Purbanchal University
Faculty of Engineering

College of Biomedical Engineering and Applied Sciences
Lalitpur, Nepal

Design of IoT- Based Patient Monitoring System

A thesis by:

Christina DC [BME/2018/A12]

Mohit Chand [BME/2018/A20]

Pinkey Chataut [BME/2018/A26]

Prayas Sapkota [BME/2018/B2]

Roman Basnet [BME/2018/B5]

Shraddha Hamal [BME/2018/B9]

Submitted in partial fulfillment of the requirements for the degree of
Bachelor in Engineering
in
Biomedical Engineering

Supervised by:

Er. Menam Pokhrel

Er. Rhishav Poudyal

September 2023

Purbanchal University

Faculty of Engineering

College of Biomedical Engineering and Applied Sciences

Lalitpur, Nepal

Design of IoT- Based Patient Monitoring System

A thesis by:

Christina DC [BME/2018/A12]

Mohit Chand [BME/2018/A20]

Pinkey Chataut [BME/2018/A26]

Prayas Sapkota [BME/2018/B2]

Roman Basnet [BME/2018/B5]

Shraddha Hamal [BME/2018/B9]

Submitted in partial fulfillment of the requirements for the degree of
Bachelor in Engineering
in
Biomedical Engineering

Supervised by:

Er. Menam Pokhrel

Er. Rhishav Poudyal

September 2023

Acknowledgement

We would like to express our deep and sincere gratitude to **College of Biomedical Engineering and Applied Science (CBEAS)** for providing us the opportunity to conduct this project in very suitable environment. We are very grateful to our supervisors **Er. Rhishav Poudyal** sir and **Er**. **Menam Pokhrel** mam for their invaluable guidance and support throughout our project. Their expertise and encouragement helped us to complete this project very easily.

We would like to extend a special thanks to our friend **Lalit Joshi** who went above and beyond to help us with our work. This project wouldn't have been in this stage without his guidelines. We would also like to specially thank you all the faculties, staffs, and students who have been involved in an intangible way which make our project come to an end.

Abstract

The Internet of Things (IoT) has revolutionized various industries, and healthcare is no exception. The IoT-based PMS designed to enhance the quality of healthcare services and improve patient outcomes. In an era where remote patient care and real time data accessibility are paramount, our system leverages IoT technologies to bridge the gap between healthcare providers and patients. The IoT-based PMS comprises a network of interconnected wearable sensors, cloud-based platform, and alarm. Wearable sensors, such as MAX30102 and DS18B20 temperature sensor, continuously collect vital signs and physiological data from the user. These devices seamlessly transmit the data to the cloud platform (ThingSpeak) with a delay of 15 seconds, as free version of ThingSpeak has been used. The cloud platform stores, processes, and analyzes the data, making it accessible to healthcare professionals, patients, and caregivers through web. When the user vital signs are below or above the standard value the alarm buzzes to inform the health professionals for an action required. We successfully able to achieve the objectives of IoT-based PMS which will possibly reduce the patient care by providing timely data-driven interventions, reducing hospital readmissions, and including patient satisfaction.

Keywords: *IoT, Patient Monitoring System, Wearable sensors, Data analytics, Raspberry pi, ThingSpeak, Remote patient care*

Abbreviations

IoT	Internet of Things
ECG	Electro Cardio-Graph
RA	Right Arm
LA	Left Arm
ADC	Analog to Digital Converter
Wi-Fi	Wireless Fidelity
App	Application
R-Pi	Raspberry Pi
SD Card	Secure Digital Card
BPM	Beats Per Minute
GPIO	General Purpose Input/output
HDMI	High-Definition Multimedia Interface
PC	Personal Computer
RAM	Random Access Memory
SpO2	Saturation of peripheral Oxygen
MCU	Micro Controller Unit
SPI	Serial Peripheral Interface
PMS	Patient Monitoring System

Table of contents

List of figures

List of Tables

Chapter 1 Introduction

1.1 Background

Patient Monitoring System (PMS) is widely used in hospital and home for monitoring the vital parameters. Patient monitoring system can be single or multi-parameters for monitoring of O_2 saturation, ECG, respiratory rate, heart rate, blood pressure, body temperature, etc. Multi- parameter monitors are designed to display multiple information of patient on screen. PMS is used in pre-operative and post-operative surgery for constant monitoring of vital signs of the patient. PMS can be used in patient having life threatening condition to get treated if any abnormal conditions arise. It even records the changes in patients' well-being. Sensors are device connected to the monitor that produce output signals for the purpose of sensing a physical process and use the information for medical information. Biomedical sensors are the sensors that are designed to sense the biological changes in the body and convert to its respective electrical signals [1].

IoT refers to the network of interconnected devices and sensors that can exchange data and perform actions without direct human intervention. Healthcare is a crucial domain for IoT applications due to its potential to improve patient care, reduce costs, and enable remote monitoring [2]. Traditional testing at specialist health facilities were the normal way for monitoring body temperature, blood pressure, ECG, and heart rate for many years. Patient monitoring is essential for tracking vital signs, managing chronic conditions, post-operative care, and early detection of health issues. With the advancement of technology today, there is a vast array of sensors that learn crucial signals such as SpO2, temperature regulators, heart rate regulators, and electrocardiograms, allowing patients to take basics on a regular basis. Doctors get daily readings and offer medicines and exercise routines to help patients enhance their quality of life and overcome such conditions. The Internet of things describes the network of physical objects that are embedded with sensor software and other technologies to connect and exchange data with other devices and systems over the internet [3]. Interconnected objects collect the data, analyze and initiate needed action, providing an associated intelligent network for analyzing, designing, and decision making. This service will help every individual by following basic health care, which can lead to more advantageous results. IoT technology is increasing to support the cost and quality of patient life and also ensure the lifespan of patients with proper medication [4].

Machine learning refers to the computer science that focuses on the use of data and algorithms to imitate the way human learn. It plays a pivotal role in enhancing the

capabilities of IoT-based patient monitoring systems, revolutionizing the way healthcare data is analyzed and providing these systems can process the vast amount of data collected from wearable sensors. This enables the detection of abnormalities in patients' physiological in real-time. Machine learning algorithms can identify subtle changes in vital signs that may indicate early signs of deterioration or potential health risks. Logistic regression is a statistical modeling technique used for binary classification tasks. It focuses on predicting the probability of an instance belonging to a specific class [5]. The logistic regression algorithm models the relationship between the independent variables (features) and the dependent variables (binary outcome) using the logistic function, also known as the sigmoid function. This function maps any input to an output between 0 and 1. The model is trained using optimization techniques to find the optimal parameters that maximize the likelihood of the observed data given the model. Once trained, the logistic regression model can make predictions for new instances, assigning them to one of the two classes based on the calculated probabilities. It's simplicity, interpretability, and effectiveness for binary classification tasks make logistic regression a popular and foundational algorithm in machine learning and data analysis [6].

Dataset refers to a collection of data pieces that can be treated by computer as a single unit for analytic and prediction purposes. A dataset used in logistic regression typically consists of structured data points, each representing an observation with several features (independent variables) and binary outcome (dependent variable). These datasets are essential for training and evaluating the logistic regression model. For binary classification tasks, the dependent variable can be categorical or numerical and describe various aspects of the observations. The dataset is split into a training set and a testing set to train the logistic regression model on a subset of the data and assess its performance on unseen instances. The quality, size, and diversity of the dataset significantly impact the model's effectiveness and generalization to new data. Ensuring a balanced distribution of the two classes, handling missing value, and preprocessing the data to address outliners or irrelevant features are important steps in preparing a suitable dataset for logistic regression analysis [7].

Our project emphasizes on monitoring the vital parameters of a patient and display the data in a webpage. It also predicts the condition of patient and alerts through machine learning model. This model uses machine learning model to predict and warn the user if the data of patient are not in normal ranges. It warns using buzzer system and a warning indicator in a webpage simultaneously in case of an emergency. The overall model

developed helps to continuously monitor the patient's oxygen saturation, heart rate and temperature. Along with monitoring it also helps to predict whether the vital signs are in normal ranges or not through machine learning model. The data taken from sensors are displayed on a webpage using ThingSpeak. In case of an emergency, the warning indicator lamp in a webpage lights up and an alarm beep simultaneously.

1.2 Problem statement

Traditional healthcare monitoring methods lack real-time, continuous, and personalized patient data, leading to delayed detection of health issues and limited remote monitoring capabilities. This gap in patient care can result in suboptimal outcomes, increased hospitalization rates, and a higher burden on healthcare resources. There is a need for an efficient and secure IoT-based patient monitoring system that integrates wearable sensors, data transmission, and advanced analytics to enable timely detection of health anomalies, personalized treatment plans, and remote monitoring in a patient's natural environment. This system must address challenges related to data security, interoperability, power efficiency, and regulatory compliance while ensuring seamless communication between devices and providing healthcare professionals with actionable insights for informed decision-making.

1.3 Objectives

1.3.1 General objectives

- Design of IoT based Patient Monitoring System (PMS).

1.3.2 Specific objectives

- To develop a device that monitors basic three vital parameters (Temperature, Heart rate, SpO_2).
- To understand the implementation of real time data analytics and machine learning algorithms for personalized healthcare recommendations.
- To use a user-friendly web for patients to access their health data and for healthcare providers to conduct remote consultations.
- To develop the system that informs the healthcare professional for the necessity of taking an appropriate action when needed.

Chapter 2 Literature Review

2.1 Sensors

Sensors are the device that produces an output signal for the purpose of sensing a physical phenomenon [8]. The input can be light, heat, motion, moisture, pressure or any number of other environmental phenomena. The output is generally a signal that is converted to a human-readable display at the sensor location or transmitted electronically over a network for reading or further processing. Sensors play a pivotal role in the internet of things. They make it possible to create an ecosystem for collecting and processing data about a specific environment so it can be monitored, managed and controlled more easily and efficiently. IoT sensors are used in homes, out in the field, in automobiles, on airplanes, in industrial settings and in other environments [9] . IoT sensors gather information so devices can be used remotely and data can be shared in real time. The data gathered by IoT sensors and sent to the cloud is analyzed by software that can make sense of the information and then sent to users [10] . Sensors measure a person's vital signs, like heart rate and oxygen level. Wearable devices like smartwatches and arm bands can be equipped with biomedical sensors. These devices communicate the data they collect back to the wearer so they can monitor their own health, or to a caregiver or medical staff, who is monitoring a patient or loved one remotely.

2.2Vital signs

The essentials functioning of your body are measured by vital signs. Body temperature, SpO_2, heart rate and respiratory (breathing) rate are among them. The normal ranges for these indicators vary depending on age, BMI, and other factors. Pediatric vital signs differ from adult vital signs. A healthcare practitioner will notify you if any of your vital signs need to be monitored [11]. Vital signs are useful in detecting or monitoring medical problems. Vital signs can be measured in a medical setting, at home, at the site of a medical emergency, or elsewhere.

The eight vital signs of patient monitoring system are temperature, pulse, respiration rate, oxygen saturation, pain, level of consciousness, urine output and blood pressure [12]. The table 2.1 expresses the vital signs with their normal ranges respectively in any normal human being. We have been using the 3 vital sign parameters that are temperature, heart rate, and SpO2. The reason behind using the only three parameters is generally because

easily availability of sensors, well interfacing with the microprocessor, their selection provides a comprehensive initial assessment that covers both cardiovascular and respiratory aspects of a patient's health.

Table 2. 1: Vital signs parameters [12]

Vital Signs	Normal range
Temperature	97.8 F to 99.1 F (36.5 C to 37.3 C)
Heart rate	50-100bpm
Pulse	60 to 100 beats per minute.
Blood pressure	90/60 mm Hg to 120/80 mm Hg.
O_2 saturation	93-100%

ECG can help to detect:

- Arrhythmias: Irregular heartbeat, either too slow or too fast.
- Coronary heart disease: where the heart's blood supply is blocked or interrupted by fatty substance.
- Heart attack: where the blood supply is blocked.
- Cardiomyopathy: where the heart wall become thickened enlarged.

2.2.1 ECG(Electrocardiogram)

An electrocardiogram is a simple non-invasive test which measure the electrical activity of the heart. Electrode are placed at certain place in left chest, arms and legs. The figure 2.1 depicts the ECG graph which must be obtained in the similar form when the electrodes in the user are placed. The placement of three lead electrode is described below:

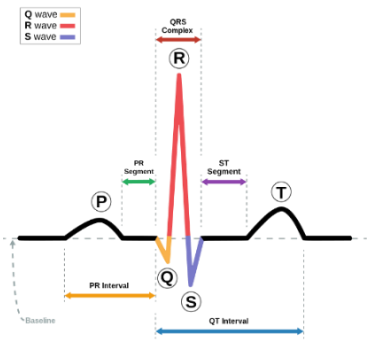


Figure 2.1: ECG waveform [13]

a) Placement of three lead electrode

In the figure 2.2 one electrode is connected to right arm second is connected to left arm and third one is connected to left leg. The Three limb electrodes I, II and III form a triangle (Einthoven's Equilateral Triangle), at the right arm (RA), left arm (LA), and left leg (LL) [14].

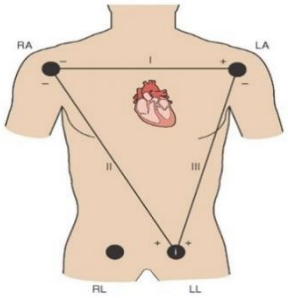


Figure 2.2: Three lead placement [14]

2.2.2 Oxygen saturation (SpO_2)

Oxygen saturation is the ratio of oxy-hemoglobin to the total amount of hemoglobin in the blood. Hemoglobin is responsible for transporting oxygen from the lungs to other part of the body. Maximum four oxygen molecules can be carried by one hemoglobin molecule. If a group of 1000 hemoglobin molecules were each carrying 3600 oxygen molecules, the oxygen saturation level would be (3600/4000) *100 or 90%. 1000 hemoglobin molecules may transport a maximum of 4000 oxygen molecules. There is arrival of various types of conditions when the SpO2 is abnormal [15]. The table 2.2 shows the various conditions with their respective SpO2 ranges. The normal range for most of the human being is generally from 95% to 100%.

Table 2. 2: Range of oxygen saturation [16]

Normal	95-100
Mild Hypoxemia	91-94
Moderate Hypoxemia	86-90
Severe Hypoxemia	<85

2.2.3 Heart rate

Heart rate, is considered as the number of times your heart beats per minute. A normal resting heart rate should be between 60 to 100 beats per minute, but it can vary from minute to minute. The heart rate is influenced by many factors, including (but not limited to) genetics, physical fitness, stress or psychological status, diet, drugs, hormonal status, environment, disease/illness, as well as the interaction between and among these factors. The heart rate can change depending on the body's physical needs, including the need to absorb oxygen and excrete carbon dioxide [17]. Heart rate variability is the variation of time intervals between the heartbeats or it is the variability of the period of time between each heartbeat. Figure 2.3 depicts the heart rate variability so that it is easy to understand the concept of heart rate variability.

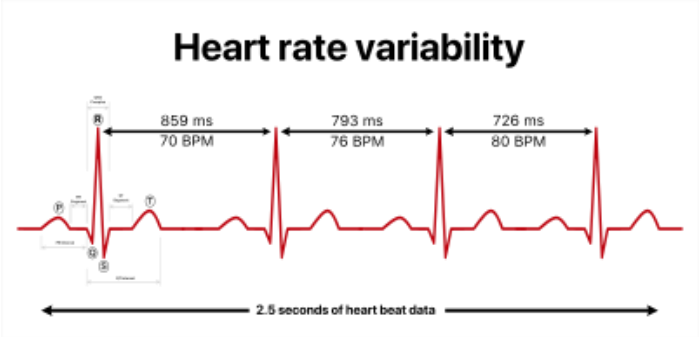


Figure 2.3: Heart rate variability [18]

The table 2.3 shown below is about the different heart rate ranges and average heart rate at different age of any normal human being. Due to table, it is easy to understand and decide the normal functioning of heart rate.

Table 2. 3: Range of heart rate [19]

Age	Heartrate (bpm)	Average heartrate
Newborn	100-180	140
2-3 yeas	80-130	110
6-8 years	75-115	95
12-16 years	60-110	85
>16	60-100	80

2.2.4 Human body temperature

The average temperature range in humans is known as normal human body temperature. Usually, it is claimed that the normal range for human body temperature is 36.5–37.5 °C (97–99.5 °F). The temperature of an individual fluctuates by sex, age, exercise level, health condition and emotion [20]. Thus, table 2.4 shows us different conditions of temperature with their respective ranges.

