

# IOT BASED WEARABLE DEVICE FOR PREDICTING A LONG COVID PATIENT'S CONDITION

ABDELRAHMAN TAMER ABDELGAWAD 1724925



DEPARTMENT OF MECHATRONICS  
KULLIAH OF ENGINEERING  
INTERNATION ISLAMIC UNIVERSITY MALAYSIA  
JUNE 2022

# IOT BASED WEARABLE DEVICE FOR PREDICTING A LONG COVID PATIENT'S CONDITION

ABDELRAHMAN TAMER ABDELGAWAD 1724925

AP DR SITI FAUZIAH BT. TOHA



A REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE  
REQUIREMENTS FOR A DEGREE OF BACHELOR OF  
ENGINEERING (B. ENG.) IN MECHATRONICS ENGINEERING

## **ABSTRACT**

This project aims to develop a wearable device that can be able to Predict the long covid-19 patients' conditions, to notify the doctors on a real-time basis. Long covid-19 patients suffer a lot during their daily activities especially if the lasting symptom is related to the respiratory system. By developing a system, that is easy and comfortable to wear during normal daily life, we believe that we will be able to predict the long covid-19 patients' condition. The system should first detect and analyze the patient's breathing pattern using artificial intelligence then store the patient's breathing pattern along with his status in an online database, then notify the doctors in case of a critical situation. To train the model the breathing pattern of current long covid patients and normal people was captured during doing daily activities such as walking, sitting, and climbing stairs. We hope that the developed system will help in easing the suffering of long covid patients by providing better monitoring of their health.

**Keywords:** Long Covid-19, Post-Covid-19 Syndrome, Breathing Classification, Deep neural network.

## **ACKNOWLEDGEMENTS**

I would like to thank Allah first for guiding me through the project and giving me the energy to overcome all the obstacles I faced. Secondly, I would like to thank everyone who has supported me in this project starting from my family members as they were giving me the support to continue finishing the project moving on to my supervisor who was always checking my progress and advising me, my friends who were helping me finding helpful resources, especially those who helped in spreading and participating in the experiment.

## TABLE OF CONTENTS

<b>ABSTRACT</b>	<b>I</b>
<b>ACKNOWLEDGEMENTS</b>	<b>II</b>
<b>LIST OF FIGURES</b>	<b>IV</b>
<b>LIST OF TABLES</b>	<b>V</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 BACKGROUND ON POST-COVID-19 SYNDROME	1
1.2 PROBLEM STATEMENT	2
1.3 MOTIVATION	2
1.4 OBJECTIVE	2
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>3</b>
2.1 OVERVIEW	3
2.2 EFFECT OF LONG COVID ON HUMAN ORGAN	3
2.3 RESPIRATION ANATOMY	5
2.4 ANALYZING THE DATA AND SIGNAL PROCESSING USING AI AND EMBEDDED SYSTEM	7
2.5 USING ARTIFICIAL INTELLIGENCE IN HEALTHCARE	8
2.6 DETECTION OF BREATHING PATTERNS USING WEARABLE SYSTEMS	8
2.7 INTEGRATION OF MOBILE APPLICATIONS WITH IoT	10
<b>CHAPTER 3 METHODOLOGY</b>	<b>12</b>
3.1 BREATHING DETECTION SYSTEM	12
3.2 SAVING THE DATA TO THE DATABASE	17
3.3 DATA GATHERING	18
3.4 CLASSIFICATION OF THE BREATHING SIGNAL	20
3.5 MOBILE APPLICATION	22
<b>CHAPTER 4 RESULTS AND DISCUSSION</b>	<b>23</b>
4.1 RESULTS OBTAINED FROM DATA GATHERING	23
4.2 DATA PREPROCESSING AND CREATION OF THE DATASET	24
4.3 MODEL TRAINING AND TESTING	25
4.4 MOBILE APPLICATION	25
<b>CHAPTER 5 CONCLUSION AND FUTURE WORK</b>	<b>29</b>
5.1 CONCLUSION	29
5.2 FUTURE WORK AND RECOMMENDATIONS	29
<b>REFERENCES</b>	<b>30</b>
<b>APPENDIX A THE CODE OF THE CAPTURING CIRCUIT</b>	<b>31</b>
<b>APPENDIX B CODE FOR DATASET CREATION AND MODEL TRAINING</b>	<b>37</b>

## LIST OF FIGURES

FIG. 2-1 LUNG'S UNDER COVID-19 PNEUMONIA	4
FIG. 2-2 RESPIRATION ANATOMY	6
FIG. 2-3 PLEURA	6
FIG. 2-4 RESPIRATION MECHANISM	6
FIG. 2-5 TYPES OF DATA ANALYSIS	7
FIG. 2-6 A) PRESSURE SENSOR FOR BREATHING DETECTION USING A BELT, B) PRESSURE SENSOR FOR BREATHING DETECTION USING A FACE MASK	9
FIG. 2-7 HUMIDITY SENSOR INTEGRATED WITH A FACE MASK	10
FIG. 2-8 IHEALTH SMART GLUCOMETER	11
FIG. 3-1 SYSTEM FLOW GRAPH	12
FIG. 3-2 RELATION BETWEEN THE APPLIED FORCE AND THE OUTPUT RESISTANCE	14
FIG. 3-3 (A) MAIN BODY, (B) COVER, (C) SENSING BASE, (D) DFA	16
FIG. 3-4 VOLTAGE DIVIDER CIRCUIT	16
FIG. 3-5 PCB LAYOUT	16
FIG. 3-6 EXAMPLE OF DATA STORED IN THE DATABASE	19
FIG. 3-7 PARTICIPANT RESTING DURING THE EXPERIMENT	19
FIG. 3-8 PARTICIPANT CLIMBING DOWN THE STAIRS DURING THE EXPERIMENT	19
FIG. 3-9 QR FOR THE DEMO VIDEO	19
FIG. 3-10 NEURAL NETWORK ARCHITECTURE	20
FIG. 4-1 QR CODE FOR THE GATHERED DATA	24
FIG. 4-2 TESTING AND TRAINING ACCURACY	25
FIG. 4-3 TESTING AND TRAINING LOSS	25
FIG. 4-4 (A) SIGN-IN PAGE, (B) REGISTRATION PAGE	26
FIG. 4-5 (A) DR'S HOMEPAGE, (B) PATIENT'S HOMEPAGE	27
FIG. 4-6 (A) PATIENT'S DETAILS ACCESSED BY THE DOCTOR (B) USER'S DETAILS PAGE	28

## **LIST OF TABLES**

TABLE 1 HELIX DNA DISCOVER PROJECT AND HEALTHY NEVADA PROJECT SURVEY	1
TABLE 2 QUESTIONS FOR PATIENTS' CLASSIFICATION	18
TABLE 3 MODEL SUMMARY	21
TABLE 4 SUMMARY OF THE PARTICIPANTS' ANSWERS	23

## CHAPTER 1 INTRODUCTION

### 1.1 BACKGROUND ON POST-COVID-19 SYNDROME

Post-Covid-19 Syndrome or what is known as Long Covid can be defined as the chronic form of severe acute respiratory syndrome coronavirus (SARS-COV) where the patient keeps suffering from some of the SARS-COV symptoms after the recovery and is tested as negative. According to Dr. Janet Diaz whom the team lead at the World Health Organization (WHO), long Covid does not have a clear recovery duration as it might last for three months or even up to nine months. (Diaz, 2021)

According to an online survey that was conducted by Helix Dna Discover Project and Healthy Nevada Project (April~October 2020), out of 21359 participants As shown in Table 1, 13838 participants suffered from one or more symptoms after the recovery. Another survey was conducted in Clichy, France, by contacting 120 patients by phone and it resulted in: a mean of 110.9 ( $\pm 11.1$ ) days

*Table 1 HELIX DNA DISCOVER PROJECT and Healthy NEVADA PROJECT Survey*

Sample size	21,359
Median age (range)	56 (18-89+)
N female (%)	14,407 (64.1%)
Ancestry N (%)	
African	462 (2.1%)
East Asian	368 (1.6%)
European	18,884 (83.8%)
Latinx	2063 (9.2%)
South Asian	144 (0.6%)
Other / mixed ancestry	613 (2.7%)
N with COVID-19 test (%)	5,854 (23.4%)
Positive (%)	357 (6.1%)
Negative (%)	5497 (93.9%)
N reporting $\geq 1$ symptom (%)	13,838 (55.4%)
$\geq 1$ symptom lasting longer than 30 days (%)	1308 (10.0%)
$\geq 1$ symptom lasting longer than 60 days (%)	1220 (7.3%)
$\geq 1$ symptom lasting longer than 90 days (%)	1128 (5.7%)



following admission. The most frequently reported persistent symptoms were fatigue (55%), dyspnea (42%), loss of memory (34%), concentration (28%), and sleep disorders (30.8%). (Cirulli et al., 2020; Yong, 2021)

## **1.2 PROBLEM STATEMENT**

Long COVID makes the patient suffers a lot during his daily activities, especially if the lasting symptom is related to the respiratory system. Long COVID requires early detection in order to provide the necessary treatment before partially damaging the organ, that is why a system is needed to predict long COVID.

## **1.3 MOTIVATION**

As a person who used to suffer from asthma, I would like to help those who are suffering more due to the Long Covid as they suffer from the same feelings I used to suffer from whenever they do any small exercise even if it is as small as climbing a couple of stairs before Covid partially damages the Lungs. Another thing is that it will help in providing better health for the community.

