

Project Engineering

Patient Health Monitoring System

Caleb Fox-Carr

G00327455

Bachelor of Software & Electronic Engineering

Galway-Mayo Institute of Technology

2019/2020

A screenshot of a cell phone

Description automatically generated

A close up of text on a white background

Description automatically generated

A screenshot of a cell phone

Description automatically generated

**A close up of text on a white background

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering 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.

Caleb Fox-Carr

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Acknowledgements

I would like to acknowledge my project supervisor Mairtin O’ Conghaile for his continuous help and guidance over the course of the module.

He helped me with all problems and challenges I had with my project.

Table of Contents

[Introduction 10](#_Toc40702013)

[Project Architecture 11](#_Toc40702014)

[Development Platform and Tools 13](#_Toc40702015)

[Raspberry Pi Development board 13](#_Toc40702016)

[SQLite3 14](#_Toc40702017)

[MQTT 15](#_Toc40702018)

[Adafruit DHT Sensor Library 16](#_Toc40702019)

[Flask 17](#_Toc40702020)

[Sensors 18](#_Toc40702021)

[DHT22 18](#_Toc40702022)

[Pi Camera Module V2 19](#_Toc40702023)

[Android Application 21](#_Toc40702024)

[Log in Screen 21](#_Toc40702025)

[Menu Screen 22](#_Toc40702026)

[Historical Data 23](#_Toc40702027)

[Graphed Data 24](#_Toc40702028)

[Patient Live Stream 25](#_Toc40702029)

[Web Page 26](#_Toc40702030)

[Historical Data Page 26](#_Toc40702031)

[Date/Time picker 26](#_Toc40702032)

[Historical Data 27](#_Toc40702033)

[Graphed Data 28](#_Toc40702034)

[Full Historical Data Page 29](#_Toc40702035)

[Live Sensor Data Page 30](#_Toc40702036)

[Live Sensor Vitals 30](#_Toc40702037)

[Patient Record 31](#_Toc40702038)

[Full Live Sensor Data Page 32](#_Toc40702039)

[Problem Solving 33](#_Toc40702040)

[The Critical Path Method 33](#_Toc40702041)

[MQTT Data Transfer 35](#_Toc40702042)

[Conclusion 37](#_Toc40702043)

[Conclusion – Hardware 37](#_Toc40702044)

[Conclusion – Software 37](#_Toc40702045)

[Conclusion – Going Forward 38](#_Toc40702046)

[References 39](#_Toc40702047)

[Figure 4‑1 Architecture Diagram 12](#_Toc40701958)

[Figure 6‑1-1 DHT22 Temperature & Humidity Sensor 18](#_Toc40701959)

[Figure 6‑1-2 DHT22 Wiring Diagram 19](#_Toc40701960)

[Figure 6‑2-1Raspberry Pi Camera Module v2 20](#_Toc40701961)

[Figure 6‑2-2 Raspberry Pi Camera Module v2 Wiring 20](#_Toc40701962)

[Figure 8‑1 Log in Screen 21](#_Toc40701963)

# Introduction

For my BEng Software & Electronic Engineering final year project I built a Smart patient Health Monitoring System which will monitor the vitals of patients and give Doctors, Nurses, home carers and family members the ability to monitor patient remotely. The inspiration for this project comes from past experiences I have had with seriously sick family members who need constant supervision. I also have experience working in the medical device industry with Aerogen where I learned about the importance of regulations and standards.

To do this I designed a python application which will gather sensor data, store it in an SQLite database and send the data to a webpage. When the user accesses the web page, they will be able to see live sensor readings, the historical data of the sensor data logged into the database and a visual representation in the form of a graph. I also took steps towards making this system available globally by deploying the website on Amazon Web service through an EC2 instance.

In addition to the website I also developed an Android application using Android Studio which was programmed in Java. Upon opening the application the user is required to enter their credentials, after successful credential entry the user is brought to the menu screen, here the user will be able to navigate the application and choose from the list of functions that the app provides such as Live Sensor Data, Historical data, Graphed data and a live video stream of the patients quarters.

This instance will be configured as an MQTT broker, the raspberry Pi will measure patient vitals using various sensors and publish those values by assigning them to the topic “test” on the broker which is running on the virtual machine. This allows the webapp and android app to subscribe to the broker for the topic test to receive the sensor data, save it in an SQLite database and display it to the user.