Table 2. 4: Condition of body temperature along with their readings [20]

Condition	Temperature
Normal	36.5–37.5 °C
Hypothermia	<35°C
Hyperthermia	>38°C
Hyperpyrexia	>40°C

2.3 Patient monitoring system

Patient monitoring systems are used for measuring continuously or at regular intervals automatically the value of the patient's important physiological parameter. The patient Monitoring System is available in both stand-alone and centralized configurations. Patient Monitoring System is mainly classified into Analog and Digital types. There are several categories of patients who may need continuous monitoring or intensive care [21]. The major concept of patient monitoring system can be understood by the block diagram shown in figure 2.3, which can be understood as the physiological parameters being taken from the patient and then processed to give the final output in user understandable form.

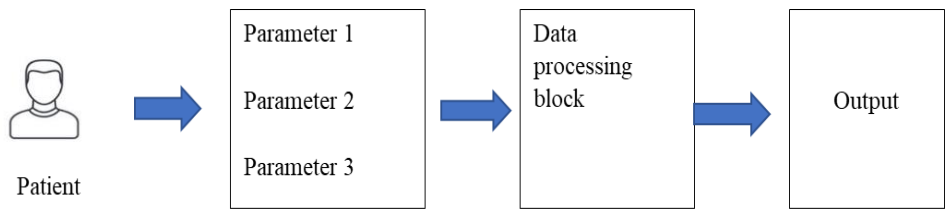


Figure 2.3: Block diagram of patient monitoring system

The long-term objective of the patient monitoring system is generally to decrease mortality and morbidity by:

1. Organizing and displaying meaningful information for improved patient care.
2. Co-relating multiple parameters for a clear demonstration of the clinical problem.
3. Processing data to set alarms on the development of abnormal conditions.
4. Providing information based on automated data.
5. Ensuring better care with fewer staff members.

2.3.1 Patient Monitoring in Intensive-Care Units

There are at least five categories of patients who need physiological monitoring:

1. Patients with unstable physiological regulatory systems; for example, a patient whose respiratory system is suppressed by a drug overdose or anesthesia.
2. Patients with a suspected life-threatening condition; for example, a patient who has findings indicating an acute myocardial infarction (heart attack).
3. Patients at high risk of developing a life-threatening condition; for example, patients immediately after open-heart surgery or a premature infant whose heart and lungs are not fully developed.
4. Patients in a critical physiological state; for example, patients with multiple trauma or septic shock.
5. Mother and baby during the labor and delivery process [22].

2.3.2 Historical perspective of Patient Monitoring System

Different researchers have done different innovation in the field of healthcare. People have been monitoring the vital signs of others since the dawn of mankind, using various methods to track heart rate, body temperature, respiratory rate, and arterial blood pressures.

- In 1625, when Santorio of Venice, with help from his good friend Galileo, published methods for measuring body temperature with a spirit thermometer, and timing the pulse rate with a pendulum. However, their findings were largely ignored. It was only with the publication of “Pulse-Watch” by Sir John Floyer in 1707 that the first scientific report pertaining to the pulse rate came to light.
- In the 1800s, the patient monitoring system as we know it started when the first two physicians consulted over the telephone. (Consultations may have been carried out by smoke signals and heliograph before the telegraph, but there is no documentation) [23].

- In 1840, a study by Albakret al on A Cloud-Assisted IoTs Framework, the authors explained that recent advances in cloud computing and Internet of Things have made their way into the healthcare industry. Their study noted that the integration of IoTs and cloud computing in the health care domain poses several technical challenges with less attention in the research community. Hence, authors proposed a smart city framework the help reduce the effect of the challenges [24].
- In 1852, Ludwig Taube published the first-ever plotted course of fever in a patient circa, adding respiratory rate to the list of human vital signs trackable at the time. Subsequent improvements in the thermometer and clock solidified the heart rate, respiratory rate and body temperature as the standard vital signs monitored by medical professionals of the time [25].
- In 1924 a speculative cover of radio new magazine showed the use of the radio for medical consultation. In 1948 technology advanced and one doctor sent x-ray images over telephone wires to another doctor in Pennsylvania. In 1959 neurological exams were electronically transmitted for consultation [26].
- In the 1970s, a remote monitoring program was developed to oversee healthcare at what was then called the Papago Indian Reservation in Arizona. The sponsors of the program were Kaiser Foundation and Lockheed. The program experienced numerous problems and was discontinued in 1977, but NASA used the knowledge gained to improve its space technology.
- By 1980 transmitting X-ray images became routine. The field shifted in the 1990s with the development of the internet. The internet explosion left old-school telemedicine behind. Not until the broadband infrastructure became available did telemedicine take off. Now remote monitoring means more than just telecommunicated examinations and consultations, more than sending X-rays and CT and MRI scans and lab data. Now the patient can stay in their own homes and still receive quality healthcare. The blood is collected and sent to the laboratory by mail, eliminating long drives and painful blood draws at the lab.
- Back to the 1990s, Bluetooth was a short-range communication technology on those dates; the first official Bluetooth device was produced this year. By 2007, 50% of all internet users have access to broadband. Changes to CPT codes make RPM one of the most profitable Medicare care management systems, but adoption remains modest by 2019 [27].

- Before the COVID-19 pandemic, patient monitoring approaches to prevent harm were linked to where the patient was treated in the hospital. The pandemic accelerated the move to monitoring and therapy based on patient risks and needs. Based on a hypothetical cohort of 3100 patients, the study estimated that remote monitoring might result in 87% fewer hospitalizations, 77% fewer deaths, $11 472 in per-patient costs savings over conventional care, and 0.013 quality-adjusted life-year improvements. Pulse oximeters used in hospitals can now be deployed at home with patient data relayed to smartphones, secure cloud servers, and web-based dashboards where physicians and hospitals can monitor the patient's status in near real time [28].

Figure 2.4: ICU-80 patient monitor [29]

2.4 Internet of Things

The Internet of Things refers to a network of physical objects or things that are embedded via the internet with sensors, software, and other technologies in order to connect to and exchange data with other devices and systems. The Internet of Things has several applications. The main use is the gathering of data on any patient's vital signs and display data in webpage [30]. IoT technology is increasing to support the cost and quality of patient life and also ensures the life span of patients with proper medication. In conventional health care, undetected health problems can be solved through this IOT Technology thereby ensuring healthcare services by maintaining a digital identity for each patient complications can be greatly reduced. The communication between the health sensors device with the computer or smartphone has the default ability to communicate with the server which makes

the whole system cost reduction and the complexity of the system is also reduced [31]. IoT in the overall have the potential to entirely reshape the healthcare system fully automating without the need of manually human driven system.

For many years the standard method for measuring body temperature, blood pressure, ECG, and heart rate was traditional tests in specialized health facilities. With the advent of technology today, there is a huge diversity of sensors that learn important signals such as a blood pressure monitor, temperature regulator, a heart rate regulator, including electrocardiograms, which allow patients to take essentials daily. Daily readings are sent to doctors and they will recommend medication and exercise procedures that allow them to improve their quality of life and overcome such disease [32]. The monitoring system measures the vital signs of the patient body and processes in ESP8266 after filtration and amplification. The processed signal is subsequently displayed in webpage in real time and early warning score predict the condition of patient normal to critical allowing for speedy diagnosis and treatment of the patient. With the use of IoT, these measures are obtained to help assess a person's general physical health, provide hints to prospective ailments, and demonstrate progress toward recovery [33]. The normal ranges for a person’s vital signs vary with age, weight, gender, and overall health. The vital signs are evidence of how the body is currently operating physically. They refer to a collection of the most critical medical indicators that show the state of the body's vital (life-sustaining) activities.

2.4.1 Relation between the patient health monitoring system and IoT

For many years the standard method for measuring body temperature, blood pressure, ECG, and heart rate was traditional tests in specialized health facilities. With the advent of technology today, there is a huge diversity of sensors that learn important signals such as a blood pressure monitor, temperature regulator, a heart rate regulator, including electrocardiograms, which allow patients to take essentials daily. Daily readings are sent to doctors and they will recommend medication and exercise procedures that allow them to improve their quality of life and overcome such disease [32]. The monitoring system measures the vital signs of the patient body and processes in ESP8266 after filtration and amplification. The processed signal is subsequently displayed in webpage in real time and early warning score predict the condition of patient normal to critical allowing for speedy diagnosis and treatment of the patient. With the use of IoT, these measures are obtained to help assess a person's general physical health, provide hints to prospective ailments, and demonstrate progress toward recovery [33]. The normal ranges for a person’s vital signs

vary with age, weight, gender, and overall health. The vital signs are evidence of how the body is currently operating physically. They refer to a collection of the most critical medical indicators that show the state of the body's vital (life-sustaining) activities.

2.4.2 Raspberry Pi in IoT

The features of raspberry make it one of the most suitable microcontrollers for IoT. I/O pins present on the board are used to interfere with the sensor. The sensor data can be processed by the embedded processor and can be shown on output screens. Raspberry pi supports the HDMI cable monitor, keyboard, and mouse with its default operating system [34]. Ethernet cable or Wi-Fi provide internet access to the board by which information can either be improved or exported. In brief, we can say raspberry pi can input sensor data along with taking user data through a custom GUI. The Raspberry Pi Foundation provides Arch Linux ARM and Debian distributions for download, and promotes Python as the main programming language, with support for BBC BASIC, Java, C, Perl, Ruby, PHP, Squeak Smalltalk, C++, etc [35].

2.5 Machine Learning

Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior. Artificial intelligence systems are used to perform complex tasks in a way that is similar to how humans solve problems.

Machine learning can also be used in prediction system. Machine learning focuses non development of computer program that can access data and use it to learn for themselves. The study of computer algorithm can improve automatically through experience and by the use of data. It is also a type of data that is used in terms of learning algorithms in a wide variety of applications such as medicine, speech recognition and computer vision [36].

Machine learning is a valuable data analysis and visualization technique. It enables to uncover insights and patterns from massive datasets for use in understanding complex systems and making educated decisions [37].

1.5.1 Types of machine learning

a) Supervised learning

Supervised learning requires outside supervision for the machine to learn. The labeled dataset is used to train the supervised learning models. After training and processing, the model is put to the test by being given a sample set of test data to see if it predicts the desired result [38].

b) Unsupervised learning

Unsupervised learning is a type of machine learning in which machine do not need any external supervision to learn from data. the unsupervised models can be trained using the un labelled dataset that is neither classed nor categorized [38].

c) Reinforcement learning

In reinforcement learning, an agent produces actions to interact with its environment and learns from feedback. The agent receives feedback in the form of rewards; for example, he receives a positive reward for each good activity and a negative reward for each bad action. Q algorithm is used in reinforcement learning [38].

2.5.2 Logistic Regression

Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on. Multinomial logistic regression can model scenarios where there are more than two possible discrete outcomes. Logistic regression is a useful analysis method for classification problems, where you are trying to determine if a new sample fits best into a category. As aspects of cyber security are classification problems, such as attack detection, logistic regression is a useful analytic technique [39].

Logistic Regression is given by Sigmond Function as follow: -

$$Y = \frac{1}{1+e^{-x}} \text{ .....1}$$

where,

Y = output

e = Euler's constant (2.718)

x = Independent variable

2.5.3 Early warning score

Early warning system (EWS) scores are tools used by hospital care teams to recognize the early signs of clinical deterioration in order to initiate early intervention and management, such as increasing nursing attention, informing the provider, or activating a rapid response or medical emergency team [40].

PHYSIOLOGICAL PARAMETER	SCORE						
	3	2	1	0	1	2	3
Respiration rate per minute	≤8		9–11	12–20		21–24	≥25
SpO2 Scale 1 %	≤91	92–93	94–95	≥96			
SpO2 Scale 2 %	≤83	84–85	86–87	88–92 ≥93 on air	93–94 on oxygen	95–96 on oxygen	≥97 on oxygen
Air or Oxygen		Oxygen		Air			
Blood pressure Systolic / mmHg	≤90	91–100	101–110	111–219			≥220
Pulse rate per minute	≤40		41–50	51–90	91–110	111–130	≥131
Consciousness				Alert			CVPU
Temperature °C	≤35.0		35.1–36.0	36.1–38.0	38.1–39.0	≥39.1	

Figure 2.5: Early warning score [41]

2.5.4 Dataset

A set of data used to train the model is known as a machine learning dataset. To educate the machine learning algorithm how to make predictions, a dataset is used as an example. To evaluate how well the model was trained, a single training dataset that has previously been processed is typically split into many pieces. A testing dataset is typically isolated from the data for this reason. Next, while not technically necessary, a validation dataset is very useful to prevent training your algorithm on the same sort of data and producing biased predictions [42].

2.6 ThingSpeak

ThingSpeak is an analytical device which help you to gather, visualize, and examine real-time data streams online. ThingSpeak instantly visualized data obtained from different devices. You can send data from any internet-connected device directly to ThingSpeak using a Rest API or MQTT. ThingSpeak helps to store and analyze data in cloud without configuring web servers [43].

Figure 2.6: ThingSpeak platform for sending data [44]

Chapter 3 Materials and Methodology

3.1 Materials

3.1.1 Hardware Required

a) DS18B20 Temperature sensor

The DS18B20 is a digital body thermometer that provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with non-volatile user programmable upper and lower trigger points. It doesn't require Analog to digital converter (ADC). It can be used to measure temperature in the range of -67°F to +257°F or -55°C to 125°C with ±5% accuracy [45].

Figure 3. 1: DS18B20 temperature sensor [46]

Table 3. 1: Pin configuration of DS18B20 sensor [47]

Pin No.	Pin name	Description
1	GND	This is the ground pin of the sensor; it needs to connect to the ground terminal of microcontroller.
2	VCC	This is the power supply pin of the sensor; it needs to connect to the 3.3V or 5V terminal of the microcontroller/microprocessor.
3	DATA	This is the output pin. It provides the output using 1 wire method that should be connected to a digital pin on the microcontroller/microprocessor.

b) Max 30102 Heart rate sensor

The MAX30102 is an integrated pulse oximetry and heart-rate monitor module. It includes internal LEDs, photodetectors, optical elements, and low-noise electronics with ambient light rejection. The MAX30102 provides a complete system solution to ease the design-in process for mobile and wearable devices. The MAX30102 operates on a single 1.8V power supply and a separate 3.3V power supply for the internal LEDs. Communication is through a standard I2C-compatible interface. The module can be shut down through software with zero standby current, allowing the power rails to remain powered at all times [48]. The pin configuration is also mentioned in table 6 below:

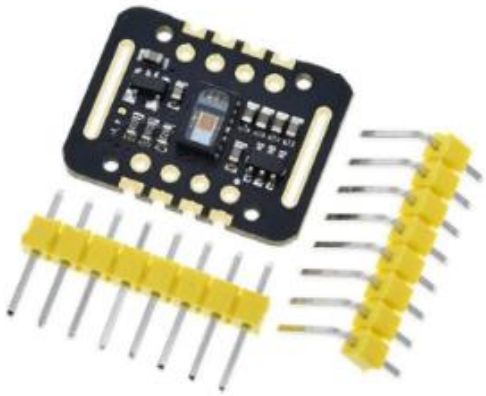


Figure 3. 2: MAX30102 sensor [48]

Table 3. 2: Pin configuration of MAX30102 sensor [49]

PIN	NAME	FUNCTION
2	SCL	I2C Clock Input
3	SDA	I2C Data, Bidirectional (Open-Drain)
4	PGND	Power Ground of the LED Driver Blocks
9 ,10	VLED+	LED Power Supply (anode connection). Use a bypass capacitor to PGND for best performance.
11	VDD	Analog Power Supply Input. Use a bypass capacitor to GND for best performance.
12	GND	Analog Ground
13	INT	Active-Low Interrupt (Open-Drain). Connect to an external voltage with a pullup resistor

d) Raspberry Pi 4

Raspberry Pi 4 Model B offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while retaining backwards compatibility and similar power consumption. For the end user, Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems. The quad-core Raspberry Pi 4 Model B is both faster and more capable than its predecessor, the Raspberry Pi 3 Model B+ With 4GB RAM, the Pi 4 no longer struggles with heavy web pages and apps, and is able to switch between full online services such as Google's G Suite and today's JavaScript laden sites without lagging [50].