## **1.4 OBJECTIVE**

- 1) To investigate parameters related to Long Covid.
- 2) To optimize the patient's status based on AI prediction the data will be stored online.
- 3) To develop an assistive wearable device that can detect the condition of Long Covid patients.
- 4) To integrate the software with the wearable device and provide an immediate Alert to the authorized officer in case of any critical situation.

## **CHAPTER 2 LITERATURE REVIEW**

### **2.1 OVERVIEW**

This chapter aims to help in understanding the project and gathering ideas about the possible methodologies that can be implemented by collecting the necessary data, analyzing similar projects, and determining their advantages and disadvantages.

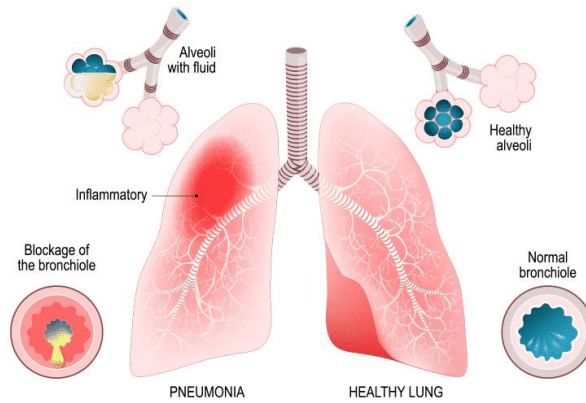
### **2.2 EFFECT OF LONG COVID ON HUMAN ORGAN**

Most Covid-19 patients do not fully recover from covid-19 overnight, especially if the patient has suffered from a serious case of Covid-19. In some cases, Serious Covid-19 Cases may lead to sepsis. Sepsis occurs as a result of the spread of inflammation through the blood vessels causing Tissue damage, that will lead to damage to organs one after another. According to Galiatsatos, “Lungs, heart and other body systems work together like instruments in an orchestra in sepsis, the cooperation between the organs falls apart. Entire organ systems can start to shut down, one after another, including the lungs and heart.” (Galiatsatos, n.d.)

#### **2.2.1 Effect of Long Covid on Lungs**

##### **A. Covid-19 Pneumonia**

Pneumonia happens when the lungs get filled with fluid leaking from the tiny blood vessels in the lungs (Fig. 2-1) that lead to problems in breathing like dyspnea (shortness of breath) and breathing difficulties, which can be severe enough that the patient will require oxygen or ventilation. Covid-19 Pneumonia happens during mild to severe Covid-19 and it tends to fill the lungs’ air sac with fluid. Covid-19 Pneumonia may lead to lasting lung damage as it is more severe than normal pneumonia.



*Fig. 2-1 Lung's under Covid-19 Pneumonia*

## B. Acute Respiratory Distress Syndrome (ARDS)

Acute respiratory distress syndrome is a form of lung failure where the lungs become unable to provide the patient's vital organs with sufficient oxygen. ARDS usually happens due to a complication of a serious existing health condition such as severe pneumonia as more of the air sacs become filled with fluid. ARDS is a life-threatening condition where 1 out of 3 patients may die.

### 2.2.2 Factors Affecting The Lungs' Health After Covid-19

There are 3 main factors behind the lungs' health according to (Galiatsatos: "The first is the severity of the coronavirus infection itself, whether the person has a mild case or a severe one. The second is whether there are existing health problems, such as chronic obstructive pulmonary disease (COPD) or heart disease that can raise the risk for severe disease, Treatment is the third factor, a patient's recovery and long-term lung health are going to depend on what kind of care they get, and how quickly." As stated by Galiatsatos, long covid patients should get treated quickly to ensure better lung health. That is why the system aims to monitor the patient's status to facilitate the patient's check-up and treatment.

### **2.2.3 Effect of Long Covid on The Other Body Organs**

Although covid-19 is a respiratory virus, the long covid patient may suffer abnormality and tissue damage in other organs. As an example, according to the research done by (Lu et al., 2020), it was reported after three months of discharging, that some of the covid-19 survivors face abnormalities in brain structure and metabolic rate, with some neurological symptoms such as memory loss, anosmia, and fatigue, although the severity of covid-19 infection was moderate. However, in the case of severe covid-19 infection, Long covid can lead to failure of the infected organ.

## **2.3 RESPIRATION ANATOMY**

### **2.3.1 Respiratory System**

The respiratory system (Fig. 2-2) starts with the nose and consists of a nasal cavity and pharynx that act as an air passage from and into the nose. After the air exits the nose, it enters the airway, which is pipe-like, which carries the air to the lungs. The airway consists of the larynx, which is the voice box, and the trachea which bifurcates into two bronchial tubes that are subdivided into a huge number of bronchioles that end up with alveoli where the gas exchange happens with the lungs. The lungs are a pair of pinkish-gray spongy texture organs. The two lungs are slightly different in size as the heart takes space on the left side that is why the left lung is divided into two sections only while the right side is divided into three sections. Both lungs are surrounded by a two-layered membrane called pleura (Fig. 2-3), between the two layers, there is a slippery liquid called pleural fluid that acts as a lubricant to reach the optimal expansion and contraction during breathing with the least friction.(Daiana Da Costa et al., 2019; *How the Lungs Work* / NHLBI, NIH, 2020).

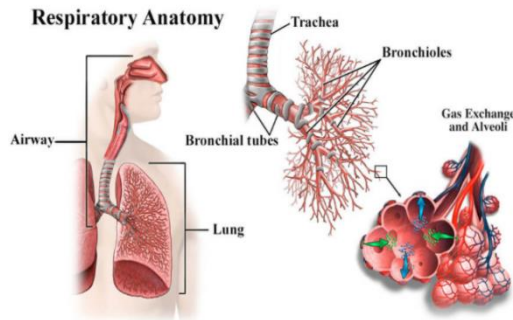


Fig. 2-2 Respiration Anatomy

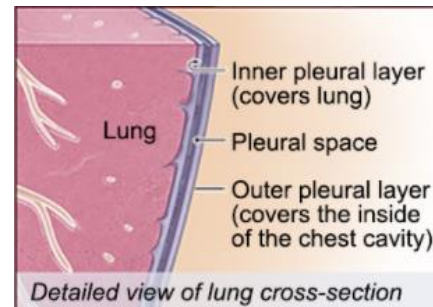
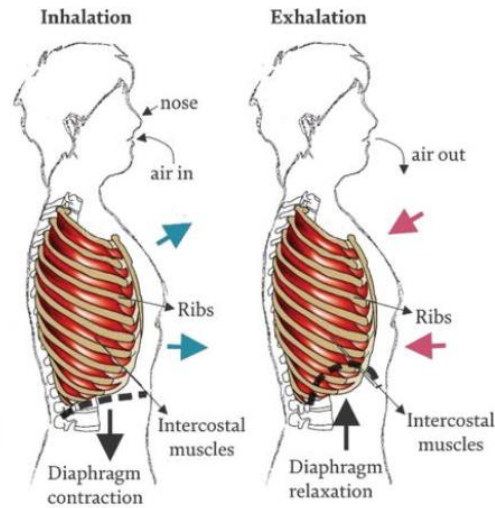


Fig. 2-3 Pleura

### 2.3.2 Respiration Mechanism

The main organ in the respiratory system is the lungs. The air starts its journey in the human body by entering either from the nose or mouth, where it is moistened and warmed as the cold and dry air can irritate your lungs. Then the air passes through the airway until it reaches the lungs through the alveoli. As the air enters the lungs, it starts expanding to increase the amount of air entering the lungs and also to increase the surface area for gas exchange with the blood vessels found inside the lungs. The lungs then contract to push the air filled with the carbon dioxide back to the airway that carries it to its final exit either the nose or mouth(*How the Lungs Work* / NHLBI, NIH, 2020).

The contraction and expansion of the lungs are done by the mean of the diaphragm (Fig. 2-4) when the diaphragm contracts, it increases the abdominal pressure as the lower ribcage expands causing expansion of the Lungs. While the diaphragm is relaxing, the lower ribcage goes back to its normal position forcing the lungs to contract and push the air out of it.(Daiana Da Costa et al., 2019)



*Fig. 2-4 Respiration Mechanism*

## 2.4 ANALYZING THE DATA AND SIGNAL PROCESSING USING AI AND EMBEDDED SYSTEM

Signal Processing is a branch of electrical engineering that models and analyzes data representations of physical events. To have an accurate data analysis, deep learning technology must be introduced as requires preprocessing, transformation, and feature extraction.

The workflow of data analysis and signal processing can be summarized in Fig. 2-5, where it starts with data gathering and labeling in order to develop a dataset that can be later accessed for data comparison. Data gathering can either be obtained from real-world data but that will be unreasonable as it will take a lot of time, that is why we tend to simulate the data based on the real-world situation to enlarge our dataset to improve the accuracy and increase the robustness of the trained system. After the data is stored, it should be visualized in multidomain as



*Fig. 2-5 Types of Data Analysis*

time and frequency domains, where the type of preprocessing can be determined, and features can be extracted easily. In order to do so, some toolboxes can do the feature extraction like Signal Processing Toolbox and Wavelet Toolbox. As a result of Deep learning of signal processing, the system should predate the type of signal and label it accordingly. As the system keeps predicting the signals, the data processing starts to be faster and more accurate.

## **2.5 USING ARTIFICIAL INTELLIGENCE IN HEALTHCARE**

Artificial intelligence spread widely in the medical fields, and that is the result of being able to reduce the cost and pressure on doctors, especially with the rapid growth of the unhealthy population. That is can be done as AI System has the capability of predicting the patients' condition by analyzing the change in their conditions and suggesting treatment options (Shah, n.d.).

