IoT Based Health Monitoring System for Pregnant Women & Children

By

Fariha Islam Ohana
18121028
Nabita Rahmina
18110003
Farin Ahmed
18121039
Anindita Basak
18121002

A thesis submitted to the Department of Electrical and Electronic Engineering in partial fulfillment of the requirements for the degree of
B.Sc. in Electrical & Electronic Engineering

Department of Electrical and Electronic Engineering
BRAC University
September 2021

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. I/We have acknowledged all main sources of help.

Student's Full Name & Signature:

_____	_____
Fariha Islam Ohana	**Anindita Basak**
18121028	18121002
_____	_____
Nabita Rahmina	**Farin Ahmed**
18110003	18121039

Approval

The thesis/project titled "IoT Based Health Monitoring System for Pregnant Women & Children" submitted by

1. Fariha Islam Ohana (18121028)
2. Nabita Rahmina (18110003)
3. Farin Ahmed (18121039)
4. Anindita Basak (18121002)

of Summer, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Electrical & Electronic Engineering on 3rd October 2021.

Examining Committee:

Supervisor:
(Member)

_____

Abu S.M. Mohsin, Ph.D.
Assistant Professor
Department of Electrical & Electronics Engineering
Brac University

Program Coordinator:
(Member)

_____

Abu S.M. Mohsin, Ph.D.
Assistant Professor
Department of Electrical & Electronics Engineering
Brac University

Departmental Head:
(Chair)

_____

Md. Mosaddequr Rahman, Ph.D.
Professor & Chairperson
Department of Electrical & Electronics Engineering
Brac University

Abstract/ Executive Summary

This research inspects the monitoring of pregnant women and children using the Internet of Things in the modern healthcare system. The main focus is to ensure the safety of pregnant women and their children. Through our current project, we put forward a system that can firmly influence the standard of living for pregnant women and children positively. Our IoT-based device can monitor temperature & humidity, stress, and fetal heart rate. We can also use this device to monitor a child's health parameters. The microcontroller collects all the data of these parameters with the help of sensors and sends these data to the ThingSpeak server via Wi-Fi module ESP8266 and further analyzes it. This system is fully automated to be able to notify the user and an emergency contact in case of emergency whenever any sensor threshold value crosses. Our second device, which incorporates fall detection along with the monitoring of the pregnant woman's heart rate and SpO2, is integrated with a vaccination reminder system that will notify the user about the immunization dates and the vaccine to be provided on a particular day. Through the different sensors incorporated in the devices, we can get all the specific health-related parameters of the pregnant woman and her child. The motive is to unobtrusively obtain crucial information about the health status of pregnant women and their children.

Keywords: IoT; IoT health monitoring system; pregnant women and child; fall detection; vaccination reminder.

Acknowledgement

We are thankful to the Almighty Allah for his infinite amount of blessings and mercy that made us complete the project smoothly.

We would like to express our heartfelt gratitude towards our supervisor Dr. Abu S.M. Mohsin, Assistant Professor, Department of Electrical and Electronic Engineering for his ceaseless motivation and support throughout the project.

Last but not the least, we would like to appreciate our parents for their endless support and love all over this long journey.

Table of Content

List of Tables

List of Figures

List of Acronyms

IoT	Internet of Things
ML	Machine Learning
GSR	Galvanic Skin Response
LCD	Liquid Crystal Display
LAN	Local Area Network
GSM	Global System for Mobile Communications
GPRS	General Packet Radio Service
MCU	Microcontroller Unit
ECG	Electrocardiogram
HTTP	Hypertext Transfer Protocol
MQTT	MQ Telemetry Transport
UART	Universal asynchronous receiver-transmitter
TCP/IP	Transmission Control Protocol/Internet Protocol
I2C	Inter-Integrated Circuit
LED	Light Emitting Diode
SOC	System on chip
Bpm	Beats per minute
SPO2	Oxygen Saturation
ALC	Ambient Light Cancellation
UID	Unique Identification
API	Application Programming Interface

Chapter 1

Introduction

1.1 Background

Time and distance both are crucial factors for pregnant women in critical situations. According to our research, the physical distance between pregnant women and health care providers should not be a limiting factor in the quality of treatment they get. According to WHO, 800 women die every day as a result of the complexities of childbirth. A pregnant woman from a growing nation is 36 times more likely to experience pregnancy-related problems compared with a pregnant woman from a highly established state[1]. Complications during labor and childbirth account for more than a third of maternal fatalities, half of all stillbirths, and a quarter of neonatal mortality. The bulk of these deaths occur in low-resource settings and may be avoided if timely interventions were implemented. It is crucial to know when to wait and when to take life-saving measures[2]. Children under 5 years are also vulnerable to many infectious diseases. Hence, it is very much necessary to monitor all the health care parameters of a pregnant woman and her child at all times to decrease the possibility of maternal and fetal complexities and chronic illnesses.

This paper assesses the use of sensors and the internet of things as an enabling technology to monitor maternal and fetal behaviors during pregnancy. This illustrates and expands new options for remotely monitoring health problems and tracking health parameters in everyday life. Through the use of these various sensors, we are creating an unruffled lifestyle for both the mother and the fetus throughout the pregnancy period. This process has a significant potential to improve the early detection of pregnancy problems with high accuracy. Our technological solution has a subsequent influence on clinical decision-making for the mother and the fetus.

Moreover, our system will provide regular reminders of the vaccination of the child after birth. We evaluated the benefits of using the various sensors, the wearable health-tracking devices, and the types of health monitoring data that the users would want to observe regularly.

Numerous IoT-based projects consisting of biomedical signal acquisition devices, wearable sensor technologies, pregnancy, and infant monitoring systems have been largely covered in the Literature Review.

1.2 Motivation & Objective behind this paper

Bangladesh, a country located in South Asia has the lowest levels of antenatal care. Antenatal care is essential for protecting the health of women and their unborn children[3]. According to UNFPA, Bangladesh has made significant efforts to improve mother and neonatal health. in collaboration with the government and civil society partners. Throughout the last 14 years, a lot of exceptional advancement has been made globally: the under-five mortality rate has decreased from 90 deaths per 1,000 live births in 1990 to 48 deaths per 1,000 in 2012. Even Bangladesh distinctively has made an immense improvement: the under-five mortality rate has been lessened down from 239 per 1000 in 1970 to 65 per 1000 in 2007, 75% of newborns are fully vaccinated by the time they cross one year, along with a considerable depletion in the maternal mortality ratio by 40%. However, some weighty challenges still persist: half of all maternal deaths take place due to situations that are avoidable under the observation of trained supervision, such as hemorrhage. The majority of these maternal deaths which take place due to the deficiency of proper resource settings are mendable through timely medications. The need for maternal health care experts is attained by only 41%. Tragically, an average of 14 infants less than one month old die every hour; most of these infant death cases take place in the home[4]. According to World Data Atlas, between 2003 and 2017, the maternal mortality ratio of Bangladesh was decreasing at a medium rate to contract from 395 deaths per 100,000 live births in 2003 to 173 deaths per 100,000 live births in 2017[5].

These situations are triggered by many factors, one of them being the physical distance between the pregnant woman and the health-care services. In critical conditions, distance and time both are pivotal factors for a pregnant woman. The situation of knowing when to hold up and when to take life-saving measures is hazardous.

This ongoing crisis has highly inspired our project development. Our goal is to monitor all the health care parameters of a pregnant woman and her child at every moment using a system to lower the risk of complications for both mother and fetus, as well as chronic illnesses.

1.3 Risks associated with different health parameters

Body temperature is one of the most important health indicators. A person's typical body temperature varies from 36.1°C to 37.2°C. A temperature over 100.4°F (38°C) can be the indication of several human diseases and illnesses such as ear, skin, lungs, throat, bladder, or kidney infection, rheumatoid arthritis, hormone abnormalities such as hyperthyroidism and a variety of other underlying ailments [6][7][8]. An extremely low body temperature can also be potentially deadly because it causes shock in the heart, neurological system, and other organs, putting the individual at danger of a heart attack, respiratory failure, and death [9]. Too much humidity causes the body temperature to rise and even aggravates the growth and spread of viruses,

bacteria, and fungus thus causing allergic reactions[10]. High blood pressure and heart disease can be caused by chronic high amounts of stress. Stress can raise the chances of having a preterm or low-birthweight baby during pregnancy, and premature newborns are more vulnerable to health issues[11]. Heart rate and blood oxygen saturation (SpO2) are yet another two important factors of the human body. A high heart rate causes the heart to pump less efficiently causing a reduction in the blood flow to the heart and the rest of the body and since the heart is beating faster the heart muscles need more oxygen and an inability to supply more oxygen causes lightheadedness and shortness of breath and in extreme cases causes heart attack[12][13]. A low heart rate causes the brain and the organs to be deprived of oxygen thus leading to fatigue, dizziness, memory difficulties, etc[14]. In our proposed system we will be monitoring these vital parameters, along with other factors, so that such health concerns can be addressed as early as possible.

1.4 Literature review

We all know that a pregnant woman and her child are susceptible to numerous complications. The physical condition of the mother remains quite delicate at that time, a slight deviation from the normal condition can lead to maternal or fetal death and in many cases both. Therefore, numerous IoT projects have been built to monitor the health condition of pregnant women, fetuses, and newborn babies. Xin Zhao, Xianyi Zeng, Ludovic Koehl, Guillaume Tartare, Julien D Jonckheere, Kehui Song[15] in their paper have designed an intelligent garment consisting of a large scale accelerometer for data collection and an embedded system for signal processing and machine learning, and an Android-based local monitoring platform for evaluation of fetal health status based on information obtained from the garment via Bluetooth and if any abnormal situation occurs, the android platform alarms the mother. L K Hema, Amalna, Athira Anil, and Kanya[16] have presented a device that is used to measure the fetal heart rate during pregnancy. The major component used for this detection is the Fetal Digital stethoscope sensor which is placed on the abdomen of the pregnant woman and the signals are processed by the microcontroller and the accurate fetal heart rate is identified and sent as a text message to the respective mobile phone by using GSM module. The EMG sensor can be utilized to simulate uterus contraction as the output on the desktop. Santhi S, Gandhi AP, Geetha M[17] have developed a device which monitors a pregnant woman imminent to delivery and aids in giving instructions during operation, in rural areas, by video conferencing with the physicians at hospitals. Their system uses a mobile cognitive radio and wireless sensors and the recurrent data are stored in cloud storage as the medical history. George K. Endo, Ibukun Oluwayomi, Victor Alexandra, Yashodhan Athavale, and Sridhar Krishnan[18] in their paper have proposed a system in where they continuously obtain essential health parameters of the pregnant woman using the sensors present on smartphones and an accessory biomedical signal acquisition device. These data after being processed and classified are then evaluated by doctors to know the condition of pregnant women. Luis M. Borges, Norberto Barroca, Fernando J. Velez, Antonio S. Lebres[19] have developed a wearable monitoring belt

consisting of several flex sensors for continuous monitoring of fetal movements. Risa Hiyama, Mondo Saito, Yasuto Nakanishi, Yuko Hirose, Sho Arisumi[20], in their paper have put forward a wearable device, consisting of sensors, for pregnant women that will monitor rough external pushes, kicks of an unborn baby, and the physical communication between a mother and a child. Moreover, the wearable device acts as a shield against sudden pushes and bumps thus reducing the stress of the mothers regarding these matters. As a result, this improves the overall quality of life for mothers and the fetus during pregnancy. Fatemah Sarhaddi, Iman Azimi, Sina Labbaf, Hannakaisa Niela-Vilen, Nikil Dutt, Anna Axelin, Pasi Liljeberg, and Amir M.Rahmani[21] have presented a system based on the Internet-of-Things (IoT) to provide omnipresent maternal health monitoring during pregnancy and after delivery. The system is made up of various data collectors that keep tabs on the health of the mother, including her stress levels, sleep patterns, and physical activity. Trishita Ghosh Troyee, Md.Kaiser Raihan, Mohammed Shahriar Arefin[22] have designed a system comprising of non-invasive anemia and glucose rate detection, sudden fall detection, and heart rate and body temperature monitoring of the expecting mother and sending those data to the ThingSpeak server. They have designed their system using NIR LED for glucose rate detection, ADXL345, a 3-axis accelerometer, for fall detection, MLX90614, MAX30100 sensors for measuring body temperature and heart rate respectively, and Wemos D1 Mini as the MCU. Despite the fact that their system can detect several essential characteristics, it lacks the ability to detect stress, which is one of the most significant variables in a pregnant woman.

Nour Mahmoud Bahbouh, Abdallah Namoun, Ahmad B. Alkhodre, Sami S. Albouq, Adnan Ahmed Abi Sen[23] have designed a system that works together with a mobile phone app to monitor children during sleep by taking images or by detecting sound, and if a problem arises it alerts the deaf parents by vibrating a wearable bracelet. T M Kadarina and R Priambodo[24] in their paper have described an IoT-based real-time heart rate and blood oxygen saturation monitoring system, for mother and child, using a sensor module and raspberry pi. These data are then uploaded to the Thingsboard IoT platform. P.Poonkuzhlai, R.Aarthi, Yaazhini.V.M, Yuvashri.S, Vidyalakshmi.G[25] have presented a system that continuously monitors health parameters of the child such as temperature, respiration rate, and heart rate alongside tracking their location for safety purposes. A smart child tracking and monitoring system is provided by this system.

Although there are plenty of publications on health monitoring devices for pregnant women and children, none of them incorporates all the features that we have integrated into our system design. Our device will comprise all the measurable perceptions that can be detected and felt during pregnancy and minimize the number of cases that might create complications during birth. This will include measuring the vital signs of the mother and child with additional software that will remind the mother of her newborn child’s immunization dates. This system will benefit a lot of children and pregnant women by ensuring proper health conditions and mitigating all the adverse complications ensuring safe pregnancies in our country.

1.5 Introduction to IoT

As our whole device is focused on IoT, it is important that we know how this technology works, its architectural layers, and the network topologies involved with it. Only then we can continue to proceed with our motive behind this project and make it a success.

