



**A STUDY OF THE EFFECTIVENESS OF IOT TECHNOLOGY AND ARTIFICIAL INTELLIGENCE TO BUILD AN EXPERT SYSTEM TO OBSERVE AND DETECT ABNORMALITIES IN HUMAN VITAL SIGNS.**

**JAMES MAHONEY**

SUBMITTED IN PARTIAL FULFILMENT OF THE AWARD OF  
B.SC. (HONS) COMPUTING WITH SOFTWARE DEVELOPMENT  
MAY 2021

## **Abstract**

Artificial Neural Networks are computational models inspired by and meant to mimic, the functionality of the human brain. A Convolutional Neural Network (CNN) is a powerful form of Artificial Neural Network that has remarkable performance when applied to machine learning problems. The CNN is most suited towards image processing problems that require pattern recognition. Some well-known use cases for a CNN are image classification for search engines or social media, applications that incorporate facial recognition and the digitisation of bank cards used by many applications. There is an emerging use for medical image analysis to assist doctors in detecting abnormal results through pattern recognition in image analysis. Artificial Intelligence techniques such as CNN can be incorporated with Internet of Things technology. When this is done, the result is the creation of machines that simulate intelligent behaviour.

This study aimed to investigate the effectiveness of monitoring human vital statistics using Internet of Things technology supported by Artificial Intelligence techniques. The study investigates the different types of sensor technology that could be used to extract data from a human body, and different Artificial Intelligence techniques that could be used to analyse the extracted data. Building an expert system that could mimic a health professional was also investigated during the study. A Prototype was developed using a Raspberry Pi single-board computer with an optical sensor attached. The prototype collected the user's data and processes it through algorithms that execute locally. The prototype then records a short audio clip of the person's chest, converts it to a spectrograph, and runs through a CNN. The prototype then sends the output from the local computations to a Web Application that runs the data through client-side inference methods that then display the output to the person using the prototype.

It was concluded from the performance of the prototype developed that these research questions are shown to be proven, and that the technology is an effective solution for observing and detecting anomalies in human vital signs.

## Table of Contents

Acronyms and Terms Used .....	6
Chapter 1: Introduction .....	7
Chapter 2: Internet of Things technologies .....	8
2.1 Introduction.....	8
2.2 Sensor technology.....	9
2.2.1 Types of Body Sensors .....	9
2.3 Max30105 High-Sensitivity Optical Sensor .....	11
Chapter 3: Wearable Fitness and Health Technology .....	12
3.1 Introduction.....	12
3.2 Smart Watches .....	12
3.3 Smart Ring .....	13
3.4 Photoplethysmography.....	14
Chapter 4: Artificial Intelligence .....	15
4.1 Introduction.....	15
4.2 Support Vector Machine .....	15
4.3 Neural Networks .....	16
4.3.1 Feed Forward Neural Network .....	17
4.3.2 Radial basis function Neural Network .....	18
4.3.3 Convolutional Neural Network.....	19
4.4 Expert Systems .....	20
4.4.1 Introduction .....	20
4.4.2 Inference engine.....	20
4.4.3 Expert systems with neural network integration .....	20
Chapter 5: Cloud Services .....	21
5.1 Introduction.....	21
5.2 Amazon Web Services .....	21
5.2.1 AWS Amplify .....	21
5.3 Google Cloud .....	21
5.3.1 Google Firebase .....	21
Chapter 6: Methodology & Design .....	23
6.1 Research Findings.....	23
6.2 Research Question .....	23
6.3 Proposed Methodology.....	23
6.4 Design.....	25

6.4.1 Architecture Diagram .....	25
6.4.2 Functional Specifications .....	26
6.5 Prototype.....	27
6.5.1 Set up Hardware for the system.....	27
Chapter 7: Implementation .....	28
7.1 Sprints.....	28
7.1.1 Sprint 1: Train CNN model using respiratory data. ....	28
7.1.2 Sprint 2: Create Specific Disease identifier CNN. ....	31
7.1.3 Sprint 3: Test CNN on Raspberry Pi .....	33
7.1.4 Sprint 4: Set up Raspberry Pi and MAX30105 Sensor to take readings. ....	35
7.1.5 Sprint 5: Analyse the data and create a cloud Web Application to view data.....	39
Chapter 8: Findings & Conclusions .....	47
8.1 Results .....	47
8.2 Conclusion .....	48
References .....	49

## Table of Figures

Figure 1 Max30105 system diagram.....	11
Figure 2 Fitbit processed sleep analysis.....	13
Figure 3 Possible and optimal hyperplane example (Ghandi,2018).....	15
Figure 4 A perceptron (Shiffman, 2012) .....	16
Figure 5 Perceptron with weights and bias added (Shiffman, 2012) .....	16
Figure 6 Topology of a neural network.....	17
Figure 7 Retina compared to Feed Forward Neural Network (Kriesel, 2007) .....	18
Figure 8 System architecture diagram.....	25
Figure 9 TensorFlow version on device .....	27
Figure 10 Labelled spectrographs from audio clips.....	29
Figure 11 Accuracy score for CNN model .....	30
Figure 12 Data in respiratory illness dataset. ....	31
Figure 13 Audio Features (Denton, 2020).....	31
Figure 14 Accuracy score and confusion matrix .....	32
Figure 15 Saving model to Google Drive.....	32
Figure 16 Loading Keras CNN model.....	33
Figure 17 Script output on laptop IDE .....	33
Figure 18 Output on Raspberry Pi® .....	34
Figure 19 MAX30105® sensor with pins soldered on .....	35
Figure 20 Circuit diagram.....	36
Figure 21 Real life circuit.....	36
Figure 22 Data collection script .....	37
Figure 23 Data preparation script.....	37
Figure 24 Final sensor data .....	38
Figure 25 Heart rate data.....	38
Figure 26 PPG data.....	39
Figure 27 HRV script.....	40
Figure 28 Fitbit® HRV data .....	40
Figure 29 Application pipeline .....	41
Figure 30 Folder structure .....	42
Figure 31 Subscription code.....	42
Figure 32 Raspberry Pi® script to collect data. ....	44
Figure 33 Updated Web Application.....	45
Figure 34 Edit account details component. ....	45
Figure 35 Client-side HRV ruleset .....	46
Figure 36 User session results.....	46

## **Acronyms and Terms Used**

AI:	Artificial Intelligence
ANN:	Artificial Neural Network
API:	Application Programming Interface
BMI:	Body Mass Index
CLI:	Command-line Interface
CNN:	Convolutional Neural Network
ECG:	Echocardiograph
GPIO:	General Purpose Input Output
GPS:	Global Positional System
HRV:	Heart Rate Variability
I2C:	Inter-Integrated Circuit
IoT:	Internet of Things
LED:	Light Emitting Diode
NN:	Neural Network
O2:	Oxygen
PCB:	Printed Circuit Board
PPG:	Photoplethysmography
REST:	Representational State Transfer
SVM:	Support Vector Machine

## Chapter 1: Introduction

There has been incredible growth in recent years in the use of and monitoring abilities of wearable health and fitness technology. Products such as the Oura<sup>®</sup> ring and the Fitbit<sup>®</sup> smartwatch have allowed users to track their health statistics and monitor their health. Although the technology is mass-produced it is still relatively expensive. Allied to the standardised monitoring which these devices can provide, in the current Covid-19 pandemic there is an additional need to monitor lung function.

Artificial Neural Networks are components of Artificial Intelligence that are designed to imitate the functioning of a human brain. A Convolutional Neural Network (CNN) is a powerful form of Artificial Neural Network that has remarkable performance when applied to machine learning problems. The CNN is most suited towards image processing problems that require pattern recognition. Some well-known use cases for a CNN are image classification for search engines or social media, applications that incorporate facial recognition, the digitisation of bank cards used by many applications. There is an emerging use for medical image analysis to assist doctors in detecting abnormal results through pattern recognition in image analysis. Artificial Intelligence techniques such as CNN can be incorporated with Internet of Things technology. When this is done the resulting is the creation of machines that simulate intelligent behaviour.

The remit of this thesis is to investigate an effective way of developing a system that can monitor a human's vital statistics with the addition of monitoring lung health using Artificial Intelligence and Internet of Things technology.

## Chapter 2: Internet of Things technologies

### 2.1 Introduction

There are several definitions of the Internet of things (IoT), one of the more salient of these was made by the IBM Corporation, when they describe the Internet of Things not in physical terms but more in a conceptual idea of connecting multiple devices, each of which has a basic property of being in one of two binary states, primarily to the Internet, but also to every other device on the network (Clark, 2016). These devices, or “*things*”, have inbuilt sensors and can communicate and transfer data across the Internet. These sensors can be embedded in devices or objects and are linked together and accessible through the Internet.

The first anecdotal, though an oft-repeated example of the Internet of things in practice occurred in the 1980s when a “Coca-Cola®” machine at the Carnegie Mellon University, in America (Ornes, 2016) was made accessible over the Internet. Students who studied computer science at the University could access it and check if there was a drink available and whether the machine was cold before they would travel to the basement of the building to visit the machine to buy a drink. In recent years the Internet of Things is said to consist of twenty-six billion units connected to the Internet (Saha, Mandal and Sinha, 2017). In simple terms, the Internet of things consists of anything with an on/off switch that also has a connection to the Internet.



## **2.2 Sensor technology**

The Merriam Webster dictionary describes a sensor as,

*“...a device that responds to a physical stimulus (such as heat, light, sound, pressure, magnetism, or a particular motion) and transmits a resulting impulse (as for measurement or operating a control)” (Webster, 2020).*

There are a variety of different sensors available that can be used to monitor a range of human vital signs. Body sensors are attached to a human body and rely on reading various aspects of a human's body, such as blood pressure, vibration, heat, texture, or types of deformations by compressive forces (Segil, 2019).

### **2.2.1 Types of Body Sensors**

#### *2.2.1.1 Echocardiography (ECG) Sensors*

ECG sensors are used to measure echocardiograph signals. These sensors are the main types of sensors used in detecting different types of heart diseases. Contractions of the heart change the current intensity of the skin and these changes are sensed by ECG sensors. The sensors are in the form of electrodes that can be attached to a patient's skin. There are different types of electrodes although dry electrodes are the most used types used for ECG machines. Dry electrodes consist of a metal strip (usually stainless steel) that acts as a conductor between a person's skin and the electrode (Lai et al., 2013).

#### *2.2.1.2 Respiration Sensors*

Respiration sensors are a combination of several different sensors combined to detect breathing motion. These sensors are made up of a pressure sensor and an accelerometer. These are fixed around a person's chest to detect expansions and contraction of the chest during the respiratory cycle (Segil, 2019).

#### *2.2.1.3 Temperature Sensors*

Temperature sensors are sensors that detect changes in the physical changes of certain material as it reacts to changes in body temperature. Some sensors use two different metal

materials and measure the current resistance between the materials. This resistance can be used to calculate the current temperature of the sensor (Soloman, 2009).