The Target audience for this project is overworked, understaffed doctors and nurses who are already under increased strain and pressure recently in the battle against Covid-19, however this system can also be used to treat sick family members or to monitor the health of children.

The webpage will be aimed at customers who want a detailed, accurate overview of the patient’s health. If the customer wants a quick on the go checkup the would be recommended to use the android application available for download.

The goal of this report is to give the reader an understanding of the hardware, software and other tools that I have utilized in the implementation of this project as well as an understanding on how to operate the system correctly.

# Project Architecture

A close up of text on a white background

Description automatically generated

Figure ‑ Architecture Diagram

A close up of text on a white background

Description automatically generated

Figure 4‑2 MQTT Dataflow

# Development Platform and Tools

## Raspberry Pi Development board

The brains of my system will be the Raspberry Pi 4 development board. It will be responsible for gathering sensor data, creating a webserver to host my website, and sending data to my android application.

A circuit board

Description automatically generated

Figure 5-1 RaspberryPi Hardware Specs

I chose the Raspberry Pi as my development board as I wanted to expand my skillset by including Python in the list of programming languages that I have covered. Here I’m going to go over some of the key talking points in relation to hardware specifications.

* BCM271 with 64bit, quadcore ARM cortex-A72 CPU running at 1.5GHz
* 4 GB of RAM
* Dual band 802.11b WIFI & Bluetooth 5.0 capabilities
* Camera & Display connector
* Gigabit Ethernet port
* 2 x USB 2.0 & 2 x USB 3.0 ports
* USB C power connector
* 2 Micro HDMI connectors that support 4k video
* 40 GPIO pins

## SQLite3

SQLite is a software library that provides a relational database management system. It is considered “lite” because it has easy setup, easy database administration and doesn’t take up much resources. Normally database services such as MySQL or PostgreSQL require a separate server to operate, However SQLite doesn’t work like this.

SQLite does not require a server to run and is directly connected to the application. The following diagram explains the SQLite architecture:

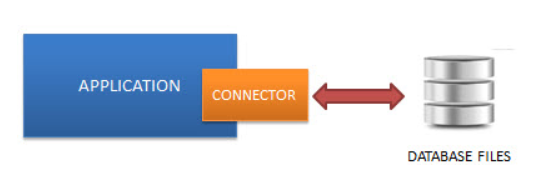


Figure 5-2 SQLite Architecture

The reason I chose SQLite was because of its easy setup with the raspberry pi. Here will show you how I set up my tables which I used to store temperature, humidity, beats per minute and blood pressure:

sqlite> begin;

sqlite> create table temperatures (rDatetime datetime, sensorID text, temp numeric);

sqlite> create table humidities (rDatetime datetime, sensorID text, hum numeric);

sqlite> create table bpm (rDatetime datetime, sensorID text, bpm numeric);

sqlite> create table bPressure (rDatetime datetime, sensorID text, bp numeric);

sqlite> commit;

I then insert sensor data into these sesnores using embedded SQL queries from my data logger script.

curs=conn.cursor()

  curs.execute("""INSERT INTO temperatures values(datetime(CURRENT\_TIMESTAMP, 'localtime'),

         (?), (?))""", (sensor\_id,temp))

  curs.execute("""INSERT INTO humidities values(datetime(CURRENT\_TIMESTAMP, 'localtime'),

         (?), (?))""", (sensor\_id,hum))

  conn.commit()

  conn.close()

## MQTT

MQTT or MQ Telemetry Transport is a machine to machine connectivity protocol. This allows a device to publish a message to that topic, all other devices which are subscribed to that topic will receive that message. In my system these messages contain the sensor data from the various sensors. For my system I’m using the mosquito MQTT broker which is an open source message broker which implements the MQTT message protocol. The broker is the central part of the network and receives all messages. The broker filters through these messages and decides which devices are interested in them, the broker then publishes the messages to all the subscribed topics.