In this 21st century, the Internet of things, or IoT, is one of the most beneficial technologies to human beings. It is a system of digital and automatic machines, interlinked computing gadgets, people, animals, or objects that are allocated with different unique identifiers (UIDs). UIDs provide them the potentiality to communicate real-time data and transmit information over a web without the need for human-to-computer or human-to-human interactivity[26][27]. IoT concerns billions of physical gadgets all collecting and sharing data all over the map by simply being linked to the internet. Almost any physical item can be metamorphosed into an IoT gadget if it can be controlled or can transfer data through internet connectivity. Using wireless networks and affordable computer chips, it's feasible to turn something as small as a capsule to something as mammoth as an airship, into a part of the IoT. The phrase IoT merges the togetherness of human lifestyle - 'things'- with the togetherness of the global computer network - 'the internet'[28].

The distinctive and proper definition for the Internet of Things is not obtainable yet because of its boundless potential and revolutionary aspects that differ within scientists and developers. By far the finest definition of it would be, 'A wide-ranging, open network that possesses the power of sharing data and resources, self-assembling, acting, and reacting according to any small change in environment or situation.'

IoT architecture consists of the perception layer, transport layer, processing layer, application layer, and business layer. The working principle of IoT involves the connection of objects and gadgets with fitted sensors to an Internet of Things platform, which combines information from various devices and inspects the necessary data, and shares them with applications to address particular issues.

1.5.1 IoT Architecture Layer

With its seamless connection of devices under one roof, the Internet of Things has transformed modern technology. Because most of these technologies are wireless, the structure becomes extremely complicated and difficult to manage. As a result, architecture is necessary. The architecture of the Internet of Things is a framework that outlines the physical components, the network's functional organization and configuration, operational procedures, and data formats to be used.

1. Perception Layer- This layer serves as a bridge between the digital and real world, forming the components of the physical design of IoT. The major function of this perception layer in the IoT architectural tiers is to convert analog signals into digital form and vice versa. It perceives physical attributes like temperature and location from objects and converts that information to digital signals for easy network transmission using sensors, actuators such as sensors, RFID, and 2-D barcode based on the applications. Sometimes the object itself cannot perceive the information directly, rather it needs some embedded microchips which can sense and process the data. This requires nanotechnology which is a key technology in this layer.

2. Transport Layer- The transport layer, commonly referred to as the network layer, connects the perception and middleware layers. It uses networking technologies such as 3G, 4G, WiFi, LAN, RFID, NFC, and others to carry and transfer data between the two layers, following multiple protocols. It ensures secure data transfer and keeps the obtained data confidential.

3. Processing Layer- This layer is also known as the middleware layer. It receives, stores, analyzes, computes, and processes massive amounts of data from the transport layer. Then as per the need of the lower layers, it can provide a diverse set of services. It employs many technologies such as databases, intelligent processing, cloud computing, ubiquitous computing, and big data processing modules.

4. Application Layer- The information gathered from the processing layer helps the application layer to manage all the application processes. It helps in developing a variety of Internet of Things applications, such as intelligent transportation, logistics management, identity verification, location-based service (LBS), safety, etc. It acts as the interface between the end IoT devices and the network IoT applications providing different services based on the preference of each application.

5. Business Layer- It gathers data and does decision-making analysis on it, which includes creating flowcharts, graphs, and analyzing findings to see how the device might be improved. With the help of data analysis business owners and stakeholders can carry out business planning and strategy to enhance their productivity which in turn can help them in achieving their future goals and objectives.

For our device to function, it needs to go through each of these layers during its whole operational mechanism. The data collected by the sensors are analog which needs to be converted into digital form as microprocessors deal with digital information. This helps the microprocessors to store, analyze, understand, process and display the results. After conversion, statistical analysis can be done to extract, isolate and manipulate information as per our requirement. Moreover, in digital format, the data is compressed, minimizing hardware storage capacity and allowing faster transmission. It can also be encrypted for security. This whole thing falls under the perception layer. Next, we have the transport layer which helps in establishing the Wi-Fi connection so that we can send the data received from the sensors to the cloud server. In our case, ThingSpeak is basically where the task of the

processing layer is taking place. The application layer is the layer which is enabling us to send messages to the user. The business layer will be essential when we will carry out analysis of predicted data through Machine Learning algorithms and when launching the device in the market.

1.5.2 Network Topologies

Network topology is defined as the arrangement of all the components and their interaction within a communication network. There are mainly two types of network topologies: 'Physical topology' which describes the physical layout of how the devices are connected with the help of cables, nodes, links, etc. and 'Logical topology' which describes the internal communication in a network, that is, it maintains a particular network protocol and allows the exchange of data flow among devices throughout the entire network.

Types of Physical Network Topologies are as follows:

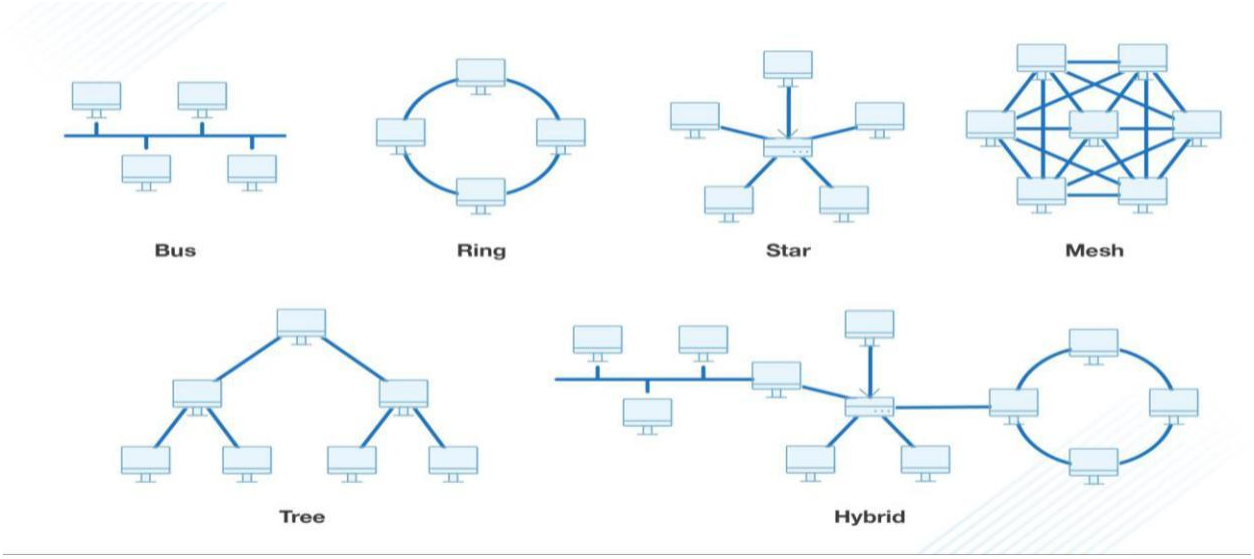

Figure 1: Different Types of Network Topologies

1. Bus topology- With the use of taps and drop lines, all of the devices connect to a shared central connection known as a bus in this topology. The bus serves as the network's backbone, allowing devices to communicate independently. When a device delivers data frames to a second device, the frame is received by all other devices connected to the central line. Only the target device accepts it; others reject it by inspecting the received frame's destination MAC address. Data transmission is one-way and the responsibility of this data flow from one point to the next is shared among the devices in the network. It is appropriate for smaller systems like LANs.

2. Ring topology- It is a topology where the nodes of each device are connected to two adjacent nodes on either side of it in a closed-loop to form a ring structure. To reach their final destination, data frames broadcasted by one device travels in one direction from that device to another in a circular fashion. This is done with the help of a repeater which keeps on forwarding the data until it is received by the designated device.

3. Star topology- In star topology, point-to-point communication links connect all of the network's devices to a central device, such as a switch or hub. When a device wants to transfer data to another device, it must first go through the central device. Depending on the type of central device utilized, the received data frame is subsequently broadcast or unicast towards the target device. As the data does not pass through every node, it provides better security.

4. Mesh topology- Each device in a mesh network has a point-to-point connectivity to every other device in the network, making it a unique network design. This means there is a dedicated link or cable between the two connected devices that allows the transfer of data providing direct communication within those nodes. Mesh can be of two types: full mesh and partial mesh. In a full mesh, each node is directly connected to the others, however in a partial mesh, certain devices are connected to a large number of devices, while others are only connected to one or two. If we use simplex links, the number of communication links for 'n' devices will be 'n(n-1)', however if we use duplex links, the number of communication links will be 'n(n-1)/2'.

5. Hybrid topology- It is the most widely used network that can be made by interconnecting two or more topologies providing flexibility to modify according to our needs. This gives it reliability and at the same time increases the cost of installation.

We will need hybrid topology for our design to function as we have both Star and Bus topology working in our system.

Advantages of Hybrid Topology:

1. It incorporates all the benefits of the different topologies which are used to fabricate it.
2. It is extremely flexible and reliable.
3. Easy error detection and troubleshooting of the problematic link.
4. Maintenance of large traffic volume.
5. A large network can be created.
6. New hardware components can be integrated easily because of their architecture which makes them scalable.

Disadvantages of Hybrid Topology:

1. Expensive network
2. Complex system design

3. Requires a lot of cables to install as they have larger architecture.
4. Difficult installation process.
5. Central hubs are costly as they need to connect and function with different networks.

Since we have some sensors operating on the I2C serial bus protocol, it falls under the bus topology where the bus is controlled in a master/slave connection. The rest of the system will follow the star topology depending on its functionality. Hence, we need the hybrid network topology.

1.6 Research methodology

Our first task was to determine the health complications a pregnant woman can face throughout the nine months of pregnancy. Once we did that, we looked for sensors to detect those problems and finally started working on the system.

There will be two devices. In one we have integrated sensors: a Temperature and Humidity Sensor (DHT22), AD8232 (ECG sensor), and a GSR sensor connected to Arduino Mega. These sensors are used to measure temperature, humidity, fetal heart rate, and autonomic nervous activity respectively. Another device is the Arduino Uno connected to two sensors which are MPU6050 for fall detection and MAX30100 for heart rate and SPO2 measurement, along with the integration of a vaccination reminder system with it.

The sensors of device 1, in turn, are connected to the ESP8266 module. The data sensed and collected are transmitted through Wi-Fi to the Internet and then to the cloud server, which can then be accessed by the users through smartphones or laptops for further actions, and decisions can be made based on these by the doctors. The sensors of device 2 are processed only through Arduino Uno and displayed onto the LCD monitor. If any abnormal reading is detected by any sensor in any device, an SMS is sent to the user immediately indicating that the user is at risk.

ESP8266 is a micro-controller with a self-contained Wi-Fi network that bridges existing micro-controllers and Wi-Fi. The leading software here will be ThingSpeak, an open-source Internet of Things (IoT) application and API. It is used to store and retrieve data from things using the HTTP protocol over the Internet.

The diagram below shows the system level architecture of the whole project.

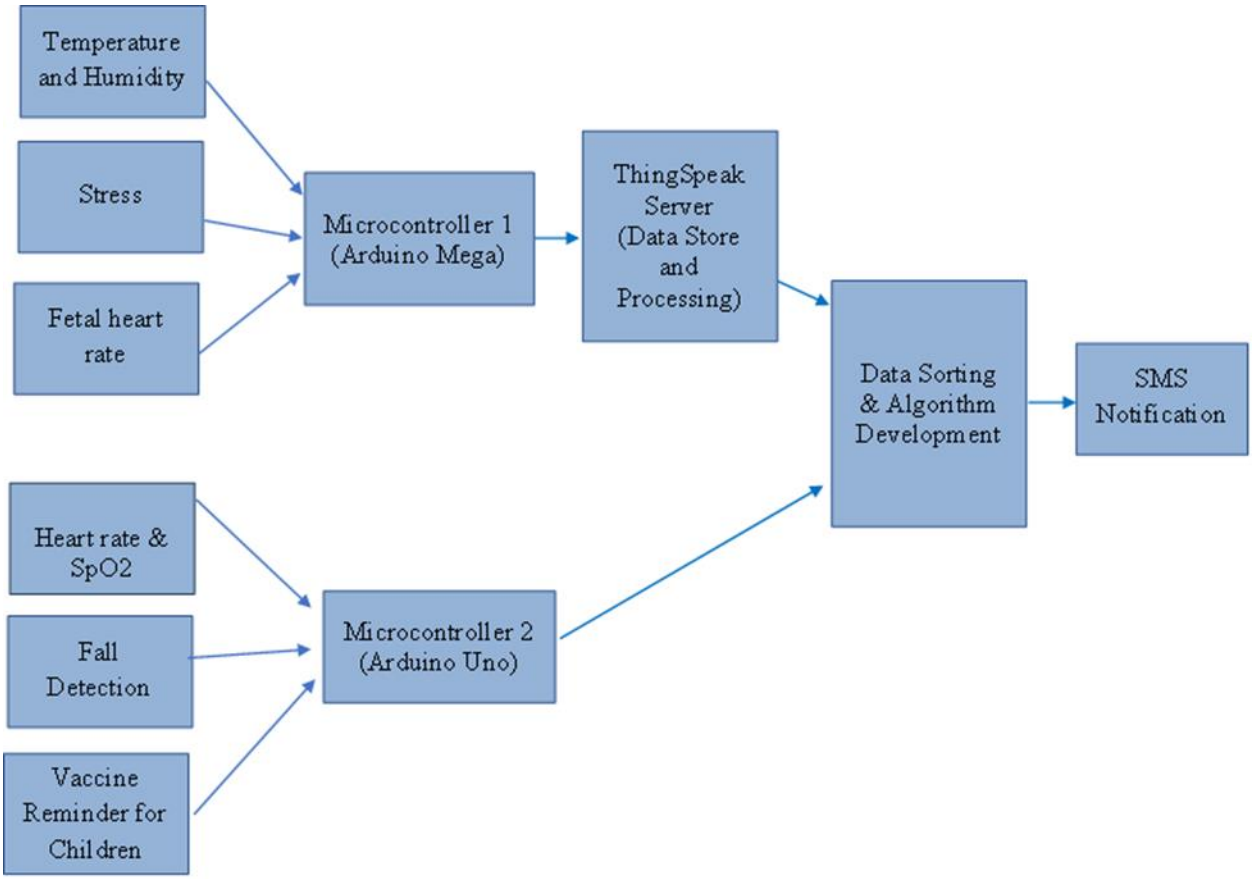

Figure 2: System-Level Architecture

10

1.7 Thesis Structure

As the title of our paper implies, we will be designing a health monitoring system that can be used to track, observe and keep in check all the parameters of a pregnant woman and her fetus that are necessary for a safe and complication-free pregnancy.

