Department of Health and Science

School of Computing

Bachelor of Science

"*Improved Navigation System for the Arduino Uno*"
Interim Report

Kieran Mac Hale

C16777089

C16777089@myudublin.ie

DT282 - Year 4

2021 - 2022

Paul Laird (Supervisor)

ABSTRACT

Becoming an electronics hobbyist is easier now than ever before due to the release of new and more accessible micro controllers and programmable circuit boards such as the Arduino Uno and Raspberry Pi. Many enthusiasts of these technologies have attempted to make robotics and other electronic devices which require navigational capabilities. This capability must be programmed separately and can often get in the way of the developer's main objective. The goal of this project is to create a codebase which implements a navigational system, which can be downloaded and used with the most commonly used hardware in the field. Using the Arduino Uno, a servo motor, and an ultrasonic sensor as the main hardware, this system will be capable of range and object detection, as well as determining a safe path forward for an object of a given width and length.

In order to achieve range and object detection across a spectrum of 180°, the HC-SR04 ultrasonic sensor and SG-90 servo motor were chosen. The travel times of sound waves emitted from the ultrasonic sensor are gathered and passed to an algorithm, which calculates the distance measurements. These measurements can then be sent to a back-end application via the serial port, where a Python program gathers the data using the "pyserial" library and completes the final calculations.

This project also provides a web application for users to download the necessary scripts, as well as launch the application from their browser. A feature of this web application allows users to configure the dimensions and specs of the project they are currently working on. Messages are then passed asynchronously between the client and server through a REST API, which provides update notifications to the user.

DECLARATION

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed: Kieran Mac Hale

Date: 10/11/2021

ACKNOWLEDGMENTS

I would like to thank Paul Laird for his supervision and guidance thus far on the project. I would also like to thank Damian Gordon for his guidance throughout the course of this documents' writing process. Finally, I would like to thank my family and friends for their support throughout the development of this project.

Table of Contents

INTRODUCTION

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

In 2005, the company Arduino released the first in what would become a line of easy-to-use programmable circuit boards. Arduino is an open-source hardware and software company who specialize in manufacturing single-board microcontrollers and microcontroller kits for building digital devices and robotics. The release of the "Uno" microcontroller in 2010 ushered in a new era for the Arduino community, drawing the interest of electronics hobbyists, programmers, and serious engineers alike. It's hard to categorise the Arduino community among the many branches of engineering, which has led to the term "Mechatronics" being widely adopted. Mechatronics is an interdisciplinary branch of computer science and electronics engineering which focuses on integrating computers with other electronic systems.

The projects that have emerged from this community have ranged from responsive bicycle backlights using LED's, to complex trackers that control the orientation of solar panels, directing them toward the sun. There is a vast amount of variety in the projects being worked on, but many of them share similar requirements. In particular, the need for navigational capabilities is common, not only for robotics, but also more ambitious projects such as smart walking sticks and self-driving cars. The problem of navigation has been tackled before, primarily by using the HC-SR04 ultrasonic sensor, an electronic module which comes with many of the popular development kits on the web. The inconvenience that comes with programming the behaviour of this module can stifle the progress of developers. Also, because the HR-SR04 is an *ultrasonic* sensor, it relies on atmospheric conditions, such as temperature, to be accounted for programmatically. The goal of this project then, is to not only provide an open-source navigational tool which can be integrated with other projects, but also to improve the

range and object detection capabilities of these sensors by creating an algorithm to handle ambient conditions which affect sound waves.

1.2 PROJECT DESCRIPTION

The improved navigation system for Arduino consists of two parts, a website where users can set up and configure the specifications of their project, and a back-end application which connects to the Arduino microcontroller via the serial port. Two modules, an ultrasonic sensor and servo motor connect to the Arduino board via digital I/O pins and are controlled by a C++ script uploaded to the Uno's Atmel ATmega8 chip. Launching the application will cause the motor to begin rotating 180° from left to right, taking in measurements at fixed intervals.

The complexity of this project comes in the implementation of algorithms which can calculate the distance measurements, as well as the system's ability to determine safe passages for objects of varying dimensions. A significant part of this project is the accuracy of these measurements. Two problems have been identified in achieving accuracy. Firstly, in order to accurately capture the environment in front of the sensor, the timing of the motor's movement and the sound waves being emitted from the sensor must be synchronised. Air density and temperature can affect the speed of sound, and since the distance measurements are derived from the time difference in sound waves leaving the trigger pin and returning to the echo pin of the sensor, these ambient conditions need to be accounted for somehow.

The approach for this project will consist of building a prototype schematic and navigation system which demonstrates range and object detection, and then testing it under various conditions. The system will be distributed to a number of graduates who have experience in the field of software and electronics engineering in order to get their feedback. This feedback along with testing and evaluation will help create a

robust system that other Arduino developers can use in their own projects.

1.3 PROJECT INSPIRATION

The need for navigational tools can be found everywhere in our day to day lives, from GPS systems that allow us to navigate our road networks, maps that allow us to explore the countryside, to sensors that make self-driving cars a possibility. For decades, creatives have imagined navigation devices, with some of them being noted for their surprisingly plausible design. However, these ideas are often left to the domain of science fiction.

Today, emerging software and hardware has made it possible for a single person to develop new, more advanced navigation tools, which can offer a wide range of use cases. These technological developments have inspired many software and electronics enthusiasts to form around online communities and collaborate on various projects. One of the most popular communities' centres around the Arduino Uno microcontroller, due to its simplicity and ready-packaged IDE.

Using the Arduino Uno and the Arduino IDE, I decided to build an open-source navigational tool that can be integrated with other projects. I have designed the system from the desire to accommodate as many developers as possible and allow them to achieve their goals more easily. This involved identifying what the most commonly used electronic modules are, their strengths and weaknesses, and how those weaknesses can be addressed in code.

The development of this system has also given me the opportunity to broaden my knowledge of these hardware and software technologies and improve my programming skills.

1.4 PROJECT OBJECTIVES

The main objective of this project is to improve upon the existing open-source navigation software for Arduino, while providing an online platform for users to access and configure projects to their liking. This can be achieved by following the objectives listed below.

- Conduct background research on common issues developers face
- Identify the most commonly used hardware components
- Implement an Arduino script to control and gather data from sensor
- Look into software and hardware solutions to the problem of atmospheric conditions in distance measurement
- Implement an algorithm to calculate a safe path
- Integrate the application with a web app
- Test and evaluate the front and back end components

1.5 PROJECT SCOPE

This is project aims to address a number of software issues in mechatronics, but it's important to state the scope of the project and what it *isn't* aiming to achieve.

While the final application will provide general navigational capabilities such as range and object detection, it has not been designed to handle any particular device or robot. The point of this system is that it can be tuned and configured to devices of varying sizes, but for which device it is eventually integrated, is ultimately in the hands of the user. Therefore, it is also not a goal of the finished system to handle the behaviour or take control of the device after the navigation algorithm has been completed and a safe path determined. Some users may want to use this feature to move a robot or some other electronic device, others may want to simply map the environment digitally. Finally, the electronic modules used in this project will be readily available and commonly used within the target audience. As this a software focused project, it does not aim to improve navigational systems by inventing new modules or modifying the electronic sensors or motors used.

LITERATURE REVIEW

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

This chapter will cover the key areas of research that were carried out for this project. These topics include identifying the potential needs and use cases for a configurable navigation system, exploring which technologies are commonly used, comparing existing solutions and applications in this area of computer science, and finally, looking at ways to design and implement the proposed application.

Other research included in this section will look at the issues that arise when using an ultrasonic sensor for distance measurement as well as a variety of possible solutions, and the best approaches for designing a web-based user interface. Finally, it is important to understand the limitations of the main hardware used, as well as review existing research papers on this topic to reveal any potential difficulties that may arise during development.

2.2 ALTERNATIVE EXISTING SOLUTIONS TO THE PROBLEM

There are many existing projects online which make use of the Arduino board and IDE in order to implement object detection. As there are many use cases for this feature, these projects can vary drastically depending on their core requirements. For some of the more user-oriented projects, the device(s) rely on fast and accurate calculations to be made, while other projects are more recreational in nature, and therefore the accuracy and speed of object detection is less important.

The following are various Arduino projects that implement object and/or range detection using ultrasonic sensors.

MIYBot

Figure 1 – MIYbot

MIYbot[1] is an open source robot circuit board project which makes it easier for electronics hobbyists to build robots. It uses two HC-SR04 ultrasonic sensors and continuous rotation servos for motors. This project uses the Adafruit Trinket circuit board as opposed to the Uno. This board uses the same ATmega328 microchip as the Uno but has been noted as being less robust and stable. On top of that, the Trinket and Pro Trinket are known to have issues when communicating over USB, and sometimes require device drivers to be reinstalled. This is one reason many current developers opt for the more popular Raspberry Pi and Arduino circuit boards.

Ultrasonic Glasses for the Blind

Figure 2 – Ultrasonic Glasses for the Blind

[1] https://hackaday.io/project/25984-miybot

Ultrasonic Glasses for the Blind[2] is a navigational device for the visually impaired, implemented with the HC-SR04 sensor. The sensor is attached to a pair of sunglasses and connects to the Uno via jumper wires. These glasses are designed to detect objects in front of the wearer and uses a buzzer to warn them. This project demonstrates how some of the most readily accessible and commonly used tools can be used to make something that benefits handicapped people.

Arduino Radar

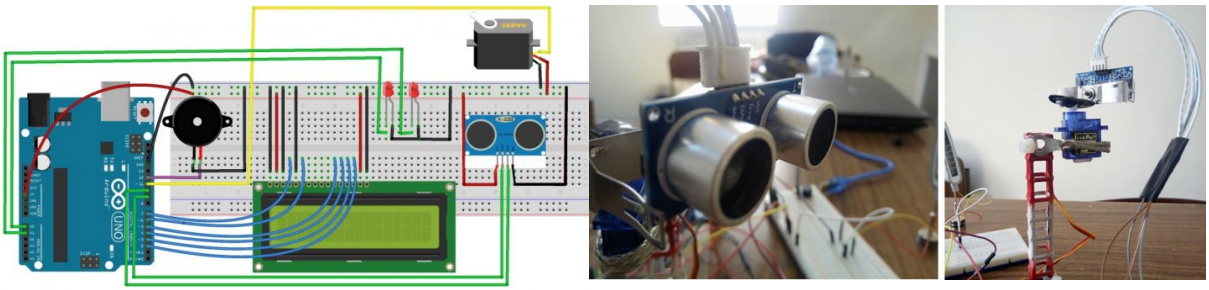


Figure 3 – Arduino Radar Model Using Ultrasonic Sensor for Detection and Ranging Schematic

This radar system[3] uses the Arduino Uno and an ultrasonic sensor to achieve object and range detection. Radio waves are used in order to identify range, altitude, direction, and the speed of objects. The project also aims for the radar system to be capable of detecting moving as well as stationary objects. A buzzer has also been implemented, which goes off when objects come within a certain range of the sensor. While radar and ranging can be achieved using radio waves, an effective radar system would likely require a more powerful sensor than the HC-SR04 model, which has the limitation of only being capable of efficient distance measurement up to 4 metres.