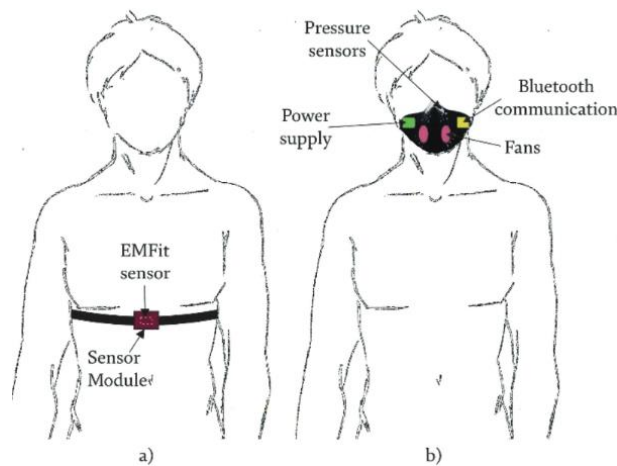
One of the recent applications of AI systems is the accurate classification of Covid-19 with chest x-ray digital images using deep convolutional neural networks, where a model was developed to classify digital x-ray images to be either normal, covid-19, or pneumonia. With such a model, it can prevent the possibility of the wrong classification due to human error, which will help the patient to have the best possible treatment(Shastri et al., 2022).

## **2.6 DETECTION OF BREATHING PATTERNS USING WEARABLE SYSTEMS**

By referring to the breathing mechanism mentioned in section 2.2.2, various types of sensors can be used for the seek breathing detection. Here are some of the sensors that are used in the detection of breathing.(Daiana Da Costa et al., 2019)

### 2.6.1 Pressure Sensor

There are various ways to use the pressure sensor in the detection of breathing patterns. The pressure sensor can be integrated with a wearable belt, and it can measure the exerted pressure on the belt. As an example, researchers in Emit (Fig. 2.6.a) have developed a belt and connected an electromechanical film to the belt where it can measure the exerted pressure done by the chest during expansion and produces a voltage change directly proportional to the exerted pressure. Another way to use the pressure sensor is to integrate it with a face mask (Fig. 2-6.b), where it can measure the change of pressure inside the mask due to the air flowing from and into the respiratory system.



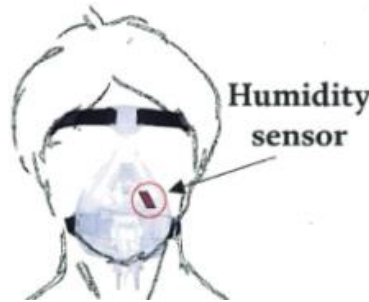
*Fig. 2-6 a) Pressure Sensor for Breathing Detection using a belt, b) Pressure Sensor for Breathing Detection using a face mask*

### 2.6.2 Humidity Sensor

By analyzing the humidity change due to the air coming out of the nose and mouth, the breathing pattern can be obtained. Researchers have developed a humidity sensor (Fig. 2-7) based on a chemical that is capable of moisture detection, called the porous graphene network, that was able to detect various



breathing patterns. The disadvantage of the sensors integrated with face masks is the uncomfortably while wearing them for a long time.



*Fig. 2-7 Humidity Sensor  
Integrated with a Face Mask*

### **2.6.3 Oximeter Sensor**

An oximeter sensor is an IR sensor, where it illuminates a small portion of the skin and measures the light absorbed by the skin depending on the oxygenated and deoxygenated blood levels. These sensors have been widely commercialized as they can be worn on different parts of the body.

### **2.6.4 Resistive Sensor**

It is a textile sensor, that works on the same principle as the pressure sensor, and consists of resistive stretch sensors that are made with a conductive material and a polymer mixture. This sensor can be integrated with the cloth to design smart textile fabrics.

All the above sensors can be used to get an accurate breathing pattern depending on the position and the expected wearing time of the system.

## **2.7 Integration of Mobile Applications with IoT**

The internet of things (IoT) started in 1993 when John Romkey and Simon Hackett developed a toaster that can be controlled to turn it on and off through the

internet. IoT allows the devices to collect and send data wirelessly, with the help of embedded sensors, processors, and communication hardware, as long as it is connected to the internet.

The concept of IoT and mobile applications have been introduced in healthcare field as they can monitor the patients' status and alert for any critical changes allowing faster response and better treatment. As an example, iHealth Smart Glucometer (Fig. 2-8) is a smart mobile application based on IoT that can monitor the blood sugar levels throughout the day and set a reminder for the medications.

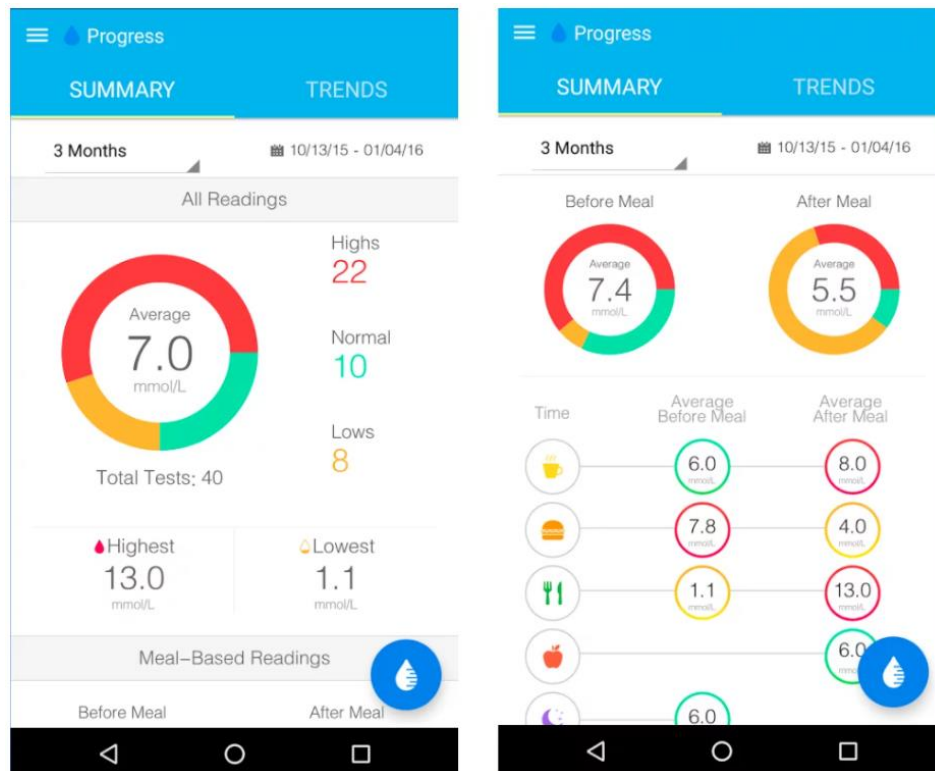


Fig. 2-8 iHealth Smart Glucometer

Mobile applications are the frontier of IoT as it acts as mediators between human IoT servers. Another reason behind integrating smartphones with a lot of useful sensors is always being in our hands which makes them the best gate for IoT.

## CHAPTER 3 METHODOLOGY

The project's methodology can be divided into five main parts, development of the breathing detection system, saving all the data online, data gathering, AI model training and evaluation, and finally development of a mobile application that can be integrated with the AI model. The system flow graph can be shown in Fig. 3-1.

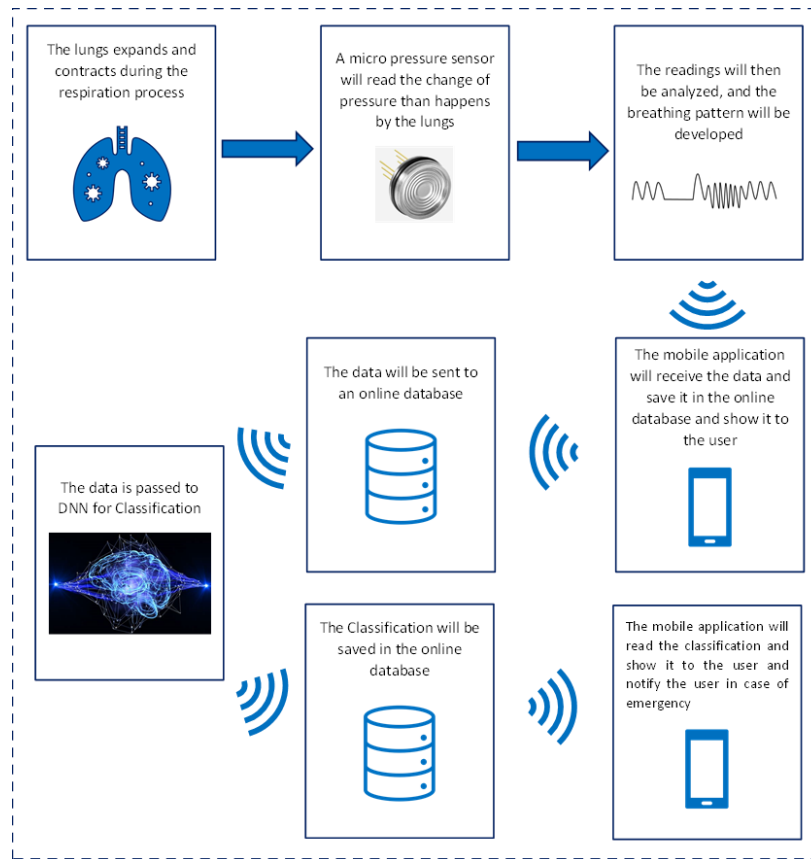


Fig. 3-1 System flow graph

### 3.1 BREATHING DETECTION SYSTEM

By looking at the mechanism of respiration, the breathing pattern can be captured either by attaching the system to the patient's body or using contactless methods. Contactless detection can be done by the usage of various sensors such as a doppler radar that can be put in front of the patient and keeps measuring the

change in distance due to inhalation and exhalation. (Giovannini et al., 2021). However, contactless detection can be inaccurate and also cannot be used for continuous detection as the patient need to stand in front of the contactless device to detect his breathing pattern, so contact devices are more preferred for continuous detection.