Figure 3. 3: Raspberry Pi 4 [50]

3.1.2 Software

a) Python

Python is an interpreted, high-level, general-purpose programming language designed by Guido van Rossum and first released in 1991. Python, like other languages, has gone through a number of versions. Python 0.9.0 was first released in 1991. In addition to exception handling, Python included classes, lists, and strings. More importantly, it included lambda, map, filter and reduce, which aligned it heavily in relation to functional programming. Some excellent features of python make it very compatible for using it in IoT based projects. Features are:

a) Python Code is clear and readable.

b) Can easily be integrated with microcontrollers like Raspberry pi

c) Data can be exported to IoT cloud with inbuilt library

d) GUI is very easy to design

The python GUI is designed using tkinter for manual inputs of the patient data and sensors connected to the pin are acquired. Python is preinstalled in operation system of Raspberry pi. The expert system present in the raspberry pi as an offline system gets inputs from these data and helps to predict the normal or critical condition of patient based on obtained data. Then Python transfers all the data to the IoT cloud for storage [51].

b) ThingSpeak:

It is an IoT platform developed by MathWorks. It provides a cloud-based environment for collecting, analyzing, and visualizing data from IoT devices and sensors. ThingSpeak enables users to easily create and manage channels to which devices can send data in the form of time-series information, such as temperature, oxygen saturation level, heart rate and more. This platform offers built-in tools for data visualization, allowing users to create customizable charts, graphs and gauges to better understand their data trends. ThingSpeak serves a versatile platform for IoT data management and analysis for working on IoT projects [51].

c) Google Collaboratory:

It is a cloud-based platform provided by Google that offers a free environment for writing and executing Python code. It is designed to facilitate collaborative and interactive coding, enabling users to create, share, and run code in a browser without requiring any setup or installation of software on the local machines. It allows users to write and execute Python code, access pre-installed libraries, and harness the computational power of Google's cloud infrastructure, making it an ideal choice for machine learning, data science, and research projects. Users can easily share and collaborate on notebooks, making it a versatile tool for both individuals and teams to work on data-driven tasks without the need for local hardware resources or software installation [52]. It provides a runtime fully configured for deep learning and free-of-charge access to a robust GPU. The Google Collaboratory infrastructure is hosted on the Google Cloud platform. Collaboratory notebooks are based on Jupyter and work as a Google Docs object: can be shared and users can collaborate on the same notebook. Collaboratory provides either Python 2 and 3 runtimes pre-configured with the essential machine learning and artificial intelligence libraries, such as TensorFlow, Matplotlib, and Keras [53].

3.2 Methodology

3.2.1 Overall system design

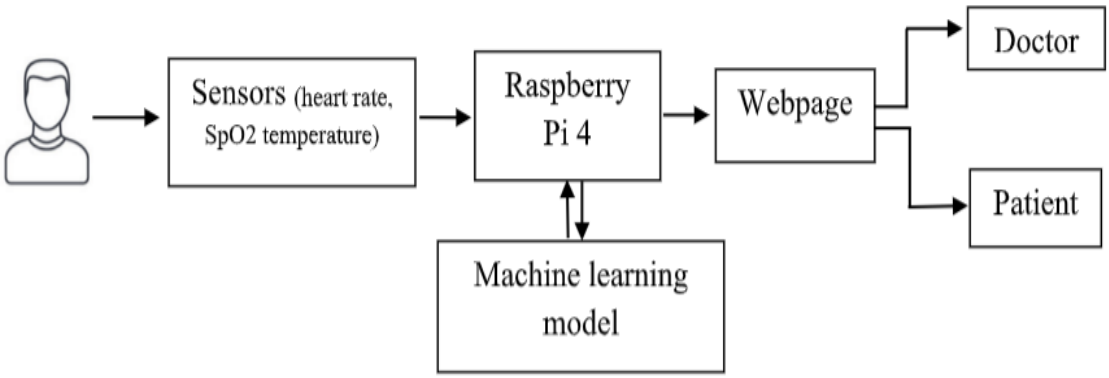


Figure 3. 4: Block diagram of overall patient monitoring system

Above figure 3.5 shows the overall block diagram of the system. Various sensors such as; temperature sensor, Heart rate sensor, and SpO2 sensor are attached to the body of patient. Those sensors get interfaced with the Raspberry Pi with help of the code. The data is received in the time of 15 seconds delay as we are using free version of ThingSpeak. Assuming the delay as real time, we keep continuing the continuous monitoring of the data at the ThingSpeak. The data is then used for prediction using machine learning which was already trained and tested using the early warning score dataset that was extracted from the Kaggle. After the prediction, the observer is clear whether the condition of patient is normal or critical. The prediction is done on the basis of early warning score (EWS), the statement can be understood more clearly on how the EWS score predicts the output by the code statements in appendix E (highlighted is done).

As an additional we also added alarm system as a warning indicator. The buzzer beeps as the early warning score are equal to or exceeds 3 and alerts respective personnel. Also, the led indicator also blinks with the red light in the ThingSpeak so that the observer in the ThingSpeak can easily be aware about the situation of the patient and if there is any critical situation the observer can inform to the health personnel and the patient might be rescued.

The flowchart showing the overall steps of the model is described in the figure 3.5 as mentioned below:

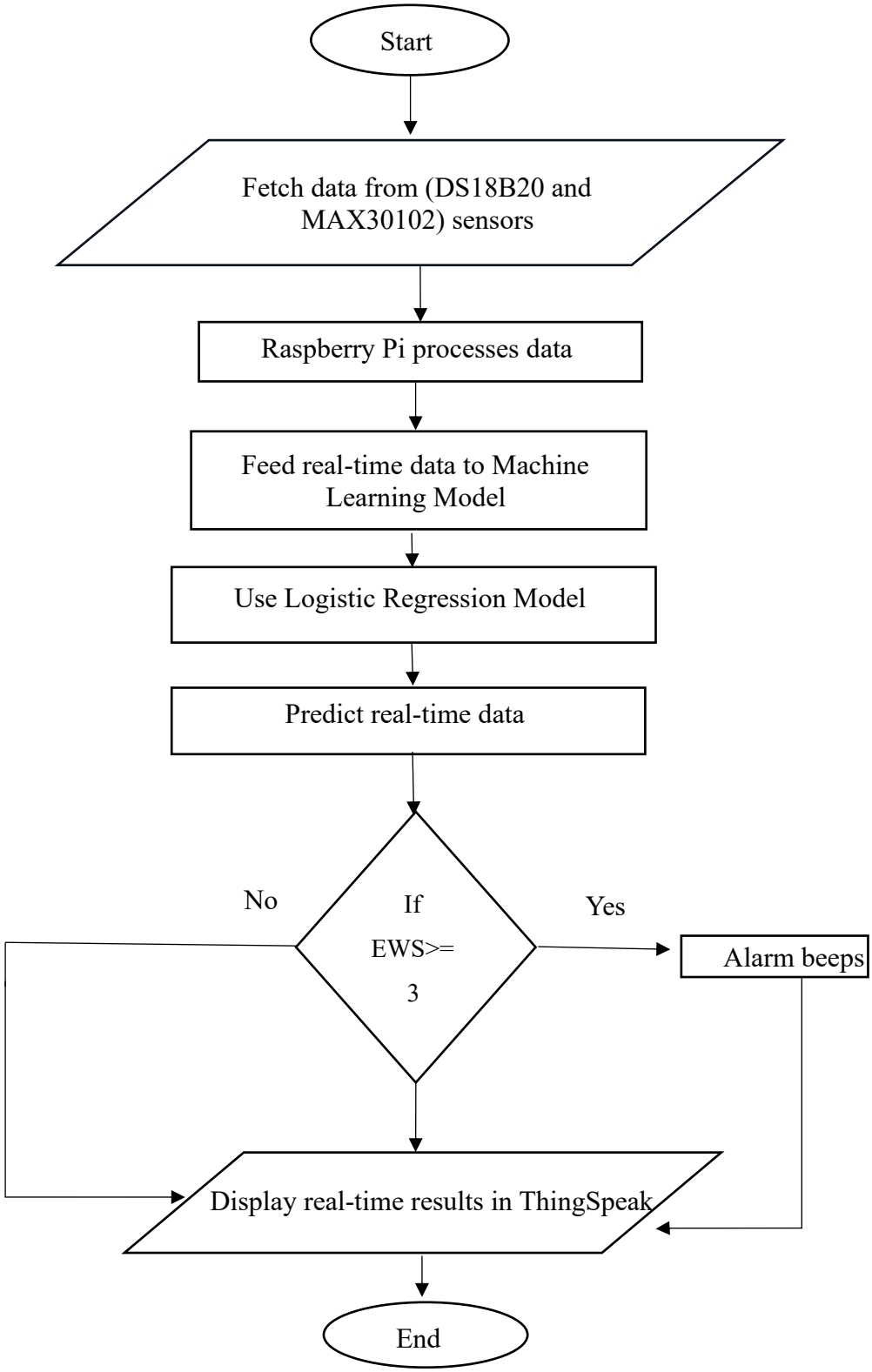


Figure 3. 5: Flowchart of the model

3.2.2 Circuit Diagram

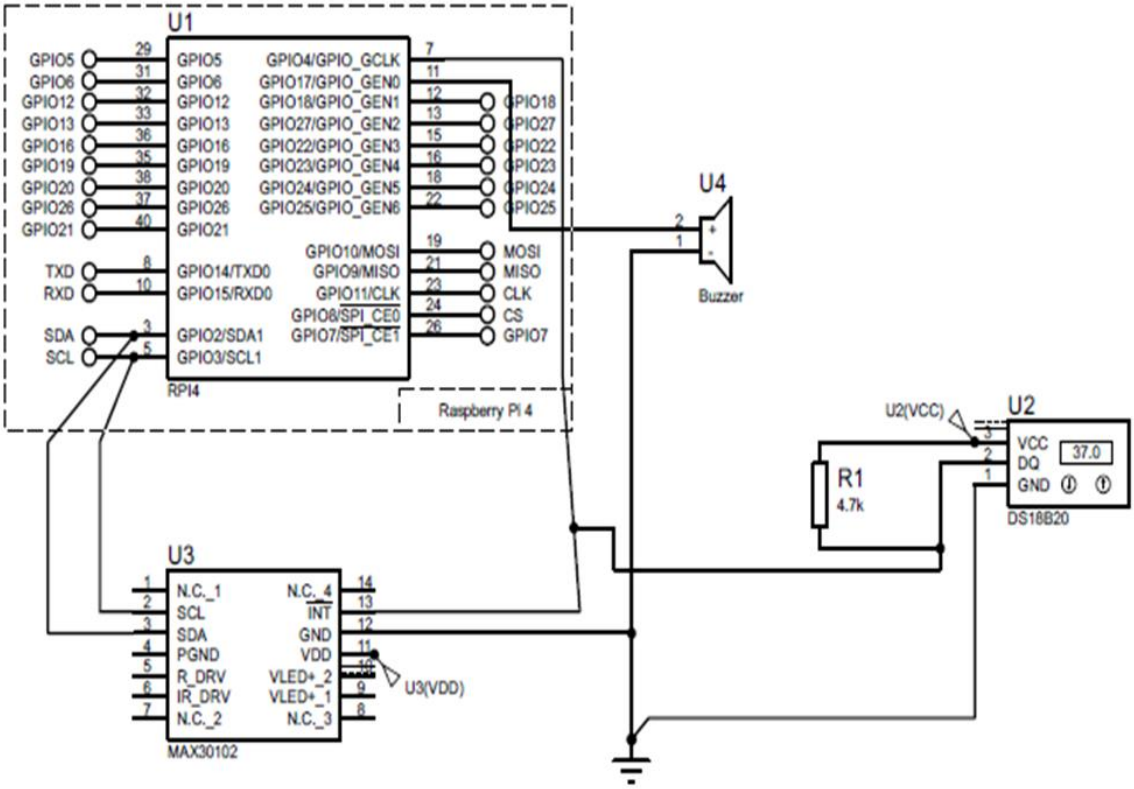


Figure 3. 6: Circuit diagram of the model

The sensors DS18B20, MAX30102 and buzzer were interfaced with Raspberry Pi. The interfacing of the above sensors is shown in the figure 3.6 above:

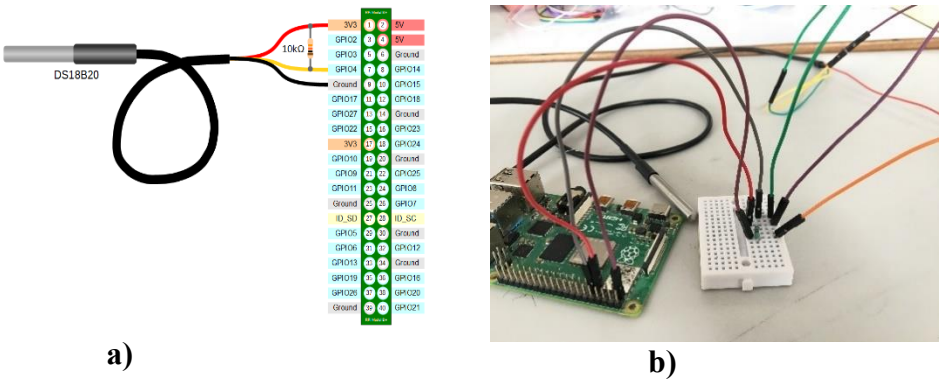
a) DS18B20 Temperature sensor

The temperature sensor has been interfaced with Raspberry pi 4 in order to measure the temperature parameter of the user. The table 3.3 shows the pin configuration of temperature sensor and raspberry pi and figure 3.7 (a) shows the interfacing of temperature sensor with the raspberry pi which we performed successfully. After the pin configuration the coding was done in order to set the proper transmission of data from sensor to the raspberry pi. Rather than the hardware, the complexity arises while working with the coding (appendix A) part. As the raspberry pi uses Linux kernel, not being familiar with Linux it was really a time-consuming work for days to set the Linux kernel modules to communicate with the sensor via 1-wire bus protocol. After the successful generation of temperature values in the terminal of Geany IDLE as shown in figure 3.7 (b), some relief arises but still there was a chaotic situation whether our temperature sensor data is right or wrong, so we took ice full of beaker and put the temperature sensor in it and it really shows the temperature below

0°C which means sensitivity of temperature sensor is working well. But still when measuring the temperature of human user, it must be highly accurate, so we planned to measure the temperature of human body with the actual PMS being used in the hospital which is calibrated. The temperature sensor has been connected with Raspberry pi 4 as followings:

Table 3. 3: Interfacing DS18B20 with Raspberry Pi

DS18B20 Temperature sensor	**Raspberry Pi 4**
VCC	3 V
Data	GPIO4
GND	Ground



a) **b)**

Figure 3. 7: a) Pin configuration of DS18B20with RaspberryPi b) DS18B20 connected with Raspberry Pi

b) MAX 30102 Heart rate sensor

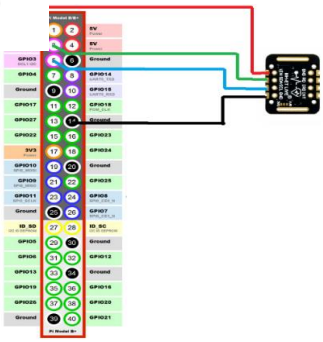
The MAX30102 sensor has been connected to Raspberry pi 4 to extract the physiological parameters of heart rate and SpO2 from the user. Max30102 is configured with raspberry pi as shown by pin configuration in table 3.4. After the pin configuration, the coding was done for setting up the connection. Some issues arise during the max30102 coding which we are included in the discussion part, but finally we gained the output from MAX30102. In figure 3.8 (a) the pin configuration of MAX30102 with RaspberryPi is mentioned and in figure 3.8 (b) the shadow of red light is being seen which was obtained after the successful execution of code (appendix B). The output was quite accurate to validate the sensor is working. The data was fluctuating due to noise from surroundings, jumper wires, zigzag wire connections, etc. So, we planned to test MAX30102 sensor whether it is accurate or

not with the actual PMS being used in the hospital which is calibrated. The MAX30102 sensor has been connected to Raspberry pi 4 as follows:

Table 3. 4: Interfacing MAX30102 with Raspberry Pi

MAX30102 Heart Rate Sensor	**Raspberry Pi 4**
Vin	3V
SDA	GPIO2
SCC	GPIO3
GND	Ground
INT	GPIO4

a)



b)

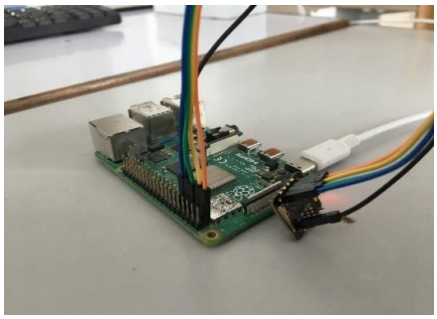


Figure 3. 8: a) Pin configuration of MAX30102 with RaspberryPi b) MAX30102 connected with RaspberryPi

c) Buzzer

Buzzer has been interfaced with Raspberry Pi to buzz as a warning sign for the given sensors. The table 3.5 shows the pin configuration of buzzer with Raspberry Pi and figure 3.9 shows the interfacing of buzzer with the raspberry pi which we performed successfully. After the pin configuration the coding was done in order to connect the buzzer with RaspberryPi. If the EWS is greater than or equal to 3 then the buzzer buzzes and if the EWS is less than 3 then buzzer doesn't buzz.

