



BB: TODDLER BUDDY

Final year project report



Shane Higgins

Galway-Mayo Institute of Technology

2019/2020

Project supervised by

Des O'Reilly

Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Hons) in Software & Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

Acknowledgements

I would like to thank Brian Costello our technician who made sure that everyone had the parts they needed and did his very best to accommodate us.

I would like to thank Stephen McIntyre who printed the 3D parts for the ultrasonic sensors and the pi camera.

I would also like to thank Des O'Reilly for supporting me throughout the year.

1 Contents

1Summary	5
2Introduction	6
3Poster	7
4Project Architecture	8
5Architecture Diagrams	9
6Flowchart for Android App	10
7Projected Project Timeline.....	11
8Project Planning and research	12
9Project	13
9.1Raspberry Pi and Pi Camera	14
9.2Ultrasonic Sensor	15
9.3DC Motors	16
9.4Android App	17
9.4.1Login app	17
9.4.2Firebase	20
9.4.3Setting up Firebase	20
9.4.4Main Menu.....	21
9.5Project Integration	25
10Problem Solving Methods.....	26
11Project issues / complications	27
12Project Reflection.....	28
13Reference	29

1 Summary

BB: The Toddler Buddy is my final year project. BB, short for Babies Buddy is a small interactive robot. It is designed to keep toddlers busy and entertained when they are learning to crawl. It can also act as a baby monitor through an android app if the parents were in a different room. This allows parents to go about doing daily tasks while knowing that BB is keeping the baby occupied and letting them keep an eye on the toddler. The idea of this project is to create a buddy for toddlers to help parents catch a break.

The project is a culmination of electronic and software knowledge that has been gained throughout my time in GMIT.

BB is built with a Raspberry Pi 4 at its core. It has a Pi Cam attached to the case in a 3-D printed holder. The Raspberry Pi comes with a 40 pin GPIO header which is being used to control 2 DC motors and three ultrasonic sensors that are attached to the chassis the Pi sits on.

The obstacle avoidance functionality of the project has been implemented using C programming language. The monitoring element that allows the parents to view their child via Android application, was setup by using RTSP, Dataplicity and Java (app development).

Two directions were taken to stream the video to the application. First one uses Dataplicity which outputs a video stream to a web page that can be opened by pressing a button that is linked to this site. Second one uses RTSP (real-time streaming protocols) which allows for streaming the video directly to the application. Through this report I will talk about the steps, processes, methods and outcomes of my project.

1. Project Poster
2. Project Architecture I will discuss the hardware and software used
3. Architecture Diagrams will show how my project is connected
4. Flowcharts and project timelines
5. Break down of the project to its software and hardware components
 - 5.1. Raspberry Pi and Pi Cam
 - 5.2. Ultrasonic sensor set up and implementation
 - 5.3. Motors and motor control
 - 5.4. Android app development
6. System integration and how I integrated everything
 - 6.1. Successes
 - 6.2. Failures
7. Project
8. Problem solving
9. A reflection on the project and its outcomes

2 Introduction


BB is a small interactive robot that helps babies learn to crawl or to keep those already crawling occupied. It lends an extra helping hand to the parents by giving them the freedom to focus on different things around the house, while still being able to keep an eye on the child and an extra helping hand to mums and dads, while giving their parents the freedom to do stuff around the house, but still being able keep an eye on them

BB sits on the floor at a distance of 4 feet in the baby's field of view, with the idea of enticing the baby and encouraging it to crawl after it.

Once the baby starts to move towards BB, one of the three ultrasonic sensors, placed around BB's circumference will detect that something has entered BB's 30 cm safe zone and send it away. On top of BB is a Pi cam that allows parents to keep an eye on their baby, via an android app developed to allow monitoring while in a different room or even when out of the house.

BB was built around a Raspberry Pi 4 running the Raspbian OS. Connected to the Raspberry Pi are three HC-SR04 ultrasonic sensors used for object detection the Raspberry Pi detects one of the three sensors being blocked and controls the two DC motors built into the chassis. Live video feed is available due to the presence of Pi cam on the top of the device. This can be viewed via Android application developed in Android Studio.

3 Poster



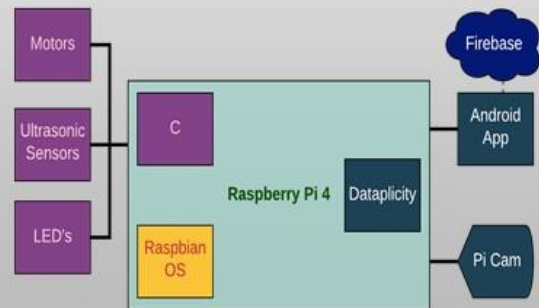
BB : Toddler Buddy

Introduction

BB is a small robot that will help kids at an early stage learn to crawl while giving their parents the freedom to do stuff around the house while keeping an eye on them.

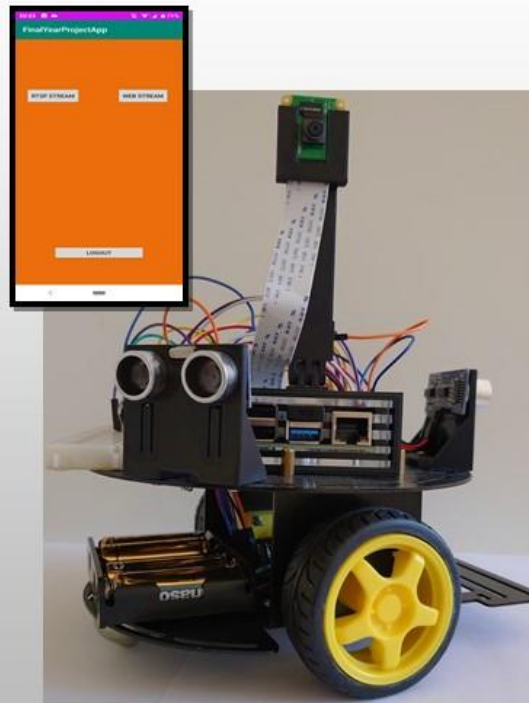
BB will be placed on the floor 4 feet from the toddler. The toddler will start to move towards BB at this point one of the 3 ultrasonic sensors will detect this and activate the motors and move BB away from the toddler

While this is happening there is a Pi cam that is streaming video back to an android app that allows the user to view the feed from anywhere in the world.




```
graph LR; Motors --- RPi4; Sensors --- RPi4; LEDs --- RPi4; RPi4 --- Dataplicity; Dataplicity --- AndroidApp; AndroidApp --- Firebase; Firebase --- PiCam; PiCam --- RPi4;
```

The diagram illustrates the system architecture. A central Raspberry Pi 4 box contains the Raspbian OS, C programming language, and Dataplicity. It is connected to three external components: Motors, Ultrasonic Sensors, and LED's. The Raspberry Pi 4 is also connected to an Android App, which in turn connects to a Firebase cloud database. The Firebase database is connected to a Pi Cam, which feeds back into the Raspberry Pi 4.



