effectiveness either. Entirely software based solutions are rare in proprietary controller design. They have been shown to be effective however, such as Vickers *et al.* [18] proposed “Game Accessibility Development Framework” (GADF). The idea of creating a library which adjusts game tasks, interaction techniques, and input device configuration is novel, but hobbled by the fact that it does not address a core issue. The users still need a game controller that they can use to interact with the game. It is clear that the software component of a controller is of vital importance, as it must be robust and adaptable enough to fit a variety of use cases. However, it is equally clear that a software only approach is ineffective.

## 2.5 Conclusion

In conclusion, a great deal of development into adaptive game controller design has already taken place, with the majority of published literature centring on the development of proprietary controllers.

Through analysis of the literature it has become clear that a readily available, cost-effective microcontroller such as an Arduino Leonardo should form the core of the design.

This microcontroller is vital not only to make the controller HID compliant, and therefore usable with PC's, but also in order to allow users to easily modify the configuration of both menus and button layouts.

Additionally, a controller consisting of at least one joystick and dedicated menu switch, and an input button is necessary. Ideally however, the controller would consist of a joystick and 3 buttons, as this would allow not only for the use of a dedicated menu switch, but also the use of macro-keys. The layout of buttons, as well as the general form factor of the controller should be left up to the user.

It has also become quite clear that the core values of the design should be ease-of-use, cost effectiveness , and adaptability. Following these core values should allow the final controller not only to be brought to market at a commercial scale, but also to find purchase in the community. The controller should prioritise the needs of the community, whilst also allowing the community to adapt it as needed. Additionally, following these values would make gaming accessible to a larger audience, and improve the user's quality-of-life. An adaptive game controller should consist of a software component programmed into a microcontroller. This software should enable the flexible use of the controller, as well as customisation of menus, buttons layouts, macro keys, and auto inputs. Through this customisability, the controller should be able to be used effectively by a wide variety of users. Furthermore, the hardware components should be 3D printable and require no tools for assembly. The files for the 3D printed parts should also be made available in an easily-edited format so that users can adjust the controller shape as needed using open source tools such as OpenSCAD or Blender.

Finally, the use of off-the-shelf sensors should not only be supported, but encouraged, with sensors having "Plug & Play" functionality. This should allow for as many people as possible to use the controller.

# Chapter 3

## Design & Implementation

### 3.1 Design Concept

The goal of this project is to build on the adaptive control system designed by C.E.S. Omtzigt for his bachelor's thesis, and by extension the work of Dr. Ir. E.C. Dertien. Omtzigt's prototype worked by sending a signal to the inputs of a torn-down PlayStation DualShock 4 controller, which then acted as an intermediary between the Arduino Leonardo and the PlayStation.

By building on this previous work, the aim is to design a standalone arduino adaptive controller which uses an XBox control scheme. The inputs used for the controller are listed in Table 1.

There are two aspects that form the core of this project, each serving a different role in aiding controller adaptability. The first is Menu Layer Switching (MLS), and the second, but no less important, is Button Remapping Tooling(BRT).

### 3.2 Menu Layer Switching

Menu Layer switching allows the user to access all 12 different inputs using only 2 buttons and a joystick. This is possible through assigning different inputs to different menu layers. The process of assigning inputs to these layers is expanded upon in Section 3.3. Once inputs have been assigned, the user needs only to actuate the menu switching button located on the front on the controller to cycle between layers. Each layer is able to support 6 different inputs: 1 for each cardinal direction of the joystick, and 1 for each button.

The reason using so few physical inputs is that many people suffering from upper limb disabilities are unable to effectively actuate more than three buttons and a joystick. Another reason is that a common set of wheelchair controls also consist of three buttons and a joystick, meaning that it would be possible for wheelchair-bound users to make use of existing and familiar inputs for the controller.

Additionally, having buttons bound to customisable layers, users are able to group commonly used inputs together, allowing them to more comfortably use the controller.













Reference Number	Icon	Name	Signal Description
1		A	Digital Signal, Button
2		B	Digital Signal, Button
3		X	Digital Signal, Button
4		Y	Digital Signal, Button
5		Left Bumper	Digital Signal, Button/Bumper
6		Right Bumper	Digital Signal, Button/Bumper
7		Menu	Digital Signal, Button
8		View	Digital Signal, Button
9		Left Trigger	Digital Signal, Trigger
10		Right Trigger	Digital Signal, Trigger
11		Left Joystick	Analogue Signal, Joystick
12		Right Joystick	Analogue Signal, Joystick

Table 3.1: List of Adaptive Controller inputs with name and signal type [19]

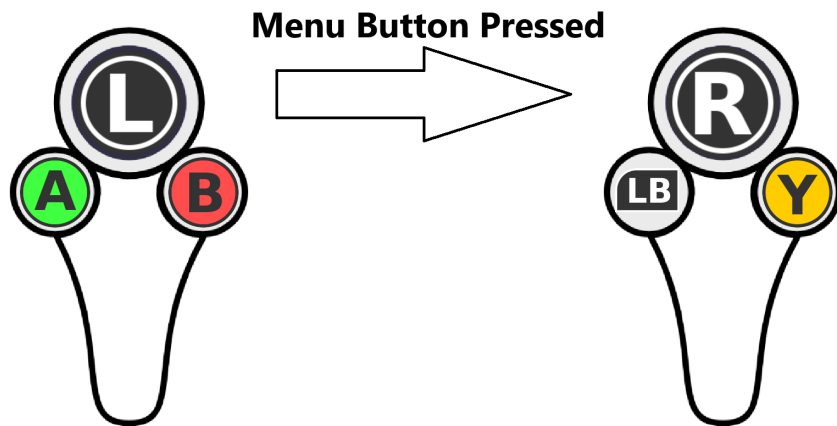


Figure 3.1: Menu Layer Switching Example

### 3.3 Button Remapping Tooling

Button remapping is vital to the functioning of the controller. As users may wish to play many different games where certain inputs are used more or less, it is important they are able to adapt the controller easily themselves. To this end some tooling was developed to make this adaptation process as simple and straightforward as possible. It was determined early on that a modifiable settings file should be stored on an SD card which the Arduino would use to determine button mappings.

#### 3.3.1 Iteration 1

In the first iteration, the settings file consisted of a simple .txt file which was structured so that users would bind inputs to specific menus. This was found to be rather clunky and unintuitive, as well as lacking any meaningful robustness. As it is imperative that users are able to intuitively understand the mapping tools, it was decided to create a different structure for the settings file.

#### 3.3.2 Iteration 2

In this iteration, a .txt was still used as the base file, but the tooling was organised horizontally rather than vertically. While it was felt that this format improved readability and intuitive understanding, it was still perceived as being too clunky. Compounding this, this version was even less robust than the last and often created parsing issues with the Arduino. With all of these problems it was determined that a different file type was needed to improve both user experience and Arduino compatibility.

```

Menu 0
Up A
Down B
Left X
Right Y
B1 L
B2 R

Menu 1
Up X
Down Y
Left A
Right B
B1 R
B2 L

```

Figure 3.2: Settings Example - Iteration 1

```

Input: Menu 0, Menu 1, Menu 2;

Up: A,B,U;
Down: B,LB,D;
Left: X,LT,L;
Right Y,RT,R;
B1: L,U,X;
B2: RB,D,LB;

```

Figure 3.3: Settings Example - Iteration 2

### 3.3.3 Iteration 3

The current, and final, iteration makes use of the .csv file type, as it is very structured, and easy to use. Users are expected to use software such as Microsoft Excel or Google Sheets to create a table of their desired mapping, and then save that mapping as a .csv on the SD card that the Arduino reads from. By using this format, it was felt that both readability and intuitive understanding improved immensely. Additionally, the CSV Parser library for Arduino allows this format to be used very robustly, as the values from the table are easily stored in arrays that are initialised anew every time the Arduino starts.

	A	B	C	D	E	F	G
1	Menu	Up	Down	Left	Right	Button 1	Button 2
2	0	A	B	X	Y	LT	RT
3	1	Up	Down	Left	Right	LB	RB
4	2	LT	RT	LB	RB	A	B

Figure 3.4: Settings Example - Iteration 3



### **3.3.4 Approach**

### **3.3.5 Microcontroller**

An Arduino Leonardo was selected as the microcontroller for this project, as they are powerful enough to run the software and act as a controller, whilst still being affordable and easily available. Additionally, in contrast to a Raspberry Pi, no specialised operating system needs to be loaded onto the microcontroller, and it can be easily programmed from the free Arduino IDE. All of these qualities make the Arduino highly accessible, and therefore ideal for the aims of this project.

### **3.3.6 SD Card Reader**

A Sparkfun MicroSD shield was selected as the SD card reader, as it doesn't need to be connected directly as a shield, but rather can have leads attached to function as a semi-detached SD card reader. This is important because the SD pins on the Arduino Leonardo aren't located where they are on other boards, but rather are in the centre of the board.

## **3.4 Requirements**

The controller should fulfil various requirements surrounding both functions, both real world and digital, as well as achieve qualitative requirements concerning the button remapping tooling.

### **3.4.1 System Pipeline**

The controller is connected via Ethernet cable to the Arduino, which parses physical inputs and matches them to outputs as determined by the settings file. From here, the Arduino then transmits these as HID-compliant output to the Windows PC it is connected to via micro-usb cable. The menu layers can be changed at any time, given that the Arduino is restarted once this has been done. Additionally, an LED ring provides visual feedback on both the menu layer as well as the joystick position. The menu layer is represented by a solid colour which lights up the whole ring, and the joystick position is displayed by a single white LED. This visual feedback is important to allow intuitive understanding of which menu layer the user is on, to show when the menu has successfully been switched, and to show which input should be actuated via the joystick.