[2] https://create.arduino.cc/projecthub/gardnertech/ultrasonic-glasses-for-the-blind-142156
[3] https://how2electronics.com/arduino-radar-model-ultrasonic-sensor/

Obstacle Avoiding Robot

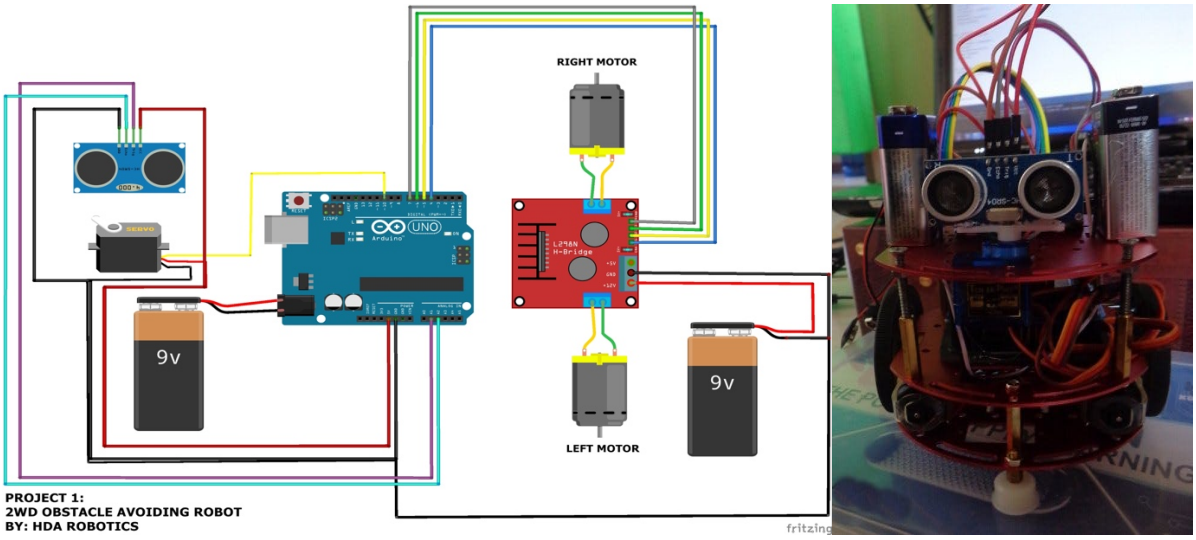


Figure 4 – Obstacle Avoiding Robot

This obstacle avoiding robot[4] developed by 'HDA Robotics' uses many of the same electronic components as the previous examples and facilitates movement for a small robot of a fixed size. Two DC motors are used to control the movement of the robot, while a servo is used to rotate the sensor. One of the functionalities of this project is that the device is capable of acting on decisions from the measurements taken. However, the system is not configurable and can only handle navigation for the robot that was designed for this particular project. If the developer of this project wished to substitute the existing robot with another of a different size, they would need to re-configure their code and schematic.

Blynk

Blynk[5] is a mobile app which allows users to configure and modify microcontrollers remotely. The app supports a wide

[4] https://create.arduino.cc/projecthub/hda-robotics/project-1-2wd-obstacle-avoiding-robot-390ef8
[5] https://blynk.io/

array of hardware modules, including all Arduino, Raspberry Pi, and Particle models.
The app also allows users to interact with the finished project by controlling components, such as the frequency of a blinking LED, the speed of a rotating motor, or by reading sensor data into the app.

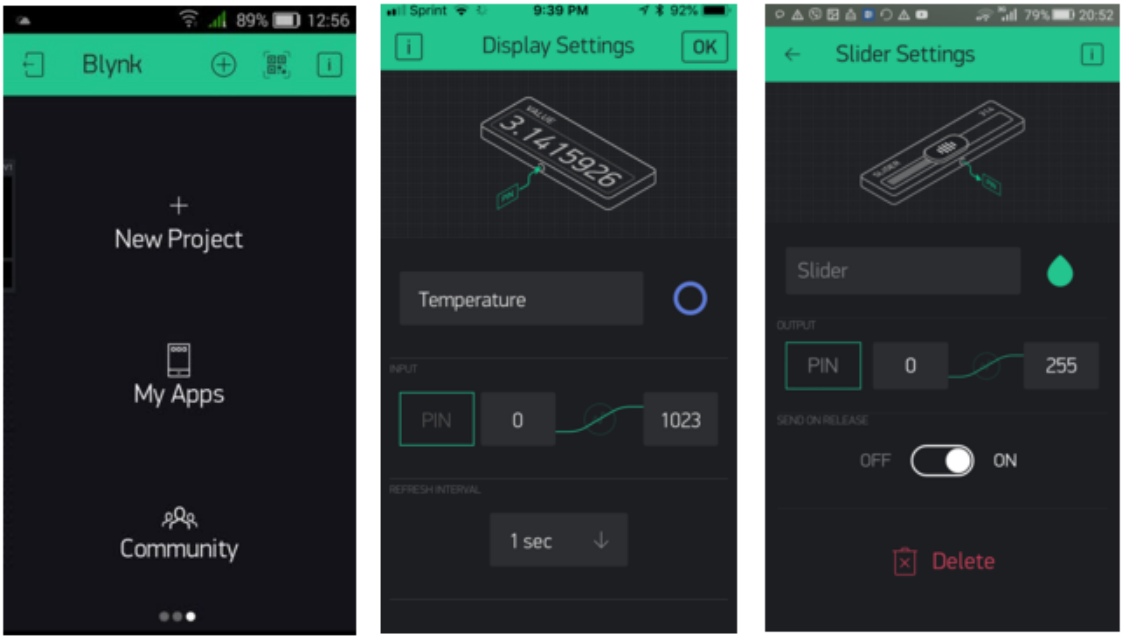


Figure 5 – Blynk App

The simple and accessible user interface makes it easy for users of all skill levels to understand. Developers can take control of their projects with intuitive sliders and switches whose utilities are clearly indicated due to their sleek and clever design. The simplicity of the app is great for convenience and accommodating a larger userbase, however, it also limits how much control developers have when interacting with and configuring projects remotely.

2.3 TECHNOLOGIES RESEARCHED

2.3.1 MAIN HARDWARE

ARDUINO UNO

The Arduino Uno[6] is a low cost and easy to use programmable circuit board which was developed at the 'Interaction Design Ivrea' in Ivrea, Italy. The board can be integrated into a variety of electronics and software projects and can be interfaced with other Arduino boards. Over the years, the suite of Arduino microcontroller models has expanded, and a fast-growing community of enthusiasts have rallied around the development of challenging projects online. These projects have evolved in scope as the community has grown, from simple prototyping projects to complex satellite GPS applications, robots, and embedded systems.

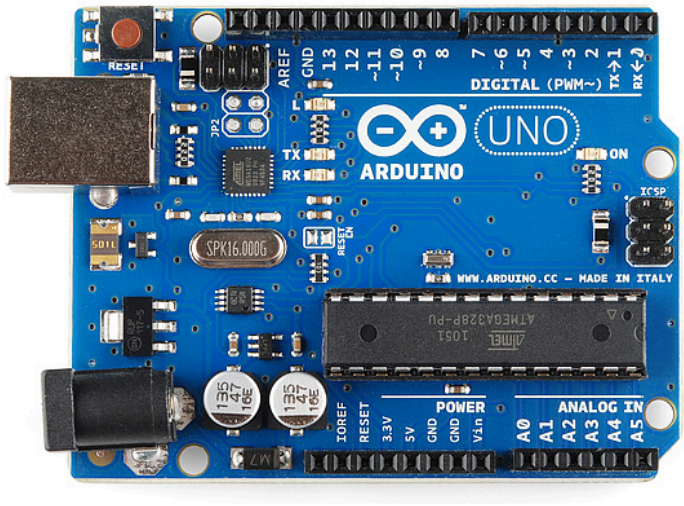


Figure 6 - Arduino Uno

The Uno (Figure 6) is comprised of various components. A USB interface is used to connect the board to a computer and the Arduino IDE (Integrated Development Environment), the software interface used to write programs for the board. Fourteen digital I/O pins (labelled 0-13) are available to connect to external electronic devices, such as a sensor or a motor. Additionally, there are several ground pins for closing electronic circuits, and six analogue pins which are used to read analogue signals and convert them to digital signals. Six 'Pulse Width Modulation' (PWM) pins can also be used to control the behaviour of certain components, such as the speed of a motor or the brightness of

[6] https://docs.arduino.cc/hardware/uno-rev3

an LED. Programs are uploaded via USB to the board and stored on the ATmega328 microchip, which contains 32Kb of non-volatile flash memory for storing instructions.

The Arduino Uno has several advantages over other popular boards such as the Adafruit Trinket and Raspberry Pi 3. For one, the Arduino comes packaged with the previously mentioned IDE, a programming environment with cross-platform support written specifically for Arduino projects. Because of the popularity of Arduino models, there is also a lot of library support for widely used languages such as Python, C, C++, and Java. Some of these libraries are critical for the purposes of this project, as they allow different languages to communicate indirectly with each other.

HC-SR04 ULTRASONIC SENSOR

The HC-SR04 (Figure 7) is an ultrasonic non-contact distance measuring instrument, capable of detecting objects up to 400cm (4 metres) away. This sensor is the most popular among hobbyists and serious Arduino developers. The sensor is comprised of four pins; a 5V VCC pin which supplies power to the sensor, a trigger (Trig) pin used to control the ultrasonic sound pulses emitted from the sensor, an echo pin which produces a pulse when reflected signals are received, and a ground pin for closing the circuit.

Figure 7 - HC-SR04

The HC-SR04 sensor works by transmitting bursts of 8 sonic pulses at 40Khz. This creates a unique 8-pulse pattern, allowing the receiver to differentiate the transmitted pattern from ambience and/or other noise. Once the sonic pulse is transmitted, the receiver immediately produces a pulse signal, returning to low voltage once the transmitted pulse is received. The width of this pulse signal is then used to calculate the distance to the reflected object. The HC-SR04 is a reliable

sensor for most purposes, however the distance measuring calculation does not take ambient conditions into account. Other sensors will be explored in the next section which offer extra functionality in order to tackle this very problem.

US-100 ULTRASONIC SENSOR

Several other sensors have been considered for this project. The US-100 sensor shares most of its design and features with the HC-SR04 and operates at the same baud rate (9600) but has the extra utility of temperature adjusted range detection.

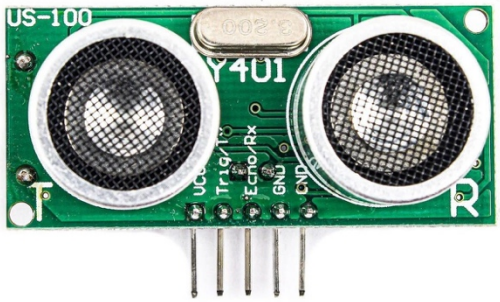


Figure 8 - US-100

The transmitter on the US-100 (Figure 8) has roughly the same beam angle as the HR-SR04 too, approximately 15°. Because of this, much of the infrastructure and library support for the HC-SR04 works with US-100[7]. As a result, the US-100 is capable of more accurate range detection. It's also slightly more flexible to work with as it can run on anything between 3-5V of power.

One final consideration to make is cost and availability. The US-100 is roughly the same price as the HC-SR04, and the next most popular choice on the market. The US-100 will be considered as an alternative to the HC-SR04, as a possible substitute. In the future, a feature may be added which allows users to choose which sensor they would like to use.

[7] https://www.digikey.com/catalog/en/partgroup/us-100-ultrasonic-distance-sensor-3v/84977

SG90 SERVO MOTOR

The TowerPro SG90 9g Mini Servo (Figure 9) is a 180° rotation motor. It is suitable for animatronics and controlling electronic devices that need to pivot around a stationary point, such as a ranging sensor scanning the environment. The SG90 is a digital servo that provides good torque (4.8V – 1.3kg/cm) and can operate from a standard pulse of ~3.3 – 5V of power.