By taking advantage of the contraction and expansion of the lungs, a pressure sensor can be used to capture the change in pressure. The pressure sensor should be integrated with a micro-controller that is needed to process the breathing signal after being captured.

### **3.1.1 Hardware and Components**

As was discussed earlier, the main components of the system are a pressure sensor, and a microcontroller, however, there are not the only components, as the system still needs a power supply and hard case to cover the whole system.

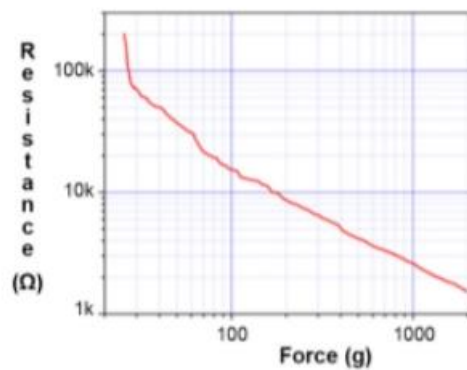
#### **A. Pressure Sensor**

To have accurate results, the pressure should have as low sensitivity as possible as the pressure exerted by the lungs is not large. And also the pressure sensor should be small in size and light in weight as the whole system should be light in weight to provide more comfort to the patient while using it.

By referring to the data sheet of the force-sensing resistor (FSR) (*Force Sensing Resistors* ®, n.d.) we can see that it is the best sensor that can meet our requirements.

Specifications:

- Size: FSRs come in different sizes starting from 5mm x 20 mm up to 13mm x 56mm.
- Sensitivity: The FSR can read from 0.2N up to 20N
- Output: by referring to Fig. 3-2, the output resistance of the sensor is inversely proportional to the applied force.



*Fig. 3-2 Relation between the applied force and the output resistance*

#### B. Signal processing using a microcontroller

As the system needs to be small in size and at the same time provide a wireless communication protocol to reduce the total number of components, ESP32 will be the best microcontroller to be used as its chip supports both Bluetooth and Wi-Fi connection, which will ease the communication between the device and the database, and it is relatively small in size as its board is 5.8 cm x 2.8 cm. The ESP32 Board will be used to read the value of FSR and to process the signal, where the system will have an automatic calibration by using the average of the previous readings as the zero axis for the next readings and transferring it to the database. Moreover, ESP32 supports FreeRTOS and dual Core which will help a lot in having better efficiency and accuracy.

The sampling rate of the signal will be 10 Hz. By referring to Nyquist's theory, the sampling rate should be at least twice the highest expected frequency. According to (Lee et al., 2014), the maximum number of breaths per min is 42 breaths which means the signal frequency is 0.7 Hz. Using 10 Hz as the sampling frequency will make sure that aliasing will not happen.

#### C. Power Source

By referring to the ESP32 datasheet, we have decided to power it up using a small long-lasting 3V battery and connected it to the 3.3V pin. After checking the available 3V batteries, we found the best battery to be used is CR123. CR123 is a small battery of the dimensions 16.5mm in diameter x 34mm in height. It also has a capacity of 1.6 Ah which will provide a long working time.

#### D. Hard Case

In order to place all of these components together, we designed a small case without any sharp edge to provide a comfortable wearing and great user experience. First, we started by designing the case using SolidWorks, the design consists of three parts, the main body, the cover, and the sensing base as shown in Fig. 3-3 below. Fig. 3-3 shows the isometric view of each part and the design of assembly (DFA). The assembled case is 55.6 mm x 71.6mm x 25.6mm for length, width, and height respectively.

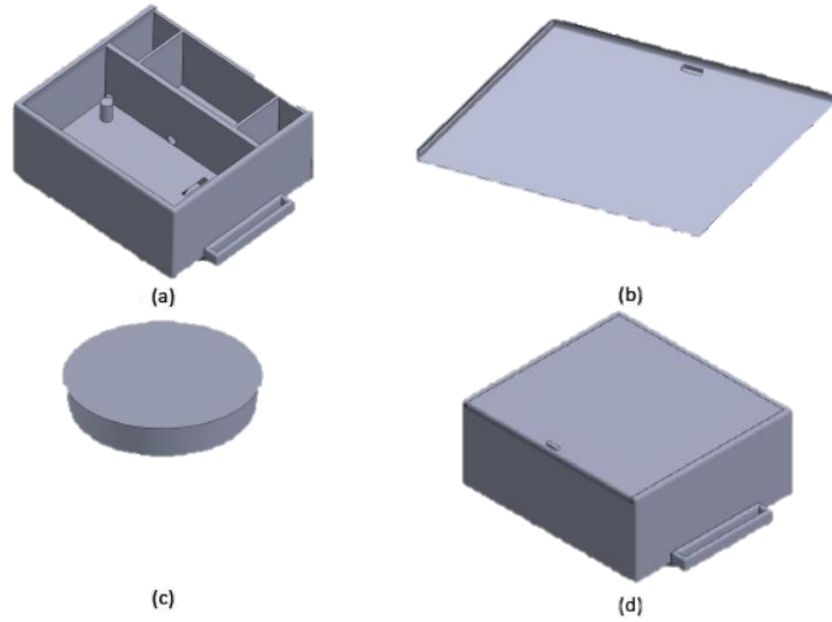


Fig. 3-3 (a) Main Body, (b) Cover, (c) Sensing base, (d) DFA

### 3.1.2 System Circuit

By referring to Fig. 3-2, we can see that the relation between the applied force and the resistive value of the FSR is inversely proportional. To inverse this relation a voltage divider circuit was designed as shown in Fig. 3-4. For a better and long-lasting connection, we fabricated a PCB that has the layout shown in Fig. 3-5.

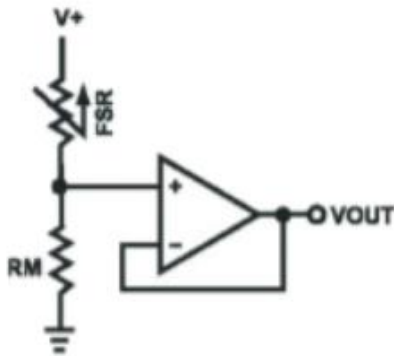


Fig. 3-4 Voltage divider circuit

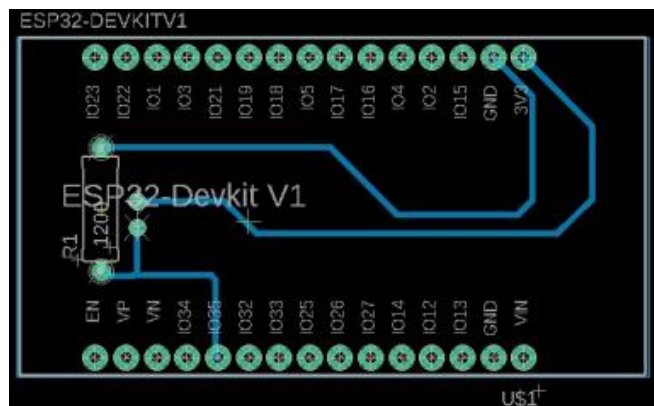


Fig. 3-5 PCB layout

### **3.1.3 System Software**

The microcontroller was programmed using freeRTOS, which helped in making sure that the reading will happen exactly on time, providing more accurate data in terms of time intervals. Also, since ESP32 is a double-cored microcontroller, one core was allocated just for the detection of the breathing pattern and the other one was allocated just for sending the data in order to prevent any interrupt or delay because of the other task. The system was designed to store the data of the previous 30 minutes in case it got disconnected to help in tracking the patient even while being away from his phone or being disconnected from Wi-Fi. The system also provides an auto-calibration function that set the zero-axis to the average of the reading of the past 10 seconds.

## **3.2 SAVING THE DATA TO THE DATABASE**

The system will be using an online database to make it possible for the user to access it from different devices as long as the user is connected to the internet and also it be easier for the doctor to access it. Firebase is a free online database developed by Google and it can be integrated with various types of applications and software. The system will be using firebase to save captured data and also to save the type of the breathing pattern after passing the captured reading to the developed artificial intelligence model. One of the interesting features of firebase is that it supports adding a customized artificial intelligence model to the system, as it supports importing a pre-trained TensorFlow model.



### 3.3 DATA GATHERING

Due to the lack of data related to the breathing pattern of Long covid patients, we have decided to collect the data ourselves.

#### 3.3.1 Experiment Preparation

A google form was spread having the questions stated in Table 2 with the expected response. The questionnaires in the form were designed to distinguish between normal participants and long covid participants and divide them accordingly.