Here I will go through some of the key code that implemented MQTT data transmission between the various devices in my system

# Define Variables  
 MQTT\_HOST = "localhost"  
 MQTT\_PORT = 1883  
 MQTT\_KEEPALIVE\_INTERVAL = 45

This section of code configures the RPI as the broker.

mqttc.publish("th",str(temp)+","+str(hum)+","+str(bpm)+","+str(bp))

The mqttc.publish() function is then used to publish the sensor data for temperature, humidity, beats per min and blood pressure to the topic “th”

Next we will move onto the android application which will be subscribed to the messages that are published by the broker.

client =  
 new MqttAndroidClient(this.getApplicationContext(), "tcp://52.91.209.21:1883",  
 clientId);

Here we set up a new MQTT broker object and give it the ip address of the MQTT broker and specify the port number which is defined in the raspberry pi code above.

client.subscribe("test",1);//subscribing temp for temp  
client.subscribe("bpm",1);//subscribing bpm for bpm  
client.subscribe("bp",1);//subscribing bp for bp

Here we subscribe to each of the topics for temperature, humidity, bpm and bp

A close up of a device

Description automatically generated

## Adafruit DHT Sensor Library

Adafruit provide a sensor library which allowed me to easily communicate with my DHT22 Temperature & Humidity sensor. Upon installing the library all I had to do was import the library at the top of the script.

  import Adafruit\_DHT

Now I can start programming my sensor to take in readings. As you’ll see below I configure my sensor to pin 16 and set up an if else statement that states if any temperature of humidity values are detected then log those values.

  humidity, temperature = Adafruit\_DHT.read\_retry(Adafruit\_DHT.AM2302, 16)

  if humidity is not None and temperature is not None:

   log\_values("1", temperature, humidity)

  else:

   log\_values("1", -999, -999)

A close up of a device

Description automatically generated

## Flask

Flask is a micro framework application written in python. It is a set of tools and libraries that make is easy to build web applications. The reason I chose to use flask is because it is lightweight and is perfectly compatible with the raspberry pi 4. Its main use in my project is sending data to my webpage.

Here I will discuss some of the key points when using flask in your application.

from flask import Flask, request, render\_template   
app = Flask(\_\_name\_\_)

First, we import the flask class, we then use an instance of this class to create the wsgi application. We also import the render template method which will be used to return HTML pages located within the templates folder and it also allows me to send sensor data for temperature humidity, beats per minute and blood pressure to the webpage.

Here I will show how I implemented this

return render\_template( "lab\_env\_db.html",  
 temp = temperature,  
 hum = humidity,  
 beats = bpm,  
 bp = bPressure,  
 from\_date = fromDateValue,  
 to\_date = toDateValue,  
 temp\_items = len(temperature),  
 hum\_items = len(humidity),  
 bpm\_items = len(bpm),  
 bp\_items = len(bPressure))

A drawing of a face

Description automatically generated

# Sensors

Sensors are an integral part in the functionality of my system. They are used to gather all of the data that will be stored in my databases. The main sensors I used in my project were the DHT22 Temperature and Humidity Sensor, MIKROE-3218 heart rate sensor and

## DHT22

The DHT22 Temperature & Humidity sensor consists of a capacitive humidity sensing component, thermistor and an IC on the backside of the sensor. The humidity measuring component has two electrodes with a moisture holding component in between them. As the humidity in the room changes so does the resistance between the two electrodes. The change in resistance is then processed and is ready to be read by our raspberry pi. The thermistor is a variable resistor that changes its resistance as the temperature increases. The thermistor is made from semi conductive materials such as ceramic and polymers and operates on the Negative Temperature Coefficient (NTC) which means as the temperature increases the resistance decreases.

When dealing with people’s health your product needs to be as accurate as possible, I chose the DHT22 over the DHT11 because it was more accurate when taking temperature and humidity readings with 0%-100% humidity readings and accurate to within 0.5C when taking temperature readings.