Figure 9 - SG90 Servo Motor

Like the sensor, the SG90 is popular among Arduino developers and consequently has extensive library support for quickly achieving the desired behaviour. It comes equipped with sophisticated internal circuitry which allows it to process pulse width modulation signals faster and better than other models. This will play an important role in synchronising the rotation of the motor with the transmission of ultrasound waves from the sensor.

In conclusion, the main hardware modules used in this project will include the Arduino Uno programmable circuit board, used to control the SG90 servo motor and HC-SR04 ultrasonic sensor, retrieving data gathered by the sensor and sending it back to the Arduino IDE via a USB interface. Getting data from the Arduino programming environment to the main application will involve the use of the serial port and the 'PySerial' library. This will be explored in the next section which covers the software components researched for the project.

2.3.2 FRONT-END SOFTWARE

REACT JS

A number of front-end technologies such as libraries and frameworks have been researched for the purpose of building the views and user interfaces for this project. React JS is an open-source Javascript library for building user interfaces based on UI components. React components can be written as functional or class components, but both are used to return JSX. JSX is syntactic sugar for Javascript which creates HTML elements in the DOM, and makes it easy to mix HTML syntax with Javascript expressions. React handles UI components by keeping a copy of the DOM (Document Object Model) called the 'shadow DOM', and when updates are made to this copy, React notifies the browser and enables it to match the update in the real DOM.

AXIOS

Axios is a Javascript library which works well with React to create HTTP requests. This allows React apps to easily communicate with API's, enabling them to fetch data, submit and/or update data on a server, and more. Axios is particularly useful for creating the necessary HTTP requests for web-based projects that interface with a RESTful API. Using Axios, it is simple and intuitive to create request methods with all the necessary information, such as the URL, request headers, web tokens, and so on.

RxJS

Many modern web applications implement the observable pattern when dealing with streams of data. RxJS is a reactive programming library which facilitates asynchronous and callback-based applications by implementing programming abstractions such as observables (producers of multiple values)

and subscribers (single instances of an observable being executed). RxJS is a widely used library in modern web development and is compatible with React.

2.3.3 BACK-END SOFTWARE

REST API

A RESTful API is an application programming interface that conforms to the REST architectural structure and which provides a medium of communication for RESTful web services. This architectural model encompasses a suite of protocols which makes networked communication simpler for clients and servers. RESTful API's enforce these rules and deliver requests in certain formats, such as JSON or XLT. The most commonly used HTTP requests, such as GET, PUT, POST, and PATCH are all methods used in RESTful services.

FLASK

Flask is a web-based framework written in Python. It provides a set of tools and libraries for building web applications and is especially useful for writing server code. Flask is easily extensible, and in fact often relies on third party libraries for implementing simple functionality in a server. It has no form validation or database abstraction layer for example; such functionalities would need to be implemented with the 'Authlib' library and an object-relational mapping tool (ORM) such as 'SQLAlchemy', respectively.

PYSERIAL

PySerial[8] is a Python library that provides support for serial connections in programs. Most notably among these are serial ports on computers, which can carry data from external devices such as microcontrollers. Programming environments such as the Arduino IDE use the serial monitor for data logging, and with this module, these data logs can be accessed by Python applications as they are being written. The ability to retrieve logs from the serial monitor will be important for collecting sensor data outside of the Arduino IDE.

NODE.JS & EXPRESS.JS

It was decided that Express.js and Node.js[9] would be the more appropriate choice as the back-end framework and Javascript runtime environment respectively. One advantage of using Node.js is that it will allow the server to be written in Javascript, which reduces the complexity of client-server communications due to both applications being written in the same language. It is also highly extensible, as resources and third-party libraries can be integrated with Node projects easily using the node package manager.

[8] https://pyserial.readthedocs.io/en/latest/
[9] https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Node.js/Introduction

2.4 OTHER RESEARCH

2.4.1 INTERPOLATION

Interpolation is a statistical method of estimating unknown values from a set of discrete values; for example, two discrete data points on a graph can be rewritten as an expression or continuous function, allowing for more thorough analysis. This method can, in many cases, enable us to derive data that may exist outside of the data initially collected. In terms of distance measurements, these values can be represented as points on a graph, and formulae such as linear interpolation can be used to make approximations relating to this data.

$$y = y_0 \left(1 - \frac{x - x_0}{x_1 - x_0}\right) + y_1 \left(1 - \frac{x_1 - x}{x_1 - x_0}\right) = y_0 \left(1 - \frac{x - x_0}{x_1 - x_0}\right) + y_1 \left(\frac{x - x_0}{x_1 - x_0}\right)$$

Figure 10 - Linear Interpolation Formula

If two points are known, such as the origin of the measuring instrument and the point an object was detected, the linear interpolant is the line between these two data points. The formula for linear interpolation (Figure 10) estimates the distance to an unknown point from a known point using weighted averages and normalization.

There are two use-cases where this may be useful in this project. As the HC-SR04 sensor will be taking measurements at finite, discrete angles, there will be some unknown measurements in between the collected data, which could be seen as ‘blind spots’ for the sensor and the project as a whole. Linear interpolation may be used to approximate these unknown measurements and create a more finely grained dataset for navigation processes. Additionally, interpolation could hypothetically be used to scale up the project in the future by estimating elevation levels from the collected sensor data. This requires a more involved and mathematical implementation to achieve reliable results but will be covered in a later section discussing potential future developments.

2.4.2 NIELSON'S HEURISTICS

Jakob Nielsen defined 10 general principles for interaction design. Nielson's Heuristics are a set of broad usability guidelines which take into account, among other things, user control, standardisation, error prevention, and flexibility of use. These guidelines will help in developing the web-based interface for users to interact with, especially when handling callback-based functionalities. For example, the first principle of Nielson's Heuristics regards visibility of the system status. This principle will be kept in mind when the user has made server requests and is waiting for a response, especially while the sensor is collecting data. Status updates will be used to notify the user of the current status of the system's progress, as well as any errors that occur.

2.4.3 UNIVERSAL DESIGN PRINCIPLES

The 7 principles of universal design are mainly concerned with the usability of services and applications for individuals with visual and cognitive impairments, and/or other impediments which make interaction difficult. This is of particular importance for web applications, as conforming to these principles is enforced by policy and legislation in various countries in the European Union[10]. The 7 principles outline guidelines for achieving flexible, error tolerant systems, whose features and data can be accessed in a variety of different ways and through different means (such as screen readers).

The Centre for Excellence in Universal Design claim that universal design principles aim to make the user experience better for everyone, rather than oversimplifying or confusing the original design of interfaces for the sake of including a wider userbase. These principles will be used to inform various design decisions for the front-end application and will be referenced where appropriate.

[10] https://universaldesign.ie/what-is-universal-design/policy-and-legislation/

2.5 SIMILAR PROJECTS AND CONCLUSION

As part of the research phase of this project, several similar existing projects from previous years were examined. The projects were chosen based on their requirements, tackling similar issues, using similar technologies, and shared goals.

2.5.1 SMART DRIVE – RICHARD MEYER

SmartDrive aimed to record driving habits and provide comprehensive feedback to its users regarding road safety and their potentially bad habits while driving. This project was implemented as a full stack web application which gathers time-stamp location data using the GPS on the users' phone while driving.

This application fetches time-stamp locations using GPS, but notably, does not use any speed measuring instruments such as a speedometer to calculate the speed, which is necessary for the application to work. Instead, the speed of the user is derived from the fetched time-stamp data, which is complex since the driver won't always be moving in a straight line.

SmartDrive struck me as a difficult system to evaluate and test, since its full utility can only truly be seen by driving over speed limits and making purposefully dangerous trips, and then reviewing how the system reacted.

2.5.2 ASSISTIVE NAVIGATION FOR THE VISUALLY IMPAIRED – RUDOLF SUGUITAN

This project aimed to use emerging technologies such as Raspberry Pi and the Arduino Microcontroller to develop a wearable device capable of range detection and object detection, for the purpose of assisting the visually impaired. The project also used the HC-SR04 ultrasonic sensor in its implementation.

On top of gathering data from the sensors and implanting an algorithm to calculate distances, the system also uses image processing to create object models and visualize objects which have been detected.

The application made some well thought considerations for the user interaction, since it is primarily designed to be used by people with partial or complete visual impairment. For example, text-to-speech technology was cleverly used to implement audio cues which are triggered to warn the user of hazards. Lastly, a special sensor was used with improved temperature corrected range detection to achieve better distance measurements.

I felt a weakness of the project that there was a missed opportunity to make use of speech recognition technologies to allow the user to control the device, which would be consistent with the universally designed spirit of the project. The library support for simple implementation of this technology is vast, especially for languages such as Python.

2.5.3 CONCLUSION

Combining the knowledge gleaned from literary research and existing projects, the next phase of development can be planned with some assurance. There is clear evidence that there is a vast array of use cases for these technologies, and that there is room for quality of life improvements. This knowledge will help in prototype development; deciding on which architectural models to adopt, which hardware and software tools to use, and what development approach is most suitable.

2.6 REQUIREMENTS TABLE

Name	Description	Priority
User register	Enable users to register through web app	HIGH
User login	Enable user to login through web app	HIGH
User logout	Enable user to logout through web app	HIGH
Download script	Provide a page for the user to download the necessary Arduino code	HIGH
Create Project	Allow users to create new projects	HIGH
Configure Project	Provide a form for users to configure specs for their projects	HIGH
Begin Scan	Allow users to remotely initialize scan	HIGH
Perform Scan	Arduino program executes	HIGH
Calculate Safe Path	Use collected sensor data to determine a safe path if one exists	HIGH
Status Update	Inform user on status of scan	HIGH
View Results	Provide visualisation of results	HIGH
Log Issues	Option for users to report issues after scan	HIGH
Save Project	Allow users to save configurations	MEDIUM
Interact with Visualiser	Provide ways for user to interact with scan results	MEDIUM
Update Configuration	Allow users to re-configure current project and scan again	MEDIUM
Interpolate measurements	Use linear interpolation to artificially enhance scan results	LOW
Approximate elevation	Use interpolation methods to make elevation approximations in results	LOW
Integrate JWT's	Use web tokens for authenticating users on web app	LOW

PROTOTYPE DESIGN

CHAPTER 3: PROTOTYPE DESIGN

3.1 INTRODUCTION

Following the background research and literary review covered in the previous section, this chapter will focus on the design of the system. Some key design decisions that will be highlighted include the architectural model chosen for the system (client-server), the design methodology, hardware schematic, use cases, and more. UML diagrams will be used to show use-cases, system entities, and describe the flow of the application from a user's point of view. Each section of this chapter will cover one of these decisions, as well as why particular architectures and methodologies were chosen.

3.2 DEVELOPMENT METHODOLOGY

Several software development methodologies were examined for this project, keeping in mind the requirements as well as the historical effectiveness of these models for similar projects. The first development methodology looked at was the 'Waterfall' approach.

3.2.1 WATERFALL METHODOLOGY

The Waterfall Methodology[11] (Figure 11) uses a sequential, linear approach to developing software applications. The project is broken down into smaller tasks which are completed in phases. These phases start with analysis and requirements elicitation, followed by design and development and finally, testing and deployment. These stages are repeated for each task until the project is completed. This is a highly structured and rigid methodology which has a long history of success with various large and small-scale projects.

The design phase of the waterfall model can be broken into two sub-phases; logical and physical design. During the logical

[11] https://www.sciencedirect.com/topics/computer-science/waterfall-methodology

design phase, possible solutions are brainstormed and theorized. Later in the physical design phase, the best of these hypothesised ideas are made into concrete specifications.