### **3.4.2 Functional Requirements**

The controller must be able to:

- Take joystick or button input from the user and accurately reflect this in the digital domain.
- Read menu layer mapping from an SD card and use this to configure the controller's outputs.

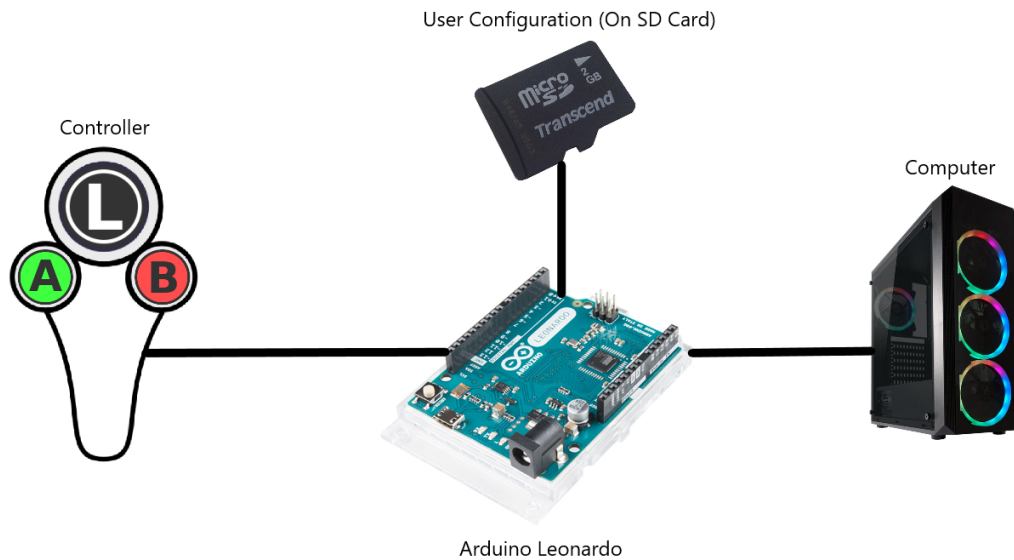


Figure 3.5: Adaptive Controller Pipeline [20][21][22]

- Be modular, so that users may implement most off-the-shelf sensors to use as inputs, and quickly replace parts when necessary.
- Make use of a configuration file that is both easy to change and intuitive to understand.
- Display the menu layer and joystick position using an LED ring.

## 3.5 Hardware

### 3.5.1 Microcontroller

An Arduino Leonardo.

### 3.5.2 Other Electrical Components

#### Computer

A computer with at least 1 USB 2.0+ port.

OS: Windows 11. Build: 22621.3007 Monitor

A monitor with either an HDMI or DisplayPort slot to connect to the PC.

#### Basic Electronic Components

Appropriate wires

Breadboard

3 Tactile pushbutton switches

1 Joystick  
USB-A to micro-USB cable  
LED ring

## 3.6 Software

### 3.6.1 Arduino IDE

Version: 2.2.1

The open-source Arduino IDE is used to upload the source code to the Arduino Leonardo, as well as make any adjustments to the code the user would like. It is lightweight and simple to use. It is advised to create a *C:/ Drive* installation of the IDE with a short PATH. Other than that it is advised to simply follow Arduino's own setup guide.

## 3.7 Libraries

1. CSV Parser by Michal Borowski, Version: 1.2.2[23]

This library is instrumental for allowing the simple use of .csv files for the purpose of configuration. The library has plenty of clear, high quality documentation, and is easy to use. This library was specifically used to parse the settings file into arrays, which would then be used to inputs and determine outputs.

2. XInput by David Madison, Version: 1.2.6[24]

This library is able to emulate an Xbox controller over controller, with the caveat that it can only run on AVR boards such as the Arduino Leonardo, Sparkfun AVR series, or Teensy boards.

3. Adafruit NeoPixel by Adafruit, Version: 1.12[25]

A library that allows easy control of Adafruit LED devices, specifically used in this case to control the LED ring.

4. SD Library by Arduino, Version 1.2.4[26]

A simple Arduino library that allows the reading of SD cards.

## 3.8 Design Process

The goal of the project is to design a HID compliant adaptive game controller, which reads user settings from an SD card. The major challenge is that this project is somewhat unique in its aims, and takes an unconventional approach to solving the problem of adaptive game control.

Therefore the first step was to investigate what existing technology is usable for this application.

### 3.8.1 XInput

XInput is able to effectively emulate an Xbox controller using the Arduino. However, due to the limited storage capacity of the Arduino Leonardo as opposed to, for example, and Arduino Due, it is important to keep the code as concise and clean as possible. As can be seen by the *joy.cpl* output in Figure 3.6 below, XInput emulates all inputs of a standard controller, including the D-Pad. It is simple to implement and use, and works well with all the other libraries of the project. Another benefit of this library over MHiernomous' Joystick Library is that it uses the actual input names, as opposed to the numerical values assigned to inputs by Windows. This makes it more easily understandable.

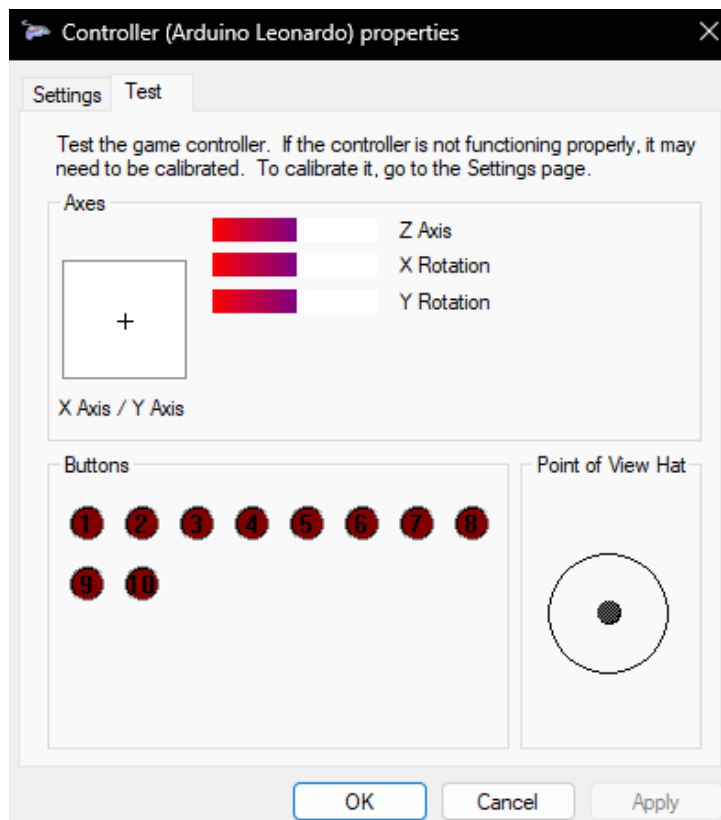


Figure 3.6: *joy.cpl* Output of the Controller

### 3.8.2 HID Library

Arduino's HID library was the easiest to implement and use. After including it in the sketch, all that was required was to allow the Arduino to automatically transfer HID data using the AutoSend function. Once this was done, Windows recognised the Arduino as a game controller, and testing could be carried out with the *joy.cpl* program, as seen in Figure 3.6.

### 3.8.3 Adafruit NeoPixel

Once the controller had been successfully set up, the Adafruit library was used to control the LED ring.

First, the joystick x and y information was calculated into degrees in order to be able to display joystick direction. Then it was a simple case of allocating each menu layer a colour and using the library to control when the LEDs assumed these colours.

### 3.8.4 SD Library

Another simple library to use, the SD library along with the test code by D.A. Mellis allowed for SD reading implementation. It was only a case of creating a checker to make sure the SD card was actually read and throwing an error if not in order to make sure the users have usable feedback.

### 3.8.5 CSV Parser

Finally, it came time to actually make use of the settings file for control customisation. This took a great deal of trial and error, not because of a lack of documentation or library functionality, but because the controller was not a use case the CSV parser was really designed for. It became clear after a while that the values parsed by the CSV library had to be stored in individual arrays determined by input type in order to optimise how the program ran. This was because Arduino does not allow on the calling of variables from other methods, and the CSV parser relies on local variables in its method to function properly.

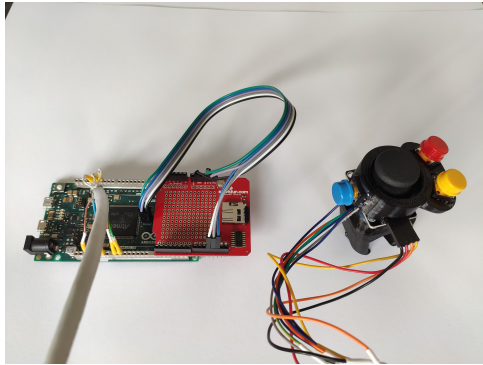
## 3.9 Prototype Setup

The controller will be preliminarily tested using *joy.cpl* to make sure that all inputs function as intended, whereupon it will be tested by users playing a level of the game Neon Abyss. The Hardware used is depicted in the following table:

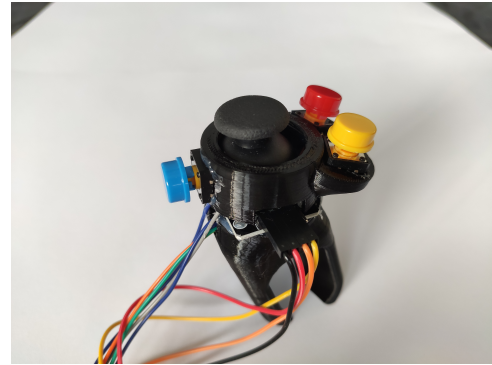
Component	Count	Image
Arduino Leonardo	1	
SD Card Shield	1	
Tactile Push Button Switch	3	
Female-Male Jumper Wires	6	
Male-Male Jumper Wires	14	
Breadboard (64 rows)	1	
Potentiometer Joystick	2	
330 Ohm Resistor	1	
LED Ring	1	
USB-A to Micro USB Cable (Data Transfer Capable)	1	

Table 3.2: List of Prototype Components

The resulting prototype can be seen below in Figure 3.7. Additionally, the wiring diagram can be seen in Figure 3.8. The preliminary code for this prototype can be found in Appendix A. To determine if the controller was working correctly, it was checked using *joy.cpl*.



