Intelligent Healthcare Monitoring System with Pulse Oximetry Analysis

by Baibhav Singh

Submission date: 08-Jun-2021 05:14PM (UTC+0530)
Submission ID: 1602794227
File name: 17BCE2359-Capstone_Project_Review_3.pdf (2.74M)
Word count: 9540
Character count: 50265

Intelligent Healthcare Monitoring System with Pulse Oximetry Analysis

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

B.Tech.

by

Baibhav Singh

17BCE2359

Under the guidance of

Prof. Madiajagan M

School of Computer Science & Engineering

VIT, Vellore.

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

June, 2021

DECLARATION

I hereby declare that the thesis entitled "Intelligent Healthcare Monitoring System with Pulse Oximetry" Analysis submitted by me, for the award of the degree of *Bachelor of Technology in Programme* to VIT is a record of bonafide work carried out by me under the supervision of Madiajagan M.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date :

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled "Intelligent Healthcare Monitoring System with Pulse Oximetry" submitted by **Baibhav Singh & 17BCE2359, SCOPE,** VIT, for the award of the degree of *Bachelor of Technology in Programme*, is a record of bonafide work carried out by him / her under my supervision during the period, 01. 01. 2021 to 09.06.2021, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : **Signature of the Guide**

Internal Examiner **External Examiner**

Head of the Department

SCOPE

ACKNOWLEDGEMENTS

The main idea of the project sparked into my mind when I was going through a lot of articles regarding techniques tacking different kind of disease and current scenario. I felt we really lacked to use the technology we have in our hand to improve the health care system of my country, Nepal. In this part Dr Madiajagan M played an intrinsic role, as my project guide and as a companion. I would like to show appreciation to him.

In addition, I would thank my family who encouraged me continuously while I was not able achieve the project goals. Some of the parts that was used to develop the device was not available in my country so my friends really helped me getting it. I provide my immense gratitude who helped me achieve this goal

Baibhav Singh

Executive Summary

In light of recent events that have exhausted the medicals facilities of developed countries, the SARS-CoV-2 patients find difficulties in breathing after an increase in the concentration of carbon dioxide in the lungs and a decrease in blood oxygen level, which causes loss of patient's life. Various researchers have found that blood oxygen level and body temperature are significant factors used to monitoring covid or chronic obstructive diseases patients. The most standardized test for these kinds of conditions is the spirometry test. Similar to this, PCR and RT are also considered standardized tests for covid suggested by WHO. Real-time monitoring of these chronic obstructive diseases to heal with proper medications and treatment.

The proposed architecture helps monitor the activity, body temperature, the oxygen saturation level parameters of disease caused by a coronavirus. Preliminary covid symptoms can be detected using this proposed system. The developed architecture includes a hardware module to monitor the patient's body temperature and oxygen saturation level. The obtained data from the hardware are sent to the patient's doctor. The data is then sent to the cloud which stores it for further analysis. After the report the generated from machine learning model. Then feedback system to further discuss reports.

CONTENTS

	Page No.
Acknowledgement	i
Executive Summary	ii
Table of Contents	Iii
List of Figures	ix
List of Tables	xiv
Abbreviations	xvi
Symbols and Notations	xix
1 **INTRODUCTION**	1
1.1 Theoretical Background	1
1.2 Motivation	2
1.3 Aim of the Proposed Work	3
1.4 Objective(s) of the Proposed Work	4
2. Literature Survey	**5**
2.1. Survey of the Existing Models/Work	5
2.2. Summary/Gaps identified in the Survey	8
3. Overview of the Proposed System	**9**
3.1. Introduction and Related Concepts	9
3.2. Framework, Architecture or Module for the Proposed System	10
3.3. Proposed System Model	12
4. Proposed System Analysis and Design	**15**
4.1. Introduction	15
4.2. Requirement Analysis	17
4.2.1.Functional Requirements	
4.2.1.1. Product Perspective	
4.2.1.2. Product features	
4.2.1.3. User characteristics	
4.2.1.4. Assumption & Dependencies	
4.2.1.5. Domain Requirements	
4.2.1.6. User Requirements	
4.2.2.Non Functional Requirements	18
4.2.2.1. Product Requirements	
4.2.2.1.1. Efficiency (in terms of Time and Space)	

4.2.2.1.2. Reliability
4.2.2.1.3. Portability
4.2.2.1.4. Usability
4.2.2.2. Organizational Requirements 19
4.2.2.2.1. Implementation Requirements (in terms of deployment)
4.2.2.2.2. Engineering Standard Requirements
4.2.2.3. Operational Requirements (Explain the applicability for your work w.r.to the following operational requirement(s))
- Economic
- Environmental
- Social
- Political
- Ethical
- Health and Safety
- Sustainability
- Legality
- Inspectability

4.2.3.System Requirements 20
4.2.3.1. H/W Requirements(details about Application Specific Hardware)
4.2.3.2. S/W Requirements(details about Application Specific Software)

5. Results and Discussion 29

6. References 50

APPENDIX A

List of Figures

Figure No.	Title	Page No.
3.1	System Model overview	13
3.2	System Design overview	15
3.3	Use Case Diagram	13
3.4	Activity Diagram	13
3.5	Correlation Formula	13
4.1	Circuit Diagram	13
4.2	Arduino uno	13
4.3	GPS Triangulation	13
4.4	GPS Picture	13
4.5	DHT11	13
4.6	MAX30100 sensor.	13
4.7	Bread Board	13
4.8	Connecting Wires	13
4.9	Figure caption	13
5.1	Data collection microprocessor output	29
5.2	Verification of live location using microprocessor.	30
5.3	Actual Circuit Diagram.	31
5.4	Output From Microprocessor.	32
5.5	Result data received in server.	33

5.6	serial port server	34
5.7	Result data stored and Visualized at Thing Speak IoT.	13
5.8	Result data .csv exported from Thing Speak IoT.	37
5.9	Machine Learning Model	13
5.10	Chat Application.	13

List of Tables

Table No.	Title	Page No.

List of Abbreviations

SARS-CoV-2	severe acute respiratory syndrome coronavirus 2
Max3010	Pulse oximeter
Neo gm GPS	GPS module
IoT	Internet of Things
LM35	Temperature Senor
DHT11	Temperature Senor
NPR	Nepalese Rupee
MLX90614	Infrared Temperature sensor

Symbols and Notations

1. INTRODUCTION

1.1. Theoretical Background

Communicable diseases through airborne is a transmission that delivers very minute liquid droplets. Those blobs are also identified as microdroplets. The size is more limited to five micrometers. In observation, water blobs usually are more significant than five micrometers. Because it is minute and feathery, the aerosol carrying the SARS-CoV-2 coronavirus can survive levitating in the air for many hours. Besides, these particles can also drift quite far. Few investigations appeal that disease caused by coronavirus can exist on the surface or living body for four hours. While separate reports say, it can last sixteen hours. Recent research shows coronaviruses that showered in the air can stay alive for at least three hours. Nevertheless, the scientists stressed that the experiment carried in a laboratory is unconventional from actual conditions where the outcomes can change. However, the cases of corona that are spreading very fast have sharpened the doubt that airborne infection is possible. For instance, a place in USA, a person contaminates at a minimum of forty-five additional person who has sung with him in the same choir. As people are getting so much infected, it would be straightforward if any device could simulate and find a relation between blood oxygen level and breathing also, with other health parameters. We could also detect the symptoms of disease caused by the coronavirus and save lives. There are devices like ventilators that aids patients while they are suffering from breathing problems but used only for severe cases in which blood oxygen drops below the standard value. Likewise, smartwatches can check blood oxygen level but aren't reliable to use it as a clinical device. The body temperature can be measured easily using a standard thermometer. So, a device that could include monitoring these things such as body temperature, blood oxygen level and patient's activity could be helpful. Recent pandemic has shown how these chronic obstructive diseases can spread rapidly and difficult to control without proper resources and management.

The World Health Organization suggests finding out and monitoring Chronic obstructive disease or covid at an earlier stage is the best way to deal with this virus. As per WHO instruction, doctors diagnose and monitor the seriousness of symptoms through PCR and spirometry tests. Rapid testing is also considered an expeditious way to check the patient's blood to check whether the suspected patient is positive to covid-19 but isn't reliable cause it might not always give accurate results.

1.2. Motivation

There thousands of deaths that has happened because of corona virus and other chronic obstructive diseases still there are no proper system that monitor patients with these diseases. With the help of this system, we going to promote the concept of homecare is becoming increasingly vital role in medical technology and will play important role in future.

Covid is a Chronic obstructive disease, which causes shortness of breathing, gasping, irritation in the respiratory tract, etc. The recurrence and grimness of the conditions vary from different age ground people, but it primarily affects old age people. Early morning cold effects, stress, common cold, pneumonia, etc., leads to aggravating of the disease. Apart from its allergies indulged, such as loss in taste and smell, a patient may rise from covid disease problems.

The monitoring application and doctor examining the sensed values can take necessary action on the medication and treatment of the covid patient. The developed architecture is a practical, authentic, and easy-to-use device to determine the symptoms of covid or chronic obstructive diseases.

1.3. Aim of the proposed Work

An average blood oxygen saturation level would be within ninety-five and hundred percent; anything under ninety is acknowledged harmful. Hence, those patient coronavirus patients with measurements get below fifty percent; they might also suffer from Hypoxemia. When oxygen levels drop this level, patients have much more apparent difficulty breathing. The fixed data and the monitoring outcomes are also directly connected and collected on an IoT system. The IoT system sends oxygen level, body temperature, heart rate readings to the cloud for analytics immediately from the IoT system.

There are many existing monitoring systems, but the recently coined concept of home isolation after the coronavirus pandemic happened. This architecture help government, healthcare department, and patients to promote home isolation and control the spread of the virus.

According to recent budget released by Nepal government, Ministry of health and population was allocated with 122.77 billion NPR out of 1647.57 billion NPR which is only 7.5 percentage of the total budget. The budget in health sector have increased recently but before it used to be less than this statistic. Hence the system was designed such a way that it can get accommodated to the budget allocated by even poor countries.

This paper aims to analyze diseases and studies aiming to automated diagnosis or monitoring of infectious diseases whose symptoms are detected with body temperature, blood pressure, and oxygen saturation level. In this work, we propose a system that monitors the patients' body parameters; if there is any unusual behavior, they will be symptoms. The monitoring system will help the area where the expert in respiratory diseases analysis may not be available. This project curbs human error while detecting these viruses or diseases by using intelligent monitoring and analysis systems. Evaluation of the percentage of detection and efficiency shows which monitoring, i.e., body temperature, blood pressure, and oxygen saturation level, has a higher predictive rate, a comparative study tested on the same input slide.

1.4. Objectives of the proposed work

In this project plan to write a server application so that we can interface using our Arduino program with other clients in the network. Finally, we plan on making froward to a cloud application that will fetch data from sensors and store it to server to display it in user friendly way. This will be an excellent way to setup a home health care monitoring system.

On the other hand, we are going to use random forest statics from the patient to predict way before whether the patient has got covid 19 or not. The accuracy might differ from algorithm to algorithm.

The list of objectives would be as follows:

- Read the data from the sensor according to the requirement of the project.
- Storing it to the ThingSpeak IoT for storage and visualization.
- Using Machine learning in the data to predict whether the user has got any disease or not.
- Finally, the feedback system which helps the user to get feedback from the doctor.

2. LITERATURE SURVEY

2.1. Survey of the Existing Models/Work

This article reviews about asthma disease and how can we monitor this disease with the help of Arduino [1]. As mentioned, this disease can be easily treated when medicine and treatment are given in time. The proposed architecture tests different activities and environmental parameters of asthma. The basic parameters that are going to be examined are temperature, humidity, air pressure, activity, and volatile gases are collected and then send to the patient's doctors via the GSM module. The doctor then examines the data and then gives suitable treatment and medicines for asthma.

Another paper reviews about e-health monitoring system which the device designed monitors the difficulties experienced by the users in flight [2]. The main goal of this system comes after knowing of the condition inside cab travelers in cab conditions, such as low humidity risk factors, lack of oxygen, low pressure, and in patients suffering from cardiovascular diseases etc. Monitoring the oxygen level of the patient during takeoff/landing and during flight hours provides valuable information on the health status of passengers. This information later could be used to travelers of similar alignment of health difficulties.

Similarly, this paper screen and regulate the saline flow rate [3]. Conscientious flow has to be clutched to foreshortening the entrance level of the patient's heart rate, body temperature, and oxygen level saturation in the blood. The into a vein mixture used intermittently in the dispensary has to be checked for its pureness. For the variation in threshold level of the patient's body condition, the saline stream has to be resolved. The estimations received from the progress of the subject to the main regulator, which is correlated to the cloud, are refreshed systematically to continue away from the waste of records. The refreshed information collections are administered to the pharmacologist and device so that the flow rate of saline is regulated naturally in accordance with the data received. The machine learning algorithm is applied to prognosticate certain further changes in figures which are collected from victims so that the regulator will work flexibly. This output provides more reliable outcomes centered on correctness level evaluation and productivity growth in courses of a further quick acknowledgment.