In the first chapter, we covered the revolutionary era of the Internet of Things, as well as what inspired us to work on the mentioned topic. The rest of the layout comprises background, risks associated with the health parameters of a pregnant woman, architectural layers of IoT, network topologies including their function in our device, research methodology, and some specific relevant studies that have been done in the past on this topic.

Chapter 2 is mainly designed to focus on the hardware connection part of our system. It is divided into subchapters that show the device's block diagram, schematic, and prototype. It also contains the components required to construct the hardware, the reason for their selection, and its limitations.

Chapter 3 focuses on the software segment where we have tried to emphasize on ThingSpeak, which will function as a cloud server platform, as well as its setup procedures and the limitations it has. We also discussed the code that our system will operate on, together with the design algorithm and its implementation.

In Chapter 4, the data collection and analysis method has been examined, followed by an explanation of the findings and a discussion.

Finally, in Chapter 5, we wrapped up our thesis by examining some of our created system's potential drawbacks, limitations, and probable future work prospects.

1.8 Conclusion

To sum up this chapter, we have discussed our motivation and objective behind doing the Pregnant Women & Child Health Monitoring study. We have talked about the risks associated with different health parameters that create a general idea regarding health concerns. As our study is linked with IoT, we have discussed different IoT architecture layers and IoT topologies, and also mentioned the layers and topology we are using. Information about the project planning and structure are included in this chapter. The next chapter talks about the sensors and the hardware implementation so that the construction of the physical device can be better understood.

Chapter 2

IoT Based Health Monitoring Device

2.1 Introduction

In this chapter, we are going to discuss the hardware components and why they are used in our system, the block diagram of our devices and the schematic diagram of the devices with all the connections. Our portable device must monitor several health parameters of a pregnant woman and her child that are vital to watch throughout the pregnancy period. It should be hassle-free, and the data collection must pass to the cloud server. Our device must notify users when a parameter crosses the threshold value, indicating risk for pregnant women. The hardware setup of these are discussed in this chapter and in the end of the chapter we will be talking about the hardware limitations.

2.2 Component Selection

The device consists of the following components:

1. Arduino Mega 2560 R3: Our first microcontroller board collects sensor data from DHT22, GSR, & AD8232 and then sends it to ESP8266 by serial communication for further processing and analysis.

2. Arduino UNO R3: The second microcontroller board collects sensor data from MPU6050 and MAX30100 and displays the readings on an LCD screen and further analyzes them.

3. DHT22: This is a temperature and humidity sensor used to take temperature and humidity measurements from mother and child.

4. MAX30100: It is used to measure pulse oximetry and heart rate.

5. MPU6050: This three-axis accelerometer and gyroscope is used for fall detection.

6. GSR sensor: It is used to detect the stress of pregnant women.

7. AD8232 sensor: The AD8232 is used to measure the electrical activity of the heart i.e. the heart rate of the fetus.

8. SIM900A: This is used for sending text messages in the user's phone number notifying users of the sensor values beyond a certain level. It is also effective for vaccine reminders of the child.

9. Wi-Fi Module ESP8266: This module will be used to give access to the Wi-Fi network which is required for sending the sensor data to ThingSpeak.

2.3 Hardware Description:

1. Arduino Mega 2560 R3- The Arduino Mega 2560 R3 is a microcontroller board based on an ATmega2560 AVR microprocessor that evolved from the Arduino Mega. A 16 MHz resonator, a power jack, a USB connection, a reset button, and an ICSP (in-circuit system programming) header are all included, as well as 70 digital input/output pins, among which 15 can be used as PWM outputs and 16 of which can be used as analog inputs, a 16 MHz resonator, a power jack, a USB connection, a reset button, and an ICSP header. Not only is the USB connection with the PC required to program the board, but it is also required to power it up. The Mega2560 gets its power from either a USB port or an external power source.

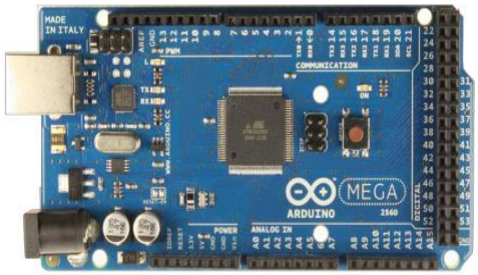

Figure 3: Arduino Mega 2560 R3 [29]

2. Arduino Uno R3- It is a microcontroller (ATmega328P). It has a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP (In-Circuit Serial Programming) header, and a reset button, as well as 14 digital input/output pins, six of which are PWM outputs.

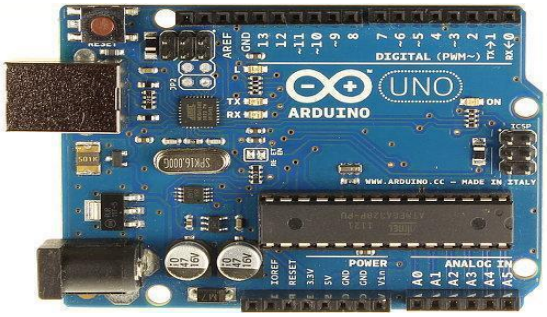

Figure 4: Arduino UNO R3 [30]

3. DHT22- It is a basic, low-cost temperature and humidity sensor. It monitors the ambient air with a capacitive humidity sensor and a thermistor and sends a digital signal to the data pin. The 3-5V power pin is attached to the first pin on the left, the data input pin is connected to the second pin, and the rightmost pin is connected to ground. It measures temperature with much greater accuracy and over a greater range of temperature/humidity compared to other temperature/humidity sensors. It has a temperature measurement range of -40°C to +125°C with an accuracy of +-0.5 degrees. The humidity measuring range of the DHT22 sensor is better, ranging from 0 to 100 percent with a 2-5 percent accuracy. It features a 0.5Hz sample rate, which means it can take a measurement every 2 seconds.

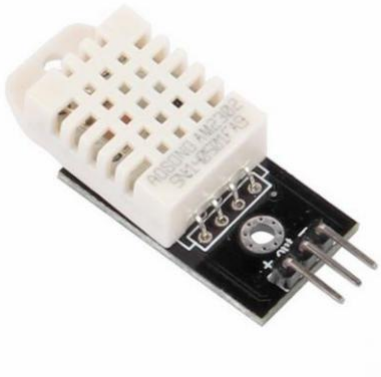

Figure 5: DHT22 [31]

4. MAX30100- It's a system that combines pulse oximetry and a heart rate monitor. To detect pulse oximetry and heart rate signals, two LEDs, a photodetector, improved optics, and low noise analog signal processing are used, which are all coupled to detect pulse oximetry and heart rate signals. It is an optical sensor that gives a reading based on the emission of two wavelengths of light from two LEDs- a red and an infrared light then measures the absorbance of pulsing blood through a photodetector. This particular light combination is optimized for reading the data through the tip of one's finger.
   Digital output data is kept in a 16-deep FIFO within the device, and it is fully customizable via software registers. It has a 12C digital interface to communicate with the host microcontroller. Ambient light cancellation (ALC), a 16-bit sigma-delta ADC, and a proprietary discrete-time filter make up the pulse oximetry subsystem of the MAX30100. It operates at ultra-low power, making it perfect for battery-powered devices. The MAX30100 requires a power supply that is between 1.8 and 3.3 volts. The MAX30100 uses 1.8V and 3.3V power supplies, and it may be turned down by software with very little standby current, allowing the power supply to be connected at all times.

Figure 6: MAX30100 [32]

5. MPU6050- It has a built-in accelerometer and gyroscope sensor for fall detection. The gyroscope is used to determine the orientation of the object and the accelerometer is used to determine the information about angular parameters. It is basically a sensor for motion processing devices. It is the world's first six-dimension motion tracking device. It was designed for low-cost and high-performance smartphones, tablets, and wearable sensors. It can run nine-axis algorithms and capture motion in the X, Y, and Z axes.

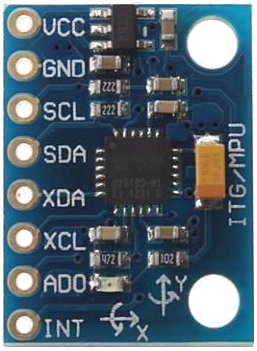

Figure 7: MPU6050 [33]

6. SIM900A- It is based on dual-band GSM/GPRS. It will work with frequencies 900 or 1800 MHz. SIM900A can automatically search for these two bands. AT Commands can also be used to change the frequency bands. The baud rate can be changed from 1200-115200 using the AT command. The inbuilt TCP/IP stack of the GSM/GPRS Modem allows us to connect to the internet via GPRS. SIM900A is a wireless module that is both small and dependable. This is a complete GSM/GPRS module in SMT format, created with a high-performance single-chip processor based on the AMR926EJ-S core, allowing us to take advantage of tiny dimensions and cost-effective solutions.

Figure 8: SIM900A [34]

7. GSR Sensor- GSR stands for galvanic skin response. Any strong emotional change causes a change in the sweat gland activity; the greater the sweat gland activity, the more the perspiration and less the skin resistance. Thus, the GSR sensor detects strong emotions by measuring the electrical activity of our skin.

Figure 9: GSR Sensor [35]

8. AD8232 Sensor-The AD8232 is a small chip that is used to measure heart electrical activity. An ECG, or electrocardiogram, can be used to track this electrical activity. Electrocardiography is a test that can be used to diagnose a variety of cardiac problems.

Figure 10: AD8232 [36]

9. Wi-Fi Module ESP8266- It is a self-contained SOC (system on chip) with an integrated TCP/IP protocol stack that can give microcontroller access to Wi-Fi networks. It is preprogrammed with AT command set firmware, i.e it can be connected with Arduino and use a Wi-Fi network to send data. It is highly durable as it can sustain a high range of temperatures which makes it suitable for use in industrial purposes as well.The chip delivers dependability, compactness, and robustness with highly integrated on-chip functionality and a low number of external discrete components.

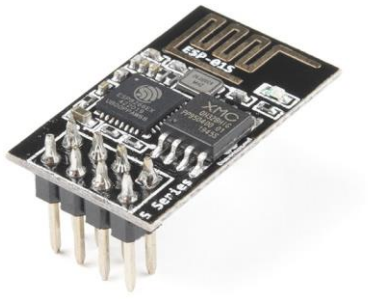

Figure 11: Wi-Fi Module ESP8266 [37]

2.4 Hardware Connection

Device 1: Three of our sensors (DHT22, GSR, and AD8232) are connected to Arduino Mega 2560 R3 as input. The microcontroller takes the data and sends it to the ESP8266 via serial connection. Afterwards, the ESP8266 sends that data to the ThingSpeak server.

Device 2: The other two sensors (MPU6050 and MAX30100) are connected to Arduino UNO R3 as input. The vaccine reminder system is included in this device.

For both of the devices, the sensor values are displayed on the LCD Screen. The data is analyzed, and a phone alert message is passed whenever the threshold crosses for any sensor range.