(a) Prototype Overview



(b) Controller Closeup

Figure 3.7: Adaptive Game Controller

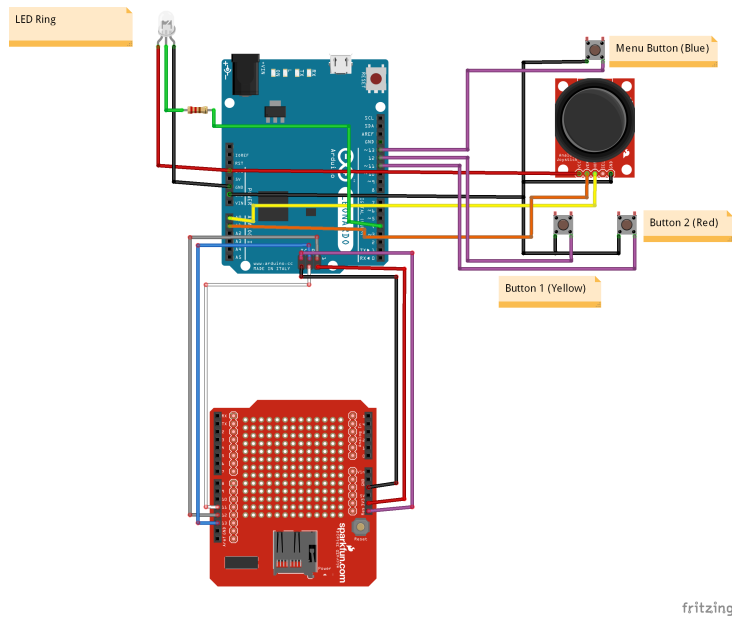


Figure 3.8: Prototype Wiring Diagram

# Chapter 4

## Evaluation Method & Results

### 4.1 Methodology

The aim of this graduation project (GP) is to create an adaptive game controller. Specifically, to create an adaptive game controller that is able to emulate a conventional controller with a greatly reduced number of inputs. To this end a controller, as well as a “menu” switching system with integrated tooling was designed and realised. This is the continuation of a graduation project by Cedric Omtzigt titled “Developing a modular gaming handheld for gamers with muscular dystrophy” [4]. As this GP is in the prototyping phase, the testing for the adaptive controller is user-centred.

The goal of this testing is to establish the on-boarding, set-up, and fluency times, as well as the efficacy of the tooling language and instruction materials. The instruction materials can be seen in Appendix D - User Instructions.

The user testing were conducted with a group of three able bodied users. Due to various reasons, it was not feasible to test with a user suffering from upper-body disabilities. These tests focused on the collection of qualitative data gathered from observation, as well as quantitative data from measuring level progress, on-boarding, set-up, and fluency times.

The on-boarding time was defined as the time it took the users to get from entering the level to leaving the first room, as the first room is always empty of enemies and traps. This allowed them to get comfortable and experiment with the controls. The set-up time was defined as the time it took the users to modify the configuration file. The fluency time was defined as the time it took the users to clear a combat room without taking damage.

The testing was conducted in the living area of a student house in order to do testing in situ. The exact setup can be seen below in Figure 4.1. The users were selected due to their differing levels of experience with video games. None of the users had played the game before, and only one user was familiar with the game used for testing. The tests were conducted during the afternoon on the weekend in order to ensure that the users were not overly tired or drained from a workday.



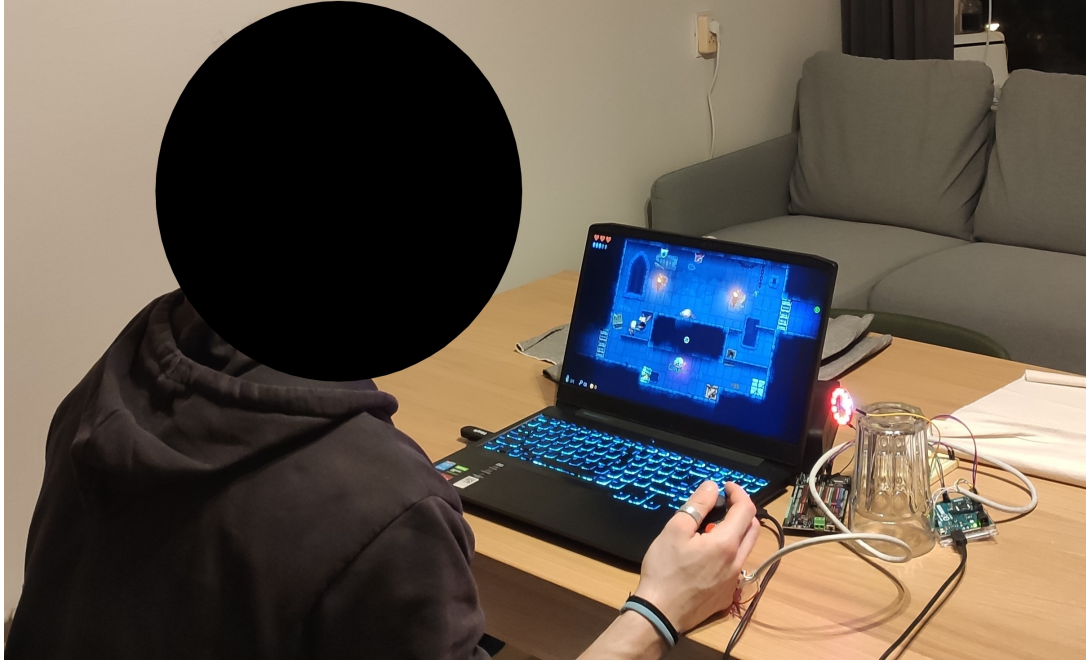


Figure 4.1: Photo of the User Testing Setup

An issue with this methodology is that the test group consists of users who fall outside of the target demographic. While this is not preferable, it is believed that the data provided from these tests can still provide valuable insights into the efficacy of the controller.

Each user only had 1 iteration with the menu remapping due to time constraints. This means that there was no data to evaluate regarding how users adapted their menu mappings over time.

## 4.2 Validation Interview

In order to gauge the effectiveness of the controller, a group of users familiar with video games was asked to play a single level of the game *Neon Abyss* before modifying the control scheme to their liking and playing the level again. Users played the game “*Neon Abyss*”, as it has a non-trivial amount of inputs, and being a roguelite game, has many high-intensity situations. Thus, it was selected as a good “stress test” for the system. There were interviews both before and after the tests, with the former serving to establish gaming experience, comfort with controllers, and comfort with control customisation. The latter focused on gathering qualitative data regarding the user’s experience with the controller as well as the Button Mapping Tooling. A short guide for the tooling can be found in Appendix B, and was given to each user before they make use of the remapping functionality.

User	Experience Level	1 <sup>st</sup> Attempt Progress	Remapping Time (in Minutes)	2 <sup>nd</sup> Attempt Progress
1	Medium	~50%	3	Final Boss (~95%)
2	High	100%	2	100%
3	High	Final Boss (~95%)	3	100%

Table 4.1: User Testing Results

User	On-Boarding Time	Set-Up Time	Fluency Time
1	2.5	4	10 (3 Rooms on the 2 <sup>nd</sup> Attempt)
2	1	2	2 (2 Rooms)
3	2	3	5 (6 Rooms)

Table 4.2: Quantitative Data from User Testing (All Times in Minutes)

### 4.3 Results

The user testing began with a short explanation of the project, covering the premise of the research, as well as the functions of the controller.

Following this, the users were shown a table showing which inputs were bound to each layer. After familiarising themselves with the controls, the users were asked to start the game, and test out the controls in the first room. By allowing the user to familiarise themselves with the controls in this way, two things were achieved: the users were able to understand the controls in a low pressure, and it allowed the on-boarding time to be easily established.

The users were encouraged to move on through the level once they felt sufficiently ready. From there on out they were given no more guidance and allowed to proceed through the level at their own pace, with the goal of beating the boss at the end of the level.

Once the users had either beaten the boss or lost the game, their progress was recorded. They were then given the instruction manual (as depicted in Appendix D), and asked to reconfigure the menu layers to their liking. An example of a user changing the configuration can be seen below in Figure 4.2. The time it took them to configure the file from opening it to saving it as a .csv was recorded as the set-up time. The users were then asked to once again attempt the first level on the game.

As can be seen in Table 4.1, all three users were able to change the mapping of the menus within 3 minutes. As an aide to ease cognitive load they were allowed to use a “cheat sheet” which contained simple instructions and a table which showed what CSV codes corresponded to which inputs. Additionally, all users reported an improved gameplay experience once they had remapped the controller to their liking. This improved gameplay experience also strongly correlated to an improvement in game progress.