Whereas, in this paper, various considered devices for particular movable medical management checking and supervision should be increased [4]. The numerous leading devices cover the measurement of blood pressure, bodily action, blood sugar level, weight, analysis, heartbeat rate, cardiogram, Spo2, and rest position. They have improved Bluetooth device Cloud storage, and medical capabilities are regulated with a smart device. The device has several comprehensive high ranges as both are cost-effective, easily usable, understand, and describe; and, it doesn't need a constant energy provider and skillful investigators. The investigator would primarily decrease medical management expenses and commence to well and long-term well-being consequences. The section presents a summary of several monetary devices, smartwatches, and other devices for medical health management adjacent to the difficulties associated.

The fifth article exhibits the computer application and tools design of the digitalized health pulse oximeter method combined for travelers regulated during flight hours [5]. The idea of the intelligent operation comes after the observation that throughout the flight hours, the privileged flight situations, for example, deficiency of oxygen, under normal pressure, and lack of humidity, are uncertain circumstances in victims grieving suffering from the constant obstructive lunglike disease, cardiovascular illnesses, asthma, lung emphysema, surgical operations experienced newly, epilepsy, abnormal sugar level, psychic disorders, venous thrombosis, and communicable diseases. Regulating of oxygen saturation blood and heartbeat rate throughout the states, for example, during flight duration, may contribute relevant knowledge on the health situation of travelers, and consequently, here knowledge gathered could be practiced in the medicinal department to make suggestions to subjects having particular illnesses and to fly via the atmosphere. According to current researches, through traveling by air, the airplanes are pressurized to seventy-five percentage of regular meteorological pressure, and under normal conditions of oxygen in the plasma will commence disease called hypoxia, a lack in the quantity of blood oxygen to arrive the body. Furthermore, the lowered force in the airplanes and deficiency of blood oxygen may threaten the life of the subjects that suffer from the continual obstructive lunglike illness.

This article offers a device-based conventional system, both stiff or bendable, through a standard power source. This involves devices with internet connectivity. Whereas in different situations, the device is more disorderly and involves garments and textiles with divided purposes, into which automatics are closely consolidated [6]. In this situation, the construction is not understood as the developed devices have to be waterproof, sustainable in quality, and portable to take it to different places and visible from inside. The latest portable device operation is to be established earlier or regularly; it has to have a trustworthy series and very reduced waste, communicating wirelessly to all medical-related knowledge without necessitating its person in either form. Association has to be completed and managed, decreasing the consequences of representatives like heat, humidity, and activity. Notwithstanding certain essential characteristics, there is a large expansion of accessories.

This writing performs the idea and experiment of a pulse oximeter in special conditions by the modified process [7]. The semiconductor includes a photodiode, unadulterated pass filter, computerized manageable converters, improved signal computer, impedance amp, and control counter for timing. The empirically included current properties of different designs that the direct current voltage of the no characteristic impedance amplifier, the no characteristic impedance gain of the no characteristic impedance amplifier, and the median repetition and communication capacity of the simplistic filters the band gives a decent competition with the design simulations. After accentuated data references and combined locked exposure, the sensor completely contains the barrier from unique data and noise. In a breath and discharge operation, the singly chipped sensor exhibits constant and relative achievement to economic pulse oximetry with a median of one percentage variation. The singly chipped sensor allows an insignificant and strong design resolution that allows a range to the wearable tools for medical condition monitoring.

2.2. Summary/Gaps identified in the Survey

There has been a lot of systems proposed to control the spread of these chronic obstructive diseases. There are different proposed architecture tests, different activities, and environmental parameters of asthma. The basic parameters that are going to be examined are temperature, humidity, air pressure, movement, and volatile gases are collected and then send to the patient's doctors via the GSM module [1].

Given all the research projects conducted, we can say that the real-time monitoring of chronic obstructive diseases like covid-19, asthma is of enormous significance in patient's health and recovery. Recently we have also come to know that spo2 and body temperature plays a vital role in monitoring the patient's health, which has been missing in all these architectures. This project solves these problems by monitoring the patient's location, body temperature, and spo2 level. The health official can quickly know the health of the patient in real-time.

Basically, with the help of the architecture proposed, we will solve the major problem that we are facing in this pandemic. Firstly, the device helps to monitor the patient's primary health condition like spo2 level, which thresholds to be more than 92% of body temperature etc. from home itself, which reduces the risk of health workers and other frontline workers to get infected. Then after that, the doctor analyses the data received from the sensors and validates whether the patients are in safe conditions. If not, then changes in medication and treatment can be easily made through feedback system or necessary steps that can be taken.

3. OVERVIEW OF THE PROPOSED SYSTEM

3.1. Introduction to Related Concepts

The base architecture is divided into four parts data collection, storage, retrieval and analysis. The Arduino will receive data from the data sensors and then sends it to the server. The server sends the data to the cloud in which it is stored and visualized. The data then can be fetched using API request or even by exporting .csv files. The data is then passed through machine learning algorithm which will use that data to analyze whether the patient has got any chronic obstructive diseases, i.e., Covid 19. The patient can also use the chat application after the analysis to get the feedback from the doctor.

The server-side contains mainly three parts, i.e., Serial port server, Thing Speak IoT and chat application. The function of serial port server is read the data from serial port as string and then pass it to a get request which will send data to Thing Speak IoT. Thing Speak IoT is a cloud architecture that takes the data, helps to visualize it and store into .csv file or even send to other application using API. The chat application server is used for communication between patients and doctor.

Many alternatives replace in place of the components that I have used in this system.
First of let's look at Microprocessor as we all know that it is the brain of any system because it the only thing that processes the data and sends it to the serial port in the system. The wide ranges of microprocessors are Arduino, Mega, mini, Genuino, Zero, Due, Leonardo, Ethernet, Nano, etc. Other than that Node MCU, STM32 are also some of the fast and reliable Microprocessors. The reason I chose Arduino Uno is that it was the most genuine product readily available.

Similarly for temperature their different sensors available such as LM35, DS160, MLX90614 which is mainly used for temperature gun these days. MLX90614 would have been the best option out of all these but due unavailability of the component leads me to choose another option.

3.2. Framework, Architecture or Module for the Proposed System

The following diagram gives an overview for the design we are going for in which the first step is data collection from sensor, then data storage and visualization to ThingSpeak IoT, then analysis of data in which we first fetch it from the cloud and then apply Machine learning algorithm on the data i.e., Random Forest Algorithm and finally feedback system in which the patient can discuss about the report with the concerned doctor by sending the text message.

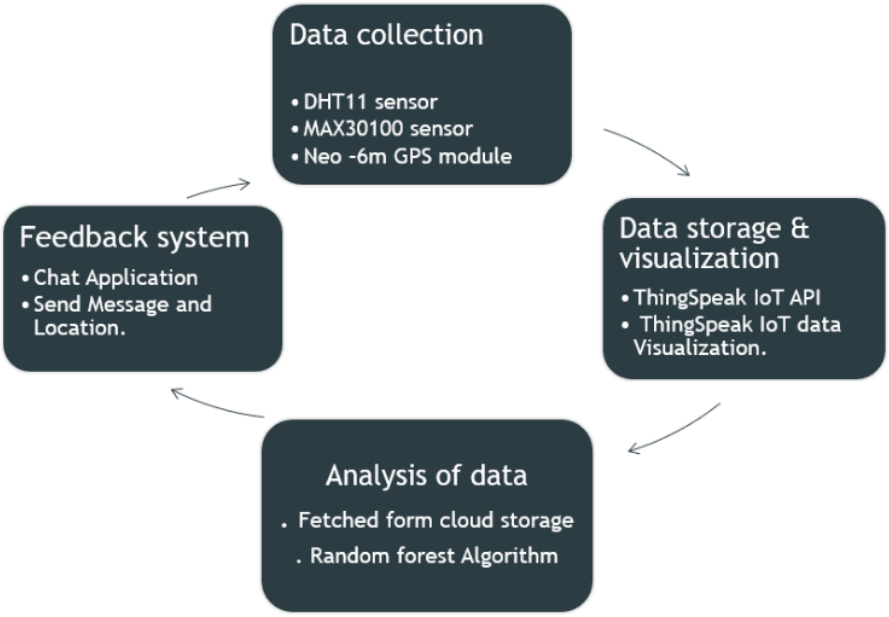


Fig 3.1. System Model Overview

The basic system architecture would consist of spo2 sensor module, body temperature sensor module, GPS sensor module. The main purpose of this project is to integrate these modules to gather data from the user. All the modules would be connected using Arduino board that can communicate with serial port. Then the server reads the data from serial port to send get request which is going to send data onto the cloud. Now the data collected can be processed and analyzed using ML. This would be really helpful in predicting the how likely a person is going to affected by the virus or diseases and thus encouraging steps to create more health posts for the betterment.

The system is also integrated with a chat application in which the patient and the doctor can chat simultaneously by sending messages and live location from the browser.

The following diagram gives an overview for the design we are going for:

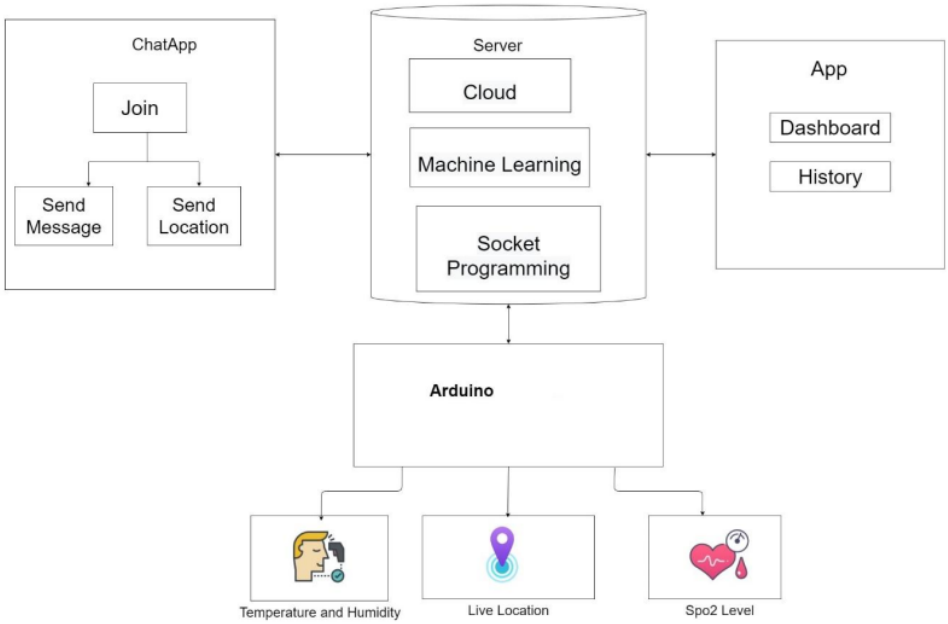


Fig 3.1. System Design Overview

3.3. Proposed System Model

Use Case Diagram

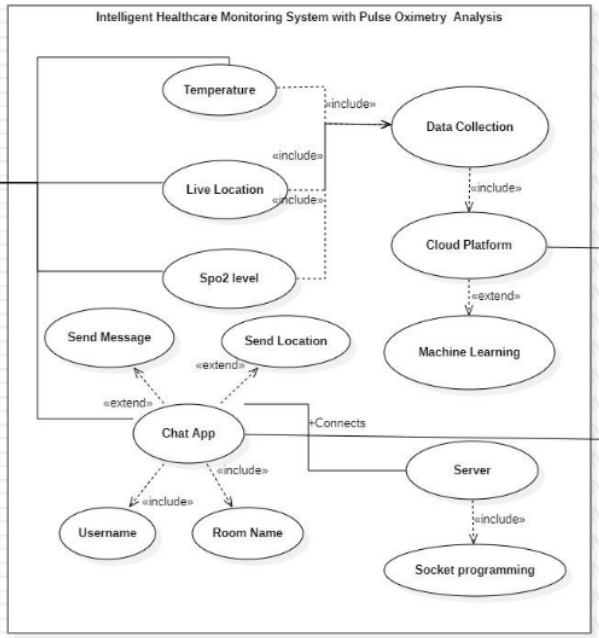


Fig 3.2. Use case Diagram

This use case model explains clearly how a user interacts with the system. First, the patient interacts with Arduino sensors and provides data on body temperature, humidity, blood saturation level, and live location coordinates. Then the information is collected through the IoT system and sent over to cloud storage. The data is then assessed through a Machine learning algorithm, and a detailed report is sent to the doctor. The patient can also chat with the doctor using a chat application by specifying a username and room name.

While at the doctor side the doctor can interact with cloud platform and chat application because the device isn't meant to be used by doctor but can assist in it.