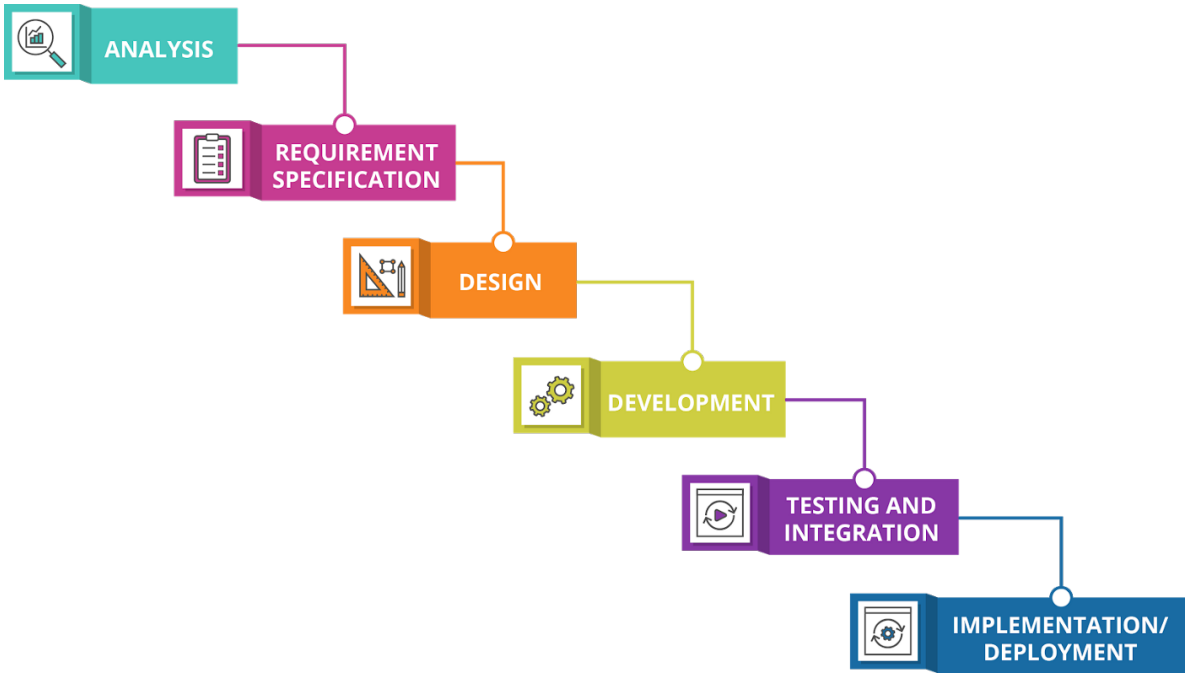


Figure 11 - Waterfall Model

The advantages of the waterfall approach are that it ensures that requirements are designed and completed earlier in the development of the app, and it easy to measure the progress of development using this methodology. It also forces developers to ensure they have a concrete understanding of the requirements before implementation begins.

Some disadvantages of this methodology are that it is less flexible to requirement changes when compared to other approaches and requires highly detailed breakdowns of the system to be completed before any implementation and testing can occur. It also discourages developers from revisiting features which have already been implemented. Although waterfall projects do not inherently have to span lengthy periods of time, the nature of the model makes it commonplace for projects to span months or years due to the emphasis of getting everything done at once. This may be suitable for projects worked on by carefully co-ordinated teams on big projects, but not for smaller individual ones.

As this is a solo project and there is some degree of uncertainty related to the requirements of this project (as seen in the requirements table), the waterfall approach would not be well suited for this project.

3.2.2 AGILE METHODOLOGY

Similar to the waterfall approach, the Agile Methodology[12] (Figure 12) breaks projects down into smaller tasks which are completed in short time-boxed periods known as sprints. This methodology allows for more flexibility in development, as project requirements and goals are re-evaluated at the end of each sprint, enabling designers and developers to innovate as they create their system.

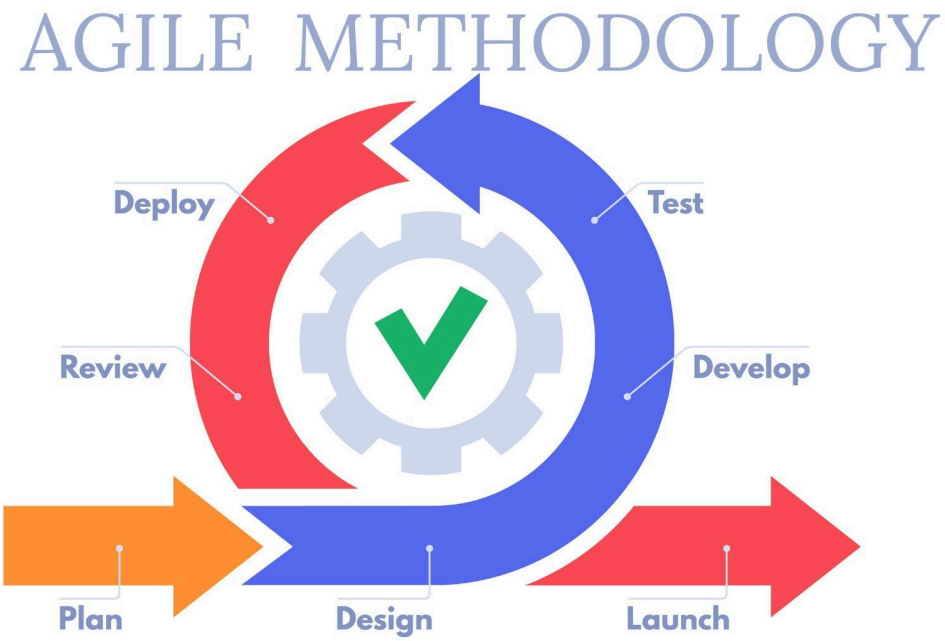


Figure 12 – Agile Model

The Agile approach works best when the full scope of requirements are unknown or it is difficult to anticipate how long certain requirements will take to implement. This methodology is suitable for the Arduino navigation app as changes are expected to occur during the development of the project. The scale of the project may also increase, and new features could be introduced depending on the success of high

[12] https://www.digite.com/agile/agile-methodology/

priority requirements being fulfilled early. Due to the nature of the project, testing and evaluation is expected to be done throughout, something the agile approach enforces. Using this methodology will allow the development of this project to progress in incremental steps, providing an opportunity to re-evaluate important facets of the system, such as accuracy and reliability. These are two particularly critical characteristics of this app, as it is not a feature rich software tool, and therefore should be capable of reliably completing its core utility.

3.3 SYSTEM OVERVIEW

For the prototype development phase of this project, an agile scrum methodology will be adopted, where individual tasks will be completed in sprints. Three categorisations of tasks have been decided, front-end tasks, back-end tasks, and hardware related tasks. At each stage of development, one task will be implemented for each of these categories, followed by testing and evaluation.

Development will initially focus on back-end functionality, such as creating API endpoints, establishing communication between the server and serial port, etc. Once this has been completed and tested using an API testing platform such as Postman, the focus will shift to the design and implementation of the front-end web interface.

The general approach for the development of the project will be as follows:

1. Design and develop a basic REST API, facilitating client-server communications
2. Design a schematic and feature for one hardware component
3. Implement API call
4. Test the feature
5. Repeat steps 2 – 4 until all desired API functionalities have been implemented
6. Design front-end interface
7. Implement front-end interface

8. Test and evaluate front-end

The technical architecture (TA) is a form of IT architectural design which describes the various layers of software components that communicate which each other in the system. A three-tier architecture (Figure 13) was chosen for this project, as it allows for flexibility and separates the user from the complexity of back-end processing. Changes made to one layer should not affect the other layers, which makes debugging and scaling simpler and more manageable.

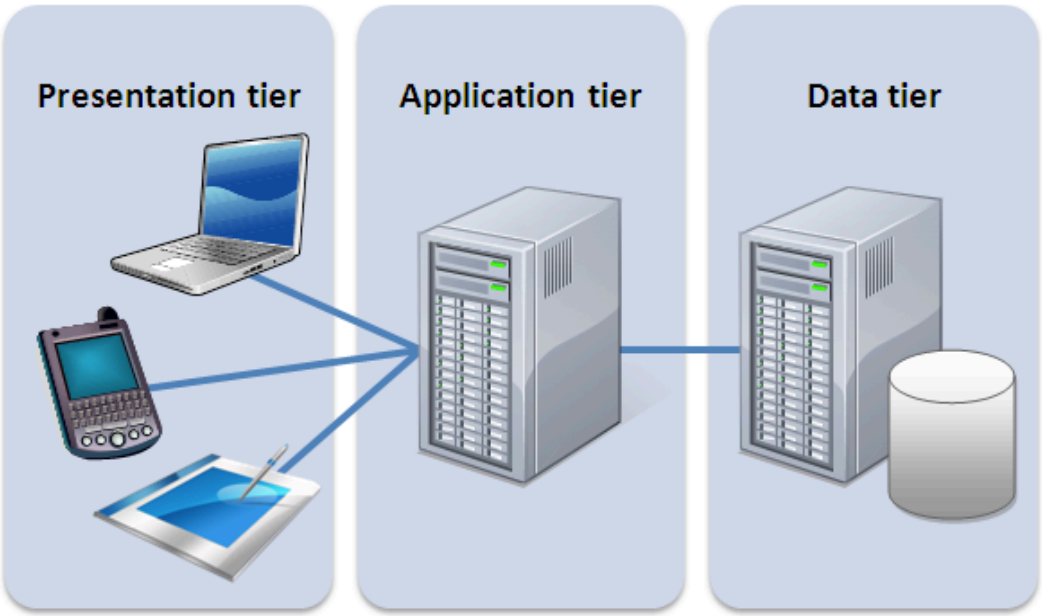


Figure 13 - 3 Tier Architecture

3.4 FRONT-END

The front-end of the system represents the presentation tier in the architectural model. Through this layer, the user will be able to interact with the system through a web-based user interface. Here, users will be able to register, log in and out, and create projects where they can configure their project specs and launch the Arduino script. As most of the functionality of this app happens on the back-end, paper prototypes were used for showing users a low fidelity representation of the user interface.

3.4.1 USE-CASE DIAGRAMS

In the Unified Modelling Language (UML) use-case diagrams are used to show the systems users (actors) and their interactions with the systems features (use-cases). The use-case diagrams below show the progression of the system functionality.