Table 3. 5: Interfacing Buzzer with RaspberryPi

Buzzer	**Raspberry Pi 4**
Vin	GPI017
Ground	Ground

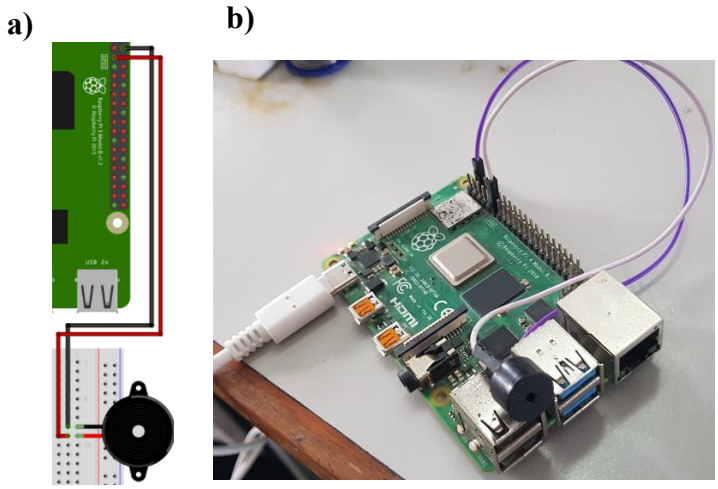


Figure 3. 9: a) Pin configuration of buzzer with RaspberryPi b) Buzzer connected with Raspberry Pi

3.2.3 ThingSpeak Platform

After the hardware configuration we combined the code of both sensors used previously. The successful integration of codes provides the temperature, heart rate and SpO2 data simultaneously at a same time. Now, we moved toward building up the ThingSpeak platform to visualize our data in more attractive and understandable way. We signed up for the ThingSpeak and logged into the IoT platform of ThingSpeak. We created channel and inside channel we created various fields that will show the data of different parameters such temperature in Celsius and Fahrenheit will be shown in different field, heart rate, SpO2 and warning indicator in different fields. The channel generates read and write API keys that can be used to send or retrieve data in or from the ThingSpeak. We take the read API key and implemented in our code along with the URL for updating the ThingSpeak channel. When the code with the API key and URL was run, the channel gets updated with the values of parameter obtained from the sensors. All the data were almost seen to be accurate but the heart rate keeps on fluctuating which might be most probably due to the noise.

3.2.4 Machine learning

After the successful setup of thingspeak platform we need to take the data from thingspeak and predict the normal or critical condition of patient. we need to understand how machine learning is integrated for real time data anaytics because the real time data coming in from sensors to the cloud and from the cloud same data is being used to predict the condition of patient . Logistic regression was used for the prediction of the condition of patient by using the Early Warning Score (ESW) dataset. The ESW dataset includes 1103 different data of

patients. Three parameters SpO_2, heart rate and temperature were used for training and testing the model and the model was used for prediction of new patients.

Using the write api key we extracted the data from thingspeak platform and then preprocessed data by stacking all the data obtained from different fields. The processed data was used as input for the trained model for making predictions. We than used the EWS threshold to transform the predictions into actionable results. If EWS is less than or equal to 2 then the person is normal and if EWS goes above 3 then the person needs attention. We added arlam system in which buzzzer buzzes if early waring score (ews) is greater or equal to 3.

3.2.5 PCB Design

We used easyEDA software for designing PCB. EasyEDA is a web-based software program that enables users to create PCB layouts and schematics for electronic circuit. Figure 3.10 is the schematic drawn from EasyEDA for our final work. The simulation was not possible as EasyEDA doesn't have feature of simulating schematic. So, we simulated schematic using the proteus, figure 3.7 is the circuit diagram that was actually made for performing simulation as schematic. The schematic was found to be correct with all connections. Actually, the simulation was done later on and before performing simulation we actually used the heat and trial method of designing PCB board. We design schematic and convert it into footprint and set everything on a single bottom layer (just for easiness while printing the board). We just performed heat and trial method because we need to revive the knowledge of printing PCB board within a time.

The different steps we came to know about how the PCB is designed during our heat and trial method are discussed as:

- We need to print the footprint after we finished designing it according our own need. Footprint file from EasyEDA is converted into the PDF file and then printed in actual size in the glossy paper. Footprint generally provides precise guidelines for placing the components in the correct positions.
- Cutting the copper clad board of desired size and cleaning it to remove any dirt or oxidation
- Then placing the printed glossy paper into the copper board and applying heat and pressure by running a hot household iron over the paper for several minutes. This is done because it melts the toner and transfers it to the copper surface.

- After heating, carefully peeling off the glossy paper while it's still warm. The toner pattern should have been transferred to the copper board.
- Etching the PCB by immersing the copper board in bucket containing with an etchant solution (Sulphuric acid) to remove excess copper. Etching process leaves behind the copper traces, while rest of the copper will be dissolved.
- Washing the PCB thoroughly with clean water to remove any etchant.
- Drilling the PCB using PCB drill bit and then soldering electronic components on the PCB.
- Finally testing the PCB board whether it is working as intended or not.

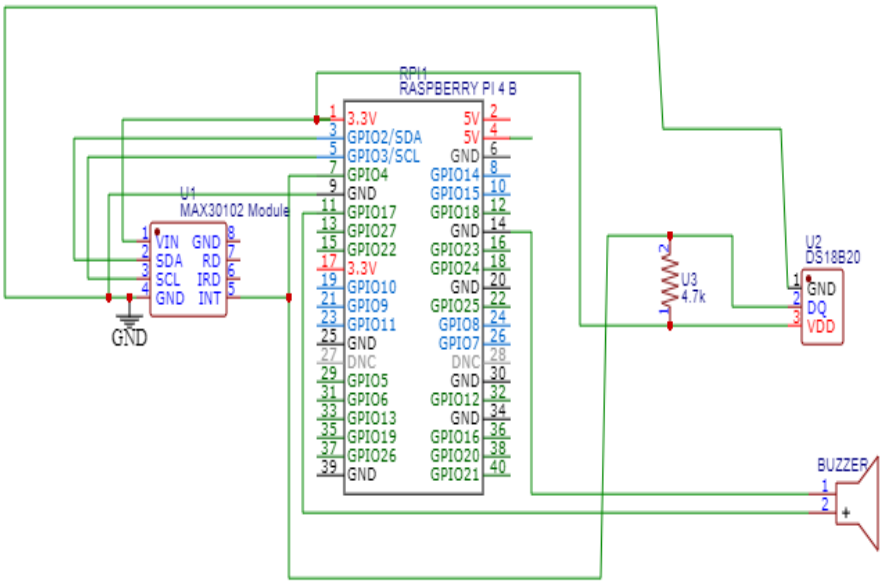


Figure 3.10: Schematic diagram of IoT based patient monitoring system

Chapter 4 Results and Discussion

4.1 Circuit implementation

After the complete PCB design, we reconfigured the entire circuit from bread board to the PCB. Following the implementation of the circuit including interfacing of the sensors to Raspberry Pi via PCB, the overall system for monitoring was successfully established as shown in the figure. By enabling I2C and one wire interface communication in Raspberry Pi, MAX30102 and DS18B20 temperature sensor was interfaced with the board. The value of heart rate and SpO2 along with temperature readings were displayed directly. Following figure 4.1 shows the output of PCB layout design. In figure 4.2 a) output of monitored values of different sensors was displayed. Here the obtained data are predicted by EWS value. shows the. Here two different conditions were displayed, if early warning score is less than 3 then the condition is said to be normal and if it is greater or equal to 3 then the buzzer buzzes. In figure 4.2 b) blink indicator is sent to ThingSpeak platform which blinks when early warning score is greater than 3.

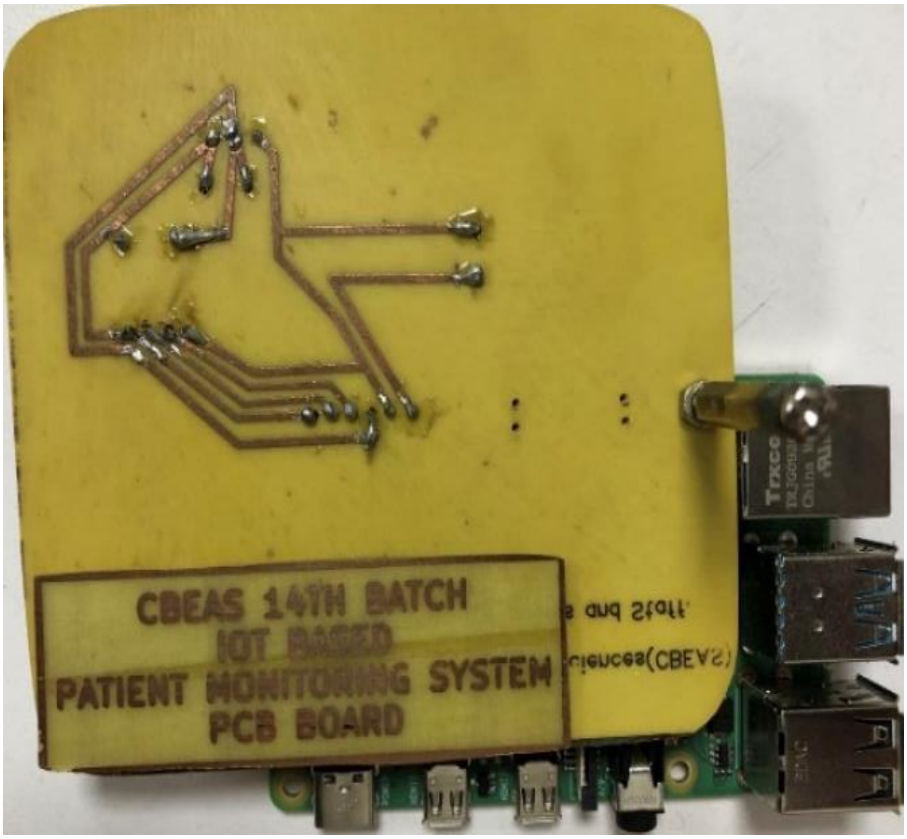
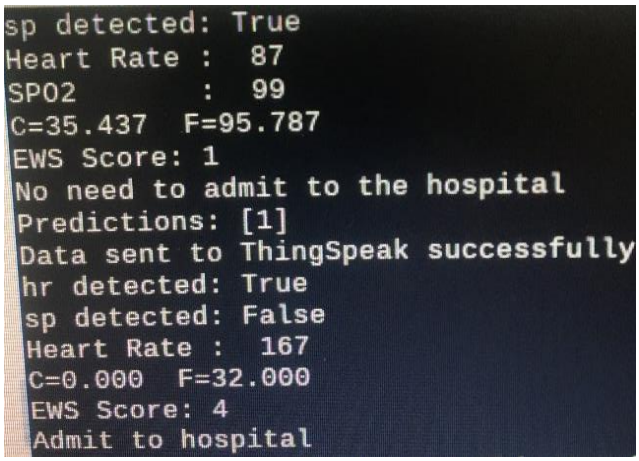


Figure 4. 1: Output of PCB layout design

a)



b)

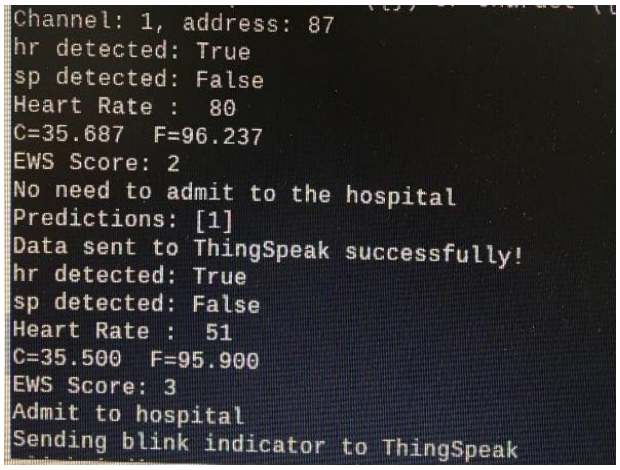


Figure 4. 2: a) Output of monitored value of Temperature, SpO2 and heartrate b) Output of monitored value and blink indicator sent to ThingSpeak

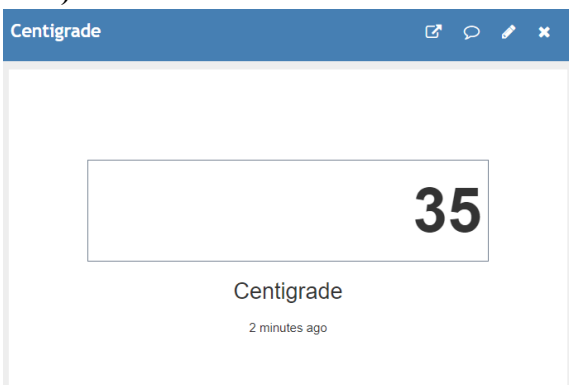
4.2 Results in ThingSpeak platform:

The result is displayed in ThingSpeak platform as shown in the figure below:

We measured the data for two different conditions, for normal and abnormal conditions. For normal condition, In figure 4.3 the value of temperature in centigrade and Fahrenheit is displayed respectively. Similarly, figure 4.4 shows the value of SpO2 in percentage and value of heartrate is displayed and figure 4.5 indicates the predicted data are acceptable so the warning indicator does not blink. For abnormal condition, In figure 4.6 the value of temperature in centigrade and Fahrenheit is displayed respectively. Similarly in figure 4.7 value of SpO2 in percentage and heartrate is displayed respectively. Figure 4.8 outlines the warning sign that signifies the patient needs attention.

4.2.1 Results at normal condition

a)



b)

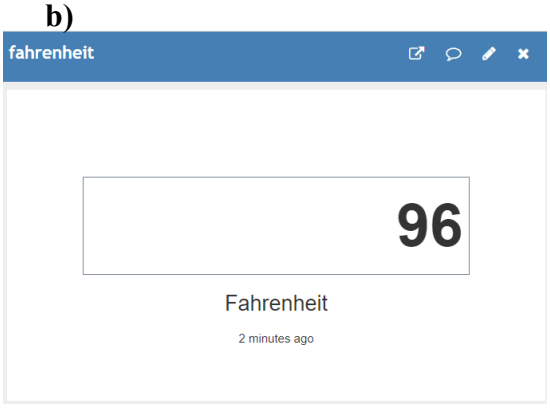


Figure 4. 3: a) Displayed value of temperature in centigrade b) Displayed value of temperature in Fahrenheit