Activity Diagram

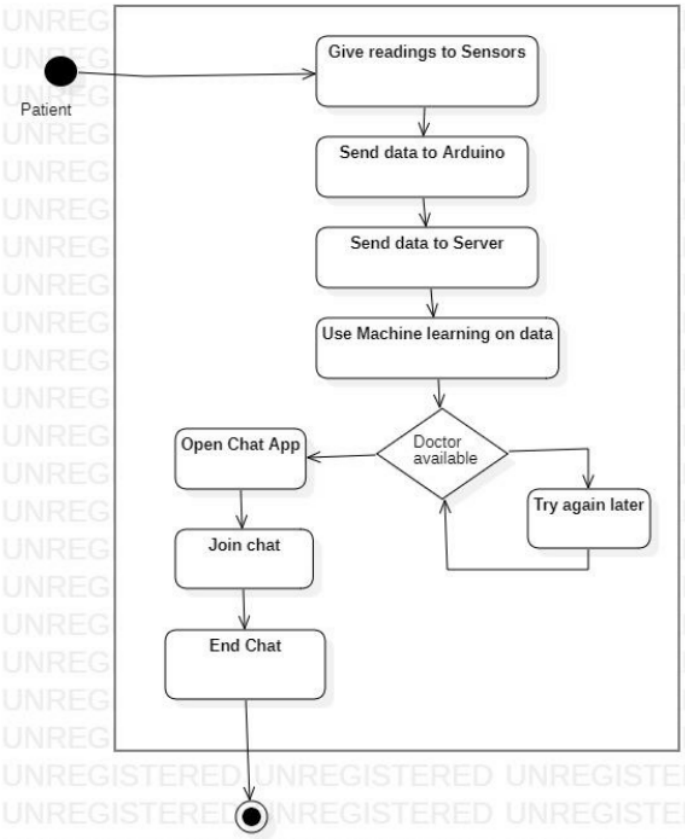


Fig 3.3. Activity Diagram

This activity diagram shows how patient activity will be like when interacting with the system or the device. The action of giving readings is just the overview part of the data collection process. For example, subcategories would be to provide reading for DHT11, Max30100, while Neo gm GPS module automatically takes the data while interacting with the other two sensors. Then comes sending data to the ThingSpeak IoT, which will do once the serial port server starts. Then the data is exported from ThingSpeak IoT to the machine learning model.

After the model predicts, the patient can use the feedback system for a doctor consultation, categorized into three categories. The first would be to open the website; the second would be to join the chat; the third would be to send the message and finally end the discussion after all the conversation is over.

Mathematical Model

The mathematical model only comes into action at the time of implementing machine learning algorithm. The basic mathematical model would be correlation between data while using the random forest algorithm.

While implementing this algorithm we user data in correlation to find out relation between all the columns in pairs.

$$r_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

Fig 3.4. Correlation Formula

The above shown figure is the simple formula that has been user in the system. Other than that, we use random forest algorithm to predict the output.

4. PROPOSED SYSTEM ANALYSIS AND DESIGN

4.1.Introduction

The microprocessor we are using is Arduino Uno, and we collect it with the sensor (GPS, Temperature, and pulse oximeter). The Arduino will now send data to the server using a Wi-Fi module that is attached externally. After that, we can visualize that data from the server or download it into CSV format, and data for ML algorithm is used using .csv form. The algorithm then predicts the results, which are then reported to the doctor. The patient can chat with the doctor in real-time and send messages or locations to get the necessary feedback.

The microcontroller we will be using Arduino uno which can run at default 490 Hz except pin 4 and 13 whose default frequency is 980 Hz and can give quick responses to temperature, spo2, heart rate, GPS module changes which makes it suitable for this purpose.

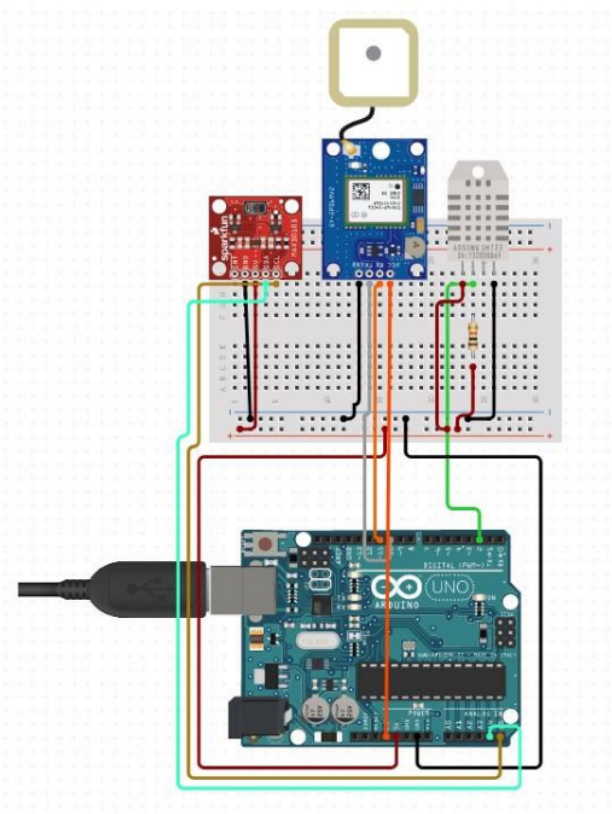


Fig 4.1. Circuit Diagram.

This circuit diagram looks close to this but this is the original representation because the website that I used to design the circuit didn't allow modify the circuit according my needs. The exact picture of circuit is shown below.

4.2.Requirement Analysis

This system helps user to provide body health conditions using the sensors which can be used to monitor his health condition so that the patient suffering from chronic obstructive diseases recover from it easily. The patient can view their data either in a serial port server or after uploading to the cloud. Cloud also has a data visualization technique that can show the user data in different charts and diagrams. Cloud data visualization will help the user and monitor the patient to know the trends of data. Users can fetch the data in either JSON format from the cloud's API or export it as a CSV file. The data then can be checked in a trained Machine learning model. There is a feedback system using messaging of a chat application.

4.2.1. Functional Requirements

4.2.1.1. Product Perspective

The product is used for gathering data from the patient and dispatching it onto the cloud server for the storage which can then be visualized. Then there is feedback system after analysis.

4.2.1.2. Product Features

The product implements different sensors so that we get data such as patients body temperature, blood oxygen saturation level, and Neo-gm GPS module. These data are then used to check whether the patient has got fever, breathing problems, activity in past days. This helps us to match the data that we have uses in the ML algorithm.

4.2.1.3. User characteristics

The user has to give input to the sensors. After that process is automated by the methodologies. The user can also chat with the doctor for feedback on the reports.

4.2.1.4. Assumption and Dependencies

The assumption is that the users have adequate knowledge about using the device and a basic knowledge of sending commands to the start the server. The dependencies would be to have zero manual error while entering the data to operate ML algorithm.

4.2.1.5. Domain Characteristics

The suggested system would prerequisite a stable internet connection on both the severs as the data is transfer from client-side server to cloud server for real-time data transfer. The MAX30100 sensor should also start properly because it will just fail to initialize the Arduino uno.

4.2.1.6. User Requirements

The proposed system would require an established internet connection on both the senders end as well as the receivers end for real-time data transfer. The user should be very well be aware about the usage of the architecture.

4.2.2 Non-Functional Requirements

4.2.2.1. Product Requirements

4.2.2.1.1. Efficiency

The product is more efficient than any of the existing model or system in which user has to input data has to be labor into the system and depends upon maintenance of sensors and strength of internet services. Overall, this is very reliable in terms of time and space

4.2.2.1.2. Reliability

The reliability of the system depends on proper use of the sensor while providing the readings. While providing the temperature place the sensor close to your body so that it provides reliable readings. Similarly, the product MAX30100 module can provide inaccurate data if the finger is not placed properly upon the sensor.

4.2.2.1.3. Portability:

The system is portable if certain cautions is taken while handling the microprocessor and the sensors. While transferring the device from one place to another just make sure that the water or any other fluid doesn't get in it. This may damage the sensor because water resistance of the device was not test during the time of deployment.

4.2.2.1.4. Usability

The device can be used in any laptop with internet connectivity after serial port server starts. If it says that "Pulse oximeter failed to initialize" then please restart the sensor properly so that pulse oximeter can start properly.

4.2.2.2. Organizational Requirement

4.2.2.2.1. Implementation Requirements

The implementation requirement will be to test the product in beta version before implementing to the mass. This will ensure that the defect in the device is addressed so that the user doesn't have to deal with it.

4.2.2.2.2. Engineering Standard Requirements

The engineering standard requirement would be to use respective libraries mention in the Arduino program with correct version respectively. The node Js server also user wide libraries whose version can keep on changing with time. If there are any changes then the program used also should be changed accordingly.

4.2.2.3. Operational Requirements

- Economic

  The device cost is going to be very low because cost factor was taking taken into consideration during the deployment of this project.

- Environmental

  As of now there are no environmental requirements because none of the sensor use are affected or have any kind of adverse effects on the environment.

- Social

  The device is built and developed in order to improve people health and make community healthy. So, there is not any kind of social requirements.

- Political

  The device doesn't qualify for any political requirements.

- Ethical

  The device is built with authentic materials and security of that data as well as the privacy of the user has been made sure and is one hundred percent authentic design.

- Health and Safety

  The device doesn't emit any kind of radiation or any other harmful things and is totally safe for any age group of users to use it.

- Legality

  The legal requirement would be making user sign user agreement because predicting this disease can sometime lead to complexities. By making a user agreement will make sure that user understand what kind of device is being used and the purpose of it.

- Inspectability

  The model is easily inspectable if the user runs into any kind of bugs. As the program is written in such a way that we can easily know what it malfunctioning.

4.2.3. System Requirements

For system requirements there are two major categories i.e., hardware that has been used and software that helps to hold it all together. The hardware and software integrated for proposed system are:

4.2.3.1. H/W Requirements

- Arduino Uno

  The Arduino Uno was used as a microprocessor in this project. It communicates with

different sensos with the help of its respective library to sense the data and print it to serial port. Arduino can quickly supply using a USB cable for any device, whether a laptop or computer. It has can also be provided from the Ac power joining to a barrel jack at the back. The reason to have a voltage regulator is to maintain the stable dc voltage throughout the circuit. Arduino calculates time using a crystal oscillator. The frequency is Arduino uno board is 16MHZ. The resetting of uno can be done using Arduino reset button in it or by providing it externally.

The output voltage pins that are available are 3.3v and 5v. There are three ground pins in Arduino. The Vin pin in Arduino is to supply power externally from an AC source. The six analog pins are available from A0 to A5. These pins can read from the sensor or any other components. Arduino Uno has its microprocessor to process the data from parts and then pass it to the serial port.

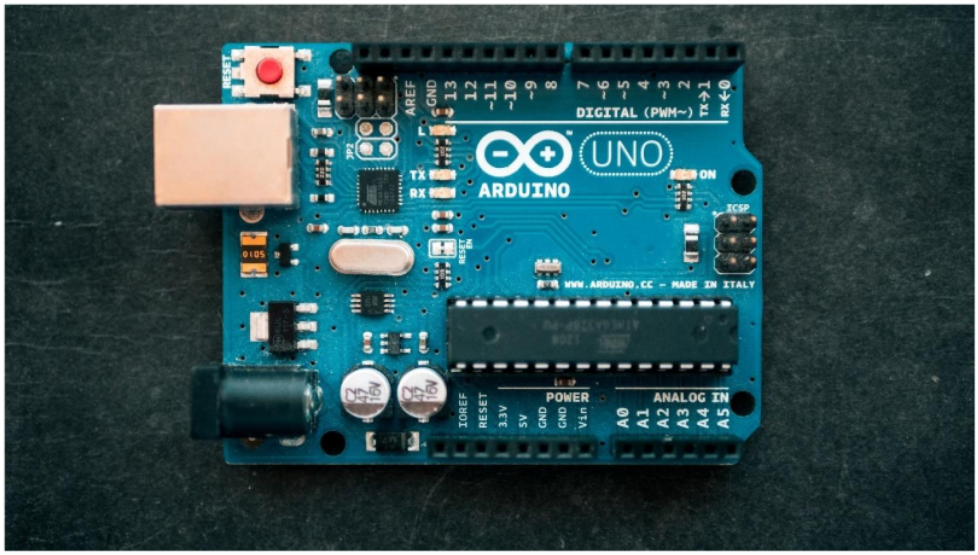


Fig 4.2. Arduino uno

Similarly, there are 14 I/O digital pins to communicate with the sensor. The LEDs indicate to the user that it is receiving or transmitting data or checking whether the Arduino is powered or not. Finally, there are other components as well, which are going to be equally crucial while using it in the system.

- Neo-GM GPS module

  This module is used to get the live location of the user. To know the working of this module, we first learn about GPS; GPS is a global positioning system made up of 24 satellites or more used for navigation purposes without any subscription charges. We need the 2D coordinates i.e., latitude, and longitude. The GPS receiver must be locked into at least three satellites. This process is to be known as triangulation.