A photograph of the BB: Toddler Buddy robot. It is a small, black, two-wheeled robot with a Raspberry Pi 4 board mounted on top. The robot has two large, black, circular sensors (ultrasonic sensors) on its front. It is connected to a camera module and a motor module. The robot is shown on a white surface.



```
graph TD; BB[BB: Toddler Buddy] --- Software; BB --- Hardware; Software --- Dataplicity; Software --- Python; Software --- C; Software --- Android; Software --- Git; Software --- Networked; Hardware --- RaspberryPi4[Raspberry Pi 4]; Hardware --- Motors; Hardware --- LEDS; Hardware --- Camera; Hardware --- UltrasonicSensor[Ultrasonic Sensor];
```

The diagram shows the project structure. The main project is 'BB: Toddler Buddy', which is divided into 'Software' and 'Hardware'. The 'Software' section includes 'Dataplicity', 'Python', 'C', 'Android', 'Git', and 'Networked'. The 'Hardware' section includes 'Raspberry Pi 4', 'Motors', 'LED'S', 'Camera', and 'Ultrasonic Sensor'.

Results

I designed, developed and built a working prototype that can avoid Toddlers while streaming a live stream back to an android app that I also developed.

Conclusion

BB is a sleek companion for any toddler. It uses a mix of software and electronic principles. I incorporated real world technologies that allow users to monitor what a toddler is getting up to while at home and when you are out. The android app offers the user security. I really enjoyed working on this project and would like to see what other routes this technology could go down.

Shane Higgins G00339730
B Eng. Software and Electronic Engineering 2020
GALWAY-MAYO INSTITUTE OF TECHNOLOGY

Designed and created by Shane Higgins

4 Project Architecture

I built my project using a Raspberry Pi 4 as my development board. A Raspberry Pi is a powerful pocket-sized computer that can run its own OS (operating system) with a fully function GUI (Graphical user interface). It runs a Quad core cortex-A72 64-bit SoC @1.5GHz along with 4gb of SDRAM.

I am making use of the 40 pin GPIO header present on the board as well as the in-built Wi-Fi. I am using two out of 4 USB ports for peripheral attachments, the HDMI port, the 2-lane MIPI CSI camera port for attachment of the camera and the micro SD card holder with and SD card running Raspbian Buster OS. I am also using the WiringPi library.

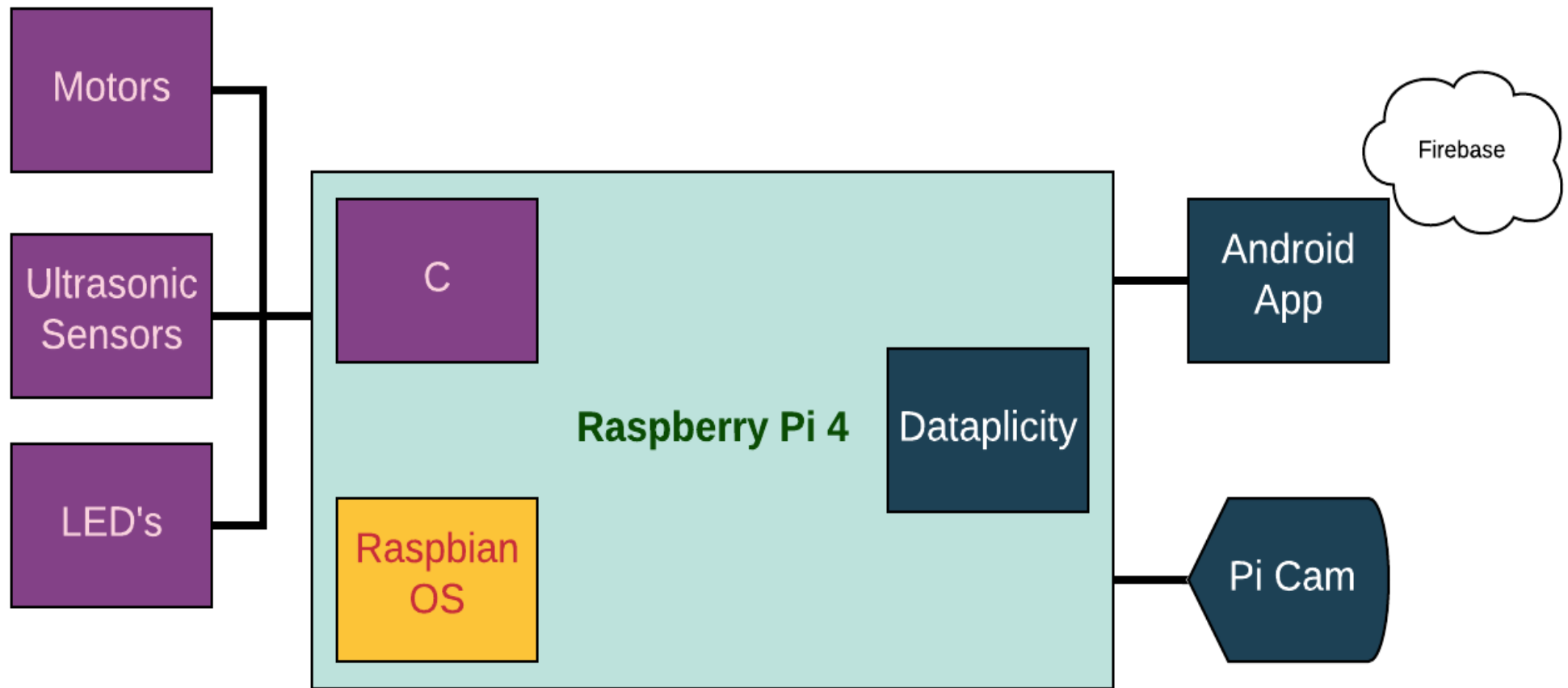
Hardware attached to the Raspberry Pi includes two DC motors and three ultrasonic sensors. The DC motors are built into a chassis which has the Pi sitting in the centre. The ultrasonic sensors are placed evenly around the chassis and held in place with 3D printed holders. The Pi also has a 3D printed attachment that holds the Pi camera in place.