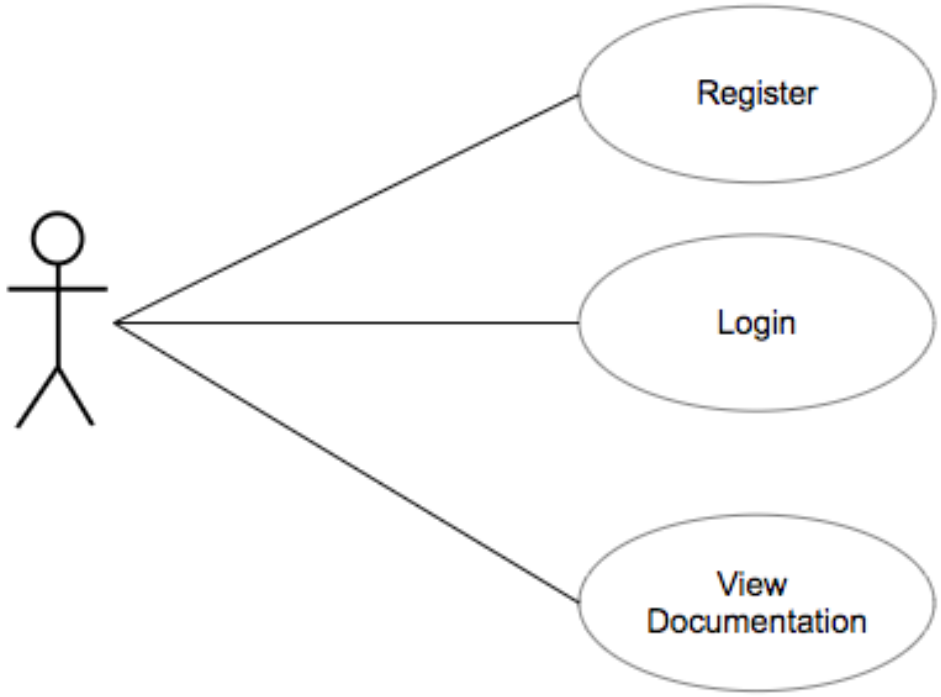


Figure 14 – 1st Iteration Use-case Diagram

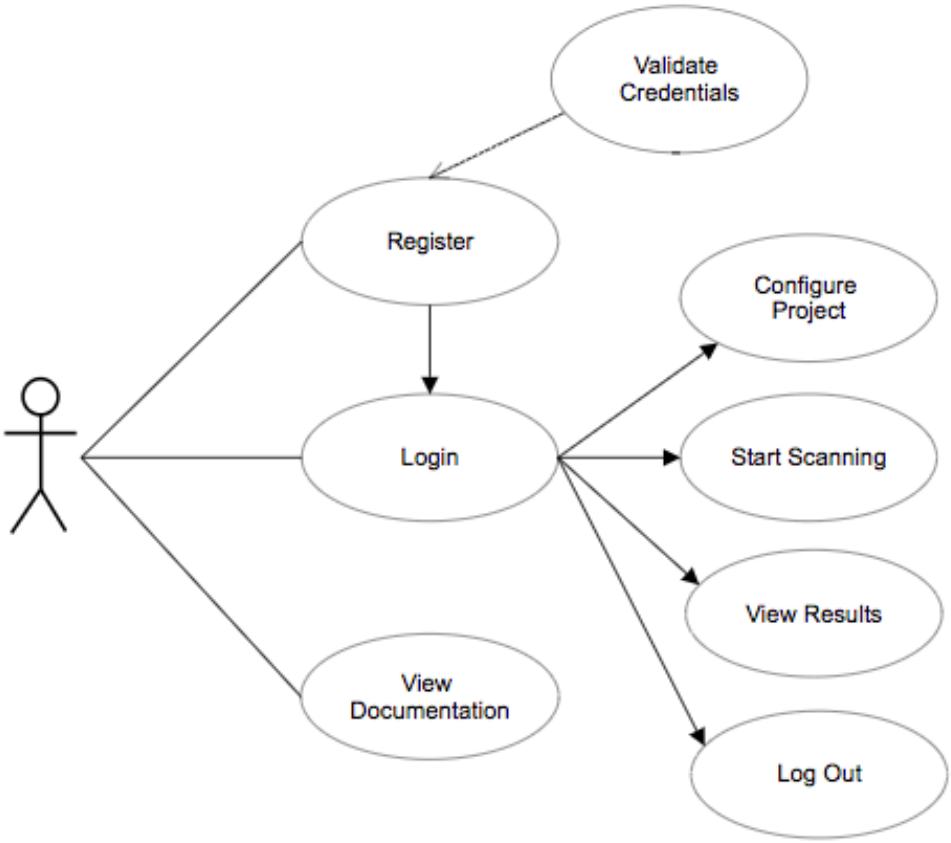


Figure 15 – 2nd Iteration Use-case Diagram

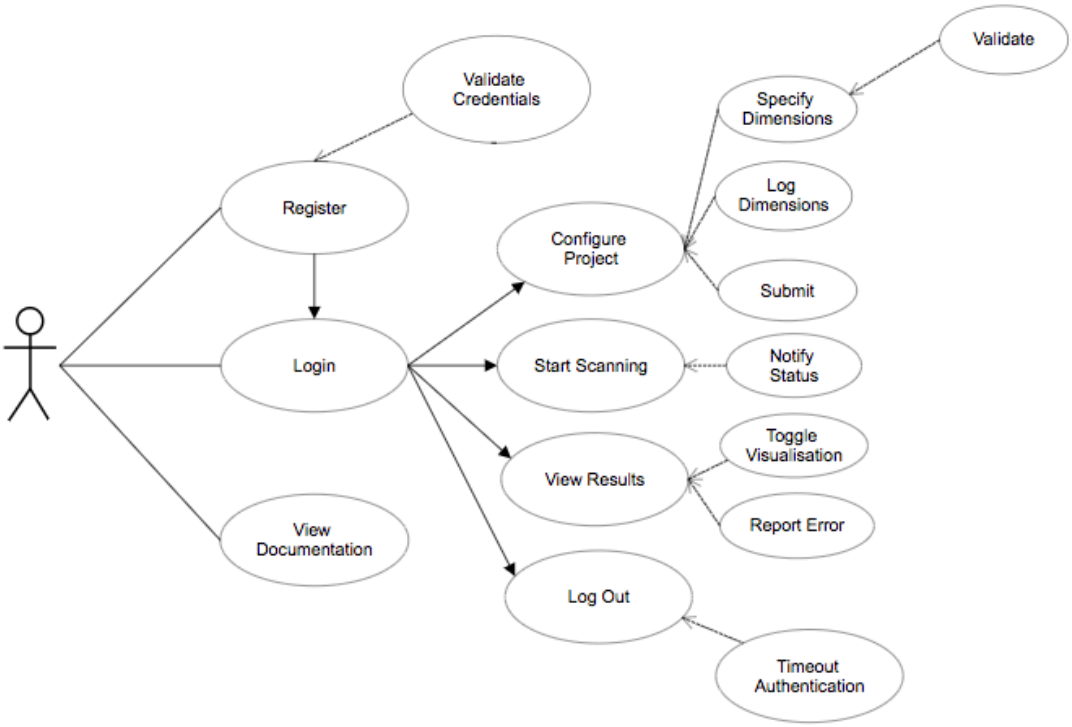


Figure 16 – 3rd Iteration Use-case Diagram

3.5 MIDDLE TIER

The middle tier or 'application' tier handles communication between the front and back-end in the application and provides the underlying functionality for users to make requests and receive responses from the server. The Model-View-Controller (Figure 17) is a software design pattern associated with RESTful web applications, and the middle tier can be thought of as the 'Controller'; the intermediary software layer that interacts with the model(back-end application) to update the view (user interface) on the front-end.

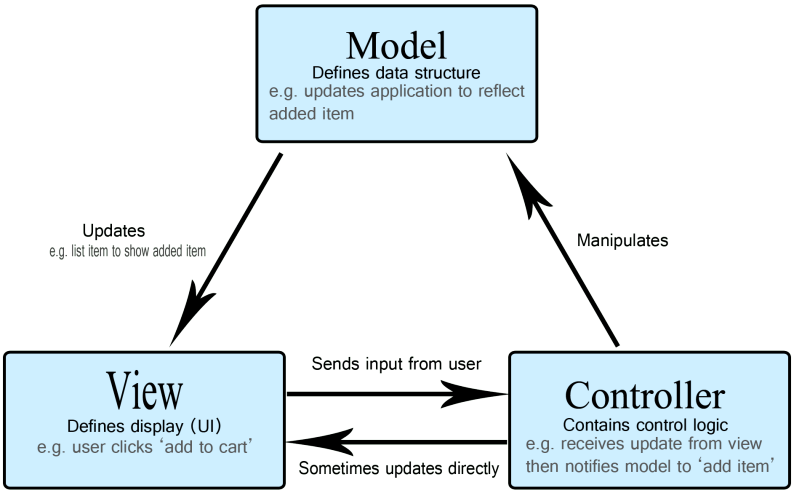


Figure 17

Sequence diagrams are another form of UML diagram, and can be helpful in describing how a group of objects work together, and in what order. PlantUML, a UML diagramming tool was used to create each of the iterations of sequence diagrams below.