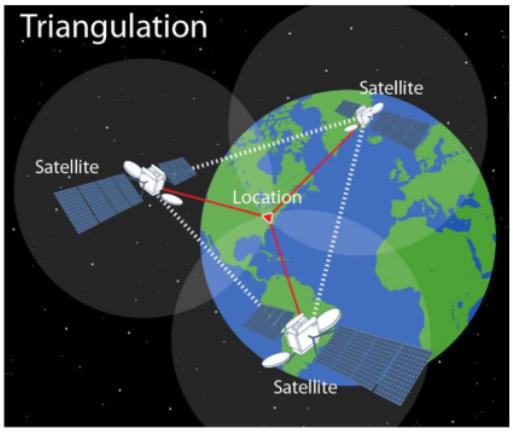


Fig 4.3. GPS Triangulation

The transmitted GPS signal is a low-power radio signal. The signal can easily travel like light, meaning that it will pass through plastic, glass, water, and clouds but will not go through opaque and solid bodies, such as mountains, houses, buildings, etc. A GPS transmitted signal carries three different types of data, i.e., Pseudorandom code, Ephemeris, Almanac.

Fig 4.4. GPS picture

Coming onto neo gm GPS module, it consists of 4 pins, i.e., Vin, Tx, Rx, and ground. In addition, the Vin is supplied with 3.3v, Tx, Rx with respective pins of Arduino and zero voltage with the ground pin. After this, GPS NMEA data is transmitted to the Arduino, which can then be parsed with the help of a tiny GPS library.

- DH11 Temperature and Humidity sensor

  This sensor is to measure temperature from the patient body. There are wide range of sensors that are available. This project uses DHT11 because it is the most robust temperature sensor available. Moreover, this sensor can easily interface between wide ranges of microcontrollers like Arduino Raspberry PI and instantly read the temperature and humidity. We don't need humidity reading but can help if required in the future. The module has got pull-up register and power LED that indicates whether the sensor is powered on or not.

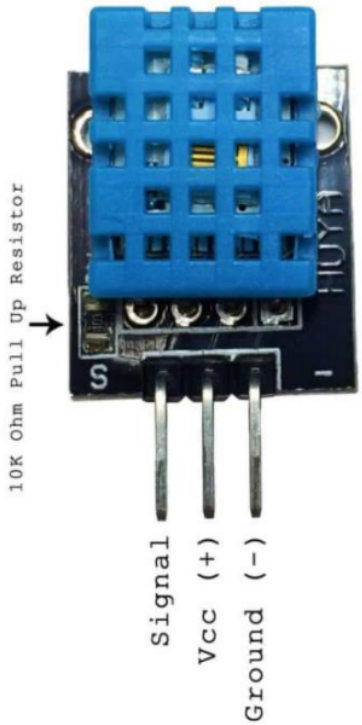


Fig 4.5. DHT11

The DHT11 Temperature sensor, it consists of 4 pins, i.e., Vin, data, and ground. In addition, the Vin is supplied with 5v, data pin with respective digital pins of Arduino and zero voltage with the ground pin. The humidity value is not necessarily required maybe in future that can be use of proper research is conducted for the relation between humidity and health conditions.

- MAX30100 Pulse Oximeter

This sensor helps the user to get the heartbeat and blood oxygen saturation level from the patient. This module is mostly user for wearable is the reason I found it most suitable to use it my device.

There is design error in the system in which there are three 4.7k ohms which register which will not let the main sensor to get the required value. Hence, we need to remove those sensors and pull out externally it externally. This will ensure that the main sensor get the accurate voltage and then provide data to the microprocessor.

Fig 4.6. MAX30100 sensor.

The MAX30100 pulse oximeter, it consists of 8 pins, i.e., Vin, SDA, SCL, INT, RD, IRD and ground. In addition, the Vin is supplied with 3.3v, SCL and SCL with the respective default I2C pins of Arduino uno i.e. A4 and A5 respectively and interrupt pin is connected to the default int pin i.e., either 2 or 3 in Arduino uno and zero voltage with the ground pin.

- Breadboard

  A breadboard is a simple prototyping construction base. The main reason used widely is because it is reusable. After all, it does not require the soldering of wires which makes it easier experimenting on the circuit.

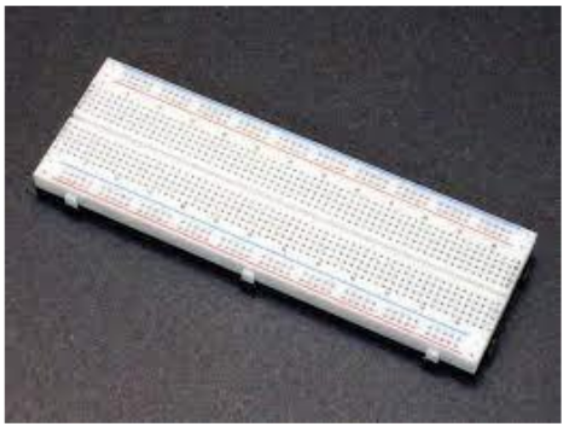


Fig 4.7. Breadboard

- Connecting wires

  The jumper wires used for the project is divided into three categories Male- to – Male, Female- to –Male, and finally Female- to –Female.

Fig 4.8. Connecting wires

4.2.3.2. S/W Requirements

- Arduino IDE with required libraries

  The Arduino IDE is official software used for integrated development. It mainly consists of a navigation bar comprising file, sketch, edit, tools and helps section. The code is written IDE in a text editor, stored into extension name. ino format. Finally, the program compiled to the language understood by Microprocessor. Then can be uploaded accordingly; use the option to upload in the navigation bar. After the code uploads, the Microprocessor starts to give the desired output. Other functionality like adding import libraries used in this project.

- Jupyter Notebook.

  The Jupyter Notebook is a simple easy to use web application used to create, store and share documents. The Jupyter name mainly consists of the language core that is supported. It supports Python (that we have used in this system), Julia, and R. This software is used to train and predict the Machine learning algorithm. The machine-learning algorithm that we are using is Random Forest.

- ThingSpeak IoT

  ThingSpeak IoT is an instant data storage, processing and data visualization platform. It has a straightforward interface that makes it easy. It is also clear to test the prototypes because ThingSpeak IoT is chosen for clod storage. Therefore, the channel where the data is uploaded and visualized is readily available to be created. Then, we are given an API key that acts as an access token to send a request to the channel. Finally, the data is embedded into the link using various fields defined in the dashboard. Now we can send a get request to store that data into the cloud. The information then can be accessed anywhere using the API link. The data accessed from the link is fetched in the form of a JSON file.

- Node Js server development and NPM libraries

  Node Js is a backend development environment built on Js applicable to make networks fast and scalable. Moreover, it is an event-driven system, non-blocking input and output model, making them lightweight and efficient. So, these are the reason which I have chosen Node Js for these specific reasons. The environment used in our system is for serial port server and messaging server user for a Chat application. Please note to have specific libraries version mentioned in the package.json file in this project.

- Browser

  The browser is used for chat application and ThingSpeak IoT visualization. The browser used should be of greater than the specified version because socket programing library socket.io is not really supported for the version before than that.

5. REULTS AND DISCUSION

1. Data collection:

   The first section of the designed model in which we collected the reading from the patient using the device developed.

- The output from this section can be looked into the data that we got from microprocessor using temperature sensor and neo gm GPS module and check the accuracy of the sensors.

```
COM4

Position: www.google.com/maps?q=loc:27.675981,85.392509
Current Humidity = 95.00% Temperature = 30.00C
Position: www.google.com/maps?q=loc:27.675981,85.392509
Current Humidity = 95.00% Temperature = 30.00C
Position: www.google.com/maps?q=loc:27.675983,85.392509
Current Humidity = 95.00% Temperature = 30.00C
Position: www.google.com/maps?q=loc:27.675983,85.392509
Current Humidity = 95.00% Temperature = 30.00C
Position: www.google.com/maps?q=loc:27.675983,85.392509
Current Humidity = 95.00% Temperature = 30.00C
Position: www.google.com/maps?q=loc:27.675983,85.392509
Current Humidity = 95.00% Temperature = 30.00C
Position: www.google.com/maps?q=loc:27.675983,85.392509
Current Humidity = 95.00% Temperature = 30.00C
Position: www.google.com/maps?q=loc:27.675983,85.392509
Current Humidity = 95.00% Temperature = 30.00C
Position: www.google.com/maps?q=loc:27.675983,85.392509
Current Humidity = 95.00% Temperature = 30.00C
Position: www.google.com/maps?q=loc:27.675983,85.392509
```

Fig 5.1. Data collection microprocessor output

- Here we are getting patient body temperature and position from the device. You can see that the body temperature is not so accurate because the user hasn't yet placed the finger on the temperature sensor. So, the out is actually coming from the environment. Now user

can take the link and then paste it in the google chrome to check the validity of the GPS module.

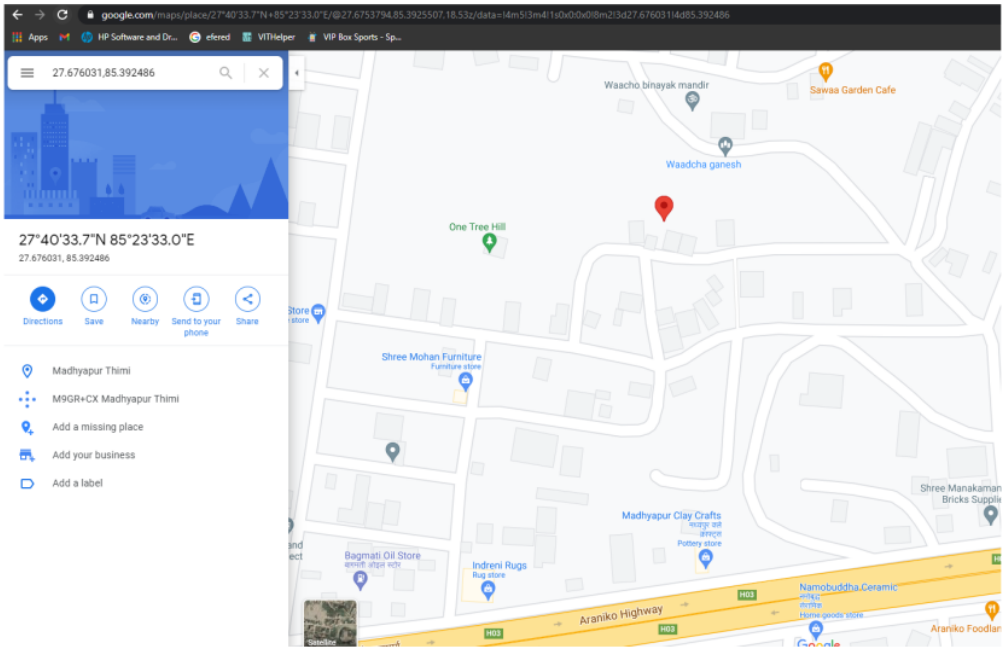
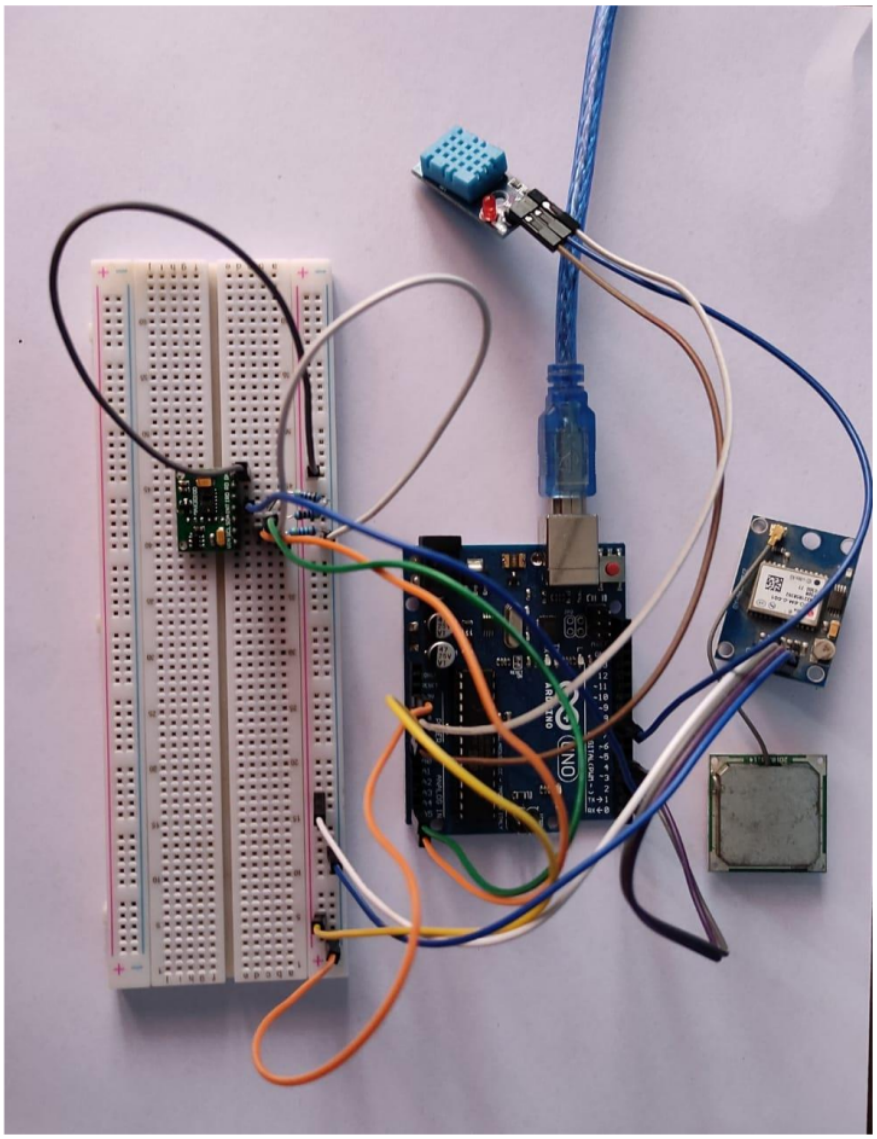


Fig 5.2. Verification of live location using microprocessor.