The Android app was developed in Android Studio using Java. There is a login page which stores a user's information for authentication using firebase (this will be covered later). There are two different types of video streaming, one on the local network and one on the internet which uses Dataplicity and a video streamer called Hawkeye.

The app is built around intents to create new activities and link these to java files.

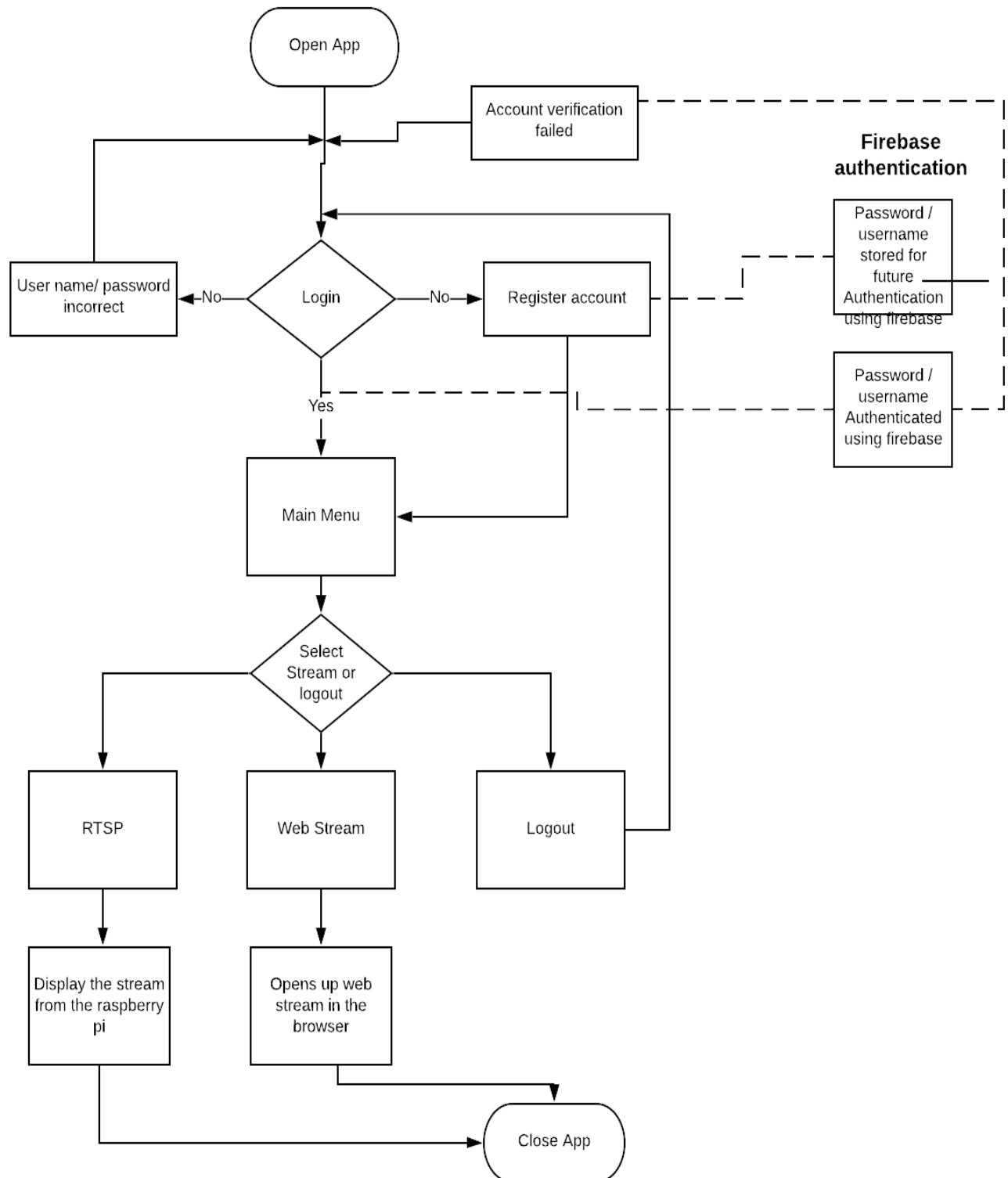
While working on the project I encountered new technologies and software. All of which are discussed below.