*Table 2 Questions for patients' classification*

Question	Response
Timestamp	
Name	name
Email	email
Phone number (with WhatsApp)	phone
Age	age
Were infected with Covid-19 Virus before?	Y/N
IF yes place state when (if still infected please say "NOW")	date
After being recovering from Covid-19 (tested as NEGATIVE) did u still suffer any of the below symptoms ? (Tiredness, Shortness of breath, Tightness, Continuous Headache, Fast or pounding heartbeat, Depression or anxiety)	checking the suffering symptoms
Do you agree on participating in Data gathering and testing process?	Y/N

The experiment's strategy was set based on covering the normal daily activities. During the experiment, the participant was asked to rest for 1 minute, then walk for 150m and go down 32 stairs followed by walking 200m and climbing up 32 stairs then walk for 50m and finally rest for 1 minute.

#### 3.3.2 Experiment Setup

The average time of the experiment was 6 minutes. During the experiment, the data were sent directly to the database in batches, where every batch represented the 10 captured readings during a particular second. The data were saved according to the format shown in Fig. 3-6. After the experiment, feedback

was collected from the participants, about the comfortability of the system and they all will not mind wearing it for a continuous time. some of the experiment's procedures can be seen in Fig. 3-7 and Fig. 3-8. A demo video for the experiment can be checked either by Clicking on the following link <https://bit.ly/3xy970M> or by scanning the QR Code in Fig. 3-9.

```
"Patients": {
  "100": {
    "2022": { //Year
      "5": { //Month
        "10": { //Day
          "14": { //Hour
            "5": { //Minutes
              "6": [ //Sec
                50,
                60,
                70,
                80,
                70,
                60,
                50,
                40,
                30,
                20
              ],
              "7": [
                10,
                0,
                -10,
                -20,
                -30,
                -40,
                -50,
                -60,
                -70,
                -80
              ]
            }
          }
        }
      }
    },
    "102": [ ],
    "103": [ ],
    "104": [ ]
  }
}
```

Fig. 3-6 Example of data stored in the database



Fig. 3-7 Particepent resting during the experiment



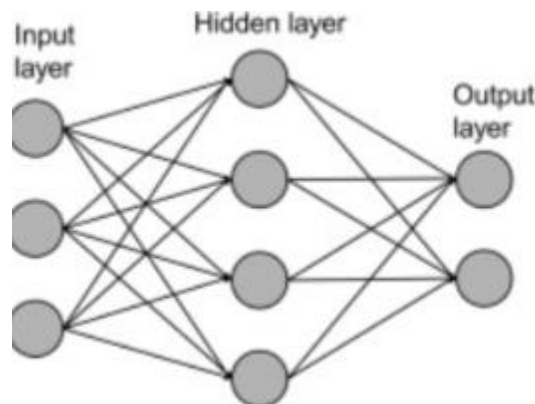
Fig. 3-8 Particepent climbing down the stairs during the experiment



Fig. 3-9 QR for the demo video

### 3.4 Classification of the breathing signal

One of the most effective and popular ways for data classification is a convolution neural network. A convolution neural network is an artificial intelligence aspect that focuses on emulating the learning approach that humans use to gain certain types of knowledge. By referring to Fig. 3-10, we can see that Neural networks mainly of an input layer, which represents the data that it will be passed to the neural network, hidden layers, which can consist of many different layers where each layer is considered as an input of the previous layer, and a final output layer, which represents the final output of the neural network. (Das & Roy, 2019).



*Fig. 3-10 Neural network Architecture*

The trained model is developed to have an input of 300 nodes, which should correspond to the values captured every 0.1 seconds for a consecutive 30 seconds. The values are then multiplied with the correct weight and then passed to a set of hidden layers. The output of these layers is then passed as an input for the output layer which is another dense layer that has two neurons where each represents the probability of being under the corresponding class. Table 3 shows the summary of the developed model with the type of each layer and its output

shape. During the model training, different parameters with different combinations of layers were tried to reach the best results.

*Table 3 Model Summary*

Layer (type)	Output Shape	Param #	Connected to
input (InputLayer)	[(None, 300, 1)]	0	[]
conv1d_43 (Conv1D)	(None, 300, 16)	64	['input[0][0]']
activation_28 (Activation)	(None, 300, 16)	0	['conv1d_43[0][0]']
conv1d_44 (Conv1D)	(None, 300, 16)	784	['activation_28[0][0]']
conv1d_42 (Conv1D)	(None, 300, 16)	32	['input[0][0]']
add_14 (Add)	(None, 300, 16)	0	['conv1d_44[0][0]', 'conv1d_42[0][0]']
activation_29 (Activation)	(None, 300, 16)	0	['add_14[0][0]']
max_pooling1d_14 (MaxPooling1D)	(None, 150, 16)	0	['activation_29[0][0]']
conv1d_46 (Conv1D)	(None, 150, 32)	1568	['max_pooling1d_14[0][0]']
activation_30 (Activation)	(None, 150, 32)	0	['conv1d_46[0][0]']
conv1d_47 (Conv1D)	(None, 150, 32)	3104	['activation_30[0][0]']
conv1d_45 (Conv1D)	(None, 150, 32)	544	['max_pooling1d_14[0][0]']
add_15 (Add)	(None, 150, 32)	0	['conv1d_47[0][0]', 'conv1d_45[0][0]']
activation_31 (Activation)	(None, 150, 32)	0	['add_15[0][0]']
max_pooling1d_15 (MaxPooling1D)	(None, 75, 32)	0	['activation_31[0][0]']
average_pooling1d_7 (AveragePooling1D)	(None, 74, 32)	0	['max_pooling1d_15[0][0]']
flatten_7 (Flatten)	(None, 2368)	0	['average_pooling1d_7[0][0]']
dense_14 (Dense)	(None, 64)	151616	['flatten_7[0][0]']
dense_15 (Dense)	(None, 32)	2080	['dense_14[0][0]']
tf.math.greater_3 (TFOpLambda)	(None, 32)	0	['dense_15[0][0]']
tf.cast_3 (TFOpLambda)	(None, 32)	0	['tf.math.greater_3[0][0]']
tf.math.multiply_3 (TFOpLambda)	(None, 32)	0	['dense_15[0][0]', 'tf.cast_3[0][0]']
output (Dense)	(None, 2)	66	['tf.math.multiply_3[0][0]']
=====			
Total params: 159,858			
Trainable params: 159,858			
Non-trainable params: 0			

### **3.5 Mobile application**

As was discussed, a mobile application needs to be developed as it will be the gateway between the system and database, and also it is considered as the monitor where the patient and the doctor can check the breathing pattern history and its classification.

The mobile application should be user-friendly and easy to understand the displayed information to be suitable for any age to use. The mobile application has two types of users doctors and patients, however, different permissions are given to each with a slight difference in the layout. As an example, the doctor can add patients to his monitoring list. Another feature that is included in the mobile application, Alerting the patient and the doctor of any critical situation like the stop of breathing or having dysrhythmic breathing for a long time. One more feature is providing an emergency communication between the patient and the doctor that can be initialized by one of them.

The mobile application was developed using Java, to enable a smooth connection between the firebase database and the mobile application. Integrating the mobile application with firebase allows reading and writing to the database, using the AI model API uploaded to the database, using a fully secure authentication system including third-party sign-in using a Facebook account or a Google account and password reset system. The mobile application has three main pages the sign-in and registration page, the user/patient page, and the home page.

## CHAPTER 4 Results and Discussion

This chapter aims to discuss the results obtained from the data gathering, data preprocessing and creation of the dataset, training and testing of the model, and finally the mobile application interfaces.

### 4.1 Results obtained from Data gathering

As a result of the spread questionnaires, 22 people responded to the form from 10 different nationalities. However, only 8 people showed up for the experiment. The actual participants had an age mean of 23 with different ranges of weight and height. Half of the participants were identified according to normal people and the other half as long covid patients. Table 4 shows the summary of the participants' answers.

*Table 4 Summary of the participants' answers*

Sample size	8
Age mean	23( $\pm$ 2) years
Height mean	174 cm
Weight mean	90 kg
Male : Female	7 : 1
Nationalities:	
Egyptian	2
Malaysian	2
Ethiopian	1
MOROCCAN	1
Palestinian	1
Indian	1

## 4.2 Data preprocessing and creation of the dataset

After the data was read from the database, the program starts checking if there is any discontinuity in the breathing pattern and separates them into different patterns to make sure to have accurate data. Then it starts removing any pattern that consists of less than 300 readings as the model was developed to have an input of 300 readings at each prediction. The data of each patient was then mapped between -100 and 100 to remove any effect that happen due to the difference in the tightness of the system on the patient's body during the experiment. As a final step, a sequence of 300 readings was generated from all the data with a corresponding list having the expected type by referring to the predetermined type for each patient. As a result of that, we had a dataset of 16861 sequences for both types. To provide a variety of breathing types in each batch of training, the dataset was shuffled twice before starting the training. During the training process, the dataset was split into 70% for training and 30% for validation, to have sufficient data for testing without affecting the training data. The whole dataset can be checked either by Clicking on the following link <https://bit.ly/3xuBn4b> or by scanning the QR Code in Fig. 4-1.



*Fig. 4-1 QR code for the gathered data*

### 4.3 Model training and testing

The model was set to be trained for 50 epochs using “Adam optimizer” as the optimization algorithm, which is considered as a combination of ideas from other optimizers and using the loss calculation’s technique as “categorical\_crossentropy”, as the model is considered as a multiclass classification model. As a preliminary test, the testing dataset was used for validation and showed an accuracy of 98.99 % and a loss of 0.09 as shown in Fig. 4-2, and Fig. 4-3 respectively.