2.4.1 Block Diagram

The block diagrams for both devices are displayed below

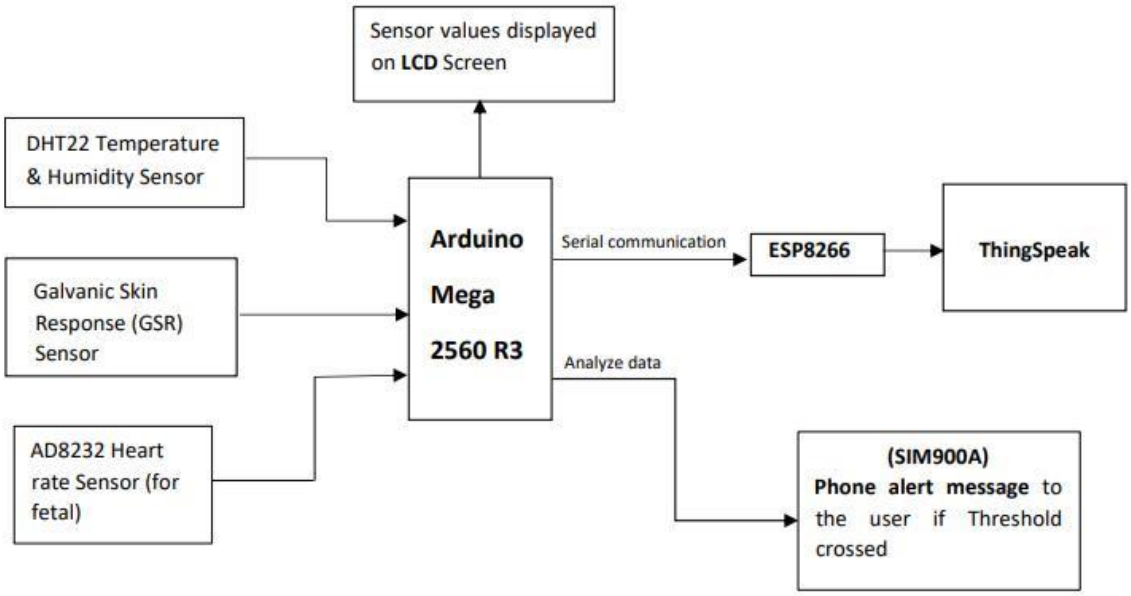

Figure 12: Block Diagram of Device 1

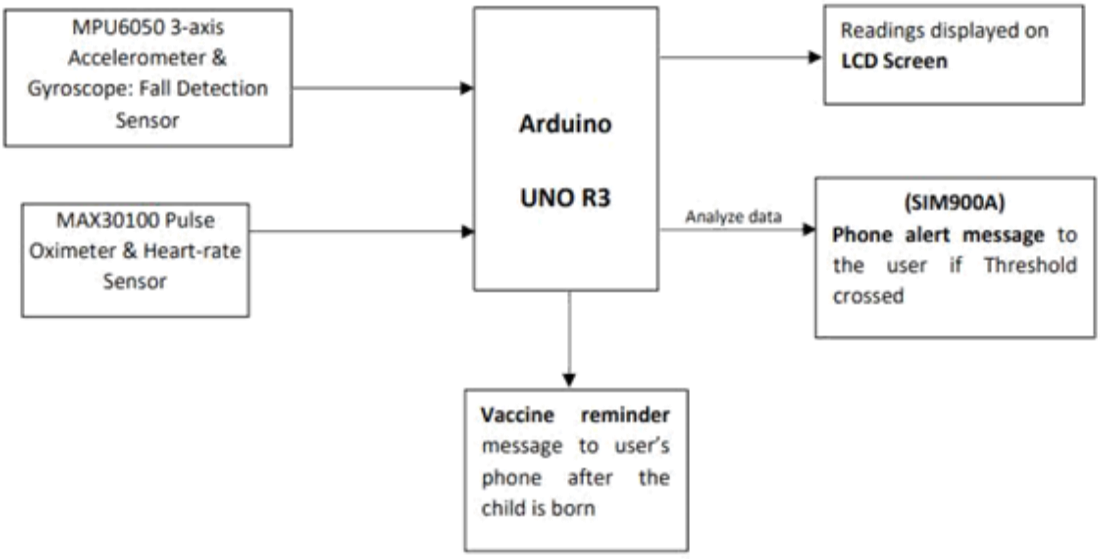

Figure 13: Block Diagram of Device 2

2.4.2 Schematic Diagram

The two schematics of our two devices were created in Proteus Software. These diagrams show all the essential connections and create a comprehensive concept regarding our whole system.

The schematic of ***Figure 14*** contains the DHT22 sensor, GSR sensor, and AD8232 sensor connected to Arduino Mega 2560 R3 with all other components. The 5V and GND pins of the DHT22 sensor, GSR sensor, SIM900A, and LCD are connected to the VCC and GND pins of the Arduino Board respectively. The 3.3V of AD8232 is connected to a DC signal of 3.3V. The Wi-Fi module ESP8266 has pins CH-PD and VCC which are both connected to the DC signal of 3.3V.

The major pin connections are provided beneath:

1. DHT22 sensor DATA pin - Arduino MEGA 2560 R3 pin 12
2. AD8232 sensor OUT pin - Arduino MEGA 2560 R3 A0 pin
3. AD8232 sensor LOD^- pin - Arduino MEGA 2560 R3 pin 11
4. AD8232 sensor LOD^+ pin - Arduino MEGA 2560 R3 pin 10
5. GSR sensor SIG pin - Arduino MEGA 2560 R3 A2 pin
6. SIM900A RX pin - Arduino MEGA 2560 R3 pin TX2
7. SIM900A TX pin - Arduino MEGA 2560 R3 pin RX2
8. ESP8266 RXD pin - Arduino MEGA 2560 R3 pin TX3
9. ESP8266 TXD pin - Arduino MEGA 2560 R3 pin RX3
10. LCD pin RS - Arduino MEGA 2560 R3 pin 36

11. LCD pin E - Arduino MEGA 2560 R3 pin 35
12. LCD pin D2 - Arduino MEGA 2560 R3 pin 34
13. LCD pin D3 - Arduino MEGA 2560 R3 pin 33
14. LCD pin D4 - Arduino MEGA 2560 R3 pin 32
15. LCD pin D5 - Arduino MEGA 2560 R3 pin 31

The schematic of ***Figure 15*** accommodates the MPU6050 sensor and MAX30100 sensor connected to Arduino UNO R3 along with SIM900A and an LCD. The 5V and GND pins of the MPU6050 sensor, MAX30100 sensor and SIM900A are connected to a DC signal of 5V and a ground terminal respectively.

The major pin connections are provided beneath:

1. MPU6050 sensor SDA pin - Arduino UNO R3 SDA pin
2. MPU6050 sensor SCL pin - Arduino UNO R3 SCL pin
3. MAX30100 sensor SDA pin - Arduino UNO R3 SDA pin
4. MAX30100 sensor SCL pin - Arduino UNO R3 SCL pin
5. SIM900A RX pin - Arduino UNO R3 pin 9
6. SIM900A TX pin - Arduino UNO R3 pin 8
7. LCD pin RS - Arduino UNO R3 pin 2
8. LCD pin E - Arduino UNO R3 pin 3
9. LCD pin D2 - Arduino UNO R3 pin 4
10. LCD pin D3 - Arduino UNO R3 pin 5
11. LCD pin D4 - Arduino UNO R3 pin 6
12. LCD pin D5 - Arduino UNO R3 pin 7

Both the Proteus schematics are provided below:

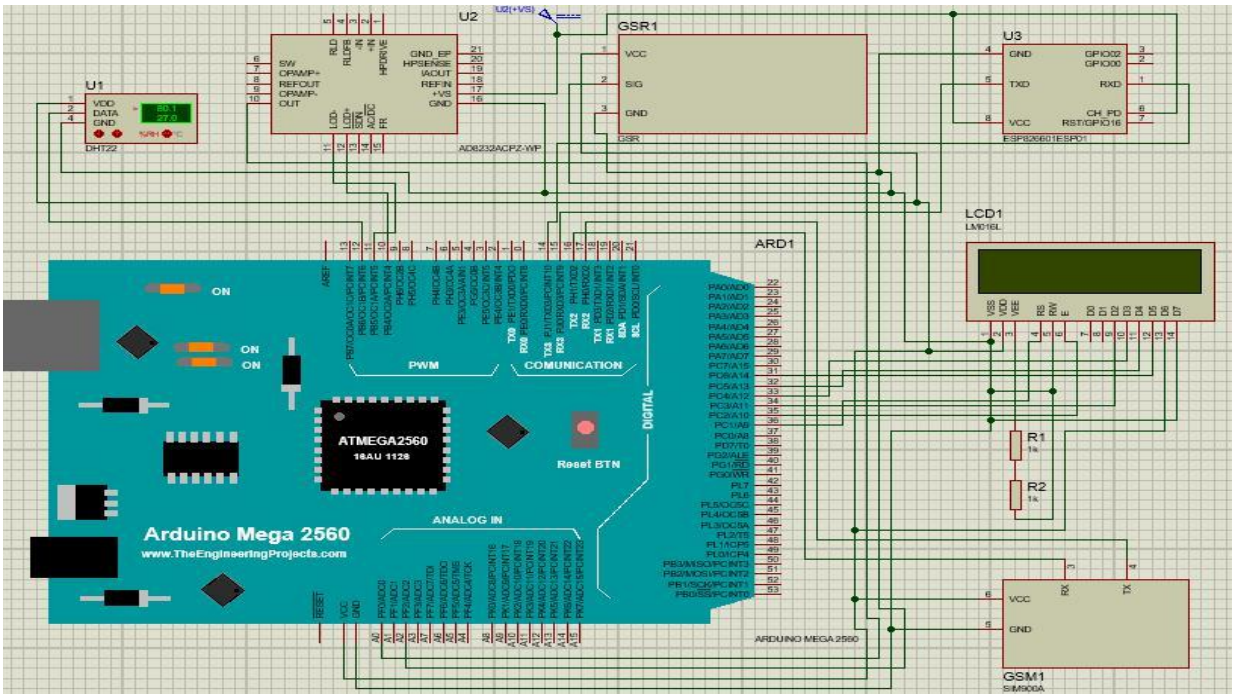

Figure 14: Schematic of Device 1

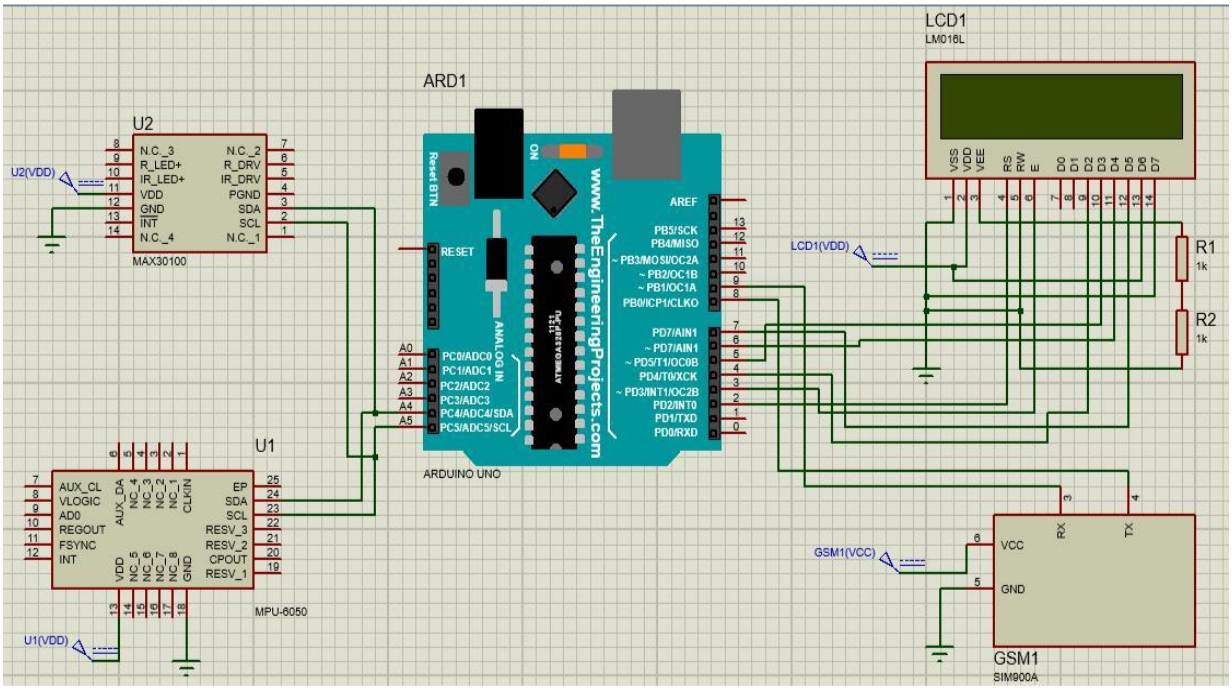

Figure 15: Schematic of Device 2

2.4.3 Prototype System Picture

These are the two prototypes of our health monitoring system, which include all of the sensors and connections necessary to collect data.

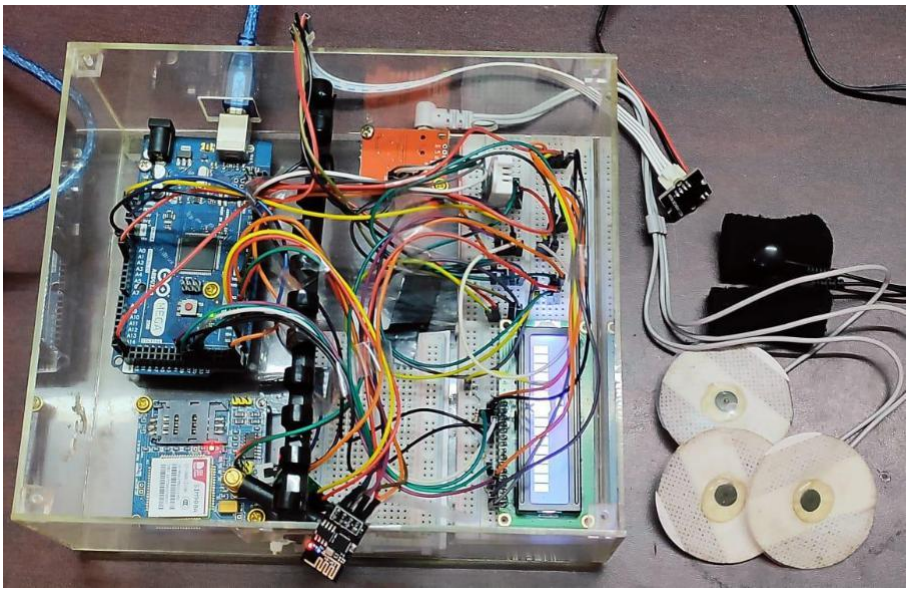

Figure 16: Device 1 prototype

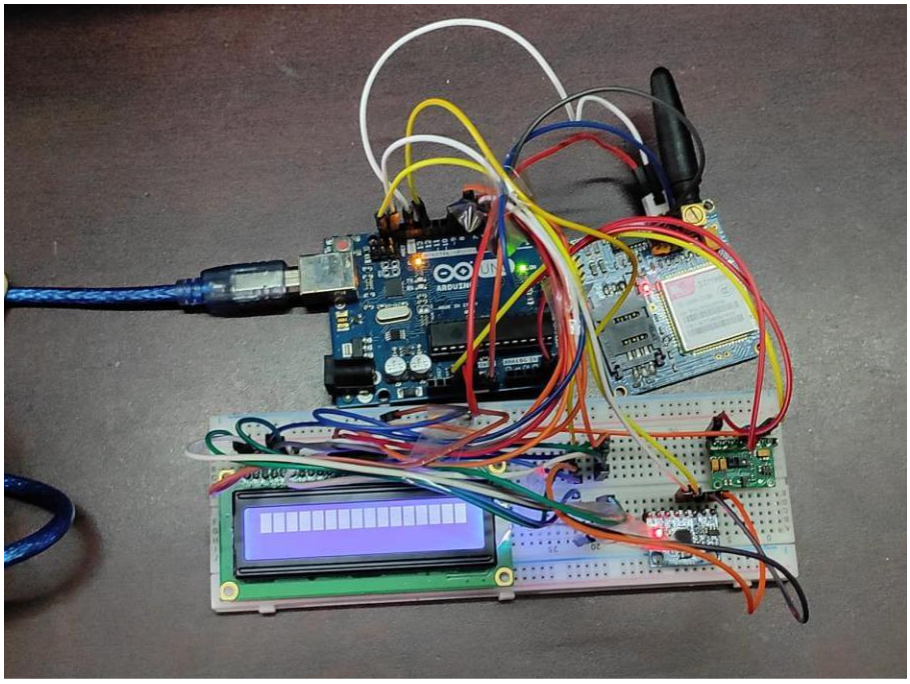

Figure 17: Device 2 prototype

2.5 Hardware Limitations

There were several issues that we had to face when we tried to combine all the parts of our devices together. This gave rise to a number of limitations:

- Some of the sensors like the MPU6050 & MAX30100 do not function when we connect them to Arduino mega along with the ESP8266 module but can operate when they are only connected with the Arduino Mega without ESP8266. This is because MAX30100 requires a non-blocking loop, while our code contains AT commands of ESP8266 which are blocking in nature. Due to this reason, we had to connect them separately with Arduino UNO, which gave rise to Device 2.
- Heart rate sensors malfunction easily and cannot perceive data accurately as they have sensitivity issues. We had to buy both MAX30100 and AD8232 a couple of times before we could manage to get data from them.
- The sensor data that we are sending to ThingSpeak has certain limitations in terms of the accuracy of the obtained data because the sensors we are using are not very sensitive compared to our budget.
- Sometimes, the sensors lose connection when the Arduino is powered up by USB. We had to try reconnecting the USB to make the sensors work properly.
- The heart rate and SPO2 readings from MAX30100 were going to be passed as a message to the user or the emergency contact number when they crossed a specified level, but this section could not be implemented due to a problem with the delay in that particular code.