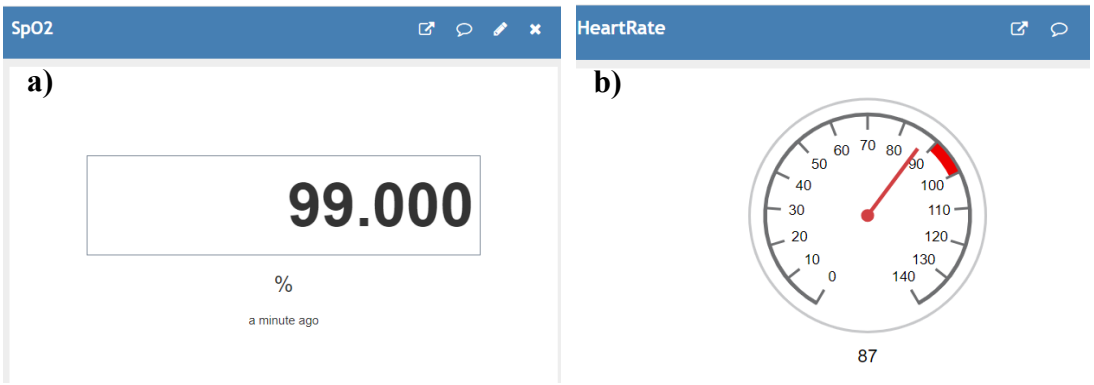


Figure 4. 4: a) Value of SpO2 in percentage b) Value of heartrate in indicator

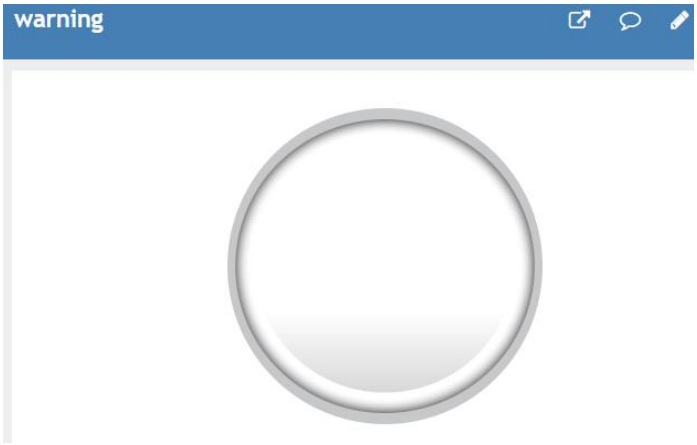


Figure 4. 5: Warning indicator in ThingSpeak

4.2.2 Results at abnormal condition

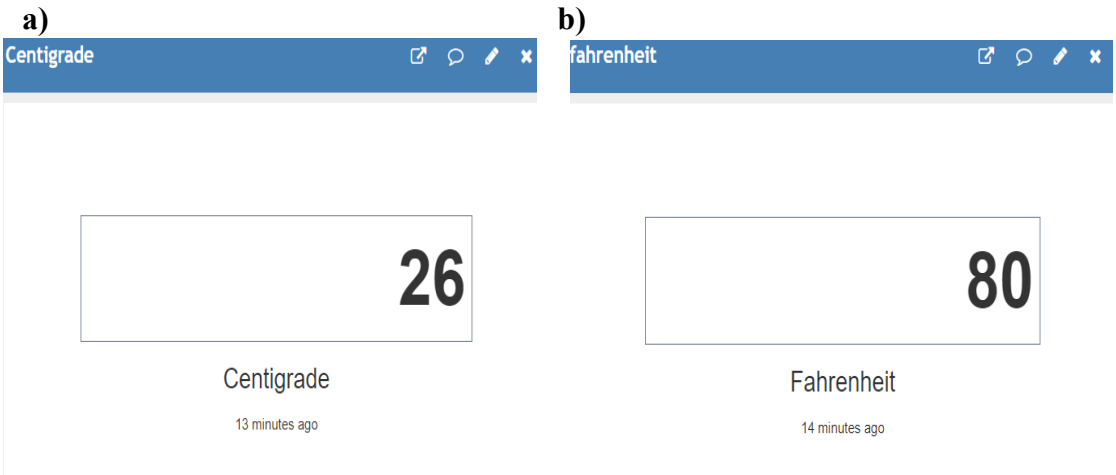


Figure 4.6: a) Displayed value of temperature in Centigrade b) Displayed value of temperature in Fahrenheit

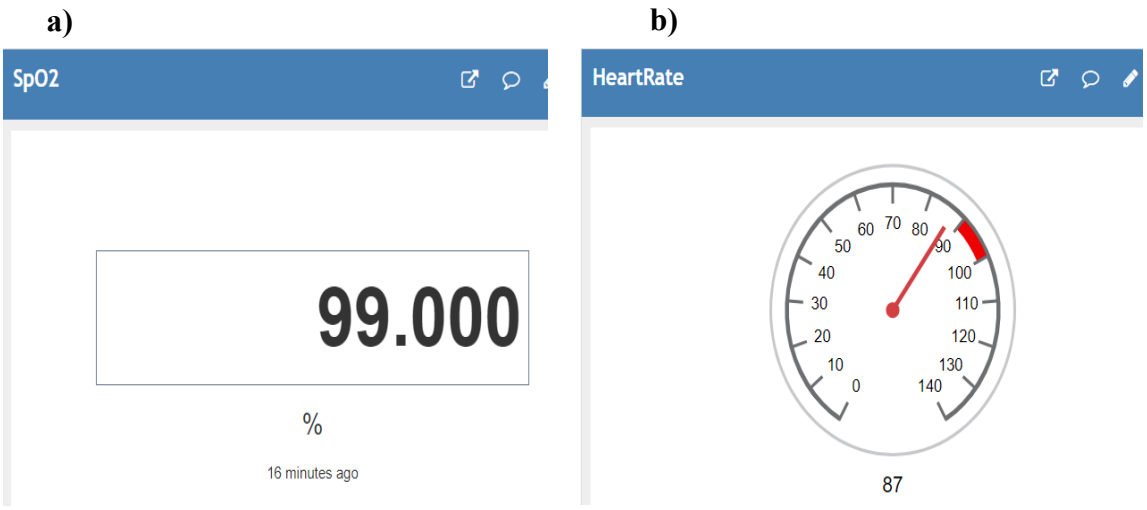


Figure 4. 7: a) Value of SpO2 in percentage b) Value of heartrate in indicator

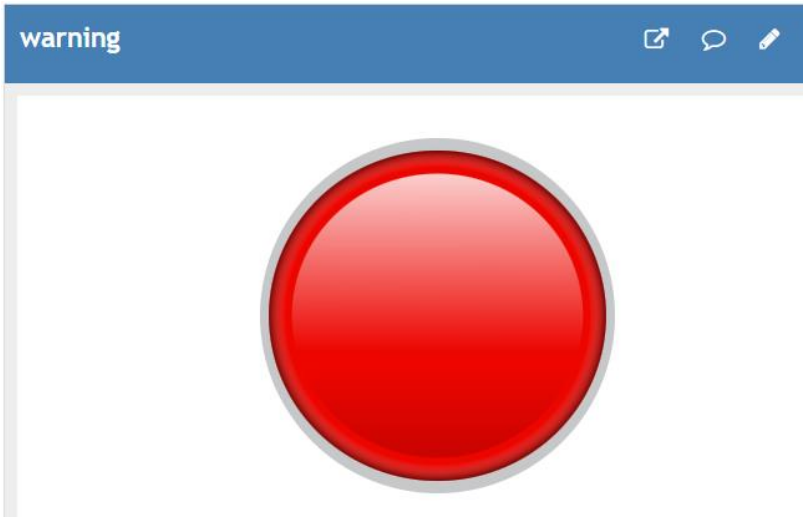


Figure 4. 8: Warning Indicator in ThingSpeak

4.3 Machine learning

Logistic regression is used as machine learning model in our project. It has been designed for the prediction of the data. The accuracy score of each parameter is determined. Accuracy score for each parameters has been generated by logged values and predicted values.

4.3.1 Accuracy of Regression model

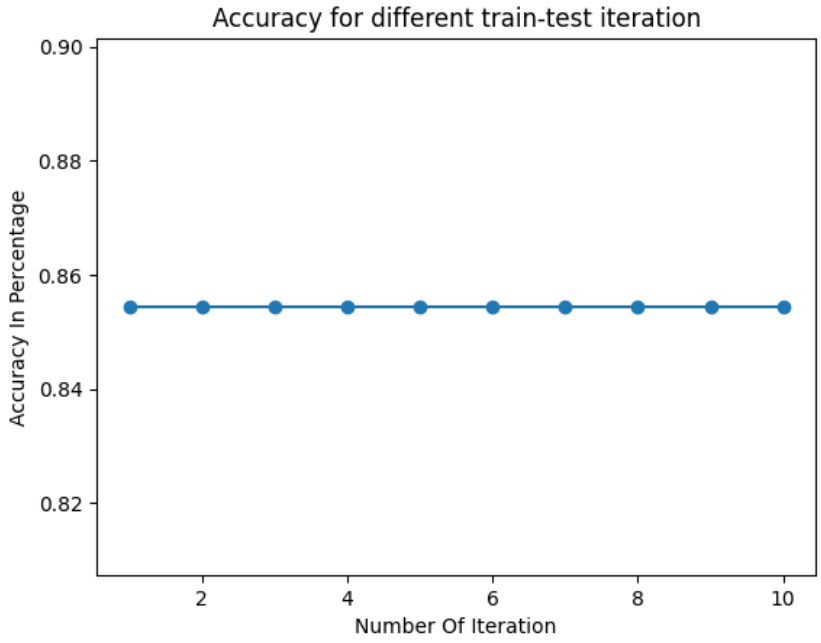


Figure 4. 9: Accuracy score of machine learning model

Figure 4.9 shows the accuracy of Logistic Regression Model. X-axis represents the number of iterations while Y-axis represents the accuracy of model in percentage. While iterating the loop for 10 times, the accuracy of the Logistic Regression Model is found to be 85%.

4.3.2 Correlation Heatmap

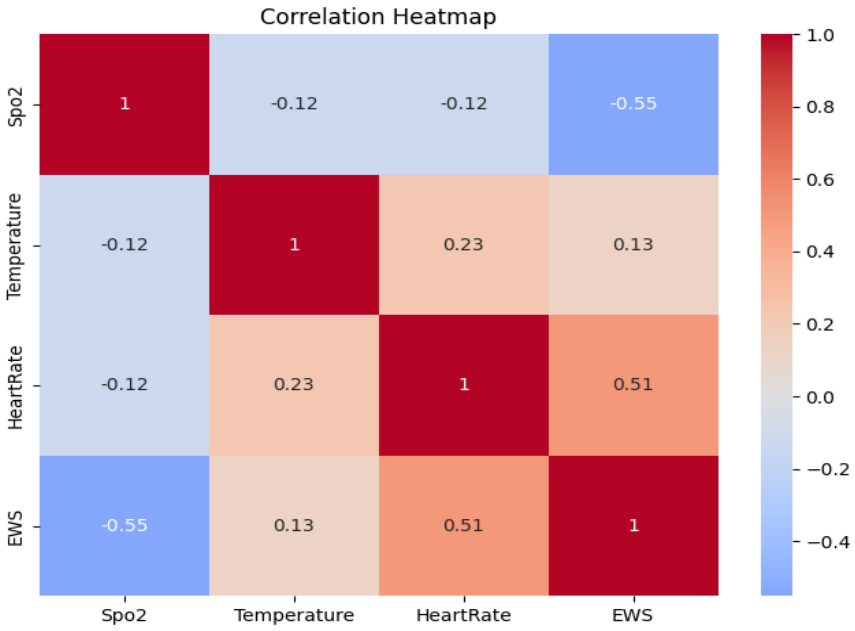


Figure 4. 10: Correlation Heatmap

Figure 4.10 shows a correlation heatmap which is a graphical representation that visualizes the correlations of relationships between variables in a dataset. It measures the statistical association between two variables, indicating how changes in one variable are related to changes in another variable. This map uses color coding to represent the strength and direction of the correlation. The heatmap arranges variables along X and Y axes and fills each cell with a color gradient. The red color is showing danger zone while blue color is showing normal condition of patient. This provides a visual summary of how different variables in the dataset are related to one another.

4.4 Data obtained from hospital PMS V/S project PMS

Our project PMS was compared with Hospital PMS. The obtained data has been tabulated as below. By comparing values between hospital PMS and our project PMS, it has been found that there is fluctuation in heartrate values due to noises.

Table 4. 1: Data comparison of Hospital PMS V/S Project PMS

Time (Min)	Hospital PMS			Project PMS		
	Heart rate	Temperature °C	SpO2	Heart rate	Temperature °C	SpO2
1	68	35	95	64	33	94
2	70	36.7	97	70	35	95
3	72	37	97	78	37	96
4	72	37	98	94	37.5	95
5	75	37	97	116	37.3	94
6	77	37.4	96	71	37.2	98
7	75	36.5	95	80	38.5	97
8	78	38	96	67	39	98
9	79	37	99	72	39.1	97
10	82	37.2	97	102	38.6	96

4.5 Final Designed case of PMS (Patient Monitoring System)

The overall system has been designed in PCB (Printed Circuit Board). Two sensors MAX30102 and DS18B20 have been assembled. Whole system has been integrated inside a case as shown inf figure 4.11.

Figure 4. 11: Case of PMS

4.6 Discussion

Runlinc had been chosen as a microprocess to work upon, as it is easy to operate and has simple coding languages. But, due to its version updating issues it failed to detect the data from sensors. We initially used the MAX30100 sensor, however due to a manufacturing flaw, we had issues with it. We initially attempted cutting the path and then adding external resistor to try and remedy this issue, but both approaches failed. The MAX30102 sensor was then used to replace this sensor. The schematic diagram of the whole circuit was first tried to made on Proteus. But, schematic diagram of Raspberry pi 4 having 40 pins was not available in Proteus. So, the entire circuit was first designed in EasyEDA. The footprint of PCB was created. The Raspberry pi model 3B needed to setup, for that, Raspbian 32 bits OS was installed and then booted into a microSD card. DS18B20 temperature sensor is a digital sensor which measures the temperature values of a patient digitally. The output of all three parameters SpO_2, heart rate and temperature were displayed in a webpage using ThingSpeak IoT platform. Machine learning was implemented on these data. Logistic regression model was used for predicting the data coming from sensors. The model tells if the condition is normal or not. If the values are not in normal range the model warns the user through

warning indicator in ThingSpeak and by beeping the buzzer. Initially, we had also worked on ECG parameter in Arduino UNO. But it shows complexity while embedding in machine learning model as it was showing noises which affects the accuracy of model in high range. The accuracy of the model was found to be 84%. The correlation heatmap was generated which visualize the correlations of relationships between variables in a dataset. The accuracy could be improved by using the improved and ore improved sensors.

Several other monitoring devices were found during literature study. Among those papers, by M. Laghrouche claimed the design of a low-cost patient monitoring device for measurement of arterial blood's oxygen saturation level and heart rate can be used by patient at home safely. Though it was claimed as a low-cost device, but the cost analysis and cost comparison were not shown. Also, the measured data was not compared with that of standard device. It also claimed that the device was lightweight and capable of wireless transmission and has capability of signal and data processing, but all details were not shown in paper. Where, in other hand we have prepared a light weight device that measures real-time data, send the data to ThingSpeak and predicts the conditions of patients [54]. Another paper by Waleed Mohmood Abdullah, Ergun Ercelebi intended design oximeter to monitor oxygen saturation level in real time. They used copies MCU-based ARM cortex 32 bit and different sensors and compared it with the results taken from literature [55]. But they did not explain why and how this method was reliable, compared to which our device not only monitors SpO_2 but also temperature and heart rate of a patient. Another paper proposed by Deivasigamani S., G. Narmadha discuss about interfacing with Internet of Things (IoT) to monitor and alert the blood oxygen saturation from any remote location in real time. The device MAX30102 sensor with Eagle software. However, the cost analysis was not shown in the paper [56]. The cost might be higher due to the enhanced features used in the device, our cost analysis is included in this paper and we have used two sensors that monitors total of three parameters. We also used IoT platform to display the data in a webpage using ThingSpeak.[57]

Turki M. Alanazi, Amani Abdulrahman Albraikan and Faris A. Almalki discusses the creation of an Arduino-based IoT health monitoring system. The system can send data to an app through Bluetooth to measure a patient's body temperature, heart rate, and blood SpO2 levels; however, Wi-Fi was not an option, and a Raspberry Pi could be used in place of the system's Arduino microcontroller [58].

Chapter 5 Conclusion and Further Work

5.1 Conclusion

We have successfully designed IoT based patient monitoring system. We have implemented temperature, SpO_2 and heartrate monitoring circuit that monitors these parameters and displays the real-time values continuously in ThingSpeak. The parameters are being continuously monitored and predicted by Logistic Regression model. The model tells if the values are in normal range or not. If the parameters are not in normal ranges, the model warns the user by warning indicator in the webpage and also alerts the user by beeping the buzzer simultaneously. The accuracy graph and correlation heatmap have been generated that shows the accuracy of the system and visualize the correlations of relationships between variables in a dataset respectively.

5.2 Further works

The number of parameters that are being monitored could be increased for better monitoring of patient. Direct call to the ambulance can be added for easy access of service. Advanced sensors could be used for more accurate data. Other vital parameters could be added which are specific to certain diseases/conditions. The accuracy of the system could be increased using other better machine learning models.