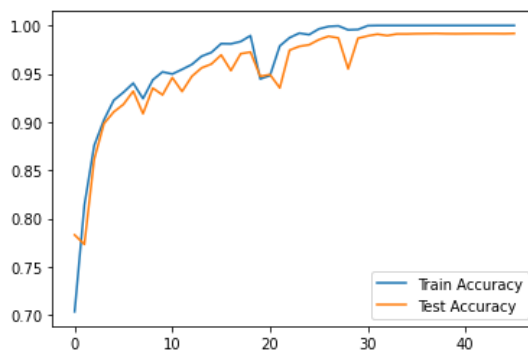


Fig. 4-2 Testing and Training Accuracy

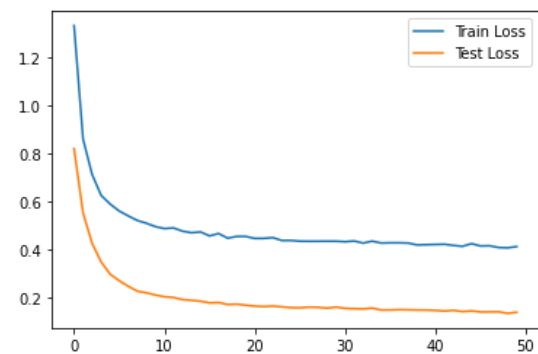


Fig. 4-3 Testing and Training Loss

### 4.4 Mobile Application


The mobile application was successfully connected to the firebase server and used the firebase server to read the breathing pattern along with the doctor and the patient’s details and also for the sign-in and registration process. As discussed earlier, the application consists of three main pages:

#### 4.4.1 Sign-In and registration

Due to the connection between the Firebase server and the mobile application, the sign-in and registration system includes many features. The system email and password and be saved in google password that can be accessed easily from other devices. Moreover, the user can reset his password easily by sending a



reset link to his registered email address. If the user is new to the system, the registration page is user-friendly as it supports autofill for each field to save the user's effort and time. After the user, finish filling in all the details, the application checks the entered data formats and stores each user's data under a unique ID given by the firebase authentication system. A sample of the sign-in and registration pages can be found in Fig. 4-4.



**Email**


Email

**Password**

Password

REGISTER SIGN IN

**Details**



Name name

Birth Day Birth Day

Phone Number Phone Number

Type ☐ Doctor ☐ Patient

Email Email

Password Password

Re-Password Re-enter Password

**In Case Of Emergency**

Name Name

Relation Relation

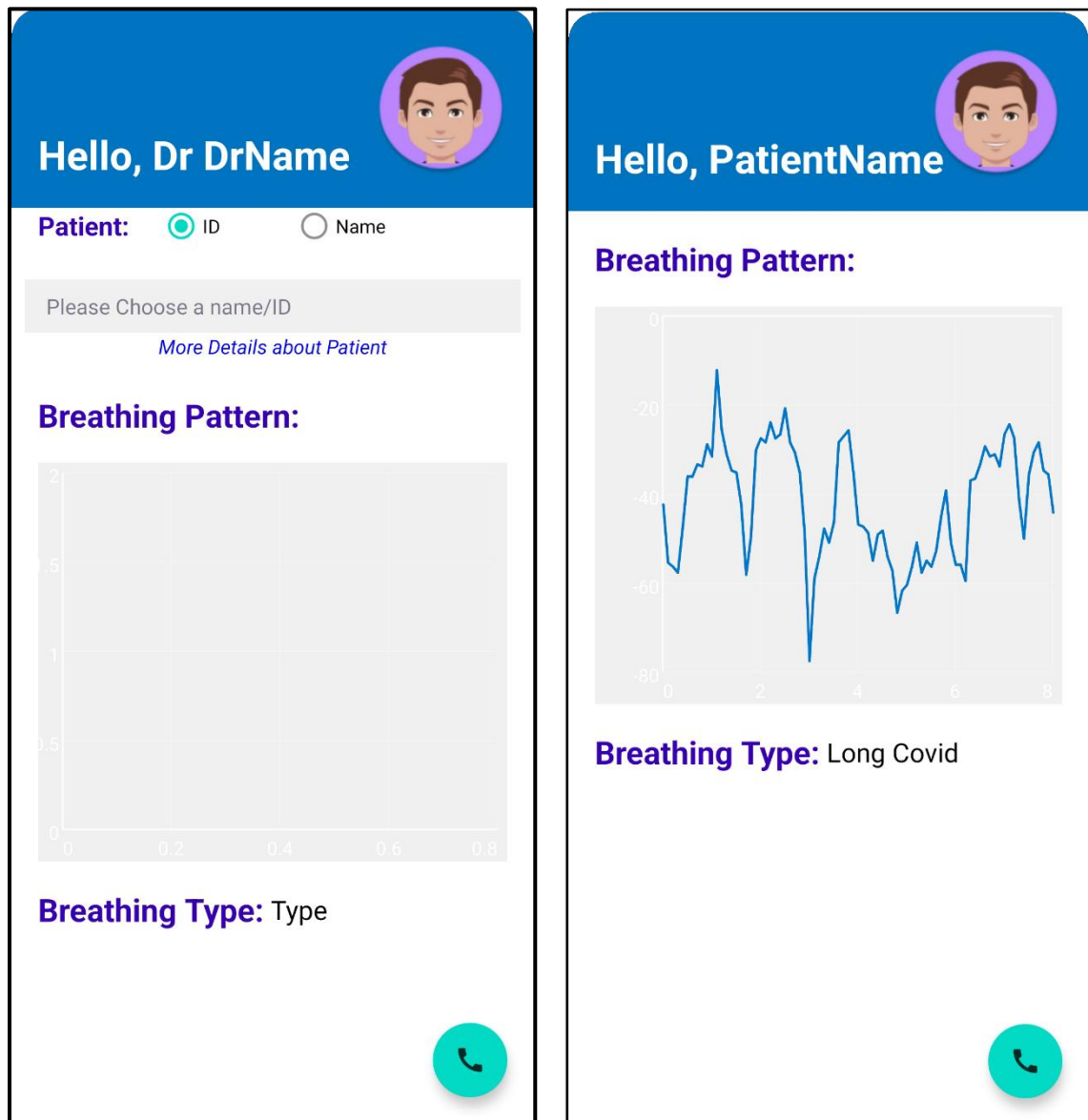
Phone Phone

REGISTER

Fig. 4-4 (a) Sign-In page, (b) Registration page

#### 4.4.2 Homepage

On the home page, the patient will be able to check his current breathing pattern along with his type of breathing. There will be a floating button to initiate a call between the patient and the doctor. The doctor's homepage will have a down drop spinner to be able to navigate between his patients' patterns. A sample of the homepage for both patients and doctors in Fig. 4-5.



(a)

(b)

Fig. 4-5 (a) Dr's homepage, (b) Patient's homepage

### 4.4.3 User's Details

The user's detail page has the same layout for both doctors and patient showing their details such as name, birthdate, phone, type, email address, and emergency contact. The user who has registered as a doctor will have an extra page where he can view his patients' details. Both pages can be found in Fig. 4-6.

The figure displays two mobile application screens, (a) and (b), which are used for viewing user details. Both screens feature a blue sidebar on the left with two main sections: 'Details' and 'In Case Of Emergency'. The 'Details' section lists fields for Patient, Name, Birth Day, Phone Number, and Email. The 'In Case Of Emergency' section lists fields for Name, Relation, and Phone. A 'Welcome Back' notification bubble is visible on screen (a). The main content area on the right of each screen contains input fields for these details. Screen (a) is titled 'CALL PATIENT' at the bottom, while screen (b) is titled 'CALL DOCTOR'. Both screens include a user profile picture at the top right of the main area.

Section	Field	Input Type
Details	Patient	Radio buttons (ID selected, Name unselected)
	Name	Text input
	Birth Day	Text input
	Phone Number	Text input
	Email	Text input
In Case Of Emergency	Name	Text input
	Relation	Text input
	Phone	Text input

(a)

(b)

Fig. 4-6 (a) patient's details accessed by the Doctor (b) user's details page

## **CHAPTER 5 Conclusion and Future work**

### **5.1 Conclusion**

In conclusion, the system is very important as it helps doctors to monitor their patients anytime without being in danger of contacting them. At the end of this project, the project's objectives were satisfied as the patient's status can be monitored and predicted using deep convolution neural network using a wearable device. However, the system currently has some limitations due to the data being only obtained from four people of each type. Another limitation, the system needs a Wi-Fi connection to operate, which might not be convenient for all the users, and the doctors will not be able to continuously monitor their patients.

### **5.2 Future work and recommendations**

Due to these limitations, more work needs to be done on the project as:

1. The mobile application and the system should be modified to enable Bluetooth communication between both.
2. Allowing the user to config his preferred way of communication between the database and the system either Bluetooth or a specific WiFi network using an easy configuration page.
3. More data should be gathered either by conducting experiments or from medical departments, as the more data gathered the more reliable the system can be.

