6COSC023W – Final Project Report

Audio Interactive LED Wristband

(AILED)

Student: Mihai Simion (w1730967)

Supervisor: Francois Roubert

This report is submitted in partial fulfilment of the requirements for the

BSc (Hons) Computer Science degree

BEng Software Engineering degree

at the University of Westminster.

School of Computer Science & Engineering

University of Westminster

Declaration

This report has been prepared based on my own work. Where other published and unpublished source materials have been used, these have been acknowledged in references.

Word Count:
Student Name: Mihai Simion
Date of Submission: 2.05.2023

Abstract

Every modern music festival prefers using pre-made light shows to entertain the crowd nowadays. Although this shows the true showmanship of the entertainer, it does not let the crowd participate. This project focuses on developing a system that allows the user to express themselves using their own light effects. The first section of the document displays the goal and state of the art of the project. The second section explains the design choices made and the actual development of a prototype.

This project is an explorative project. It is a path that allows extensive learning of multiple fields. A fully working product might not be reached but only a mere prototype.

All the 3D models, code files documents and videos can be found in the GitHub repository. The link is in the appendix.

Table of contents

Cuprins

6COSC023W – Final Project Report 1
Declaration 2
Abstract 3
Table of contents 4
List of figures 5
1. AILED Project Foundations 8
1.1 Problem statement 8
1.2 Aims and Objectives 8
2. Background 9
2.1 Manufacturing 9
2.1.1 Subtractive Manufacturing 9
2.1.1.1 Subtractive Manufacturing Basics 9
2.1.1.2 Usage of Subtractive Manufacturing 9
2.1.1.3 The Subtractive Manufacturing Advantage 10
2.1.1.4 The Subtractive Manufacturing Limitation 10
2.1.2 Additive Manufacturing 10
2.1.2.1 Additive Manufacturing Basics 10
2.1.2.2 Usage of Additive Manufacturing 12
2.1.2.3 The Additive Manufacturing Advantage 13
2.1.2.4 The Additive Manufacturing Limitations 14
2.1.3 Decision 14
2.2 Sound Processing 14
2.3 Communicating 16
2.3.1 Wireless Communication Technologies 16
..... 18
2.3.2 Timing of Communication 18
2.3.3 Bandwidth & Data Rate of Sensors 18
2.3.4 Decision 19
3. Law, societal norms and moral principles 20
4. Methodology 20
5. Requirements Engineering Summary 20
6. Development of the hardware component of the AILED 21

6.1 Functionality 22
6.2 Design 22
6.2.1 Electronic components schematics 22
6.2.2 3D printed Bracelet 29
6.3 Development 30
6.3.1 Soldering 30
6.3.1 Wiring electronic components 35
6.3.2 3D Printing 36
6.4 Testing 44
7. Development of the software component of the AILED 47
7.1 Functionality 47
7.2 Design 47
7.2.1 Choice of programming language 47
7.2.2 Algorithm for sound-light accuracy 50
..... 51
7.2.3 Creating the Android application 52
7.2.4 Implementing the Bluetooth protocol 52
7.3 Development 53
..... 61
7.4 Testing 68
8. Critical Evaluation of the AILED Project and Conclusions 69
8.1 Review of the work completed in the AILED project 69
8.2 Further expansions 69
8.3 Closing statements 69
9. References 70
Appendix I 72

List of figures

Figure 1. AM process [5] 10
Figure 2. Nozzle filament generation [31] 11
Figure 3. Material produced using AM [5] 13
Figure 4. Definition of Discrete Fourier Series. [8] 15
Figure 5. Audio sinusoid in time domain converted to frequency domain using FFT. [7] 15
Figure 6. Communication technologies for applications, and their properties [6] 17

Figure 7. Suitability of communication technologies for application domains [6] 18
Figure 8. Circuit diagram for AILED 23
Figure 9. Datasheet for SM5050 LED showing the 3 individual diodes. [11] .24
Figure 10. Picture with a WS2812B LED where you can see the smart controller. [12] 25
Figure 11. LM393 microphone [15] 26
Figure 12. MAX4466 microphone [17].....27
Figure 13. MAX9814 microphone with auto-gain [19] 28
Figure 14. Datasheet for Raspberry Pi Pico W [20].....29
Figure 15. Workplace on Tinkercad with all the models 30
Figure 16. Process of soldering [21].....31
Figure 17. Actual canonical tips used.....32
Figure 18. Actual sponge used 32
Figure 19. Actual soldering iron with stand used.....33
Figure 20. Actual solder material used.....34
Figure 21. Soldered components 35
Figure 22. Actual prototype used 36
Figure 23. Basic shapes in Tinkercad 37
Figure 24. Microcontroller model with its real dimensions in millimetres 38
Figure 25. Basic shape of a bracelet.....39
Figure 26. Cap model that holds the cylinders together.....39
Figure 27. Cap with hole for micro-USB.....40
Figure 28. Actual 3D printer used.....41
Figure 29. Interface of Cura 42
Figure 30. GIF with working 3D printer.....43
Figure 31. 3D printed components used for AILED.....44
Figure 32. Breadboard with a microcontroller, microphone and LED band with headers. 45
Figure 33. Miniature sized components made in 25 minutes instead of 12 hours. 46
Figure 34. Fading an LED in Python [22] 48
Figure 35. Fading an LED in C [23].....49
Figure 36. Composition of 24 bits data.....51
Figure 37. Cascade method for passing data on linked WS2812B LEDS [13] 51
Figure 38. Data transmission method on linked WS2812B LEDS [13].....52

Figure 39. Boilerplate code for CMakeList.txt file 54
Figure 40. Example from github for fading a simple LED [23] 55
Figure 41. Blinking LED 56
Figure 42. Blinking RGB LED.....57
Figure 43. Code for lighting one WS2812B LED [28] 58
Figure 44. Rainbow effect on a WS2812B LED band 59
Figure 45. Code for setting up ADC and sampling the signal [29].....61
Figure 46. The frequency was found when a frequency of 440 Hz was played using a tuner app. 62
Figure 47. Code for mapping the frequency to an RGB value.....63
Figure 48. AILED found on the android phone 64
Figure 49. Flutter code for the Android application.....65
Figure 50. UI of the Android application 66
Figure 51. Bluetooth controller class. 67
Figure 52. Application communicating with the bracelet.**Eroare! Marcaj în document nedefinit.**
Figure 53. AILED reacting to music from speaker and then changes its colour based via Bluetooth .....**Eroare! Marcaj în document nedefinit.**

1. AILED Project Foundations

AILED or Audio Interactive Light Emitting Diode represents an innovative and dynamic product that is capable of reacting to the environment, specifically music, and emitting an array of colours that are synchronised with the frequency and beat of the sound.

1.1 Problem statement

Stage lighting is vital in creating a memorable event in today's entertainment industry. Audiences expect a high level of energy and spectacle, and lighting and special effects can enhance a show and draw in even the back of the crowd. Different lighting elements can inspire and motivate the audience, capture attention, or create breathtaking moments, which is why lighting is essential in any show. An excellent stage designer must install and coordinate expensive equipment such as video walls, laser lighting, or pyrotechnics. However, while these effects may excite the audience, they can make them feel detached from the show. A 2005 study by Michelle Duffy found that music can help develop a sense of belonging and place. Good lighting can distinguish between a good show and a great one.

1.2 Aims and Objectives

The goal of this project is to create a system that helps people feel connected to events. It involves a wristband that changes colour based on the music or lights up according to user preferences. When worn by a group of friends at an entertainment venue, the bracelets will synchronize, making the wearers easily identifiable and promoting bonding among them. The system will be investigated, analysed, designed, implemented, and critically evaluated to ensure its effectiveness.

This is an experimental project, and its potential academic use is still being evaluated. It falls into the category of embedded systems and can be used in various environments, such as parties, clubs, and music festivals. In addition, it can make urban transport more visible.

1. Here are the objectives I have for this project:
2. Establish clear foundations such as goals, objectives, and scope. The first chapter of your dissertation, titled "Project Foundations," will serve as the deliverable for this objective.
3. Thoroughly research embedded systems and provide chapter 2 in the dissertation titled "Background research" as the outcome for this objective.
4. Provide a step-by-step explanation of the methodology used to create the bracelet that changes colours based on music or to make the bracelet itself. Chapter 3 of the dissertation, "Methodology," will deliver this objective.
5. Define the development steps that I need to do to get working hardware. The deliverable for this objective will be chapter 6 of the dissertation titled "Development of the hardware component of the AILED."

6. Define the development steps that I need to do to get working software. The deliverable for this objective will be chapter 7 of the dissertation titled "Development of the software component of the AILED."
7. Do a critical review of the entire project where I explain my accomplishments, areas for improvement, and suggestions for future actions. The deliverable for this objective will be chapter 6 of the dissertation titled "Critical Evaluation of the AILED Project and Conclusions."

2. Background

The project's steps will be broken down in this debate, and the options that are accessible for each phase will be taken into account. There will also be a discussion of the decisions that will be taken for each step.

2.1 Manufacturing

2.1.1 Subtractive Manufacturing

The project will utilise subtractive manufacturing techniques to produce prototypes constructed from plastic.

2.1.1.1 Subtractive Manufacturing Basics

The industrial revolution shaping the world today is largely driven by traditional production methods, also known as subtractive manufacturing. Despite its influence, subtractive manufacturing is inherently limited and requires innovation. The term 'manufacturing' comes from the French for 'made by hand', but modern manufacturing techniques have evolved beyond manual labour. Complex processes such as casting, moulding and machining use tools, machines, computers and robots to make products. These subtractive processes remove material from the part to create the final product, but the capabilities of the tools used are limited.

2.1.1.2 Usage of Subtractive Manufacturing

The idea of subtractive production is that there is something at the beginning and less at the end. The sculptor starts with a piece of stone or wood and gradually reduces it until the sculpture is complete. This is an example of subtractive work. When this happens in the industry, it is called subtractive production.

Examples of industrial applications of SM are:

- Metal-cutting: Sheet metal cutting is an important process in many industries, such as automotive, construction, aerospace, electronics, and marine.

- Laser cutting: Aerospace is a very important industry and one of the biggest users of laser metal cutting due to the precision and versatility requirements of structural and mechanical components.

- Water jet cutting: Aerospace and automotive workshops for efficient cutting of shapes and parts. For very hard materials, a granular abrasive is added to the water stream to increase cutting power. The waterjet is capable of cutting almost any material.

2.1.1.3 The Subtractive Manufacturing Advantage

SM offers a number of benefits, such as:

- Versatility: It applies to various materials like metal, plastic, wood, and composites
- Accuracy: Produces high accuracy with 0.0025 mm close tolerance
- Speed: Molding is one of the fastest ways to manufacture products

2.1.1.4 The Subtractive Manufacturing Limitation

SM has many advantages but also some limitations: the main limitation of SM is that the material used is wasted. Also, metal casting and injection moulding requires new products and new moulds to cast the parts. Different products require different machining tools.

2.1.2 Additive Manufacturing

2.1.2.1 Additive Manufacturing Basics

Additive manufacturing (AM) uses 3D model data to create objects by joining materials layer by layer rather than subtractive manufacturing methods.

Additive Manufacturing (AM) refers to a collection of advanced techniques used for creating objects by adding layers of material. This process can be compared to building with blocks or Legos®. Figure 1 shows the fundamental steps involved in AM technologies:

Figure 1. AM process [5]

Additive Manufacturing (AM) is a process that requires using a 3D model. This model can be created using specialised software like CAD or scanning an existing object.

Afterwards, the 3D model is divided into distinct layers using specialised software to generate a computer file sent to the AM machine. The AM machine forms each layer of the object through the selective placement of the material, much like an inkjet printer adds layers of ink to create a printed page. Several AM processes differ in making each layer, including "Fused Filament Fabrication", which extrudes thermoplastic or wax material through heated nozzles to create part cross-sections. Figure 2 provides an example of this technique:

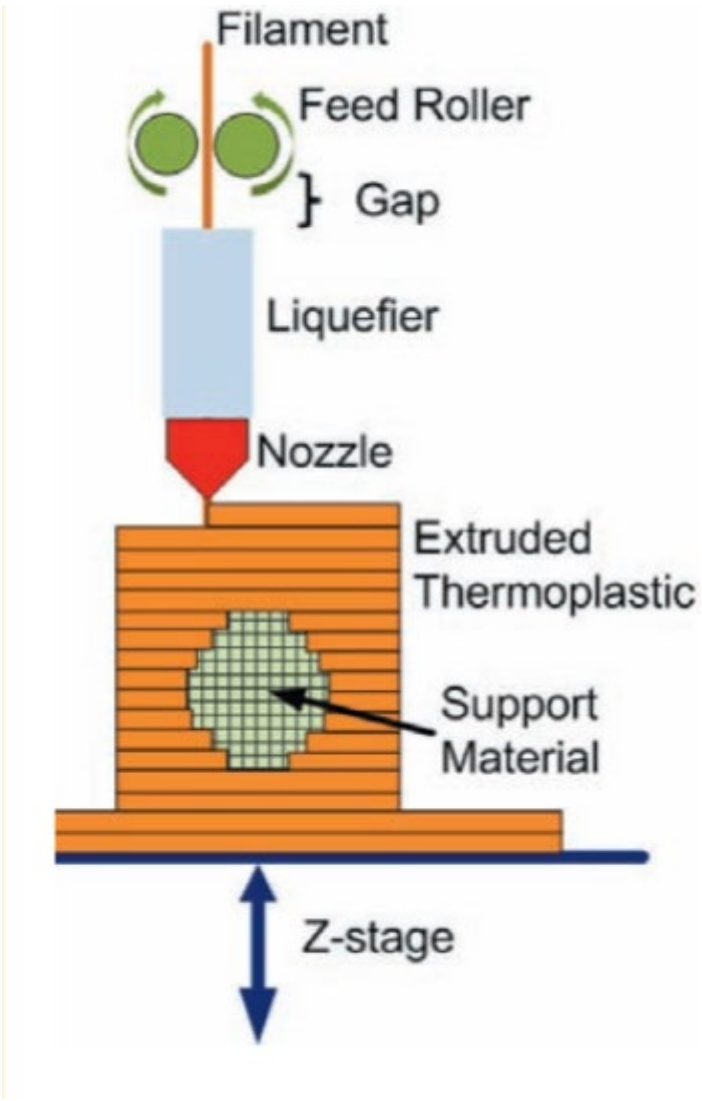


Figure 2. Nozzle filament generation [31]

A roller guides the filament feedstock into a liquefier heated temperature higher than the filament's melting point. The material can move quickly through the nozzle and onto the substrate, which cools and becomes solid. Once a layer is finished, the Z-stage lowers the build platform, and the next layer is added. A secondary sacrificial material may also be deposited and later removed to support the creation of overhanging shapes. Different additive manufacturing technologies use various methods to create each layer, such as jetting a binder into a polymeric powder for 3D

printing or using a UV laser to harden a photosensitive polymer for stereolithography. There has also been a recent development in using multiple materials simultaneously for end-use products, similar to an inkjet printer with six colour cartridges.

2.1.2.2 Usage of Additive Manufacturing

At first, AM was known as "rapid prototyping" and was mainly used to produce quick models of new product concepts for form and fair assessment. For example, an architect could create a building design on a computer and print a 3D model to show the client or improve the design further. An automotive engineer could design and print a prototype front for a vehicle. As the material properties and process repeatability improved, AM technology evolved from creating only prototypes to producing parts for functional testing, tooling for injection moulding and sand casting, and, ultimately, directly manufacturing end-use parts. In 2009, Wohlers reported that 16% of AM process use was for direct part production, 21% for functional models, and 23% for tooling and metal casting patterns. Some notable examples of using AM for part production in the industry include:

- Automobile components: Currently, Additive Manufacturing (AM) is not a viable option for the mass production of automobile parts. However, it is gaining traction in creating high-end and specialised vehicle components. A prime example is the successful use of direct metal laser sintering in fabricating engine parts for prestigious Formula 1 race cars.
- Aircraft components: The aerospace industry has the potential to be significantly impacted by additive manufacturing (AM) due to its production of limited quantities. While direct metal AM techniques have not yet attained critical component grade, AM parts have already been integrated into aircraft. One such instance is the F-18's environmental control system duct, which was redesigned with the aid of AM. The intricate nature of AM allowed for the assembly to be reimagined, and the number of parts was reduced from sixteen to a single component. Unlike traditional manufacturing, where designs are customized to fit machine tools, AM parts are constructed to serve their purpose with precision.
- Custom orthodontics: Align Technology, Inc. employs advanced additive manufacturing technology to produce bespoke clear braces for a diverse range of patients across the globe. The process entails utilizing stereolithography to generate moulds from 3D scans of each patient's dental impressions. These moulds are subsequently filled with FDA-approved polymer to manufacture customized braces, ensuring the highest quality and safety for patients.
- Custom hearing aids: Siemens and Phonak have adopted laser sintering technology to produce tailored hearing aids efficiently. The procedure involves obtaining 3D scans of ear canal impressions, which enables the development of a hearing aid that fits flawlessly and is practically invisible to the unaided eye.

2.1.2.3 The Additive Manufacturing Advantage

There are several advantages to using additive manufacturing compared to traditional techniques like injection moulding, casting, stamping, and machining.

- Increased part complexity: Additive manufacturing (AM) offers a significant advantage in producing complex shapes that are impossible to create with other methods. One such example is the ability to incorporate curved internal cooling channels in components. Essentially, AM technology permits designers to place material only where necessary. By drawing inspiration from natural structures like coral, wood, and bone, designers can now create cellular materials that are strong, stiff, and lightweight, as shown in Figure 3:

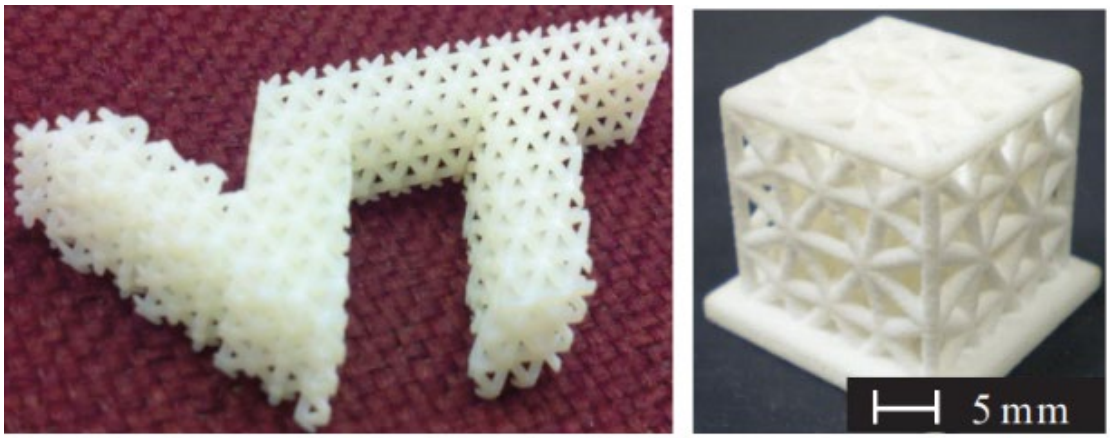


Figure 3. Material produced using AM [5]

- Digital design and manufacturing: Additive manufacturing (AM) processes involve creating physical parts directly from a standardised digital file, typically represented in the.STL format as a three-dimensional solid model. These computer-controlled processes require minimal operator expertise and reduce the need for human interaction during the manufacturing process. Often, these processes operate unmonitored, facilitating overnight builds and significantly decreasing the time required to produce products. Additionally, the direct creation of the part from the computer model ensures that the resulting product accurately reflects the designer's intent, minimising inaccuracies often found in traditional manufacturing processes.

- Complexity is free: A new mould must be made when creating a new product using metal casting or injection moulding. Machining requires several tool changes to complete the product. However, with the additive manufacturing (AM) process, only one tool is needed regardless of the desired shape. This means complex objects can be created without incurring additional costs or extending lead time. AM processes are perfect for producing customised and intricate geometries. For instance, in custom orthodontics, AM can produce multiple unique moulds in a single batch run, allowing for printing many teeth

moulds all at once. This level of customisation cannot be achieved economically using traditional manufacturing methods.