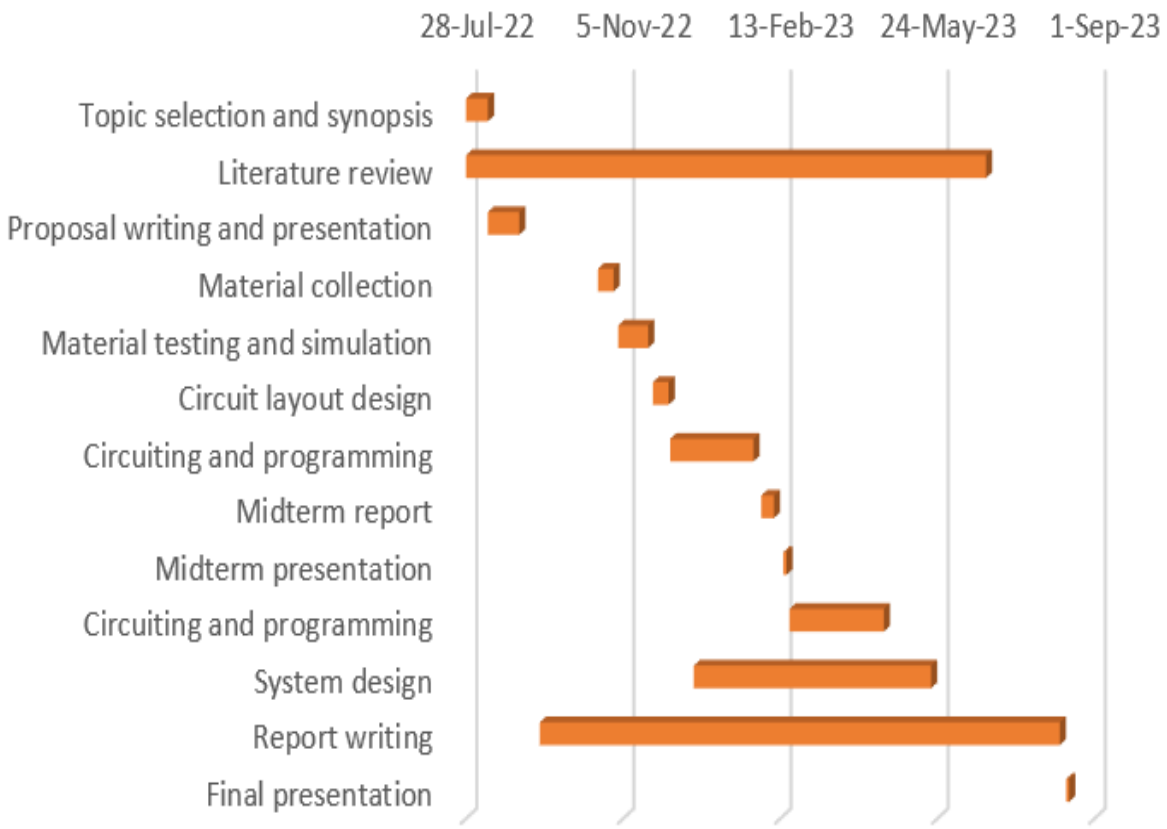
5.3 Limitations

Though the system design meets all of its objectives and safe to use but it hasn't reached to the potential it could have achieved. There are various reasons due to which the rooms for further improvements are left among them. It could not work on ECG as we had expected and aimed initially due to its noise issues and complexity while working in machine learning model. The system could also be designed to make an ambulance call directly in case of an emergency and send the message directly to WhatsApp for an easy access of information to doctor and patient. But, due to time limitation we could not work upon these parameters.

Project Cost

SN	Components	Quantity	Price
1	Raspberry Pi	1	9000
2	AD8232 ECG sensor	1	2100
3	MAX30100 heart rate sensor	2	1350
4	DS18B20 temperature sensor	1	250
5	MAX30102 heart rate sensor	2	4600
6	Disposal electrodes	50	1800
7	Runlinc	2	18500
8	ESP8266 WiFi module	1	800
9	Pulse sensor	1	500
10	Breadboard big	1	140
11	VGA to HDMI cable	1	1200
12	SpO2 Sensor	1	4500
	Total		44,740

Gantt Chart


28-Jul-22
5-Nov-22
13-Feb-23
24-May-23
1-Sep-23
Topic selection and synopsis
Literature review
Proposal writing and presentation
Material collection
Material testing and simulation
Circuit layout design
Circuiting and programming
Midterm report
Midterm presentation
Circuiting and programming
System design
Report writing
Final presentation

References

[1] V. Bhardwaj, R. Joshi, and A. M. Gaur, "IoT-Based Smart Health Monitoring System for COVID-19," *SN Comput Sci*, vol. 3, no. 2, Mar. 2022, doi: 10.1007/S42979-022-01015-1.

[2] N. S. Mohamad Hadis, M. N. Amirnazarullah, M. M. Jafri, and S. Abdullah, "IoT Based Patient Monitoring System using Sensors to Detect, Analyse and Monitor Two Primary Vital Signs," *J Phys Conf Ser*, vol. 1535, no. 1, Jun. 2020, doi: 10.1088/1742-6596/1535/1/012004.

[3] M. Hamim, S. Paul, S. I. Hoque, M. N. Rahman, and I. Al Baqee, "IoT Based remote health monitoring system for patients and elderly people," *1st International Conference on Robotics, Electrical and Signal Processing Techniques, ICREST 2019*, pp. 533–538, Feb. 2019, doi: 10.1109/ICREST.2019.8644514.

[4] M. M. Khan, T. M. Alanazi, A. A. Albraikan, and F. A. Almalki, "IoT-Based Health Monitoring System Development and Analysis," *Security and Communication Networks*, vol. 2022, 2022, doi: 10.1155/2022/9639195.

[5] Ö. ÇELİK, "A Research on Machine Learning Methods and Its Applications," *Journal of Educational Technology and Online Learning*, vol. 1, no. 3, pp. 25–40, Sep. 2018, doi: 10.31681/JETOL.457046.

[6] M. Wazid, A. K. Das, V. Chamola, and Y. Park, "Uniting cyber security and machine learning: Advantages, challenges and future research," *ICT Express*, vol. 8, no. 3, pp. 313–321, Sep. 2022, doi: 10.1016/j.icte.2022.04.007.

[7] "Overview of datasets involved in a machine learning diagnostic... | Download Scientific Diagram." https://www.researchgate.net/figure/Overview-of-datasets-involved-in-a-machine-learning-diagnostic-algorithm-model_fig1_339224791 (accessed Sep. 07, 2023).

[8] A. Vanarse, A. Osseiran, and A. Rassau, "A Review of Current Neuromorphic Approaches for Vision, Auditory, and Olfactory Sensors," *Front Neurosci*, vol. 10, no. MAR, p. 115, Mar. 2016, doi: 10.3389/fnins.2016.00115.

[9] A. Sharma, S. Sharma, and D. Gupta, "A Review of Sensors and Their Application in Internet of Things (IOT)," *Int J Comput Appl*, vol. 174, no. 24, pp. 27–34, Mar. 2021, doi: 10.5120/IJCA2021921148.

[10] F. Wen, T. He, H. Liu, H. Y. Chen, T. Zhang, and C. Lee, "Advances in chemical sensing technology for enabling the next-generation self-sustainable integrated wearable system in the IoT era," *Nano Energy*, vol. 78, Dec. 2020, doi: 10.1016/j.nanoen.2020.105155.

[11] I. J. Brekke, L. H. Puntervoll, P. B. Pedersen, J. Kellett, and M. Brabrand, "The value of vital sign trends in predicting and monitoring clinical deterioration: A systematic review," *PLoS One*, vol. 14, no. 1, Jan. 2019, doi: 10.1371/JOURNAL.PONE.0210875.

[12] M. Elliott and A. Coventry, "Critical care: The eight vital signs of patient monitoring," *British Journal of Nursing*, vol. 21, no. 10, pp. 621–625, May 2012, doi: 10.12968/BJON.2012.21.10.621.

[13] P. Hoyland *et al.*, "A Paced-ECG Detector and Delineator for Automatic Multi-Parametric Catheter Mapping of Ventricular Tachycardia," *IEEE Access*, vol. 8, pp. 223952–223960, 2020, doi: 10.1109/ACCESS.2020.3043542.

[14] "Electrode placement for three-lead ECG measurement. | Download Scientific Diagram." https://www.researchgate.net/figure/Electrode-placement-for-three-lead-ECG-measurement_fig2_280769273 (accessed Sep. 07, 2023).

[15] B. B. Hafen and S. Sharma, "Oxygen Saturation," *Oxygen Saturation - StatPearls - NCBI Bookshelf (nih.gov)* , pp. 4–9, Nov. 2022, Accessed: Sep. 07, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK525974/

[16] J. C. P. Harper *et al.*, "Determination of oxygen saturation compared to a prescribed target range using continuous pulse oximetry in acutely unwell medical patients," *BMC Pulm Med*, vol. 21, no. 1, pp. 1–9, Dec. 2021, doi: 10.1186/S12890-021-01700-6/TABLES/4.

[17] R. M. Gardner and M. M. Shabot, "Patient-Monitoring Systems," pp. 585–625, 2006, doi: 10.1007/0-387-36278-9_17.

[18] E. Lam, S. Aratia, J. Wang, and J. Tung, "Measuring Heart Rate Variability in Free-Living Conditions Using Consumer-Grade Photoplethysmography: Validation Study," *JMIR Biomed Eng*, vol. 5, no. 1, p. e17355, Nov. 2020, doi: 10.2196/17355.

[19] W. M. Jubadi and S. F. A. M. Sahak, "Heartbeat monitoring alert via SMS," *2009 IEEE Symposium on Industrial Electronics & Applications*, vol. 1, pp. 1–5, Dec. 2009, doi: 10.1109/ISIEA.2009.5356491.

[20] A. Garami and M. Székely, “Body temperature: Its regulation in framework of energy balance,” *Temperature*, vol. 1, no. 1, pp. 28–29, Jun. 2014, doi: 10.4161/TEMP.29060/SUPPL_FILE/KTMP_A_10929060_SM0001.ZIP.

[21] R. M. Gardner and M. M. Shabot, “Patient-Monitoring Systems”.

[22] R. M. Gardner and M. M. Shabot, “Patient-Monitoring Systems”.

[23] K. Boikanyo, A. M. Zungeru, B. Sigweni, A. Yahya, and C. Lebekwe, “Remote patient monitoring systems: Applications, architecture, and challenges,” *Sci Afr*, vol. 20, p. e01638, Jul. 2023, doi: 10.1016/J.SCIAF.2023.E01638.

[24] M. M. Hassan, H. S. Albakr, and H. Al-Dossari, “A cloud-assisted Internet of Things framework for pervasive healthcare in smart city environment,” *EMASC 2014 - Proceedings of the 1st International Workshop on Emerging Multimedia Applications and Services for Smart Cities, Workshop of MM 2014*, pp. 9–13, Nov. 2014, doi: 10.1145/2661704.2661707.

[25] “Monitor Medical History: Founding, Timeline, and Milestones - Zippia.” https://www.zippia.com/monitor-medical-careers-474169/history/ (accessed Jun. 01, 2023).

[26] A. Archip, N. Botezatu, E. Șerban, P. C. Herghelegiu, and A. ZalĂ, “An IoT based system for remote patient monitoring,” *Proceedings of the 2016 17th International Carpathian Control Conference, ICCC 2016*, pp. 1–6, Jun. 2016, doi: 10.1109/CARPATHIANCC.2016.7501056.

[27] B. on H. C. Services and I. of Medicine, “The Evolution of Telehealth: Where Have We Been and Where Are We Going?,” Nov. 2012, Accessed: Sep. 08, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK207141/

[28] P. J. Pronovost, M. D. Cole, and R. M. Hughes, “Remote Patient Monitoring During COVID-19: An Unexpected Patient Safety Benefit,” *JAMA*, vol. 327, no. 12, pp. 1125–1126, Mar. 2022, doi: 10.1001/JAMA.2022.2040.

[29] Y. Pole, “Evolution of the pulse oximeter,” *Int Congr Ser*, vol. 1242, no. C, pp. 137–144, Dec. 2002, doi: 10.1016/S0531-5131(02)00803-8.

[30] I. Al Baqee, M. N. Rahman, and I.- Al-Baqee, “IoT Based Remote Health Monitoring System for Patients and Elderly People,” *2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST)*, 2019, doi: 10.1109/ICREST.2019.8644514.

[31] "(1) Design of Iot based smart health monitoring and alert system | Request PDF." https://www.researchgate.net/publication/311950986_Design_of_Iot_based_smart_health_monitoring_and_alert_system (accessed Jun. 01, 2023).

[32] N. S. Mohamad Hadis, M. N. Amirnazarullah, M. M. Jafri, and S. Abdullah, "IoT Based Patient Monitoring System using Sensors to Detect, Analyse and Monitor Two Primary Vital Signs," *J Phys Conf Ser*, vol. 1535, no. 1, p. 012004, May 2020, doi: 10.1088/1742-6596/1535/1/012004.

[33] M. Weenk *et al.*, "Continuous monitoring of vital signs using wearable devices on the general ward: Pilot study," *JMIR Mhealth Uhealth*, vol. 5, no. 7, Jul. 2017, doi: 10.2196/MHEALTH.7208.

[34] V. Vujovic, N. Davidović, and B. Perisic, *ELI1.6 Maksimovic Vujovic Davidovic Milosevic Perisic*. 2015. [Online]. Available: https://www.researchgate.net/publication/280344140

[35] A. N. Ansari, M. Sedky, N. Sharma, and A. Tyagi, "An Internet of things approach for motion detection using Raspberry Pi," *Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things, ICIT 2015*, pp. 131–134, May 2015, doi: 10.1109/ICAIOT.2015.7111554.

[36] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable AI: A Review of Machine Learning Interpretability Methods," *Entropy 2021, Vol. 23, Page 18*, vol. 23, no. 1, p. 18, Dec. 2020, doi: 10.3390/E23010018.

[37] Z. Kang, C. Catal, and B. Tekinerdogan, "Machine learning applications in production lines: A systematic literature review," *Comput Ind Eng*, vol. 149, p. 106773, Nov. 2020, doi: 10.1016/J.CIE.2020.106773.

[38] J. Alzubi, A. Nayyar, and A. Kumar, "Machine Learning from Theory to Algorithms: An Overview," *J Phys Conf Ser*, vol. 1142, no. 1, p. 012012, Nov. 2018, doi: 10.1088/1742-6596/1142/1/012012.

[39] D. Nettleton, "Data Modeling," *Commercial Data Mining*, pp. 137–157, 2014, doi: 10.1016/B978-0-12-416602-8.00009-1.

[40] M. E. B. Smith *et al.*, "BACKGROUND," 2014, Accessed: Jun. 01, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK259029/

[41] D. R. Prytherch, G. B. Smith, P. E. Schmidt, and P. I. Featherstone, "ViEWS—Towards a national early warning score for detecting adult inpatient deterioration,"

Resuscitation, vol. 81, no. 8, pp. 932–937, Aug. 2010, doi: 10.1016/J.RESUSCITATION.2010.04.014.

[42] A. Janowczyk and A. Madabhushi, "Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases," *J Pathol Inform*, vol. 7, no. 1, p. 29, Jan. 2016, doi: 10.4103/2153-3539.186902.

[43] A. Al-Omary, H. M. AlSabbagh, and H. Al-Rizzo, "Cloud based IoT for smart garden watering system using Arduino Uno," *IET Conference Publications*, vol. 2018, no. CP747, 2018, doi: 10.1049/CP.2018.1401/CITE/REFWORKS.

[44] "Android Arduino Control: Android ESP8266 IoT ThingSpeak Sensor Data Monitor." http://androidcontrol.blogspot.com/2016/04/android-esp8266-iot-thingspeak-sensor.html (accessed Jun. 02, 2023).

[45] Z. Runjing, X. Hongwei, and R. Guanzhong, "Design of temperature measurement system consisted of FPGA and DS18B20," *Proceedings - 2011 International Symposium on Computer Science and Society, ISCCS 2011*, pp. 90–93, 2011, doi: 10.1109/ISCCS.2011.32.

[46] U. Jovanovic, I. Jovanovic, and D. Mancic, "Overview of Temperature Sensors for Temperature Measurement of PV Modules," *2018 26th Telecommunications Forum, TELFOR 2018 - Proceedings*, 2018, doi: 10.1109/TELFOR.2018.8612096.

[47] Ramesh Saha, S. Biswas, S. Sarmah, S. Karmakar, and P. Das, "A Working Prototype Using DS18B20 Temperature Sensor and Arduino for Health Monitoring," *SN Comput Sci*, vol. 2, no. 1, pp. 1–21, Feb. 2021, doi: 10.1007/S42979-020-00434-2/FIGURES/26.

[48] Z. Jincheng, L. Yanfei, Z. Boyuan, and W. Kai, "Design and implementation of wearable oxygen saturation monitoring system," *2021 IEEE International Conference on Electrical Engineering and Mechatronics Technology, ICEEMT 2021*, pp. 71–74, 2021, doi: 10.1109/ICEEMT52412.2021.9601533.

[49] A. R. Dhruba, K. N. Alam, M. S. Khan, S. Bourouis, and M. M. Khan, "Development of an IoT-Based Sleep Apnea Monitoring System for Healthcare Applications," *Comput Math Methods Med*, vol. 2021, 2021, doi: 10.1155/2021/7152576.

[50] W. Gay, "The Raspberry Pi," *Advanced Raspberry Pi*, pp. 1–13, 2018, doi: 10.1007/978-1-4842-3948-3_1.