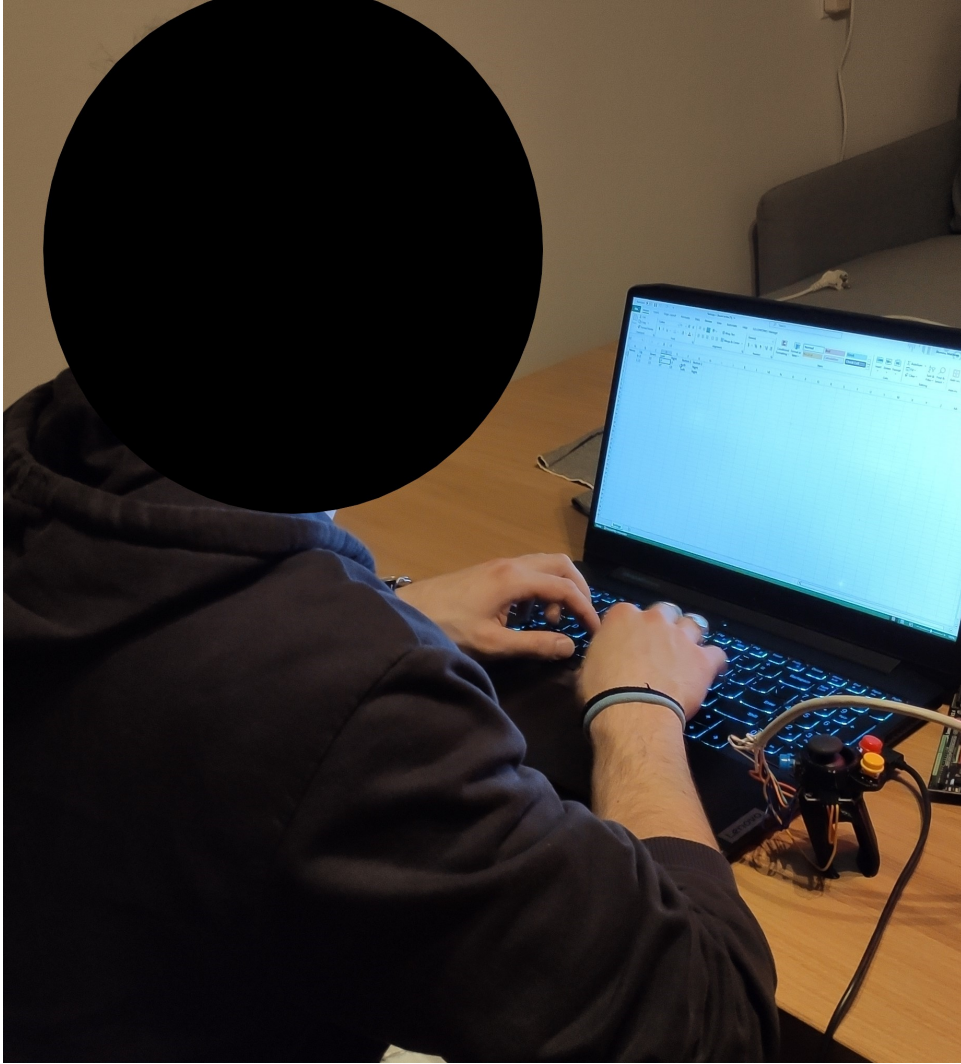


Figure 4.2: A User Changes the Menu Configuration

User testing results be seen in Tables 4.1 and 4.2. As previously mentioned, users were only allowed a single iteration of remapping due to time constraints. Additionally, the base menu configuration can be seen in Table 4.3, while the changes made by users can be seen in Tables 4.4, 4.5, and 4.6.

All users were able to make remap the menus within 3 minutes, and become accustomed to the controller within 4 minutes, as can be seen above.

User 1 showed the largest improvement after remapping, but User 3 also improved.

Menu	Up	Down	Left	Right	Button 1	Button 2
0	Up	Down	Left	Right	A	B
1	Aim Up	Aim Down	Aim Left	Aim Right	LR	RT
2	Up	Down	Left	Right	Menu	View

Table 4.3: Default CSV Configuration

Menu	Up	Down	Left	Right	Button 1	Button 2
0	Up	Down	Left	Right	A	B
1	Aim Up	Aim Down	Aim Left	Aim Right	LR	RT
2	Aim Up	Aim Down	Aim Left	Aim Right	A	View

Table 4.4: User 1 CSV Configuration

Menu	Up	Down	Left	Right	Button 1	Button 2
0	Up	Down	Left	Right	A	B
1	Aim Up	Aim Down	Aim Left	Aim Right	LR	RT

Table 4.5: User 2 CSV Configuration

Menu	Up	Down	Left	Right	Button 1	Button 2
0	Up	Down	Left	Right	A	B
1	Aim Up	Aim Down	Aim Left	Aim Right	LR	RT
2	Up	Down	Left	Right	X	RT

Table 4.6: User 3 CSV Configuration

# Chapter 5

## Discussion

In this discussion, the results as well as the interviews will be analysed in order to draw conclusions which will be used to answer the research questions posed at the beginning of this GP.

From the post-test interviews it quickly became clear that the users initially struggled to make use of the controller due to their familiarity with traditional controllers. Thus, at first they had challenges grasping the menu switching system, as well as dealing with the limited amount of inputs at their disposal. After a few minutes of playtime however, they began to understand the system and became more comfortable with switching between menus.

It is notable that while the users adapted quickly, they noted that they found the menu switching system initially challenging, as although they were made aware of the inputs on each menu layer, it wasn't intuitive for each of them. It has become clear that the limited number of inputs can prove challenging in high intensity situations, as users attempting to perform complex actions or multiple actions in rapid succession have to keep track of the game and the menu layer at the same time.

Another suggestion by the users was to have the menus in the .csv file be coloured the same as the LEDs for that menu layer. This would aid intuitive understanding of the menu remapping tool. The challenge with this however, lies within the functionalities of the .csv file type itself, as .csv files are not capable of saving or displaying colour information, as the file only contains characters.

It was noted that during stressful sections of the game, users had a tendency to lose their position in the menu layers, or need to actuate inputs on different layers rapidly, resulting in "panic switching". During this state of "panic switching" users would switch so rapidly between menus that they would get even more lost, and as a result actuate the wrong inputs. It is possible that this phenomenon could be avoided if a user is sufficiently used to the system and the game, however testing that lies outside of the scope of this project.

As can be seen in the Tables from Section 4.3, the users didn't make sweeping changes to the menu layers, preferring to switch around a few inputs rather than enact sweeping changes. It is notable however, that they all mentioned how much more comfortable they

were with the system once they had tweaked it to their liking. Notably, User 2 chose to remove the third menu layer, as seen in Table 4.5. This allowed them to quickly go from shooting to moving and vice versa.

As can be seen in Table 4.1, there was no room for User 2 to improve in terms of level progress, but they still reported better performance after modifying the menu layer to their liking.

User 1 had the most difficulty, which might have to do with their lack of recent experience playing video games. They did, however, improve remarkably after remapping the inputs, and did so in a similar time to Users 1 and 2.

It is additionally notable that Users 1 and 3 took longer to achieve fluency than User 2. Upon observation, this could be attributed to the fact that they were not able to move and shoot at the same time. This technique is ubiquitous when playing on a normal controller, and the inability to use it often left the Users in vulnerable positions. They had to learn to pace themselves and prioritise being in a safe position before shooting. This was not always possible however, as some rooms require the User to dodge incoming attack whilst simultaneously retaliating.

## 5.1 Limitations

A limitation of the prototype is the lack of integration for less standard input types such as eye trackers. While these should be implementable given the versatility, this once again lies outside of the scope of this project.

A further limitation, albeit minimal, is the case being 3D printed, as the User will need access to 3D printing facilities in order to create the casing, as well as an understanding of 3D or computer assisted design (CAD) software if they wish to modify it. In order to mitigate this it would be ideal if users had access to university facilities or maker spaces, as well as experienced helpers in either location.

Continuing on, there is a limitation in the Arduino itself, as only boards with AVR chips are usable. As mentioned previously, boards without this chip will not be able to make use of the XInput library, rendering the controller non-functional.

A final limitation is that this prototype cannot be used with consoles such as the Nintendo Switch, Xbox Series X, or PlayStation 5, as the Arduino lacks the necessary proprietary chip which the consoles require for controller verification.

# Chapter 6

## Conclusion

To conclude, the design and realisation of an adaptive game controller is a time intensive process, especially when many revisions are required in order to allow for easier use. While a working prototype was created, it has been shown that the system still requires development in order to fulfil its potential.

Next, the main research question will be answered: “*How can a modular adaptive controller system for people with disabilities be designed and realised?*” As well as the sub research questions: “*How can this system be made available online as open source hardware and software?*” And: “*Does a fixed button to switch menus work best?*”

RQ: *How can a modular adaptive controller system for people with disabilities be designed and realised?*

In order for the controller to be effective, all three buttons as well as the joystick must be easily usable with only the index finger and thumb. Thus, the controller should have two buttons situated behind the joystick and one button in front of the joystick. Additionally, as there is only a limited number of inputs available, the system needs to be able to switch layers in order to allow the actuation of all standard inputs. This requires the development of a layer switching software within the interpreter that connects the controller to the PC, as well as a tool to assign the standard inputs to different layers. Furthermore, the system needs to make use of easily accessible, off-the-shelf components and a cheap, 3D-printed casing. The controller itself must also be modular in order to allow the connection of various inputs.

Sub-RQ1: *How can this system be made available online as open source hardware and software?*

The system’s software can be uploaded to a GitHub page in order to allow easy open-source development and forking branches. The files for the 3D printed case can also be published on the same GitHub in order to allow users to modify the case files and create new designs, as well as print their own cases.