Here are some important features of the DHT22:

* 3V-5V power and i/o
* 2.5mA max current use
* Very accurate humidity readings between 0-100%
* Very accurate temperature reading between -40c-80c
* Max 0.5Hz Sampling rate
* 4 pins

A close up of a device

Description automatically generated

Figure ‑-1 DHT22 Temperature & Humidity Sensor

A circuit board

Description automatically generated

Figure ‑-2 DHT22 Wiring Diagram

## Pi Camera Module V2

The Raspberry Pi Camera Module V2 is a high quality 8 megapixel Sony IMX219 image sensor which has been customized for the raspberry pi. The Raspberry Pi 4 now comes with a special socket on the face of the board with CSi interface which has been designed with camera interface in mind. Here is a list of the Pi cameras main features :

* Fixed focus lens on-board
* 8-megapixel camera
* Supports 1080p, 720p and 480p video
* Size 25mm X 23mm X 9mm
* Weight: 3g

A close up of a device

Description automatically generated

Figure ‑-1Raspberry Pi Camera Module v2

A circuit board

Description automatically generated

Figure ‑-2 Raspberry Pi Camera Module v2 Wiring

To describe your software, it is not suffient to just paste some code in here, you should describe what your code is designed to do, in English. Write out any mathematical equations or calculations and explain them. If you decided to put your code in functions or libraries or objects, describe this. A good layout is to include a snippets of code alongside its explanation. You do not have to explain every part of your code, pick the important parts.

# Android Application

## Log in Screen

Upon opening the app, the user will be required to identify and authenticate themselves in order to enter the application. This prevents trespassers from accessing valuable medical data contained in the application.

A screenshot of a cell phone

Description automatically generated

Figure ‑ Log in Screen

If the user enters the wrong password 3 times, he/she is locked out of the application and will not be able to access it until the cool down period is finished. This is achieved by setting count to 3 and whenever a wrong password is entered the count decrements by 1, eventually deactivating the ability to log in.

count --;  
if(count == 0 )  
{  
 log.setEnabled(false);  
}

By pressing the cancel button, the application closes.

cancel.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 finish();  
 }  
});

The correct log in credentials are hard coded into the application, when they are met the user will be brought into the menu screen. To implement this, I used OnClickListener functions and intents. When the login button is pressed a new intent is created, the app checks the credentials and upon approval starts the new intent.

## Menu Screen

Once the user credentials are verified the user is brought to the Menu Screen. Here the user is able to navigate the various features that my system provides including Live sensor data, historical data, graph data and a live stream of the patient.

A screenshot of a cell phone

Description automatically generated

Figure ‑2 Menu Screen

In the menu screen I use intents to start the activities that each of the buttons are associated with.

history.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Intent startIntent = new Intent(getApplicationContext(),History.class);  
 startActivity(startIntent);  
 }  
});

getApplicationContext() returns the context for the entire application while History.class references which activity will start when the button is pressed. The same process is applied for all of the other buttons.

## Historical Data

The historical data screen allows the user to view all the data collected by the sensors in table format with time stamps. Gathering historical data is useful as it ensures that vital information is retained so it can be revisited later. Nurses or doctors will be able to review the historical data at the beginning and end of every shift.

A screenshot of a computer

Description automatically generated

Figure ‑3 Historical Data Screen

To implement this, I set the number of columns in the TableView using :

TableView<String[]> tableView = (TableView<String[]>) findViewById(R.id.table);

TableColumnWeightModel columnModel = new TableColumnWeightModel(4);

To display the data we then convert the values within our database into TextViews using the line :

tableView.setDataAdapter(new SimpleTableDataAdapter(this, abc));

Where “abc” is the variable assigned to the database

## Graphed Data

Graphing data has many benefits, it allows the user to quickly draw conclusions and assess the patient’s health over a period quickly. The graph screen lets the user compare multiple data sets.

To graph my historical data, I am using GraphView which is a library for android studio which allows you to create flexible, easy to understand diagrams.

A screenshot of a computer

Description automatically generated

Figure ‑4 Graphed Data Screen

## Patient Live Stream

When the user selects the video function, they are brought to a screen which displays the patient’s quarters in real time. This allows the user to keep tabs on the patient and is useful if the user needs to have a quick update on the patient’s situation.

This function was achieved using a software called Motion. This allowed me stream the patients quarters as a background task broadcast a live stream of the patient at all times as long as the Raspberry Pi was turned on.

We then use WebView in our android application to open this stream within our application by using the loadUrl() method and going to the ip address of the raspberry pi on port 8081 where the live stream is being broadcast locally

myWebView.loadUrl("http://192.168.0.59:8081");

A picture containing indoor, monitor, sitting, computer

Description automatically generated

# Web Page

## Historical Data Page

The home page of the website is the historical data page. This page acts as an insight into the patient’s historical vital information. Historical data is essential in monitoring a patient’s wellbeing as doctors and nurses can track the progress of a patient over time.