[51] M. Ali *et al.*, "An IoT based Approach for Efficient Home Automation with ThingSpeak," 2020. [Online]. Available: www.ijacsa.thesai.org

[52] T. Carneiro, R. V. M. Da Nobrega, T. Nepomuceno, G. Bin Bian, V. H. C. De Albuquerque, and P. P. R. Filho, "Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications," *IEEE Access*, vol. 6, pp. 61677–61685, 2018, doi: 10.1109/ACCESS.2018.2874767.

[53] T. Carneiro, R. V. M. Da Nobrega, T. Nepomuceno, G. Bin Bian, V. H. C. De Albuquerque, and P. P. R. Filho, "Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications," *IEEE Access*, vol. 6, pp. 61677–61685, 2018, doi: 10.1109/ACCESS.2018.2874767.

[54] M. Laghrouche, S. Haddab, S. Lotmani, K. Mekdoud, and S. Ameur, "Low-cost embedded oximeter," *Measurement Science Review*, vol. 10, no. 5, pp. 176–179, Jan. 2010, doi: 10.2478/V10048-010-0030-6.

[55] W. M. Abdullah and E. Ercelebi, "DEVELOPMENT OF PULSE OXIMETER BY USING 32-BIT ARM BASED MICROCONTROLLER," 2018. [Online]. Available: http://ijieee.org.in

[56] G. Narmadha, M. Ramasamy, H. Prasad, and P. Nair, "Design of Smart Pulse Oximeter using ATMEGA 328 Microcontroller," *International Journal on Emerging Technologies*, vol. 11, no. 3, pp. 696–700, 2020, [Online]. Available: www.researchtrend.net

[57] V. Bhardwaj, R. Joshi, and A. M. Gaur, "IoT-Based Smart Health Monitoring System for COVID-19," *SN Comput Sci*, vol. 3, no. 2, Mar. 2022, doi: 10.1007/S42979-022-01015-1.

[58] M. M. Khan, T. M. Alanazi, A. A. Albraikan, and F. A. Almalki, "IoT-Based Health Monitoring System Development and Analysis," *Security and Communication Networks*, vol. 2022, 2022, doi: 10.1155/2022/9639195.

Appendices

Appendix 1

Code for Temperature Sensor

```
import os
import glob
import time

# These tow lines mount the device:
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
# Get all the filenames begin with 28 in the path base_dir.
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'
def read_rom():
    name_file=device_folder+'/name'
    f = open(name_file,'r')
    return f.readline()

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines
def read_temp():
    lines = read_temp_raw()
    # Analyze if the last 3 characters are 'YES'.
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    # Find the index of 't=' in a string.
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        # Read the temperature .
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c, temp_f

print(' rom: '+ read_rom())
```

```
while True:
   print(' C=%3.3f  F=%3.3f'% read_temp())
time.sleep(1)
```

Code for MAX30102

```
import max30102
import hrcalc

m = max30102.MAX30102()

hr2 = 0
sp2 = 0

while True:
   red, ir = m.read_sequential()

   hr,hrb,sp,spb = hrcalc.calc_hr_and_spo2(ir, red)

   print("hr detected:",hrb)
   print("sp detected:",spb)

   if(hrb == True and hr != -999):
      hr2 = int(hr)
      print("Heart Rate : ",hr2)
   if(spb == True and sp != -999):
      sp2 = int(sp)
      print("SPO2       : ",sp2)
```

Code for Buzzer

```
import RPi.GPIO as GPIO
import time

BuzzerPin = 18

GPIO.setmode(GPIO.BCM)
GPIO.setup(BuzzerPin, GPIO.OUT)
GPIO.setwarnings(False)

global Buzz
Buzz = GPIO.PWM(BuzzerPin, 440)
Buzz.start(50)

B0=31
C1=33
```

CS1=35
D1=37
DS1=39
E1=41
F1=44
FS1=46
G1=49
GS1=52
A1=55
AS1=58
B1=62
C2=65
CS2=69
D2=73
DS2=78
E2=82
F2=87
FS2=93
G2=98
GS2=104
A2=110
AS2=117
B2=123
C3=131
CS3=139
D3=147
DS3=156
E3=165
F3=175
FS3=185
G3=196
GS3=208
A3=220
AS3=233
B3=247
C4=262
CS4=277
D4=294
DS4=311
E4=330
F4=349
FS4=370
G4=392

GS4=415
A4=440
AS4=466
B4=494
C5=523
CS5=554D5=587
DS5=622
E5=659
F5=698
FS5=740
G5=784
GS5=831
A5=880
AS5=932
B5=988
C6=1047
CS6=1109
D6=1175
DS6=1245
E6=1319
F6=1397
FS6=1480
G6=1568
GS6=1661
A6=1760
AS6=1865
B6=1976
C7=2093
CS7=2217
D7=2349
DS7=2489
E7=2637
F7=2794
FS7=2960
G7=3136
GS7=3322
A7=3520
AS7=3729
B7=3951
C8=4186
CS8=4435
D8=4699

```
DS8=4978

song = [
  G4,
  E4, F4, G4, G4, G4,
  A4, B4, C5, C5, C5,
  E4, F4, G4, G4, G4,
  A4, G4, F4, F4,
  E4, G4, C4, E4,
  D4, F4, B3,
  C4
]

beat = [
  8,
  8, 8, 4, 4, 4,
  8, 8, 4, 4, 4,
  8, 8, 4, 4, 4,
  8, 8, 4, 2,
  4, 4, 4, 4,
  4, 2, 4,
  1
]

while True:
    for i in range(1, len(song)):
        Buzz.ChangeFrequency(song[i])
        time.sleep(beat[i] * 0.5)  # Adjust the sleep time to match the beat duration

        # After playing the note, check if 10 seconds have passed
        # If yes, stop the buzzer and exit the loop
        start_time = time.time()
        while time.time() - start_time < 10:
            pass
        Buzz.stop()
        GPIO.cleanup()
                Exit()
```

Code for Thingspeak

```
import os
import glob
import time
import max30102
import hrcalc
```

```
import requests

# These two lines mount the device:
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'

if len(glob.glob(base_dir + '28*')) == 0:
    print("No DS18B20 sensor found")
    exit()

device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

# ThingSpeak configuration
THINGSPEAK_API_KEY = 'CB44XQDOIYIH6BFS'
THINGSPEAK_UPDATE_URL                                                    =
f'https://api.thingspeak.com/update?api_key=CB44XQDOIYIH6BFS&field1=0'

def read_rom():
    name_file = device_folder + '/name'
    f = open(name_file, 'r')
    return f.readline()

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    # Analyze if the last 3 characters are 'YES'.
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    # Find the index of 't=' in a string.
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        # Read the temperature.
        temp_string = lines[1][equals_pos+2:]
```

```
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c, temp_f

m = max30102.MAX30102()

hr2 = 0
sp2 = 0

while True:
    red, ir = m.read_sequential()
    hr, hrb, sp, spb = hrcalc.calc_hr_and_spo2(ir, red)

    print("hr detected:", hrb)
    print("sp detected:", spb)

    if hrb == True and hr != -999:
        hr2 = int(hr)
        print("Heart Rate : ", hr2)
    if spb == True and sp != -999:
        sp2 = int(sp)
        print("SPO2       : ", sp2)

    temp_c, temp_f = read_temp()
    print('C=%3.3f  F=%3.3f' % (temp_c, temp_f))

    # Sending data to ThingSpeak
    payload = {'field1': temp_c, 'field2': temp_f, 'field3': hr2, 'field4': sp2}
    response = requests.get(THINGSPEAK_UPDATE_URL, params=payload)
    if response.status_code == 200:
        print("Data sent to ThingSpeak successfully!")
    else:
        print("Failed to send data to ThingSpeak.")

    time.sleep(1)
```

Code for Machine Learning

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import joblib
```

```
# Read dataset from CSV file
dataset = pd.read_csv("/content/gdrive/MyDrive/icudata/icudata.csv")

EWS_Scores = []
for index, row in dataset.iterrows():
  row_ews = []

  if pd.notnull(row['Spo2']):
    if row['Spo2'] >= 96:
      row_ews.append(0)
    elif row['Spo2'] >= 94 and row['Spo2'] <= 95:
      row_ews.append(1)
    elif row['Spo2'] >= 92 and row['Spo2'] <= 93:
      row_ews.append(2)
    elif row['Spo2'] <= 91:
      row_ews.append(3)
  else:
    row_ews.append(-1)

  if pd.notnull(row['HeartRate']):
    if row['HeartRate'] >= 51 and row['HeartRate'] <= 90:
      row_ews.append(0)
    elif (row['HeartRate'] >= 41 and row['HeartRate'] <= 50) or (row['HeartRate'] >= 91
and row['HeartRate'] <= 110):
      row_ews.append(1)
    elif row['HeartRate'] >= 111 and row['HeartRate'] <= 130:
      row_ews.append(2)
    elif row['HeartRate'] <= 40 or row['HeartRate'] >= 131:
      row_ews.append(3)
  else:
    row_ews.append(-1)

  if pd.notnull(row['Temperature']):
    if row['Temperature'] >= 36.1 and row['Temperature'] <= 38.0:
      row_ews.append(0)
    elif (row['Temperature'] >= 35.1 and row['Temperature'] <= 36) or
(row['Temperature'] >= 38.1 and row['Temperature'] <= 39.0):
      row_ews.append(1)
    elif row['Temperature'] >= 39.1:
      row_ews.append(2)
    elif row['Temperature'] <= 35:
      row_ews.append(3)
```

```
    else:
        row_ews.append(-1)

    if len(row_ews) == 3:
        EWS_Scores.append(sum(row_ews))
    else:
        EWS_Scores.append(-1)

# Add EWS scores to the dataset
dataset['EWS'] = EWS_Scores
#Extract features and labels from the dataset
spo2_data = dataset["Spo2"].values
temperature_dat a = dataset["Temperature"].values
heart_rate_data = dataset["HeartRate"].values
ews_scores = dataset["EWS"].values
icu_labels = dataset["ICU_Admission"].values

# Initialize a list to store accuracies
accuracies = []

# Model evaluation
for _ in range(10):
    # Splitting the dataset
    X_train, X_test, y_train, y_test, ews_train, ews_test = train_test_split(
        np.column_stack((spo2_data, temperature_data, heart_rate_data)),
        icu_labels,
        ews_scores,
        test_size=0.2,
        random_state=42
    )

    # Training the model
    model = LogisticRegression()
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)

print("Accuracies:", accuracies)

# Plot the accuracy graph
plt.plot(range(1, len(accuracies) + 1), accuracies, marker='o')
```

```
plt.xlabel('Number Of Iteration')
plt.ylabel('Accuracy In Percentage')
plt.title('Accuracy for different train-test iteration')
plt.show()

joblib.dump(model, "/content/gdrive/MyDrive/icudata/model_1.pkl")

# Load the saved model
loaded_model = joblib.load("/content/gdrive/MyDrive/icudata/model_1.pkl")

# Deployment and prediction
real_time_spo2 = np.array([93, 91, 89])
real_time_temperature = np.array([99.1, 99.3, 98.8])
real_time_heart_rate = np.array([82, 72, 91])
real_time_ews = np.array([5, 6, 4])

# Preprocess the real-time data
preprocessed_data = np.column_stack((real_time_spo2, real_time_temperature,
real_time_heart_rate))

# Make predictions using the trained model
predictions = loaded_model.predict(preprocessed_data)

# Decide admission based on EWS score
for i in range(len(predictions)):
    if real_time_ews[i] >= 3:  # Define your EWS threshold
        print("Admit to hospital")
    else:
        print("No need for hospital admission")

# Print the predictions
print("Predictions:", predictions)
```

Code for overall circuit implementation

```
import os
import glob
import time
import max30102
import hrcalc
import requests
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.metrics import accuracy_score
import joblib

# These two lines mount the device:
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'

if len(glob.glob(base_dir + '28*')) == 0:
   print("No DS18B20 sensor found")
   exit()

device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

# ThingSpeak configuration
THINGSPEAK_API_KEY = 'CB44XQDOIYIH6BFS'
THINGSPEAK_UPDATE_URL                                                    =
f'https://api.thingspeak.com/update?api_key=CB44XQDOIYIH6BFS&field1=0'

def read_rom():
   name_file = device_folder + '/name'
   f = open(name_file, 'r')
   return f.readline()

def read_temp_raw():
   f = open(device_file, 'r')
   lines = f.readlines()
   f.close()
   return lines
def read_temp():
   lines = read_temp_raw()
   # Analyze if the last 3 characters are 'YES'.
   while lines[0].strip()[-3:] != 'YES':
      time.sleep(0.2)
      lines = read_temp_raw()
   # Find the index of 't=' in a string.
   equals_pos = lines[1].find('t=')
   if equals_pos != -1:
      # Read the temperature.
      temp_string = lines[1][equals_pos+2:]
      temp_c = float(temp_string) / 1000.0
```

```
        temp_f = temp_c * 9.0 / 5.0 + 32.0
        return temp_c, temp_f
def get_thingspeak_data(channel_id, api_key, num_points):
  url = f'https://api.thingspeak.com/channels/2030562/feeds.json?results=1'
  response = requests.get(url)
  data = response.json()
  field_name = 'field1'
  temperature = []
  heartrate = []
  spo2 = []
  for entry in data['feeds']:
    spo2.append(float(entry['field4']))
    heartrate.append(float(entry['field3']))
    temperature.append(float(entry['field2']))
  return spo2, heartrate, temperature

def buzz():
    #!/usr/bin/env python
    import RPi.GPIO as GPIO
    import time
    BuzzPin = 11 # Raspberry Pi Pin 17-GPIO 17
    def setup(pin):
        global BuzzerPin
        BuzzerPin = pin
        GPIO.setmode(GPIO.BOARD) # Set GPIO Pin As Numbering
        GPIO.setup(BuzzerPin, GPIO.OUT)
        GPIO.output(BuzzerPin, GPIO.HIGH)

    def on():
        GPIO.output(BuzzerPin, GPIO.LOW)

    def off():
        GPIO.output(BuzzerPin, GPIO.HIGH)

    def beep(x):
        on()
        time.sleep(x)
        off()
        time.sleep(x)

    def loop():
        while True:
```

```
        beep(0.5)
    def destroy():
        GPIO.output(BuzzerPin, GPIO.HIGH)
        GPIO.cleanup() # Release resource

    if __name__ == '__main__': # Program start from here
        setup(BuzzPin)
    try:
        for i in range(len(predictions)

if real_time_ews[i] >= 3:  # Define your EWS threshold
            print("EWS Score:", real_time_ews[i])
            print("Admit to hospital")
            loop()
            continue

        else:
            print("EWS Score:", real_time_ews[i])
            print("No need to admit to the hospital")
    except KeyboardInterrupt: # When 'Ctrl+C' is pressed, the child program destroy() will
be executed.
        destroy()

m = max30102.MAX30102()

hr2 = 0
sp2 = 0

# Load the saved model
model_file_path = "/home/pi/raspberry pythons/icudata/model_1.pkl"
loaded_model = joblib.load(model_file_path)

# Constants for scoring
NORMAL_RANGE = {
    'heart_rate': (60, 100),            # bpm (beats per minute)
    'oxygen_saturation': (95, 100),     # percentage
    'temperature': (36.1, 37.8),        # degrees Celsius
}

# Assign scores for each parameter based on deviation from normal range def
assign_score(parameter, value):
lower, upper = NORMAL_RANGE[parameter]
    if lower <= value <= upper:
```

```
        return 0
    elif value < lower:
        return 1
    else:
        return 2
while True:
    red, ir = m.read_sequential()
    hr, hrb, sp, spb = hrcalc.calc_hr_and_spo2(ir, red)

    print("hr detected:", hrb)
    print("sp detected:", spb)

    if hrb == True and hr != -999:
            hr2 = int(hr)
            print("Heart Rate : ", hr2)
    else:
        hr2 = 0

    if spb == True and sp != -999:
        sp2 = int(sp)
        print("SPO2       : ", sp2)
    else:
        sp2 = 0

    temp_c, temp_f = read_temp()
    print('C=%3.3f  F=%3.3f' % (temp_c, temp_f))

    # Deployment and prediction
    #real_time_spo2 = np.array([sp2])
    #real_time_temperature = np.array([temp_c])
    # Calculate EWS scores based on the deviation from normal ranges
    hr_score = assign_score('heart_rate', hr2)
    spO2_score = assign_score('oxygen_saturation', sp2)
    temp_score = assign_score('temperature', temp_c)
    #real_time_heart_rate = np.array([hr2])
    #real_time_ews = np.array([sp2 + hr2])  # Assuming ews score is the sum of SPO2 and
Heart Rate
    # Calculate the EWS score by summing the individual scores
    real_time_ews = np.array([hr_score + spO2_score + temp_score])
    #real_time_ews = np.array([2])
    channel_id = '2030562'
    api_key = 'VTPT8ZZSGGUQRPIX'
```