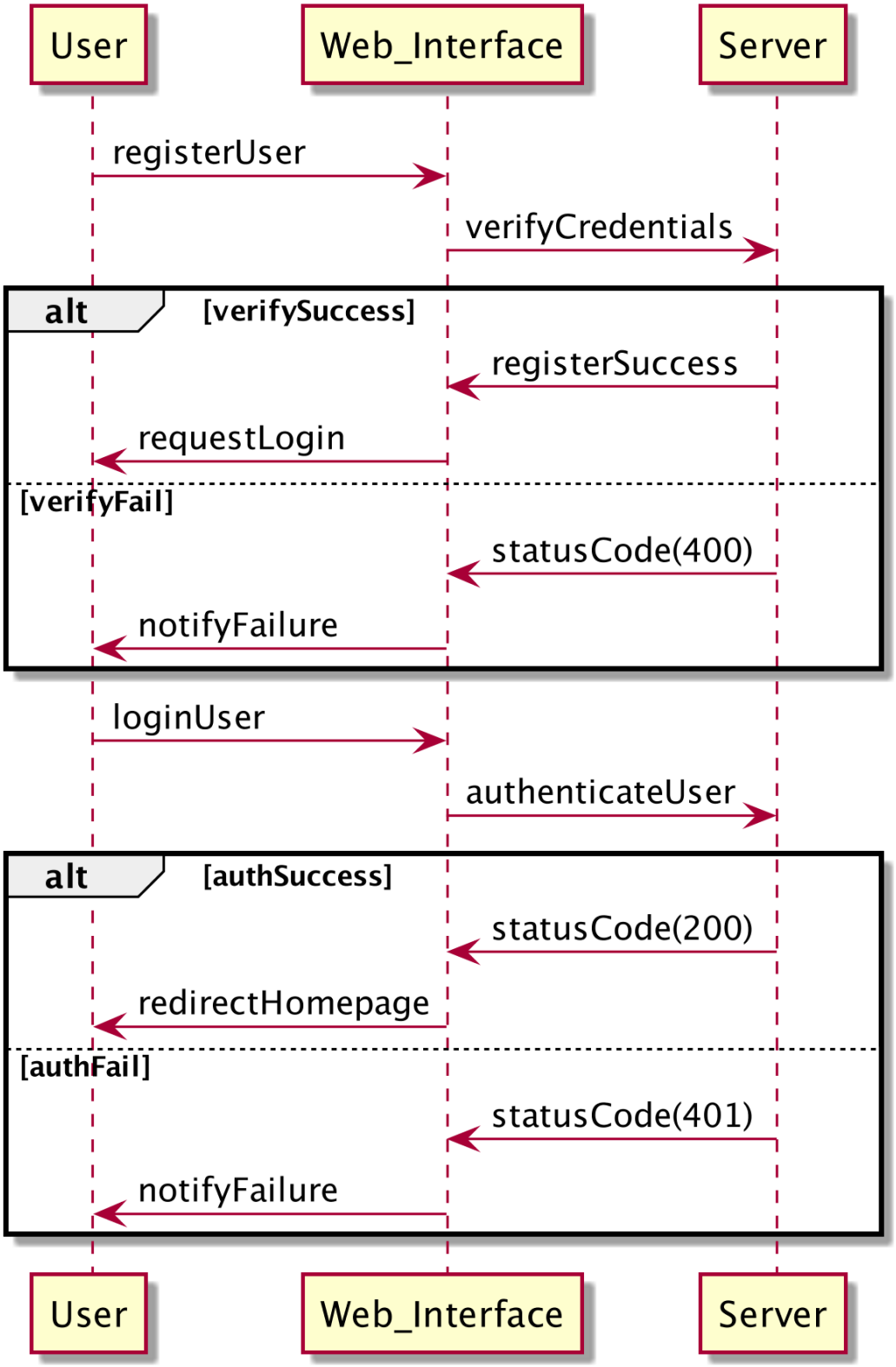


Figure 18 – 1st Iteration Sequence Diagram

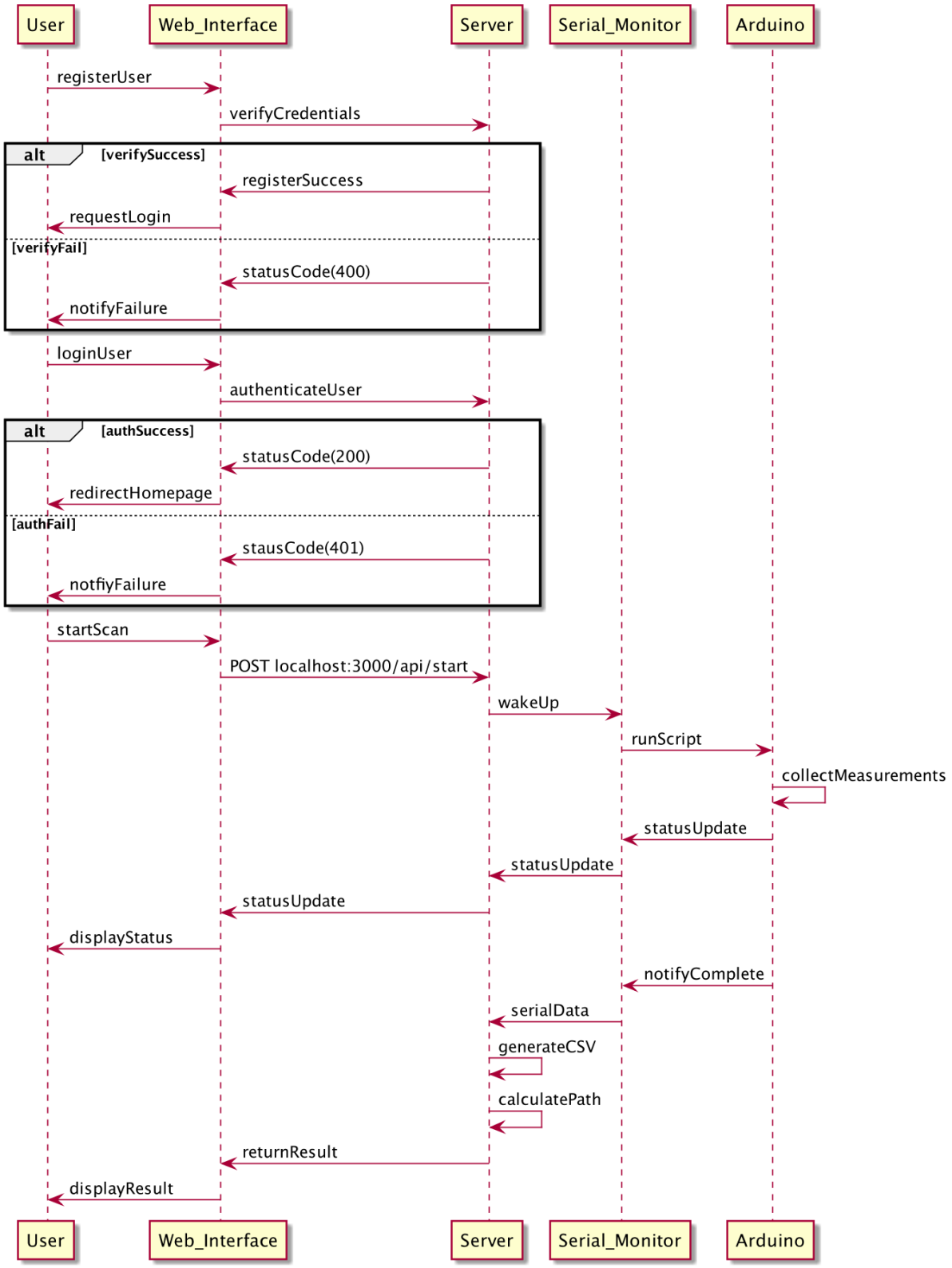
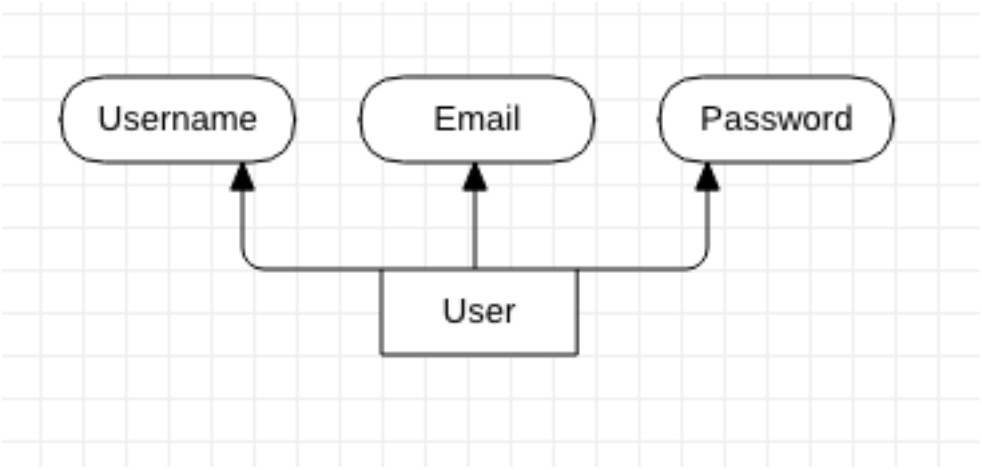


Figure 19 – 2nd Iteration Sequence Diagram

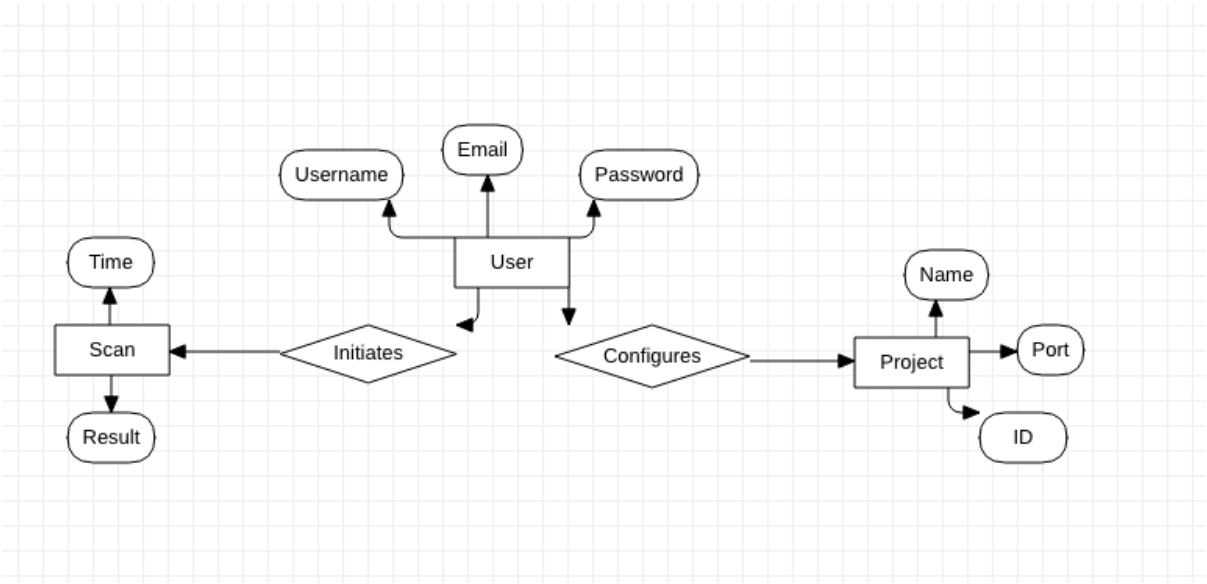
3.6 BACK-END

The back end is comprised of various components for data storage and processing. This is the data layer in the 3-tier architectural model and is where all non-transient user data such as emails and passwords will be stored. Two DBMS have been considered for data storage, SQLite and PostgreSQL. Both are compatible with Sequelize, a promise-based object-relational mapping tool which allows servers written in Python or Javascript to make database queries and perform C.R.U.D operations. A relational model was chosen as this application is not required to store a lot of data at this layer, and each entity represents an object with the same fields of data.

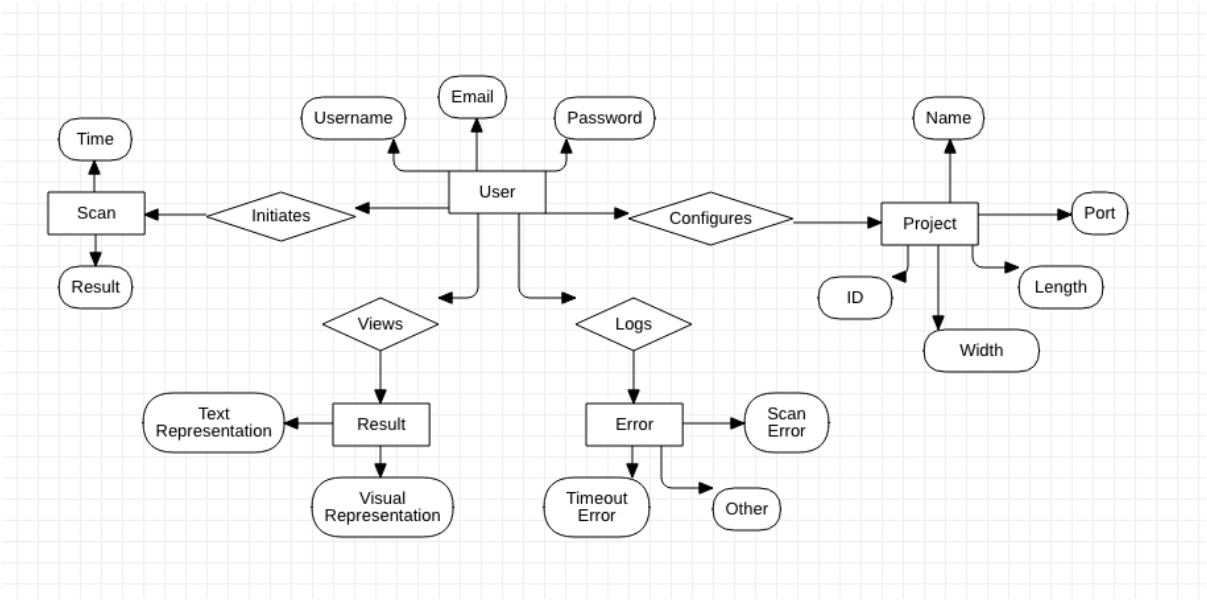
1ST ITERATION ERD



2ND ITERATION ERD



3RD ITERATION ERD



3.7 CONCLUSION

This chapter covered the design of the system, discussing the architectural model development methodology chosen, as well as a brief overview of the system's design from the front to the back end. The next chapter will cover the prototype development phase, building on many of the ideas and design choices outlined in this chapter.