### Date/Time picker

When designing my webpage, I wanted to give the user the ability to select from any range of date of time that they wanted. This would enable the user to be more throughout and accurate in the monitoring of the patient’s vitals over time.

A screenshot of a computer

Description automatically generated

This date/time picker is implemented in the HTML code through a script using jQuery.

jQuery is a JavaScript library which allows traversal, manipulation and event handling within HTML code.

  <script>

    jQuery('#datetimepicker1').datetimepicker(

      {

        format:'Y-m-d H:i',

        defaultDate:'{{from\_date}}'

      });

    jQuery('#datetimepicker2').datetimepicker({

        format:'Y-m-d H:i',

        defaultDate:'{{to\_date}}'

      });

      jQuery("#range\_select input[type=radio]").click(function(){

        jQuery("#range\_select").submit();

      });

  </script>

Here I configure the JQuery plugin so that the default date range is specified using “{{…}}” which call the variable in the python application code that gets the from/to date values from the Url.

### Historical Data

This system allows the user to navigate the historical data gathered by the sensors which is stored in the SQLite database.

A picture containing person, holding

Description automatically generated

I use embedded SQL queries to select the range of values between fromDateValue and toDateValue and use the function curs.fetchall() to retrieve those values and return them so that the bpm and bPressure can be sent to the webpage.

curs.execute("SELECT \* FROM bpm WHERE rDateTime BETWEEN ? AND ?", (fromDateValue, toDateValue))

    bpm             = curs.fetchall()

    curs.execute("SELECT \* FROM bpm WHERE rDateTime BETWEEN ? AND ?", (fromDateValue, toDateValue))

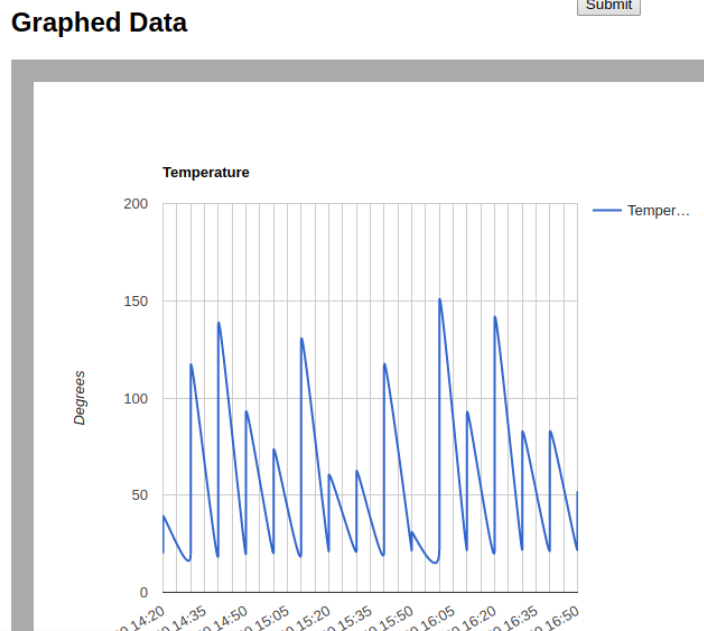
    bPressure       = curs.fetchall()

    conn.close()

    return [temperature,bpm,bPressure, humidity,fromDateValue, toDateValue]

### Graphed Data

To visually represent the historical data in graph form I used google charts API to give my website this capability.



To implement the graph we see above we must first specify the location that is going to hold the graph. This is done in the historical data HTML code:

    <h2>Graphed Data</h2>

            <div class="fakeimg">

              <div id="chart\_temps"></div>

              <div id="chart\_humid"></div>

              <div id="chart\_bpm"></div>

              <div id="chart\_bp"></div>

            </div>

As you can see all of my charts are stored within their own individual <div>

var data = new google.visualization.DataTable(); //Create new data object that will contain data

      data.addColumn('datetime', 'Time');

      data.addColumn('number', 'Temperature');

      data.addRows([

          {% for row in temp %}

          [new Date({{row[0][0:4]}},{{row[0][5:7]}},{{row[0][8:10]}},{{row[0][11:13]}},{{row[0][14:16]}}),

                {{'%0.2f'|format(row[2])}}],

          {% endfor %}

        ]);

We start by creating a new data object that will contain all of the data we want to visualize using. The values for the X and Y axis will be contained in the “time” and “temperature” column using data.addColumn. We use data.addRows to begin constructing the tables, here we extract the first 4 digits of the year by replacing {{row[0][0:4]}} with the corresponding values from “temp”. So in this case “{{row[0][0:4]}}” will be replaced with “2020” and the process is completed until the full date and time is complete. The same process is applied to get the temperature value.