- The result can be easily verified from the google maps using the link that was generated from the Arduino serial monitor.

- The picture below shows the actual implementation of the project.



` Fig 5.3. Actual Circuit Diagram.

- Now get output from all three sensor as a string in serial monitor:


```
COM5
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 38.00, "Heartbeat": 118.98, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 37.00, "temperature": 37.00, "Heartbeat": 92.55, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 37.00, "temperature": 36.00, "Heartbeat": 84.02, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 38.00, "Heartbeat": 85.46, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 36.00, "Heartbeat": 87.49, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 36.00, "Heartbeat": 85.38, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 38.00, "Heartbeat": 90.20, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 38.00, "Heartbeat": 96.30, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 38.00, "Heartbeat": 90.32, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 37.00, "Heartbeat": 90.32, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 37.00, "Heartbeat": 77.68, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 37.00, "Heartbeat": 94.70, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 37.00, "Heartbeat": 66.78, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 37.00, "Heartbeat": 90.81, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 37.00, "Heartbeat": 102.41, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 37.00, "Heartbeat": 97.81, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 37.00, "Heartbeat": 95.42, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 37.00, "Heartbeat": 94.65, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 38.00, "Heartbeat": 97.89, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 37.00, "Heartbeat": 88.19, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 38.00, "temperature": 37.00, "Heartbeat": 88.19, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 37.00, "Heartbeat": 74.22, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 36.00, "Heartbeat": 74.22, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 36.00, "Heartbeat": 80.61, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 36.00, "Heartbeat": 96.11, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 36.00, "Heartbeat": 102.54, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 36.00, "Heartbeat": 101.52, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 37.00, "Heartbeat": 99.68, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 36.00, "Heartbeat": 92.57, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 37.00, "Heartbeat": 173.65, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 37.00, "Heartbeat": 113.10, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 39.00, "temperature": 37.00, "Heartbeat": 109.93, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 40.00, "temperature": 38.00, "Heartbeat": 96.29, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 40.00, "temperature": 37.00, "Heartbeat": 94.84, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 40.00, "temperature": 37.00, "Heartbeat": 89.66, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 40.00, "temperature": 37.00, "Heartbeat": 93.89, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 40.00, "temperature": 37.00, "Heartbeat": 105.35, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 40.00, "temperature": 37.00, "Heartbeat": 104.31, "Spotwo": 96}
{"latitude": 27.68, "longitude": 85.39, "humidity": 40.00, "temperature": 37.00, "Heartbeat": 98.74, "Spotwo": 93}
Autoscroll  Show timestamp
```


Fig 5.4. Output From Microprocessor.

- Finally, the node Js server read the data from the serial port and displays in the console before sending it to the ThingSpeak IoT. There are Three outputs in the terminal the first one in curly bracket is actually the string that server reads from the serial port. The other one is the JSON data printed after the data is parsed from string.

```
{"latitude": 27.68, "longitude": 85.39, "humidity": 43.00, "temperature": 38.00, "Heartbeat": 79.96, "Spotwo": 94}
{
  latitude: 27.68,
  longitude: 85.39,
  humidity: 43,
  temperature: 38,
  Heartbeat: 79.96,
  Spotwo: 94
}
{"latitude": 27.68, "longitude": 85.39, "humidity": 43.00, "temperature": 38.00, "Heartbeat": 48.80, "Spotwo": 94}
{
  latitude: 27.68,
  longitude: 85.39,
  humidity: 43,
  temperature: 38,
  Heartbeat: 48.8,
  Spotwo: 94
}
{"latitude": 27.68, "longitude": 85.39, "humidity": 41.00, "temperature": 37.00, "Heartbeat": 120.33, "Spotwo": 94}
{
  latitude: 27.68,
  longitude: 85.39,
  humidity: 41,
  temperature: 37,
  Heartbeat: 120.33,
  Spotwo: 94
}
{"latitude": 27.68, "longitude": 85.39, "humidity": 41.00, "temperature": 37.00, "Heartbeat": 93.68, "Spotwo": 94}
{
  latitude: 27.68,
  longitude: 85.39,
  humidity: 41,
  temperature: 37,
  Heartbeat: 93.68,
  Spotwo: 94
}
{"latitude": 27.68, "longitude": 85.39, "humidity": 42.00, "temperature": 37.00, "Heartbeat": 61.62, "Spotwo": 94}
{
  latitude: 27.68,
  longitude: 85.39,
  humidity: 42,
  temperature: 37,
  Heartbeat: 61.62,
  Spotwo: 94
}
{"latitude": 27.68, "longitude": 85.39, "humidity": 43.00, "temperature": 38.00, "Heartbeat": 155.13, "Spotwo": 94}
{
  latitude: 27.68,
  longitude: 85.39,
  humidity: 43,
  temperature: 38,
  Heartbeat: 155.13,
  Spotwo: 94
```

Fig 5.5. Result data received in server.

```javascript
var SerialPort = require('serialport')
const Readline = SerialPort.parsers.Readline;
const socketio = require('socket.io')


const app =express()
const server = http.createServer(app)
const io = socketio(server)

const port2 = process.env.PORT || 3000
const publicDirectoryPath=  path.join(__dirname, '../public')
app.use(express.static(publicDirectoryPath))

var port = new SerialPort("COM5", {
    baudRate: 9600,
});

const parser = new Readline();
port.pipe(parser);


function tryParseJson (str) {
    try {
        JSON.parse(str);
    } catch (e) {
        return false;
    }
    return JSON.parse(str);
}

console.log('Initialising...');
console.log('Opened port ....');
parser.on('data', function(data) {
    console.log(data);
    const sensorData = tryParseJson(data);
    if(sensorData){
        console.log(sensorData);


    let baseurl = 'https://api.thingspeak.com/update?api_key=4772VX96H8HICDU7&field1='+sensorData.temperature+

        https.get(baseurl, (res) => {  }); }
        });

 server.listen(port2, () => {
    console.log(`Server is up on port ${port2}`)
})
```

Fig 5.6. serial port server.

2. Data Storage and Visualization:

   The data will be received by the Thing Speak IoT server to store it and then visualize it. This cloud storage gives us different option like visualizing the into graph, step line, bar, column, step etc.

- The picture show how data is stored in the database of ThingSpeak IoT.

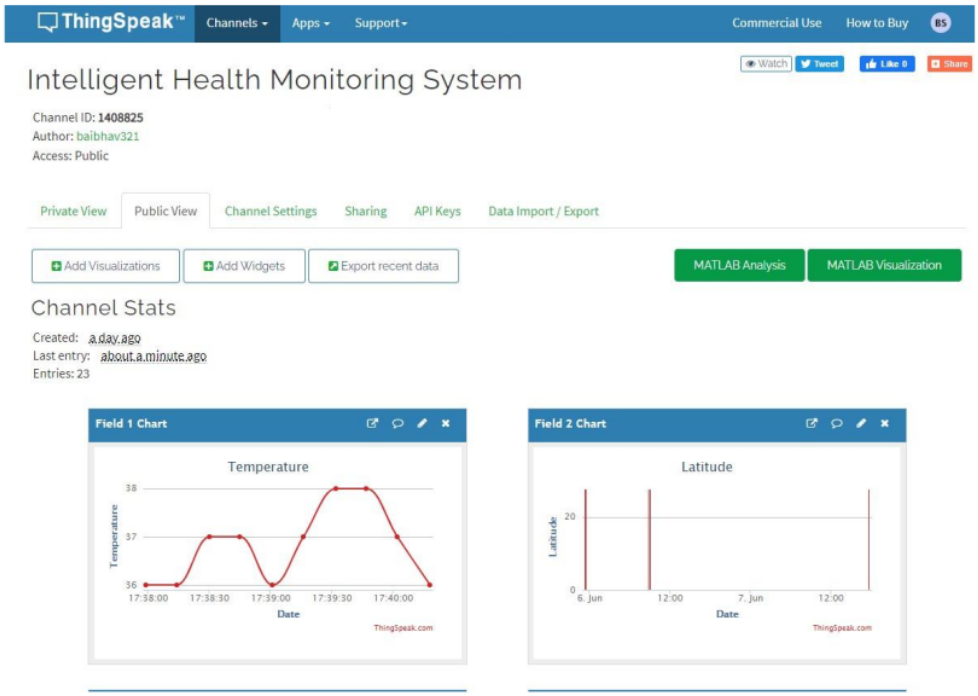
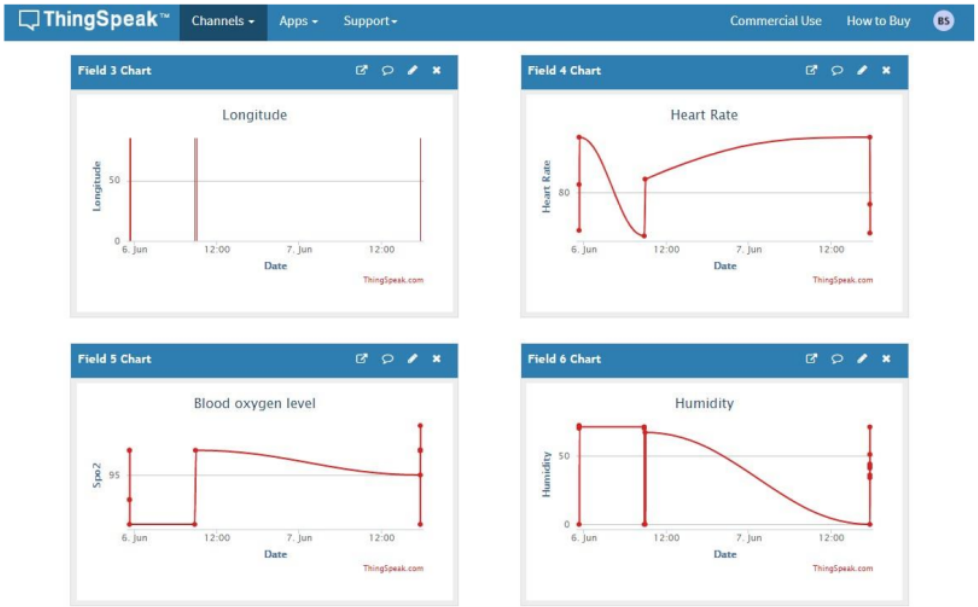


Fig 5.7. Result data stored and Visualized at Thing Speak IoT.

- Here the temperature data and longitude has been visualized. Sometime the data is stored as 0 is when microprocess is initializing the data was send to ThingSpeak IoT



Result data stored and Visualized at Thing Speak IoT.

- In this figure there are four values displayed that is displayed in the channel. i.e., longitude, latitude, blood oxygen level, humidity. We don't really require humidity but is stored for the future references.

- As you can the data is available in ThingSpeak cloud database we can export it in two ways:

  One as shown below in csv format:

	A	B	C	D	E	F	G	H	
1	created_a	entry_id	field1	field2	field3	field4	field5	field6	la
2	2021-06-0	2	27	27.67	85.39	69.44	94	72	
3	2021-06-0	3	27	27.67	85.39	82.33	96	71	
4	2021-06-0	4	28	27.67	85.39	186.83	94	72	
5	2021-06-0	5	27	27.67	85.39	95.62	96	70	
6	2021-06-0	6	28	27.67	85.39	59.1	93	71	
7	2021-06-0	7	29	27.68	85.39	67.87	93	71	
8	2021-06-0	10	28	27.68	85.39	38.88	0	70	
9	2021-06-0	12	29	27.68	85.39	83.83	96	67	
10	2021-06-0	14	36	27.68	85.39	103.63	95	42	
11	2021-06-0	15	36	27.68	85.39	95.61	96	36	
12	2021-06-0	16	37	27.68	85.39	123.86	96	34	
13	2021-06-0	17	37	27.68	85.39	110.79	93	71	
14	2021-06-0	19	37	27.68	85.39	101.19	96	41	
15	2021-06-0	20	38	27.68	85.39	145.09	96	42	
16	2021-06-0	21	38	27.68	85.39	178.03	96	43	
17	2021-06-0	22	37	27.68	85.39	68.68	96	44	
18	2021-06-0	23	36	27.68	85.39	76.78	97	43	
19									

Fig 5.8. Result data .csv exported from Thing Speak IoT.