- Instant production on a global scale: The use of digital files to represent physical objects allows for quick and easy distribution of products worldwide, similar to how MP3s revolutionised the music industry. These digital files can be sent to any printer capable of producing the product within the specifications of the file, including size, resolution, and materials used. This process is environmentally friendly because it reduces waste, as only the necessary material is used in production. This contrasts traditional manufacturing methods, such as machining, often producing excess material waste.

2.1.2.4 The Additive Manufacturing Limitations

Although AM technologies have numerous advantages over traditional manufacturing methods, they have some limitations that prevent them from being a one-stop solution for every manufacturing challenge. Due to their limitations, AM processes could be more suitable for mass production. While AM processes are expected to become faster, moulding technologies may be slower due to the fundamental physics of the processes. For instance, it takes about an hour for AM processes to create a 1.5-inch cube, whereas an injection moulding machine can produce several similar parts in less than a minute. However, this bottleneck only applies to producing several thousand common parts, as tooling needs to be created for each unique part that one wishes to mould. AM is the preferred process for custom parts.

2.1.3 Decision

For this project, using additive manufacturing will be the best choice as it will reduce costs and increase flexibility.

2.2 Sound Processing

As Rocchesso, D explains in its book, “Introduction to sound processing Mondo estremo”. A sound processing system is defined as a single-input single-ouput (SISO) system where the processing algorithm takes one single input signal and outputs a modified single-ouput signal. A mathematically widely used algorithm for sound processing is Fast-Fourier Transform (FFT) which is the fast version of Discrete Fourier Transform (DFT) used for transforming a time-domain representation of an audio signal (waveforms for example) into a frequency-domain representation. This information can later be used for audio equalization, noise filtering or audio compression:

Discrete Fourier Series:

$$X(j) = \sum_{n=0}^{N-1} A(n) W_N^{jn}$$

where $W_N = \exp(2\pi i/N)$

Inversion Formula:

$$A(n) = \frac{1}{N} \sum_{j=0}^{N-1} X(j) W_N^{-jn}$$

Let us write: $X(j) \longleftrightarrow A(n)$

Periodicity:

$$W_N^N = 1, \qquad W_N^{j+N} = W_N^j$$

$$W_N^j = W_N^{j \bmod N} \qquad 0 \leq j < N$$

$$X(j) = X(j \bmod N)$$

$$A(n) = A(n \bmod N)$$

Orthogonality:

$$\sum_{j=0}^{N-1} W_N^{nj} W_N^{-mj} = \begin{cases} N & \text{if } n = m \bmod N \\ 0 & \text{otherwise.} \end{cases}$$

Figure 4. Definition of Discrete Fourier Series. [8]

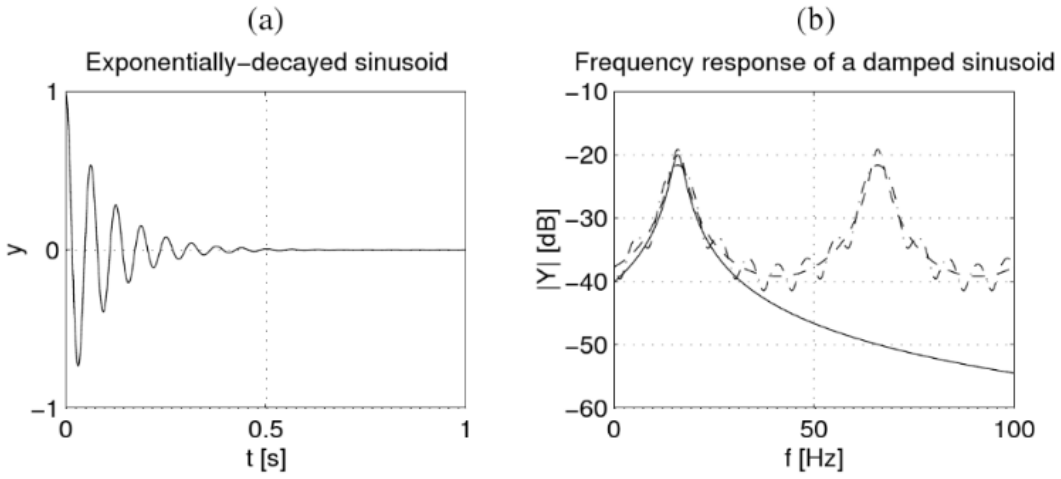


Figure 5. Audio sinusoid in time domain converted to frequency domain using FFT. [7]

2.3 Communicating

2.3.1 Wireless Communication Technologies

Various wireless communication technologies are available to connect a device to local networks and the internet. These technologies facilitate connecting local networks or individual devices to the internet [6].

- NFC: This technology allows devices to communicate wirelessly over a short range of approximately 20 cm. It utilises a tag which stores a small amount of data that can either be read-only (like an RFID tag for identification) or re-writable, allowing for changes to be made later by the device.

- Bluetooth: This technology has become increasingly common in smartphones, tablets, laptops, and headsets. There are three types of Bluetooth: Classic Bluetooth, Bluetooth Low Energy (BLE), and Bluetooth 5.0 (BT v5). Classic Bluetooth is suitable for data stream applications but has limitations, such as the network's limited number of nodes and topology. BLE, also known as Bluetooth Smart, is designed for short-range, low bandwidth, and low latency applications. It has advantages over Classic Bluetooth, such as lower power consumption, lower setup time, and the ability to support a star topology with unlimited nodes. Bluetooth 5.0 (BT v5) increases the range and doubles the speed of low-energy connections while also increasing the capacity of broadcasts.

- ZigBee: This wireless specification is small in size, low in cost and power consumption. It can support network topologies such as mesh, star, and tree. The transmission range is vast and depends on the output power. However, even though ZigBee has been used in some industrial applications and WSN nodes, it faces market barriers due to the emergence of better alternatives like BLE. BLE provides higher bandwidth and consumes less energy.

- WiFi:
    - Conventional WiFi (IEEE 802.11 b/g/n): Urban districts benefit significantly from this option's high bandwidth and availability. However, there may be better choices for ultra-low-power devices due to their high energy consumption.
    - Low-power WiFi (802.11 ah) or HaLow: Wireless networks powered by unconventional WiFi technology offer several advantages over conventional systems. These networks are designed to transmit data

over greater distances with less energy consumption and a lower data rate. Moreover, they experience less interference from other wireless networks due to using a unique frequency band (0.9 GHz).

- Cellular network: Mobile networks like 3G and LTE offer dependable high-speed internet connectivity. However, these networks consume much power and must be better suited for M2M or local network communication.

- Low Power Wide Area Network (LPWAN)]: Certain technologies are well-suited for low-power applications requiring long-range transmission. These technologies enable distances of up to 10 kilometres between end nodes and gateway but with a trade-off of a low data rate of less than 1 kilobit per second. LPWAN, comprising SigFox, LoRaWAN, and Weightless, is the primary technology in this space, operating in sub-GHz bands. One of the challenges faced by LPWAN is the need for a universally available sub-GHz band.

When designing wireless devices, it is essential to consider the specific characteristics of the available technologies. Table 1 shows typical values of these parameters according to the design constraints of the device. For example, the transmission range of a device can be increased or reduced by adjusting the transmission power and antenna size. Table 2 shows the suitability of different wireless technologies for different application areas, each with its advantages and disadvantages. However, some technologies, such as Bluetooth, WiFi and ZigBee, are susceptible to interference from other devices operating in the same frequency range (e.g. 2.4 GHz), especially as the number of devices increases. This interference can lead to a significant drop in data transfer rates, resulting in higher power consumption, reduced quality of service and missed deadlines in real-time applications, affecting overall system optimisation goals.

		NFC	Bluetooth	BLE	BT v5	ZigBee	HaLow 802.11 b/g/n	LP WiFi 802.11ah	LPWAN	Cellular network	
										3G	LTE
Range	indoor	<0.2 m	1–100 m	∼100 m	<300 m	<20 m	<70 m	<700 m	<10 Km	>5 Km	>5 Km
	outdoor					<1500 m	<230 m	<1000 m			
Bit rate [Mpbs]		0.424	1–3	1	2	0.25	>1	0.15–40	<0.05	0.17	75–300
Throughput [Mbps]		0.22	1.5	0.30	1.5	0.15	2-50	>0.1	<0.05	NA	
freq. [GHz]		0.014	2.4–2.5	2.4–2.5	2.4	2.4	2.4/5	0.9	sub-GHz	0.8–1.9	2.1
Network topology		p2p	scatternet	star, scatternet	NA	star, tree, mesh	star	star	star	NA	

Figure 6. Communication technologies for applications, and their properties [6]

		Application domains					Local Network (M2M)
		Healthcare	Smart Cities	Smart Building	Automotive	Industry	
wireless tech.	NFC	medium	high	low	very low	very high	medium
	BLE	very high	low	low	very low	low	high
	ZigBee, BT v5	medium	high	very high	low	high	high
	WiFi b/g/n	low	high	medium	medium	low	high
	HaLow	high	very high	high	high	high	very high
	LPWAN	low	very high	high	high	very high	high
	Cellular networks (3G, LTE, etc.)	low	high	high	high	medium	very low

Figure 7. Suitability of communication technologies for application domains [6]

2.3.2 Timing of Communication

The timing of data transmission schemes in applications can be classified into three categories.

- **Continuous**: The devices constantly send or receive data, such as real-time health monitoring.
- **Sporadic**: The device gathers and saves data and sends it when a connection is available.
- **On-demand**:
  - **User-driven**: The device can send the collected data when the operator requests.
  - **Event-driven**: Communication occurs only when a specific event takes place.

When designing a high-performance system, it is crucial to consider data transmission timing, mainly when dealing with low-power modes such as deep sleep, standby, and active. To efficiently use power, a predictive model that uses learning techniques can manage the wireless transceiver's low-power modes when data transmission is not continuous.

2.3.3 Bandwidth & Data Rate of Sensors

This section provides an overview of the main applications of sensors. It also presents typical data collection rates for different sensors based on the usage scenarios reported by these sensors. It also includes comments on the required wireless transmission bandwidth and related techniques. Finally, the processing power required to handle the typical processing operations is estimated, and suitable hardware cores are proposed to support them. [6]

- **Ambient/Object Temperature, Humidity**: Different applications, including such as smart homes, smart cities, and smart industries, use these sensors.

- **Accelerometers and Gyroscopes**: Sensors like these are commonly utilised in various industries, such as industrial machinery, medical and fitness equipment, and wearable devices. They are primarily used for monitoring physical activity, detecting falls, navigation, and checking the structural health of machinery or buildings.

- **Magnetometer**: This device is typically found alongside accelerometers and gyroscopes and is utilised for navigation by determining direction.

- **Light**: This sensor is usually used for saving energy consumption by adjusting the lighting (e.g. smartphones, smart homes, smart cities).

- **Chemical Sensors**: These sensors are mainly used for measuring the air quality (e.g. CO, NO2, SH2, and CO2) in smart city applications, environment monitoring, or smart factories.

- **Location**: These sensors primarily utilise GPS technology and are commonly employed in smart industries and transportation for tracking and localising objects.

- **Acoustic**: Both analogue and digital microphones are utilised in smart cities for security and surveillance purposes, as well as to monitor noise pollution.

- **Ultra Violet (UV)**: This sensor measures the UV radiation level, which is useful for healthcare, wellness, and smart city applications. Additionally, it provides information on the strength of sun exposure, which is important for smart farming and agriculture.

- **Ultrasonic**: Detecting obstacles and their distance is a useful function that can aid assisted living for visually impaired individuals and prevent robot collisions in smart industries.

2.3.4 Decision

I chose to use BT or WIFI as the method of communication between my bracelet and the Android phone. This is due to extensive options of BT or WIFI modules that are cheap and are a common protocol of communicating between a wearable and an Android phone.

3. Law, societal norms and moral principles

While AILED will definitely use a microphone and capture the environment audio, it does not record any conversation. The product just computes the audio and divides it into frequencies. We can also prove that AILED is complying with security issues as it does not contain any storage to store data for long-term and is not connected to WI-FI to store it in the cloud.

The software built uses open-source code from github which is free to use. The usage of any third party code is specifically mentioned in the software development section of this document.

4. Methodology

Agile is a lightweight software development method that aims to speed up the development process. It is an incremental approach which breaks down complex projects into smaller, manageable tasks. This strategy helps catch mistakes in the early phases of the project and allows easy, not time-consuming fixes. This document breaks down the development process into two significant sections, hardware and software, further broken down by designing, developing and testing. [9]

5. Requirements Engineering Summary

Requirements Engineering (RE) is a systematic and iterative process that aims to elicit, analyse, specify, validate, and manage the requirements of a software system. The RE process involves several phases, including stakeholder gathering, requirements elicitation, analysis, specification, validation, and management.

This section was discussed in more detail in another document named “Project Specifications Design and Prototype”. Below is a summary of that document and if someone wants to read the full document, he is free to do so.

After carefully choosing the stakeholders and giving them a form to elucidate their requirements, a list of requirements had been generated, classified as functional or non-functional and prioritised as essential, desirable and luxury respectively:

Requirement number	Requirement name	Requirement description	Functional or Non-Functional Requirements	Prioritisation on level
1	Writable bracelet	Have a big enough LEDS grid to write sentences using the application.	Functional	Luxury

2	Song recognition	Have the bracelet recognise songs and look on the internet for pre-created light patterns	Functional	Luxury
3	Preset creator	The application should allow users to create presets for certain frequencies/amplitudes	Functional	Luxury
4	Water resistance	The bracelet should be water resistance	Non-Functional	Luxury
5	Clock	The bracelet should be able to show the time by a click of a button for a short period of time	Functional	Luxury
6	Low battery warning	The bracelet should be able to vibrate when the battery is almost empty	Functional	Luxury
7	Feasibility study	Test whether is it possible to convert the audio input to colours in real time using a small microcontroller	Non-Functional	Essential
8	Sound-light accuracy	Have the bracelet accurately react to different frequencies and amplitude of the sound.	Functional	Essential
9	Android application communication	Have the bracelet communicate with an android application and be able to change the colour of the bracelet from the app.	Functional	Essential
10	Long lasting battery	Have the bracelet run for at least 5 hours	Non-Functional	Essential
11	Elegant and ergonomic design	Have the bracelet comfortable to wear and with an elegant design	Non-Functional	Desirable
12	Party mode	Be able to connect multiple bracelets and phones together and change the colour of them using only one “leader” phone.	Functional	Desirable
13	Brightness adjuster	Adjust the brightness of the bracelet to get a better battery life	Functional	Desirable

Source: [10]

6. Development of the hardware component of the AILED

In this section of the report we will discuss everything there is to discuss about the physical part of our project.

6.1 Functionality

With the help of our stakeholders we were able to create a list of project requirements which we discussed in more detail in the previous section. From those features, the following have a hardware impact:

- Writable bracelet: A matrix of individually addressable RGB LEDS
- Water resistance: Water resistance filament such as PETG
- Clock: A matrix of individually addressable RGB LEDS and a button
- Low battery warning: A vibration motor
- Sound-light accuracy: A powerful microcontroller and a microphone
- Android application communication: Bluetooth or WI-FI antennas
- Long lasting battery: A battery such as Li-ion battery
- Elegant and ergonomic design: Filament for 3D printer

The components specified above will provide a good base for AILED to be designed, developed and tested correctly. Due to the many components that need to be integrated, we will start with a selected few and improve from there in future iterations.

6.2 Design

The goal of this subsection is to explain the design decisions we have made and to get rid of design flaws that later would cost us dearly in terms of time.

For the first iteration, we went with only having an LED band that lights based on the audio input and designing a bracelet that would fit the following components: matrix of RGB LEDs, Raspberry PI Pico W micro-controller and a microphone.

6.2.1 Electronic components schematics

Before we look at fitting all the components in a 3D printer bracelet-shaped case, we need to see how all the electric components will communicate with each other. In figure TBD is a circuit diagram made using draw.io:

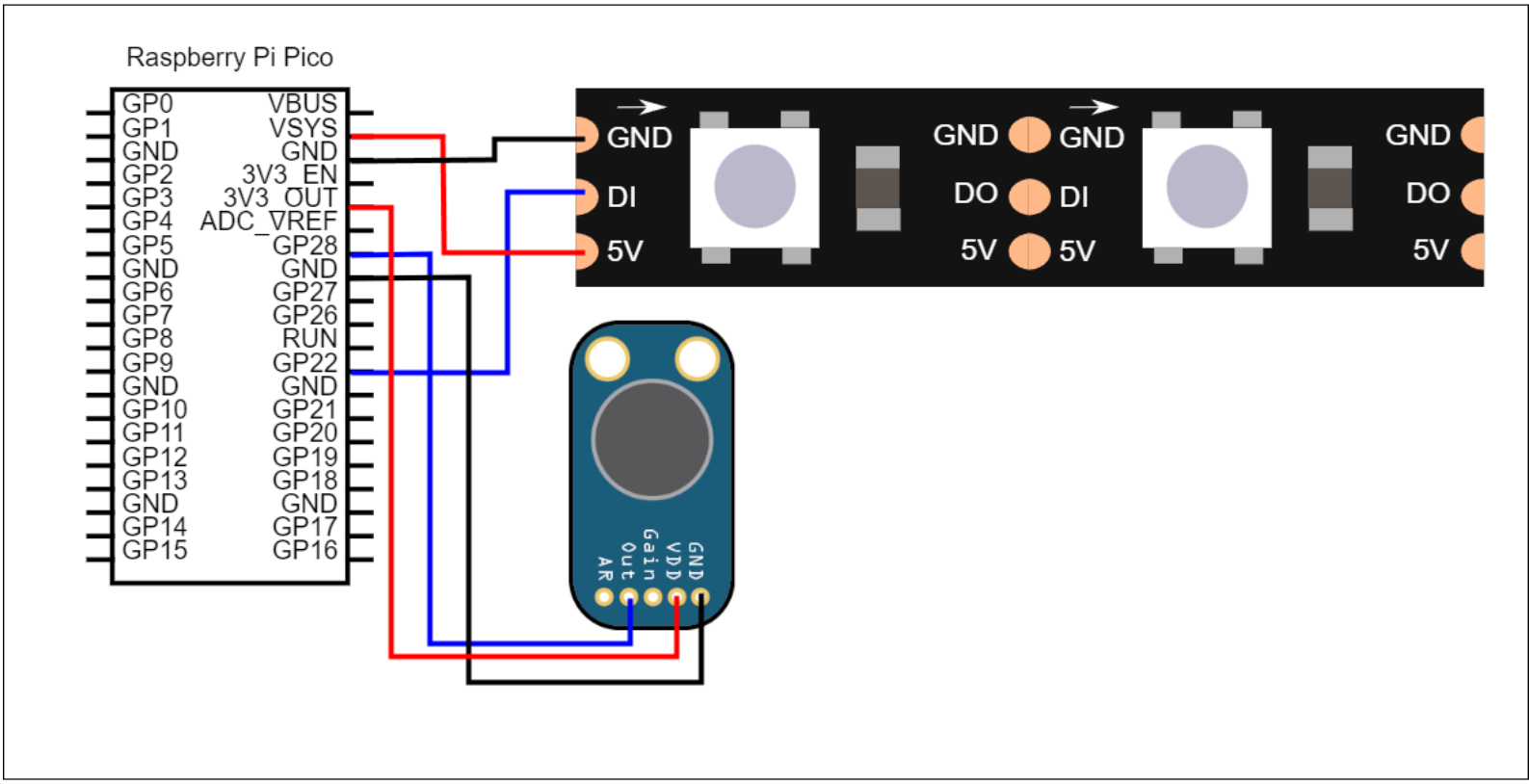


Figure 8. Circuit diagram for AILED

In the circuit mentioned, we used:

- A matrix of individually addressable WS2812B RGB LEDS

There are two types of LEDS that were considered for this type of project: SMD 5050 or WS2812B. SMD 5050 (see figure TBD) are RGB LEDS with 4 pins, one ground and 3 other pins for red, green and blue diodes respectively. By sending an electrical current between 0 and 5V in each diode, you can create any RGB colour.