2.6 Conclusion

In this chapter, we have described all the components that were required for the completion of our project. The first schematic diagram, named Device 1, shows all the connections of the sensors and the Wi-Fi module with Arduino Mega and the second schematic diagram, named Device 2, shows the separate connection that we had to make with Arduino Uno. We could only display the system's prototype, despite our desire to turn it into a wearable gadget, which we were unable to do due to time constraints. To make it cost-effective, we did the project with available sensors in the market which gave us limitations in data collection. In the following chapter, we will be discussing in detail the process of sending the collected data to the server.

Chapter 3

IoT & Comprehensive System Design

3.1 Introduction

The following chapter will give ideas about IoT and ThingSpeak setup, how data is processed in it, implementation of ThingSpeak i.e., working of ThingSpeak and ESP8232 based on a certain protocol. There is the full code of the devices upon which the system is running. Finally, it is ended by discussing the limitations regarding the ThingSpeak server.

3.2 IoT & ThingSpeak setup

The IoT and ThingSpeak setup utilized in this project is a thorough system architecture in which all of the sensors are integrated into one Arduino Mega 2560 (serial communication). As a result, the necessity to perform the same method for each sensor is reduced. We also need to communicate this sensor data to the ThingSpeak server, which is why all of the sensors were placed on one Arduino from which it was transferred to the server.

The data is displayed on the serial monitor before being sent to the server, and this data is saved as a CSV file, which allows us to view the real-time data of the specified sensors in a tabular format. Also, we need to send this sensor data together in the ThingSpeak server, so that is why it was needed to put the sensors all under one Arduino from where it is sent to the server. This data is sent to the ThingSpeak server via Wi-Fi module ESP8266 where this data is being plotted to make a graph in every real-time and from there, we can observe the trend or the change in the data. The graphs shown are for different sensors and for each graph we needed to create a different field under a certain channel. There is an API key for the specified channel we have created for the sensor fields. This API code of the channel is copied and pasted in the code where required along with the Wi-Fi username and password. The sensors connected with the Arduino Mega are DHT22, GSR, and AD8232. The sensor data which we are sending to ThingSpeak are temperature and humidity (from DHT22) and stress detection (GSR sensor). The AD8232 sensor value (fetal heart rate) is not sent to ThingSpeak rather it is shown in the LCD display. The data obtained in ThingSpeak is accessible to anyone who is concerned, e.g., doctors in nearby hospitals. However, we had to separate two sensors (using I2C communication) from the other sensors in an Arduino UNO. These sensors are MAX30100 and MPU6050 which work with the help of I2C communication but were not working with the other sensors so we separated them and connected them with Arduino UNO. Then observed their data in the serial monitor and showed their data on an LCD screen.

3.3 Implementation of ThingSpeak & ESP8266

The two major components of our device are the ESP8266 Wi-Fi module and the IoT Cloud platform ThingSpeak. The Arduino Mega 2560 collects data from all the sensors attached to it and sends it to ESP8266. The Wi-Fi module needs to have access to our Wi-Fi network so that it can upload the data to ThingSpeak. ThingSpeak works on both HTTP and MQTT protocol for sending the IoT device data to its cloud. We can see live data streams of all the sensors in ThingSpeak. For visualization, automatic graphs are plotted which gets accurate depending on the values that it is receiving. The diagram below shows the ThingSpeak interface showing the GSR, temperature and humidity curves.

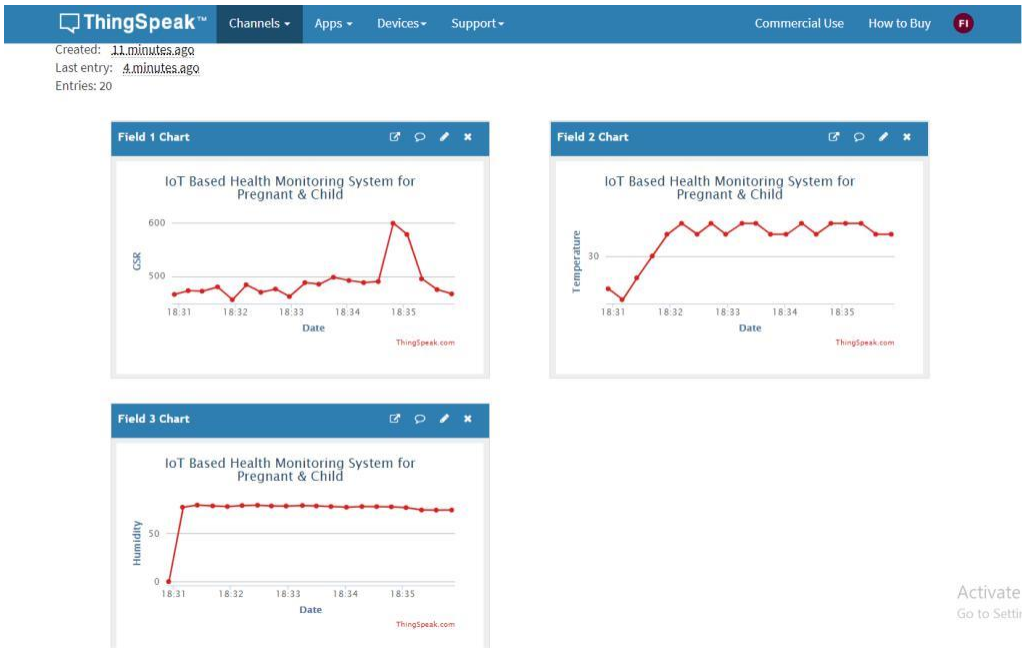

Figure 18: ThingSpeak Interface

3.4 Operational Mechanism

In this section, we will be discussing the system setup i.e., how we are collecting the sensor data and processing them to show in ThingSpeak server or LCD display. Our system's algorithm has also been described in detail. We must also make certain that the data we are working with is only accessible to those who need to know, and that no one else has access. This is for the sake of system security and privacy. To reduce the likelihood of unauthorized access, we can restrict IP

address access. The Wi-Fi password and username for the ThingSpeak server have been set up specifically.

3.4.1 System Setup

The sensors on the device 1 prototype are utilized to collect data from the user. It was planned to be a wearable device, but we have built a prototype to demonstrate that our sensors operate properly. Firstly, to take temperature and humidity data by DHT22, the user will hold the sensor. The sensor allows serial communication with the Arduino mega and the sensor values are then sent to the ThingSpeak server via Wi-Fi module ESP8266. Their graphs are generated in their respective fields in ThingSpeak and simultaneously the sensor values are displayed onto the LCD monitor. Secondly, for stress detection, we are using a GSR sensor which has two-finger straps attached to the chip with wires. The straps are worn in two fingers of the right hand by the user and this sensor data is similarly sent to ThingSpeak and LCD monitor. Thirdly, the AD8232 chip which measures the fetal heart rate has 3 foam electrodes that are placed on the mother's abdomen while taking readings. The heart rate is sometimes not detected properly in which case it shows an exclamation mark but other times it shows readings. The fetal heart rate reading is displayed on the LCD display. These sensors discussed so far were connected with Arduino Mega. All these sensor codes along with LCD and ESP8266 codes are put together as one. With this Arduino Mega, the Sim900A GSM module is connected which is used to send messages to the emergency contact number if any of these sensor values cross their set threshold value or condition.

In device 2, the sensors connected to Arduino Uno are MAX30100 which is used to detect heart rate and blood oxygen saturation of the user and MPU6050 for fall detection. For that, the user has to hold the chip with her index finger and observe the sensor readings from the LCD screen. The data measurements from MPU6050 are taken for three different cases. Firstly, when the user is in stable state (sitting or standing). Secondly, if the user falls down and cannot get up then it will show fall detected and this message will be displayed onto the LCD monitor. Thirdly, when the user falls down and gets up. All other times the accelerometer and gyroscope values will be displayed in the LCD display. Here also we are sending data to the emergency contact number by SIM900A if the sensor detects a fall in the case of MPU6050. We have also incorporated a system for vaccine reminders which has been included in the system algorithm. The vaccine reminder is designed in such a way that based on the age of the child; a message will be sent to the parent's phone number. Some of the vaccine schedules [38][39] that will be shown are as follows:

Vaccine Name	Time of dose
BCG, Oral Polio(1st dose)	After Birth
Pentavalent(1st dose),Oral Polio(2nd dose)	6 weeks
Pentavalent(2nd dose),Oral Polio(3rd dose)	10 weeks
Pentavalent(3rd dose),Oral Polio(4rth dose)	14 weeks
MR(Measles and Rubella)	9 months
Varilrix	1 year
Oral Polio(5th dose), Measles	15 months
Typhoid	2 years

Table 1: Immunization schedule for a child up to 2 years

3.4.2 System Algorithm for Device 1

The Arduino Mega reads data from the sensors DHT22, GSR, and AD8232. Out of these 3 sensors we only needed to include the library for the DHT22 sensor and created a DHT object so that we can access special functions related to the library. The sensor is connected to Digital Pin 12 of the Arduino meaning the sensor sends a digital signal to the Arduino, and the Arduino reads the signal by calling the function ***DHT.read22()*** from the library. We do not need a library for the GSR sensor since the MCU just simply reads the analog signal from the analog pin A2 by calling the function ***analogRead()*** and gives us analog values for GSR. Similarly, we do not have to include any library for AD8232 since the MCU reads the analog signal sent by the sensor from analog pin A0. In the output, we get analog values that indicate the electrical signals generated from the heart (ECG). This electrical activity can be charted like an ECG (Electrocardiogram). The figure below shows the ECG graph of a normal person generated by a serial plotter from the AD8232 readings.

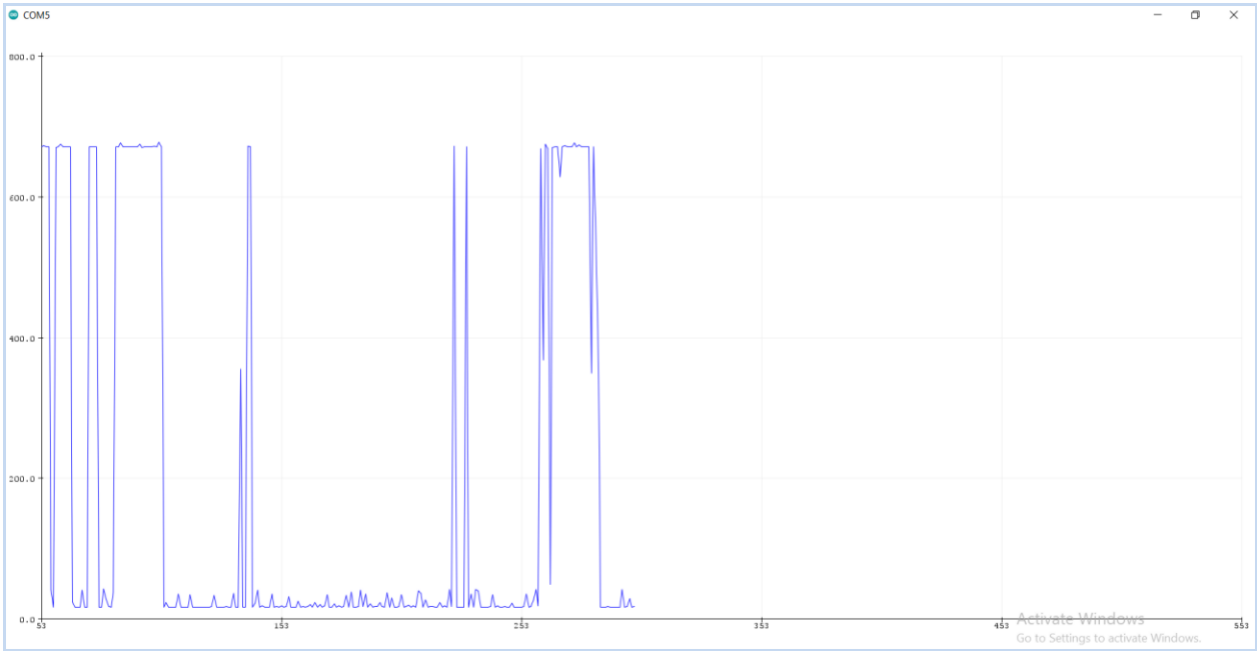

Figure 19: Graph of ECG on Arduino serial plotter

But if the sensor is unable to detect any electrical signal, the LO+ and LO- pins of the sensor send a high signal to the digital pins 10 and 11 of the Arduino. The MCU reads this signal by calling the ***digitalRead()*** function. In this case, we will not get any value in the output and hence we have denoted this null value with an exclamation mark in the code. These sensors' data are printed into the serial and LCD monitor at the same time.