#### *2.2.1.4 Accelerometer*

Accelerometers are used to measure the acceleration of components in a closed three-dimensional system (Atallah et al., 2010). These sensors can be placed on a person's body and can collect data about the person's movements and energy expenditure. These sensors can collect information and determine the frequency and intensity of energy expenditure by running the data collected through an algorithm.

#### *2.2.1.5 Optical Sensors for Pulse and Oxygen saturation levels.*

Optical sensors use red and infrared light passing through the person's skin. Oxygen saturation and pulse rate can be calculated by the absorption ratio of this light by the skin. These sensors are non-invasive and are attached to the skin. The absorption of light at the wavelengths of the red and infrared Light Emitting Diodes (LED) is very different between oxygenated and blood with less oxygen. Oxygenated blood will absorb more infrared light and will not absorb as much red light and vice versa. With this fact, the amount of oxygen in tissue and pulse rate can be calculated (Jubran, 2015).

### 2.3 Max30105 High-Sensitivity Optical Sensor

The Max30105 sensor has LEDs and photodetectors that can be used to detect light and light absorption from the built-in LEDs. It also contains ambient light rejection built-in. The sensor can be used for particle detection and could be used as a smoke and particle detector. It can be controlled through an inter-integrated circuit(I2C) interface (MAXIM, 2016). These interfaces are available on microcontrollers and single-board computers such as the Arduino® and Raspberry Pi® meaning it is ideal for use in any projects that require optical sensors.

#### System Diagram

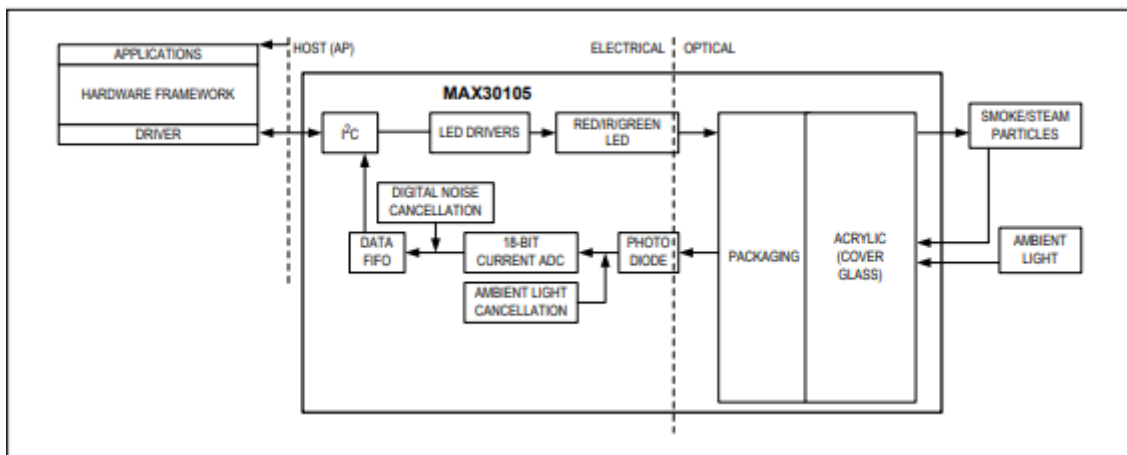


Figure 1 Max30105 system diagram

## Chapter 3: Wearable Fitness and Health Technology

### 3.1 Introduction

As technology has advanced and electronics have become smaller and more powerful, wearable technology has become a new norm. These wearable technologies range from smartwatches to smart rings and chest straps. According to Mordor Intelligence (Smartwatch Market | 2020-2027 | Industry Report, 2020), in 2020 the smartwatch market was at 68.8 million units and it is expected to have a compound annual growth rate of 14.5% from 2021 to 2026. The surge in new users must be in part the want for advanced new products and the usefulness of the health and fitness monitoring they supply the owner.

### 3.2 Smart Watches

Smartwatches are wrist-worn devices containing a PCB board with a display screen and various sensors. Most smartwatches now will contain an accelerometer, GPS, Bluetooth chipset, gyroscope, and pulse oximeter at least and these are all controlled by a micro control unit. The watch allows the owner to monitor their heart rate and step count. This technology uses Photoplethysmography to detect changes in the volumetric variations of circulating blood which will in turn allow the watch to track pulse data. The watch will usually connect to an application that sends the owner's collected data to Fitbit® cloud architecture to be processed. Once processed the owner's application will display data to be viewed as seen in Figure 2 (*Fitbit dashboard, 2021*).



Time Asleep

Edit Goal

6 hr 59 min

Sleep Stages

Learn more

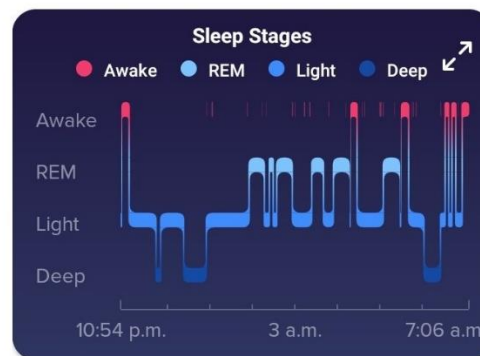


Figure 2 Fitbit processed sleep analysis.

The prices of these watches can vary from €150 to €400 depending on the model.

### 3.3 Smart Ring

The Oura® smart ring is a wearable device containing infrared LED sensors, a temperature sensor, an accelerometer, and a gyroscope. It is lightweight approximately six grams, with a width of 7.9 mm and thickness of 2.5 mm (Oura, 2021). The ring uses a Bluetooth chipset to connect to an application on the owners' phone and will transmit the data across and onto the cloud infrastructure for processing. The ring keeps track of steps and monitors sleep patterns and cycles. It also uses Photoplethysmography to track pulse data. The ring can be purchased for €310.

### **3.4 Photoplethysmography**

Photoplethysmography (PPG) is an optical measurement method that uses a light source and a photodetector to measure changes in the light reflected through the finger. PPG waves can be used to detect heart rate data (Castaneda et al., 2018). Once, it was thought that heart rate variability could only be derived from an ECG. (Lu et al., 2009) concluded that PPG could accurately provide pulse intervals that heart rate variability can be calculated from. Heart rate variability can then be used to calculate the respiratory rate (Mirmohamadsadeghi *et al.*, 2016). This is the method that most wearable and fitness technology uses to estimate fitness scores and health data for the owner of the device.

## Chapter 4: Artificial Intelligence

### 4.1 Introduction

The term Artificial intelligence (AI) is used to describe computers that from a human perspective appear to mimic the natural intelligence of humans. AI as a field of study addresses how a small organic or inorganic (machine) brain can perceive the world around it and manipulate its surroundings or predict outcomes from what it perceives in the present. The goal is to create something with these properties (Russell and Norvig, 1995). Neural networks are one of the methods used to approach the problem.

### 4.2 Support Vector Machine

A Support Vector Machine (SVM) is a supervised learning machine learning method that can be applied to tasks with the objective of finding a suitable hyperplane in N-dimensional space to classify data points (Ghandi, 2018). An example of finding the optimal hyperplane can be seen in the image below.

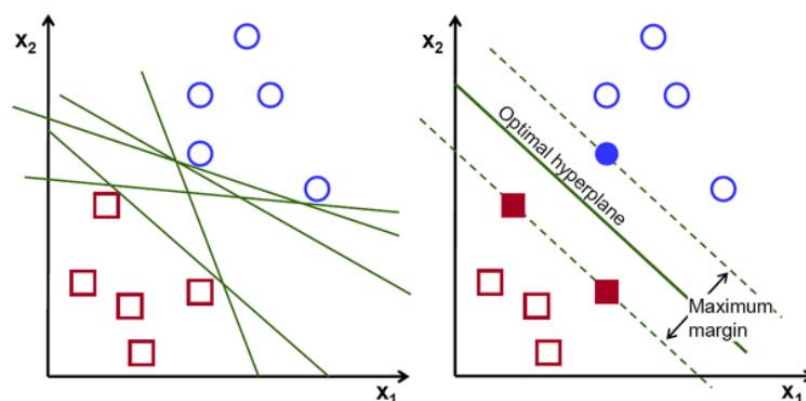


Figure 3 Possible and optimal hyperplane example (Ghandi,2018)

Although the Support Vector Machine is mostly applied to classification and regression tasks, they have been applied to a wide variety of tasks such as image and speech processing, time-series prediction, and data mining (Ukil, 2007).

### 4.3 Neural Networks

Neural Networks are loosely modelled on the human brain where each node acts as a neuron. Alan Turing first proposed a concept like a neural network in a paper he wrote in 1948, titled *“Intelligent Machines”*. In this paper, Turing spoke of *“B-Type unorganised machines”* which he refers to training this neural network as you would train a modern neural net (Turing, 2004). In a neural network, each neuron will receive data and calculate the data outputting a value based on weights supplied from the inputs. This single neuron is called a perceptron and the connections between these neurons are what collectively make up a neural network (Gallant, 1990). The perceptron was initially created by Frank Rosenblatt in 1957 is the simplest possible neural network. They follow a Feed Forward model, this means the input travels into the perceptron and the result of a calculation is output from left to right as demonstrated by Shiffman in the following figure.

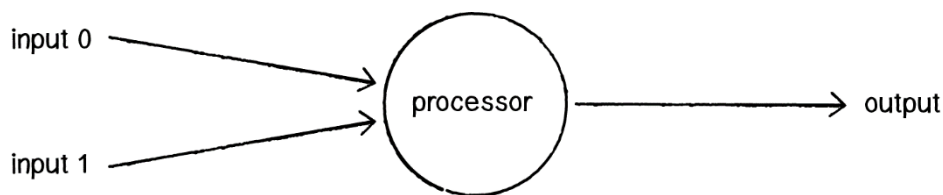


Figure 4 A perceptron (Shiffman, 2012)

The network of neurons can then be used to solve problems using input values along with weights for each value and a bias. A bias is a value that is used alongside the input values and input weights to solve a problem that occurs if the values passed in are zero. The bias value is always equal to 1 and is also weighted.