PROTOTYPE DEVELOPMENT

CHAPTER 4: PROTOTYPE DEVELOPMENT

4.1 INTRODUCTION

This chapter will focus on the development process of this project, covering prototype development for user interfaces, middle tier software, and the back end. It was decided that this stage would be broken into two categories, user-focused prototypes and test-focused prototypes. User-focused prototypes mainly focus on the front-end user interface, such as register and login screens. Test-focused prototypes were developed for the purpose of identifying challenges that may occur when collecting sensor data and marshalling it to a format that would work with Python. This would guarantee that, before proper development begins, a reliable method has been identified for accomplishing communication between the Arduino circuit board and the computer where the server and data layer are located.

4.2 PROTOTYPE DEVELOPMENT

The first step of this stage was to set up a method of version control for this project. GIT, a distributed version control system was chosen as it provides a comprehensive suite of useful features for tracking changes made to files and integrates well with Visual Studio Code due to a number of helpful extensions. GIT also offers a web-based hosting service, Github, where files and folders which have been initialised by GIT can be accessed. Github offers the version control functionalities of GIT, along with a number of other features which make it easy to view the history of a project. This will provide a reliable web-based back up of the project should local files ever be lost and allow for experimentation without risk of compromising existing progress. The structure of the project is divided into two file systems, one folder containing front-end code such as HTML, CSS, Javascript, as well as ReactJS components, and a back-end folder containing server code, the

DDL and DML code for the DBMS, as well as the C++ script for the Arduino board. Finally, when initialising a GIT repository, a '.gitignore' file was included to avoid hidden files on macOS such as '.DStore' from being added to the repository.

4.3 FRONT-END

For the front-end, a prototype login (Figure 23) and home screen were developed using HTML, CSS, and Javascript. As this is just a prototype, it was not necessary to include React components at this stage. In order to create responsive web pages, Bootstrap was used along with CSS. Bootstrap is a CSS framework which contains Javascript-based design templates for forms, buttons, navigation and other interface components.

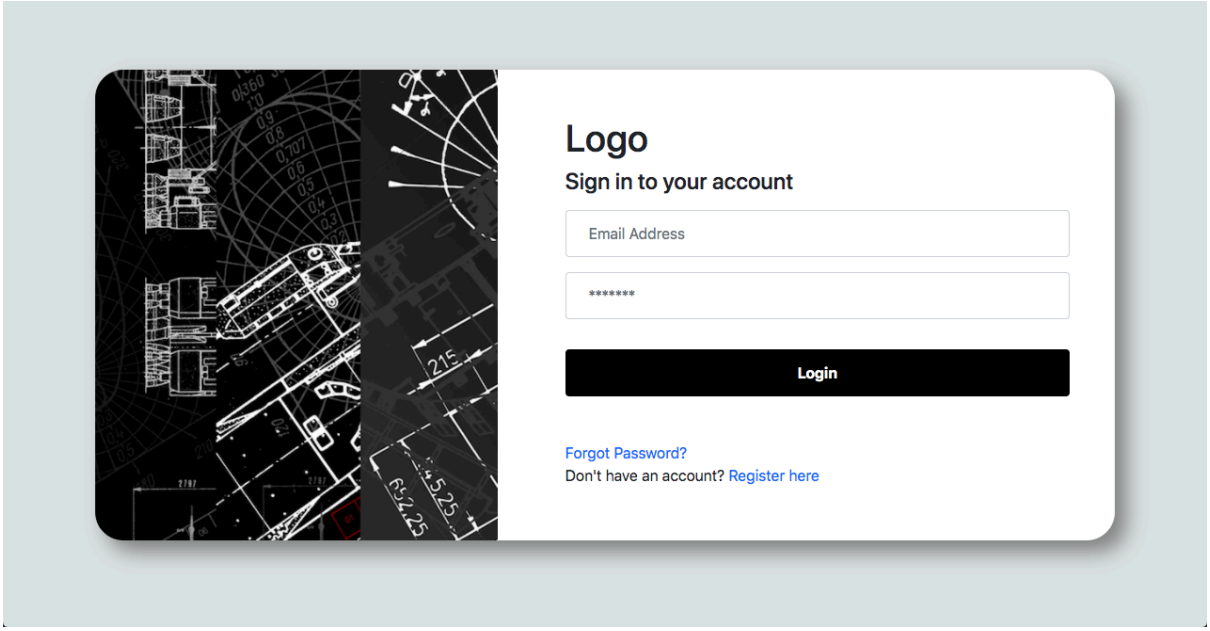


Figure 23 – Login Screen

As mentioned before, user interfaces have been designed with universal design principles in mind, including appropriate alternative labels and ARIA attributes where necessary, and high contrast between text and backgrounds. As the API endpoints have yet to be implemented, relative paths were used to allow navigation between pages at this stage. However, a Javascript file was created in order to write any promise-based requests, such as HTTP requests, in anticipation of later stages

of development which will involve connecting the front and back end.

```html
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap CSS -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sh
<link rel="stylesheet" href="style.css" type="text/css">

<title>Nav Project</title>
</head>
<body>
<section class="Form my-4 mx-5">
    <div class="container">
        <div class="row no-gutters">
            <div class="col-lg-5">
                <img src="image.jpeg" class="img-fluid" alt="logo">
            </div>
            <div class="col-lg-7 px-5 pt-5">
                <h1>Logo</h1>
                <h4>Sign in to your account</h4>
                <form>
                    <div class="form-row">
                        <input type="email" placeholder="Email Address" class="form-control my-3 p-4">
                    </div>
                    <div class="form-row">
                        <input type="password" placeholder="*******" class="form-control my-3 p-4">
                    </div>
                    <div class="form-row">
                        <button type="button" class="btn1 mt-3 mb-5">Login</button>
                    </div>
                    <a href="#">Forgot Password?</a>
                    <p>Don't have an account? <a href="/register.html">Register here</a></p>
                </form>
            </div>
        </div>
    </div>
</section>
```

4.4 MIDDLE TIER

An API routing table was created using markdown on Github, listing all of the necessary HTTP requests required for the app to perform its main functions. API calls are divided between authenticated and unauthenticated users. This ensures that a user must be logged in before accessing their projects and connecting to the Arduino Uno.

Operations on the auth resources

URI	Method	Auths?	Operation
auth/login	POST	No	Verify if the credentials of a user and return the User and a JWT token if he's ok
auth/register	POST	No	create an user create a JWT token for him and return the User and a JWT token
auth/logout	GET	Yes	Logout the user return status code.

Operations on the user ressources

URI	Method	Auths?	Operation
user/	GET	Yes	Returning all the users.
user/{id}	GET	Yes	Returning a single user.
user/id	DELETE	Yes	Remove a single user.
createProject/	POST	YES	Create a project for a single user
startScan/	POST	YES	Launch the Arduino program and collect sensor data

This is a tentative list of API methods which fulfils the basic needs of the high priority requirements, but more methods may be added later in development. This table will be referred back to when implementing the REST API with Node.js.

4.5 BACK-END

Back end prototype development was conducted by testing three key elements. Firstly, a simple Node.js server was created and tested using the 'Nodemon' package, which allows developers to make alterations to server code and see changes and errors in real time. A few API routes were defined for testing purposes in order to ensure that the server wasn't experiencing difficulty navigating file paths and fetching resources.

```
const path = require('path');
// Set up relative paths for static server
server.use(express.static(path.join(__dirname, '/dist')));

server.use(passport.initialize());

server.get("/", (req, res) => {
    res.json({ message: "Server is running!" });
});

/**
 * Start server
 */
server.listen(port, () => {
    console.log(`Server running on port ${port}!`);
});

module.exports = server;
```

Figure 24 - Server code

As mentioned in a previous section, the primary focus of back end prototype development would be testing the compatibility of Python with the Arduino IDE, and following that, Python with Node.js. A simple Arduino script was written which activates the HC-SR04 ultrasonic sensor at timed intervals and writes sensor data to the serial monitor in CSV format. This script initializes several variables which are necessary for

controlling the sensor, such as the baud rate and the digital pins that are connected to the trigger and echo pins of the sensor.

```
// Function for calculating the distance measured
int calculateDistance(){

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro s
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // Reads the
  distance= duration*0.034/2;
  return distance;
```

Figure 25 – Arduino Measurement Algorithm

Once this had been completed, a Python program was written which attempts to read data from the serial monitor as it is being written. This was done by using the previously mentioned PySerial library. The program reads in an arbitrary number of sample measurements (270) before writing them to a CSV file. This test proved to be successful, as can be seen from the collected sensor readings below.

	A	B	C
1	Index	Measurement in cm	Angle in Degrees
2	1	257cm	1
3	2	255cm	5
4	3	258cm	9
5	4	257cm	13
6	5	257cm	17
7	6	258cm	21
8	7	257cm	25
9	8	259cm	29
10	9	257cm	33
11	10	45cm	37
12	11	47cm	41
13	12	230cm	45
14	13	-1cm	49
15	14	49cm	53
16	15	49cm	57
17	16	48cm	61
18	17	47cm	65
19	18	44cm	69
20	19	37cm	73
21	20	36cm	77
22	21	33cm	81
23	22	31cm	85
24	23	34cm	89
25	24	29cm	93
26	25	-1cm	97
27	26	-1cm	101

Figure 26 - Sensor Data

```python
def print_serial():

    arduino_port = "/dev/cu.usbmodem1421" # Port connected to arduino via USB
    arduino_baud = 9600 # Baud rate data in transmitted between arduino and serial port
    fileName = "measurement_data.csv" # New file created each time
    maxSamples = 270 # This number is based on trial and error
    print_labels = False # Current line being printed

    # ----------------------------------------------------------------------------------------------------

    ser = serial.Serial(arduino_port, arduino_baud)
    print("Connected to arduino port" + arduino_port)

    if(glob.glob('**/*.csv', recursive=True)):
        print("File already exists")
        time.sleep(1)
        print("Deleting...\n")
        time.sleep(1)
        os.remove(glob.glob('**/*.csv', recursive=True)[0])

    file = open(fileName, "w")              # Create new CSV file
    print("Created CSV file")

    # ----------------------------------------------------------------------------------------------------

    line = 0

    # Should run 270 times until all measurements have been written
    while line <= maxSamples:

        if print_labels:
            if line == 0:
                print("Printing Column Headers")
            else:
                print("\t\tLine " + str(line) + ": writing...")
        getData = str(ser.readline())
        data = getData[2:][:-5]
        print(data)

        file = open(fileName, "a")
        file.write(data + "\n")
        line = line + 1

    print("Data collection complete!")
```

Figure 27 – Python script

Finally, a prototype communication layer between the server and python program was developed, using Node.js to spawn a child process after an API call. This shows that back end processes should work despite the heterogeneity of programming languages being used. It was also necessary to prove that it was possible to send as many parameters as necessary to Python through Node.js.

Figure 28

4.6 CONCLUSIONS

This chapter covered the development of front and back end prototypes. User interfaces were implemented using basic HTML, CSS, and Bootstrap. An API routing table was created for use in future development, and the critical back end functionalities were confirmed to work as intended. GIT was chosen the method of version control for the project; all prototypes have been pushed to a remote repository on Github and will serve as the foundation for the rest of the project's development. Due to the simplistic nature of the data layer for this project, SQL and DBMS development were not covered in this chapter, but a mock SQL schema was written in SQLite and PostgreSQL for future use.

The next chapter will explore testing and evaluation methods, as well as a general testing plan. These tests will be based on the core requirements defined in an earlier section, and will assist in finding errors, gaps, and missing requirements as development progresses.