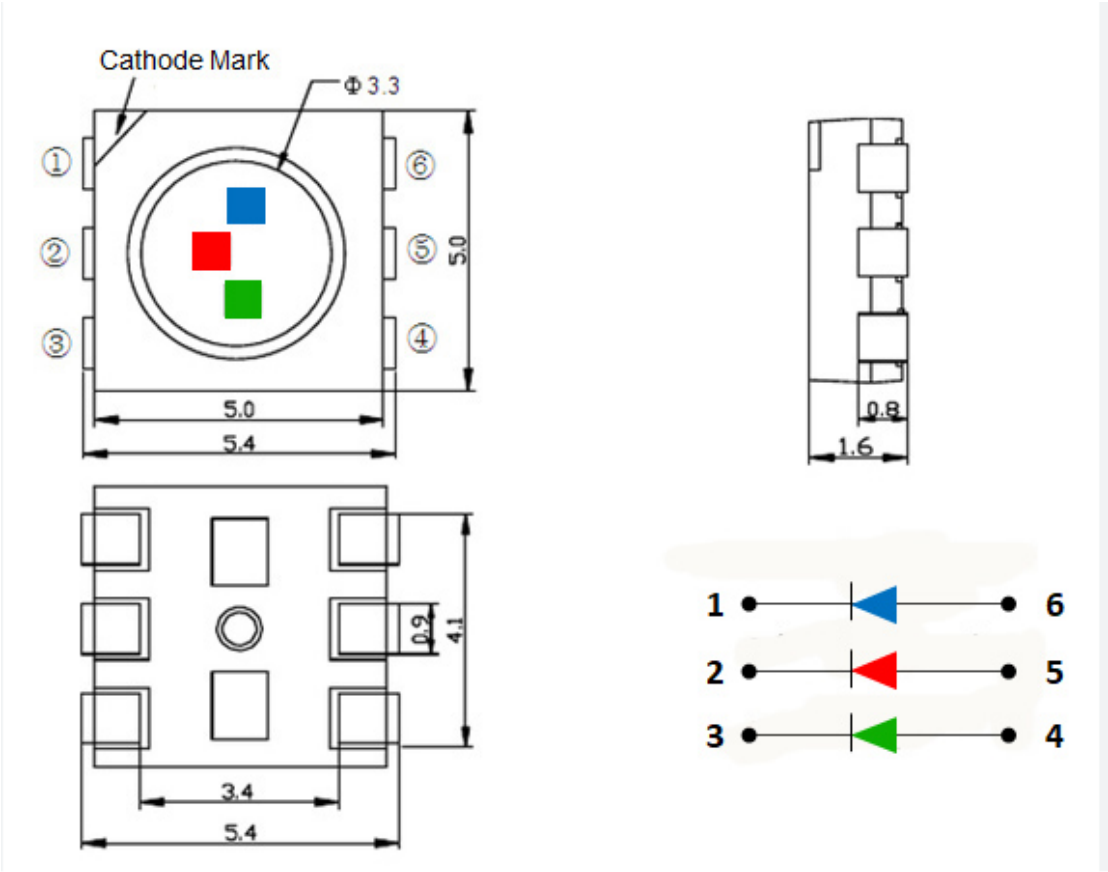


Figure 9. Datasheet for SM5050 LED showing the 3 individual diodes. [11]

The problem with SMD 5050 is that they are not individually addressable, meaning the LEDS could only have the same colour at a time.

WS2812B (see figure TBD) is basically a SMD 5050 LED with a smart control integrated circuit (IC). It has 4 pins, one for 5V input, one for ground and two for data. Data input to receive the data and data output to further pass the data. By sending an electric current with specific timings, you allow the controller to light up the LED with any RGB value. Also, the IC allows a LED matrix to have LEDS individually coloured [13]. This is especially important if we want to write sentences on our bracelet.

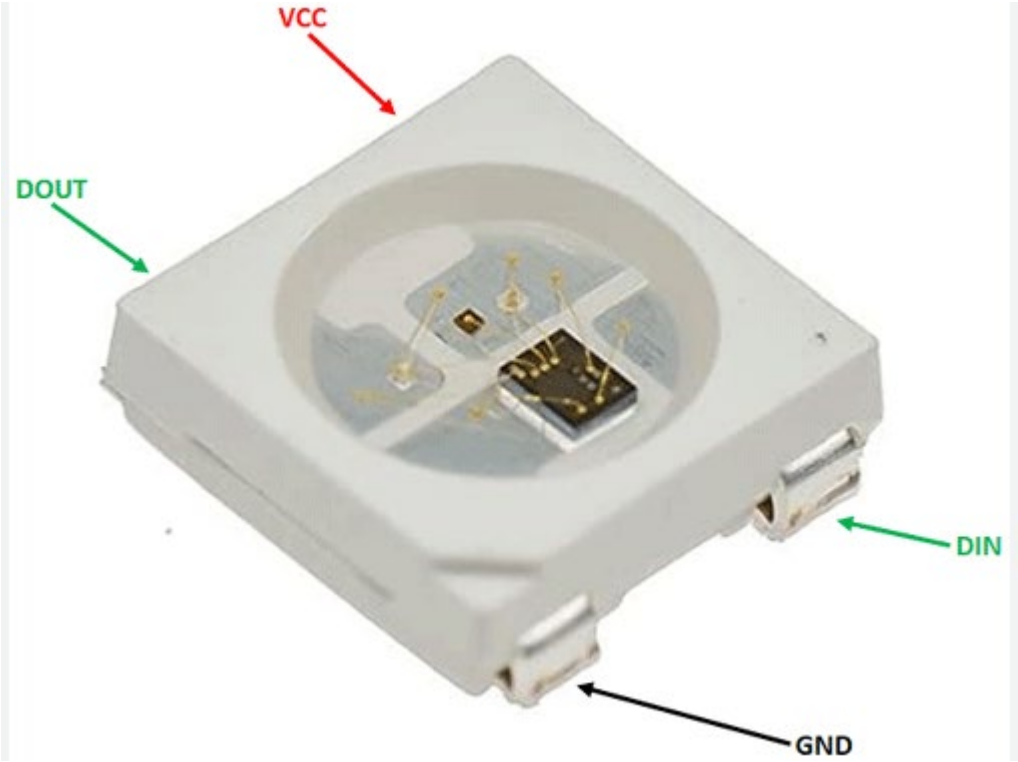


Figure 10. Picture with a WS2812B LED where you can see the smart controller. [12]

- MAX9814 microphone with Auto Gain Control

Three types of microphones were considered for AILED: LM393, MAX4466 and MAX9814.

LM393 is the standard microphone for DIY projects. It is a 3.3 - 5V microphone that can hear frequencies between 100 and 10,000 Hz. You can see from Figure TBD that the microphone's size is quite large. Also, to get a proper sound accuracy, we need to recognise frequencies between 100 and 20,000 Hz at least. [14]

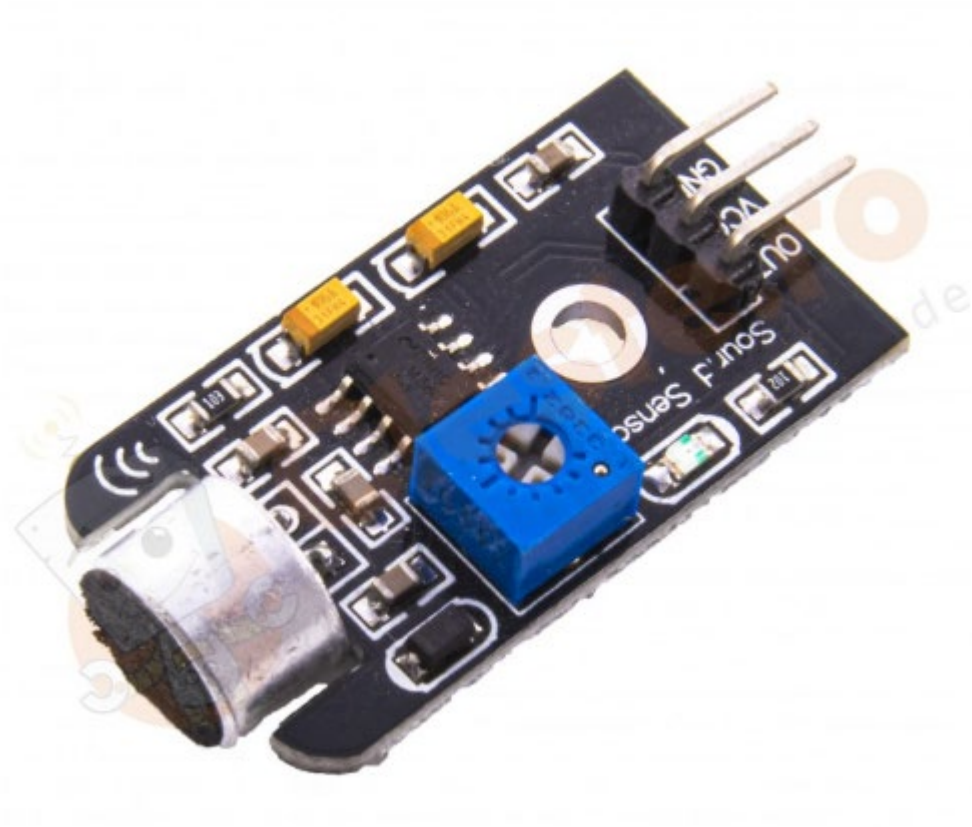


Figure 11. LM393 microphone [15]

MAX4466 is a microphone of a smaller size (see figure TBD) that can hear frequencies of up to 20,000 Hz. It also works at 3.3 - 5V and has adjustable gain [16]. Adjustable gain means that you manually choose, by screwing a bolt tighter or looser, the sensitivity of the microphone to the sound. The problem with having an adjustable gain is you never know how loud the environment will be. If you are at a festival, the music will be loud enough for one gain setting but too quiet environmental music in a room at home.

Figure 12. MAX4466 microphone [17]

And finally, MAX9814 (see figure TBD), the option we went with. It is a 3.3 - 5V microphone that can recognise frequencies between 20 and 20,000 Hz. It also has auto-gain meaning that it will automatically adjust the microphone sensitivity based on the environment [18]. More information is found in the development section.

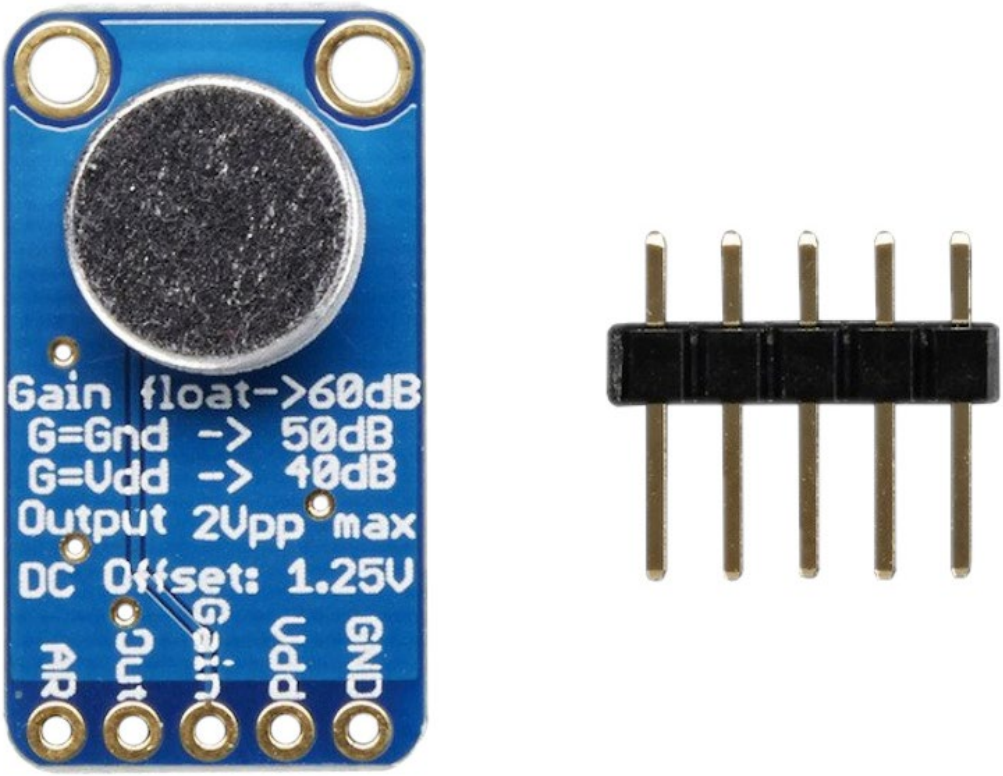


Figure 13. MAX9814 microphone with auto-gain [19]

- A Raspberry Pi Pico W micro-controller

For computing, we went with the Raspberry Pi Pico W micro-controller. It is a 5V, small, powerful and cheap microcontroller that can do the conversion from audio input to frequencies, mapping to colours and sending it to the LEDS. It also has bluetooth and WI-Fi antennas, crucial for communicating with the android application. It has 26 GPIO pins for digital input/output and pulse width modulation (PWM), crucial for communicating with the LEDS. An analog-to-digital (ADC) converter is also present for getting the analog signal of the microphone and digitising it for data processing. And finally, it supports Direct Memory Access (DMA) for quick data transfers. [20]

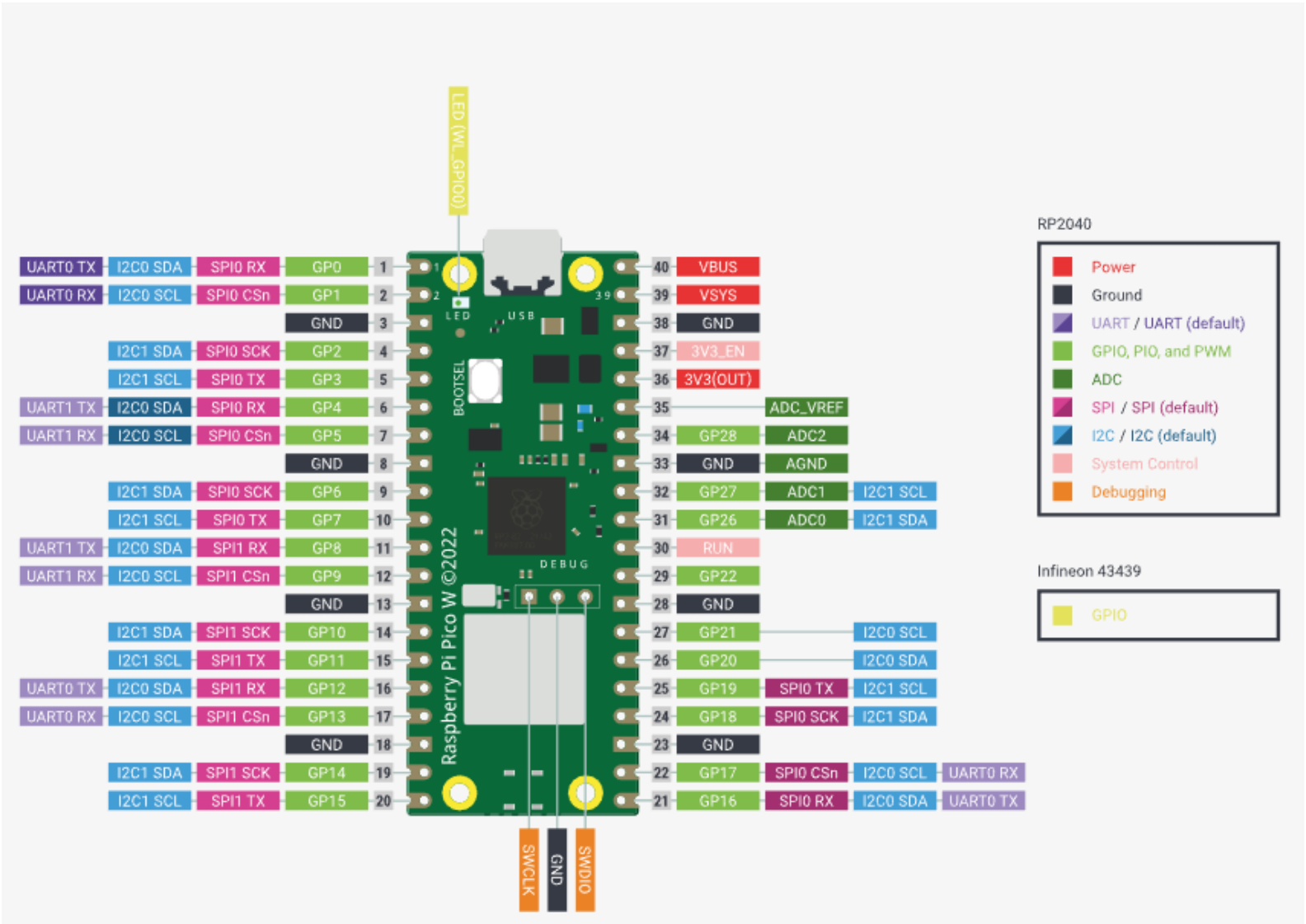


Figure 14. Datasheet for Raspberry Pi Pico W [20]

With the designed circuit, we have a strong base for prototyping and further improvements.

6.2.2 3D printed Bracelet

Now with knowing the specific components we will have in the bracelet, we can start 3D modelling the bracelet. Below is the workplace with all the models generated for AILED (Figure TBDs). The model was made using Thinkercad. More information about the design is found in the development section.

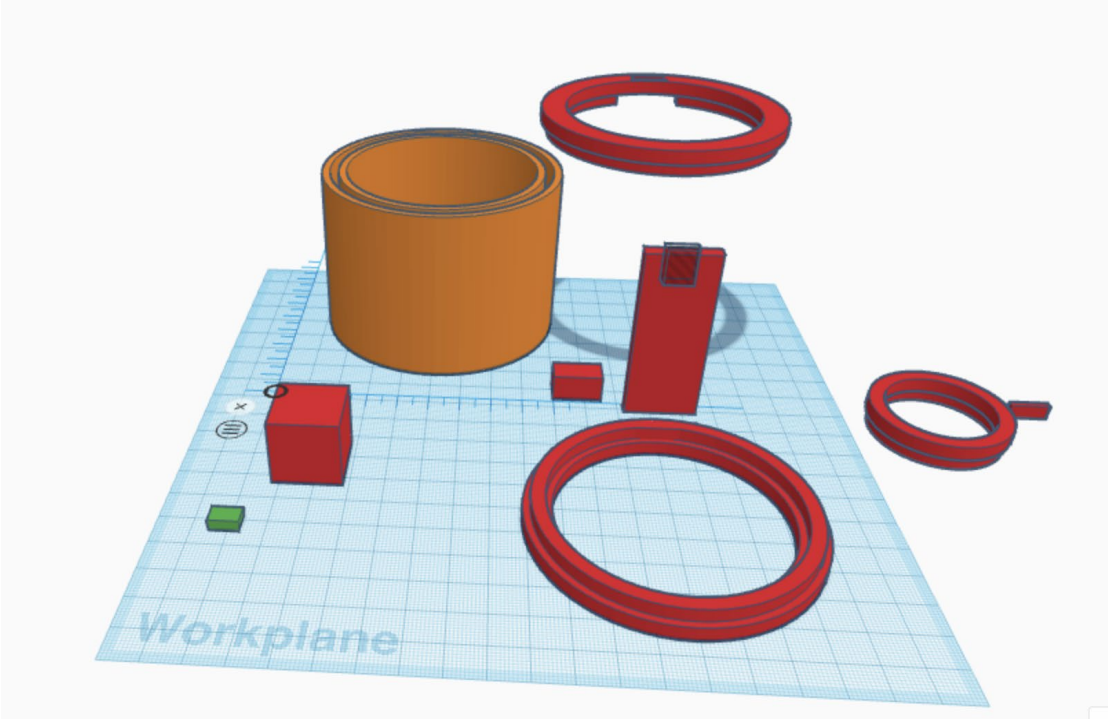


Figure 15. Workplace on Tinkercad with all the models

6.3 Development

The development of a working prototype is divided in 3 main parts: soldering headers on the components for easy wiring and testing, wiring electronic components as per the electronic diagram (Fig. TBD from 6.2.1) and printing the bracelet case.