We have created one void function for the SIM900A setup [***sendMessageSetup()***] and 3 other void functions [***SendMessage1(), SendMessage2(), SendMessage3()***], for including the message sending functionality. Since we want to send 3 different alert messages, we had to include 3 separate void functions. We had set some particular threshold values or conditions for each of the sensors and if those thresholds are crossed, one or more of the sendMessage functions are called depending upon the situation, and an alert message is sent to the emergency contact number.

To connect to the local WiFi we use the ESP8266 module. After connecting to the WiFi, it then connects to the cloud platform, Thingspeak, over TCP/IP protocol. The Arduino implements this protocol by passing AT commands serially to the ESP8266 module. We have listed all the AT commands in the ***sendCommand(String command, int maxTime, char readReplay[])*** function**.** This function is being called into the ***void setup()*** to initialize the ESP8266 module. In the ***sendtoServer()*** function, we merge all the sensor data into a single string and call ***sendCommand()***.

Finally, we call the functions ***printData(), notify() & sendtoServer()*** from the void ***loop()*** function to print data to the LCD and serial monitor, send messages, and to send the string to the ThingSpeak server respectively.

Code:

Including the libraries for the components. ***LiquidCrystal.h*** is for the LCD display and ***dht.h*** is for temperature and humidity sensor.

```
#include <dht.h> //library for DHT

#include <LiquidCrystal.h> //library for LCD
```

Defining variables, parameters and setting threshold value of the sensors

```
dht DHT; //creates a DHT object

#define DHT22_PIN 12

LiquidCrystal lcd(36,35,34,33,32,31) // defining pins of LCD

const int GSR=A2; // defining the analog pin of GSR

int GSRsensorValue=0;

int gsr_average=0;

float high_temp_threshold = 37; // Setting the higher threshold for temperature sensor

float low_temp_threshold = 31; // Setting the lower threshold for temperature sensor

float heart_rate=0;

float gsr_threshold = 540;   // Setting the threshold for GSR sensor
```

```
boolean noheartrate= false; //stores no heart rate detection condition

byte noheartratecount=0; //store counts since no heart rate detected
```

Setting Wi-Fi access points and password

```
String AP = "Azharul98"; // AP NAME String

PASS = "project123"; // AP PASSWORD

String API = "22BMG2L62GI0WW8Y"; // Write API KEY from ThingSpeak channel

String HOST = "api.thingspeak.com"; // connecting to ThingSpeak server String

PORT = "80";

int countTrueCommand;

int countTimeCommand;

boolean found = false;
```

Initializing the LCD display, ESP8266, AD8232 and SIM900A module

```
void setup() {

  Serial.begin(9600); // Setting Baud rate

  lcd.begin(16, 2);

  lcd.print("Initializing");

  lcd.clear();

  delay(100);
```

```
  Serial.println("Date & Time, Temperature, Humidity, GSR, AD8232");
  Serial3.begin(115200);
  sendCommand("AT",5,"OK");
  sendCommand("AT+CWMODE=1",5,"OK");
  sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS +"\"",20,"OK");
  AD8232Setup();
  sendMessageSetup();
}
```

For the serial connection of ESP8266, we have used the hardware serial pins TX3 and RX3 of the Arduino Mega R3 2560.

In the void loop we call the three functions that we want to perform repeatedly

```
void loop() {
    printData();   // for printing sensor value
    notify(); // for sending alert messages to emergency contact number
    sendtoServer(); // for sending sensor data to ThingSpeak
}
```

Created two different void functions, to set up the AD8232 sensor and the SIM900A module, which are then called into the "void setup ()" for initialization.

```
void AD8232Setup(){
  pinMode(10, INPUT); // Setup for leads off detection LO +
  pinMode(11, INPUT); // Setup for leads off detection LO -

}

void sendMessageSetup(){
  Serial2.begin(9600);
  Serial.println ("SIM900A Ready");
  Serial.println ("Type ‘s’ to send message or ‘r’ to receive message");
}
```

Code for Temperature and Humidity Sensor (DHT22)

```
String getTemperatureValue(){

  int chk = DHT.read22(DHT11_PIN);

  float t= DHT.temperature;

  delay(50);

  return String (t);

}

String getHumidityValue(){

  float h= DHT.humidity;

  delay(50);

  return String (h)   // returning as string

}
```

Code for GSR Sensor

```
String getGSRValue(){
  long sum=0;
  for(int i=0;i<10;i++)       // Taking average of the 10 measurements to remove any glitch
     {
     GSRsensorValue=analogRead(GSR);
     sum += GSRsensorValue;
     delay(5);
     }
   gsr_average = sum/10;
   return String(gsr_average);
}
```

Code for printing data in the serial monitor and the LCD display and sending a message to the user if no fetal heart rate is detected by AD8232.

```
void printData(){

  Serial.print(",");
  Serial.print(DHT.temperature);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temp:");
  lcd.print(DHT.temperature);
  Serial.print(",");
  Serial.print(DHT.humidity);
  Serial.print(",");
  Serial.print(gsr_average);
  lcd.setCursor(9, 2);
  lcd.print("GSR:");
  lcd.print(gsr_average);
  Serial.print(",");
  if((digitalRead(10) == 1)||(digitalRead(11) == 1)){
    Serial.print('!');
    lcd.setCursor(0, 2);
    lcd.print("FH:!");

    noheartratecount++;
    Serial.print(", noheartratecount:");
    Serial.println(noheartratecount);
    if (noheartratecount>=5 && noheartrate== false ){ // when no heart rate detected
      noheartrate= true;

      Serial.println("No Heartbeat detected");
      SendMessage3();
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("No Heartbeat");
      lcd.setCursor(0, 2);
      lcd.print("Detected");
      delay(1000);
      noheartratecount=0;
      noheartrate= false;

    }
  }
  else{
    noheartratecount=0;

    float sum=0;
    for (int i = 0; i < 10; i++){
      float heartrate= analogRead(A0);
      sum=sum+heartrate ;
    }
  float heart_rate= (sum/10);
  Serial.println(heart_rate);

  lcd.setCursor(0, 2);
  lcd.print("FH:");
  lcd.print(heart_rate);
 }

}
```

All the sensors' data are printed into the serial monitor as well as the LCD monitor. At the beginning of the code, we had set a boolean variable named "***noheartrate"*** and had set its initial condition "FALSE" and had declared ***byte noheartratecount*** and set its initial value to "0". When the digital pins 10 and 11 of the MCU read "1", a "!" will be printed in the Serial monitor and LCD and the value of ***noheartratecount*** will be incremented by "1". Next the compiler will check the value and condition of ***noheartratecount*** and ***noheartrate***. If the value of ***noheartratecount*** is greater than or equal to 5 and ***noheartrate*****=** FALSE, then "No Heartbeat detected" will be printed and a message will be sent to the emergency number after which the ***noheartratecount*** and ***noheartrate*** will be made "0" and "FALSE" respectively. Otherwise, the MCU will read the analog pin A0 and show us the readings.

Comparing with the threshold values that had been set earlier and sending messages if the thresholds are crossed.

```
void notify(){
 If ((((DHT.temperature) > high_temp_threshold))|| (((DHT.temperature) < low_temp_threshold)))
    { SendMessage1();
  }
  else if (gsr_average>gsr_threshold){
   SendMessage2();
  }
}
```

Code for sending alert messages to the user's phone number. Here we have created 3 different voids to send 3 different types of messages.

```
void SendMessage1(){
```

```
   Serial.println ("Sending Message");

   //Serial2.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode

   delay(110);

   Serial2.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to send
message

   delay(110);

   Serial2.println("Patient has a fever!"); // Message content

   Serial2.println((char)26);// ASCII code of CTRL+Z

   Serial.println ("Message has been sent");

}

void SendMessage2(){

   Serial.println ("Sending Message");

   //Serial2.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode

   delay(110);

   Serial2.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to send
message

   delay(110);

   Serial2.println("High Stress Level Detected"); // Message content

   Serial2.println((char)26); // ASCII code of CTRL+Z Serial.println

   ("Message has been sent");

}

void SendMessage3(){

   Serial.println ("Sending Message");
```

```
    //Serial2.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode

    delay(110);

    Serial2.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to send
message

    delay(110);

    Serial2.println("No Heartbeat detected"); // Message content

    Serial2.println((char)26); // ASCII code of CTRL+Z

    Serial.println ("Message has been sent");

}
```

Sending sensors' data to ThingSpeak server

```
void sendtoServer( ){

  String getData = "GET /update?api_key="+ API +"&field1="+ getGSRValue()+ "&field2="+
getTemperatureValue()+ "&field3="+ getHumidityValue();

 sendCommand("AT+CIPMUX=1",5,"OK");

 sendCommand("AT+CIPSTART=0,\"TCP\",\""+ HOST +"\","+ PORT,15,"OK");

 sendCommand("AT+CIPSEND=0," +String(getData.length()+4),4,">");

 Serial3.println(getData);

 countTrueCommand++;

 sendCommand("AT+CIPCLOSE=0",5,"OK");

}
```

Building up the connection between Esp8266 and the Network

```
void sendCommand(String command, int maxTime, char readReplay[]) {
 while(countTimeCommand < (maxTime*1))
 {
  Serial3.println( command);//at+cipsend
   if(Serial3.find(readReplay))//ok
   {
     found = true;
     break;
   }
   countTimeCommand++;
 }
 if(found == true)
 {
   countTrueCommand++;
   countTimeCommand = 0;
 }
 if(found == false)
 {
   countTrueCommand = 0;
   countTimeCommand = 0;
 }
```

```
 found = false;

}
```

3.4.3 System Algorithm for Device 2

In the next segment, we have the working mechanism of the fall detection system along with the vaccination notification application. At first, we started with including the necessary libraries for the code. The MPU6050 uses the I2C protocol for communication and transfer of data. An I2C address needs to be defined for this MPU6050 to work. We defined the pins of SIM900A and LCD which will be connected with Arduino UNO and had declared the variables of the 6 axes of the accelerometer and gyroscope. SIM900A communicates with the MCU by serial communication. Here we have used Software Serial communication and have defined digital pins 8 and 9 as Rx and Tx respectively. We have initially set the values of these 6 axes and change in angle to zero and the Boolean variables for fall and the three Trigger conditions to FALSE. To keep a count of those trigger conditions, we have three bytes' variables set to zero as well. For taking the schedule of the vaccine reminder system, a string is created.

Code for Fall Detection:

We need to include all the required libraries at first. The ***Wire.h*** library allows us to communicate with I2C devices, the ***SoftwareSerial.h*** library enables serial data communication and the ***LiquidCrystal.h*** library is needed for the LCD monitor.

```
#include <Wire.h> //library for I2C devices' communication
#include<SoftwareSerial.h> //library for serial
communication #include <LiquidCrystal.h> //library for LCD
```

Here, we have defined the pins of Arduino UNO, SIM900A, LCD monitor as well as declared the variables of the 6 axes of accelerometer and gyroscope.

```
#define vcc2 12
SoftwareSerial sim900(8,9);
LiquidCrystal lcd(2,3,4,5,6,7);
const int MPU_addr=0x68; // I2C address of the MPU-6050
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
float ax=0, ay=0, az=0, gx=0, gy=0, gz=0; boolean
fall = false; //stores if a fall has occurred
boolean trigger1=false; //stores if lower threshold has occurred
boolean trigger2=false; //stores if upper threshold has occurred
boolean trigger3=false; //stores if orientation change has occurred
byte trigger1count=0; //stores the counts since trigger 1 was activated
byte trigger2count=0; //stores the counts since trigger 2 was activated
byte trigger3count=0; //stores the counts since trigger 3 was activated
int angleChange=0;
```

Inside the ***void setup()***, we have initialized the serial monitor and SIM900A at a baud rate of 9600. The wire library, the data transmission through the power management register as well as the LCD are also initialized.

```
void setup(){
  pinMode (vcc2,OUTPUT); // Setting pin 12 as output
  digitalWrite(vcc2,HIGH); // Setting vcc2 as second voltage source
  Serial.begin(9600);

  sim900.begin(9600);
  Serial.println ("SIM900A Ready");
  Serial.println ("Type s to send message or r to receive message");
  lcd.begin(16, 2);
  lcd.print("Initializing");
  delay(100);
  lcd.clear();
  Serial.println("Date & Time , Amp , Angle_change");
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0);    // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
```

```
}
```

Inside the ***void loop(),*** we read the MPU6050 sensor data. 2050, 77, 1947 are values for calibration of an accelerometer. 1947, 351 and 136 are values for calibration of the gyroscope. We use the calibration values when calculating amplitude vector and angle change.

```
void loop(){
 mpu_read();
 ax = (AcX-2050)/16384.00;
 ay = (AcY-77)/16384.00;
 az = (AcZ-1947)/16384.00;
 gx = (GyX+270)/131.07;
 gy = (GyY-351)/131.07;
 gz = (GyZ+136)/131.07;
```

After getting the accelerometer and gyroscope values, we calculate the amplitude vector of the accelerometer values. The serial monitor shows the amplitude vector and the angle change. We have also shown those values on the LCD monitor.

```
 // calculating Amplitude vector and angle change for 6 axes

float Raw_Amp = pow(pow(ax,2)+pow(ay,2)+pow(az,2),0.5);
angleChange = pow(pow(gx,2)+pow(gy,2)+pow(gz,2),0.5);

int Amp = Raw_Amp * 10;  // Multiplied by 10 because values are between 0 to 1
Serial.print(",");
Serial.print(Amp);
Serial.print(",");
Serial.println(angleChange);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Amp:");
lcd.print(Amp);
lcd.setCursor(0, 4);
lcd.print("angleChange:");
lcd.print(angleChange);
```