5 Architecture Diagrams

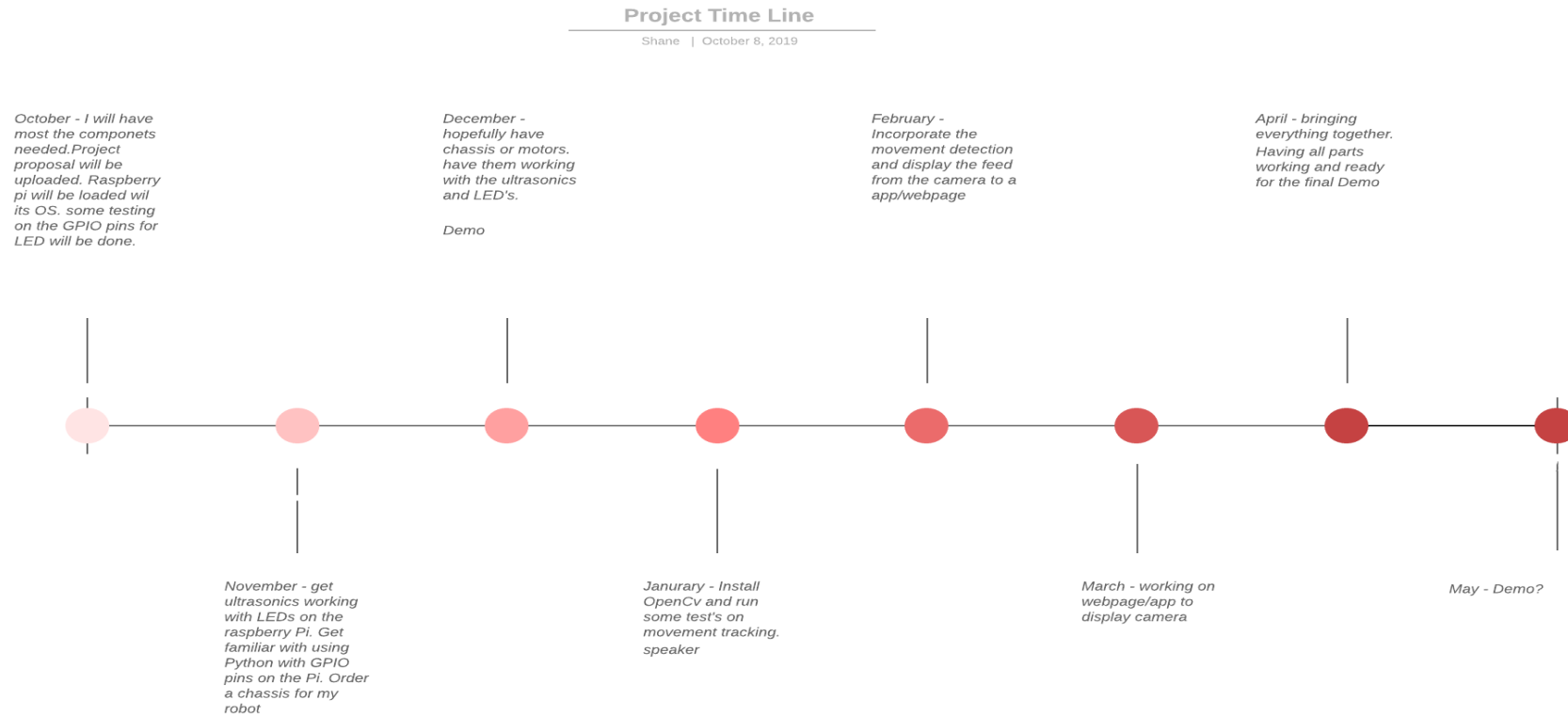


6 Flowchart for Android App

FinalYearProjectApp



7 Projected Project Timeline



8 Project Planning and research

The planning of the project was done at the beginning. The project was broken down into 7 sections. Each of these revolved around the hardware in the project, the software, the integration, and the documentation.

Once I had a project idea, I started to research the parts I needed. To get started I got a Raspberry Pi, ultrasonic sensors and two DC motors for testing.

Before I could start testing the sensors and motors, I needed to first do some GPIO testing on the pi and to get WiringPi installed. Blink was the first example I ran. This showed me that it was installed correctly, and I could move to the next stage of testing.

Ultrasonic and motor testing was next, and it was done on the workbench with one sensor and one motor as I did not have the chassis available yet. In the original plan I was to use Python, but I decided to switch to C as this is what I knew.

Up until Christmas a lot of work was put into the hardware aspect and the research of possible technologies to use and incorporate into the project.

I installed OpenCV and did some research on developing an app. While testing out OpenCV I decided to move away from it as I did not think it would add anything to the project in terms making my project stand out.

With the decision made to drop OpenCV I began working on my app. The app needed to have video streaming capabilities, so I began looking at various methods that would let me stream from the Pi to an app.

I ran into several issues while trying to set up the stream one of those being lagging issues. I decided to keep this in the project because I did spend a bit of time trying to get it to work. I began exploring other options. One of these was using Python/HTML which worked perfectly on the local network, but I wanted to have something that streamed to the internet and I came across Dataplicity.

The second hurdle was integrating all the different parts of the app, the control of the motors and the ultrasonic sensors functionality in to a working prototype.

9 Project

There were three main aspects in my project:

- 1) Hardware
- 2) Software
- 3) Integration

I wanted my project to be something that I built and programmed myself giving me a challenge.

The hardware was as follows:

1. Laptop
2. Raspberry Pi 4
3. Pi camera
4. DC Motors built into chassis
5. Ultrasonic sensors
6. Android Phone
7. 3D printer ultrasonic sensor holders & Pi camera holder
8. Breadboard
9. Wires
10. LED's

Software

1. Windows laptop
2. Raspbian OS
3. C
4. Android Studio/Java
5. Python/html
6. RTSP (Real time streaming protocols)
7. Dataplicity
8. Github
9. Word
10. Lucidicharts

Integration:

- Building the Chassis and connecting the Raspberry Pi
- Streaming live video from the Raspberry Pi to laptop and an android app using the real time streaming protocol.
- Streaming a live feed on the web using Dataplicity which allows for a local video or webpage hosted on the Raspberry Pi to be hosted on a URL provided using a wormhole.

9.1 Raspberry Pi and Pi Camera

What is a Raspberry Pi?

A Raspberry Pi is a small card sized development board. Very much like a desktop computer or laptop except much smaller. It is powerful and capable of being used in a wide range of ways. For my project it is being used as an embedded board for obstacle avoidance robot that is streaming a live video feed to both the internet and an app.

I chose to use a Raspberry Pi as it is an extremely versatile and powerful piece of hardware that can be used in many ways.

The first thing I did when getting started on the Pi was to install the WiringPi library so I could control the GPIO pins. WiringPi was developed solely for the Raspberry Pi by Drogon to interface with the pins on the Pi. Sadly, though after August 2019 will not be releasing any new versions.[3]

```
pi@raspberrypi:/ $ gpio readall
```

-----Pi 4B-----											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1	2		5v			
2	8	SDA.1	IN	1	3	4		5v			
3	9	SCL.1	IN	1	5	6		0v			
4	7	GPIO. 7	IN	1	7	8	1	TxD	15	14	
		0v			9	10	1	RxD	16	15	
17	0	GPIO. 0	IN	0	11	12	0	GPIO. 1	1	18	
27	2	GPIO. 2	IN	0	13	14		0v			
22	3	GPIO. 3	IN	0	15	16	1	GPIO. 4	4	23	
		3.3v			17	18	1	GPIO. 5	5	24	
10	12	MOSI	IN	0	19	20		0v			
9	13	MISO	IN	0	21	22	0	GPIO. 6	6	25	
11	14	SCLK	IN	0	23	24	1	CE0	10	8	
		0v			25	26	1	CE1	11	7	
0	30	SDA.0	IN	1	27	28	1	SCL.0	31	1	
5	21	GPIO.21	IN	1	29	30		0v			
6	22	GPIO.22	IN	1	31	32	1	GPIO.26	26	12	
13	23	GPIO.23	IN	0	33	34		0v			
19	24	GPIO.24	IN	0	35	36	1	GPIO.27	27	16	
26	25	GPIO.25	IN	0	37	38	1	GPIO.28	28	20	
		0v			39	40	1	GPIO.29	29	21	
-----Pi 4B-----											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	

Above shows the layout configuration of the pins on the Raspberry Pi

The first test I ran was Blink. I had to build a small circuit with an LED and connect it to a GPIO. From here I moved on to test out the motors and the ultrasonic sensors.