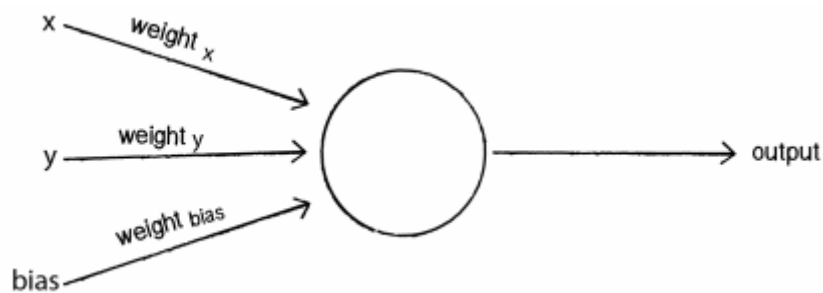
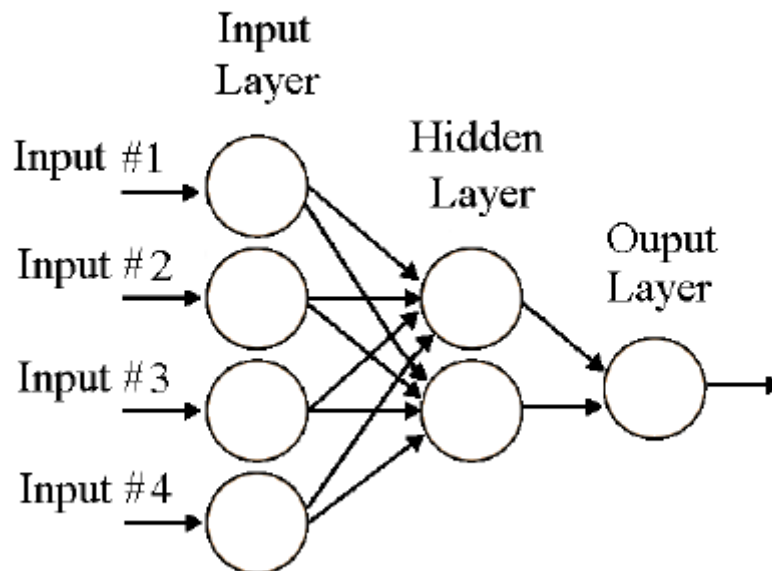


Figure 5 Perceptron with weights and bias added (Shiffman, 2012)



The next figure details the topology of a neural network with multiple inputs feeding each perceptron.



*Figure 6 Topology of a neural network*

There are many types of neural networks that all using different principles to determine their rulesets. Each type of neural network has its strengths for computing different types of problems.

#### **4.3.1 Feed Forward Neural Network**

A Feed Forward Neural Network is an artificial neural network that processes data in one direction just as a perceptron does. This type of Neural Network does not use loops or backpropagation and would have been the first type of neural network created. Information is processed from inputs to hidden layers to output. David Kriesel used an example of the human eye to show the working of a Feed Forward Network as shown below (Kriesel, 2007).

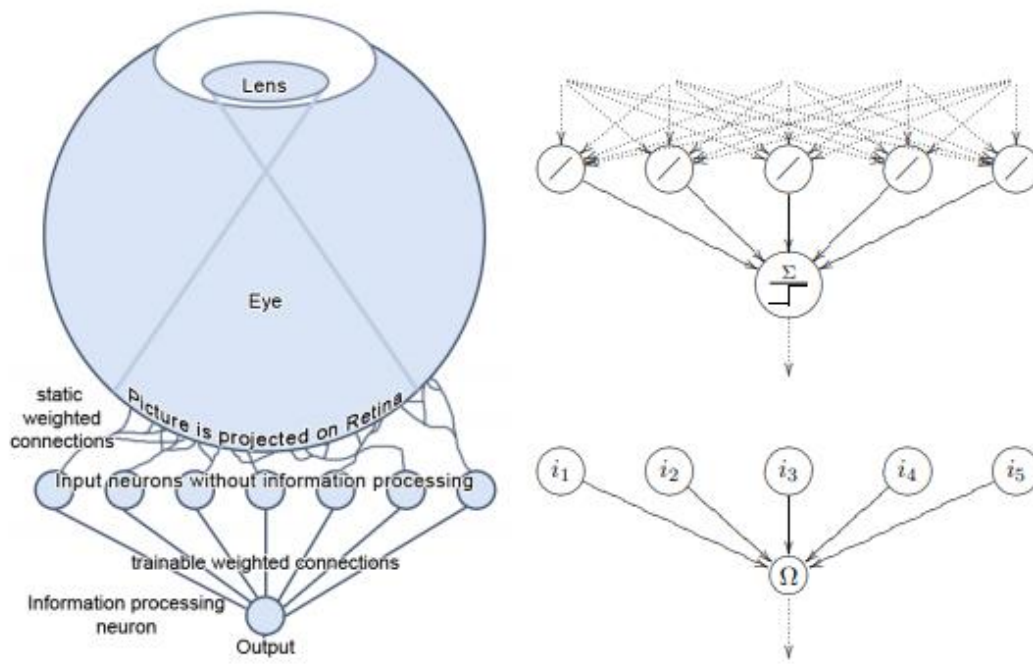


Figure 7 Retina compared to Feed Forward Neural Network (Kriesel, 2007)

### 4.3.2 Radial basis function Neural Network

Radial basis neural networks can be defined as a Feed Forward Neural Network that consists of a single layer of hidden units whose responses are the outputs of radial basis functions. The inputs of these functions are made of the distance from the input vector (activation) to its centre (or location) (Karayiannis and Mi, 1997).

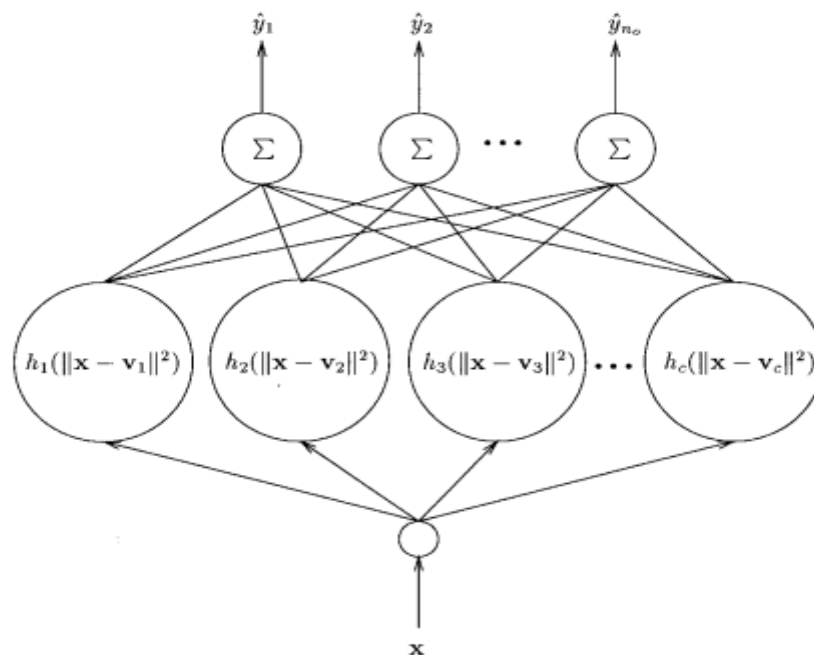


Figure 5 An RBF neural network (Karayiannis and Mi, 1997)

### **4.3.3 Convolutional Neural Network**

Convolutional neural networks and Feed Forward Neural Networks are similar in that each neuron in the network has weights and biases. These networks are most applied to images or anything with a grid-like topology. The name Convolution comes from the mathematical function convolution. Convolutional neural networks are neural networks that use convolution instead of matrix multiplication. This replacement needs only be in one of the layers to make the network a convolutional neural network (Goodfellow, Yoshua and Courville, 2016).

In the study (Hasan, Ullah, Khan and Khurshid, 2019) the CNN is compared to SVM and ANN in the task of classifying vegetation species using hyperspectral thermal infrared data. The study concluded that although all the methods showed a high accuracy in the task of classifying the data, the CNN outperformed the ANN and SVM.

## **4.4 Expert Systems**

### **4.4.1 Introduction**

An expert system is a computer system that will emulate the decision-making of an expert in the same field the system is designed for. This would consist of knowledge acquisition either from an expert or because of data mining. Expert systems were one of the earliest forms of AI and are in use in various fields today such as engineering, chemistry, medicine, industry, and more (Mehmet, Seda and Kasim, 2016). Input is presented to an expert system and a decision is made by feeding that input to an if-then decision tree. In essence, today expert systems will use artificial intelligence for knowledge acquisition and use that knowledge gained to solve specific problems (Mehmet, Seda and Kasim, 2016). The architecture of an expert system consists of three main components Knowledge base, inference system, and user interface.

### **4.4.2 Inference engine**

The inference engine of an expert system is the interpreter that will take inputs and analyse them against the rules. The main task of the Inference Engine is to trace its way through the rules and arrive at a conclusion (Tripathi, 2011). It is the brain of the system.

### **4.4.3 Expert systems with neural network integration**

A case study by (Becraft, Lee and Newell, 1991) concluded that training a neural network with sensor data and embedding the model in an expert system (designed to detect faults from sensor data in a chemical plant) exhibited good performance even in the presence of sensor noise. The neural networks can take input data from sensors and the input data will output the presence or absence of a fault. This value can then be fed to the expert system which can output a specific answer to a problem. The use of the neural network makes using sensors much easier as the expert system does not have to deal with the numbers and is presented with a Boolean value (Becraft, Lee and Newell, 1991).

## Chapter 5: Cloud Services

### 5.1 Introduction

Cloud services have become popular in recent years. These services take away a lot of the pressure of creating an infrastructure in the office with expensive servers and can help businesses to save money on server upkeep and maintenance. Cloud Services also take pressure off the customer when it comes to security and redundancy. Cloud service providers, provide services that would usually be very expensive to set up for the customer. They provide services, that would usually only be accessible to larger companies, to small start-ups or even hobbyists at home. Companies providing these services usually offer a free period and for people not in the free tier, there is a period of grace if the customer is testing out a certain service to decide if it is right for them. These companies provide services such as online computing power, cloud servers, load balancing, continuous integration, and deployment just to name a few (AWS, 2020b).

### 5.2 Amazon Web Services

#### 5.2.1 AWS Amplify

Amazon Web Services (AWS) Amplify provides a customer with the ability to build and deploy serverless applications. A customer can add a backend Application Program Interface (API) to the application which can either use REST or GraphQL and can also add S3 storage or other cloud services to the application backend. A customer can front-end frameworks such as Angular, Vue, or ReactJS as the frontend technology. There is a command-line interface available that allows a customer to add backend features from the command line at home and deploy. The workflow is a git-based workflow that allows the customer to initiate the pipeline with a build, test, deploy and verify off of a single push to the main repository (AWS, 2020a).

### 5.3 Google Cloud

#### 5.3.1 Google Firebase

Google Firebase is a Backend as a Service offered by Google. It was developed in 2011 and is their main product offered to tackle application development. Firebase is a NoSQL database and is a real-time database allowing users to sync applications securely in real-time. Since

the data can be accessed directly from the end device this means that there is no need for a server. Apps that use this technology will persist data locally. When an application is offline it will store its local changes and as soon as the application is online again it can sync with the online database adding its data which will, in turn, be updated on each connected device (Google, 2020).

## **Chapter 6: Methodology & Design**

### **6.1 Research Findings**

CNN is best suited to image classification and has a higher accuracy with images than other types of neural networks. The wearable fitness and health technology market is a relatively new market and is experiencing massive growth in recent years. It is also offering very expensive products making them less available for use for lower-income families.

Most wearable fitness and health trackers use optical sensors to collect data to monitor the person's health and then later apply algorithms on the data to determine the person's health stats.

Cloud computing services are more available now than they ever have been, meaning that the same infrastructure multimillion-euro corporations use is readily available to anybody who wants access. There are free tiers that also offer the services for free for certain periods and certain use thresholds.