## REFERENCES

- Cirulli, E. T., Schiabor Barrett, K. M., Riffle, S., Bolze, A., Neveux, I., Dabe, S., Grzymiski, J. J., Lu, J. T., & Washington, N. L. (2020). Long-term COVID-19 symptoms in a large unselected population. *MedRxiv*, 2020.10.07.20208702. <https://doi.org/10.1101/2020.10.07.20208702>
- Daiana Da Costa, T., de Fatima, M., Vara, F., Santos Cristino, C., Zanella, Z., Nunes, G., Neto, N., & Nohama, P. (2019). Breathing Monitoring and Pattern Recognition with Wearable Sensors. *Wearable Devices - the Big Wave of Innovation*. <https://doi.org/10.5772/INTECHOPEN.85460>
- Das, H. S., & Roy, P. (2019). A Deep Dive Into Deep Learning Techniques for Solving Spoken Language Identification Problems. *Intelligent Speech Signal Processing*, 81–100. <https://doi.org/10.1016/B978-0-12-818130-0.00005-2>
- Diaz, J. (2021, July 30). *Science in 5 - Episode #47 - Post COVID-19 condition*. World Health Organization. [https://www.who.int/emergencies/diseases/novel-coronavirus-2019/media-resources/science-in-5/episode-47---post-covid-19-condition?gclid=CjwKCAjwk6-LBhBZEiwAOUDp2EHbaBDcHH5Jr3gxXrZIPf3i5DTcYqMUpwzWNF1q-\\_g0rsnFcR29xoC2UQQAvD\\_BwE](https://www.who.int/emergencies/diseases/novel-coronavirus-2019/media-resources/science-in-5/episode-47---post-covid-19-condition?gclid=CjwKCAjwk6-LBhBZEiwAOUDp2EHbaBDcHH5Jr3gxXrZIPf3i5DTcYqMUpwzWNF1q-_g0rsnFcR29xoC2UQQAvD_BwE)
- Galiatsatos, P. (n.d.). *COVID-19 Lung Damage Featured Experts*. <https://www.hopkinsmedicine.orghttps://www.hopkinsmedicine.org/profiles/details/panagis-galiatsatos>
- How the Lungs Work | NHLBI, NIH*. (2020, October 6). <https://www.nhlbi.nih.gov/health-topics/how-lungs-work>
- Lee, Y. S., Pathirana, P. N., Steinfert, C. L., & Caelli, T. (2014). Monitoring and Analysis of Respiratory Patterns Using Microwave Doppler Radar. *IEEE Journal of Translational Engineering in Health and Medicine*, 2, 1–12. <https://doi.org/10.1109/JTEHM.2014.2365776>
- Lu, Y., Li, X., Geng, D., Mei, N., Wu, P. Y., Huang, C. C., Jia, T., Zhao, Y., Wang, D., Xiao, A., & Yin, B. (2020). Cerebral Micro-Structural Changes in COVID-19 Patients - An MRI-based 3-month Follow-up Study. *EClinicalMedicine*, 25. <https://doi.org/10.1016/J.ECLINM.2020.100484>
- Shah, R. (n.d.). *IOT AND AI IN HEALTHCARE: A SYSTEMATIC LITERATURE REVIEW*. [https://doi.org/10.48009/3\\_iis\\_2018\\_33-41](https://doi.org/10.48009/3_iis_2018_33-41)
- Shastri, S., Kansal, I., Kumar, S., Singh, K., Popli, R., & Mansotra, V. (2022). CheXImageNet: a novel architecture for accurate classification of Covid-19 with chest x-ray digital images using deep convolutional neural networks. *Health and Technology*, 12(1), 193–204. <https://doi.org/10.1007/s12553-021-00630-x>
- Yong, S. J. (2021). Long COVID or post-COVID-19 syndrome: putative pathophysiology, risk factors, and treatments. *Infectious Diseases*, 53(10), 737–754. <https://doi.org/10.1080/23744235.2021.1924397>

## APPENDIX A THE CODE OF THE CAPTURING CIRCUIT

```
#include "ArduinoQueue.h"
#include "ESPDateTime.h"
#include <Firebase_ESP_Client.h>
#include <addons/RTDBHelper.h>

// 1. Define the WiFi credentials
#define WIFI_SSID "POCO X3 Pro" //Wifi Name
#define WIFI_PASSWORD "123123123" //Wifi Password

// 2. Define the RTDB URL and database secret
#define DATABASE_URL "breathingclassifier-default-rtdb.firebaseio.com/"
//<databaseName>.firebaseio.com or <databaseName>.<region>.firebasedatabase.app
#define DATABASE_SECRET "*****" // DataSecret token

// 3. Define the Firebase Data object
FirebaseData fbdo;

// 4. Define the FirebaseAuth data for authentication data and FirebaseConfig data for config data
FirebaseAuth auth;
FirebaseConfig config;

// 5. Define the System Configuration
#define FSR 35
#define NewIDPath "/NewID"
#define LED_BUILT_IN 2

// 6. Define Global Variables
ArduinoQueue<int> ReadingsQueue(18000);
String patientIDPath = "/Patterns/";
int monthDays[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

void SetDateTime(long int TimeStamp, int* Year, int* Month, int* Day, int* Hour, int* Mint, int*
Sec)
{
    // Setting the Data and Time from the global time stamp
    Serial.printf("Time Stamp now: %ld\n", TimeStamp);
    *(Sec) = TimeStamp % 60;
    TimeStamp /= 60;
    Serial.printf("Time Stamp now: %ld\n", TimeStamp);
    TimeStamp -= 60 * 4;
    *(Mint) = TimeStamp % 60;
    TimeStamp /= 60;
```

```

Serial.printf("Time Stamp now: %ld\n", TimeStamp);
*(Hour) = TimeStamp % 24;
TimeStamp /= 24;
TimeStamp -= 12;
*Year = 1970 + TimeStamp / 365;
TimeStamp %= 365;
*Month = 0;

//Check if the year is a leap year
if (*Year % 4 == 0)
    monthDays[1] = 29;

while (TimeStamp > 0)
{
    Serial.printf("Time Stamp In lop: %ld\n", TimeStamp);
    TimeStamp -= monthDays[*Month];
    *Month = *Month + 1;
}
*Day = TimeStamp + monthDays[*Month - 1];
}

//Reading the FSR Value
void ReadSensor (void * par)
{
    // Clearing the Queue before Startiing
    while (ReadingsQueue.item_count())
    {
        Serial.println(ReadingsQueue.dequeue());
    }

    // Continous Reading of the data
    while (1)
    {

        static float SumOfReadings = 0; // -> To Calculate the Average
        static float Aver = 0; //-> Intitalize the average
        static int Count = 0;

        //Reading the change in voltage and calibrate it by deducting the average
        float Reading = analogRead(FSR) * (4000 / 4095.0);
        Reading -= Aver;
        SumOfReadings += Reading;
    }
}

```

```

//Check if the Queue is full, remove the first reading before adding a new reading
if (ReadingsQueue.item_count() == 18000)
    ReadingsQueue.dequeue();

ReadingsQueue.enqueue(Reading);

//Calibrating the average every 10 Seconds
if (Count == 100)
{
    Aver += SumOfReadings / Count;
    SumOfReadings = 0;
    Count = 0;
}
Serial.print("The Sensors Reading is:");
Serial.println(Reading);
vTaskDelay(99 / portTICK_PERIOD_MS); //-> holds the function untill the next 100ms comes
}
}

//Sending the Data to the database
void DataBaseSend (void * par)
{
    // Initializing Variables
    long int GlobalTimeStamp = 0;
    float reading = 0;
    String patientIDPathWithCounter;
    FirebaseJson json;
    String DayNumber = "/Day", HourNumber = "/Hour", MinNumber = "/Minute", SecNumber =
"/Seconds";
    int Year = 0, Month = 0, Day = 0, Hour = 0, Mint = 0, Sec = 0, QueCount = 0;

    //getting the current TienStamp and set the Data and time Accordingly
    Serial.printf("Set timestamp... %s\n", Firebase.RTDB.setTimestamp(&fbdo, "/Global
Timestamp") ? "ok" : fbdo.errorReason().c_str());
    Serial.print("TIMESTAMP (Seconds): ");
    Serial.println(fbdo.to<String>());
    Serial.println("Setting Time.....");
    SetDateTime(fbdo.to<int>(), &Year, &Month, &Day, &Hour, &Mint, &Sec);
    Serial.printf("Year NOW:  %d\n", Year);
    Serial.printf("Month NOW:  %d\n", Month);
    Serial.printf("Day NOW:  %d\n", Day);
    Serial.printf("Hour NOW:  %d\n", Hour);
    Serial.printf("Min NOW:  %d\n", Mint);
    Serial.printf("Sec NOW:  %d\n", Sec);

```



```

while (1)
{
    QueCount = ReadingsQueue.item_count();
    if (WiFi.status() != WL_CONNECTED)
    {
        WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
        Serial.print("Reconnecting to Wi-Fi.....");
        while (WiFi.status() != WL_CONNECTED)
        {
            digitalWrite(LED_BUILT_IN, HIGH);
            delay(50);
            digitalWrite(LED_BUILT_IN, LOW);
            delay(50);
        }
    }

    if (QueCount >= 10)
    {
        for (int i = 0; i < 10; i++)
        {
            json.set(String(i), ReadingsQueue.dequeue());
        }

        patientIDPathWithCounter = patientIDPath + "/" + String(Year) + "/" + String(Month) + "/" +
        String(Day) + "/" + String(Hour) + "/" + String(Mint) + "/" + String(Sec);

        //modiy the Data and time each reading
        Sec++;
        if (Sec == 60)
        {
            Mint++;
            Sec = 0;
            if (Mint == 60)
            {
                Hour++;
                Mint = 0;
                if (Hour == 24)
                {
                    Day++;
                    Hour = 0;
                    if (Day > monthDays[Month - 1])
                    {
                        Month++;
                        Day = 1;
                    }
                }
            }
        }
    }
}