### Full Historical Data Page

A screenshot of a social media post

Description automatically generated

## Live Sensor Data Page

In the navigation bar the user will have the option to click “Live sensor data” this will bring the user to a screen which will display the patients “Profile” and also the patients “Live Sensor vitals” which are taken directly from the systems sensors. This page will automatically update every 10 seconds to give the user fresh, up to date information.

### Live Sensor Vitals

Sensor data is taken within the python application and sent to the sensor.html using the render\_template() function as follows as follows:

   humidity, temperature = Adafruit\_DHT.read\_retry(Adafruit\_DHT.AM2302, 16)

    if humidity is not None and temperature is not None and BeatsPerMin is not None and Bpressure is not None:

        return render\_template("sensor.html",temp=temperature,hum=humidity,

        bpm=BeatsPerMin,bp=Bpressure)

The values temp, hum, bpm and bp are then called from within the html page:

   <h2>Live Patient Vitals</h2>

    <h5>Temperature: {{"{0:0.1f}".format(temp) }}°C</h5>

    <h5>Humidity: {{"{0:0.1f}".format(hum)}}%</h5>

    <h5>Beats per Minute: {{"{0:0.1f}".format(bpm) }}°C</h5>

    <h5>Blood Pressure: {{"{0:0.1f}".format(bp)}}%</h5>

A screenshot of a cell phone

Description automatically generated

### Patient Record

In the Live sensor Data page, there is also a section which specifies the patients sex, age, condition and contact details. This is implemented purely in HTML currently but further down the line I would like to have multiple patients who all have their individual profiles stored within database.

Here I will demonstrate how I implemented the patient record section

<div class="row">  
 <div class="side">  
 <h2>Patient Record</h2>  
 <h5>Caleb Fox-Carr:</h5>  
 <div class="fakeimg" style="height:200px;"><img src="/home/pi/Desktop/FullStack2/templates/john.jpg"></div>  
 <p>Sex: Male</p>  
 <p>Age: 23</p>  
 <p>Condition: Covid-19</p>  
 <p>Phone: 089234156</p>  
 <p>Email: g00327455@gmit.ie</p>

A screenshot of a cell phone

Description automatically generated

### Full Live Sensor Data Page

A screenshot of a cell phone

Description automatically generated

# Problem Solving

Over the course of the last year I have learned the importance problem solving. I have utilized many already existing problem-solving tools to work through problems while also developing my own variation of problem-solving methods that allow me to speed up the process of troubleshooting so that I don’t get bogged down on 1 problem for too long. In this section I am going to go through some of the problem-solving techniques that I utilized while on work placement with Thermo King while troubleshooting and working through problems.

## The Critical Path Method

When taking on this project initially I found it hard to focus in on what aspects of my project needed most attention. One of the skills I utilized while on Internship with Thermo King was how to decide upon the critical path of a project. We used this problem-solving tool during rapid improvement events which mapped out the essential parts to the project that needed to be completed before the team moved onto the next task.

A close up of a map

Description automatically generated

Figure 7‑1 Critical Path Method

Here is a visual representation of the critical path of my project. The Critical path is represented by the list of tasks which follow the red arrows. As you can see the critical path is made up of the main parts of my project such as the Webpage and Android Application and all of the project engineering deliverables such as the Proposal, Video and Report.

So, we have talked about what makes up the critical path but there are more aspects to the project that must be included and completed before we have a fully functional system such as all the subtask that make up the building of the Website and the android application. All of these tasked needed to be completed before the video was created so that I was able to demonstrate my working application.