### **6.2 Research Question**

The research project aimed to answer two primary questions:

1. An evaluation to determine if Convolutional Neural Networks can detect breathing anomalies in Humans, utilising audio input as a detection method.
2. Can IoT Technologies be used to host these systems effectively in tandem with cloud technology?

### **6.3 Proposed Methodology**

The argument that hypothesis testing drives quantitative research, and that theories and concepts must be identified first and then tested has been refuted in the literature (Bryma, 2001). As the research questions in this study are primarily exploratory in nature, it was felt that presenting a series of hypotheses to be tested was inappropriate. Instead, several research objectives were formulated:

- Could an expert system be developed using CNN for the knowledge acquisition phase of the expert system?
- Could IoT technologies be used with Artificial Intelligence and Cloud technologies to detect rapid changes in Human vital signs leading to the prediction of Human life cycle events?

The importance of choosing one's research design to compliment the nature of the research questions being asked has been emphasised in the literature (Creswell, 2002; Mertler and Charles, 2005; Cohen, Manion and Morrison, 2007). The research design for the present study was guided by the research questions posed. A methodology was then designed that could best address these research questions.

The project aims to build a system to detect anomalies and defects in a Person's vital signs. This system will use a Convolutional Neural Network (CNN) trained to detect symptoms of respiratory illness from an audio input listening to the persons breathing. The CNN will be trained on spectrograph images, these images will be created from the audio taken from a dataset of audio clips of people with respiratory diseases. The audio clips will be labelled with the corresponding disease. The dataset has timestamps allowing the audio to be refined down to the start and finish of a breath. Along with this, sensor data from a person can be run through an algorithm to detect anomalies from these sensors. Pulse data will be collected from a MAX30105 sensor. The system will use photoplethysmography (PPG) and will be able to detect a person's heart rate, temperature and the data samples sent back from the sensor can be used to calculate a person's heart rate variability along with resting heart rate. Using this data, the system assesses a person's initial health and over time detects changes from a person's normal vital signs. This system will be using IoT technologies and will be loaded onto a Raspberry Pi® single-board computer and will take advantage of the Raspberry Pi's® general-purpose input/output pins (GPIO) and will connect the sensors directly to these pins. A Web Application will be developed to view the person's data and be deployed to AWS® making use of AWS® cloud technologies.



## 6.4 Design

### 6.4.1 Architecture Diagram

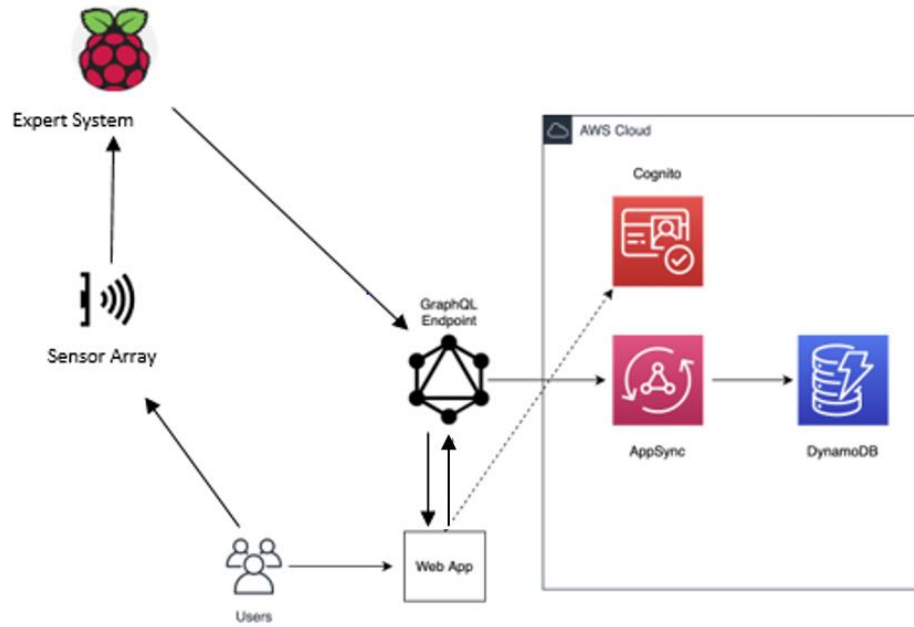


Figure 8 System architecture diagram

### 6.4.2 Functional Specifications

Using the MoSCoW method project deliverables have been prioritised.

Must Have	1
Should Have	2
Could Have	3
Won't Have	4

Feature ID	Feature Detail	MoSCoW Prioritisation
01	GUI Interface	1
02	CNN for Spectrographs	1
03	NN for sensor data	3
04	Notify user of results	1
06	Train NN on Fitbit data	3
07	Data available to user	1
08	User ability to configure	4
9	Be trained using the cloud	2
10	Be deployed to the cloud	3

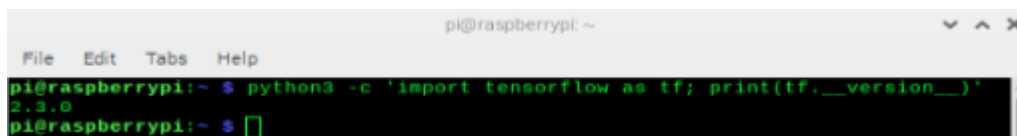
## 6.5 Prototype

### 6.5.1 Set up Hardware for the system

Prototype	Start Date	Finish Date
1	01/11/2020	02/11/2020

Task Number	Details	Status
1	Purchase Raspberry Pi® hardware and monitor	Complete
2	Install Jupyter Notebook on machine	Complete
3	Install TensorFlow and Keras on the machine	Complete

Configuring the Raspberry Pi® to be able to load a model using python or Jupyter Notebook was a must for this project moving forward. Once it was shown to possible for the Raspberry Pi® to run TensorFlow the project could move forward, as this software was essential for using a Keras model on the device.

A terminal window on a Raspberry Pi. The title bar reads 'pi@raspberrypi: ~'. The menu bar contains 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows a command being executed: 'python3 -c 'import tensorflow as tf; print(tf.\_\_version\_\_)''. The output is '2.3.0'. The prompt 'pi@raspberrypi:~ \$' is visible at the end of the line.

```
pi@raspberrypi:~ $ python3 -c 'import tensorflow as tf; print(tf.__version__)'
2.3.0
pi@raspberrypi:~ $
```

Figure 9 TensorFlow version on device

## Chapter 7: Implementation

### 7.1 Sprints

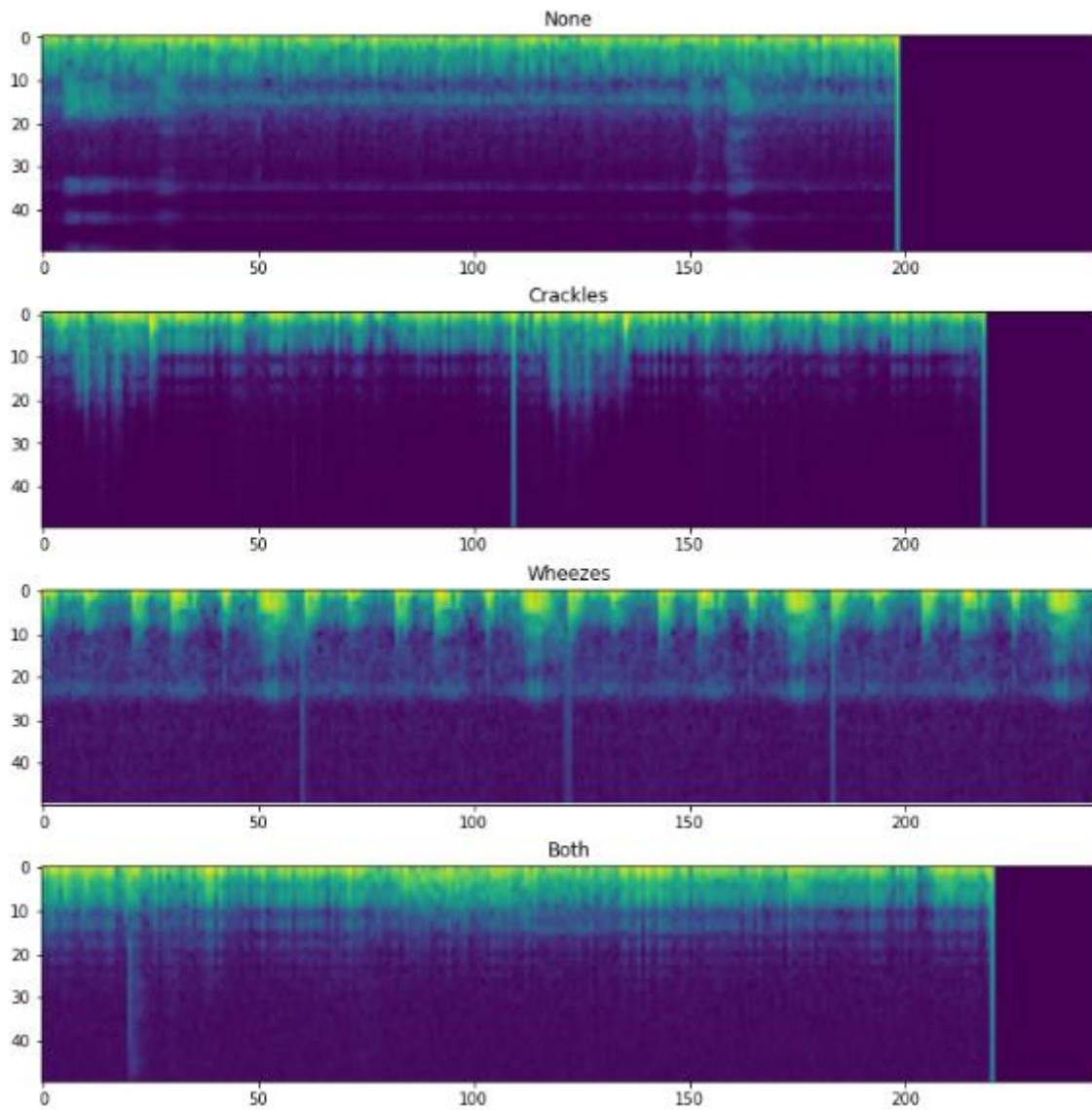
#### 7.1.1 Sprint 1: Train CNN model using respiratory data.

Sprint	Start Date	Finish Date
1	01/11/2020	02/11/2020

Task Number	Details	Status
1	Download dataset from Kaggle	Complete
2	Follow a tutorial to train a CNN to detect different types of breathing defects using Google Colab®	Complete
3	Save model and load it onto Raspberry Pi®	Complete
4	Feed test data into the model and get predictions	Complete
5	Format predictions using argmax function to view	Complete

The dataset used is available at <https://www.kaggle.com/vbookshelf/respiratory-sound-database>

The CNN was trained by converting audio clips of respiratory sounds to spectrographs and using them to train the CNN.