The GPIO's are used to take in information from the Ultrasonic and use that to control two DC Motors.

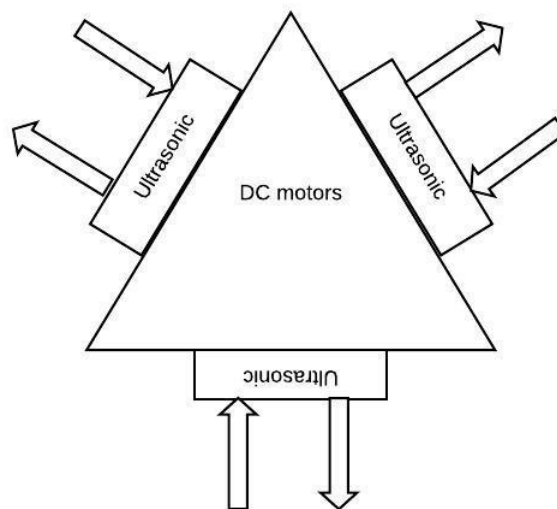
There is also a 2-lane MIPI CSI camera port which has a Pi cam attached to it for video streaming. The camera is being used to stream in three different ways. Python and html, RTSP and using Dataplicity and the Hawkeye video streamer.

9.2 Ultrasonic Sensor

My project uses three HC-SR04 ultrasonic sensors. They are placed around the circumference of the chassis and held in place with 3D printed holders. The sensors use sonar to measure the distance between the sensor and the object in front of it. The readings they give is both stable and accurate which is why they make a good choice for obstacle avoidance robots.

To calculate the distance, I used a script from [4] tasmanianfox. This is calculated by measuring the time between the emitted high frequency sound and the time it is received back by being reflected by an obstacle and by knowing the speed at which sound waves travel through the air. This code originally just takes one reading and displays in the terminal. I modified it to continue to take readings and control motors from this. Link to code - [Ultrasonic Code](#)

The sensors are set up so only one sensor is on at a time. It takes a reading if it detects an object / baby within 30cm it will move back away from what was blocking the sensor. The sensor will then go to sleep and move on to the next. While this is not time critical there is a delay of around 100 ms for each sensor to take a reading wait for a response and then act. I began with working with just one sensor and motor and built this up to using three sensors and two motors.



9.3 DC Motors

For my project I am using a 2wd Robotics Chassis that has 2 high speed motors with rubber wheels. which sit inside a high-strength aluminium alloy body. The Raspberry Pi sits on top. Both motors are connected to a H-bridge (H-B) motor driver. The motors are powered by 5 * 1.5v batteries. The H-bridge chip is powered from the 5V pin of the Raspberry Pi. The H-bridge takes the low-level signals from the pi and can turn it in to a high-power signal which can then drive the motor. It can control two DC motors at a time.

The motor is capable of sourcing 71mA while using 6V when not pulling a load. These motors are most efficient when using the 7.5v they need.

The motors are set to work with the ultrasonic sensors. When an obstacle is detected, the motors will turn on. The motors are controlled in different ways based on which sensor has been activated at the time.

These define one of the motors and assign the GPIO pins on the Raspberry Pi to it:

```
#define MotorPin1 0
#define MotorPin2 2
#define MotorEnable 3
```

Once the pins have been defined, they need to be setup as outputs this is done by:

```
int motorSetup() {
    pinMode(MotorPin1, OUTPUT);
    pinMode(MotorPin2, OUTPUT);
    pinMode(MotorEnable, OUTPUT);
    pinMode(Motor2Pin1, OUTPUT);
    pinMode(Motor2Pin2, OUTPUT);
    pinMode(Motor2Enable, OUTPUT);
}
```

The next step involves the ultrasonic sensor taking a reading and using this information to perform a calculation. This is then compared against an IF STATEMENT which if less than 30cm will activate the motors to back BB away from an obstacle by calling the motorReverse() function.

```
int motorReverse() {
    digitalWrite(MotorEnable, HIGH);
    digitalWrite(Motor2Enable, HIGH);
    digitalWrite(MotorPin1, LOW);
    digitalWrite(MotorPin2, HIGH);
    digitalWrite(Motor2Pin1, LOW);
    digitalWrite(Motor2Pin2, HIGH);
    delay(1500);
}
```

Here we see both motors being enabled and set to high. To make sure BB reverses properly I have created two reverse functions depending on which sensor detects an obstacle will affect which way BB reverses. There is also a stop function and two turn functions.

9.4 Android App

9.4.1 Login app

I had to consider security, when designing the android application as there was going to be a video streamed from within people's homes showing their children. It was important that this was addressed so I began by creating a simple login app.

The login app I started with allowed the user to sign-in with a hardcoded username and password. After a five attempts the user would not be allowed any more until they closed the app and reopened it again. This was not very efficient or secure and it did not allow for multiple users to be created and login.

While I was researching, I came across a technology called Firebase which I had been aware of in the context of database through our Mobile App Development module. I found the technology to be exactly what my app needed, and I decided to incorporate this into my Final Year Project App.

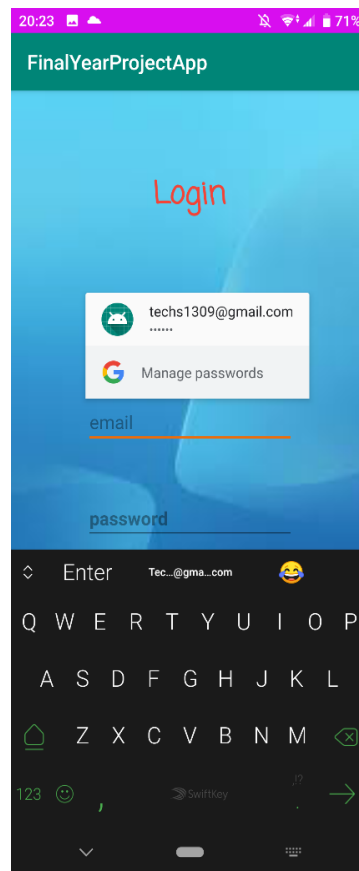
Below are the screenshots for the login and register section with explanations and code snippets.