Sub-RQ2: *Does a fixed button to switch menus work best?*

A fixed button works best, as in a high stress scenario, “panic switching” would be exacerbated by the menu button not being a set input. This would lead to a degradation

of the user experience, and increase the mental workload of using the controller, further hindering its implementation and use.

### **6.0.1 Future Research**

In the future, it is important that new input types are integrated into the system in order to allow more people to make use of the controller. Such input types could be eye trackers, inertial measurement units (IMU), or surface electromyography (sEMG). The implementation of these inputs should be fairly straightforward as they are fundamentally compatible with the Arduino.

There should be investigation into either a graphical user interface (GUI) or changing the settings file format in order to allow users to more easily use the menu switching tool. Such a GUI could take the form of an app or web-app, depending on the further development of the controller.

It would also be of interest to find a way to spoof the authentication chips of console controllers in order to allow the system to work with consoles. This would allow a great deal more people to make use of the adaptive controller, as well as allowing access to a greater variety of games.

Furthermore, in the interest of increasing accessibility, it would be interesting for a researcher to get into contact with Arduino in order to develop a PCB, shield, or even a whole kit that allows for users to more easily realise their own controllers.



# APPENDIX A - Prototype Code

```
1 //Project: Bachelor's Thesis BSc Creative Technology: "DIY Adaptive
  Gaming"
2 //Written by Dominic Matthews 01/01/2024
3 //Adapted from Beginners Guide by Daniel Cantore
4 //Also adapted from SimulateAll example of the XInput Library by David
  Madison
5
6 //Library Inclusion
7 #include <SD.h>
8 #include <LiquidCrystal.h>
9 #include <Adafruit_NeoPixel.h>
10 #include <CSV_Parser.h>
11 #include <XInput.h>
12
13 //Pin Definitions
14 #define PIN_SD 4
15 #define PIN_NEO_PIXEL 3 //Data pin for the LED ring
16 #define NUM_PIXELS 12 //Defining the number of LEDs in the ring
17 #define BUTTON_1 13 //The red button by default
18 #define BUTTON_Left 12 //The blue button by default
19 #define BUTTON_Right 11 //The yellow button by default
20 #define JOYSTICK_X A0 //Joystick X-axis pin
21 #define JOYSTICK_Y A1 //Joystick Y-axis pin
22 #define menu_count 3 //Defining the number of menus used
23
24 //Integer Initialisation
25 int x_axis = 0;
26 int y_axis = 0;
27 int menu_delay = 50;
28 int unif_delay = 50;
29 float joystick_angle;
30 //Arrays to store settings values
31 String up_values[menu_count];
32 String down_values[menu_count];
33 String left_values[menu_count];
34 String right_values[menu_count];
35 String button_1_values[menu_count];
36 String button_2_values[menu_count];
37 //Xinput Setup
38 const boolean UseLeftJoystick = true; //enables the left joystick
39 const boolean UseRightJoystick = true; //enables the right joystick
40 const boolean UseTriggerButtons = true; //sets the triggers as digital
  buttons
41 //Menu integer to track current menu
```

```

42 int menu = 0;
43 //Settings File Initialisation
44 File settings;
45 String settings_string;
46 const char *settings_chars;
47
48 //Initialise joystick maximum value
49 const int joystick_max = 32767;
50
51 //Initialising LED Ring
52 Adafruit_NeoPixel led_ring(NUM_PIXELS, PIN_NEO_PIXEL, NEO_GRB +
    NEO_KHZ800);
53
54 void setup() {
55     //Begin Serial Communication
56     Serial.begin(9600);
57     //Setting up controller config from the "options" file on the SD card
58     sd_setup();
59     //Parse the Settings CSV file
60     csv_parsing();
61     //Initialise Buttons
62     //Buttons set up between Digital Pin and Ground
63     pinMode(BUTTON_1, INPUT_PULLUP);
64     pinMode(BUTTON_Left, INPUT_PULLUP);
65     pinMode(BUTTON_Right, INPUT_PULLUP);
66     //Controller Initialisation
67     XInput.setAutoSend(false); //Wait for input before sending
68     XInput.begin();
69     //Adjust LED Brightness
70     led_ring.setBrightness(12);
71 }
72
73 void loop() {
74     joystick();
75     inputs();
76     button_state_reset();
77     led_control();
78     menu_switch();
79     //Pole Delay/Debounce
80     delay(unif_delay);
81 }
82
83 void joystick() {
84     //Initialise joystick axis values by reading from joystick pins
85     int joystick_x = analogRead(JOYSTICK_X);
86     int joystick_y = analogRead(JOYSTICK_Y);
87
88     //Determine joystick angle using algebra
89     joystick_angle = atan2(joystick_y, joystick_x) * (180 / M_PI) + 180;
90 }
91
92 void inputs() {
93     //Check if joystick is tilted up and then use button_check method
    using saved settings inputs
94     if (analogRead(JOYSTICK_Y) <= 250) {

```

```

95     button_check(up_values[menu]);
96 }
97 //Check if joystick is tilted down and then use button_check method
    using saved settings inputs
98 if (analogRead(JOYSTICK_Y) >= 750) {
99     button_check(down_values[menu]);
100 }
101 //Check if joystick is tilted left and then use button_check method
    using saved settings inputs
102 if (analogRead(JOYSTICK_X) <= 250) {
103     button_check(left_values[menu]);
104 }
105 //Check if joystick is tilted right and then use button_check method
    using saved settings inputs
106 if (analogRead(JOYSTICK_X) >= 750) {
107     button_check(right_values[menu]);
108 }
109 //Check if the left button was actuated and button_check method using
    saved settings inputs
110 if (digitalRead(BUTTON_Left) == LOW) {
111     button_check(button_1_values[menu]);
112     Serial.print("Left");
113 }
114 //Check if the left button was actuated and button_check method using
    saved settings inputs
115 if (digitalRead(BUTTON_Right) == LOW) {
116     button_check(button_2_values[menu]);
117     Serial.print("Right");
118 }
119 }
120
121 void led_control() {
122     //Determine LED by calculating joystick angle/(360/number of LEDs)
123     int led_number = joystick_angle / 30;
124     //Setting LED colour according to menu
125     for (int i = 0; i < NUM_PIXELS; i++) {
126         if (menu == 0) {
127             led_ring.setPixelColor(i, led_ring.Color(255, 0, 0) //Sets
the LEDs to red on menu 0
128         } else if (menu == 1) {
129             led_ring.setPixelColor(i, led_ring.Color(0, 255, 0)); //Sets the
LEDs to green on menu 1
130         } else if (menu == 2) {
131             led_ring.setPixelColor(i, led_ring.Color(0, 0, 255)); //Sets the
LEDs to blue on menu 2
132         } else if (menu == 3) {
133             led_ring.setPixelColor(i, led_ring.Color(0, 255, 255)); //Sets
the LEDs to turquoise on menu 3
134         } else if (menu == 4) {
135             led_ring.setPixelColor(i, led_ring.Color(255, 255, 0)); //Sets
the LEDs to yellow on menu 4
136         }
137     }
138
139     //Setting position of the direction indicator LED using the joystick

```

```

    angle
140  if (led_number >= 12) {
141      led_number = 0;
142  }
143  led_ring.setPixelColor(led_number, led_ring.Color(255, 255, 255));
    //Set direction indicator LED to white regardless of menu
144
145  //Display the changes
146  led_ring.show();
147 }
148
149 void menu_switch() {
150     //Reading the Menu switching buttons
151     //Much simpler than other buttons, as it is not configurable and is
        always ready to be read
152     //Reading the current Button digital pin to the Current Button State
        for processing
153     //Checking for menu switching
154     if (digitalRead(BUTTON_1) == LOW) {
155         //Increment menu by 1
156         menu += 1;
157         //Overflow clause
158         if (menu > menu_count - 1) {
159             menu = 0;
160         }
161         //Delay to prevent over-switching
162         delay(menu_delay);
163     }
164 }
165
166 void sd_setup() {
167     //Adapted from SD_Test example code by David A. Mellis
168     Serial.print("Initializing SD card...");
169
170     //If SD can't be initialised, throw error
171     if (!SD.begin(4)) {
172         Serial.println("initialization failed!");
173         while (1)
174             ;
175     }
176     Serial.println("initialization done.");
177
178     //Reading settings file and storing values in strings
179     settings = SD.open("Settings.csv", FILE_READ);
180     settings_string = settings.readString();
181     //Convert settings_string to C String, as the CSV parser needs a C
        String
182     settings_chars = settings_string.c_str();
183 }
184
185 void button_check(String temp_string) {
186     if (temp_string.equals("A")) {
187         //Actuate A
188         XInput.press(BUTTON_A);
189         Serial.println("A");