This program first checks if the accelerometer values (Amp) exceed the lower threshold which is 2. If the lower threshold is crossed and Trigger 2 is in FALSE condition, Trigger 1 is Activated and it waits for about 0.5 seconds to see whether the upper threshold,11, has crossed or not. The upper threshold signifies that the person might have fallen with some speed. If it crosses then Trigger 2 will be activated, otherwise Trigger 1 will become Deactivated. Once Trigger 2 is activated, it checks for the change in orientation given by the Gyroscope i.e., the angleChange value. If the orientation change is between 30° to 120° then Trigger 3 is Activated and it waits for some time to check whether the angleChange is within 0-10°. This signifies that the person who has fallen, is unable to move much and needs help to get up, so the screen displays “Fall Detected” and an alert message is sent to the emergency contact. But if the angleChange is more than 10°, it means the person has recovered from the fall and Trigger 3 becomes Deactivated.

```
    if (Amp<=2 && trigger2==false) {  //if system breaks lower threshold

   trigger1=true;
   Serial.println("TRIGGER 1 ACTIVATED");
   }
  if (trigger1==true) {
   trigger1count++;
   if (Amp>=11) { //if system breaks upper threshold
    trigger2=true;

    Serial.println("TRIGGER 2 ACTIVATED");
    trigger1=false; trigger1count=0;
    }
  }
  if (trigger2==true) {
   trigger2count++;
   angleChange = pow(pow(gx,2)+pow(gy,2)+pow(gz,2),0.5);
   //Serial.println(angleChange);
   if (angleChange>=30 && angleChange<=120) { //if orientation changes between 30-
    120⁰ trigger3=true; trigger2=false; trigger2count=0;
    Serial.println("TRIGGER 3 ACTIVATED");
  }
    }
  if (trigger3==true) {
    trigger3count++;
    if (trigger3count>=10) {
      angleChange =
      pow(pow(gx,2)+pow(gy,2)+pow(gz,2),0.5); delay(10);
      if ((angleChange>=0) && (angleChange<=10)){ //if orientation change remains
between 0-10 degrees
         fall=true; trigger3=false; trigger3count=0;
        }
      Else {  //user regained normal orientation
        trigger3=false; trigger3count=0;
        Serial.println("TRIGGER 3 DEACTIVATED");
      }
    }
  }
  if (fall==true){ //in case of a fall detection
```

```
    Serial.println("FALL DETECTED");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Fall Detected!");
    SendMessage();
    delay(1000);

    fall=false;
    }
  if (trigger2count>=6) { //allowing some time for orientation
    change trigger2=false; trigger2count=0;
    Serial.println("TRIGGER 2 DEACTIVATED");
    }
  if (trigger1count>=6) { //allowing some time for system to break upper
    threshold trigger1=false; trigger1count=0;
    Serial.println("TRIGGER 1 DEACTIVATED");
    }
    delay(100);
}
```

Inside the ***mpu_read loop()***, we read all the six registers for the X, Y, and Z axes of the Accelerometer and Gyroscope.

```
void mpu_read(){

  Wire.beginTransmission(MPU_addr);
  Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true); // request a total of 14 registers
  AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp=Wire.read()<<8|Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
 }
```

Lastly, we set the number and the context of the text that we want to send through SIM900A so that we get an SMS immediately when a fall is detected.

```
   void SendMessage()

{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   //Serial.println ("Set SMS Number");
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to send
message
   delay(110);
   //Serial.println ("Set SMS Content");
   sim900.println("Patient has fallen!");// Message content
   //Serial.println ("Finish");
   sim900.println((char)26);// ASCII code of CTRL+Z
```

```
    Serial.println ("Message has been sent");
}
```

Code for Vaccine notification:

We had created an array of Strings and had listed the age of the newborn child within this array.

```
char *myStrings[] = {"AfterBirth","6weeks","10weeks","14weeks","9months","1year","15months","2years"};
```

A For loop is used to send the vaccine alert messages according to the age of the child. First the string is called, and then the age within the array is checked, after that their corresponding message is sent to the given contact number.

```
for (int i = 0; i < 8; i++) {
     myStrings[i];
        if (myStrings[i]==myStrings[0]){
          SendMessage1();
        }
        else if (myStrings[i]==myStrings[1]){
          SendMessage2();
        }
        else if (myStrings[i]==myStrings[2]){
          SendMessage3();
        }
        else if (myStrings[i]==myStrings[3]){
          SendMessage4();
        }
        else if (myStrings[i]==myStrings[4]){
          SendMessage5();
        }
        else if (myStrings[i]==myStrings[5]){
          SendMessage6();
        }
        else if (myStrings[i]==myStrings[6]){
          SendMessage7();
        }
        else{
           SendMessage8();
        }
        delay(5000);

   }
```

We set the number and the context of the text that we want to send through SIM900A during vaccine notification. 8 different messages for 8 different age groups.

```
void SendMessage1()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1");    //Sets the GSM Module in Text Mode
```

```
   delay(110);
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to
send message
   delay(110);
    sim900.println("Time for vaccine: BCG, Oral Polio(1st dose)");// Message
   content sim900.println((char)26);// ASCII code of CTRL+Z Serial.println
   ("Message has been sent");
}

void SendMessage2()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to
send message
   delay(110);
   sim900.println("Time for vaccine: Pentavalent(1st dose), Oral Polio(2nd dose)");//
Message content
     sim900.println((char)26); // ASCII code of CTRL+Z
    Serial.println ("Message has been sent");
}

void SendMessage3()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to
send message
   delay(110);
   sim900.println("Time for vaccine: Pentavalent(2nd dose), Oral Polio(3rd dose)");//
Message content
   sim900.println((char)26);// ASCII code of CTRL+Z
   Serial.println ("Message has been sent");
}

void SendMessage4()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   //Serial.println ("Set SMS Number");
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to send
message
```

```
   delay(110);
   //Serial.println ("Set SMS Content");
   sim900.println("Time for vaccine: Pentavalent(3rd dose), Oral Polio(4th dose)");// Message
content
   //delay(100);
   //Serial.println ("Finish");
   sim900.println((char)26);// ASCII code of CTRL+Z
   //delay(1000);
   Serial.println ("Message has been sent");
}

void SendMessage5()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   //Serial.println ("Set SMS Number");
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to send
message
   delay(110);
   //Serial.println ("Set SMS Content");
   sim900.println("Time for vaccine: MR(Measles and Rubella)");// Message content
   //delay(100);
   //Serial.println ("Finish");
   sim900.println((char)26);// ASCII code of CTRL+Z
   //delay(1000);
   Serial.println ("Message has been sent");
}

void SendMessage6()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   //Serial.println ("Set SMS Number");
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to send
message
   delay(110);
   //Serial.println ("Set SMS Content");
   sim900.println("Time for vaccine: Varilrix");// Message content
   //delay(100);
   //Serial.println ("Finish");
   sim900.println((char)26);// ASCII code of CTRL+Z
   //delay(1000);
   Serial.println ("Message has been sent");
}
```

```
void SendMessage7()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   //Serial.println ("Set SMS Number");
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to send
message
   delay(110);
   //Serial.println ("Set SMS Content");
   sim900.println("Time for vaccine: Oral Polio(5th dose), Measles");// Message content
   //delay(100);
   //Serial.println ("Finish");
   sim900.println((char)26);// ASCII code of CTRL+Z
   //delay(1000);
   Serial.println ("Message has been sent");
}

void SendMessage8()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   //Serial.println ("Set SMS Number");
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to send
message
   delay(110);
   //Serial.println ("Set SMS Content");
   sim900.println("Time for vaccine: Typhoid"); // Message content
   //delay(100);
   //Serial.println ("Finish");
   sim900.println((char)26); // ASCII code of CTRL+Z
   //delay(1000);
   Serial.println ("Message has been sent");
}
```

Code for MAX30100:

We need to include the library files for MAX30100 and LCD monitor. Since this sensor works by I2C communication, we had to include the ***Wire.h*** library. After that, we defined the reporting period of the

```
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"
#include <LiquidCrystal.h>

#define REPORTING_PERIOD_MS     1000
LiquidCrystal lcd(36,35,34,33,32,31);

PulseOximeter pox;

uint32_t tsLastReport = 0;
```

"Beat" is printed into the serial monitor whenever a heartbeat is detected. In the ***void setup()*** the LCD monitor and the MAX30100 sensor is initialized.

```
void onBeatDetected()
{
    Serial.println("Beat!");   // printed when heartbeat is detected
}

void setup()
{
    Serial.begin(9600);
    lcd.begin(16, 2);
    lcd.print("Initializing");
    lcd.clear();
    delay(100);
    Serial.print("Initializing pulse oximeter..");
    Serial.println("HeartRate , SpO2");

    if (!pox.begin()) {
        Serial.println("FAILED");          // when initialization fails
        for(;;);
    } else {
        Serial.println("SUCCESS");       // when initialization is successful
    }
```

The Arduino calls the ***pox.getHeartRate()*** and ***pox.getSpO2()*** functions from the library to read the Heart rate and SpO2 readings and these readings are printed into the serial and LCD monitor. The MAX30100 sensor requires the use of ***millis()*** function due to which each of the set of readings comes at 1 second interval.

```
void loop()
{
   pox.update();

   if (millis() - tsLastReport > REPORTING_PERIOD_MS)
      { Serial.print(",");
      Serial.print(pox.getHeartRate()); // printing heart rate value in serial monitor
      Serial.print(",");
      Serial.print(pox.getSpO2()); // printing SpO2 value in serial monitor

      Serial.println("%");

      tsLastReport = millis();   // readings taken at 1 second interval
   }
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("HR:");
   lcd.print(pox.getHeartRate());
   lcd.setCursor(0, 2);
   lcd.print("SpO2:");
   lcd.print(pox.getSpO2());

}
```

3.5 Images of the LCD Interface

Below are some of the images to show that we are getting all the readings of the fixed parameters from the devices which can be displayed on the LCD monitor.

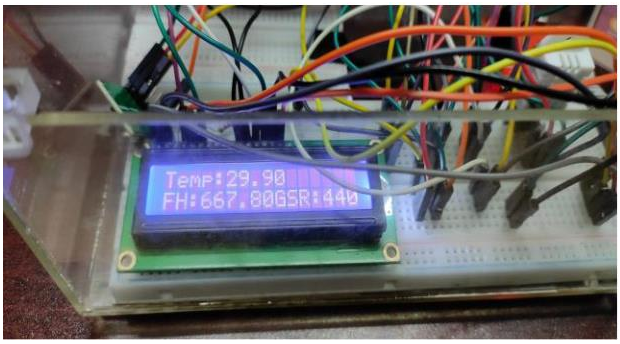

Figure 20: Readings from the first prototype

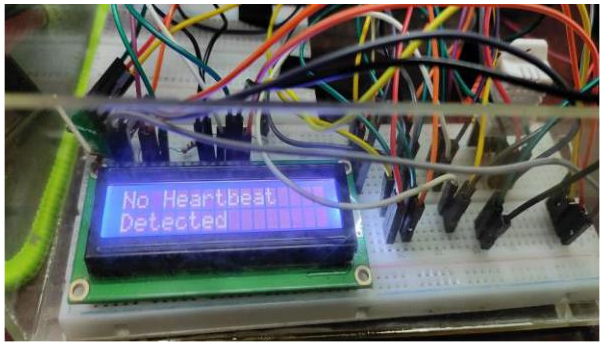

Figure 21: When no fetal heart rate detected

Figure 22: Readings from MPU6050

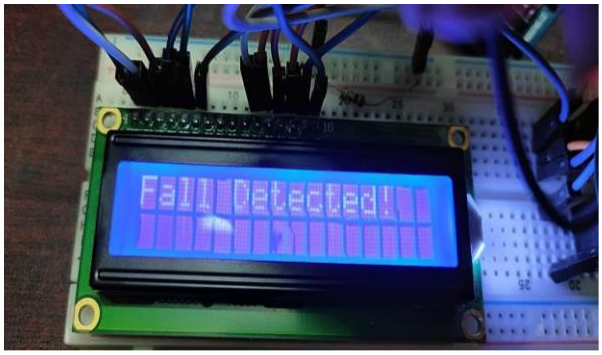

Figure 23: Fall detected in Fall Detection System

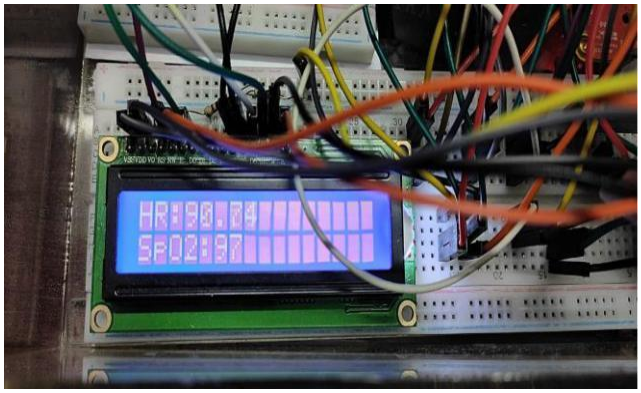

Figure 24: Readings from MAX30100

3.6 ThingSpeak Limitations

The free version that we are using has certain limitations:

- Only four channels can be created at a time.
- Data is updated at a 15 seconds interval so we could not send data every second.
- Private channel sharing is limited to only 3 shares.
- The channels will not be able to receive any data if its storage limit is exceeded.

3.7 Conclusion

To conclude the chapter, we would like to mention a few major points. We have seen the working mechanism with the code and device setup. Though we started with NodeMCU for sending the data to ThingSpeak, we could not implement it. It was hard interfacing it with Arduino Mega and we could not send data to the server, which is why we had to replace it with the ESP8266 Wi-Fi module. ThingSpeak is chosen as the cloud as it is a hassle-free open-source IoT platform service for building IoT prototypes rapidly. Apart from the restrictions arising because of its free version, it has several key features which makes it user-friendly like we do not need to set up servers or develop web software and we can visualize our sensor data in real-time. In the next chapter, we will be covering our data collection mechanism and its analysis.