A close up of a red background

Description automatically generated

Figure 7‑1 Critical Path for patient health monitor

## MQTT Data Transfer

One of the main problems I faced during this project was the communication of sensor data from the Raspberry Pi to the virtual EC2 instance where my website is being deployed to AWS. The problem was that the virtual instance had no capability to connect sensors to it, so the data had to be transferred via MQTT. Originally my plan was to use a cloud MQTT service which would have made things a lot easier as there was a lot of support for this type of data transfer, however the cloud MQTT service was not available currently so I needed to find a solution that did not involve cloud MQTT.

My solution was to use Mosquito MQTT similarly to what I was already using for sending data with the raspberry pi. When installed the system now has broker capabilities.

Here is how I Implemented this:

broker\_address="52.91.209.21"  
print("creating new instance")  
client = mqtt.Client("VM") #create new instance  
client.on\_message=on\_message #attach function to callback  
print("connecting to broker")  
client.connect(broker\_address,1883) #connect to broker  
client.subscribe("test")

Here as you can see, I specify the broker address as the IP of the virtual instance, we then connect to the broker using the variable assigned to the instance and specify the port number. We the subscribe to the topic “test” which is being published from the RPI. So when a message is received the on\_message function is called which decodes the data, splits it up into 4 data values and logs that data into the database.

Here I will demonstrate this:

def on\_message(client, userdata, message):  
 #Called when message is recieved on subscribed topic test  
 print("message received " ,str(message.payload.decode("utf-8")))  
 data = str(message.payload.decode("utf-8")).split(",")  
 print(data)  
 temperature = data[0]  
 humidity = data[1]  
 bpm = data[2]  
 bPressure = data[3]  
 print(data[0],data[1],data[2],data[3])  
 log\_values("1", temperature, humidity,bpm,bPressure)

As you can see all incoming data is assigned to a variable of the respective measurement as the log\_values function is used to log that data into the data base.

A close up of text on a white background

Description automatically generated

Figure 7‑1 MQTT flow of data

# Conclusion

In conclusion I have confirmed that it is possible to create a fully operational patient health monitoring system using just a Raspberry pi and some sensors however there are a few key areas that I will be looking to improve on in the future. In this section I am going to give a run down on certain topics and talk about what aspects to that topic were a success and where I see room for improvement.

## Conclusion – Hardware

I was surprised with how powerful the raspberry pi was compared to the Arduino mega that I used for my ordinary level project. The raspberry pi provided me the ability to

The DHT22 is a brilliant little sensor if the user is recording the temperature and humidity of the room however for the purpose of this system, I would insist on adding an additional sensor that is capable of giving more accurate readings of the patient’s body temperature. I would still however keep the DHT22 sensor in the system, but it would be demoted to only recording data on the temperature and humidity of the room.

For recording blood pressure, I currently am generating random values which are hardcoded in the data generating python script as sourcing a blood pressure module for the raspberry pi was difficult. I did not see this as a major issue as the main purpose of my project was the gathering of data and what could be done with the data that was gathered.

Going forward I would be looking to upgrade the hardware components to improve the accuracy and reliability of the system.

## Conclusion – Software

For this project I chose Python in developing my web application because python is the number one most used programming language in the world currently however the Software & Electronic Engineering course in GMIT doesn’t provide any modules in python programming so I saw this as a great opportunity to upskill and stand out from the crowd when it came to applying for graduate positions as Python is being used more and more in the manufacturing industry. The raspberry pi gave me the perfect platform to begin my journey learning python as there is so much support online as fellow python developers are always sharing ideas on how to troubleshoot issues that arose for me during my project.

On the app development side I made the decision to write my app in java as at the beginning of the year I felt that Java was my weakest programming language that we had covered and I saw this as an opportunity to improve on some of my weaker attributes as well as familiarize myself in the growing industry of app development. I found programming applications in Java much easier to understand than what I have previously covered as there is a visual representation of the code when you run it on the emulator.

I would have questioned my programming skills at the beginning of this semester before I began work on my final year project but now upon completion I can look back and see how much improvement I have made and this has translated to other subjects such as Software Engineering with Test and Mobile App Development.

## Conclusion – Going Forward

The reason for choosing to build a smart patient health monitoring system was my ambition for securing a graduate role and one of the top medical device companies in Ireland. I tried to align the skills that I displayed in this project with the skills that top companies look for when recruiting talent. As I move forward with my journey as a Software & Electronic Engineer, I will constantly be looking to upskill which will allow me to progress with improving this system whether that be hardware or software upgrades.