```

```

190 } else if (temp_string.equals("B")) {
191     //Actuate B
192     XInput.press(BUTTON_B);
193     Serial.println("B");
194 } else if (temp_string.equals("X")) {
195     //Actuate X
196     XInput.press(BUTTON_X);
197     Serial.println("X");
198 } else if (temp_string.equals("Y")) {
199     //Actuate Y
200     XInput.press(BUTTON_Y);
201     Serial.println("Y");
202 } else if (temp_string.equals("Up")) {
203     //Set left joystick to up
204     XInput.setJoystickY(JOY_LEFT, joystick_max);
205     XInput.setJoystickX(JOY_LEFT, 0);
206     Serial.println("Up");
207 } else if (temp_string.equals("Down")) {
208     //Set left joystick to down
209     XInput.setJoystickY(JOY_LEFT, -joystick_max);
210     XInput.setJoystickX(JOY_LEFT, 0);
211     Serial.println("Down");
212 } else if (temp_string.equals("Left")) {
213     //Set left joystick to left
214     XInput.setJoystickX(JOY_LEFT, joystick_max);
215     XInput.setJoystickY(JOY_LEFT, 0);
216     Serial.println("Left");
217 } else if (temp_string.equals("Right")) {
218     //Set left joystick to right
219     XInput.setJoystickX(JOY_LEFT, -joystick_max);
220     XInput.setJoystickY(JOY_LEFT, 0);
221     Serial.println("Right");
222 } else if (temp_string.equals("LT")) {
223     //Actuate LT
224     XInput.press(TRIGGER_LEFT);
225     Serial.println("LT");
226 } else if (temp_string.equals("RT")) {
227     //Actuate RT
228     XInput.press(TRIGGER_RIGHT);
229     Serial.println("RT");
230 } else if (temp_string.equals("LB")) {
231     //Actuate LB
232     XInput.press(BUTTON_LB);
233     Serial.println("LB");
234 } else if (temp_string.equals("RB")) {
235     //Actuate RB
236     XInput.press(BUTTON_RB);
237     Serial.println("RB");
238 } else if (temp_string.equals("Aim Up")) {
239     //Set right joystick to up
240     XInput.setJoystickY(JOY_RIGHT, joystick_max);
241     XInput.setJoystickX(JOY_RIGHT, 0);
242     Serial.println("Aim Up");
243 } else if (temp_string.equals("Aim Down")) {
244     //Set right joystick to down

```

```

245     XInput.setJoystickY(JOY_RIGHT, -joystick_max);
246     XInput.setJoystickX(JOY_RIGHT, 0);
247     Serial.println("Aim Down");
248 } else if (temp_string.equals("Aim Left")) {
249     //Set right joystick to left
250     XInput.setJoystickX(JOY_RIGHT, joystick_max);
251     XInput.setJoystickY(JOY_RIGHT, 0);
252     Serial.println("Aim Left");
253 } else if (temp_string.equals("Aim Right")) {
254     //Set right joystick to right
255     XInput.setJoystickX(JOY_RIGHT, -joystick_max);
256     XInput.setJoystickY(JOY_RIGHT, 0);
257     Serial.println("Aim Right");
258 } else if (temp_string.equals("Menu")) {
259     //Actuate Menu button
260     XInput.press(BUTTON_START);
261     Serial.print("Menu");
262 } else if (temp_string.equals("View")) {
263     //Actuate View button
264     XInput.press(BUTTON_BACK);
265     Serial.print("View");
266 }
267 }
268
269 void button_state_reset() {
270     //Release all buttons neither button is pressed
271     if ((analogRead(BUTTON_Left) == HIGH) && (analogRead(BUTTON_Right) ==
HIGH)) {
272         XInput.release(BUTTON_A);
273         XInput.release(BUTTON_B);
274         XInput.release(BUTTON_X);
275         XInput.release(BUTTON_Y);
276         XInput.release(BUTTON_LB);
277         XInput.release(BUTTON_RB);
278         XInput.release(TRIGGER_LEFT);
279         XInput.release(TRIGGER_RIGHT);
280     }
281 }
282
283 void csv_parsing() {
284     //Initialise the parser
285     CSV_Parser cp(settings_chars, /*format*/ "-ssssss-");
286     int8_t *menus = (int8_t *)cp["Menu"];
287     char **ups = (char **)cp["Up"];
288     char **downs = (char **)cp["Down"];
289     char **lefts = (char **)cp["Left"];
290     char **rights = (char **)cp["Right"];
291     char **button_1s = (char **)cp["Button 1"];
292     char **button_2s = (char **)cp["Button 2"];
293     char **macros = (char **)cp["Macro"];
294
295     //Parse all the csv values into individual arrays for easy use
296     for (int i = 0; i < cp.getRowsCount(); i++) {
297         up_values[i] = String(ups[i]);
298         down_values[i] = String(downs[i]);

```

```
299     left_values[i] = String(lefts[i]);
300     right_values[i] = String(rights[i]);
301     button_1_values[i] = String(button_1s[i]);
302     button_2_values[i] = String(button_2s[i]);
303     //Print all the values to serial monitor for user to see
304     Serial.print(up_values[i]);
305     Serial.print(" - ");
306     Serial.print(down_values[i]);
307     Serial.print(" - ");
308     Serial.print(left_values[i]);
309     Serial.print(" - ");
310     Serial.print(right_values[i]);
311     Serial.print(" - ");
312     Serial.print(button_1_values[i]);
313     Serial.print(" - ");
314     Serial.print(button_2_values[i]);
315     Serial.println();
316 }
317 }
```

# APPENDIX B - CSV Editing Instructions

1. Open the Settings.csv file using Excel or Google Sheets
2. Change Button positions, add menus, or remove menus as you wish
3. Make sure to FILL EVERY MENU, this is very important for the software to work
4. Save the file as .csv(UTF-8)
5. Upload the file to the SD card
6. Plug the SD card into the reader and turn on the Arduino

Legend (Case Sensitive):

Input	CSV Code	In-Game-Action
Up	Up	Jump
Down	Down	Drop Down
Left	Left	Move Left
Right	Right	Move Right
Aim Up	Aim Up	Move Cursor Up
Aim Down	Aim Down	Move Cursor Down
Aim Left	Aim Left	Move Cursor Left
Aim Right	Aim Right	Move Cursor Right
A	A	Interact
B	B	Cancel
X	X	Drop Bomb
Y	Y	Switch Item
Left Bumper	LB	N/A
Right Bumper	RB	Character Ability
Left Trigger	LT	Active Item
Right Trigger	RT	Shoot
View	View	Map
Menu	Menu	Menu

Table 6.1: CSV Input Legend



# APPENDIX C - Information Letter

The research for the thesis “DIY Adaptive Gaming” centres on the design and realisation of an adaptive game controller for people with upper limb disabilities. To this end research needs to be conducted on the effectiveness of the controller, which will take the form of a user test which is pre-empted and followed by individual interviews. In the scope of the test, users will be asked to complete a single stage of a video game (Neon Abyss) using the adaptive controller. Following this they will be permitted to change the key bindings of the controller to their preferences and asked to complete the stage again. There are no known risks to the participants.

Participation is entirely voluntary, and consent may be withdrawn at any time without negative consequences and without any explanation.

Participants have the right, in principle, to access, rectify, delete, restrict, or object to the processing of personal data.

Interview responses will be recorded and deleted once transcribed (within 14 days). The interview recordings will be locally stored (not uploaded to the cloud) on the researcher’s phone and will only be accessible to him. Additionally, interview responses will be anonymised.

The only other data which will be collected will be the completion times for the stages, which will also be anonymised and randomised.

In case you have any additional questions or would like additional information, please email [d.s.matthews@student.utwente.nl](mailto:d.s.matthews@student.utwente.nl). Alternatively, you can contact my supervisor Edwin Dertien by emailing [e.dertien@utwente.nl](mailto:e.dertien@utwente.nl), or by contacting the ethics committee itself by emailing [ethicscommittee-CIS@utwente.nl](mailto:ethicscommittee-CIS@utwente.nl).

# APPENDIX D - User Instructions

1. Open the Settings.csv file using Excel or Google Sheets.
2. Change Button positions, add menus, or remove menus as you wish.
3. Make sure to FILL EVERY MENU, this is very important for the software to work.
4. Save the file as .csv(UTF-8).
5. Upload the file to the SD card.
6. Plug the SD card into the reader and turn on the Arduino.

Input	CSV Code	In-Game Action
Left Joystick Up	Up	Jump
Left Joystick Down	Down	Drop Down
Left Joystick Left	Left	Move Left
Left Joystick Right	Right	Move Right
A	A	Interact
B	B	Cancel
X	X	Use Bomb
Y	Y	Switch Item
Left Bumper	LB	
Right Bumper	RB	Character Skill
Left Trigger	LT	Active Ability
Right Trigger	RT	Shoot
Right Joystick Up	Aim Up	Move Cursor Up
Right Joystick Down	Aim Down	Move Cursor Down
Right Joystick Left	Aim Left	Move Cursor Left
Right Joystick Right	Aim Right	Move Cursor Right
Menu	Menu	Open Menu
View	View	Open Map

Table 6.2: Legend (Case Sensitive)

# APPENDIX E - User Test Interviews

## 6.1 User 1

### 6.1.1 Pre-Test Interview

*This is a pre-testing interview for the adaptive game controller. Am I allowed to record and transcribe your responses to use in my report?*

That's all right.

*Thank you. Do you play video games regularly?*

I haven't played video games regularly in a few years.

*What genre or what games did you play most often?*

I used to play a lot of FIFA and [Tony Hawk's] Pro Skater. But I also played some FPS games like Call of Duty on PC.

*Do you have experience with 2D roguelite games?*

No, I don't think I've ever played one of those, I'm not even sure what that is.

*Imagine Super Mario but with guns and only one life*

Ah ok ok.

*Do you have experience using controllers to play games?*

Yes, I would say I'm pretty comfortable with a controller. I played on the PlayStation and on the PSP.

*Do you have experience with customised or adaptive controllers?*

No just regular controllers.

*Do you have experience using tools to rebind keys in games?*

No, I never really felt the need to change the controller layout.

*Finally, have you played Neon Abyss before?*

I have not.

### 6.1.2 Post-Test Interview

*Did you feel that the controller was intuitive to use, and why?*

For me yes, once I got used to the hand position. I liked that the menu button is further away so you don't trigger it accidentally.

*Was it clear which menu layer you were on, and what did you think of using LEDs to show the layer?*

Yeah, it was very clear. The LED feedback was nice.

*Were you able to actuate the right inputs with the joystick?*

That worked just fine, but I did sometimes struggle to remember where each input was.