- The other way to use the API data link to get file in json form


```
    "field1": "37",
    "field2": "27.68",
    "field3": "85.39",
    "field4": "123.86",
    "field5": "96",
    "field6": "34"
  },
  {
    "created_at": "2021-06-07T12:08:45Z",
    "entry_id": 17,
    "field1": "37",
    "field2": "27.68",
    "field3": "85.39",
    "field4": "110.79",
    "field5": "93",
    "field6": "71"
  },
  {
    "created_at": "2021-06-07T12:09:01Z",
    "entry_id": 18,
    "field1": "36",
    "field2": "27.68",
    "field3": "85.39",
    "field4": "0",
    "field5": "0",
    "field6": "51"
  },
  {
    "created_at": "2021-06-07T12:09:16Z",
    "entry_id": 19,
    "field1": "37",
    "field2": "27.68",
    "field3": "85.39",
    "field4": "101.19",
    "field5": "96",
    "field6": "41"
  },
  {
    "created_at": "2021-06-07T12:09:32Z",
    "entry_id": 20,
    "field1": "38",
    "field2": "27.68",
    "field3": "85.39",
    "field4": "145.09",
    "field5": "96",
    "field6": "42"
  },
  {
    "created_at": "2021-06-07T12:09:47Z",
    "entry_id": 21,
    "field1": "38",
    "field2": "27.68",
```


Result data. json exported from Thing Speak IoT.

This the data we get from cloud but this will not be enough for Machine learning now so we use now use reliable available data from the internet. So, I have taken data that was provided by Israeli government on internet.

3. Analysis of data

The data is analyzed using machine learning using Random Forest algorithm but before that we will have to import the data we created.

- To do analysis on the data received we will have to import all the dependencies and library that are required. In Line 2 we have read CSV file which was taken by Israeli government. The data then was translated to English and then import to our Machine Learning system.

1. Import the libraries

```
In [1]: import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import confusion_matrix
        #from imblearn.under_sampling import OneSidedSelection
        from sklearn.model_selection import StratifiedKFold
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import precision_score, matthews_corrcoef, accuracy_score
        import seaborn as sns

In [2]: pd.set_option('display.max_columns', None)
        pd.set_option('display.max_rows', None)
```

2. Read the csv

```
In [3]: #To remove diffrent data type df as data from.
        df = pd.read_csv('corona_sample.csv', error_bad_lines=False, index_col=False, dtype='unicode')
```

a. look at the shape of the data

```
In [4]: # To list number the rows and cloumn
        df.shape
Out[4]: (278848, 10)
```

b. look at the data

```
In [5]: # to list first 20 data from the csv file.
        df.head(20)
Out[5]:
```

Fig 5.9. Machine Learning Model

- In line 4 we can see that the data is huge has almost reading from 278848 people. Hence after using this data, we will get more accurate results.

```
In [5]: # to list first 20 data from the csv file.
        df.head(20)
```

Out[5]:

	test_date	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gender	test_indication
0	2020-04-30	0	0	0	0	0	negative	None	female	Other
1	2020-04-30	1	0	0	0	0	negative	None	female	Other
2	2020-04-30	0	1	0	0	0	negative	None	male	Other
3	2020-04-30	1	0	0	0	0	negative	None	female	Other
4	2020-04-30	1	0	0	0	0	negative	None	male	Other
5	2020-04-30	1	0	0	0	0	negative	None	female	Other
6	2020-04-30	1	1	0	0	0	negative	None	male	Abroad
7	2020-04-30	0	0	0	0	0	negative	None	female	Other
8	2020-04-30	0	0	0	0	0	negative	None	male	Other
9	2020-04-30	0	0	0	0	0	negative	None	male	Contact with confirmed
10	2020-04-30	1	1	0	0	0	negative	None	female	Other
11	2020-04-30	0	0	0	0	0	negative	None	female	Other
12	2020-04-30	0	0	0	0	0	negative	None	female	Other
13	2020-04-30	0	0	0	0	0	negative	None	female	Other
14	2020-04-30	0	0	0	0	0	negative	None	male	Other
15	2020-04-30	1	0	0	0	0	negative	None	male	Other
16	2020-04-30	0	0	0	0	0	negative	None	female	Abroad
17	2020-04-30	1	1	0	0	0	negative	None	male	Abroad
18	2020-04-30	1	0	0	0	0	negative	None	female	Other
19	2020-04-30	1	0	0	0	0	negative	None	male	Abroad

c. feature engineering on the columns

```
In [6]: # none means it is not 0 or 1
        df['cough'].value_counts()
```

```
In [6]: # none means it is not 0 or 1
        df['cough'].value_counts()

Out[6]: 0       236368
        1        42228
        None       252
        Name: cough, dtype: int64

In [7]: df['fever'].value_counts()

Out[7]: 0       256844
        1        21752
        None       252
        Name: fever, dtype: int64

In [8]: df['sore_throat'].value_counts()

Out[8]: 0       276921
        1         1926
        None         1
        Name: sore_throat, dtype: int64

In [9]: df['shortness_of_breath'].value_counts()

Out[9]: 0       277270
        1         1577
        None         1
        Name: shortness_of_breath, dtype: int64

In [10]: df['head_ache'].value_counts()

Out[10]: 0       276433
         1         2414
         None         1
         Name: head_ache, dtype: int64

In [11]: df['corona_result'].value_counts()

Out[11]: negative    260227
         positive     14729
         other         3892
         Name: corona_result, dtype: int64
```

Listing of data in Machine Learning Model

In [12]:
```
df['age_60_and_above'].value_counts()
```

Out[12]:
```
None    127320
No      125703
Yes      25825
Name: age_60_and_above, dtype: int64
```

In [13]:
```
df['gender'].value_counts()
```

Out[13]:
```
female    130158
male      129127
None       19563
Name: gender, dtype: int64
```

In [14]:
```
df['test_indication'].value_counts()
```

Out[14]:
```
Other                     242741
Abroad                     25468
Contact with confirmed     10639
Name: test_indication, dtype: int64
```

In [15]:
```
# remove rows that are not useful other than gender

def remove_unnecessary(df):
    df = df.loc[df.cough != 'None']
    df = df.loc[df.fever != 'None']
    df = df.loc[df.sore_throat != 'None']
    df = df.loc[df.shortness_of_breath != 'None']
    df = df.loc[df.head_ache != 'None']
    df = df.loc[df.age_60_and_above != 'None']
    df = df.loc[df.corona_result != 'other']
    df = df.drop('test_date', axis=1)
    return df

df = remove_unnecessary(df)
```

In [16]:
```
# checking the shape again default 5

df.head()
```

In [16]:
```
# checking the shape again default 5

df.head()
```

Out[16]:

	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above	gender	test_indication
122808	1	0	0	0	0	negative	Yes	male	Other
122809	1	0	0	0	0	positive	No	female	Other
122810	0	0	0	0	0	negative	No	female	Other
122811	0	1	0	0	0	negative	No	female	Abroad
122812	1	0	0	0	0	negative	Yes	female	Other

In [17]:
```
# converting the categorical variables to 1/0s

ls =['age_60_and_above', 'gender', 'test_indication']

df = pd.get_dummies(df, columns=ls)
```

In [18]:
```
df.head()
```

Out[18]:

	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above_No	age_60_and_above_Yes	gender_None	gender_female
122808	1	0	0	0	0	negative	0	1	0	0
122809	1	0	0	0	0	positive	1	0	0	1
122810	0	0	0	0	0	negative	1	0	0	1
122811	0	1	0	0	0	negative	1	0	0	1
122812	1	0	0	0	0	negative	0	1	0	1

In [19]:
```
# set the y-value
def cat(row):
    if row['corona_result'] == 'negative':
        val = 0
    else:
        val = 1
    return val

df['covid'] = df.apply(cat, axis=1)
```

In [20]:
```
df.head()
```

Out[20]:

Out[20]:

	cough	fever	sore_throat	shortness_of_breath	head_ache	corona_result	age_60_and_above_No	age_60_and_above_Yes	gender_None	gender_female
122808	1	0	0	0	0	negative	0	1	0	0
122809	1	0	0	0	0	positive	1	0	0	1
122810	0	0	0	0	0	negative	1	0	0	1
122811	0	1	0	0	0	negative	1	0	0	1
122812	1	0	0	0	0	negative	0	1	0	1

In [21]:
```
df = df.drop('corona_result', axis=1)
```

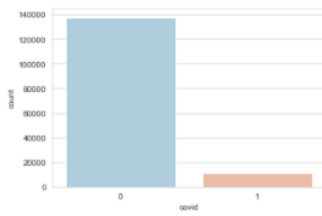
In [22]:
```
df.head()
```

Out[22]:

	cough	fever	sore_throat	shortness_of_breath	head_ache	age_60_and_above_No	age_60_and_above_Yes	gender_None	gender_female	gender_male
122808	1	0	0	0	0	0	1	0	0	1
122809	1	0	0	0	0	1	0	0	1	0
122810	0	0	0	0	0	1	0	0	1	0
122811	0	1	0	0	0	1	0	0	1	0
122812	1	0	0	0	0	0	1	0	1	0

In [23]:
```
# Obeserve the 1 and 0 values

sns.set_style('whitegrid')
sns.countplot(x='covid',data=df,palette="RdBu_r")
```

Out[23]: <AxesSubplot:xlabel='covid', ylabel='count'>

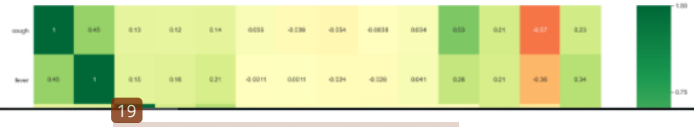


In [24]:
```
df.dtypes
```

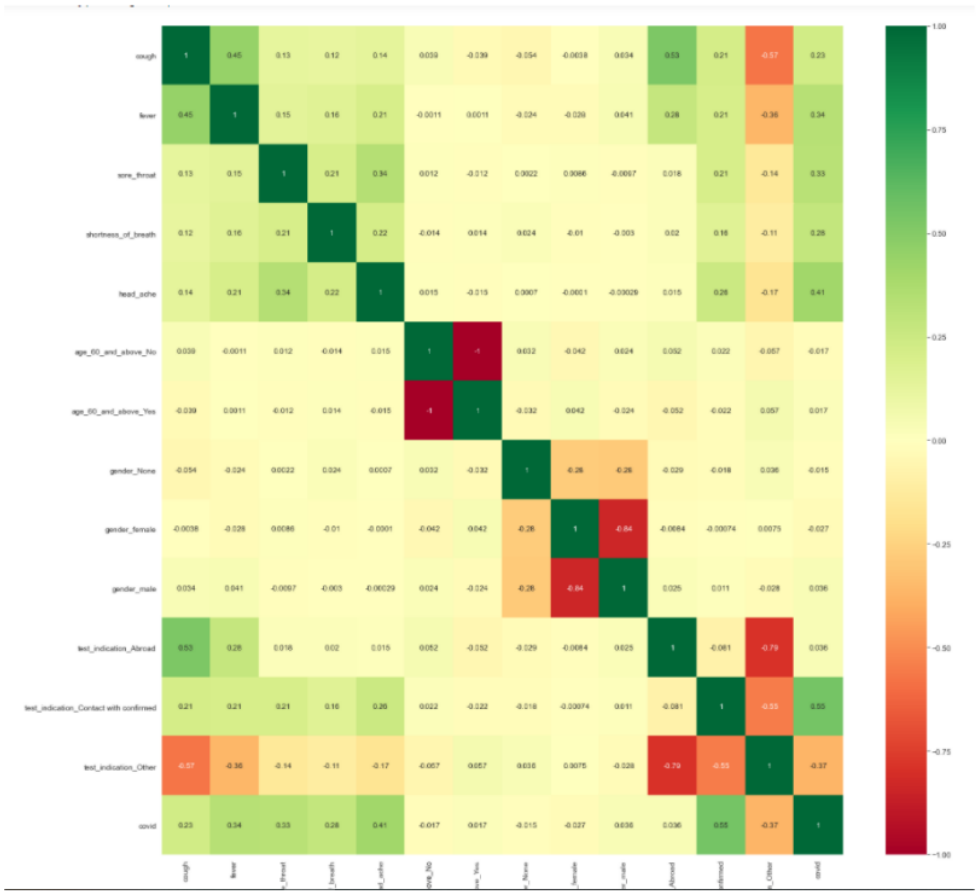
Out[24]:
```
cough                                  object
fever                                  object
sore_throat                            object
shortness_of_breath                    object
head_ache                              object
age_60_and_above_No                     uint8
age_60_and_above_Yes                    uint8
gender_None                             uint8
gender_female                           uint8
gender_male                             uint8
test_indication_Abroad                  uint8
test_indication_Contact with confirmed  uint8
test_indication_Other                   uint8
covid                                   int64
dtype: object
```