# References

[1]Pi camera streaming:

* tutorialspoint, 2020. Android - Webview. Available at: <https://www.tutorialspoint.com/android/android\_webview\_layout.htm> [Accessed 17 May 2020].
* Parreno, M., 2017. Spy Your Pet With A Raspberry Pi Camera Server. Available at: <https://hackernoon.com/spy-your-pet-with-a-raspberry-pi-camera-server-e71bb74f79ea> [Accessed 17 May 2020].

[2] Data Storage SQLite:

* developer.android.com, 2019. Save Data Using Sqlite. Available at: <https://developer.android.com/training/data-storage/sqlite> [Accessed 17 May 2020].
* javatpoint.com, 2020. Android Sqlite Tutorial. [image] Available at: <https://www.javatpoint.com/android-sqlite-tutorial> [Accessed 17 May 2020].
* Android, 2020. Android-Chapter13-SQL-Databases. [image] Available at: <https://learnonline.gmit.ie/pluginfile.php/150361/mod\_resource/content/2/Android-Chapter13-SQL-Databases.pdf> [Accessed 17 May 2020].

[3] MQTT:

* appcodelabs, 2019. Introduction To Iot: Build An MQTT Server Using Raspberry Pi. [image] Available at: <https://appcodelabs.com/introduction-to-iot-build-an-mqtt-server-using-raspberry-pi> [Accessed 17 May 2020].
* 2017. How To Subscribe MQTT And Receive Message In Android. [video] Available at: <https://www.youtube.com/watch?v=QK2yfwcIV-s> [Accessed 17 May 2020].
* Bremen, L., 2017. API Builder And MQTT For Iot – Part 1. [image] Available at: <https://devblog.axway.com/apis/api-builder-and-mqtt-for-iot-part-1/> [Accessed 17 May 2020].
* Pypi.org., 2017. Paho MQTT 1.5.0. Available at: < <https://pypi.org/project/paho-mqtt/> >[Accessed 17 May 2020].

[4] Android Application

* jjoe64, 2020. Android Graph Library For Creating Zoomable And Scrollable Line And Bar Graphs.. [image] Available at: <https://github.com/jjoe64/graphview> [Accessed 17 May 2020].
* tutorialspoint, 2020. Android - Webview. Available at: <https://www.tutorialspoint.com/android/android\_webview\_layout.htm> [Accessed 17 May 2020].
* Schwarz, I., 2018. An Android Library Containing A Simple Tableview And An Advanced Sortabletableview Providing A Lot Of Customisation Possibilities To Fit All Needs.. [image] Available at: <https://github.com/ISchwarz23/SortableTableView> [Accessed 17 May 2020].
* 2017. How To Subscribe MQTT And Receive Message In Android. [video] Available at: <https://www.youtube.com/watch?v=QK2yfwcIV-s> [Accessed 17 May 2020].

[5] Website

Completed Online Course in Full stack Web Development with Raspberry Pi:

* Udemy, 2018. Raspberry Pi Full Stack Raspbian-A Whirlwind Tour Of Full-Stack Web Application Development On The Raspberry Pi. [image] Available at: <https://www.udemy.com/course/raspberry-pi-full-stack-raspbian/> [Accessed 17 May 2020].
* w3schools, 2018. How TO - Make A Website. [image] Available at: <https://www.w3schools.com/howto/howto\_make\_a\_website.asp> [Accessed 17 May 2020].
* Amazon Web Services, 2020. Getting Started With Amazon EC2 Linux Instances. [image] Available at: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2\_GetStarted.html> [Accessed 17 May 2020].

[6] Images

* SeedStudio, 2020. Hardware Overview. [image] Available at: <<https://www.seeedstudio.com/Raspberry-Pi-4-Computer-Model-B-2GB-p-4079.html>> [Accessed 17 May 2020].
* Hackster.io, 2020. DHT22 Wiring Diagram. [image] Available at: <<https://www.hackster.io/adamgarbo/raspberry-pi-2-iot-thingspeak-dht22-sensor-b208f4>> [Accessed 17 May 2020].
* Wikipedia, 2020. Critical Path Method. [image] Available at: <<https://en.wikipedia.org/wiki/Critical_path_drag>> [Accessed 17 May 2020].