```

```

        if (Month > 12)
        {
            Year++;
            Month = 1;
            if (Year % 4 == 0)
                monthDays[1] = 29;
            else
                monthDays[1] = 28;

        }
    }
}

Serial.printf("Sending data... %s\n", Firebase.RTDB.setJSON(&fbdo,
patientIDPathWithCounter, &json) ? fbdo.to<FirebaseJson>().raw() : fbdo.errorReason().c_str());
}
vTaskDelay(1 / portTICK_PERIOD_MS);
}
}

void setup() {
    String ID;
    // put your setup code here, to run once:
    Serial.begin(115200);

    //Setting Pins Configuration
    pinMode(FSR, INPUT);
    pinMode(LED_BUILT_IN, OUTPUT);
    delay(100);

    //Connecting to WIFI
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Connecting to Wi-Fi.....");
    while (WiFi.status() != WL_CONNECTED)
    {
        digitalWrite(LED_BUILT_IN, HIGH);
        delay(50);
        digitalWrite(LED_BUILT_IN, LOW);
        delay(50);
    }
    WiFi.setAutoConnect(1);
    Serial.println();
    Serial.print("Connected with IP: ");

```

```

Serial.println(WiFi.localIP());
Serial.println();

//Connecting to Firebase Database
config.database_url = DATABASE_URL;
config.signer.tokens.legacy_token = DATABASE_SECRET;
Firebase.begin(&config, &auth);
Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);
Serial.println("Connected To the FireBase");

//Setting the Patients ID to the next new ID
Serial.printf("Patient ID: %s\n", Firebase.RTDB.getString(&fbdo, F(NewIDPath)) ? fbdo.to<const
char *>() : fbdo.errorReason().c_str());
ID = Firebase.RTDB.getString(&fbdo, F(NewIDPath)) ? fbdo.to<const char *>() :
fbdo.errorReason().c_str();
patientIDPath = patientIDPath + ID;
Serial.printf("Setting New Patient ID: %s\n", Firebase.RTDB.setInt(&fbdo, F(NewIDPath),
ID.toInt() + 1) ? "ok" : fbdo.errorReason().c_str());

//Starting the 2 Main Tasks, and pinning each to a core
xTaskCreatePinnedToCore(DataBaseSend, "Sending Data to the Data Base", 10000, NULL, 1,
NULL, 0);
xTaskCreatePinnedToCore(ReadSensor, "Reading From The Sensor", 1024, NULL, 1, NULL, 1);

// Delete "setup and loop" task
vTaskDelete(NULL);

}

void loop() {
    // put your main code here, to run repeatedly:

}

```

## APPENDIX B CODE FOR DATASET CREATION AND MODEL TRAINING

```
# Importing Libraries

# In[1]:

import pandas as pd
import tensorflow as tf
from tensorflow import keras
import numpy as np
from matplotlib import pyplot as plt
import pyrebase

# Getting the data from the database

# In[2]:

cr = {

    "apiKey": "AIzaSyDC40VinaqprhULKkZ_g1fWfzCdukwfzAE",
    "authDomain": "breathingclassifier.firebaseio.com",
    "databaseURL": "https://breathingclassifier-default-
rtddb.firebaseio.com",
    "projectId": "breathingclassifier",
    "storageBucket": "breathingclassifier.appspot.com",
    "messagingSenderId": "212495536430",
    "appId": "1:212495536430:web:dca2aae1a99c913cd7aabf",
    "measurementId": "G-SEED5QBFQ0",
    "serviceAccount": "serviceAccountKey.json"
}

# Fetch the service account key JSON file contents
firebase = pyrebase.initialize_app(cr)
db = firebase.database()

Pat = db.child("Patients").get().val()

# In[3]:

for ID in Pat:
    if ID == '100':

db.child("Patterns").child(ID).child(2022).child(5).child(25).set(Pat[ID
][ '6' ])
    continue
    db.child("Patterns").child(ID).child(2022).child(5).set(Pat[ID])
```

```

# Preparing the dataset

# In[4]:

unmappedPat = {}
mappedPat = {}
for ID in Pat:
    lst = []
    mx,mn = -9999,9999
    unmappedPat[ID] = []
    for D in Pat[ID]:
        for H in Pat[ID][D]:
            #print(Pat[ID][D])
            #print("D:" + D + " H:" + str(H))

            if not(H is None):
                if type(H) is int or type(H) is str:
                    X = Pat[ID][D][H]
                else:
                    X = H
                for M in X:
                    if type(M) is int or type(M) is str:
                        Y = X[M]
                    else:
                        Y = M
                for S in Y:
                    if S is None:
                        if len(lst) >=300:
                            unmappedPat[ID].append(lst)
                            lst = []
                            continue
                    if type(S) is list:
                        mx = max(mx,max(S))
                        mn = min(mn,min(S))
                        lst.extend(S)
                    else:
                        mx = max(mx,max(Y[S]))
                        mn = min(mn,min(Y[S]))
                        lst.extend(Y[S])

            if len(lst) >=300:
                unmappedPat[ID].append(lst)
                mappedPat[ID]= []

    for i in range(len(unmappedPat[ID])):
        mappedPat[ID].append(np.interp(unmappedPat[ID][i],[mn,mx],[-100,100]))

```

```
# In[5]:
```

```
normList = []
lCList = []
normIDs = [100,104,105,106,108]
lCIDs = [102,103,107,109,110]

for ID in mappedPat:
    try:
        normIDs.index(int(ID))
        normList.append(mappedPat[ID])
    except:
        lCIDs.index(int(ID))
        lCList.append(mappedPat[ID])
```

```
# In[7]:
```

```
dataLst = []
trgtLst = []
for i in normList:
    for j in i:
        Data =
tf.keras.preprocessing.timeseries_dataset_from_array(j, None, sampling_rate=1, sequence_length = 300, batch_size = 1, shuffle=True)
        dataLst.extend(list(Data.as_numpy_iterator()))
trgtLst = [[1,0]]*len(dataLst)
for i in lCList:
    for j in i:
        Data =
tf.keras.preprocessing.timeseries_dataset_from_array(j, None, sampling_rate=1, sequence_length = 300, batch_size = 1, shuffle=True)
        dataLst.extend(list(Data.as_numpy_iterator()))
trgtLst.extend( [[0,1]]*(len(dataLst)-len(trgtLst)))
```

```
# Shuffling the datasets
```

```
# In[8]:
```

```
npData = np.array(dataLst)
npTrgt = np.array(trgtLst)
```

```
# In[9]:
```

```
Shuf = np.random.permutation(len(npData))
npData = npData[Shuf]
npTrgt = npTrgt[Shuf]
Shuf = np.random.permutation(len(npData))
npData = npData[Shuf]
```

```

npTrgt = npTrgt[Shuf]

# Creating the final dataset as a numpy array

# In[10]:

DataSet = np.array([elem for twod in npData for elem in twod])
Trgt = np.array([elem for elem in npTrgt])
print(len(DataSet), "\t", len(Trgt))

# Saving the data offline

# In[11]:

df1 = pd.DataFrame(DataSet)
df2 = pd.DataFrame(Trgt, columns=['Normal', 'Long Covid'])
df = pd.concat([df1, df2], axis=1)
df.to_csv("new Data\Final Data set.csv", float_format="%.2f")

# Compiling and evaluating the model

# In[12]:

def residual_block(x, filters, conv_num=3, activation="relu"):
    # Shortcut
    s = keras.layers.Conv1D(filters, 1, padding="same")(x)
    for i in range(conv_num - 1):
        x = keras.layers.Conv1D(filters, 3, padding="same")(x)
        x = keras.layers.Activation(activation)(x)
    x = keras.layers.Conv1D(filters, 3, padding="same")(x)
    x = keras.layers.Add()([x, s])
    x = keras.layers.Activation(activation)(x)
    return keras.layers.MaxPool1D(pool_size=2, strides=2)(x)

def build_model(input_shape, num_classes):
    inputs = keras.layers.Input(shape=input_shape, name="input")

    x = residual_block(inputs, 16, 2)
    x = residual_block(x, 32, 2)

    x = keras.layers.AveragePooling1D(pool_size=2, strides=1)(x)
    x = keras.layers.Flatten()(x)
    x = keras.layers.Dense(64, activation="relu")(x)
    x = keras.layers.Dense(32)(x)
    x = tf.keras.activations.relu(x, threshold=0.6)
    outputs = keras.layers.Dense(num_classes, activation="softmax",
name="output")(x)

    return keras.models.Model(inputs=inputs, outputs=outputs)

```

```

model = build_model((300,1), 2)

model.summary()

# Compile the model using Adam's default learning rate
model.compile(
    optimizer="Adam", loss="categorical_crossentropy",
    metrics=["accuracy"]
)
model_save_filename = "model.h5"

earlystopping_cb = keras.callbacks.EarlyStopping(patience=10,
restore_best_weights=True)
mdlcheckpoint_cb = keras.callbacks.ModelCheckpoint(
    model_save_filename, monitor="val_accuracy", save_best_only=True
)
History = model.fit(DataSet,Trgt,epochs =
50,validation_split=0.3,callbacks=[earlystopping_cb, mdlcheckpoint_cb])
score = model.evaluate(DataSet,Trgt)
print(score)

# In[13]:

converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the TF Lite model.
with tf.io.gfile.GFile('model2.tflite', 'wb') as f:
    f.write(tflite_model)

# In[14]:

print(History.history['val_loss'][-1])
plt.plot(History.history['loss'],label = 'Train Loss')
plt.plot(History.history['val_loss'],label = 'Test Loss')
plt.legend()
plt.show()

plt.plot(History.history['accuracy'],label = 'Train Accuracy')
plt.plot(History.history['val_accuracy'],label = 'Test Accuracy')
plt.legend()
plt.show()

model.summary()

# In[5]:

keras.utils.plot_model(model, "my_first_model_with_shape_info.png")

```