6.3.1 Soldering

Soldering is the process of bonding different types of metals together by melting solder. Solder is a metal alloy usually made of tin and lead which is melted using a hot iron. The iron is heated to temperatures above 300 degrees celsius which then cools to create a strong electrical bond [https://www.twi-global.com/technical-knowledge/faqs/what-is-soldering]

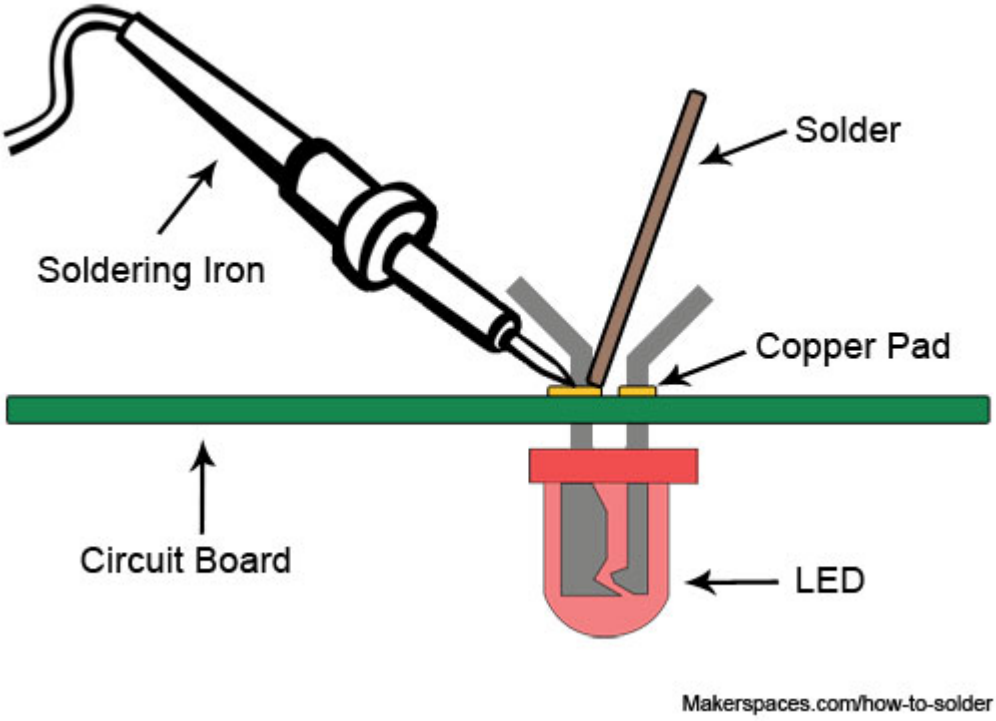


Figure 16. Process of soldering [21]

Below is a list of components used:

- Conical Tip: a tip used for precision soldering. It is used for soldering small electronic components.

Figure 17. Actual canonical tips used

- Sponge: Used for cleaning the oxidation formed on the tip.

Figure 18. Actual sponge used

- Soldering Iron with stand: Used to prevent the tip from touching flammable materials while hot and not in use.

Figure 19. Actual soldering iron with stand used

- Solder: Metal Alloy used for soldering the connections

Figure 20. Actual solder material used

The result is 2 microphones, one LED band and a microcontroller. (Figure TBD)

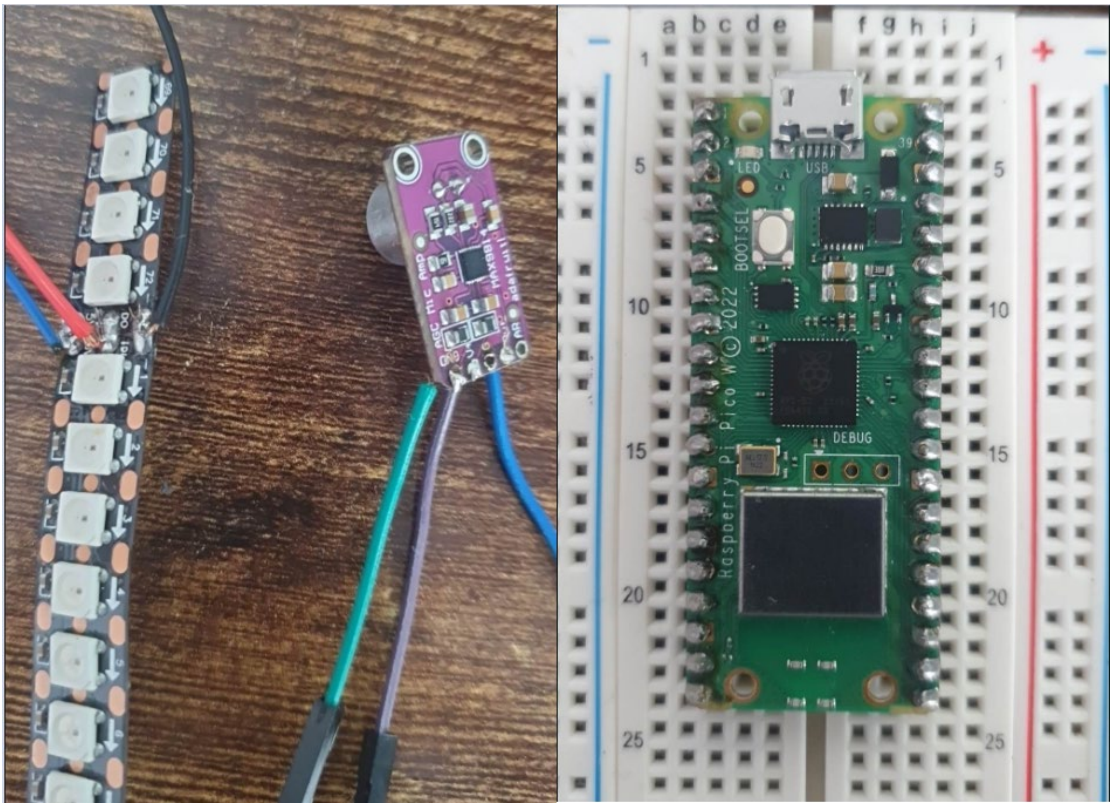


Figure 21. Soldered components

6.3.1 Wiring electronic components

After adding headers to the required components, we can attach them on a breadboard. The breadboard, also named a solderless board, is a construction base used for creating prototypes of electronic circuits. Instead of soldering the components, jumpers are used. Below is the actual prototype used for writing the software of AILED.

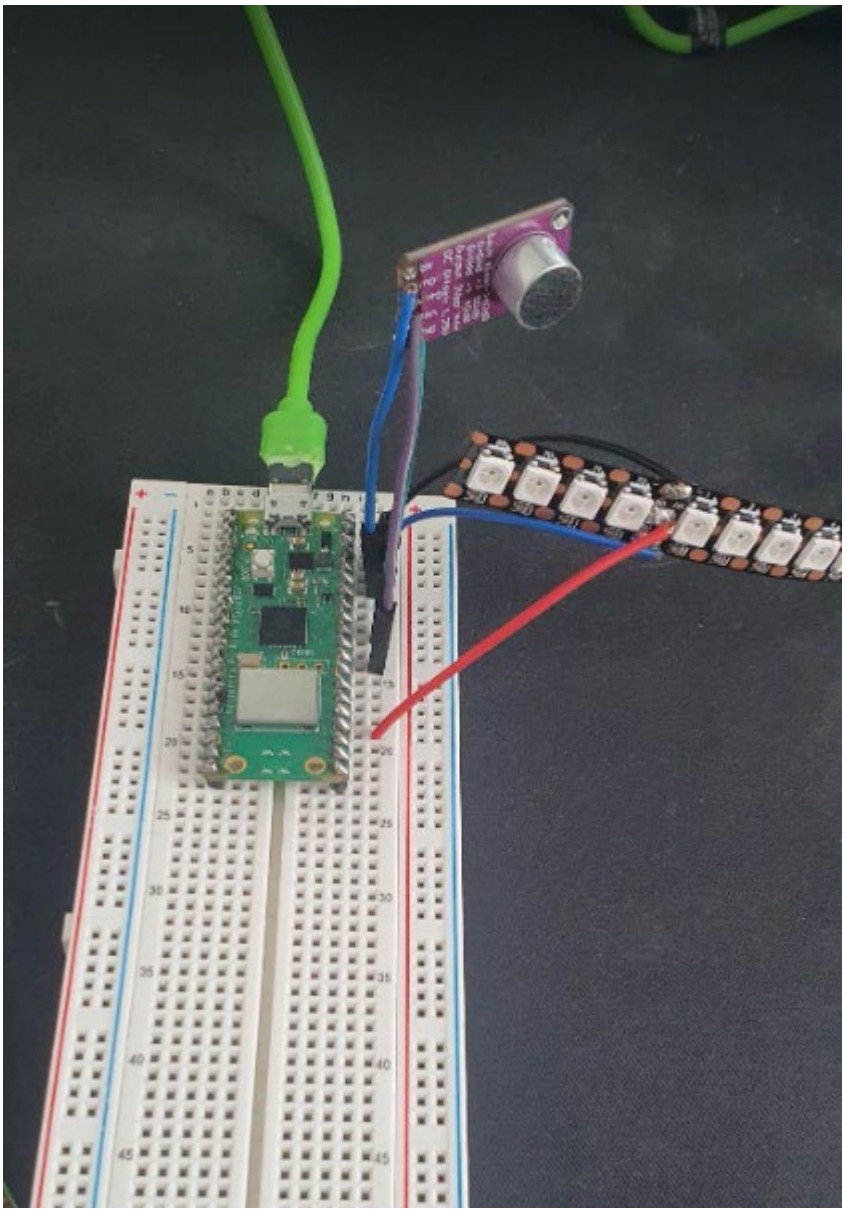


Figure 22. Actual prototype used

6.3.2 3D Printing

This was one of the most time-consuming parts of the project as it required a lot of waiting for printing, trial and error and debugging issues. It is also where it is described the development process of the 3D model and the printing.

6.3.2.1 Developing the model

The goal of the first 3D model was to hold the microcontroller in a special compartment and have enough space to fit the LED matrix, a microphone and a hole for the micro-usb for powering up the microcontroller as we will later look into attaching a standalone battery.

Tinkercad is a beginner friendly and also a web based 3D modelling software. It offers you models for basic shapes such as squares, sphere, cones:

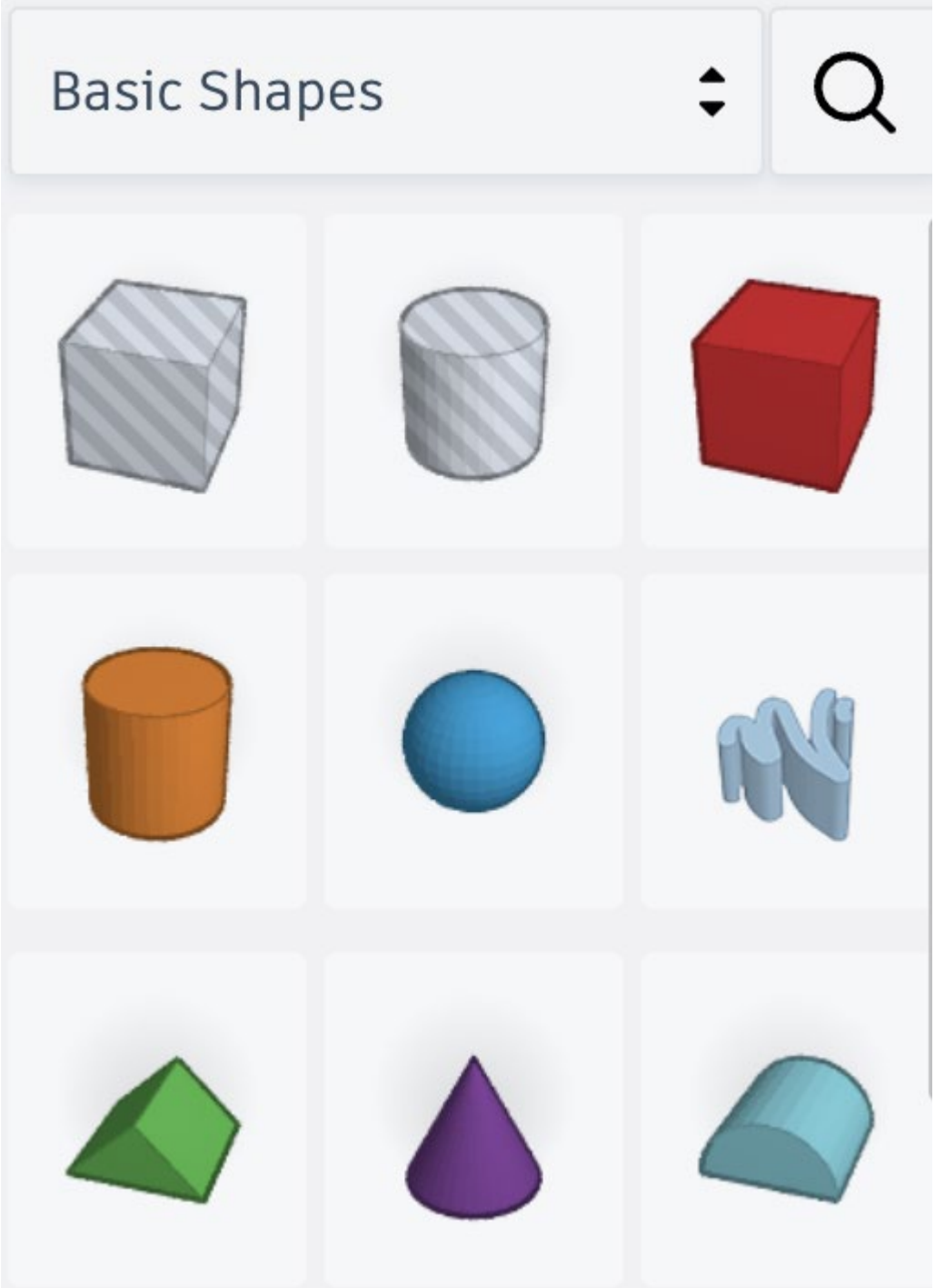


Figure 23. Basic shapes in Tinkercad

As seen in figure TBD above, the shapes can be considered solid object but also holes (the first two shapes). By grouping the solid object and the hole shape, we can create more complex shapes.

First we measured the microcontroller and created a basic rectangle box with its dimensions, this way we can check if the compartment we will make will fit or not.

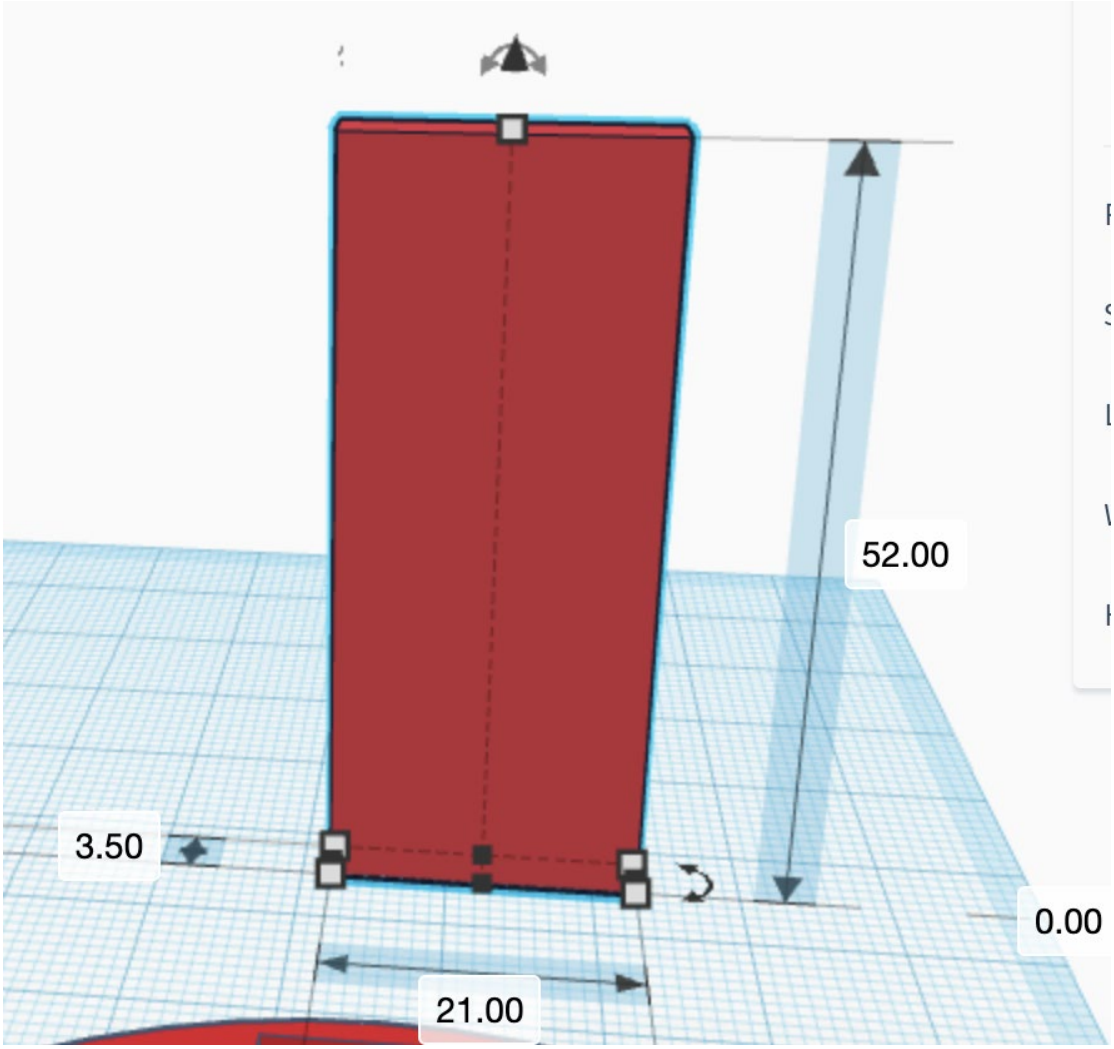


Figure 24. Microcontroller model with its real dimensions in millimetres

We then proceeded with creating a bracelet shaped model. We started with a cylinder of 80 mm and made a hole of 68 mm diameter in it. The hole will definitely not fit everyone without dividing the bracelet in 2 and adding a locking mechanism with a hinge (this can be added in a second iteration) but it is the bare minimum to allow a 16 mm hole to fit all the components with this design. After also generating a cylinder with the 16mm and combining it with the microcontroller shape, we made the hole shape inside the bracelet. By combining all together, we have a very basic bracelet shape:

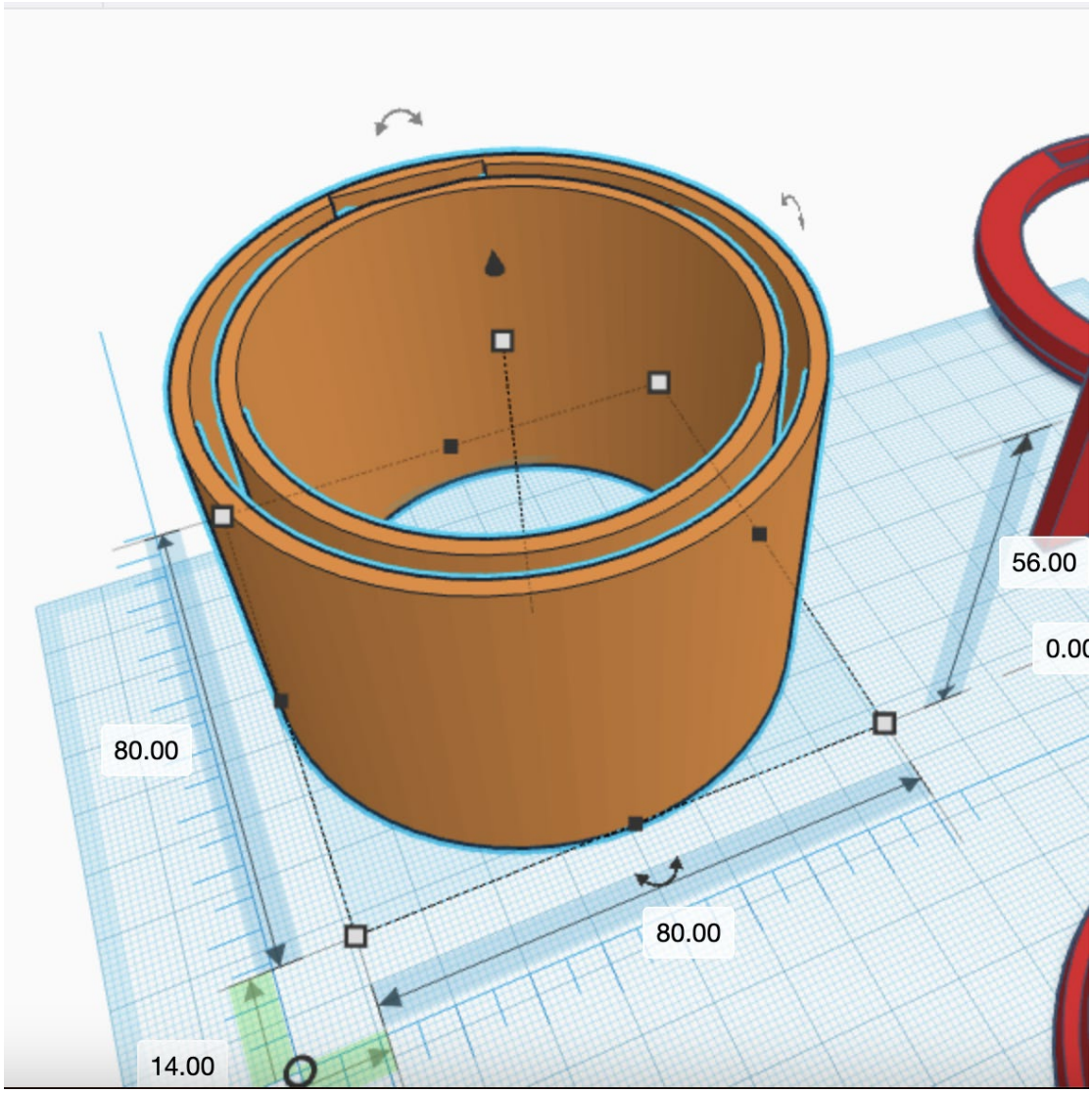


Figure 25. Basic shape of a bracelet

Until now we have two distinct cylinder shapes, one being 80x80x56 mm and one around 78x78x56 mm. Next, we need to create two caps which will combine the cylinders together:

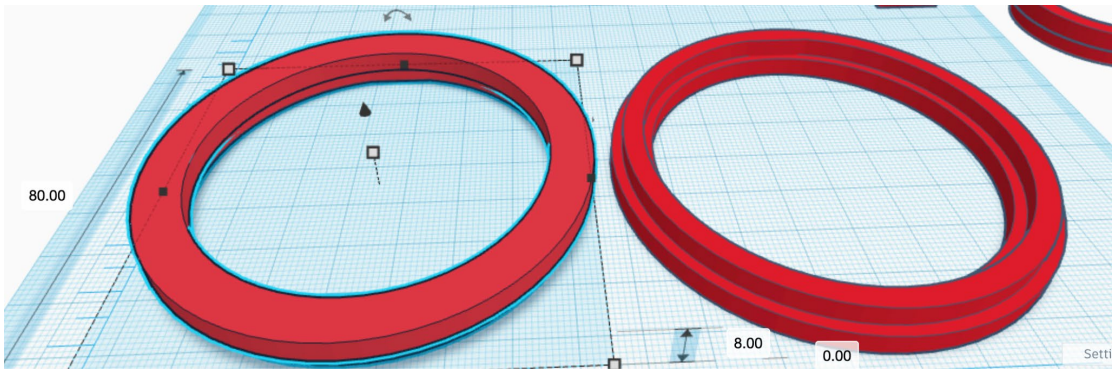


Figure 26. Cap model that holds the cylinders together

The cap has around the same dimensions as the cylinders, minus 1 mm to allow some wiggle room when we fit everything together. Also it has 8 mm in height, 4 mm being the exterior cap and 4 mm to be between the cylinders.

And lastly, in order to power up the AILED, we added a micro-usb hole in one of the caps:

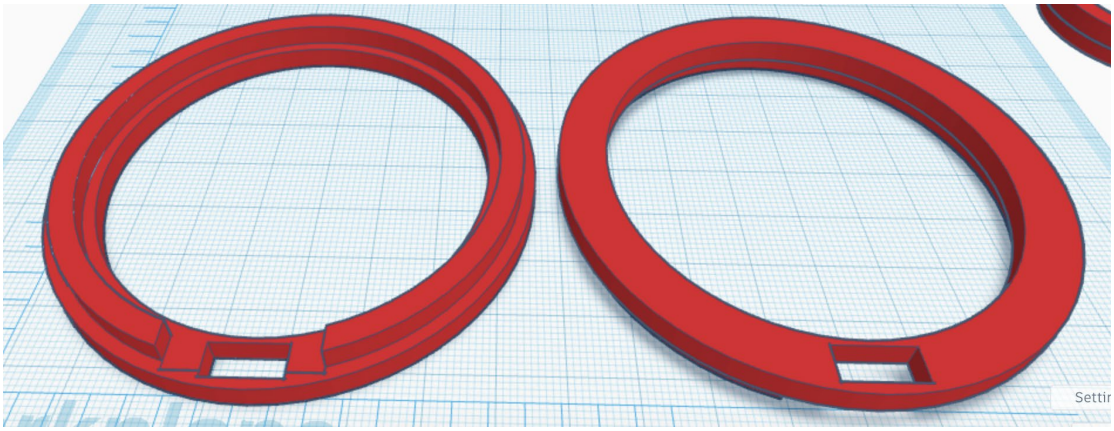


Figure 27. Cap with hole for micro-USB

All the 3D models can be found in the GitHub repository from the appendix.

6.3.2.2 Printing the model

For 3D printing the models, we used Ender 3 V2, an entry-level 3D printer and two types of PLA filaments, grey and transparent coloured. The reasons we chose this particular model are:

- Cost: This particular model has a low cost due to its manual levelling (more on that later) and it comes semi-assembled.
- Popularity: This model is one of, if not the most popular model of 3D printers. Because of that, for any issues we might encounter (and we did), we can find a solution on forums.

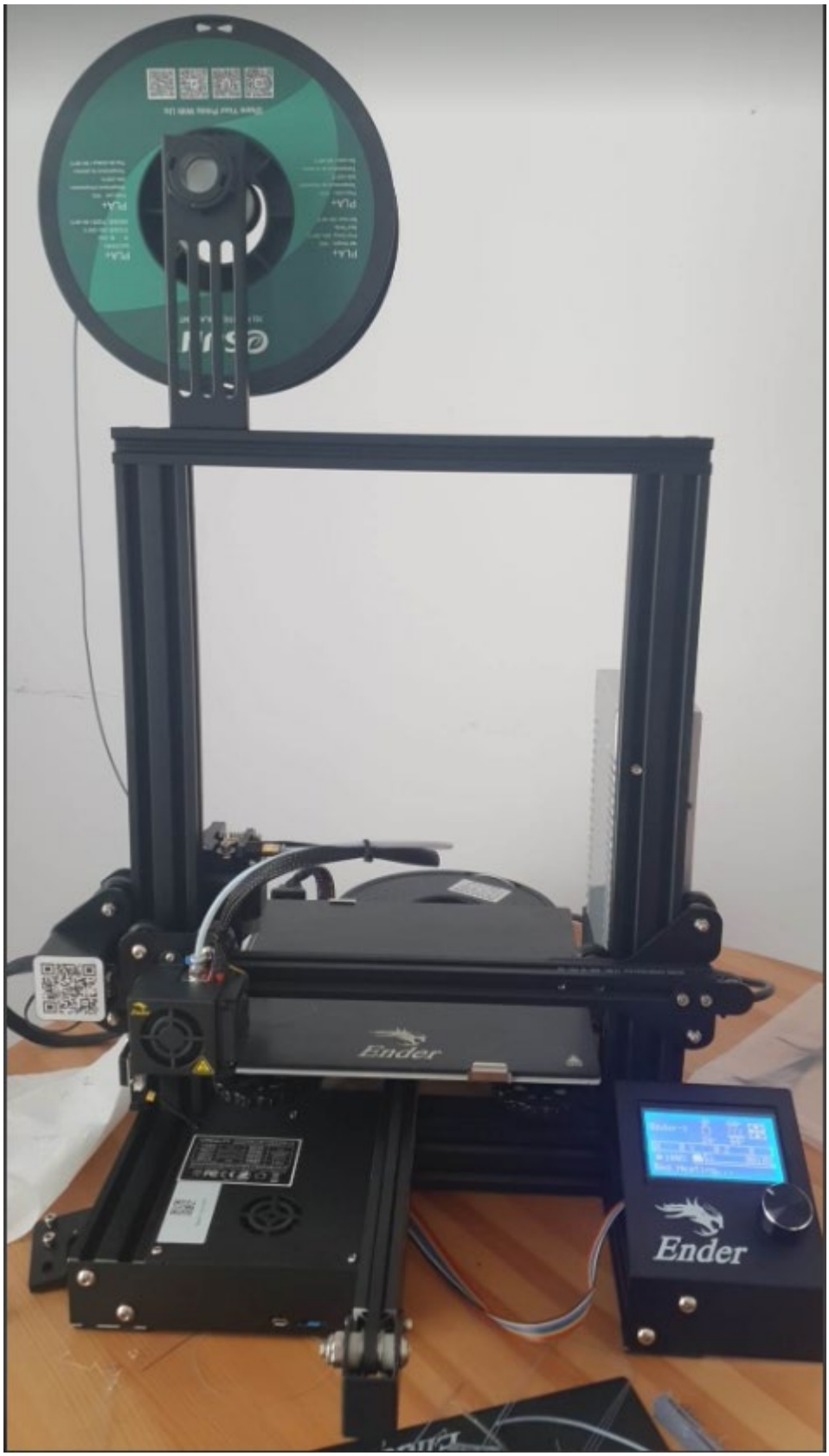


Figure 28. Actual 3D printer used

In order to print a 3D model, we must use slicing software, which is used for the following:

- Choosing the temperature of the bed and nozzle of the 3D printer has to reach (this is usually based on the temperatures recommended by the filmanet's manufacturer)
- The percentage and shape of the infill. This affects the sturdiness of the object. We can leave the object hollow inside or choose to fill it with material at a specified percentage.
- Supports. We can choose whether we use or not supports. The supports helps with printing models that have parts at more than 45 degrees celsius (Letter T, for example)
- "Slicing" the model in levels and explaining the 3D printer, in exact steps and coordinates, how to move to print the object. It is called slicing because a 3D printer prints in layers, from bottom to top.

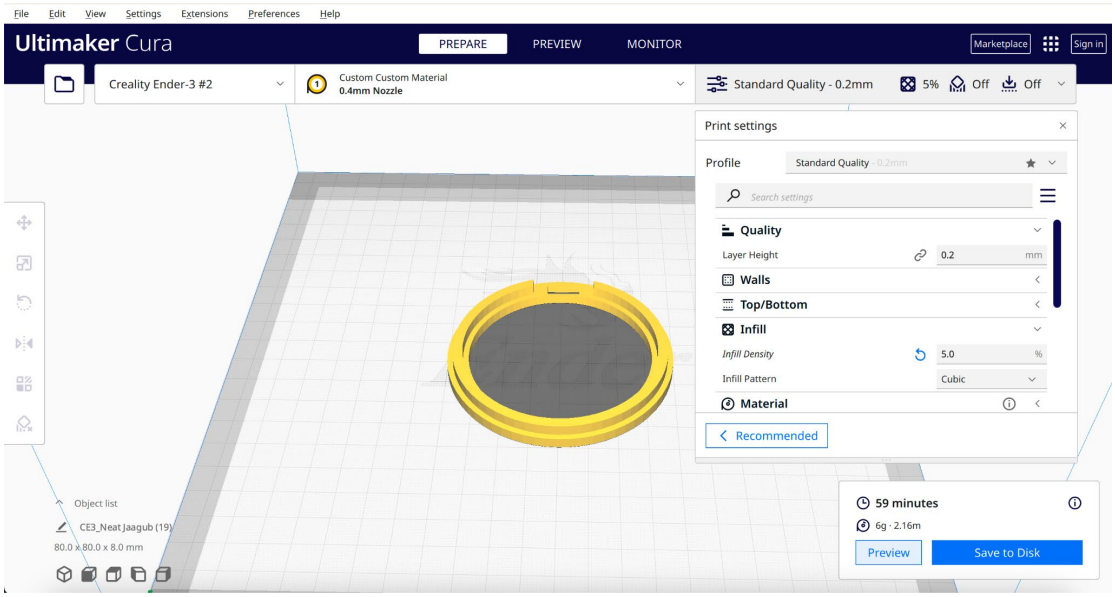


Figure 29. Interface of Cura

Below is a GIF that shows how the 3D printer moves while printing and all the models designed for AILED printed:

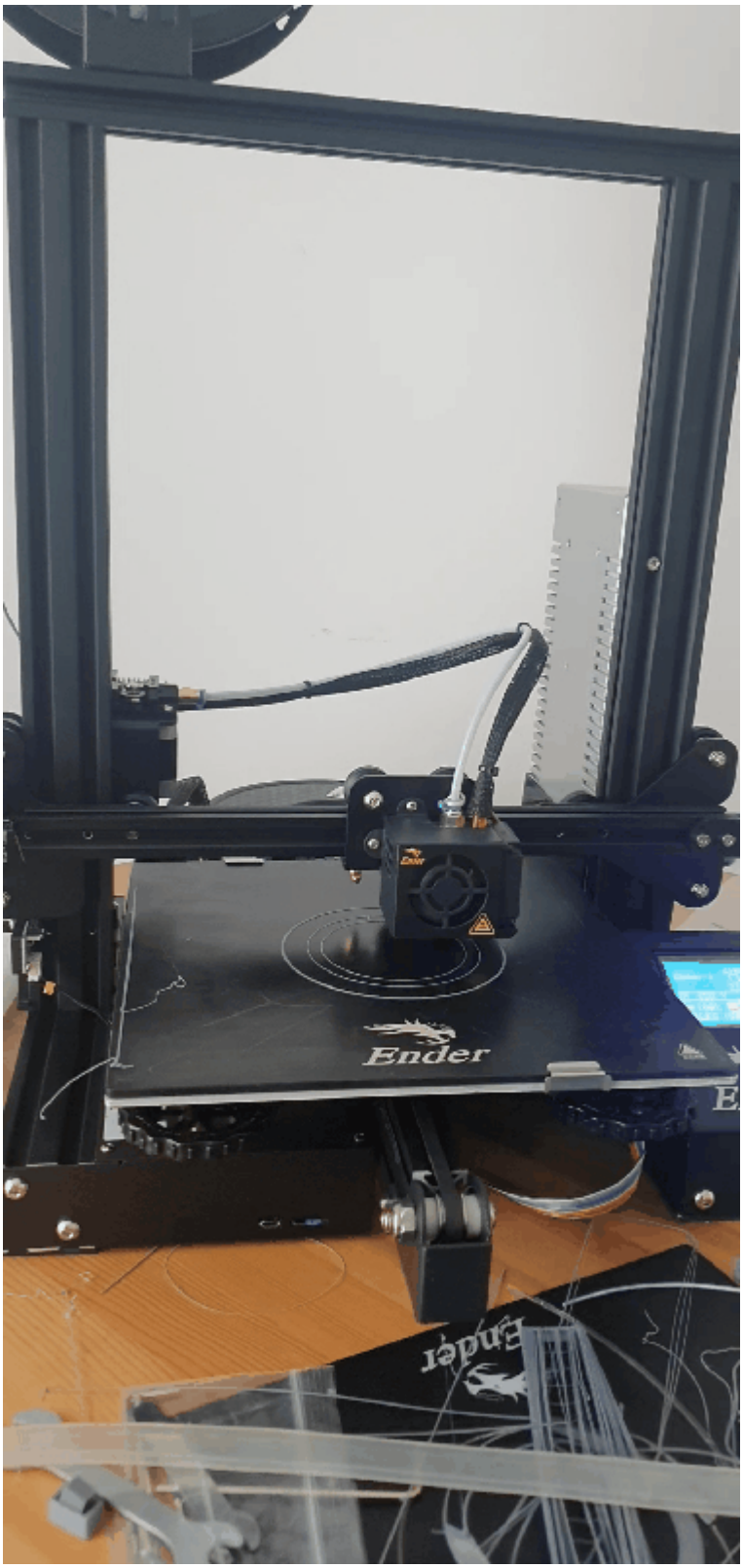


Figure 30. GIF with working 3D printer

The final result:

Figure 31. 3D printed components used for AILED

6.4 Testing

Electronic testing was mainly achieved using the breadboard and the components with header pins. Very simple code was added to see if the wiring is correct and no short-circuit is created.

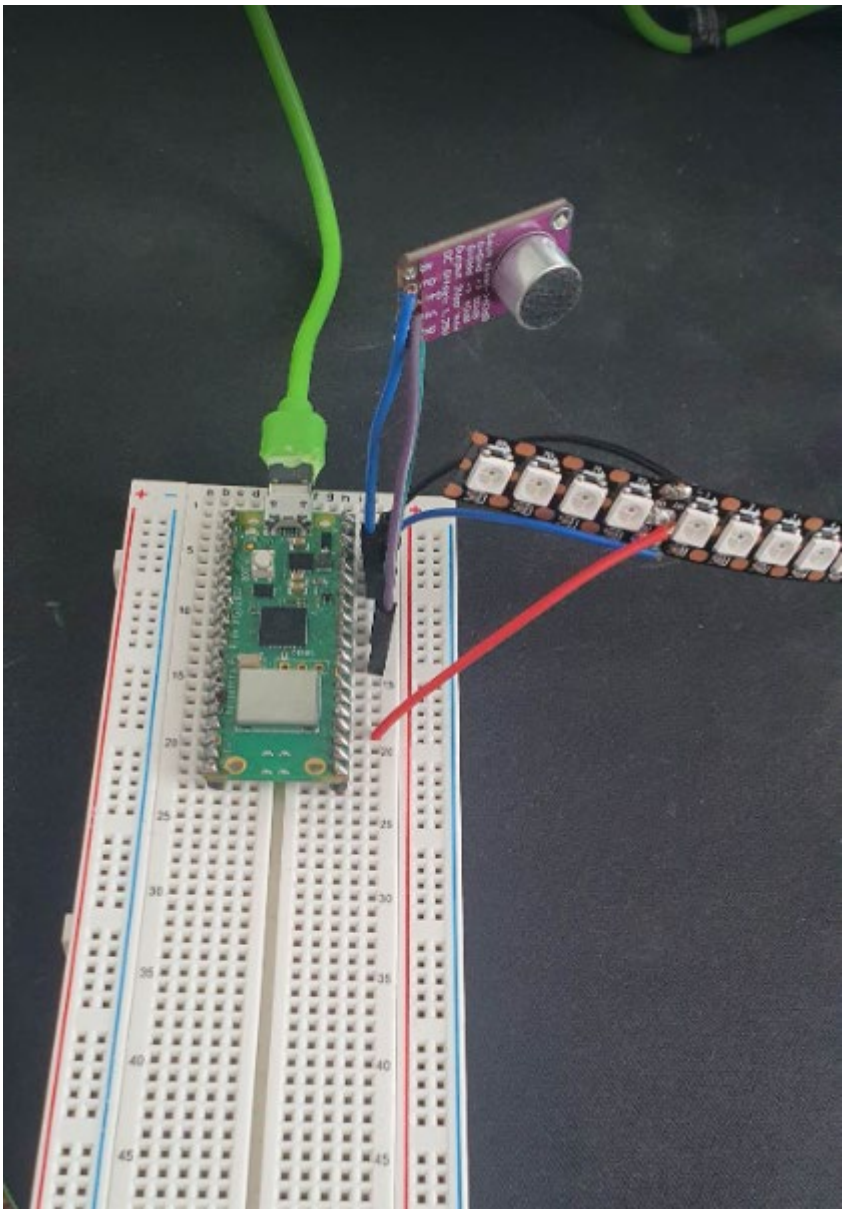


Figure 32. Breadboard with a microcontroller, microphone and LED band with headers.

For 3D models, multiple, smaller sizes iterations were made. By making smaller sized iterations we could see if the ratios were right without having to wait 12 hours (the full length of printing the bracelet).

Figure 33. Miniature sized components made in 25 minutes instead of 12 hours.