*What was your experience with the remapping tool?*

Yes, I had some difficulty when I was first starting, and had to really keep an eye on the cheat sheet. But I pretty quickly figured it out.

*How did your experience with the game change after you personalised the menu layers?*

It felt a bit better, but I only made changes to one layer. If I used it for longer I feel it would be better.

*Was anything unclear about the controller or remapping tool?*

No, for me it was quite intuitive.

*What would you like to change about the controller to make it easier to use?*

I think it's fine. I think it takes time to get really used to it, and remember what layer your inputs are on.

*What would you like to change about the remapping tool to make it easier to use?*

It would be nice to have it in game. It would also be easier if the individual layers were coloured the same as the LEDs.

*Do you have any questions about the system or my research?*

No, I think it was all quite clear.

*Well then, thank you for your time.*

My pleasure.

## 6.2 User 2

### 6.2.1 Pre-Test Interview

*This is a pre-testing interview for the adaptive game controller. Am I allowed to record and transcribe your responses to use in my report?*

Of course.

*Thank you. Do you play video games regularly?*

Yes, I play loads of games.

*What genre or what games did you play most often?*

Honestly I play a huge variety of games, I really like 4x style games like Victoria 3. I also enjoy total war and Company of Heroes.

*Do you have experience with 2D roguelite games?*

I played a tiny bit of Ori and the Will of the Wisps, but that's more of a metroidvania than a roguelite. So I'd say overall no, but I do know what they are.

*Do you have experience using controllers to play games?*

I have a decent amount of experience, but I do play on mouse and keyboard most of the time.

*Do you have experience with customised or adaptive controllers?*

No, I've only used PlayStation or Xbox controllers.

*Do you have experience using tools to rebind keys in games?*

Yes, I do it fairly often, but mostly in strategy games to make it more intuitive and faster for me.

*Finally, have you played Neon Abyss before?*

Nope.

## 6.2.2 Post-Test Interview

*Did you feel that the controller was intuitive to use, and why?*

I really like the controller. Initially, I struggled with the menu switching, it didn't really feel intuitive with the menu switching. I feel like I could get much better if I had a second controller, or time to master the current setup. But I really liked it, it felt like I didn't have to move as much. I would've enjoyed it more if the controller was a bit smaller so it fits my hand better, also a place for my index finger to sit would've been nice.

*Was it clear which menu layer you were on, and what did you think of using LEDs to show the layer?*

The second I had the lights figured out, I was locked in. But I did find that in the more stressful situations I struggled to pay attention to the lights, and was switching menus rapidly out of panic.

*Were you able to actuate the right inputs with the joystick?*

Towards the end yes, but I think with more time I would. I also feel like if the controller was a bit smaller this wouldn't be an issue.

*What was your experience with the remapping tool?*

Super simple. I feel like some more depth in the guide would be nice. It would've been good to see that I can add or remove whole menus. An image would also be nice so that I can tell which is button 1 or 2.

*How did your experience with the game change after you personalised the menu layers?*

My experience definitely changed for the better. I think getting rid of the third menu made life much easier for me. Perhaps adding an extra button would be nice. Perhaps also using the joystick hat as an input would be nice.

*Was anything unclear about the controller or remapping tool?*

Adding or removing menus was the only thing. A graphical interface instead of just a csv would also be nice.

*What would you like to change about the controller to make it easier to use?*

I just think the extra button would be nice. Additionally, depending on the game it might be nice to be able to use button combinations to trigger other effects.

*What would you like to change about the remapping tool to make it easier to use?*

Just a graphical interface.

*Do you have any questions about the system or my research?*

So the target group is wheelchair users and who else?

*People with upper limb disabilities. So people who struggle to use their hands or can't hold a controller properly. Also people who can't use certain fingers.*

Ah ok gotcha. That clears everything up I think.

*Was there anything else the controller made you realise?*

I just had no idea how many buttons I could use on a normal controller until now.

*I understand, I felt the same way initially. Well, that concludes our interview, thank you for your time.*

Not to worry, my pleasure.

## 6.3 User 3

### 6.3.1 Pre-Test Interview

*This is a pre-testing interview for the adaptive game controller. Am I allowed to record and transcribe your responses to use in my report?*

Yes yes.

*Thank you. Do you play video games regularly?*

Yes, I play games pretty much every day.

*What genre or what games did you play most often?*

I play pretty much everything, at the moment it's a lot of PUBG (Player Unknown's Battlegrounds) and some single player RPGs like Hogwarts Legacy.

*Do you have experience with 2D roguelite games?*

I played one or two, but it's not really my kind of game.

*Do you have experience using controllers to play games?*

Yeah I play with controllers a lot, especially Rocket League.

*Do you have experience with customised or adaptive controllers?*

No, not at all.

*Do you have experience using tools to rebind keys in games?*

Yeah I rebind keys often.

*Finally, have you played Neon Abyss before?*

I don't think I've even heard of it before, so no.

### 6.3.2 Post-Test Interview

*Did you feel that the controller was intuitive to use, and why?*

I felt that once I got the hang of it, and understood that I couldn't do everything all at the same time it was pretty good. I just struggle with the fact that I can normally do so much at one time, like moving, aim, and shooting all together.

*Was it clear which menu layer you were on, and what did you think of using LEDs to show the layer?*

Yeah when I was focusing it was pretty easy, but in like the boss fight it got a bit hard. I was just switching really fast and then it became difficult to tell what layer I was on.

*Were you able to actuate the right inputs with the joystick?*

I had no problems with that.

*What was your experience with the remapping tool?*

I think because I'm used to doing it, I found it quite easy. I just had to adjust from not having a table instead of a user interface.

*How did your experience with the game change after you personalised the menu layers?*

It felt so much easier once I put everything where I wanted it. Not that I changed a lot, but I think being able to decide where everything was just made it click a bit more.

*Was anything unclear about the controller or remapping tool?*

No no, I think I understood it all.

*What would you like to change about the controller to make it easier to use?*

I think that having a second controller would make life easier, because then you can just do more things, but I understand that's not really feasible for these tests.

*What would you like to change about the remapping tool to make it easier to use?*

Yeah I think a GUI (graphical user interface) would be really nice, because then I don't have to feel like I'm editing a spreadsheet to play my game.

*Do you have any questions about the system or my research?*

No, I think you explained everything I was confused about.

*Thank you for your time.*

Of course.