TESTING AND EVALUATION

CHAPTER 5: TESTING AND EVALUATION

5.1 INTRODUCTION

This chapter will focus on a number of testing methodologies and testing frameworks which will be used to identify bugs and other issues in the software. A general plan for testing will also be defined, as well as a plan for evaluating the application once it has reached an appropriate stage of development and can be shown to potential users. The methods of testing will be chosen based on the requirements of particular software components and explained accordingly.

5.2 PLAN FOR TESTING

In order to remain consistent with the scrum agile development methodology chosen for this project, testing will be carried out throughout the project at the end of each sprint, followed by a period of evaluation towards the end of the development cycle, where the application will be made available to a select group of engineers who have agreed to co-operate during this stage. Tests to be carried out have been categorised according to relevant software and hardware layers and will require different methodologies and frameworks. The following section will address the method of choice for each of these layers as well as any frameworks chosen.

5.3 HARDWARE TESTING

In order to evaluate the accuracy of horizontal measurements collected and compiled from sensor data, as well as the accuracy of the pathing algorithm, unit testing[13] has been chosen as the testing methodology. The purpose of unit testing is to test individual functions or subroutines in order to ensure they produce an expected outcome. In the context of this project, this will involve taking a dataset of known measurements and correct paths and comparing them to the results of sensor data

[13] https://softwaretestingfundamentals.com/unit-testing/

and Python functions. PyTest was the obvious choice for a testing framework; comparing values can be achieved easily using ‘assert’ statements such as ‘isEqual()’. A log of test results can be viewed in the terminal by invoking a PyTest script, displaying successes and failures for each method in an easily readable format.

5.4 BACK-END TESTING

Mocha and Chai are testing frameworks designed specifically for Node.js projects. One of the advantages of using these frameworks is that they support the concept of ‘CI/CD’ (continuous integration/continuous deployment) by enabling developers to automate testing. Automated testing is helpful when developing web applications which implement C.R.U.D operations to a database, as any errors that might arise between the server and DBMS will immediately be identified upon running the server. Mocha and Chai are easily integrated into Node.js projects through the node package manager (npm) and by specifying the path of testing files in a ‘package.lock.json’ file. Normally, these tests are fired each time the server initially starts up, but with Nodemon, tests are fired each time changes are made to server code. This method of testing will reduce the risk of bugs going unnoticed for long periods of time.

5.5 MIDDLE-TIER TESTING

Testing the middle layer of the application will involve the use of a third-party platform called Postman. Postman is an API platform which simplifies the process of building and testing various kinds of API’s. Visual Studio Code now offers an extension which integrates features of Postman into the editor, making it easier than ever before to test API methods. This platform allows developers to build HTTP requests such as GET and POST requests, and include various additional information, such as headers, parameters, web tokens, and more. This can be delivered in raw format, or in commonly marshalled formats such as JSON or XLT. The flexibility of the platform will enable changes to be made in the way data is transmitted from client to server and vice versa, should the need arise.

5.6 TEST PLAN

Test No.	Description	Expected Outcome	Pass?
1	Can the user access the web pages from URI?	User can access relevant web pages	
2	Is user authenticated after login?	User is given authenticated privileges on login	
3	Can user login with invalid credentials?	Validation should appear for any incorrect credentials entered	
4	Can user login with correct details?	User should be redirected to homepage	
5	Can user register?	User can create an account	
6	Can user delete all account information?	All data stored in DBSM related to the user is dropped	
7	C.R.U.D operations work correctly?	Database queries and updates work as intended	
8	Can the user create a new project?	New project component is created after selection	
9	Are configurations sent correctly?	Sends configurations as parameters to server	
10	Arduino program can be launched from browser	Client can invoke Arduino Uno and Python script	
11	Arduino program launched with no port specified	Status code, error message returned	

12	No circuit board connected on launch	Status code, error message returned	
13	Is sensor data accurate?	Sensor data consistent with known outcome	
14	Is pathing algorithm accurate?	Python algorithm correctly calculates safe path	
15	Can user see update of scan?	User is updated as processes are being executed	
16	Can user see result?	Result of scan and pathing algorithm is visible to the user when complete	
17	Issuing Error	Errors logged by the user are cached on the server side	
18	Attempting to relaunch during execution	Server should reject any subsequent requests to launch the Arduino program while it is running	

5.7 EVALUATION PLAN

Evaluating this system and its features will be equally as important and testing them. This will enable us to determine the success and progress of the project and identify gaps in requirements which may have been overlooked. An important characteristic of this application was that it made prototyping and developing Arduino projects easier and less stultifying for developers. Getting feedback from potential users and experienced developers in this field will shed light on how well realised these ambitions have been.

As previously mentioned in an earlier section, user interfaces and interactable components were designed with usability guidelines such as Nielson's Heuristics in mind, as well as the 7 universal design principles. While online tools exist, which are capable of evaluating the conformity of web applications with these guidelines, it will be important to hear the opinions of average users, who may measure the usability and responsiveness of the application more or less favourably than software tools.

5.8 CONCLUSION

In this chapter, a plan was developed for the testing and evaluation process of this project, with explanations given for each methodology and framework selected. The full test plan consists of unit tests for back end processes and measurement algorithms, automated tests for the data layer, and the use of a third-party API platform to test the REST API. The final section on evaluation recalled the importance of usability from previous sections, as well as collecting user feedback in order to assess the success of the project once it has been completed.

ISSUES AND FUTURE WORK

CHAPTER 6: ISSUES AND FUTURE WORK

6.1 INTRODUCTION

There are a number of plans for future development of this application covered in the next section. This chapter will also highlight some of the challenges and risks involved with the project, such as a lack of familiarity with certain frameworks and libraries, security issues, and the unreliability of hardware components. During these sections an effort will be made to identity and anticipate issues that may hinder the progress of development.

6.2 ISSUES AND RISK

The challenges that are unresolved in the project thus far are as follows:

- Lack of experience with complex architecture especially when working with hardware modules
- Ensuring data integrity and security when handling C-S operations
- Complexity of asynchronous communications with many working parts
- Lack of familiarity with integrating SQLite with Node.js
- Availability of organisations and people who can give feedback
- Uncertainty about requirements for different operating systems especially when using ephemeral ports for external device connections

How the author plans to approach these challenges are as follows (respectively):

- Build and test a simple app which involves the participation of all components
- Conduct additional research on web security resources for Node.js and implement hashing for stored user information
- Test a number of small prototype web applications with promise or callback-based functionality
- Complete an online course which covers database integration with Node.js
- Test prototype application on a number of different systems such as Windows, macOS, and popular Linux distributions

The risks that the author faces in the project are as follows:

- Malfunctioning or broken hardware, especially if it is difficult to reorder parts
- Wi-Fi could hinder the demoing of the interaction between the browser interface and back end processes
- Failing to implement all of the requirements specified for the project
- Scope of the project is either too big or too small

How the author plans on approaching these various risks are as follows:

- Two copies of all essential components have been obtained; i.e two controllers, two sensors, two motors
- For demoing purposes, the app will work with a server which runs on an ephemeral local port
- Stick to scrum agile methodology, complete small manageable tasks in time allotted for each sprint

- Address any concerns about project scope with assigned supervisor

6.3 RISK REGISTER

Below is a risk register which details all of the risks involved in this project in a more readable format. Risk registers are useful as project management tools, because as well as helping to identify risks, they provide more detailed descriptions and methods of mitigation, which can be referred back to if certain risks actualizes.

ID	Description of risk	Method of risk management	Risk evaluation
1	Loss of progress, uncommitted changes lost	Loss prevention and reduction	LOW
2	Malfunctioning hardware	Pre-emptive measures	LOW
3	Package loss between client, server	Retention	MODERATE
4	Unable to complete project in time	Pre-emptive measures	MODERATE
5	Issues with the scope of the project	Avoid	MODERATE
6	Wi-Fi issues occur while demoing	Pre-emptive measures	LOW
7	Delays in the project due to illness/ other hindrances	Accept	MODERATE

8	Security vulnerabilities in DBMS	Control	HIGH
9	Difficulty getting many languages to work together	Accept	HIGH

6.4 PLANS AND FUTURE WORK

The plan for the project can be seen in the GANTT chart below and will remain consistent with the philosophy of the agile development methodology. Once the project has been completed, another GANTT chart will be created and compared to this one in order to measure the management and progress of the project over the course of the entire development cycle.

The initial set of sprints will prioritise the implementation of a working REST API in order to quickly enable communication with the front and back end code developed for the prototyping stage. Once this has been completed, sprint tasks will shift to focus on implementing a pathing algorithm and other processes related to the navigation features of this project.

As the project continues, the author of this document will continue to engage with software and engineering professionals in order to get feedback on the usability and accessibility. Concerning the future of the project beyond the immediate point in time, a number of plans for the further development of this project have been included below.

One of the considerations made when deciding to separate this project into separate software layers, was to create a tool which would be malleable and easy to refactor in later development. This means that, hypothetically, new UI features could be added to make the user experience better with minimal changes needed to the other layers. For example, a real time visualisation of the Arduino board and other hardware components completing their tasks could be displayed to the user while waiting on the server's response. This would make the waiting process less

tedious. Another future improvement might implement linear interpolation as mentioned earlier. This would enhance sensor data and, in the grander scheme of things, even allow for 3-D environments to be approximated from horizontal scans.

6.5 GANTT CHART

BIBLIOGRAPHY

1. MIYbot | Hackaday.io [Internet]. [cited 2021 Nov 24]. Available from: https://www.hse.ie/eng/services/list/4/mental-health-services/
2. Ultrasonic Glasses for the Blind – Arduino Project Hub [Internet]. [cited 2021 Nov 24]. Available from: https://create.arduino.cc/projecthub/gardnertech/ultrasonic-glasses-for-the-blind-142156
3. Arduino RADAR model with Ultrasonic Sensor Servo & LCD [Internet]. [cited 2021 Nov 25]. Available from: https://create.arduino.cc/projecthub/gardnertech/ultrasonic-glasses-for-the-blind-142156
4. Project 1: 2WD Obstacle Avoiding Robot [Internet]. [cited 2021 Nov 25]. Available from: https://create.arduino.cc/projecthub/gardnertech/ultrasonic-glasses-for-the-blind-142156
5. Blink IoT platform: for business and developers [Internet]. [cited 2021 Nov 25]. Available from: https://blynk.io/
6. UNO R3 | Arduino Documentation [Internet]. [cited 2021 December 4]. Available from: https://docs.arduino.cc/hardware/uno-rev3
7. US-100 Ultrasonic Distance Sensor [Internet]. [cited 2021 December 4]. Available from: https://www.digikey.com/catalog/en/partgroup/us-100-ultrasonic-distance-sensor-3v/84977
8. PySerial Documentation [Internet]. [cited 2021 December 10]. Available from: https://pyserial.readthedocs.io/en/latest/
9. Express/Node Introduction [Internet]. [cited 2021 December 10]. Available from: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

10. Policy and Legislation | Centre for Excellence in Universal Design [Internet]. [cited 2021 December 18]. Available from: https://universaldesign.ie/what-is-universal-design/policy-and-legislation/
11. Waterfall Methodology – an overview | ScienceDirect Topics [Internet]. [cited 2021 December 20]. Available from: https://www.sciencedirect.com/topics/computer-science/waterfall-methodology
12. What is Agile Methodology? [Internet]. [cited 2021 December 20]. Available from: https://www.digite.com/agile/agile-methodology/
13. Unit Testing – SOFTWARE TESTING Fundamentals [Internet]. [cited 2021 December 27]. Available from: https://softwaretestingfundamentals.com/unit-testing/