7. Development of the software component of the AILED

This section of the report thoroughly discussed the software component of the AILED project. It documents the development of the digital interface at the heart of the project.

7.1 Functionality

With the help of our stakeholders we were able to create a list of project requirements which we discussed in more detail in the requirements section. From those features, the following have a software impact:

- Writable bracelet: Be able to send data to individual LEDS via phone.
- Song recognition: Integrate a song recognition solution.
- Preset creator: Be able to create patterns for the light show via phone.
- Clock: Be able to send data to individual LEDS so AILED displays the time.
- Low battery warning: Be able to tell the percentage of the battery.
- Sound-light accuracy: Be able to recognise frequencies and amplitude of the environment.
- Android application communication: Be able to interact with the bracelet using a phone via bluetooth.

Not all features were pursued due to the time constraints and prototyping nature of the project.

7.2 Design

The goal of this subsection is to explain the design decisions we have made and to get rid of design flaws that later would cost us dearly in terms of time.

For the first iteration, we went with implementing an algorithm that lights an LED band based on the audio input and an Android application that communicates with the prototype via Bluetooth in order to light the bracelet.

7.2.1 Choice of programming language

Lighting an individual addressable multi-colored LED can be quite daunting at first, especially using Raspberry Pi Pico controller and C programming language. For the programming language we had two choices to choose from: Python and C. Python is definitely much easier to use but is slow. C is harder but much faster, being known as a low-level programming language. For example, in figure TBD is how you would light a single coloured LED using Python and in figure TBD is how someone would do it in C:

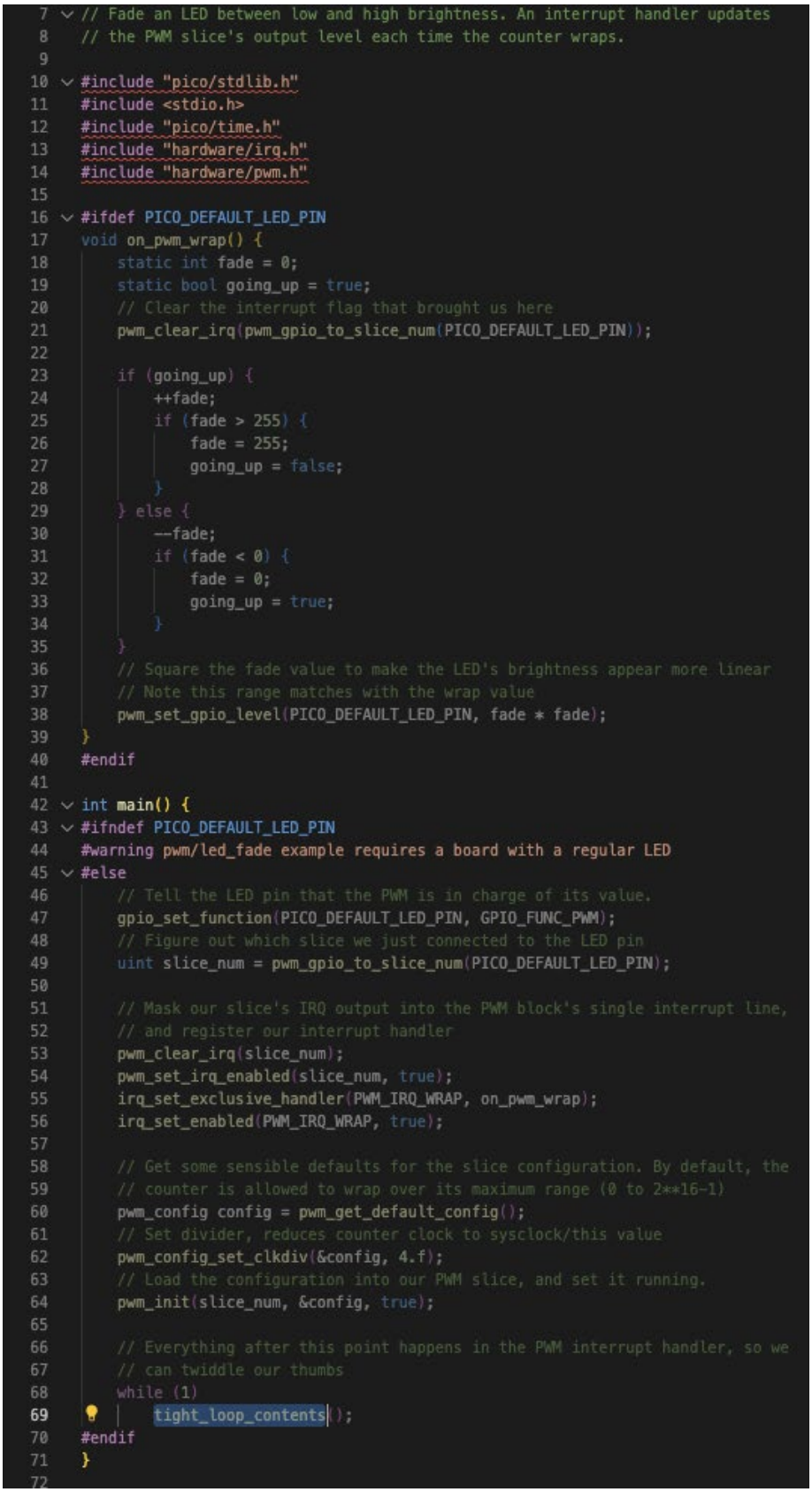
```python
from machine import Pin, PWM
from time import sleep

pwm = PWM(Pin(15))

pwm.freq(1000)

while True:
    for duty in range(65025):
        pwm.duty_u16(duty)
        sleep(0.0001)
    for duty in range(65025, 0, -1):
        pwm.duty_u16(duty)
        sleep(0.0001)
```

Figure 34. Fading an LED in Python [22]



```c
// Fade an LED between low and high brightness. An interrupt handler updates
// the PWM slice's output level each time the counter wraps.

#include "pico/stdlib.h"
#include <stdio.h>
#include "pico/time.h"
#include "hardware/irq.h"
#include "hardware/pwm.h"

#ifdef PICO_DEFAULT_LED_PIN
void on_pwm_wrap() {
    static int fade = 0;
    static bool going_up = true;
    // Clear the interrupt flag that brought us here
    pwm_clear_irq(pwm_gpio_to_slice_num(PICO_DEFAULT_LED_PIN));

    if (going_up) {
        ++fade;
        if (fade > 255) {
            fade = 255;
            going_up = false;
        }
    } else {
        --fade;
        if (fade < 0) {
            fade = 0;
            going_up = true;
        }
    }
    // Square the fade value to make the LED's brightness appear more linear
    // Note this range matches with the wrap value
    pwm_set_gpio_level(PICO_DEFAULT_LED_PIN, fade * fade);
}
#endif

int main() {
#ifndef PICO_DEFAULT_LED_PIN
#warning pwm/led_fade example requires a board with a regular LED
#else
    // Tell the LED pin that the PWM is in charge of its value.
    gpio_set_function(PICO_DEFAULT_LED_PIN, GPIO_FUNC_PWM);
    // Figure out which slice we just connected to the LED pin
    uint slice_num = pwm_gpio_to_slice_num(PICO_DEFAULT_LED_PIN);

    // Mask our slice's IRQ output into the PWM block's single interrupt line,
    // and register our interrupt handler
    pwm_clear_irq(slice_num);
    pwm_set_irq_enabled(slice_num, true);
    irq_set_exclusive_handler(PWM_IRQ_WRAP, on_pwm_wrap);
    irq_set_enabled(PWM_IRQ_WRAP, true);

    // Get some sensible defaults for the slice configuration. By default, the
    // counter is allowed to wrap over its maximum range (0 to 2**16-1)
    pwm_config config = pwm_get_default_config();
    // Set divider, reduces counter clock to sysclock/this value
    pwm_config_set_clkdiv(&config, 4.f);
    // Load the configuration into our PWM slice, and set it running.
    pwm_init(slice_num, &config, true);

    // Everything after this point happens in the PWM interrupt handler, so we
    // can twiddle our thumbs
    while (1)
        tight_loop_contents();
#endif
}
```

Figure 35. Fading an LED in C [23]

Both the Python and C version of fading an LED use 1 PWM pin. PWM or Pulse Width Modulation is the standard technique of getting analog results with digital means. This can translate in instead of just being able to turn off an LED when you give it 5V and off when you give it 0V, you can give more granular values for voltage. This results in different brightness for the LED. This will be particularly useful using RGB LEDs as different voltages translate into different colours.

7.2.2 Algorithm for sound-light accuracy

The functionality of the algorithm can be divided in 4 parts: getting the environment audio, computing the audio signal, generating the colours and lighting up the AILED. Below each part is described in great details:

7.2.2.1 Getting the environment audio signal

The purpose of this step of the algorithm is to take the analog input generated by the microphone and convert it to digital values that the microcontroller can understand. Fortunately, Raspberry Pi Pico W has a built-in analog to digital converter (ADC) and saves us from looking at third party ones.

7.2.2.2 Computing the audio signal

With the digital signal captured by the microphone, we are going to apply an algorithm named Fast-Fourier Transform (FFT) which will take the signal from its original domain, time, and convert it to frequency domain. This way, we will be able to know, with approximation, the most predominant frequency in the audio over a period of time. For any implementation of FFT, we need to take the following parameters into account [24]:

- FSAMP: The upper bound for the range of frequencies we will be computing for, the lower bound being 0. This will be set to 50,000 as FFT asks for double the highest frequency we want to recognise, which is 25,000, the highest frequency used in music.
- NSAMP: Number of samples that are taken in consideration for computing. The more samples you use for computing, the better accuracy we can have for the predominant frequency, but it will take longer and more memory. The number needs to be a power of 2 as it allows a more efficient computation with the FFT algorithm. Also the FSAMP and NSAMP impact the accuracy of our frequency detection. For example, by having FSAMP 20,000 and NSAMP 1024, FFT would group the frequencies in bins of 20hz (20,000 divided by 1024). If the actual frequency is 20, it will display 20. But if the frequency is 21, it will display 40. This will be calculated while developing as it depends on when the computing “feels” real time and accurate and when is too slow and inaccurate.

After FFT, we will have an array of frequencies and their magnitude. Based on that we can choose the most predominant frequency and do something accordingly.

7.2.2.3 Generating the colours

The generating colours step is quite straightforward and where we can tweak based on preferences. The idea will be of having ranges of frequencies for which we will return an RGB value. An RGB value is a pair of 3 values, each between 0 and 255. Combining these values, we can generate any colour! For example, if the most

predominant frequency is 440 Hz, we want to have the AILED lighting up RED, so we will generate the RGB value (255,0,0).

That RGB value will then be sent to the LEDS.

7.2.2.4 Lighting the LEDS

After being able to light up a single colour and then a multi-colour LED, we will try lighting up an LED band containing multiple WS2812s . As described in section 6.2.1, WS2812B is an individually addressable LED with 4 pins, one for ground, one for 5V, one for data input and one for data output. Based on the datasheet of the WS2812B [13], there are two main concepts to have in mind:

- Composition of the data

Compared to a normal RGB led where you would have individual pins for red, green and blue respectively, WS2812B has only one pin. Figure TBD tells us that we have to send chunks of 24 bits data via the PWM pin, the first 8 bits being for the green colour, next 8 bits for the red colour and the last 8 for blue.

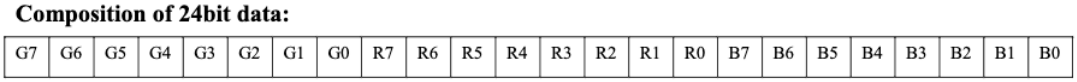


Figure 36. Composition of 24 bits data

- Data transmission method

Multiple WS2812B LEDS linked in series would communicate with each other with something they named cascade method (Figure TBD). For example: For 3 leds linked and 72 bits of data were sent via the data pin, 24 bits will be read by the first led and the rest of 48 bits will be passed on the next leds. The second led will take again 24 bits and pass the last 24 bits on the last led.

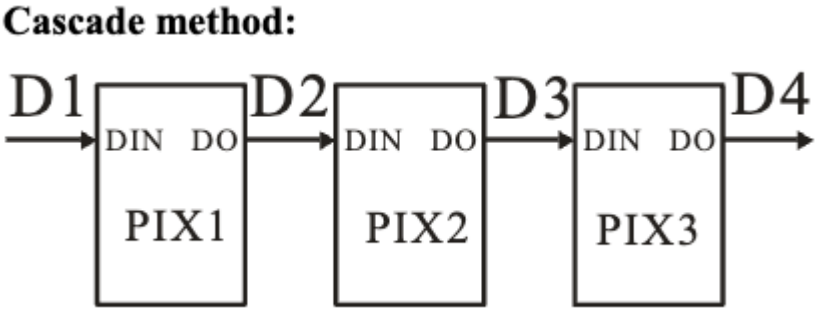


Figure 37. Cascade method for passing data on linked WS2812B LEDS [13]

Now that we know how data would be passed in an led band, we have to understand how the band knows when it should continue passing the data to the next consecutive led or go back to the first led with new data. Here they present a concept named “reset

code”. Reset code means a “window” of time bigger than 50 μs (microseconds) which tells the band to go back to the first led. (Figure TBD)

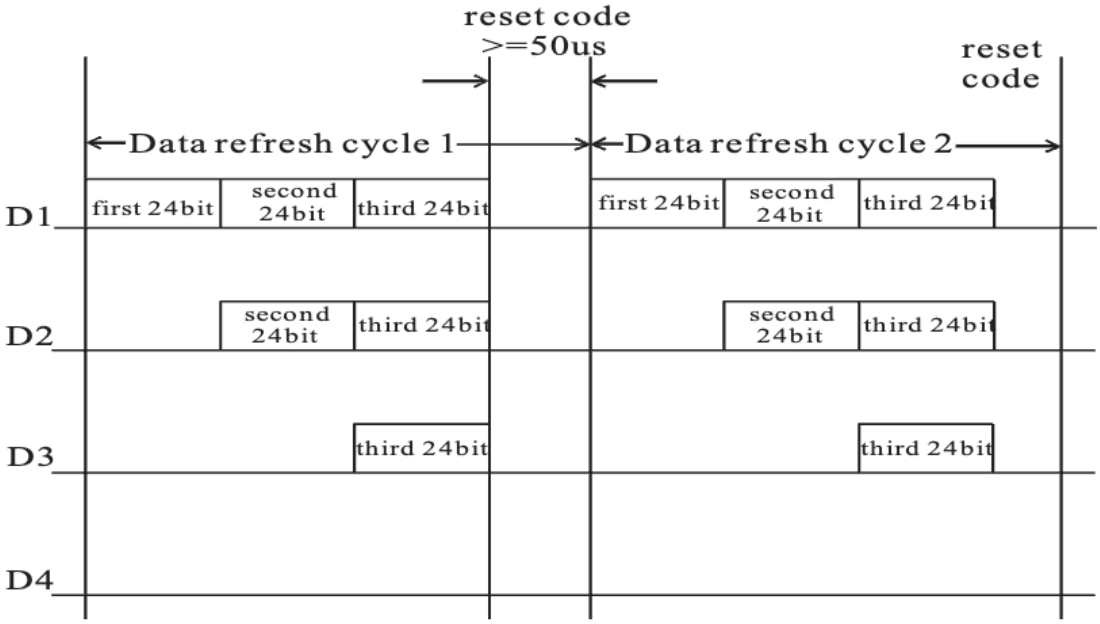


Figure 38. Data transmission method on linked WS2812B LEDS [13]

7.2.3 Creating the Android application

For creating the phone application we decided to go with Flutter. Flutter is a framework developed by Google that allows you to learn Dart language and build native mobile apps with the following advantages [25]:

- **Cross-platform development**: By using Flutter, we can use the same codebase and deploy our application on Android, iOS, web or desktop with minimal modifications.
- **Fast and dynamic code writing**: Changes made on the codebase are made almost instantly, without losing state.
- **Highly customizable UI design**: Flutter already includes a UI library with reusable elements such as buttons, text inputs, and sliders
- **Future-proof**: Flutter is a new framework which is actively improved by Google.

The purpose of our application is to show proof of communicating with AILED. At first iteration, we would want to be able to connect to the wristband via Bluetooth and change the colours of the LEDs.

7.2.4 Implementing the Bluetooth protocol

Implementing Bluetooth to our project has two steps: Activating the Bluetooth antennas from the Raspberry Pi Pico W and make our Android application handle Bluetooth operations:

7.2.4.1 Bluetooth for Raspberry Pi Pico W

The Pico W contains a CYW43439 that allows us to use wireless and bluetooth. The Raspberry Pi Foundation recently added examples on their open source github repository on how to connect to the microcontroller using BTstack, an open-source bluetooth stack designed for embedded systems. We will use those examples to activate the bluetooth antennas and make the microcontroller discoverable. [26]

7.2.4.2 Bluetooth for Android application

The implementation of bluetooth on flutter framework is quite well documented. We can use the package “Flutter reactive BLE library”, the main library that handles Bluetooth operations on flutter. More about it in the development process.

7.3 Development

The goal of this section is to explain the development process we took for creating a working AILED. With AGILE in mind, multiple subtasks were created to make a rather difficult task into a more manageable one.

7.3.1 Creating the development environment

Thanks to our extensive research during the design phase, we know that we will use macOS, C as the programming language. We will need to install Pico SDK in our environment, pull the examples from the official codebase of the Raspberry Pi Pico, use CMake for build automation and packaging and Visual Code Studio as the IDE. The following steps were made [27]:

1. Install the SDK
    a. In Terminal, we created a new directory named git in the root path using command *mkdir ~/git*
    b. Cloning the Pico SDK by running:

        git clone -b master --recurse-submodules https://github.com/raspberrypi/pico-sdk.git

    c. Add the PICO SDK PATH to .zshrc. This way we won’t have to tell to every project where to find PICO SDK:

        export PICO_SDK_PATH="$HOME/git/pico-sdk"

2. Install the toolchain
    a. brew install cmake
    b. brew tap ArmMbed/homebrew-formulae
    c. brew install arm-none-eabi-gcc
3. Configure the IDE
    a. Install Microsoft Visual Studio Code from the official website
    b. Install CMake Tools by Microsoft extension
    c. Specify the kit/compiler as GCC for arm-none-eabi x.y.z

For future projects, we will have to use a boilerplate code for the CMakeLists.txt file. This file helps in building and importing code:

CMakeLists.txt

```
# What CMake to start at
cmake_minimum_required(VERSION 3.12)

# Include the subsidiary .cmake file to get the SDK
include(pico_sdk_import.cmake)

# Set the name and version of the project
project(PicoTest VERSION 1.0.0)

# Link the Project to a source file (step 4.6)
add_executable(PicoTest source.c)

# Link the Project to an extra library (pico_stdlib)
target_link_libraries(PicoTest pico_stdlib)

# Initalise the SDK
pico_sdk_init()

# Enable USB, UART output
pico_enable_stdio_usb(PicoTest 1)
pico_enable_stdio_uart(PicoTest 1)

# Enable extra outputs (SWD?)
pico_add_extra_outputs(PicoTest)
```

Figure 39. Boilerplate code for CMakeList.txt file

When we want to compile our code and upload it on our Pico W, we will connect the microcontroller via the microUSB while pressing the physical reset button and run in terminal cmake .. && make && open . There we will find a newly compiled .utf2 file which we will drag and drop on the microcontroller's partition.