*Figure 10 Labelled spectrographs from audio clips.*

Figure 10 shows the audio.wav files converted into spectrographs.

The model prediction accuracy came to around 71 percent. As seen in Figure 11.

	precision	recall	f1-score	support
none	0.75	0.84	0.79	760
crackles	0.71	0.60	0.65	380
wheezes	0.62	0.60	0.61	184
both	0.55	0.44	0.49	109
accuracy			0.71	1433
macro avg	0.66	0.62	0.64	1433
weighted avg	0.71	0.71	0.71	1433

```

[[638 70 44 8]
 [133 228 6 13]
 [ 50 5 110 19]
 [ 29 16 16 48]]

```

*Figure 11 Accuracy score for CNN model*

### 7.1.2 Sprint 2: Create Specific Disease identifier CNN.

Sprint	Start Date	Finish Date
2	01/12/2020	20/12/2020

Task Number	Details	Status
1	Using the same data, Train the CNN to detect specific diseases.	Complete
2	Use audio files and the disease labels to create spectrographs.	Complete
3	Train model using spectrographs	Complete
4	Test model.	Complete

0	101	3.00	F	NaN	19.0	99.0	URTI
1	102	0.75	F	NaN	9.8	73.0	Healthy
2	103	70.00	F	33.00	NaN	NaN	Asthma
3	104	70.00	F	28.47	NaN	NaN	COPD
4	105	7.00	F	NaN	32.0	135.0	URTI
5	106	73.00	F	21.00	NaN	NaN	COPD

Figure 12 Data in respiratory illness dataset.

Figure 12 shows a snapshot of the data relating to each disease. There is missing data, but that data is not needed to create the model. Once the model was created, the number of epochs was tested on various attempts and it was found that 45 epochs did show the best results for prediction.

This time around the audio feature were extracted using the code below in Figure 13.

```
def audio_features(filename):
    sound, sample_rate = sf.read(filename)
    stft = np.abs(librosa.stft(sound))

    mfccs = np.mean(librosa.feature.mfcc(y=sound, sr=8000, n_mfcc=40, fmin=30).T,axis=0)
    chroma = np.mean(librosa.feature.chroma_stft(S=stft, sr=8000).T,axis=0)
    mel = np.mean(librosa.feature.melspectrogram(sound, sr=8000, fmin=30).T,axis=0)
    contrast = np.mean(librosa.feature.spectral_contrast(S=stft, sr=8000, fmin=30).T,axis=0)
    tonnetz = np.mean(librosa.feature.tonnetz(y=librosa.effects.harmonic(sound), sr=sample_rate, chroma=librosa.feature.chroma_qct(y=sound, sr=8000, fmin=30).T,axis=0)

    concat = np.concatenate((mfccs,chroma,mel,contrast,tonnetz))
    return concat
```

Figure 13 Audio Features (Denton, 2020)

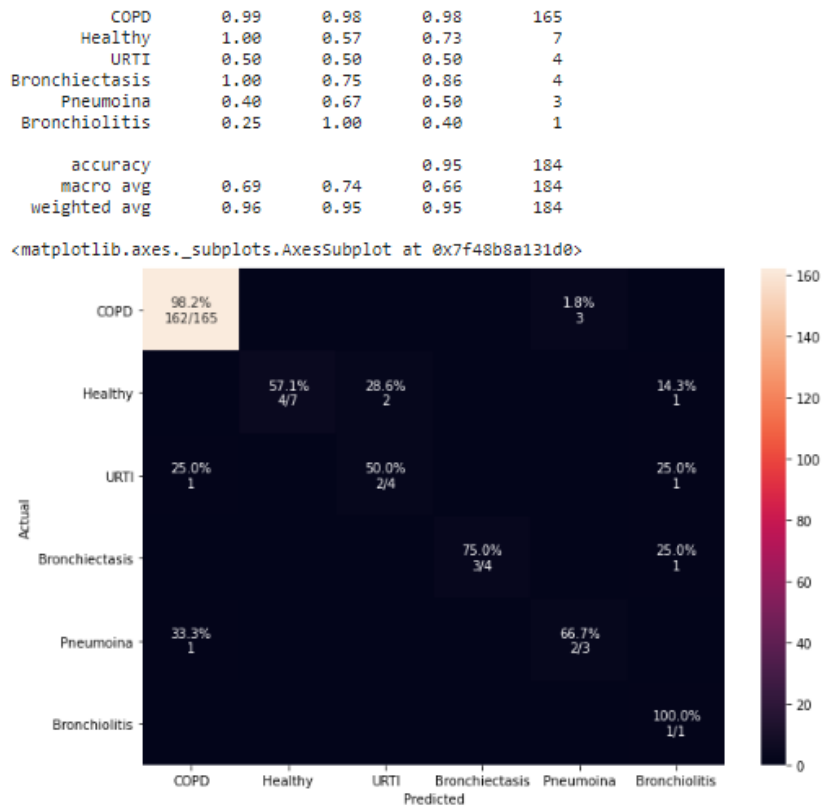


Figure 14 Accuracy score and confusion matrix

Figure 14 shows the model was 95 percent accurate on the test data.

Once the model has been trained and tested, the model was saved to google drive in Hierarchical Data Format version 5 (HDF5) format. This format stores the model's architecture, weights values, and compile() information.

```
[ ] model.save("/content/gdrive/My Drive/RespModel.h5")
```

Figure 15 Saving model to Google Drive.



### 7.1.3 Sprint 3: Test CNN on Raspberry Pi

Sprint	Start Date	Finish Date
3	01/01/2021	14/01/2021

Task Number	Details	Status
1	Configure Raspberry Pi®	Complete
2	Download the Model	Complete
3	Create a script to load each model, record and convert audio to a spectrograph, and then feed that spectrograph to the model.	Complete
4	Run Script on Raspberry Pi®	

The script first loads the model into the script (Figure 16).

```
model = keras.models.load_model('/home/pi/Downloads/Jan21RespModel.h5')
```

Figure 16 Loading Keras CNN model.

The script then records a five-second clip and runs the .wav file through a function that returns an object of type ndarray. The array that is returned is of size [1, 193,1] and this array is then fed to the Keras model.predict() method. The model.predict() method will return the predicted values for the recorded sound. The output of the script can be seen in Figure 17 below. Here [0] represents a label. E.g., “Healthy”, “COPD”, “LRTI”.

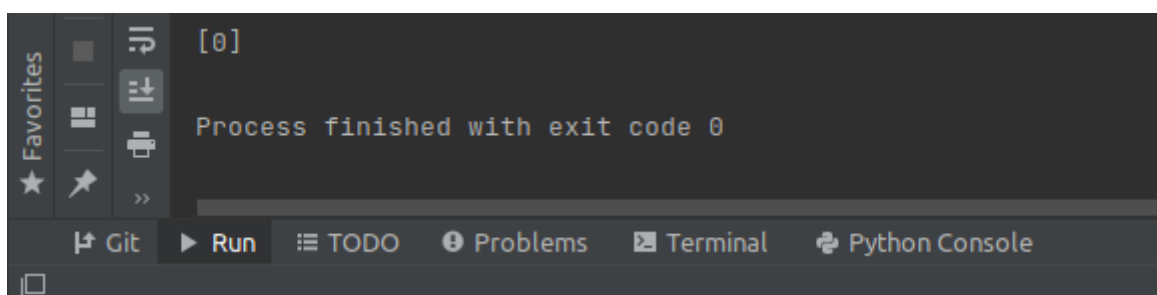


Figure 17 Script output on laptop IDE

To run the code on the Raspberry Pi® several packages had to be installed. PyAudio for recording the data. Librosa for creating the spectrogram and Keras for loading the model was the main packages. Once installed the code will run on the Raspberry Pi® with no problems output can be seen below in Figure 18.

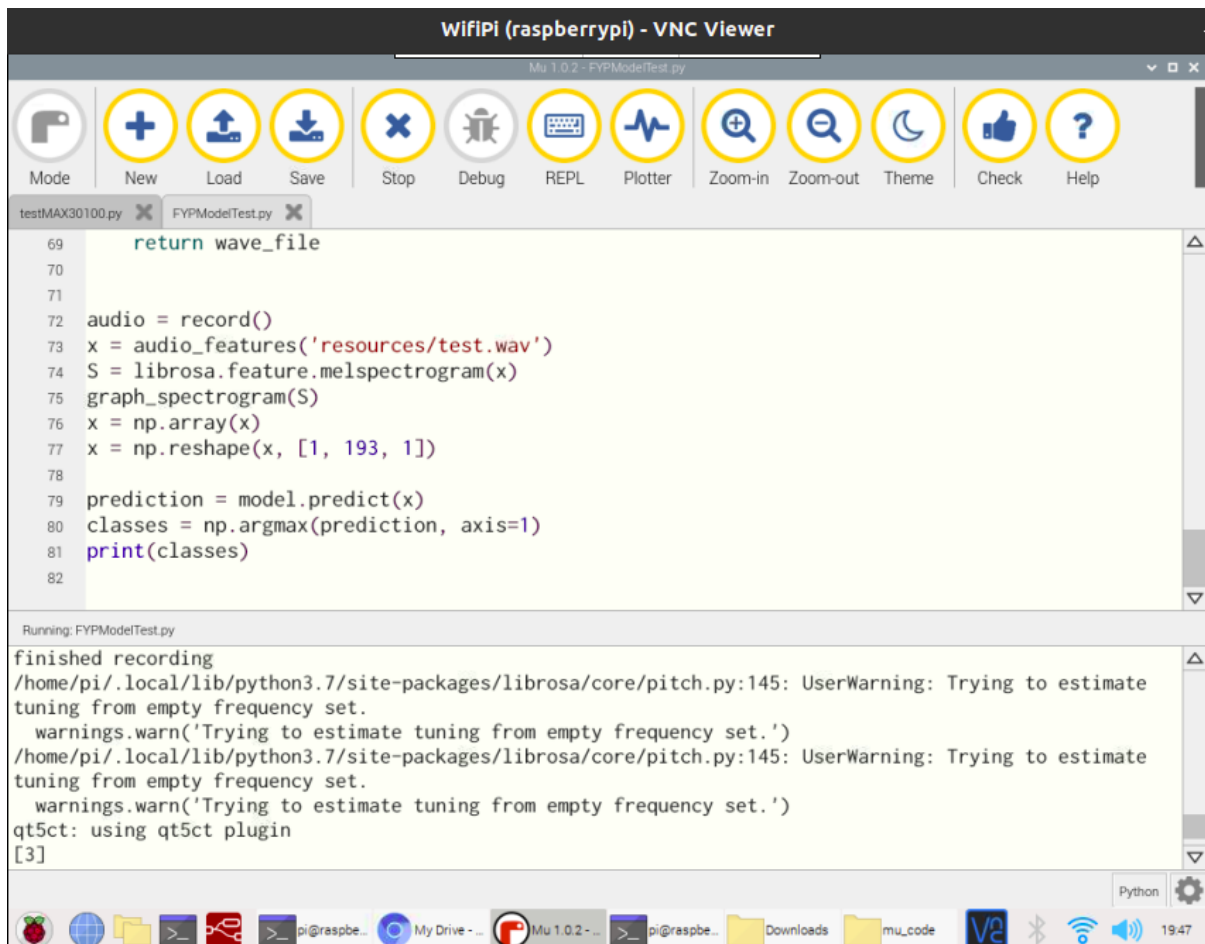


Figure 18 Output on Raspberry Pi®

#### 7.1.4 Sprint 4: Set up Raspberry Pi and MAX30105 Sensor to take readings.

Sprint	Start Date	Finish Date
4	15/01/2021	29/01/2021

Task Number	Details	Status
1	Enable i2c interface	Complete
2	Solder pins to Max30105® sensor	Complete
3	Wire sensor to Raspberry Pi®	Complete
4	Clone and install the MAX30105 library from the Pimoroni GitHub page	Complete
5	Write script to collect pulse and temperature data	Complete
6	Prepare data	Complete

The i2c interface can be enabled through the raspi-config options and set to route power to the pins. This allows the sensor to be attached to the Raspberry Pi®. First pins were soldered onto the chip so wires could be attached Figure 19.