The creation of this app was part done by a Udemy course and through a YouTube tutorial [2].

When the app is launched it goes straight to the login page. It does this because a regular user will use the login function far more the register functions the user can enter his/her details, email and password, which is then authenticated by the Firebase system. If the credentials match, the app launches the main menu view. Before this though a new user needs to register their details.

```
fAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(new OnCom-
pleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()){
            Toast.makeText(Login.this, "user logged in.",
            Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(Login.this, MainActivity.class);
            startActivity(intent);

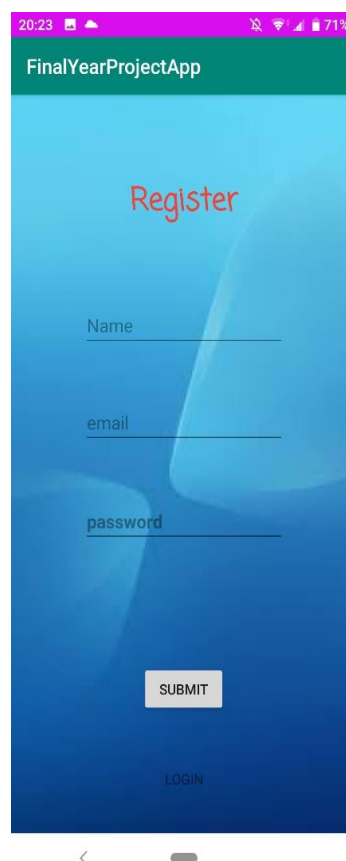
        }else{
            Toast.makeText(Login.this, "Error" + task.getException().getMessage(),
            Toast.LENGTH_SHORT).show();
        }
    }
});
```



Before a user can login, they must first register his/hers details and create an account so they can be stored for Firebase authentication. The user enters name, email, and password. With email and password being passed in to create a user.

```
fAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener(new On-
CompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()){
            Toast.makeText(Register.this, "user created.",
            Toast.LENGTH_SHORT).show();
            startActivity(new Intent(getApplicationContext(),MainActivity.class));

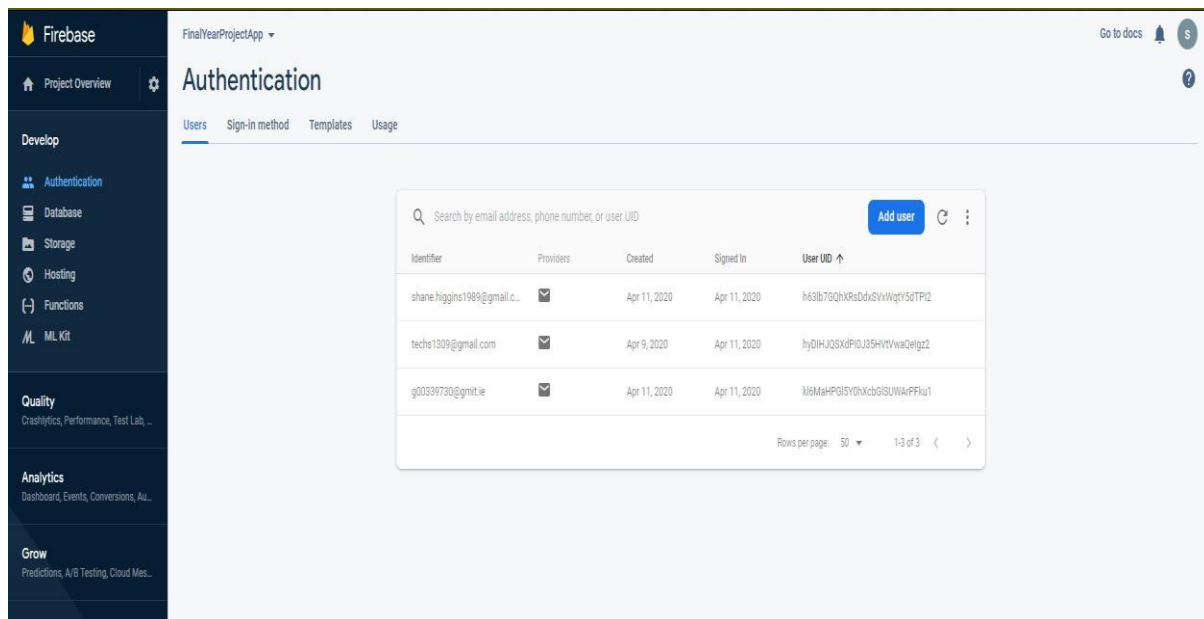
        }else{
            Toast.makeText(Register.this, "Error" + task.getException().getMes-
            sage(), Toast.LENGTH_SHORT).show();
        }
    }
});
```



The screenshot shows a mobile application interface for a registration form. At the top, there is a status bar with the time 20:23, signal strength, Wi-Fi, and battery level at 71%. Below the status bar is a green header bar with the text "FinalYearProjectApp". The main background is a blue gradient. In the center, the word "Register" is written in a red, stylized font. Below it are three input fields labeled "Name", "email", and "password" in a light blue font. Each field has a horizontal line for text entry. Below the input fields is a white button with the text "SUBMIT" in black. At the bottom, there is a link labeled "LOGIN" in a light blue font. The bottom of the screen shows a white navigation bar with a back arrow and a home button.

9.4.2 Firebase

This is the Firebase authentication tab. Which is linked to my Gmail account.



Why did I use firebase? My first login app was very basic. The username and email were hardcoded in which did not give the user the freedom to pick their own password or to sign-in using their own credentials. This was not secure and did not offer the users peace of mind, especially when they were going to stream their children from inside their home. They would want to know who has access and that the people using the app are the people who they have given access to. After researching the topic of authenticating an app the first place I saw it in was in an online course in Android app development where its used with kotlin. I have no experience in kotlin so I looked up how to do it in java and through the use of YouTube videos and online tutorials. Since it is both widely used and well documented I was able to start developing my FinalYearProjectApp that would include user authentication and storage.