7.3.2 Lighting a simple LED

While lighting an LED is quite trivial, in special a one coloured one, doing that using Pico and its GPIO (General Purpose Input Output) pins in C is not as easy.

main.c > main()

```c
/**
 * Copyright (c) 2020 Raspberry Pi (Trading) Ltd.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

// Fade an LED between low and high brightness. An interrupt handler updates
// the PWM slice's output level each time the counter wraps.

#include "pico/stdlib.h"
#include <stdio.h>
#include "pico/time.h"
#include "hardware/irq.h"
#include "hardware/pwm.h"
#define PICO_DEFAULT_LED_PIN 7
void on_pwm_wrap() {
    static int fade = 0;
    static bool going_up = true;
    // Clear the interrupt flag that brought us here
    pwm_clear_irq(pwm_gpio_to_slice_num(PICO_DEFAULT_LED_PIN));

    if (going_up) {
        ++fade;
        if (fade > 255) {
            fade = 255;
            going_up = false;
        }
    } else {
        --fade;
        if (fade < 0) {
            fade = 0;
            going_up = true;
        }
    }
    // Square the fade value to make the LED's brightness appear more linear
    // Note this range matches with the wrap value
    pwm_set_gpio_level(PICO_DEFAULT_LED_PIN, fade * fade);
}

int main() {
    // Tell the LED pin that the PWM is in charge of its value.
    gpio_set_function(PICO_DEFAULT_LED_PIN, GPIO_FUNC_PWM);
    // Figure out which slice we just connected to the LED pin
    uint slice_num = pwm_gpio_to_slice_num(PICO_DEFAULT_LED_PIN);

    // Mask our slice's IRQ output into the PWM block's single interrupt line,
    // and register our interrupt handler
    pwm_clear_irq(slice_num);
    pwm_set_irq_enabled(slice_num, true);
    irq_set_exclusive_handler(PWM_IRQ_WRAP, on_pwm_wrap);
    irq_set_enabled(PWM_IRQ_WRAP, true);

    // Get some sensible defaults for the slice configuration. By default, the
    // counter is allowed to wrap over its maximum range (0 to 2**16-1)
    pwm_config config = pwm_get_default_config();
    // Set divider, reduces counter clock to sysclock/this value
    pwm_config_set_clkdiv(&config, 4.f);
    // Load the configuration into our PWM slice, and set it running.
    pwm_init(slice_num, &config, true);

    // Everything after this point happens in the PWM interrupt handler, so we
    // can twiddle our thumbs
    while (1)
        tight_loop_contents();
}

```

Figure 40. Example from github for fading a simple LED [23]

There were multiple steps that we had to do:

1. Telling the GPIO pin that we want the PWM functionality. This is necessary as GPIO can also be a PIO pin.
2. Choose a slice for the pin. There are 26 GPIO pins on the microcontroller but not all of them can be PWM at the same time. There is a limit of 8 PWM that can be used at the same time. What we do is just allocating a PWM slice for our pin.
3. Setting some default values we can send to the LED. The pin can send from 0 to 3.3V to the LED. The program does not understand the concept of voltages, so we map those values to any granularity we want. Usually they are mapped from 0 to 255 (this being the maximum value you can have in an RGB value) or multiple of 255 for better granularity.
4. Have a loop where we fade the LED.

The result:

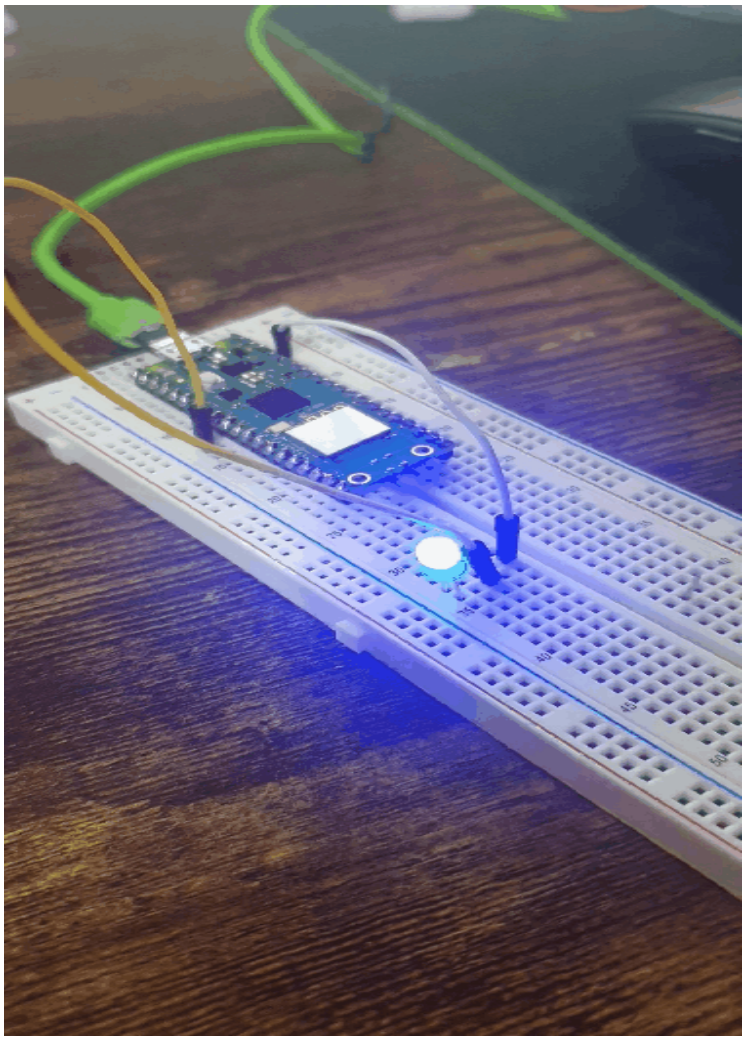


Figure 41. Blinking LED

7.3.3 Lighting an RGB LED

We can think of an RGB LED as 3 LEDS combined in 1. We want to be able to give values to each LED individually. By doing so, we can create any colour we want. The code is very similar to the code on the previous subsection. The only difference is we allocate 2 other PWM slices and 2 different GPIO pins. If someone wants to see the full code, they can check it on the github repository found in the Appendix.

The result:

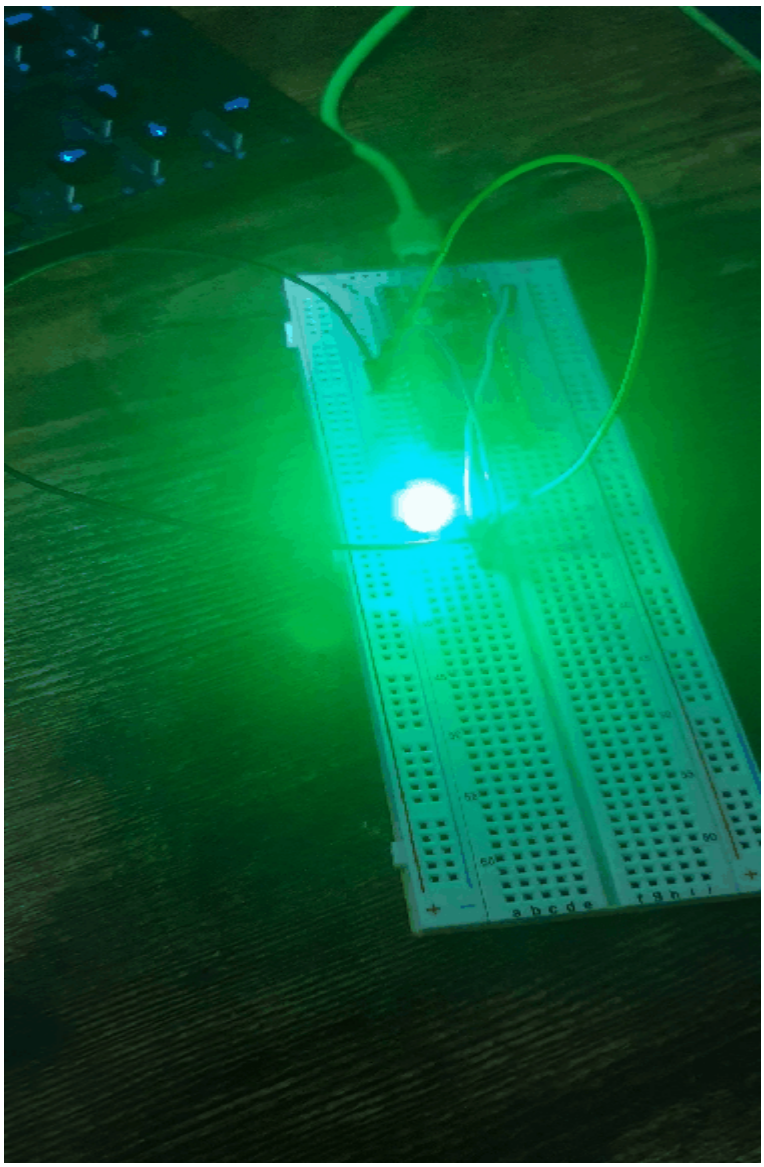


Figure 42. Blinking RGB LED

7.3.4 Lighting multiple WS2812B LEDS

As described in section 7.2.2.4, lighting up a WS2812B LEDS is much different than a regular one. You have to convert your data in binary, in a specific order and also with specific timings. This could have been a project on its own but luckily, there are libraries that abstract all the hard work:

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "pico/stdlib.h"
#include "hardware/pio.h"
#include "hardware/clocks.h"
#include "hardware/pwm.h"
#include "ws2812.pio.h"

#define NUM_PIXELS 10
#define WS2812_PIN 7
#define IS_RGBW false

void setup();

static inline void put_pixel(uint32_t pixel_grb)
{
    pio_sm_put_blocking(pio0, 0, pixel_grb << 8u);
}

static inline uint32_t urgb_u32(uint8_t r, uint8_t g, uint8_t b)
{
    return ((uint32_t)(r) << 8) |
           ((uint32_t)(g) << 16) |
           (uint32_t)(b);
}
static inline void light_led(uint8_t r, uint8_t g, uint8_t b)
{
    pio_sm_put_blocking(pio0, 0, urgb_u32(r, g, b) << 8u);
}
void setup_leds()
{
    PIO pio = pio0;
    int sm = 0;
    uint offset = pio_add_program(pio, &ws2812_program);
    ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW);
}

int main()
{
    setup();

    while(1) {
        light_led(255, 0, 0);
        sleep_ms(1000);
        light_led(0, 255, 0);
        sleep_ms(1000);
        light_led(0, 0, 255);
        sleep_ms(1000);
    }
}

void setup()
{
    stdio_init_all();
    setup_leds();
}
```

Figure 43. Code for lighting one WS2812B LED [28]

In the code from figure TBD, we had to set the GPIO now as a PIO, not as PWM as we did for simpler LEDS. By making it PIO, we are able to send bits of data to that specific port. There we have to initialise it with the pin, the frequency and whether the led band is RGB or RGBW. As described in the design section we will generate the RGB values that we want and then convert them into 8 bit values. That binary value is then being sent to the pio channel allocated to our pin. As you might have noticed, this will only light up only one LED in the band. We can light up as many as we want(and

different colours if we want to) by calling the light_led method multiple times. The only criteria is to have less than 50 microseconds between the calls. The result:

7.3.5 Getting the environment audio signal

To get the environment audio signal we will take advantage of the ADC built-in to the microcontroller. For using the ADC, we need to include the “hardware.adc.h” library found in Raspberry Pi Pico SDK. A few configurations parameters need to be declares, relevant ones are [29]:

Figure 44. Rainbow effect on a WS2812B LED band

1. CAPTURE_CHANNEL: Pico contains 4 ADC channels that can work simultaneously. For our purpose we only need one and we specify it as 2.
2. GPIO: The general input output pin used. It is calculated as 26+CAPTURE_CHANNEL. For our case we are using pin 28.
3. ADC_FIFO_SETUP: All the analog that is being put in a first in, first out queue and where it stays until it's computed.
4. ENABLE_DMA: Direct Memory Access is a feature that allows the ADC system to transfer data without adding overhead to the CPU. This is set to true for optimization purposes.
5. CLOCK_DIV: The clock speed set for the CPU to do the conversion. This is important as it sets the sampling rate for processing the audio signal. More details in the next section.
6. NSAMP: Number of samples/values that are considered at a time for audio processing.

Below is a snippet with the code that handles the ADC:

```c
void sample(uint8_t *capture_buf)
{
    adc_fifo_drain();
    adc_run(false);

    dma_channel_configure(dma_chan, &cfg,
                          capture_buf,    // dst
                          &adc_hw->fifo, // src
                          NSAMP,         // transfer count
                          true           // start immediately
    );

    gpio_put(LED_PIN, 1);
    adc_run(true);
    dma_channel_wait_for_finish_blocking(dma_chan);
    gpio_put(LED_PIN, 0);
}

void setup()
{
    stdio_init_all();
    setupLEDS();
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_IN);

    adc_gpio_init(26 + CAPTURE_CHANNEL);

    adc_init();
    adc_select_input(CAPTURE_CHANNEL);
    adc_fifo_setup(
        true,  // Write each completed conversion to the sample FIFO
        true,  // Enable DMA data request (DREQ)
        1,     // DREQ (and IRQ) asserted when at least 1 sample present
        false, // We won't see the ERR bit because of 8 bit reads; disable.
        true   // Shift each sample to 8 bits when pushing to FIFO
    );

    // set sample rate
    adc_set_clkdiv(CLOCK_DIV);

    sleep_ms(1000);
    // Set up the DMA to start transferring data as soon as it appears in FIFO
    uint dma_chan = dma_claim_unused_channel(true);
    cfg = dma_channel_get_default_config(dma_chan);

    // Reading from constant address, writing to incrementing byte addresses
    channel_config_set_transfer_data_size(&cfg, DMA_SIZE_8);
    channel_config_set_read_increment(&cfg, false);
    channel_config_set_write_increment(&cfg, true);

    // Pace transfers based on availability of ADC samples
    channel_config_set_dreq(&cfg, DREQ_ADC);
```

Figure 45. Code for setting up ADC and sampling the signal [29]

Method sample takes the digitised signal and dumps it in capture_buf array for further processing.

7.3.6 Computing the audio signal

For computing the audio signal, we will use an open source library. We chose the following values for the parameters mentioned in the design phase:

- Maximum scanned frequency: 50,000 Hz
- Number of samples: 2048. This values gives a “realtime” sensation of the algorithm and also gives us an interval of 12 Hz
- Clock speed: 960. This is the speed of the CPU set up for FFT. This is necessary to get the 50,000 HZ maximum scanning frequency.

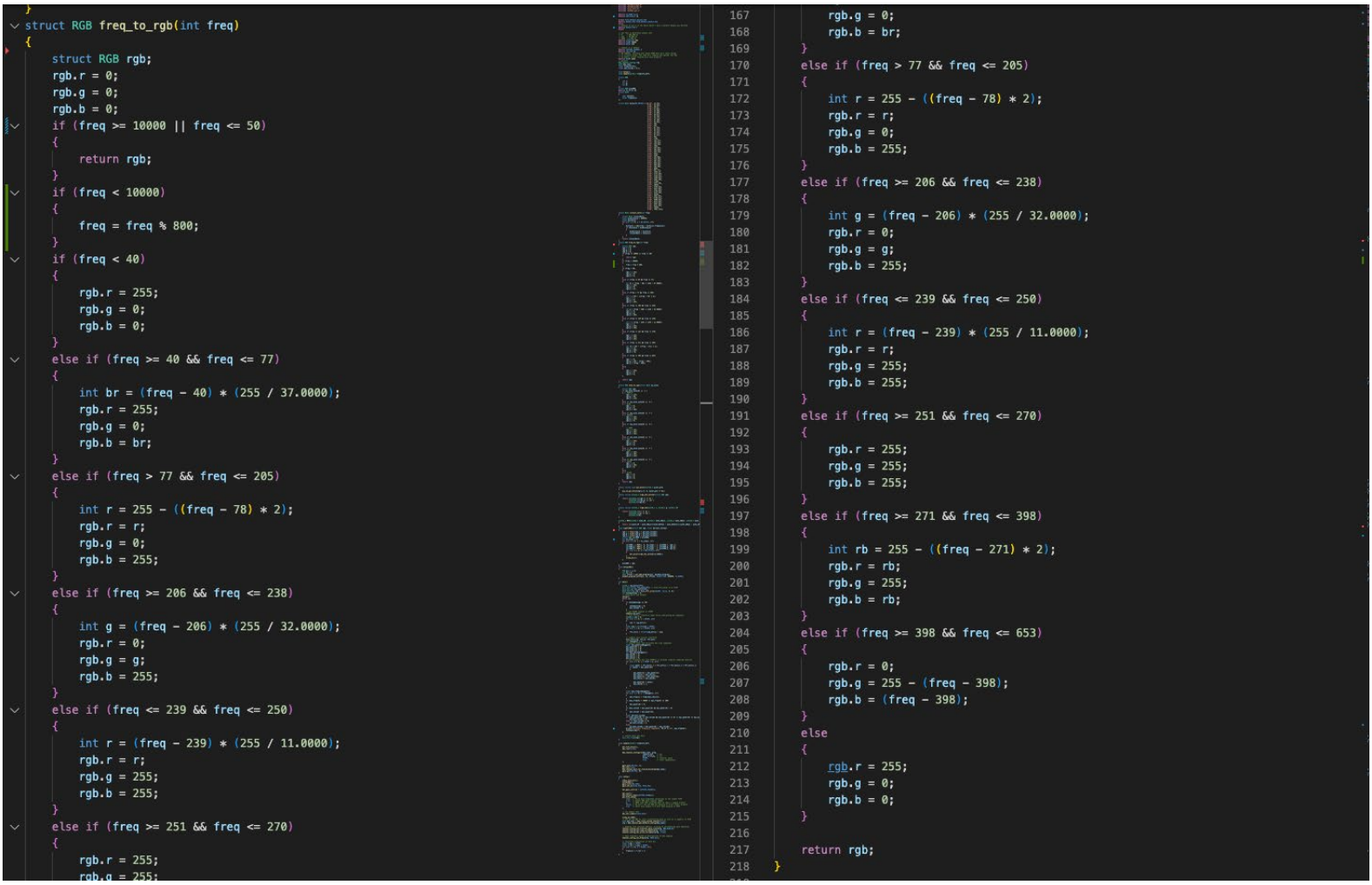
The result is an algorithm that takes the digital audio signal values from an array and gives you another array of the frequencies heard in the environment over a period of time and their power(magnitude).