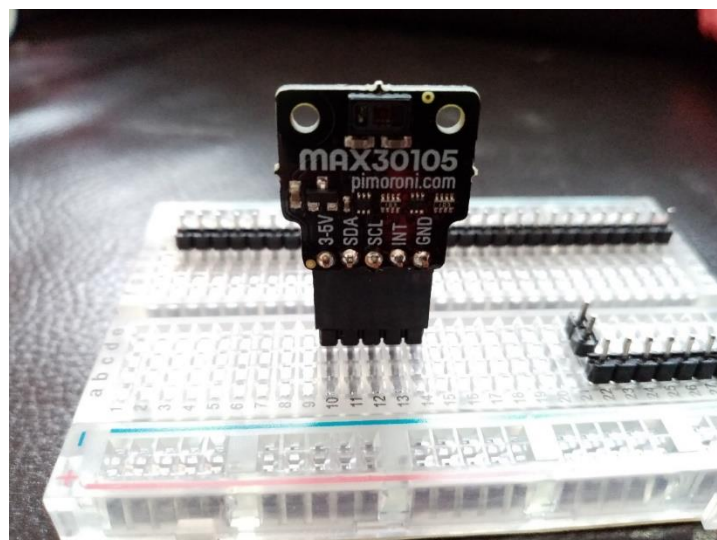


Figure 19 MAX30105® sensor with pins soldered on

Next, the wires could be attached as shown in the circuit diagram Figure 20 and Figure 21.

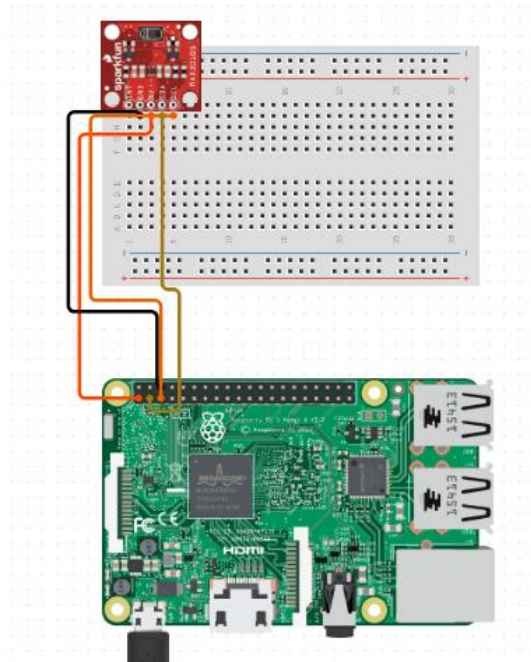


Figure 20 Circuit diagram

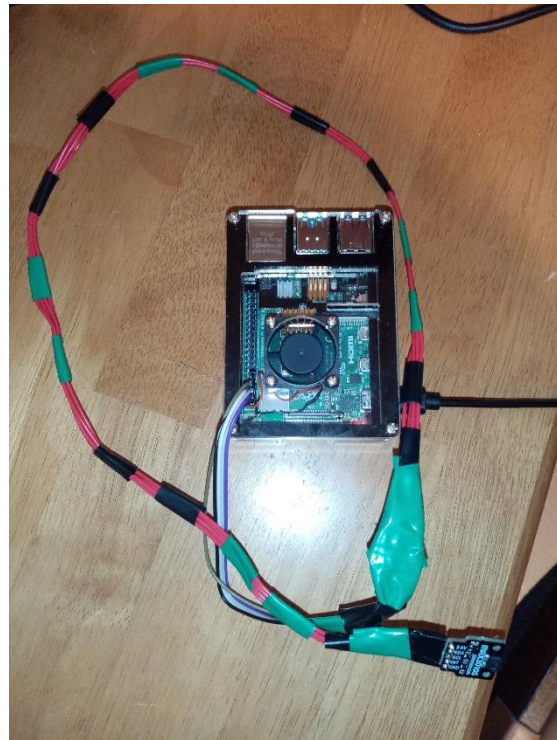


Figure 21 Real life circuit

The Library was cloned and installed by using the following commands in the terminal:

- git clone <https://github.com/pimoroni/max30105-python>
- cd max30105-python
- sudo ./install.sh

The library was modified and used in a script to collect sensor data Figure 22.

```

In [2]: import matplotlib.pyplot as plot
import time
from max30105 import MAX30105, HeartRate

max30105 = MAX30105()
max30105.setup(leds_enable=2)

max30105.set_led_pulse_amplitude(1, 0.2)
max30105.set_led_pulse_amplitude(2, 12.5)
max30105.set_led_pulse_amplitude(3, 0)

max30105.set_slot_mode(1, 'red')
max30105.set_slot_mode(2, 'ir')
max30105.set_slot_mode(3, 'off')
max30105.set_slot_mode(4, 'off')

hr = HeartRate(max30105, time)

delay = 5
print("Starting readings in {} seconds...\n".format(delay))
time.sleep(delay)

try:
    while True:
        samples = max30105.get_samples()
        if samples is not None:
            for i in range(0, len(samples), 2):
                # Process the least significant byte, where most wiggliness is
                ir = samples[i + 1] & 0xff
                d = hr.low_pass_fir(ir)
                with open("HRVdata.txt", "a") as file_object:
                    d2 = int(d/3)
                    file_object.write(str(round(time.time()*1000)) + ";" +str(d2) +",")
                    time.sleep(1.0 / 100) # 400sps 4 sample averaging = 100sps
except KeyboardInterrupt:
    pass

Starting readings in 5 seconds...

```

Figure 22 Data collection script

The code imports the time, MAX30105®, and Heart Rate modules and used them to write data to a file called HRVdata.txt. The sensor is very sensitive to touch which can skew the data so the code will only write to the file once a heartbeat has been detected. Next, a small python script was written to separate each row to a new line using ',' as a delimiter and ';' as markers to mark each column. The data was then written to a CSV file Figure 23.

```

1  f1=open("HRVdata.txt","r+")
2  input=f1.read()
3  print(input)
4  input=input.replace(',','\n')
5  print(input)
6  input=input.replace(';',';')
7  print(input)
8  f2=open("HRVdata.csv","w+")
9  f2.write(input)
10 f1.close()
11 f2.close()

```

Figure 23 Data preparation script

The final data can be seen below in Figure 24.

1614086382528	13
1614086382561	15
1614086382601	17
1614086382640	18
1614086382681	19
1614086382721	20
1614086382761	20
1614086382802	21
1614086382842	22
1614086382882	23

*Figure 24 Final sensor data*

The Same was done to collect pulse rate data and the data can be seen below in Figure 25.

Timestamp	HR	Temp
Thu Feb 11 16:01:25 2021	75	23
Thu Feb 11 16:01:26 2021	68	24
Thu Feb 11 16:01:27 2021	79	24
Thu Feb 11 16:01:28 2021	71	24
Thu Feb 11 16:01:29 2021	65	24
Thu Feb 11 16:01:29 2021	65	24
Thu Feb 11 16:01:30 2021	58	24
Thu Feb 11 16:01:31 2021	62	24

*Figure 25 Heart rate data*

### 7.1.5 Sprint 5: Analyse the data and create a cloud Web Application to view data.

Sprint	Start Date	Finish Date
5	30/01/2021	14/02/2021

Task Number	Details	Status
1	Analyse Collected data	Complete
2	Create and Deploy AWS amplify Web Application	Complete

Using the collected data, peaks were first identified to identify each heartbeat and these peaks were saved to a NumPy array for use later in calculating heart rate variability. This data is visualised along with the peaks and minima Figure 26.

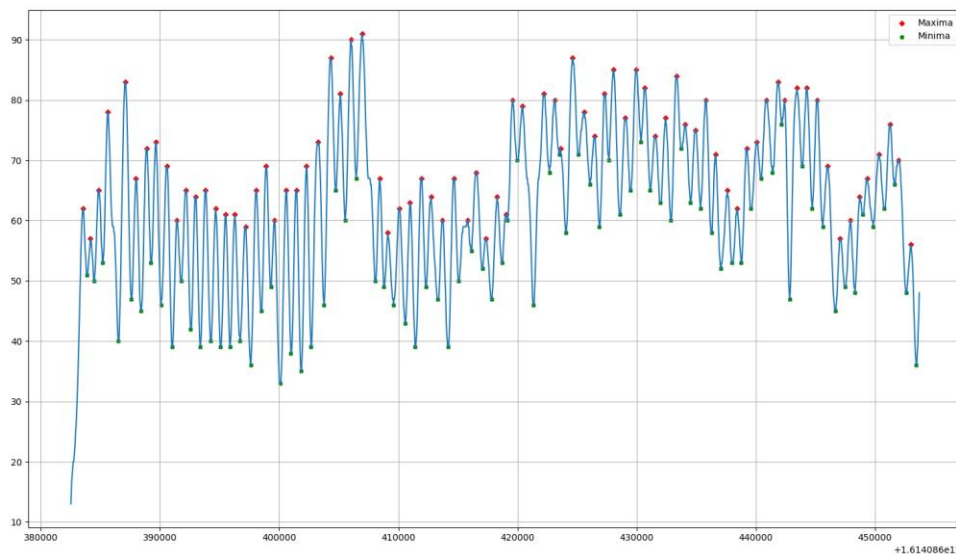


Figure 26 PPG data

Next, the root mean square of successive differences between heartbeats was calculated, Figure 27.

```

x = []
for index, value in peak_pos.items():
    x.append(value)

xdiff = [x[n]-x[n-1] for n in range(1,len(x))]

def square_list(x):
    for i in range(0, len(x)):
        x[i] = x[i] * x[i]
    return x

def Average(lst):
    return sum(lst) / len(lst)

HRV = str(round(math.sqrt(Average(square_list(xdiff)))))
print(HRV)

```

Figure 27 HRV script

When compared to data from the same person wearing a Fitbit® smartwatch, the HRV calculated was in was the same range Figure 28. A difference of 5 milliseconds.