9.4.3 Setting up Firebase

Firebase option rests in the drop-down menu within the tools option. Once in the Firebase options it gives you the option to install the Firebase SDK, if you have never done this, you will need to sign into firebase with your Gmail account and allow access to Android Studio. Then back in Android Studio (AS) you must connect your app to Firebase.

The development of the apps and the evolution from a simple login to something that has real world applications was a good learning experience. I learned about the need for security in apps as they are used on a mobile device which is regarded as one of a person's most essential items nowadays. Having an app where the password is hardcoded would not fill the user with confidence. Because of this I think it was the right choice to switch methods and the outcomes are more in line with what I had originally planned.

9.4.4 Main Menu

Main Menu of the app is the first opened after the username and password has been authenticated. Once this happens an intent is launched showing the view below. This main menu has three buttons.

Real time streaming protocol (RTSP) STREAM button will launch a new intent and activity that will display a live feed coming from the Raspberry Pi.

The WEB STREAM button originally started out as a stream from the Raspberry Pi where a Python script was running, displaying the camera feed to a html page. I thought that it would be better to try and find another way to stream the video feed on the web. While I was researching, I came across Dataplicity. This was something I found extremely interesting as it acts like a VPN for a Raspberry Pi that lets me access it from anywhere. Using this I was able to set up a video feed that was viewable from anywhere which got rid of the need for port forwarding. This will be discussed in more detail later.

The LOGOUT button uses the firebase credentials to end the session and return the user to the login intent where the users can once again login.

This shows the three buttons. Each button starts an intent which has a java class associated with it. The logout signs the user out of the app ending the Firebase session and returning the user to the login view.

9.4.4.1 Main Menu Code and screenshot

```
public void StreamOne(View view){

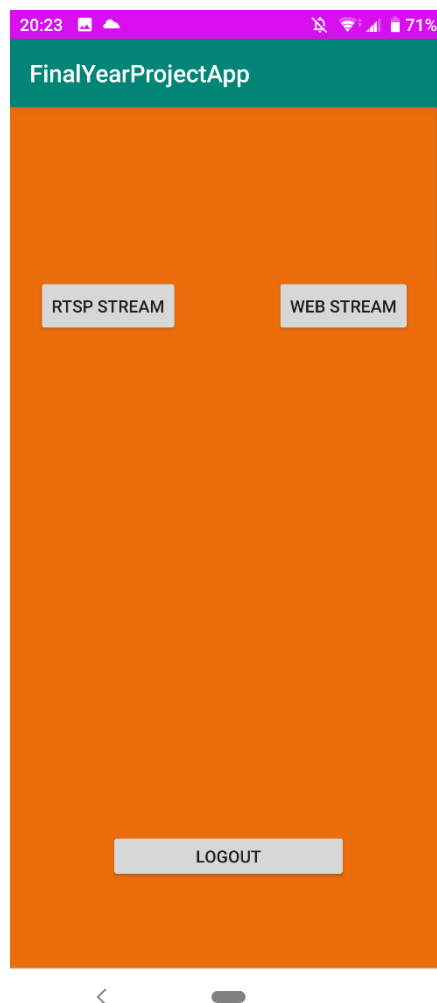
    Intent intent = new Intent(MainActivity.this, RTSPActivity.class);
    startActivity(intent);
    Toast.makeText(this, "RTSP Stream", Toast.LENGTH_LONG).show();
}

public void StreamTwo(View view){

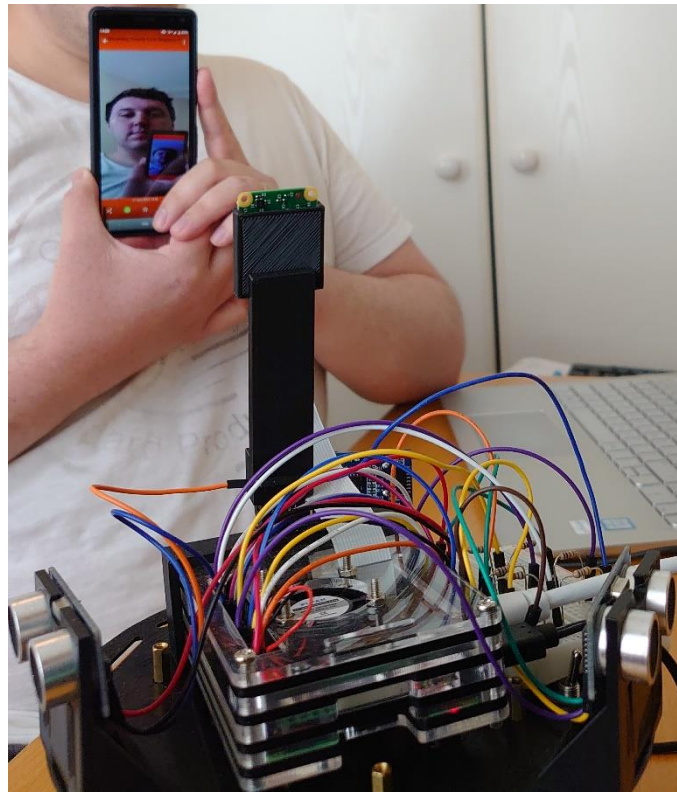
    Intent intent = new Intent(MainActivity.this, HTMLActivity.class);
    startActivity(intent);
    Toast.makeText(this, "HTML Stream", Toast.LENGTH_LONG).show();
}

public void logout(View view) {

    FirebaseAuth.getInstance().signOut();
    startActivity(new Intent(getApplicationContext(), Login.class));
    finish();
}
```



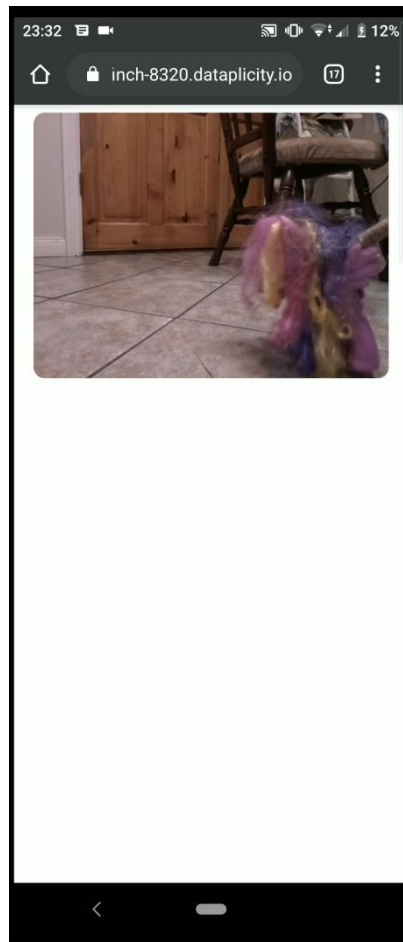
9.4.4.2 RTSP video stream



RTSP streaming app came from an online tutorial. [1]. I modified it slightly as I did not need the username and password of an IP camera. The app was built as a standalone one first and then integrated in to the FinalYearProjectApp using intents.