In [25]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline

# converting into int beacuse .corr function only works with integer value
df = df.astype(int)

corrmat = df.corr()
# assigns the name to diagram
top_corr_features = corrmat.index
print(top_corr_features)
plt.figure(figsize=(20,20))
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```

```
Index(['cough', 'fever', 'sore_throat', 'shortness_of_breath', 'head_ache',
       'age_60_and_above_No', 'age_60_and_above_Yes', 'gender_None',
       'gender_female', 'gender_male', 'test_indication_Abroad',
       'test_indication_Contact with confirmed', 'test_indication_Other',
       'covid'],
      dtype='object')
```



Co-relation between data in Machine Learning Model


cough
fever
sore_throat
shortness_of_breath
head_ache
age_60_and_above_No
age_60_and_above_Yes
gender_None
gender_female
gender_male
test_indication_Abroad
test_indication_Contact with confirmed
test_indication_Other
covid
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

3. Machine Learning Model

In [26]:
```
# Assigning y as covid column
y = df['covid'].values

X = df.drop(columns=['covid'])
X.shape
```

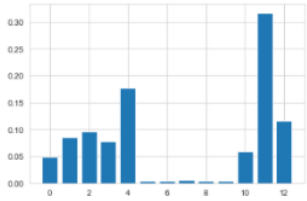
Out[26]: (149043, 13)

In [27]:
```
def sratified_k_fold(X, y):
    skf = StratifiedKFold(n_splits=10)
    # StratifiedKFold(n_splits=10, random_state=None, shuffle=False)
    for train_index, test_index in skf.split(X, y):
        X_train, X_test, y_train, y_test = train_test_split(X,
                                                            y,
                                                            test_size=0.30,
                                                            random_state=0)

        # define the undersampling method
        #undersample = OneSidedSelection(n_neighbors=1, n_seeds_S=200)
        # transform the dataset
        #X_train, y_train = undersample.fit_resample(X_train, y_train)


        #### Gradient boost classifier
        clf = RandomForestClassifier()
        clf.fit(X_train, y_train)
        importance = clf.feature_importances_
        X_columns = X_train.columns
        importances = list(zip(X_columns, clf.feature_importances_))
        for x in importances:
            print(x[0],':', x[1])
        plt.bar([x for x in range(len(importance))], importance)
        plt.show()
        y_pred = clf.predict(X_test)
        tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
        print('tn: {}\nfp: {}\nfn: {}\ntp: {}'.format(tn,fp,fn,tp))
        print(matthews_corrcoef(y_test, y_pred), " MCC")
        print(precision_score(y_test, y_pred, average='binary'), "Precision Score")

sratified_k_fold(X, y)
```

```
cough : 0.04871431634074766
fever : 0.08552756754063735
sore_throat : 0.09652010742212 59
shortness_of_breath : 0.07804523152348344
head_ache : 0.1769866213663118
age_60_and_above_No : 0.004498585584832575
age_60_and_above_Yes : 0.004067171830882313
gender_None : 0.005777677083524849
gender_female : 0.004159317322052 88
gender_male : 0.004018087638392 9705
```

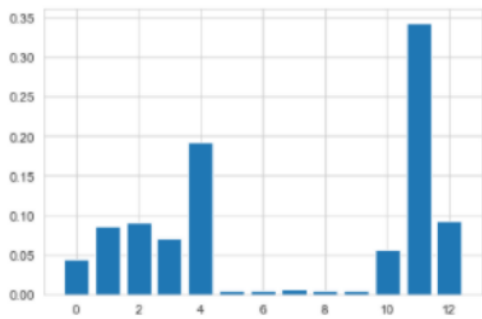
```
sratified_k_fold(X, y)
```

```
cough : 0.048714316340747 66
fever : 0.08552756754063735
sore_throat : 0.0965201074221259
shortness_of_breath : 0.07804523152348344
head_ache : 0.1769866213663118
age_60_and_above_No : 0.004498585584832575
age_60_and_above_Yes : 0.004067171830882313
gender_None : 0.005777677083524849
gender_female : 0.00415931732205288
gender_male : 0.0040180876383929705
test_indication_Abroad : 0.059349458935524575
test_indication_Contact with confirmed : 0.31682816982831796
test_indication_Other : 0.11550768758316575
```




```
tn: 40771
fp: 445
fn: 1449
tp: 2048
0.672687803393204  MCC
0.8215002005615724 Precision Score
cough : 0.04816312517591347
fever : 0.08758438390017953
sore_throat : 0.09682388567156773
shortness_of_breath : 0.06628482177342276
head_ache : 0.1898582876672854
age_60_and_above_No : 0.00432313850915848
age_60_and_above_Yes : 0.004208479889817119 5
gender_None : 0.005322652978100713
gender_female : 0.004745777724046222
gender_male : 0.004303097748550033
test_indication_Abroad : 0.05075559571531146
test_indication_Contact with confirmed : 0.33302161106167777
test_indication_Other : 0.10460514218496907
```

```
tn: 40771
fp: 445
fn: 1445
tp: 2052
0.6735181439677091  MCC
0.8217861433720465 Precision Score
cough : 0.044518046034065735
fever : 0.0853090576946867
sore_throat : 0.0906032842690403
shortness_of_breath : 0.07087642575431735
head_ache : 0.19211302008287667
age_60_and_above_No : 0.004041687909834456
age_60_and_above_Yes : 0.00524878578249256
gender_None : 0.0055423866641652305
gender_female : 0.004012200272297468
gender_male : 0.0044349650043107045
test_indication_Abroad : 0.056821318681160964
test_indication_Contact with confirmed : 0.34347059885167447
test_indication_Other : 0.09300822299907757
```




```
tn: 40771
fp: 445
fn: 1447
tp: 2050
0.6731030696041267  MCC
0.8216432865731463 Precision Score
cough : 0.04797172203794406
fever : 0.08774342371312875
sore_throat : 0.09484523055875534
shortness_of_breath : 0.06994958146873159
head_ache : 0.17840246695734516
age_60_and_above_No : 0.003950786205708059
age_60_and_above_Yes : 0.00468186728132085
gender_None : 0.005519413073999778
gender_female : 0.0037870390046891595
gender_male : 0.004545130489860082
test_indication_Abroad : 0.04769359712654217
test_indication_Contact with confirmed : 0.35960875627769656
test_indication_Other : 0.0913009858042785
```

4. Feedback System:

The last part of the model is feedback System consist of chat application. The users can join the chat application using display name which can be taken as patient id or name whatever has been decided before. The login screen looks like the picture that is show below.

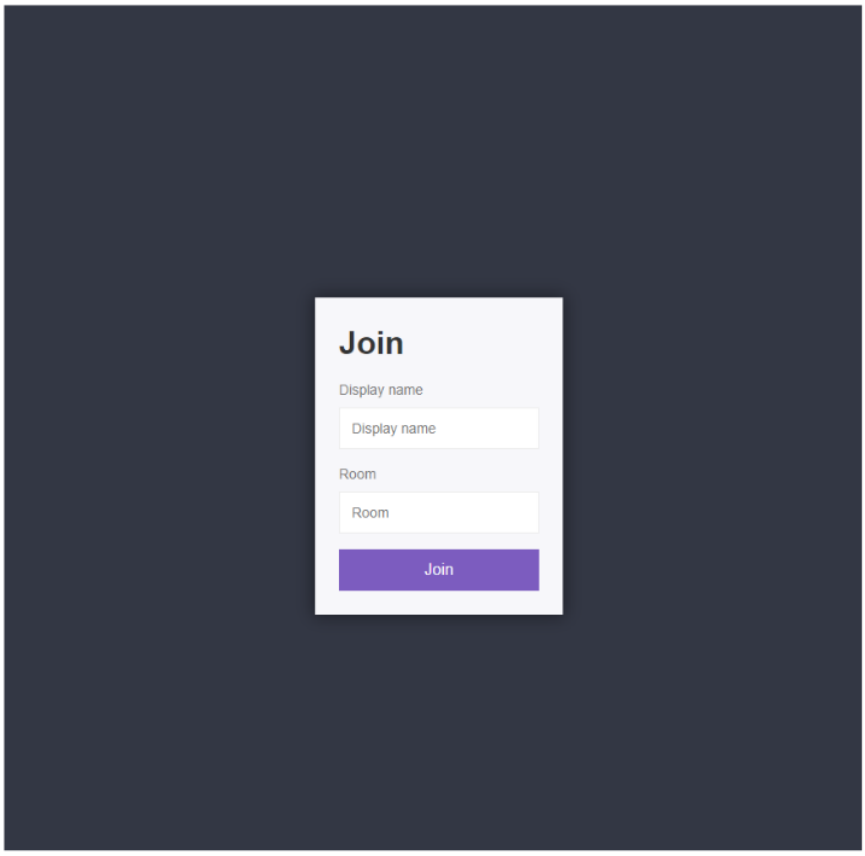
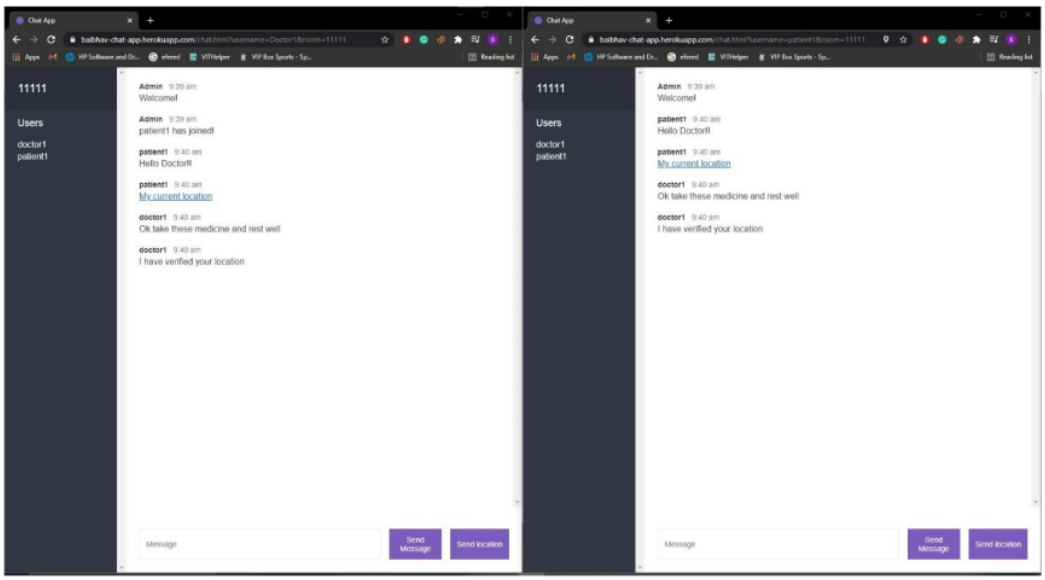


Fig 5.10. Chat Application.

- The user can now join the chat application after opening the chat application into the browser. The user doesn't have to worry about uppercase and lowercase because the system will take care of that issue by lowercasing all the alphabets while joining the chat application.
- This what the system looks like after joining the feedback system. The navigation bar on the right side shows the



Chat Application join screen.

This is end of the model that we have proposed. Hence this is how our system in helps to monitor the health of the patient.

The device was very lost cost to be built that any other medical devices. The overall cost of the device that was required to build was around 2.5k INR which can be easily accommodated in any country's health allocated budget. If the device is taken into mass production, then the cost can be reduced accordingly.

We can get the position of patients easily and will stop other people from getting affected. We also can get his body's temperature & blood oxygen level which is basic criteria. In future this project could be further develop as product. The product in form of watch could help people to be monitored remotely than physically. We could use better sensors than we have used in this project to increase the accuracy of this project. The project could be further developed as product by developing it into watch by designing a PCB which could make it portable like a fitness band. The fitness band could easier for the patient to wear it.

References

[1] Abinayaa, B., Kiruthikamani, G., Saranya, B., & Gayathri, R. An Intelligent Monitoring Device for Asthmatics using Arduino. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, ISSN (Print)*, 2320-3765.

[2] Venema, B., Schiefer, J., Blazek, V., Blanik, N., & Leonhardt, S. (2013). Evaluating innovative in-ear pulse oximetry for unobtrusive cardiovascular and pulmonary monitoring during sleep. *IEEE journal of translational engineering in health and medicine*, *1*, 2700208-2700208.

[3] Kumar, J. D., Babu, C. G., Karthi, S. P., Soundari, D. V., & Priyadharsini, K. (2020). A novel system design for intravenous infusion system monitoring for betterment of health monitoring system using ML-AI. *International Journal of Innovative Technology and Exploring Engineerin*, *9*(3), 2649-2655.

[4] Vashist, S. K., & Luong, J. H. (2018). Wearable Technologies for Personalized Mobile Healthcare Monitoring and Management. *Wearable Technology in Medicine and Health Care*, *235*.

[5] AILENI, R. M., PAŞCA, S., & FLORESCU, A. (2019, March). E-health monitoring by smart pulse oximeter systems integrated in SDU. In *2019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE)* (pp. 1-4). IEEE.

[6] Aliverti, A. (2017). Wearable technology: role in respiratory health and disease. *Breathe*, *13*(2), e27-e36.