## Heart rate variability (HRV)

Recent sleep: 35 milliseconds

[LEARN MORE](#)



Figure 28 Fitbit® HRV data



A Web Application was then developed using ReactJS and AWS® Amplify.

The starting point for the Web Application was to first create react app with the command:

```
npx create-react-app **appname**
```

Next, install the amazon amplify command-line interface with the command:

```
npm install -g @aws-amplify/cli
```

A GraphQL API, Authentication, and S3 Storage were added to the application using the commands:

- amplify add storage.
- amplify add api.
- and amplify add auth.

Once the setup was complete and configuration information was input to the CLI dialog the backend can be pushed to the cloud using the command amplify push.

Once the Application was created on AWS® and the GitHub repository was connected to the amplify application a pipeline was then set up to build the application automatically once new code was pushed to the app repository on GitHub (Figure 29).

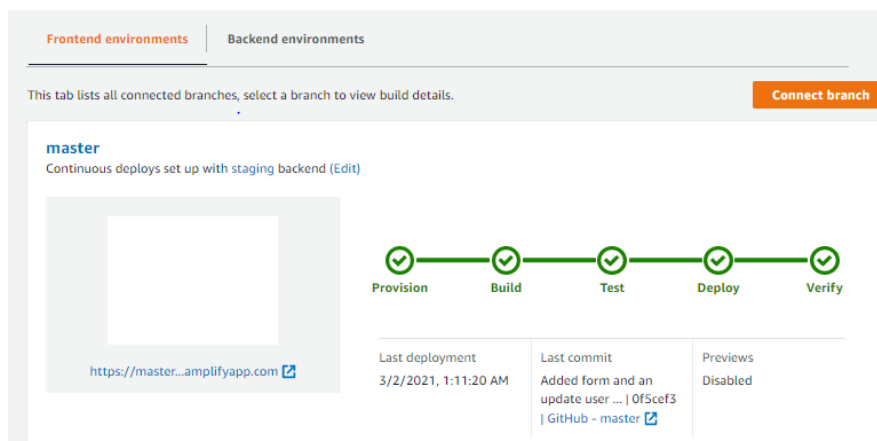


Figure 29 Application pipeline

The folder structure of the application can be seen in Figure 30.

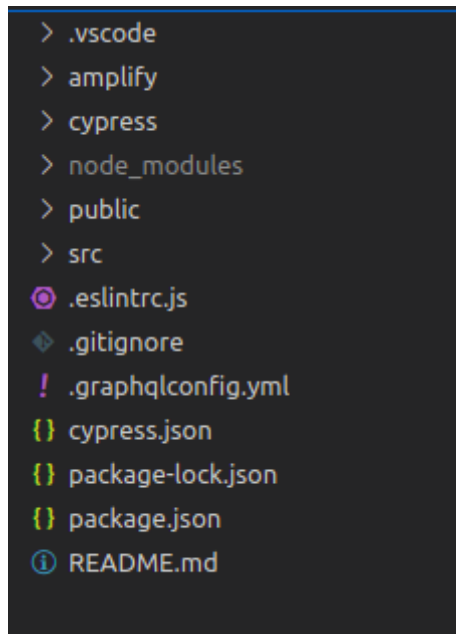


Figure 30 Folder structure

Now the application was ready to be modified to meet the needs of this project. A dashboard component was added along with a navigation bar component and a component to input the user's height, weight, and age. A subscription was then set up to listen to specific changes in the data that would in turn update the application with these changes in real-time without refresh, offering real-time updates. An example of the code to set up a subscription can be seen below in Figure 31. This code will act as a listener, and when the HRV value in the database is updated, it will return the values id, value, createdAt, and updatedAt.

```
export const onUpdateHrv = /* GraphQL */ `
  subscription OnUpdateHrv {
    onUpdateHRV {
      id
      value
      createdAt
      updatedAt
    }
  }
`;
```

Figure 31 Subscription code

Next, the script for the Raspberry Pi® was written (Figure 32).

This script will run several smaller scripts:

- First, take samples from the optical sensor module in the MAX30105® sensor and write them to HRVValues.txt.
- HRV\_preparation.py will format the data and write it to a CSV file.
- get\_hrv\_value\_from\_samples.py will calculate the HRV value from the session data and will send the HRV value to the GraphQL API.
- on\_beat() method is called. This method will write the pulse rate data to the file every time a heartbeat is detected. The beat per minute value is added to a Welfords Algorithm object and once the session is complete the mean value of pulse rate is sent to the Web Application through the GraphQL API.
- get\_temperature() is called to get the temperature from the MAX30105 sensor.
- The user is notified that the chest recording will start in 10 seconds, and then the record method is called which takes a five-second recording.
- The recording is then run through the audio\_features() method which returns a numpy array of audio feature data.
- The data is then reshaped and input into the CNN model for a prediction. The prediction is then given its label and sent through an API call to the Web Application.

```

try:
    print("stageone")
    while stage_one<4:
        samples = max38185.get_samples()
        if samples is not None:
            for i in range(0, len(samples), 2):
                ir = samples[i + 1] & 0xff
                d = hr.low_pass_fir(ir)
                with open("HRVdata.txt", "a") as file_object:
                    d2 = int(d/3)
                    file_object.write(str(round(time()*1000)) + ";" +str(d2) +",")
                    sleep(1.0 / 100)
                stage_one+=0.01
    os.system("python3 ~/proto_hb_data/HRV_preparation.py")
    os.system("python3 ~/proto_hb_data/get_hrv_from_samples.py")
    print("stagetwo")
    hr.on_beat(display_hearttrate, average_over=4)
    mean_hb = list(w.mean)
    mean_hb = str(round(mean_hb[1]))
    command="python3 ~/proto_hb_data/AppRequest/APICreateRHR.py "+mean_hb
    print(mean_hb)
    os.system(command)
    print("stagethree")
    temp = max38185.get_temperature()
    temp = str(round(temp))
    temp_command="python3 ~/proto_hb_data/AppRequest/TempCreateAPI.py "+temp
    os.system(temp_command)

    delay = 10
    print("Starting chest recording in {} seconds...\n".format(delay))
    sleep(delay)
    print("Recording started...")
    audio = record()
    x = audio_features('resources/test.wav')
    S = librosa.feature.melspectrogram(x)
    graph_spectrogram(S)
    x = np.array(x)
    x = np.reshape(x, [1, 193, 1])

    prediction = model.predict(x)
    classes = np.argmax(prediction, axis=1)
    disease_list = ["COPD", "Healthy", "URTI", "Bronchiectasis", "Pneumonia", "Bronchiolitis"]
    classes = classes[0]
    pred_command="python3 ~/proto_hb_data/AppRequest/APIPredictionEntry.py "+disease_list[classes]
    os.system(pred_command)
    print("Dashboard updated")

```

Figure 32 Raspberry Pi<sup>®</sup> script to collect data.

Once the script has been run, each subscription Web Socket will update the Web Application with the new values as shown in Figure 33.

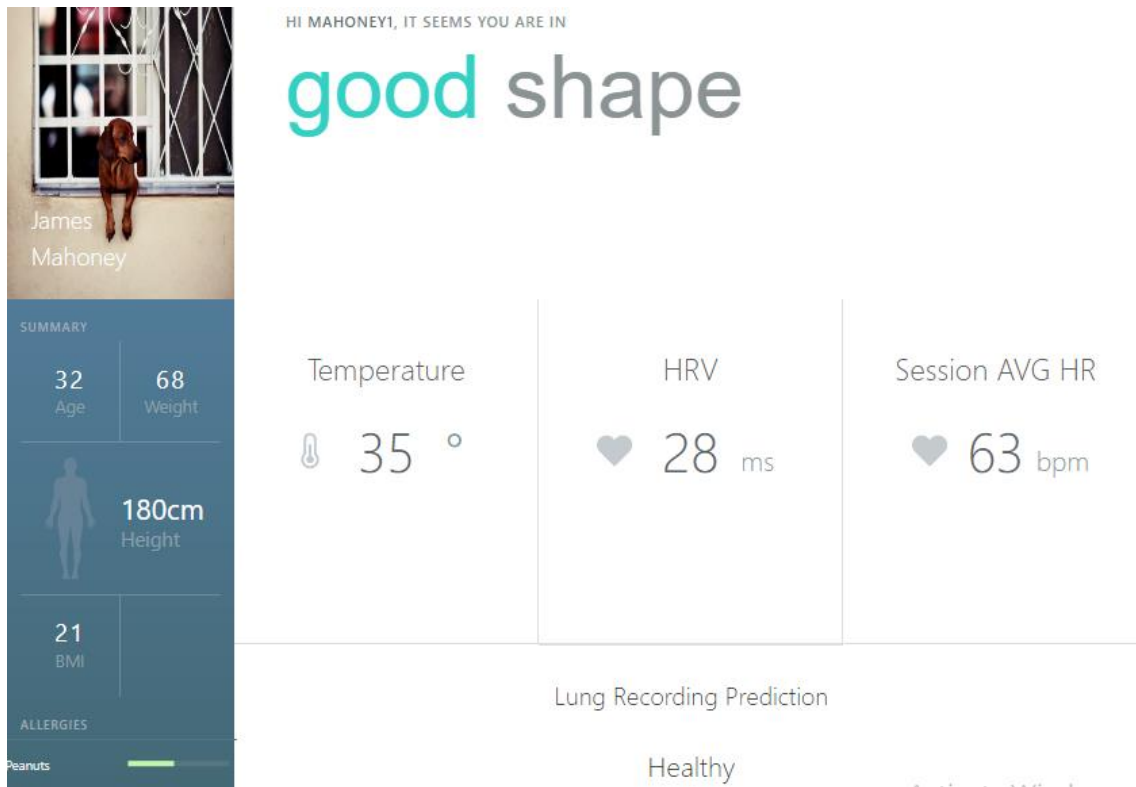


Figure 33 Updated Web Application.

An edit account details page was created so the user can update height, age, and weight values and the user's body mass index can be calculated. This page was created as a component using ReactJS. This page (Figure 34) will update the user details and will in turn update the details on the dashboard component.

V.0 FYP . Dashboard Edit Account Details

---

Age:

Weight:

Height:

Figure 34 Edit account details component.