# APPENDIX F - Wiring Diagram

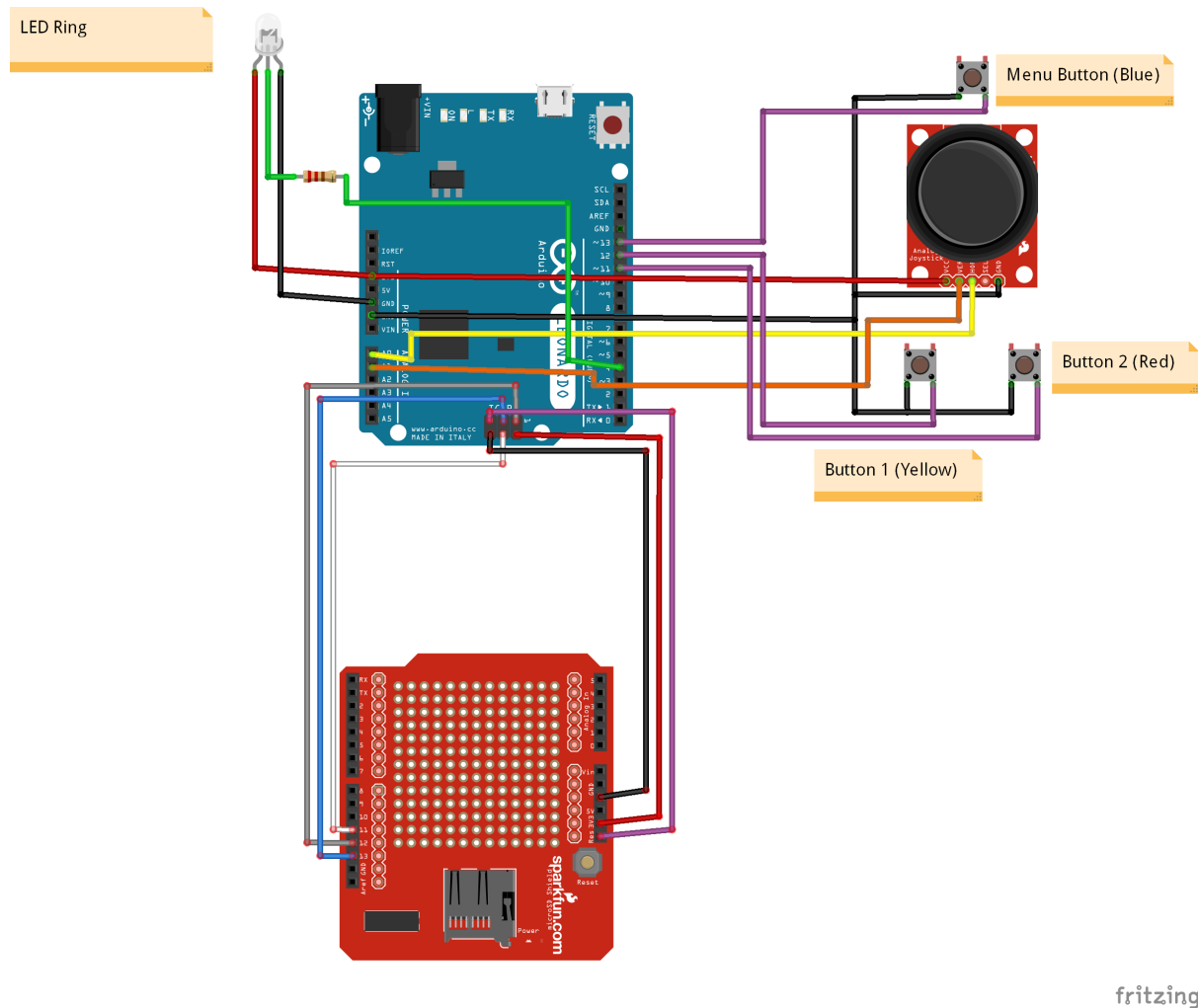


Figure 6.1: Prototype Wiring Diagram



# APPENDIX G - Arduino Leonardo Pinout

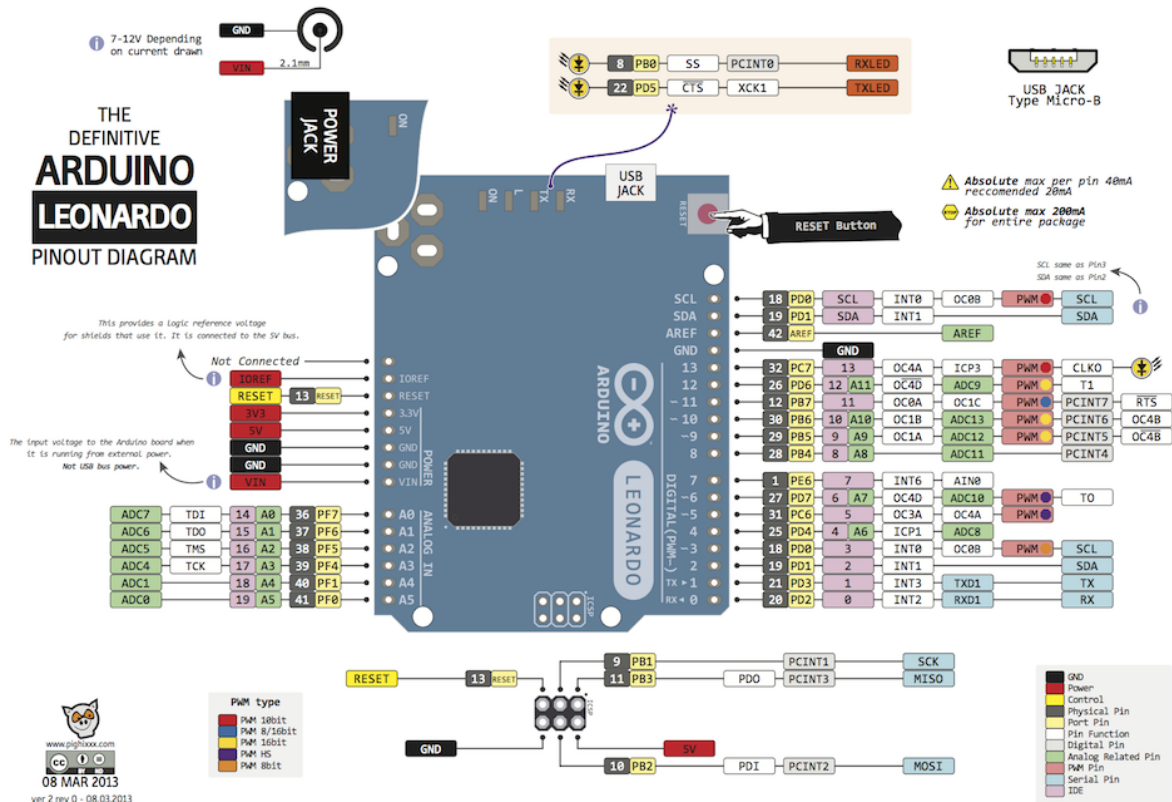


Figure 6.2: Arduino Leonardo Pinout [27]

# Bibliography

- [1] E. Yiu and A. Kornberg, “Duchenne muscular dystrophy,” *Journal of paediatrics and child health*, vol. 51, no. 8, pp. 759–764, 2015.
- [2] M. H. Brooke, G. M. Fenichel, R. C. Griggs, *et al.*, “Duchenne muscular dystrophy,” *Neurology*, vol. 39, no. 4, pp. 475–475, 1989, ISSN: 0028-3878. DOI: 10.1212/WNL.39.4.475. eprint: <https://n.neurology.org/content/39/4/475.full.pdf>. [Online]. Available: <https://n.neurology.org/content/39/4/475>.
- [3] E. Ellcessor, *Restricted access: Media, disability, and the politics of participation*. NYU Press, 2016, vol. 6.
- [4] C. Omtzigt, *Developing a modular gaming handheld for gamers with muscular dystrophy*, Jun. 2022. [Online]. Available: <http://essay.utwente.nl/93471/>.
- [5] F. Iacopetti, L. Fanucci, R. Roncella, D. Giusti, and A. Scebba, “Game console controller interface for people with disability,” *2008 International Conference on Complex, Intelligent and Software Intensive Systems*, pp. 757–762, 2008. DOI: 10.1109/CISIS.2008.77.
- [6] J. Hamadeh, L. Lawrence, R. Hasslinger, and L. Fike, “Redesigning a gaming controller,” 2008.
- [7] *We gaan het maken*, 2021. [Online]. Available: [https://www.uitzendinggemist.net/aflevering/547717/We\\_Gaan\\_Het\\_Maken.html](https://www.uitzendinggemist.net/aflevering/547717/We_Gaan_Het_Maken.html).
- [8] N. Villar, K. M. Gilleade, D. Ramdunyellis, and H. Gellersen, “The voodooio gaming kit: A real-time adaptable gaming controller,” *Computers in Entertainment (CIE)*, vol. 5, no. 3, p. 7, 2007.
- [9] C. L. Fall, F. Quevillon, M. Blouin, *et al.*, “A multimodal adaptive wireless control interface for people with upper-body disabilities,” *IEEE transactions on biomedical circuits and systems*, vol. 12, no. 3, pp. 564–575, 2018.
- [10] M. Hassan, Y. Shimizu, Y. Hada, and K. Suzuki, “Joy-pros: A gaming prosthesis to enable para-esports for persons with upper limb deficiencies,” *IEEE Access*, vol. 10, pp. 18 933–18 943, 2022.
- [11] J. K. Muguro, P. W. Laksono, W. Rahmaniari, *et al.*, “Development of surface emg game control interface for persons with upper limb functional impairments,” *Signals*, vol. 2, no. 4, pp. 834–851, 2021.
- [12] S. Vickers, H. Istance, and A. Hyrskykari, “Performing locomotion tasks in immersive computer games with an adapted eye-tracking interface,” *ACM Transactions on Accessible Computing (TACCESS)*, vol. 5, no. 1, pp. 1–33, 2013.

- [13] K.-J. Wang, Q. Liu, Y. Zhao, *et al.*, “Intelligent wearable virtual reality (vr) gaming controller for people with motor disabilities,” *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pp. 161–164, 2018.
- [14] T. A. Scardovelli and A. F. Frère, “The design and evaluation of a peripheral device for use with a computer game intended for children with motor disabilities,” *Computer methods and programs in biomedicine*, vol. 118, no. 1, pp. 44–58, 2015.
- [15] Y. Lin, J. Breugelmans, M. Iversen, and D. Schmidt, “An adaptive interface design (aid) for enhanced computer accessibility and rehabilitation,” *International Journal of Human-Computer Studies*, vol. 98, pp. 14–23, 2017, ISSN: 1071-5819. DOI: <https://doi.org/10.1016/j.ijhcs.2016.09.012>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1071581916301264>.
- [16] C. Godineau, “The new xbox adaptive controller, another step towards digital inclusion,” *Masters of Media*, 2018.
- [17] H. Nishino, *Introducing project leonardo*, 2023. [Online]. Available: <https://blog.playstation.com/2023/01/04/introducing-project-leonardo-for-playstation-5-a-highly-customizable-accessibility-controller-kit/>.
- [18] S. Vickers, H. Istance, and M. J. Heron, “Accessible gaming for people with physical and cognitive disabilities: A framework for dynamic adaptation,” pp. 19–24, 2013.
- [19] Elduderino, *Controller input icons*, 2020. [Online]. Available: <https://opengameart.org/content/controller-input-icons>.
- [20] *Microsd card icon*. [Online]. Available: <https://nl.farnell.com/en-NL/transcend/ts2gusdc/card-sd-micro-2gb/dp/2290242>.
- [21] *Arduino leonardo icon*. [Online]. Available: <https://www.eitkw.com/product/arduino-leonardo/>.
- [22] *Desktop icon*. [Online]. Available: <https://www.bol.com/nl/nl/p/circular-rgb-gaming-pc-intel-core-i7-10700f-geforce-rtx-3060-12-gb-gddr6-32-gb-ddr4-1-tb-ssd-nvme-windows-11-pro/9300000122730616/>.
- [23] M. Borowski, *Csv parser library for arduino*. [Online]. Available: <https://github.com/michalmonday/CSV-Parser-for-Arduino>.
- [24] D. Madison, *Xinput library for arduino*. [Online]. Available: <https://github.com/dmadison/ArduinoXInput>.
- [25] Adafruit, *Adafruit neopixel library for arduino*. [Online]. Available: [https://github.com/adafruit/Adafruit\\_NeoPixel](https://github.com/adafruit/Adafruit_NeoPixel).
- [26] A. Foundation, *Sd library for arduino*. [Online]. Available: <https://www.arduino.cc/reference/en/libraries/sd/>.
- [27] *Arduino leonardo pinout*. [Online]. Available: <https://europe1.discourse-cdn.com/arduino/original/4X/e/7/1/e7108daa6430b7f6a38fb122811ef990642d6449.png>.