```
23:32:55.952 -> Greatest Frequency Component: 439.5 Hz;
23:32:56.018 -> Greatest Frequency Component: 439.5 Hz;
23:32:56.117 -> Greatest Frequency Component: 439.5 Hz;
23:32:56.182 -> Greatest Frequency Component: 439.5 Hz;
23:32:56.282 -> Greatest Frequency Component: 439.5 Hz;
23:32:56.347 -> Greatest Frequency Component: 439.5 Hz;
23:32:56.446 -> Greatest Frequency Component: 439.5 Hz;
23:32:56.544 -> Greatest Frequency Component: 439.5 Hz;
23:32:56.610 -> Greatest Frequency Component: 439.5 Hz;
23:32:56.710 -> Greatest Frequency Component: 439.5 Hz;
23:32:56.775 -> Greatest Frequency Component: 439.5 Hz;
23:32:56.874 -> Greatest Frequency Component: 439.5 Hz;
23:32:56.972 -> Greatest Frequency Component: 439.5 Hz;
23:32:57.038 -> Greatest Frequency Component: 439.5 Hz;
```

Figure 46. The frequency was found when a frequency of 440 Hz was played using a tuner app.

7.3.7 Generating the colours

Taking the idea we had in the designing phase, we quickly made a method that takes the frequency as an input and outputs a struct with an RGB value:



```
}
struct RGB freq_to_rgb(int freq)
{
    struct RGB rgb;
    rgb.r = 0;
    rgb.g = 0;
    rgb.b = 0;
    if (freq >= 10000 || freq <= 50)
    {
        return rgb;
    }
    if (freq < 10000)
    {
        freq = freq % 800;
    }
    if (freq < 40)
    {
        rgb.r = 255;
        rgb.g = 0;
        rgb.b = 0;
    }
    else if (freq >= 40 && freq <= 77)
    {
        int br = (freq - 40) * (255 / 37.0000);
        rgb.r = 255;
        rgb.g = 0;
        rgb.b = br;
    }
    else if (freq > 77 && freq <= 205)
    {
        int r = 255 - ((freq - 78) * 2);
        rgb.r = r;
        rgb.g = 0;
        rgb.b = 255;
    }
    else if (freq >= 206 && freq <= 238)
    {
        int g = (freq - 206) * (255 / 32.0000);
        rgb.r = 0;
        rgb.g = g;
        rgb.b = 255;
    }
    else if (freq <= 239 && freq <= 250)
    {
        int r = (freq - 239) * (255 / 11.0000);
        rgb.r = r;
        rgb.g = 255;
        rgb.b = 255;
    }
    else if (freq >= 251 && freq <= 270)
    {
        rgb.r = 255;
        rgb.g = 255;
        rgb.b = 255;
    }
    else if (freq >= 271 && freq <= 398)
    {
        int rb = 255 - ((freq - 271) * 2);
        rgb.r = rb;
        rgb.g = 255;
        rgb.b = rb;
    }
    else if (freq >= 398 && freq <= 653)
    {
        rgb.r = 0;
        rgb.g = 255 - (freq - 398);
        rgb.b = (freq - 398);
    }
    else
    {
        rgb.r = 255;
        rgb.g = 0;
        rgb.b = 0;
    }

    return rgb;
}
```

Figure 47. Code for mapping the frequency to an RGB value

7.3.8 Implementing Bluetooth on the microcontroller

Using the BTStack implementation offered to us by the github repository [30], we were able to find our microcontroller using our Android phone:

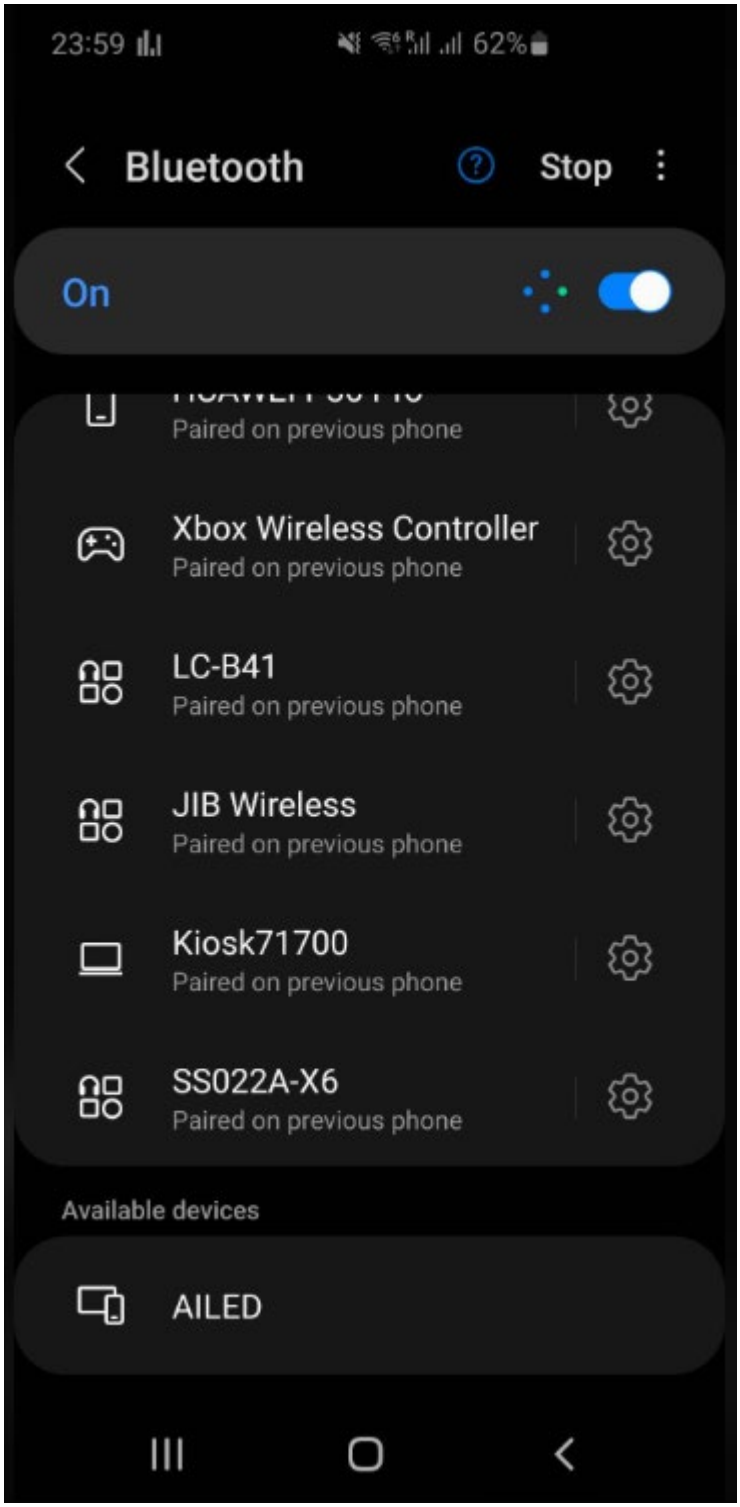


Figure 48. AILED found on the android phone

7.3.9 Creating the Android application

The following creates a basic android application using flutter. It only has 4 buttons: One button to connect the phone to AILED, and 3 buttons to change the colour of the LEDs. This is to show the bluetooth functionality as a proof of concept for further improvements:

```dart
// filename: main.dart
import 'package:flutter/material.dart';
import 'package:get/get.dart';

void main() => runApp(GetMaterialApp(home: Home()));

class Home extends StatelessWidget {
@override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(title: const Text('AILED Android Application')),
body: Center(child: Column(children:[

SizedBox(height: 50.0),

ElevatedButton(
onPressed: () => {},
child: Obx(() => Text('Connect',
style: TextStyle(
fontSize: 25, fontWeight: FontWeight.bold))),  // TextStyle, Text, Obx, ElevatedButton

SizedBox(height: 20.0),
button(()=> {},'RED'),

SizedBox(height: 20.0),
button(()=> {},'BLUE'),

SizedBox(height: 20.0),
button(()=> {},'GREEN'),

]))};  // Column, Center, Scaffold

Widget button(func,text){
  return ElevatedButton(child:Text(text,style:TextStyle(fontSize:20)),
  onPressed:func);}  // ElevatedButton
}
```

Figure 49. Flutter code for the Android application

The result:

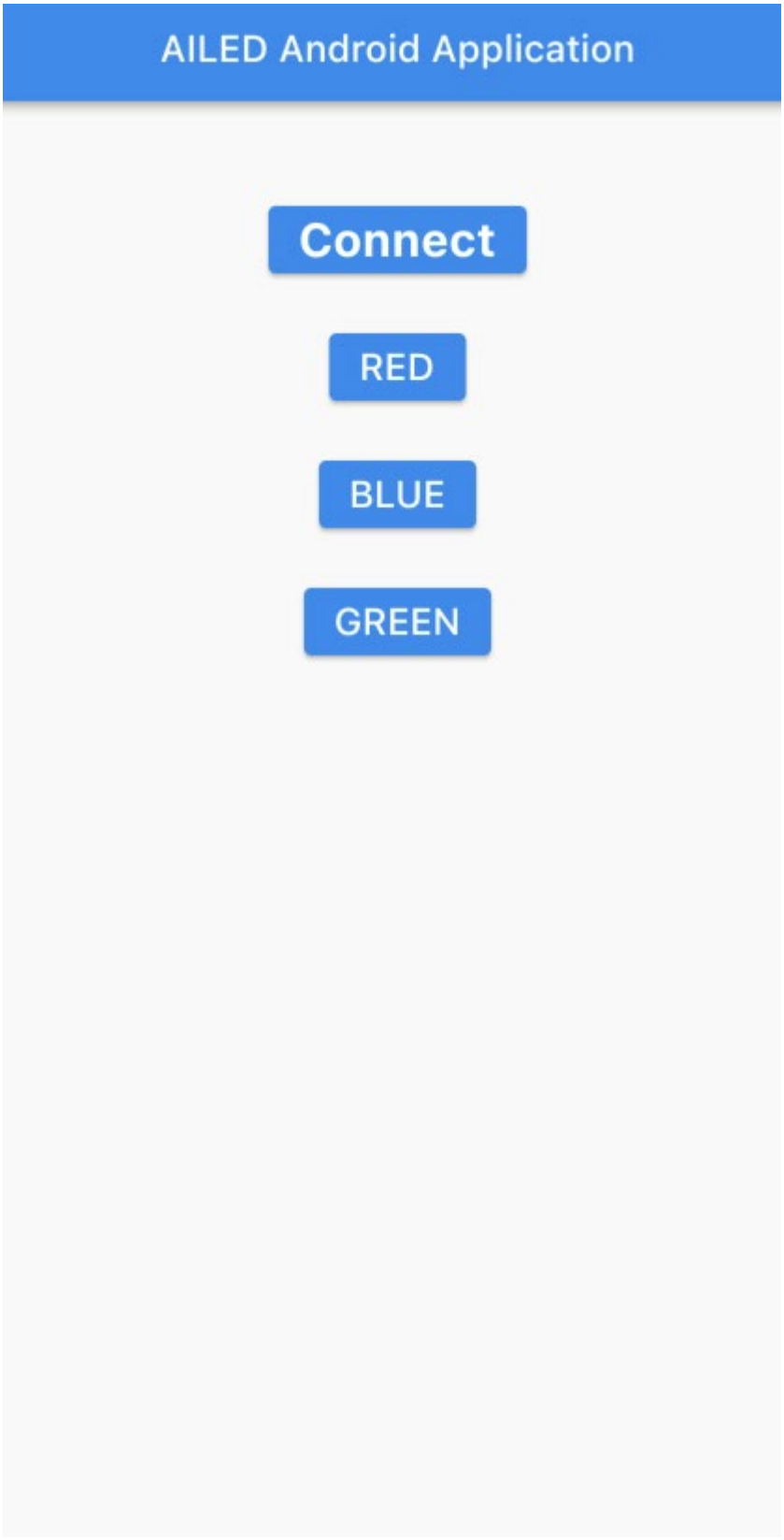


Figure 50. UI of the Android application

7.3.10 Adding Bluetooth capabilities on the application

As discussed in the design phase, we will add bluetooth using the “Flutter reactive BLE” library. We have to make sure that the application asks the user for the following permissions:

- BLUETOOTH_SCAN
- BLUETOOTH_CONNECT
- ACCESS_FINE_LOCATION
- ACCESS_COARSE_LOCATION

All these permissions are mandatory to find and connect the microcontroller.

The option we chose looks specifically for our microcontroller. Its device id is hardcoded and was found using an application named “nRF Connect”.

```
import 'package:flutter/material.dart';
import 'package:flutter_reactive_ble/flutter_reactive_ble.dart';
import 'package:get/get.dart';
import 'dart:async';
import 'dart:typed_data';

class BleController {
  final frb = FlutterReactiveBle();
  late StreamSubscription<ConnectionStateUpdate> c;
  late QualifiedCharacteristic tx;
  final devId = '43:43:A2:12:1F:AC'; // use nrf connect from playstore to find
  var status = 'connect to bluetooth'.obs;
  var sliderVal = 0.0.obs;
  var currentSpeed = 0;
  var currentDirection = 0;
  List<int> packet = [0, 0];

  void sendData(val) async{
    packet[0]=val.toInt();
    await frb.writeCharacteristicWithoutResponse(tx, value: packet);}

  void connect() async {
    status.value = 'connecting...';
    c = frb.connectToDevice(id: devId).listen((state) {
      if (state.connectionState == DeviceConnectionState.connected) {
        status.value = 'connected!';

        tx = QualifiedCharacteristic(
            serviceId:        Uuid.parse("6e400001-b5a3-f393-e0a9-e50e24dcca9e"),
            characteristicId: Uuid.parse("6e400002-b5a3-f393-e0a9-e50e24dcca9e"),
            deviceId: devId);
}});}
```

Figure 51. Bluetooth controller class.

sendData is used to send information from an application to the microcontroller. By pressing one of the 3 colour buttons, the user sends a hexadecimal value.

7.3.12 Combining everything together

Up until this point we had two different applications for the microcontroller: one that changed the LEDs' colour based on the environment's audio and one that changed it based on the button pressed in the android application. We added a new button in the application named "Toggle Music Mode", which when pressed it toggles whether the AILED listens for the environment's audio.

Also, to have both sound processing capabilities and be "ready" for any incoming message via bluetooth, we will take advantage of the multiple cores on the microcontroller. Pico W has 2 cores which allows us to run 2 processes in parallel. For our use case, sound processing and bluetooth communication respectively. Using the multi-core example provided to us by Raspberry Pi Foundation on their github: multicore_launch_core1 method will trigger the second core and start running the code from the function specified as a parameter. Eg. multicore_launc_core1(setup_bluetooth) will run the code within the function "setup_bluetooth"on the second core. To interact between cores, we will use the method "multicore_fifo_push_blocking"and "multicore_fifo_get_status" which sends and checks a queue that is shared between cores for new data. This way, we can let the other core know if we need it to do something or not.

The results:

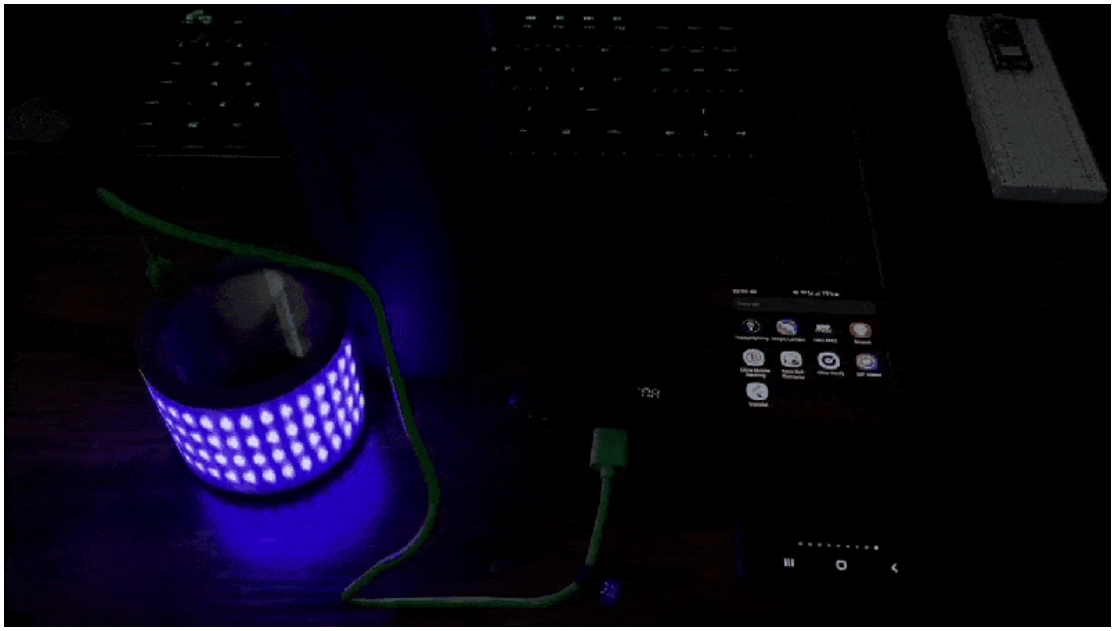


Figure 52. AILED reacting to music from speaker and then changes its colour based via Bluetooth

7.4 Testing

Testing the software for AILED has been done using console outputs and listening to them via the micro usb on the computer. Most important feature of the project is frequency recognition, which was tested using https://onlinetonegenerator.com/ , a website that generates a sound with specified frequency and outputs the result to the console.

Also, in the spirit of agile methodology, subdividing the task in smaller subtasks allowed us to manually unit test each feature before doing an integration test at the end with all the features.

8. Critical Evaluation of the AILED Project and Conclusions

This section evaluates the work completed in the project, what can be done in the future to further improve it, what are my thoughts regarding the experience and the conclusion.

8.1 Review of the work completed in the AILED project

The end result is a wristband that can achieve the desired outcome: connect people during music events using in sync light effects! You can also connect it to your phone and change the colour of the LEDS. Below is a video demonstration:

8.2 Further expansions

This from the beginning of the project was considered an open-ended idea with unlimited room to grow. Below is a list of further improvements that can be made:

- Better bracelet design: The 3D Model is the most basic way we could have achieved it. In the future, a hinge would be nice to have so it fits multiple hand sizes.
- Fully portable: Adding a portable battery is going to be challenging due to size of the bracelet and the power consumption but would be desirable. Changes in components might be necessary
- More android application features: Being able to choose any colour you want, changing the brightness or creating your own light show via the app would be nice to have. It would further expand the goal of connecting people and allowing them to express themselves.
- Hardware changes: Now that a final prototype has been created, I believe better microphones that reduce the environment noise or smaller, specialised micro-controller exist. By making these changes we could further improve the design and battery life!

8.3 Closing statements

The main purpose of this project was to explore, evaluate, construct, realise and assess a new way to connect people during events and I believe we did just that. The learning outcome was considerable as I had to learn about:

- Research: I had to learn how to do background research, structure and write an academic document.
- Hardware: I had to learn about multiple microcontrollers, microphones and types of LEDS.
- Signal processing: I had to learn how to convert analog to digital signal and also how to use the FFT algorithm.
- 3D Modelling: I had to learn how to 3D model based on an idea and simple shapes.
- 3D Printing: I had to learn how to assembly, configure and debug a 3D printer

- Soldering: I had to learn how to use a solder iron to make permanent connections between electronics.
- Electronics: I had to learn how to read diagrams and datasheets, wire, debug and work in general with electric components of small figures.

Overall, I believe it was a great project that gave me an end result that I am happy and will personally use. Also the entire software development cycle gave a huge satisfaction that will make me pursue more projects like his.

9. References

[1] David L. Andrews, Thomas Nann, Robert H. Lipson, editors in chief (2019). Comprehensive nanoscience and nanotechnology, Second edition. London: Academic Press.

[2] ASTM F2792-12a. Standard Terminology for Additive Manufacturing Technologies; ASTM International: West Conshohocken, PA, 2012.

[3] S. S. Crump, “Apparatus and Method for Creating Three-Dimensional Objects,” USA Patent, 1989

[4] Terry Wohlers, Wohlers Report 2009, ISBN 0-9754429-5-3.

[5] Campbell, T., Williams, C., Ivanova, O., Garrett, B. (2011). Could 3D printing change the world? Technologies, Potential, and Implications of Additive Manufacturing, Atlantic Council, Washington, DC.

[6] F. Samie, L. Bauer and J. Henkel, "IoT technologies for embedded computing: A survey," 2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2016, pp. 1-10.

[7] Rocchesso, D. (2003). Introduction to sound processing Mondo estremo.

[8] J. W. Cooley, P. A. W. Lewis, P. D. Welch (1969). The Fast Fourier Transform and Its Applications, .

[9] Al-Saqqa, S., Sawalha, S., AbdelNabi, H. (2020). Agile software development: Methodologies and trends. International Journal of Interactive Mobile Technologies. 14.

[10] Simion, M., Project Specifications Design and Prototype

[11] https://www.lc-led.com/products/slt-5050rgb.html

[12] https://ar.aliexpress.com/item/32859834798.html

[13] https://pdf1.alldatasheet.com/datasheet-pdf/view/1179113/WORLDSEMI/WS2812B.html

[14] https://www.alldatasheet.com/datasheet-pdf/pdf/3068/MOTOROLA/LM393.html

[15] https://cleste.ro/modul-microfon-sensibilitate-inalta.html

[16] https://www.alldatasheet.com/datasheet-pdf/pdf/73367/MAXIM/MAX4466.html

[17] https://www.adafruit.com/product/1063

[18] https://www.alldatasheet.com/datasheet-pdf/pdf/217128/MAXIM/MAX9814.html

[19] https://learn.adafruit.com/adafruit-agc-electret-microphone-amplifier-max9814/assembly

[20] https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html

[21] https://www.makerspaces.com/how-to-solder/

[22] https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/7

[23] https://github.com/raspberrypi/pico-examples/blob/master/pwm/led_fade/pwm_led_fade.c

[24] https://github.com/mborgerding/kissfft

[25] https://flutter.dev/

[26] https://github.com/raspberrypi/pico-examples/tree/master/pico_w/bt

[27] How to program the Raspberry Pi Pico in C on a Mac | smittytone messes with micros

[28] https://github.com/raspberrypi/pico-examples/tree/master/pio/ws2812

[29] https://github.com/raspberrypi/pico-examples/tree/master/adc

[30] https://github.com/raspberrypi/pico-examples/tree/master/pico_w/bt

[31] Chaka, K.T., Fambri, L., Govindan, N. (2017). Kaolinite/Polypropylene Nanocomposites. Part 3: 3D Printing. International Research Journal of Engineering and Technology. 4 132-147. Available from.

Appendix I

Github repository: https://github.com/arkannay/bracelet