Next, the client-side methods were written to act as the client-side inference engine for the system. These functions will be called once the data from the final part of the data collection script load onto the front-end. The final piece of data, in this case, is the result of the CNN model. An example of a function can be found seen in Figure 35. This method will return a result on whether the HRV value is healthy or not based on the person's age and HRV value. These rules were taken from (Moore, 2016) who performed a study on 24,764 people and found the mean results per age group.

```

const HRVScore = () => {
  let HRVScore;
  let age = userDetails.age;
  if(age>17&&age<26&&Between(HRVValue,60, 76))
    HRVScore = "Good";
  else if(age>=26&&age<36&&Between(HRVValue,55,72))
    HRVScore = "Good";
  else if(age>=36&&age<46&&Between(HRVValue,52,69))
    HRVScore = "Good";
  else if(age>=46&&age<56&&Between(HRVValue,47,68))
    HRVScore = "Good";
  else if(age>=56&&age<66&&Between(HRVValue,42,64))
    HRVScore = "Good";
  else if(age>=66&&age<76&&Between(HRVValue,40,63))
    HRVScore = "Good";
  else if(age>75&&Between(33,68))
    HRVScore = "Good";
  else
    HRVScore = "HRV not in normal range, this could be normal if you are sick or have recently been stressed or participated in strenuous activity";
  return HRVScore
}

```

Figure 35 Client-side HRV ruleset

Functions were also written for BMI, CNN model prediction, and temperature. These functions were then all run together, and the results were displayed on a ReactJS component (Figure 36). In the result section below the person is deemed to be in bad shape because of the low HRV result, otherwise, the person would be deemed to be in good health.

## Web Application

V.0 FYP Dashboard Edit Account Details

HI MAHONEY1, IT SEEMS YOU ARE IN **Bad shape**

James Mahoney

SUMMARY

32 Age

180cm Height

21 BMI

ALLERGIES

Peanuts

**Results of last session**

- HRV not in normal range, this could be normal if you are sick or have recently been stressed or participated in stressful activity. value is 28
- This AI model predicts you are Healthy
- You have a Good temperature. temp is 35
- You are Healthy with a BMI of 21

Close Window

Session AVG HR **63 bpm**

Lung Recording Prediction

Healthy

Figure 36 User session results

## Chapter 8: Findings & Conclusions

### 8.1 Results

The research project set out to answer two primary questions:

1. Can Convolutional Neural Networks detect breathing anomalies in Humans, using an audio input as a detection method?
2. Can the component system, to answer question 1 (An expert system) be hosted effectively using IoT Technologies in tandem with cloud technology?

The approach that was adopted by the author to answer these questions was to develop a prototype. This prototype consisted of a Raspberry Pi® 4b single board computer, a Max30105 optical sensor, and a microphone. This hardware was then supported by a Convolutional Neural Network (CNN) which was created, and then trained on Google Colab®. The CNN was trained using a respiratory illness sounds database that was downloaded from Kaggle. The author trained the network to these sounds by extracting the audio features from them and creating a spectrograph from those features for each of the sounds. Each of the spectrographs was labelled with the corresponding disease, thus the author had a baseline reference of data to which a comparison could be made to the “live” data that would be generated when the device was tested. The data was then separated into training and testing data using a method that takes the whole dataset as input and returns two subsets of the original, a training subset and a testing subset. The CNN was first trained using the training subset. This separation allowed the model to be trained on new data which allowed for a more accurate simulation of how the model would behave with live data. Once the model was trained and tested showing 95% accuracy, the model was then saved to Google drive. The CNN model was downloaded onto the Raspberry Pi®, and a script created by the author loaded the CNN model and recorded an audio clip from a microphone attached to the Raspberry Pi®. The audio clip was then processed and input to the CNN for a prediction.

Functionality was then added to the script to collect data from a person using the MAX30105 sensor which was then passed through a series of algorithms that extracted meaningful data. The algorithms calculated the root mean square of successive differences

between detected heartbeats. The output of this calculation is the heart rate variability of the person, which can be used as an indicator to determine the health of the person. The temperature was also collected along with the average heart rate for the session. The data extrapolated from the MAX30105 sensor and the CNN was then sent via HTTPS through an API to an AWS® cloud-hosted DynamoDB instance. A Web Application that was developed using ReactJS then receives the data via web sockets and processes the data using client-side inference methods that calculate and display the results of the system to a dashboard for viewing.

The nett result of this technological innovation was that the author was able to attach the device to themselves, and other subjects, and read and detect breathing anomalies. Furthermore, the author was able to measure, detect, and conclude from the various baseline metrics their levels of fitness.

## **8.2 Conclusion**

This dissertation set out to demonstrate that a collection of technologies can be used to observe and detect anomalies in human vital signs. The author believes that the trend in the purchase and use of mass-produced technology in this field will manifest itself in even greater use of such devices in the future. The size of the prototype developed for the proof of the concept of this dissertation was large, however, improvements in the field of photomicrography could make such devices smaller.

While it is the opinion of the author that there is a long way to go before any solution will ultimately take the place of a human doctor, devices such as the prototype developed could assist medical professionals in detecting the presence of lung diseases correctly. If implemented as a compact wearable device, the prototype could be used for continuous monitoring and could be used as an early detector of undiagnosed underlying conditions and perhaps play a part in the early diagnoses of these conditions.

It can be concluded from the performance of the prototype developed that these research questions are shown to be proven, and that the technology is an effective solution for observing and detecting anomalies in human vital signs.



## References

- Atallah, L. *et al.* (2010) 'Sensor Placement for Activity Detection Using Wearable Accelerometers', in *2010 International Conference on Body Sensor Networks. 2010 International Conference on Body Sensor Networks (BSN)*, Singapore, Singapore: IEEE, pp. 24–29. doi: 10.1109/BSN.2010.23.
- AWS (2020a) *AWS Amplify Console - Modern Application Development on AWS*. Available at: <https://docs.aws.amazon.com/whitepapers/latest/modern-application-development-on-aws/aws-amplify-console.html> (Accessed: 8 March 2021).
- AWS (2020b) *Overview of Amazon Web Services*. Available at: <https://docs.aws.amazon.com/whitepapers/latest/aws-overview/introduction.html> (Accessed: 8 March 2021).
- Becraft, W. R., Lee, P. L. and Newell, R. B. (1991) 'INTEGRATION OF NEURAL NETWORKS AND EXPERT SYSTEMS FOR PROCESS FAULT DIAGNOSIS', p. 6.
- Bryma, A. (2001) 'Social Research Methods', *Oxford University Press*.
- Castaneda, D. *et al.* (2018) 'A review on wearable photoplethysmography sensors and their potential future applications in health care', *International journal of biosensors & bioelectronics*, p. 195.
- Clark, J. (2016) *What is the Internet of Things, and how does it work?* Available at: <https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/> (Accessed: 7 December 2020).
- Cohen, L., Manion, L. and Morrison, K. (2007) *Research Methods in Education*. Routledge.
- Creswell, J. W. (2002) *Research Design: Qualitative, Quantitative, and Mixed Method Approaches*. Second. Sage Publications Inc.
- Denton, M., 2020. CNN: Disease Classification, Linked Features. [online] Kaggle.com. Available at: <<https://www.kaggle.com/markdenton/cnn-disease-classification-linked-features-95>> [Accessed 15 December 2021].
- Fitbit dashboard* (2021). Available at: [www.fitbit.com](http://www.fitbit.com).
- Gallant, S. (1990) 'Perceptron-based learning algorithms', *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 1, pp. 179–91. doi: 10.1109/72.80230.
- Ghandi, R., 2018. *Support Vector Machine — Introduction to Machine Learning Algorithms*. [online] towardsdatascience. Available at: <<https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>> [Accessed 7 January 2021].
- Goodfellow, I., Yoshua, B. and Courville, A. (2016) *Deep Learning*.

- Google (2020) *Firestore Realtime Database*. Available at: <https://firebase.google.com/docs/database> (Accessed: 10 March 2021).
- Hasan, M., Ullah, S., Khan, M. and Khurshid, K., 2019. COMPARATIVE ANALYSIS OF SVM, ANN AND CNN FOR CLASSIFYING VEGETATION SPECIES USING HYPERSPECTRAL THERMAL INFRARED DATA. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W13, pp.1861-1868.
- Jubran, A. (2015) 'Pulse oximetry', *Critical Care*, 19(1), p. 272. doi: 10.1186/s13054-015-0984-8.
- Karayiannis, N. B. and Mi, G. (1997) 'Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques', *Neural Networks, IEEE Transactions on*, 8, pp. 1492–1506. doi: 10.1109/72.641471.
- Kriesel, D. (2007) 'A brief introduction to neural networks'. [dkriesel.com](http://dkriesel.com).
- Lai, X. *et al.* (2013) 'A Survey of Body Sensor Networks', *Sensors (Basel, Switzerland)*, 13(5), pp. 5406–5447. doi: 10.3390/s130505406.
- Lu, G. *et al.* (2009) 'A comparison of photoplethysmography and ECG recording to analyse heart rate variability in healthy subjects', *Journal of Medical Engineering & Technology*, 33(8), pp. 634–641. doi: 10.3109/03091900903150998.
- MAXIM (2016) 'MAX30105 High-Sensitivity Optical Sensor for Smoke Detection Applications'. Maxim Integrated Products, Inc. Available at: <https://datasheets.maximintegrated.com/en/ds/MAX30105.pdf>.
- Mehmet, T., Seda, S. and Kasim, O. (2016) *Expert Systems*.
- Mertler, C. A. and Charles, C. M. (2005) *Introduction to Educational Research*. Allyn & Bacon.
- Mirmohamadsadeghi, L. *et al.* (2016) 'Real-time respiratory rate estimation using imaging photoplethysmography inter-beat intervals', in *2016 Computing in Cardiology Conference (CinC)*. *2016 Computing in Cardiology Conference (CinC)*, pp. 861–864.
- Moore, J. (2016) *Normative HRV Scores by Age and Gender [Heart Rate Variability Chart], Elite HRV*. Available at: <https://elitehrv.com/normal-heart-rate-variability-age-gender> (Accessed: 17 March 2021).
- Ornes, S. (2016) 'Core Concept: The Internet of Things and the explosion of interconnectivity', *Proceedings of the National Academy of Sciences*, 113, pp. 11059–11060.
- Oura (2021). Available at: <https://ouraring.com/product/heritage-silver/step1>.
- Russell, S. J. and Norvig, P. (1995) *Artificial intelligence: a modern approach*. Englewood Cliffs, N.J: Prentice Hall (Prentice Hall series in artificial intelligence).
- Saha, H. N., Mandal, A. and Sinha, A. (2017) 'Recent trends in the Internet of Things', in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*. *2017*

*IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 1–4.  
doi: 10.1109/CCWC.2017.7868439.

Segil, J. (2019) *Handbook of Biomechatronics*.

*Smartwatch Market | 2020-2027 | Industry Report (2020)*. Available at:  
<https://www.mordorintelligence.com/industry-reports/smartwatch-market> (Accessed: 18 February 2021).

Soloman, S. (2009) *Sensors Handbook*. McGraw-Hill, Inc.

Tripathi, K. P. (2011) *A Review on Knowledge-based Expert System: Concept and Architecture*.

Turing, A. (2004) 'The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life: Plus The Secrets of Enigma', in. OXFORD UNIVERSITY PRESS.

Ukil, A., 2007. *Intelligent systems and applied signal processing in power engineering*. Berlin: Springer.

Webster, M. (2020) *Sensor | Definition of Sensor by Merriam-Webster*. Available at:  
<https://www.merriam-webster.com/dictionary/sensor> (Accessed: 11 November 2020).