[7] He, D., Morgan, S. P., Trachanis, D., Van Hese, J., Drogoudis, D., Fummi, F., ... & Hayes-Gill, B. R. (2015). A single-chip CMOS pulse oximeter with on-chip lock-in detection. *Sensors*, *15*(7), 17076-17088.

[8] Eisenberg, M. E., Givony, D., & Levin, R. (2020). Acoustic respiration rate and pulse oximetry-derived respiration rate: a clinical comparison study. *Journal of*

clinical monitoring and computing, *34*(1), 139-146.

[9] Von Chong, A., Terosiet, M., Histace, A., & Romain, O. (2019). Towards a novel single-LED pulse oximeter based on a multispectral sensor for IoT applications. *Microelectronics Journal*, *88*, 128-136.

[10] Azizulkarim, A. H., Jamil, M. M. A., & Ambar, R. (2017, August). Design and development of patient monitoring system. In *IOP Conference Series: Materials Science and Engineering* (Vol. 226, No. 1, p. 012094). IOP Publishing.

[11] Yassin, F. M., Sani, N. A., & Chin, S. N. (2019, November). Analysis of Heart Rate and Body Temperature from the Wireless Monitoring System Using Arduino. In *Journal of Physics: Conference Series* (Vol. 1358, No. 1, p. 012041). IOP Publishing.

[12] MAX30100 sensor datasheet:
https://datasheets.maximintegrated.com/en/ds/MAX30100.pdf

[13] DHT11 sensor datasheet:
https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

[14] Neo gm GPS datasheet:
https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf

[15] Arduino schematic diagram:
https://www.arduino.cc/en/uploads/Tutorial/595datasheet.pdf

APPENDIX A

ARDUINO CODE:

Final.ino

```
#include <TinyGPS.h>
#include <TimerOne.h>
#include <Wire.h>
#include <dht.h>
#include <SoftwareSerial.h>
#include "MAX30100_PulseOximeter.h"

#define TempHumAnalogIn 8

int RXPin = 4;
int TXPin = 5;
TinyGPS gps;
SoftwareSerial gpsSerialData(RXPin, TXPin);

float lat = 0, lon = 0;

void ServerSendData();
void gpsData();

dht DHT;
PulseOximeter pulseOxydata;


void setup()
{
  Serial.begin(9600);
  gpsSerialData.begin(9600);
  if (!pulseOxydata.begin())
  {
    Serial.println("MAX30100 has failed to initalize");
    while (true);
  }

  Timer1.initialize(1000000);
  Timer1.attachInterrupt(ServerSendData);
}

void gpsData(){
 while (gpsSerialData.available() > 0) {
    if (gps.encode(gpsSerialData.read())) {
      gps.f_get_position(&lat, &lon);
    }
  }

}

void loop()
{
  pulseOxydata.update();
  DHT.read11(TempHumAnalogIn);
  gpsData();

}

void ServerSendData()
```

```
{

  String dataString ="";

  dataString = dataString + "{\"latitude\": ";
  dataString = dataString + String(lat);
  dataString = dataString + ", \"longitude\": ";
  dataString = dataString + String(lon);

  dataString = dataString + ", \"humidity\": ";
  dataString = dataString + String(DHT.humidity);
  dataString = dataString + ", \"temperature\": ";
  dataString = dataString + String(DHT.temperature);

  dataString = dataString + ", \"Heartbeat\": ";
  dataString = dataString + String(pulseOxydata.getHeartRate());
  dataString = dataString + ", \"Spotwo\": ";
  dataString = dataString +  String(pulseOxydata.getSpO2());
  dataString = dataString + "}";


  Serial.println(dataString);

}
```

MACHINE LEANING MODEL:

Intelligent-Health.py

```
### 1. Import the libraries

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
#from imblearn.under_sampling import OneSidedSelection
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score, matthews_corrcoef,
curacy_score
import seaborn as sns

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

### 2. Read the csv

o remove diffrent data type df as data from.
df = pd.read_csv('corona_sample.csv', error_bad_lines=False,
index_col=False, dtype='unicode')

#### a. look at the shape of the data

# To list number the rows and cloumn
df.shape

#### b. look at the data

# to list first 20 data from the csv file.
df.head(20)

#### c. feature engineering on the columns

# none means t is not 0 or 1
df['cough'].value_counts()

df['fever'].value_counts()

df['sore_throat'].value_counts()

df['shortness_of_breath'].value_counts()

df['head_ache'].value_counts()

df['corona_result'].value_counts()

df['age_60_and_above'].value_counts()

df['gender'].value_counts()

df['test_indication'].value_counts()

# remove rows that are not useful other than gender

def remove_unnecessary(df):
```

```
    df = df.loc[df.cough != 'None']
    df = df.loc[df.fever != 'None']
    df = df.loc[df.sore_throat != 'None']
    df = df.loc[df.shortness_of_breath != 'None']
    df = df.loc[df.head_ache != 'None']
    df = df.loc[df.age_60_and_above != 'None']
    df = df.loc[df.corona_result != 'other']
    df = df.drop('test_date', axis=1)
    return df

df = remove_unnecessary(df)

# checking the shape again default 5

df.head()

# converting the categorical variables to 1/0s

ls =['age_60_and_above', 'gender', 'test_indication']

df = pd.get_dummies(df, columns=ls)

df.head()

# set the y-value
def cat(row):
    if row['corona_result'] == 'negative':
        val = 0
    else:
        val = 1
    return val

df['covid'] = df.apply(cat, axis=1)
df.head()

df = df.drop('corona_result', axis=1)

df.head()

# Obeserve the 1 and 0 values

sns.set_style('whitegrid')
sns.countplot(x='covid',data=df,palette="RdBu_r")

df.dtypes
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline

# converting into int beacuse .corr function only works with integer value
df = df.astype(int)

corrmat = df.corr()
# assigns the name to diagram
top_corr_features = corrmat.index
print(top_corr_features)
plt.figure(figsize=(20,20))
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```

```
### 3. Machine Learning Model

# Assigning y as covid column
y = df['covid'].values

X = df.drop(columns=['covid'])
X.shape

def sratified_k_fold(X, y):
    skf = StratifiedKFold(n_splits=10)
    # StratifiedKFold(n_splits=10, random_state=None, shuffle=False)
    for train_index, test_index in skf.split(X, y):
        X_train, X_test, y_train, y_test = train_test_split(X,
                                                            y,

test_size=0.30,

random_state=0)

        # define the undersampling method
        #undersample = OneSidedSelection(n_neighbors=1, n_seeds_S=200)
        # transform the dataset
        #X_train, y_train = undersample.fit_resample(X_train, y_train)


        ## Gradient boost classifier
        clf = RandomForestClassifier()
        clf.fit(X_train, y_train)
        importance = clf.feature_importances_
        X_columns = X_train.columns
        importances = list(zip(X_columns, clf.feature_importances_))
        for x in importances:
          print(x[0],':', x[1])
        plt.bar([x for x in range(len(importance))], importance)
        plt.show()
        y_pred = clf.predict(X_test)
        tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()
        print('tn: {}\nfp: {}\nfn: {}\ntp: {}'.format(tn,fp,fn,tp))
        print(matthews_corrcoef(y_test, y_pred), " MCC")
        print(precision_score(y_test, y_pred, average='binary'),
"Precision Score")

sratified_k_fold(X, y)
```

SERIAL PORT SERVER CODE:

index.js

```
var http = require('http')
const https = require('https');
var SerialPort = require('serialport')
const Readline = SerialPort.parsers.Readline;


const server = http.createServer(appserver)


const port2 = process.env.PORT || 3000


var port3 = new SerialPort("COM5", {
    baudRate: 9600,
});

const parser = new Readline();
port3.pipe(parser);


function ParsetoJson (portData) {
    try {
        JSON.parse(portData);
    } catch (e) {
        return false;
    }
    return JSON.parse(portData);
}

parser.on('data', function(data1) {
    console.log(data1);
    const portData = ParsetoJson(data1);
    if(portData){
        console.log(portData);


    let thingSpeakurl =
'https://api.thingspeak.com/update?api_key=4772VX96H8HICDU7&field1='+portD
ata.temperature+'&field2='+portData.latitude+'&field3='+portData.longitude
+'&field4='+portData.Heartbeat+'&field5='+portData.Spotwo+'&field6='+portD
ata.humidity;

        https.get(thingSpeakurl, (res) => {  }); }
        });

 server.listen(port2, () => {
    console.log(`Server is up on port ${port2}`)
})
```

SERVER CODE AT CHAT APPLICATION:

```
const pathDirect = require('path')
 const http = require('http')
 const express = require('express')
 const socketio = require('socket.io')
 const Filter = require('bad-words')
 const { generateMessage, generateLocationMessage }=
require('./utils/messages')
 const {addUser, removeUser, getUser, getUserInRoom} =
require('./utils/users')

 const appServer =express()
 const server = http.createServer(appServer)
 const inputOutput = socketio(server)

 const serverPort = process.env.PORT || 3000
 const publicPathServer=  pathDirect.join(__dirname, '../public')

 appServer.use(express.static(publicPathServer))


 inputOutput.on('connection', (socket)=>{
     console.log('New Websocket connection')


    socket.on('join', (options, callback)=>{
        const {error, user } = addUser({ id: socket.id, ...options})

        if(error) {
            return callback(error)
        }
        socket.join(user.room)


        socket.emit('message', generateMessage('Admin','Welcome!'))
        socket.broadcast.to(user.room).emit('message',
generateMessage('Admin', `${user.username} has joined!`))
        inputOutput.to(user.room).emit('roomData', {
            room: user.room,
            users: getUserInRoom(user.room)
        })

        callback()
    })

    socket.on('sendMessage', (message, callback) =>{

        const user = getUser(socket.id)
        const filter = new Filter()

        if(filter.isProfane(message)){
            return callback('Profanity is not allowed')
        }

        inputOutput.to(user.room).emit('message',
generateMessage(user.username, message))
        callback()
    })
```

```
    socket.on('sendLocation', (coords, callback)=>{
        const user = getUser(socket.id)
        inputOutput.to(user.room).emit('locationMessage',
generateLocationMessage(user.username,
`https://www.google.com/maps?q=${coords.latitude},${coords.longitude}`))
        callback()
    })

    socket.on('disconnect', ()=>{
        const user = removeUser(socket.id)

        if(user)    {
            inputOutput.to(user.room).emit('message',
generateMessage('Admin',`${user.username} has left!`))
            inputOutput.to(user.room).emit('roomData', {
                room: user.room,
                users: getUserInRoom(user.room)
            })
        }


    })
})

 server.listen(serverPort, () => {
     console.log(`Server is up on serverPort ${serverPort}`)
 })
```

Intelligent Healthcare Monitoring System with Pulse Oximetry Analysis

ORIGINALITY REPORT

14%	12%	4%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to VIT University Student Paper	2%
2	www.coursehero.com Internet Source	2%
3	Submitted to Yogi Vemana University, Vemanapuram Student Paper	1%
4	Submitted to Udayana University Student Paper	1%
5	Submitted to University of Houston, Downtown Student Paper	1%
6	www.ijitee.org Internet Source	1%
7	Submitted to University of Leicester Student Paper	1%
8	Raluca Maria AILENI, Sever PASCA, Adriana FLORESCU. "E-Health Monitoring by Smart	<1%

Pulse Oximeter Systems Integrated in SDU", 2019 11th International Symposium on Advanced Topics in Electrical Engineering (ATEE), 2019
Publication

9	Indira Knight. "Connecting Arduino to the Web", Springer Science and Business Media LLC, 2018 Publication	<1 %
10	medium.com Internet Source	<1 %
11	inblog.in Internet Source	<1 %
12	Submitted to University of Wales, Bangor Student Paper	<1 %
13	indianaiproduction.com Internet Source	<1 %
14	stackoverflow.com Internet Source	<1 %
15	Submitted to Indian Institute of Information Technology, Allahabad Student Paper	<1 %
16	machinelearningmastery.com Internet Source	<1 %
17	towardsdatascience.com Internet Source	<1 %

18	Submitted to University of East London Student Paper	<1 %
19	Manohar Swamynathan. "Mastering Machine Learning with Python in Six Steps", Springer Science and Business Media LLC, 2017 Publication	<1 %
20	share.cocalc.com Internet Source	<1 %
21	uknowledge.uky.edu Internet Source	<1 %
22	B. Venema, J. Schiefer, V. Blazek, N. Blanik, S. Leonhardt. "Evaluating Innovative In-Ear Pulse Oximetry for Unobtrusive Cardiovascular and Pulmonary Monitoring During Sleep", IEEE Journal of Translational Engineering in Health and Medicine, 2013 Publication	<1 %
23	Submitted to University of York Student Paper	<1 %
24	world.museumsprojekte.de Internet Source	<1 %
25	Submitted to University of Southern California Student Paper	<1 %
26	abipage.ru Internet Source	<1 %

27	www.hindawi.com Internet Source	<1 %
28	www.bartleby.com Internet Source	<1 %

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		