Issue when working with this especially the lag which pushed me in the direction to try different techniques for a video streaming.

9.4.4.3 Dataplicity video stream using Hawkeye



Web stream Button opens a new view with a button to run a link where a live feed was being displayed on a html page that was running from a Python script. I changed this to instead use a software called Dataplicity.

This is essentially a VPN for a Raspberry Pi. That allows access from anywhere in the world. The way it works is I install a Dataplicity agent on the Pi and create an account that allows me to access the Dataplicity servers which creates a ssh connection.

It routes the traffic through an encrypted WebSocket connection. As it is originating from my device it bypasses NAT, Firewalls and dynamic IP addressing which normally cause issues when using remote access. Once I have connected to the Pi, I can start a stream using the Hawkeye video streamer using this command.

- `sudo /opt/webcam_streamer/start`

It begins the stream. Then I use the wormhole which creates an encrypted connection between the servers and the agent on the Pi. It assigns it to unique URL and starts streaming. I decided to use this in place of the HTML/Python which was streaming in the app on a local network this new method now streams the video to the app no matter what network the user is on. Compared to the way I was doing it I think this is much more usable. I also learned about a new technology and software system.

9.5 Project Integration

My project was built in a modular way. Each phase consisting of the research, development, and testing. After the first phase was complete each of the following phases once tested were tested with what came before. This was to allow for any error or incompatibility between parts to be found in the early stages. It also aided in the final integration as each part had been tested independently and with the others.

To give an example, testing of both the motor and the sensors was done individually and then tested again when integrated together. This allowed for a smooth transition when all parts were combined.

Another example was the build of three separate apps and then building them into each other.

Integrating Dataplicity and Firebase into the app and the Pi was done in the later stages of the project as these were things I came across after my initial research.

10 Problem Solving Methods

My first problem was a what was my project going to be. Would it be software or hardware.

The problem-solving methods that I tried to apply throughout the project were that of a structured problem-solving method. Knowing what my problem was and where I wanted to be at the end of it greatly helped in finding solutions.

An example of such would be my login app. I knew that my simple login was not secure and did not offer the user the ability to use their own credentials and password which was not very appealing.

This was my problem to solve.

I knew that I wanted the app to have the ability to register a new user and let users sign in while storing this information. With this knowledge I found a technology that I was already slightly familiar with called Firebase. Which allowed users to register and login while saving this information in the cloud. I implemented this into my app in place of the login function I already had. This improved the functionality of the app and it increased its complexity.

With this being my default method of problem solving I would need to sometimes use trial and error methods if the problem were one that could be solved on a smaller scale.

I would have liked to develop this further and allow the user to register in the main menu view, but this created lots of errors which I unfortunately did not have the time to solve due to the current circumstances.

I used various resources to help me solve my challenges. These being.

- Google
- Udemy
- YouTube
- Classmates
- Lectures
- Online articles / websites

11 Project issues / complications

Through this project I encountered some small problems with app and hardware integration.

Streaming to the app lagging issue when using RTSP worked but caused severe delays at times. This could become dangerous in real-world situations where children are involved.

Which is what lead me to attempting the three different methods of streaming.

Sourcing the right chassis took the first semester and needed to be ordered from the United States.

Power supply issues for the Raspberry Pi to make it mobile so BB could move on its own. Initially I used a 20mAh power bank, but I was not completely able to make it mobile as it needed to be plugged in. Through my research I did find smaller power banks that could have suited my needs but with the closure of the college I was not able to source them.

Decided to move away from OpenCV in the project while it would have been a nice feature due to the fact that it was not a main aspect of my project I felt that trying to incorporate it in would have delayed other aspects of the project.

One of the biggest issues was the adjustment to working from home. I know a lot of people found this an issue and especially towards the end of the project when things were starting to come together. It took some time to get used to my new work environment and having my workflow completely disrupted. Even harder so trying to adapt when you have kids that are looking for attention.

12 Project Reflection

The outcome of my project was to build a small robot with obstacle avoidance that streamed a video feedback to an android app. While along the way I did not meet all the goals I set, or they changed along the way my finished project has demonstrated many of the outcomes I desired.

By working on this project, I have learned a lot about new real-world technologies that I hope will benefit me in the future.

I feel that throughout the project one of the biggest things I learned was the project planning aspect. It became clear just how important project planning is when working on big projects. I also learned about several real-world technologies like Dataplicity, Googles Firebase and mobile app development. Being able to integrate all the different aspect of the project to produce a working prototype I feel has been a great success.

If I were to do the project again somethings, I would like to do differently would be to plan out the certain hardware aspect a bit more in advanced for example the speaker/buzzer. Which I was unable to incorporate into the project, but I feel would have made nice features. I would also like to explore different types of sensor such as Lidar.

Where could I take this project if I were to develop it further, I would like to make it more interactive to be able to respond to sound as well as movement.

I would also like to explore different types of applications it could be applied to as it is a very versatile technology that I personally feel could benefit parents and children alike.

Overall, I do feel this was a successful project with lots of challenges along the way that have given me a better understand of different technologies and I am happy I have had the chance to follow through with it.

One last thing I would like to mention is that throughout the lockdown it has been very difficult at times to work from a home environment and that without the support of my wife the outcome to my project could have been very different.

13 Reference

- [1] <https://ijoshsmith.com/2014/01/25/video-streaming-from-an-ip-camera-to-an-android-phone/>
- [2] Small Academy, YouTube, <https://www.youtube.com/watch?v=TwHmrZxiPA8&list=PLL9-wdLL6PAdPQCkHFb0mY0K-1ocm1gYW>
- [3] Drogon, Self-employed ICT consultant <http://wiringpi.com/>
- [4] tasmanianfox, github,
<https://gist.github.com/tasmanianfox/a13e8aab2aa5ade1b283d51432039029>