```
    num_points = 100
    real_time_spo2, real_time_temperature,        real_time_heart_rate
            =get_thingspeak_data(channel_id, api_key, num_points)

    # Preprocess the real-time data
    preprocessed_data =         np.column_stack((real_time_spo2,real_time_temperature,
real_time_heart_rate))

    # Make predictions using the trained model
    predictions = loaded_model.predict(preprocessed_data)

    for i in range(len(predictions)):
        if real_time_ews[i] >= 3:
            print("EWS Score:", real_time_ews[i])
            print("Admit to hospital")
            if predictions[i] == 1:
                print("Sending blink indicator to ThingSpeak")
                # Sending blink indicator to ThingSpeak (field 5)

   blink_payload = {'field5': 1}
    blink_response = requests.get(THINGSPEAK_UPDATE_URL, params=blink_payload)
     if blink_response.status_code == 200:
      print("Blink indicator sent to ThingSpeak successfully!")
                else:
                    print("Failed to send blink indicator to ThingSpeak.")
            buzz()
            time.sleep(5)
        else:
            print("EWS Score:", real_time_ews[i])
            print("No need to admit to the hospital")

    # Print the predictions
    print("Predictions:", predictions)

    # Sending data to ThingSpeak
    payload = {'field1': temp_c, 'field2': temp_f, 'field3': hr2, 'field4': sp2}
    response = requests.get(THINGSPEAK_UPDATE_URL, params=payload)
    if response.status_code == 200:
        print("Data sent to ThingSpeak successfully!")
    else:
        print("Failed to send data to ThingSpeak.")

    time.sleep(1)
```

Appendix 2

Datasheet for MAX30102 sensor

MAX30102

High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health

General Description

The MAX30102 is an integrated pulse oximetry and heart-rate monitor module. It includes internal LEDs, photodetectors, optical elements, and low-noise electronics with ambient light rejection. The MAX30102 provides a complete system solution to ease the design-in process for mobile and wearable devices.

The MAX30102 operates on a single 1.8V power supply and a separate 3.3V power supply for the internal LEDs. Communication is through a standard I²C-compatible interface. The module can be shut down through software with zero standby current, allowing the power rails to remain powered at all times.

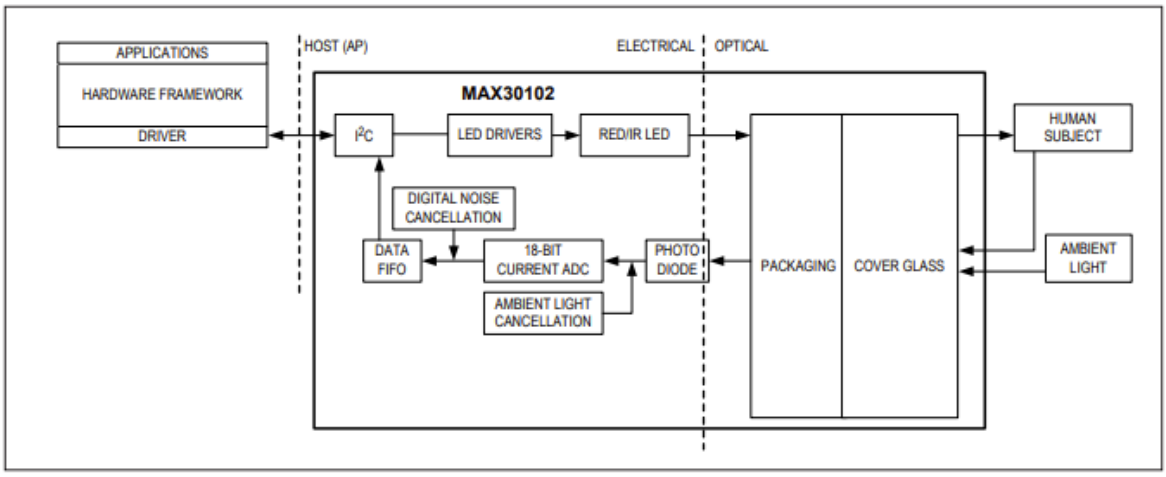
Applications

- Wearable Devices
- Fitness Assistant Devices
- Smartphones
- Tablets

Benefits and Features

- Heart-Rate Monitor and Pulse Oximeter Sensor in LED Reflective Solution
- Tiny 5.6mm x 3.3mm x 1.55mm 14-Pin Optical Module
  - Integrated Cover Glass for Optimal, Robust Performance
- Ultra-Low Power Operation for Mobile Devices
  - Programmable Sample Rate and LED Current for Power Savings
  - Low-Power Heart-Rate Monitor (< 1mW)
  - Ultra-Low Shutdown Current (0.7µA, typ)
- Fast Data Output Capability
  - High Sample Rates
- Robust Motion Artifact Resilience
  - High SNR
- -40°C to +85°C Operating Temperature Range

System Diagram



19-7740; Rev 1; 10/18

maxim integrated

Absolute Maximum Ratings

V_{DD} to GND .....-0.3V to +2.2V
GND to PGND.....-0.3V to +0.3V
V_{LED+} to PGND.....-0.3V to +6.0V
All Other Pins to GND .....-0.3V to +6.0V
Output Short-Circuit Current Duration.....Continuous
Continuous Input Current into Any Terminal .....±20mA
ESD, Human Body Model (HBM).....2.5kV
Latchup Immunity.....±250mA

Continuous Power Dissipation (T_A = +70°C)
OESIP (derate 5.5mW/°C above +70°C) 440mW
Operating Temperature Range..... -40°C to +85°C
Junction Temperature.....+90°C
Soldering Temperature (reflow) .....+260°C
Storage Temperature Range..... -40°C to +105°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only; functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Package Information

PACKAGE TYPE: 14 OESIP	
Package Code	F143A5MK+1
Outline Number	21-1048
Land Pattern Number	90-0602
THERMAL RESISTANCE, FOUR-LAYER BOARD	
Junction to Ambient (θ_{JA})	180°C/W
Junction to Case (θ_{JC})	150°C/W

Package thermal resistances were obtained using the method described in JEDEC specification JESD51-7, using a four-layer board. For detailed information on package thermal considerations, refer to www.maximintegrated.com/thermal-tutorial.

For the latest package outline information and land patterns (footprints), go to www.maximintegrated.com/packages. Note that a "+", "#", or "-" in the package code indicates RoHS status only. Package drawings may show a different suffix character, but the drawing pertains to the package regardless of RoHS status.

Electrical Characteristics

(V_{DD} = 1.8V, V_{LED+} = 5.0V, T_A = -40°C to +85°C, unless otherwise noted. Typical values are at T_A = +25°C) (Note 1)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
POWER SUPPLY							
Power-Supply Voltage	V_{DD}	Guaranteed by RED and IR count tolerance		1.7	1.8	2.0	V
LED Supply Voltage V_{LED+} to PGND	V_{LED+}	Guaranteed by PSRR of LED driver		3.1	3.3	5.0	V
Supply Current	I_{DD}	SpO_2 and HR mode, PW = 215µs, 50sps			600	1200	µA
		IR only mode, PW = 215µS, 50sps			600	1200	
Supply Current in Shutdown	I_{SHDN}	T_A = +25°C, MODE = 0x80			0.7	10	µA
ADC Resolution					18		bits
Red ADC Count (Note 2)	REDC	LED1_PA = 0x0C, LED_PW = 0x01, SPO2_SR = 0x05, ADC_RGE = 0x00			65536		Counts
IR ADC Count (Note 2)	IRC	LED2_PA = 0x0C, LED_PW = 0x01, SPO2_SR = 0x05 ADC_RGE = 0x00			65536		Counts
Dark Current Count	LED_DCC	LED1_PA = LED2_PA = 0x00, LED_PW = 0x03, SPO2_SR = 0x01 ADC_RGE = 0x02			30	128	Counts
					0.01	0.05	% of FS
DC Ambient Light Rejection	ALR	ADC counts with finger on sensor under direct sunlight (100K lux), ADC_RGE = 0x3, LED_PW = 0x03, SPO2_SR = 0x01	Red LED		2		Counts
			IR LED		2		Counts
ADC Count—PSRR (V_{DD})	$PSRRV_{DD}$	1.7V < V_{DD} < 2.0V, LED_PW = 0x01, SPO2_SR = 0x05			0.25	1	% of FS
		Frequency = DC to 100kHz, 100mV_{P-P}			10		LSB
ADC Count—PSRR (LED Driver Outputs)	$PSRR_{LED}$	3.1V < V_{LED+}, < 5.0V, LED1_PA = LED2_PA = 0x0C, LED_PW = 0x01, SPO2_SR = 0x05			0.05	1	% of FS
		Frequency = DC to 100kHz, 100mV_{P-P}			10		LSB
ADC Clock Frequency	CLK			10.32	10.48	10.64	MHz

Datasheet for RaspberryPi 4

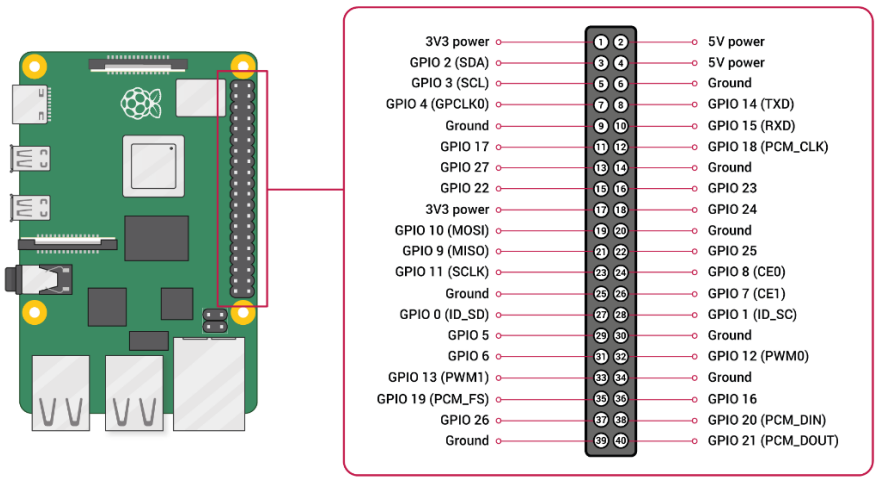
2 Features

2.1 Hardware

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
- 1, 2 and 4 Gigabyte LPDDR4 RAM options
- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- VideoCore VI 3D Graphics
- Supports dual HDMI display output up to 4Kp60

2.2 Interfaces

- 802.11 b/g/n/ac Wireless LAN
- Bluetooth 5.0 with BLE
- 1x SD Card
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)
- 1x Raspberry Pi camera port (2-lane MIPI CSI)
- 1x Raspberry Pi display port (2-lane MIPI DSI)
- 28x user GPIO supporting various interface options:
  - Up to 6x UART
  - Up to 6x I2C
  - Up to 5x SPI
  - 1x SDIO interface
  - 1x DPI (Parallel RGB Display)



Datasheet for DS18B20 Temperature sensor

DS18B20

Programmable Resolution 1-Wire Digital Thermometer

General Description

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

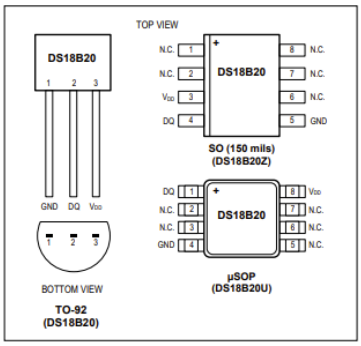
Applications

- Thermostatic Controls
- Industrial Systems
- Consumer Products
- Thermometers
- Thermally Sensitive Systems

Benefits and Features

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
  - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
  - ±0.5°C Accuracy from -10°C to +85°C
  - Programmable Resolution from 9 Bits to 12 Bits
  - No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
  - Each Device Has a Unique 64-Bit Serial Code Stored in On-Board ROM
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages

Pin Configurations



Ordering Information ***appears at end of data sheet.***

1-Wire is a registered trademark of Maxim Integrated Products, Inc.

Absolute Maximum Ratings

Voltage Range on Any Pin Relative to Ground ..... -0.5V to +6.0V
Operating Temperature Range ..... -55°C to +125°C
Storage Temperature Range ..... -55°C to +125°C
Solder Temperature ..... Refer to the IPC/JEDEC J-STD-020 Specification.

These are stress ratings only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

DC Electrical Characteristics

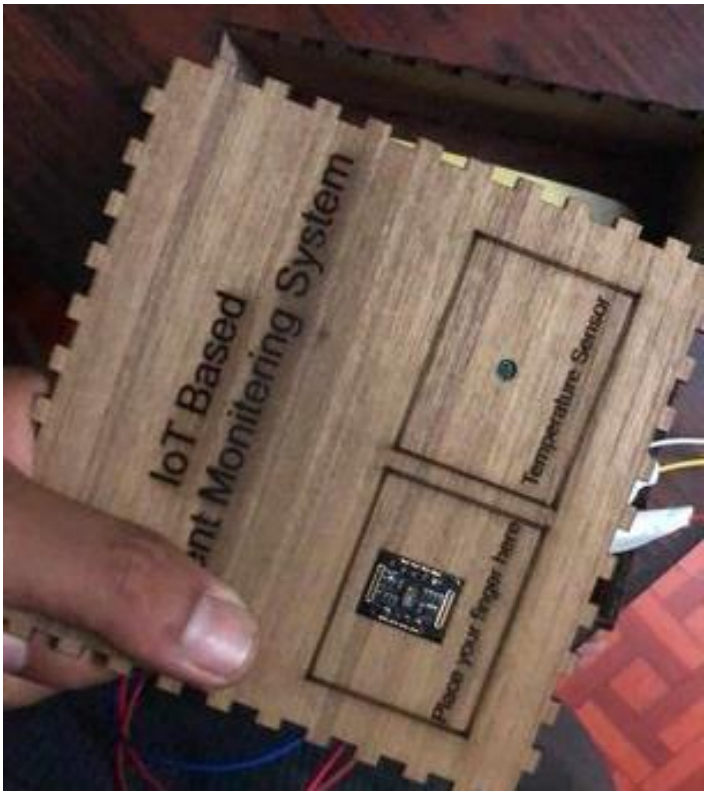
(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Supply Voltage	V_{DD}	Local power (Note 1)		+3.0		+5.5	V
Pullup Supply Voltage	V_{PU}	Parasite power	(Notes 1, 2)	+3.0		+5.5	V
		Local power		+3.0		V_{DD}	
Thermometer Error	t_{ERR}	-10°C to +85°C	(Note 3)			±0.5	°C
		-30°C to +100°C				±1	
		-55°C to +125°C				±2	
Input Logic-Low	V_{IL}	(Notes 1, 4, 5)		-0.3		+0.8	V
Input Logic-High	V_{IH}	Local power	(Notes 1,6)	+2.2		The lower of 5.5 or V_{DD} + 0.3	V
		Parasite power		+3.0			
Sink Current	I_L	$V_{I/O}$ = 0.4V		4.0			mA
Standby Current	I_{DDS}	(Notes 7, 8)			750	1000	nA
Active Current	I_{DD}	V_{DD} = 5V (Note 9)			1	1.5	mA
DQ Input Current	I_{DQ}	(Note 10)			5		µA
Drift		(Note 11)			±0.2		°C

Note 1: All voltages are referenced to ground.
Note 2: The Pullup Supply Voltage specification assumes that the pullup device is ideal, and therefore the high level of the pullup is equal to V_{PU}. In order to meet the V_{IH} spec of the DS18B20, the actual supply rail for the strong pullup transistor must include margin for the voltage drop across the transistor when it is turned on; thus: $V_{PU\_ACTUAL} = V_{PU\_IDEAL} + V_{TRANSISTOR}$.
Note 3: See typical performance curve in Figure 1. Thermometer Error limits are 3-sigma values.
Note 4: Logic-low voltages are specified at a sink current of 4mA.
Note 5: To guarantee a presence pulse under low voltage parasite power conditions, V_{ILMAX} may have to be reduced to as low as 0.5V.
Note 6: Logic-high voltages are specified at a source current of 1mA.
Note 7: Standby current specified up to +70°C. Standby current typically is 3µA at +125°C.
Note 8: To minimize I_{DDS}, DQ should be within the following ranges: GND ≤ DQ ≤ GND + 0.3V or V_{DD} – 0.3V ≤ DQ ≤ V_{DD}.
Note 9: Active current refers to supply current during active temperature conversions or EEPROM writes.
Note 10: DQ line is high ("high-Z" state).
Note 11: Drift data is based on a 1000-hour stress test at +125°C with V_{DD} = 5.5V.

Appendix 3


A6103
ELECTRIC DRILL


IoT Based
Temperature Sensor