Chapter 4

Data Collection and Analysis

4.1 Introduction

This chapter focuses on the data collection of the health parameters from a pregnant woman, and four other people of different age groups at a specific time of the day using the two devices. These data are observed carefully to see the trend that they have created and find out the possible reasons behind certain types of data trends. There are also some anomalies in the data reading which is also justified with reason in this chapter. These data are also used to generate graphs with the help of 'Origin' software.

4.2 Device 1 and MAX30100 Data Analysis

In this section, we will be analyzing the data collected from a 20-year-old pregnant woman and 4 different people of ages 45, 23, 10 and 55 using Device 1 and MAX30100 from Device 2. The MAX30100 analysis has been put here to make it easier to evaluate each person's overall health condition. Therefore, we will be looking into each of these five people's temperature, humidity, heart rate, blood oxygen saturation and stress levels along with the pregnant woman's fetal heart rate. Since we are using DHT22 to measure the temperature, we are getting humidity values as well but as humidity does not indicate any specific health issue in a person so we did not analyze the humidity readings elaborately here.

4.2.1 Analysis on a 20-year-old pregnant woman

To confirm the functionality of our device, we took readings from a 20-year-old pregnant woman who was in her third trimester (7 months). We have tabulated the temperature, humidity, GSR and AD8232 readings in the table below. The table contains 32 sets of data.

We took the data for 8 minutes at an interval of around 15 seconds since ThingSpeak does not allow us to take readings at less than a 15-second interval. The graphs would have been more accurate if we could take data at one-second intervals.

Time (HH:MM:SS)	**Temperature(°C)**	**Humidity (%)**	**GSR (µS)**	**AD8232(mV)**
12:49:59	33.8	83.6	321	652.9
12:50:14	33.9	83.4	318	77.8
12:50:30	34.1	83.7	310	223.3
12:50:45	34.2	83.3	297	643.4
12:51:01	34.3	83.1	302	678.2
12:51:18	34.3	83.9	317	13.6
12:51:33	34.3	83.8	321	62.5
12:51:49	34.4	84.8	314	87.1
12:52:05	34.4	84.8	264	307.3
12:52:20	34.4	84.7	271	563.6
12:52:37	34.4	84.1	302	691.3
12:52:53	34.5	85.9	291	691.7
12:53:08	34.5	85.9	326	691.9
12:53:25	34.5	85.6	323	691.9
12:53:40	34.5	85.3	330	97.6
12:53:56	34.5	85.6	221	132.7
12:54:12	34.5	85.7	332	668.9
12:54:28	34.6	84.2	315	692.2

12:54:44	34.6	85.8	318	592.7
12:54:59	34.7	88.4	318	582.7
12:55:15	34.7	89.4	316	677.4
12:55:32	34.8	89.3	303	431.8
12:55:47	34.8	89.5	305	677.7
12:56:02	34.8	86.7	310	573.5
12:56:18	34.9	84.9	305	62.9
12:56:33	35.1	92.1	304	434.8
12:56:49	35.2	93.4	284	12.6
12:57:06	35.3	93.3	279	14.7
12:57:21	35.3	92.7	303	434.6
12:57:37	35.2	93.1	332	432.8
12:57:52	35.2	93.1	283	405.2
12:58:08	35.1	93.2	246	259.5

Table 2: Temperature, Humidity, GSR, and AD8232 readings for the 20-year-old pregnant woman

The place where she was staying was quite heated, so her body temperature readings showed high values which were above 35°C. Since the woman we experimented on is of a young age and has no academic pressure to worry about, her GSR values were within the range of 217-332µS which is considered to be low, meaning the subject has less stress. The sensor AD8232 gave high readings when we placed the ECG electrodes on the subject's abdomen, indicating that it was able to detect the fetal heart rate. When the ECG electrodes were placed on a normal person's abdomen it showed no readings. We have plotted these readings using the "Origin" software. The graphs are shown below.

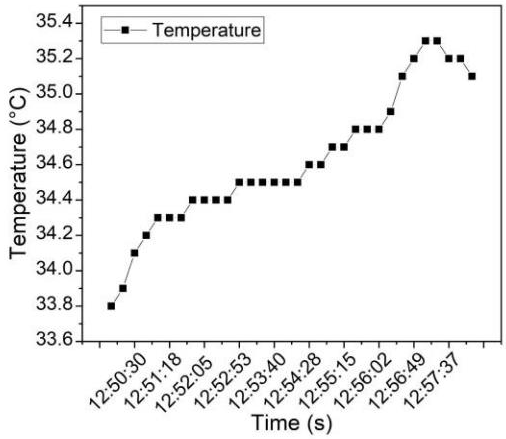

Figure 25: Pregnant woman Temperature curve

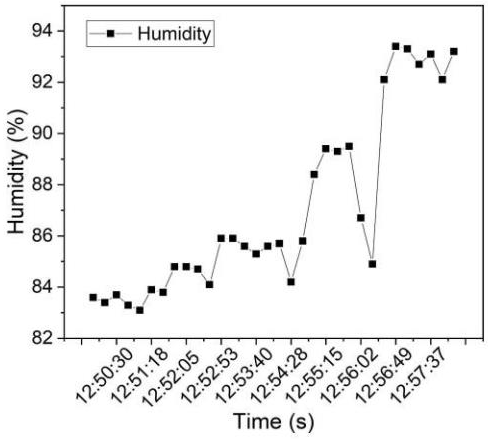

Figure 26: Pregnant woman Humidity curve

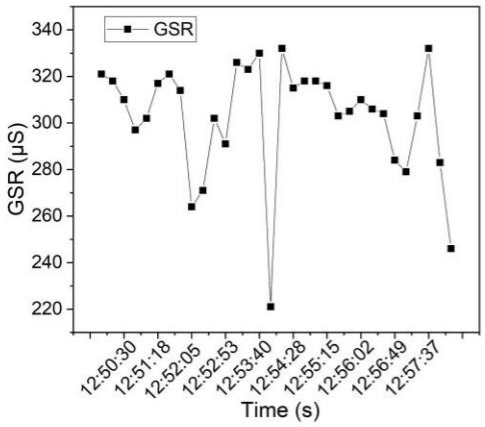

Figure 27: Pregnant woman GSR curve

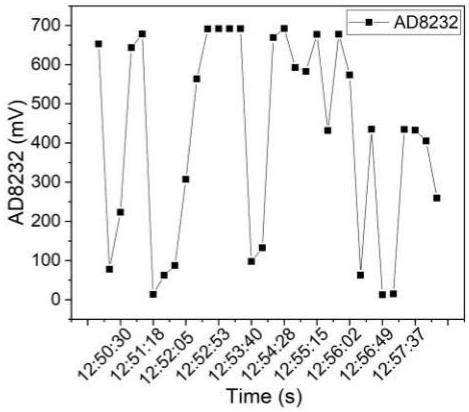

Figure 28: Pregnant woman AD8232 curve

The table below shows the heart rate and the blood oxygen saturation data of the pregnant woman. We have taken the readings for 45 seconds at an interval of 1 second. The table contains 45 data.

Time (HH:MM:SS)	Heart rate (bpm)	SpO2 (%)
13:06:32	23.54	0
13:06:33	75.81	97

13:06:34	118.2	97
13:06:35	82.28	94
13:06:36	59.38	94
13:06:37	59.38	94
13:06:38	33.56	94
13:06:39	33.56	94
13:06:40	38.58	95
13:06:41	38.58	95
13:06:42	43.2	0
13:06:43	78.44	95
13:06:44	98.72	95
13:06:45	141.75	96
13:06:46	120.12	96
13:06:47	106.98	95
13:06:48	101.51	94
13:06:49	96.33	94
13:06:50	92.72	96

13:06:51	92.72	96
13:06:52	77.55	96
13:06:53	132.57	95
13:06:54	107.93	95
13:06:55	94.82	95
13:06:56	99.73	95
13:06:57	95.59	95
13:06:58	96.34	96
13:06:59	100.23	96
13:07:00	97.74	96
13:07:01	122.99	96
13:07:02	104.9	96
13:07:03	102.43	96
13:07:04	67.18	95
13:07:05	60.92	95
13:07:06	66.55	95
13:07:07	111.88	95

13:07:08	193.33	95
13:07:09	126.44	96
13:07:10	103.72	96
13:07:11	130.16	95
13:07:12	124.95	95
13:07:13	76.5	94
13:07:14	76.5	94
13:07:15	0	0
13:07:16	19.55	0

Table 3: Heart Rate and SpO2 readings for the 20-year-old pregnant woman

During pregnancy, the mean maternal heart rate usually increases by an average of 10 to 20 beats per minute which is normal[40]. Moreover, as mentioned before, the body temperature of the subject came out to be high and she was not in a resting state when we went there which has caused the heart rate to increase up to 120bpm which is a bit more than the usual pregnancy heart rate, that is up to 113bpm in the third trimester[41]. Our table shows some anomaly values which are above 120bpm and below 59bpm. This is due to the fact that the sensors we used are not high-end sensors, thus the readings are not entirely precise and vary with even a slight movement.

The blood oxygen saturation (SpO2) values of a pregnant woman in her third trimester are supposed to be in the range 93.4%-98.5%[41] and almost all of the values of our subject were within the range 94%-96%, which is normal.

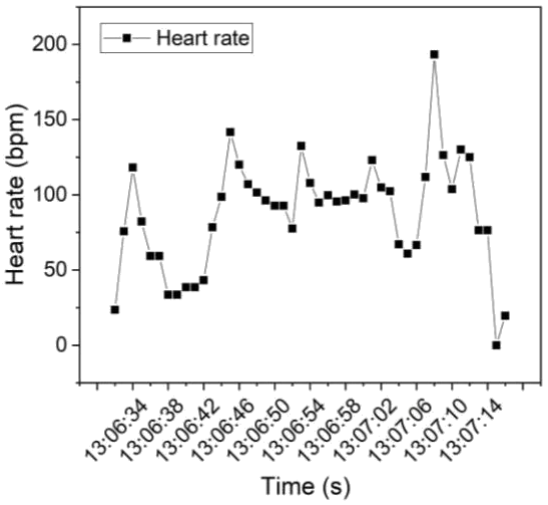

Figure 29: Pregnant woman Heart rate curve

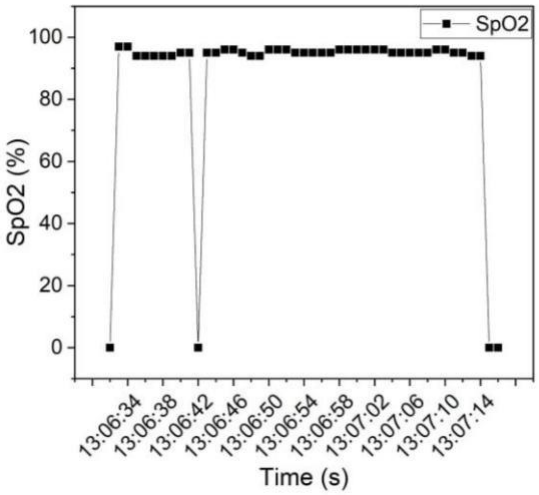

Figure 30: Pregnant woman SpO2 curve

We were informed by this pregnant lady that she does not have any serious health issues. This was confirmed by the sensor readings which were within the normal range and did not cross any threshold values. Therefore, we can conclude that this person was in good health. The image below shows the pregnant woman we conducted our analysis on.

Figure 31: 20-year-old pregnant woman

Next, we had carried out the similar procedure with normal people of different age groups. We took readings from a 45-year-old woman, a 23-year-old girl, a 10-year-old boy and a 55-year-old man. The tables consisting of the temperature, humidity and GSR values contain 32 sets of data and the tables containing the heart rate and SpO2 readings contain 45 sets of data. The AD8232 sensor is used for measuring fetal heart rate, therefore in those 4 cases, the sensor gave no readings.

4.2.2 Analysis on a 45-year-old woman

The table below shows the temperature, humidity and the GSR sensors' readings of the 45-year-old woman who has no known health issue. We have taken the readings for 8 minutes at an interval of around 15 seconds. The table contains 32 sample data.

Time (HH:MM:SS)	**Temperature (°C)**	**Humidity (%)**	**GSR (µS)**
22:41:33	33.2	90	495
22:41:48	33.2	90.7	497
22:42:03	33.2	91	495
22:42:19	33.2	91.4	497
22:42:35	33.2	91.7	496
22:42:50	33.2	91.4	498
22:43:07	33.2	90.2	496
22:43:24	33.2	90.9	499
22:43:39	33.2	91.2	499
22:43:55	33.2	91.6	499

22:44:10	33.2	92.1	501
22:44:26	33.2	92.7	502
22:44:43	33.2	92.7	501
22:44:58	33.2	92.4	509
22:45:14	33.2	91.4	506
22:45:29	33.2	88.3	507
22:45:46	33.2	88.4	508
22:46:03	33.2	87.9	507
22:46:19	33.2	86.6	507
22:46:34	33.2	85.7	507
22:46:49	33.2	85.5	511
22:47:06	33.2	87	510
22:47:22	33.2	88.1	510
22:47:37	33.2	87.9	509
22:47:52	33.1	86	510
22:48:08	33.1	87.2	510
22:48:24	33.1	89.3	509

22:48:41	33.1	90.2	509
22:48:56	33.1	90.5	509
22:49:11	33.1	90.8	508
22:49:28	33.1	91.1	508
22:49:44	33.1	91.2	508
22:49:59	33.1	91.4	508

Table 4: Temperature, Humidity, and GSR readings for the 45-year-old woman

The temperature of this person is around 33°C which is normal but is less than the pregnant woman since she stays in a relatively cooler environment. The GSR value of this 45-year-old woman ranged from 495 to 511µS, which is reasonable given that, when compared to the stress level of the 20-year-old pregnant woman, a much older woman is expected to be under a lot of stress because she is responsible for the entire household as well as her children.

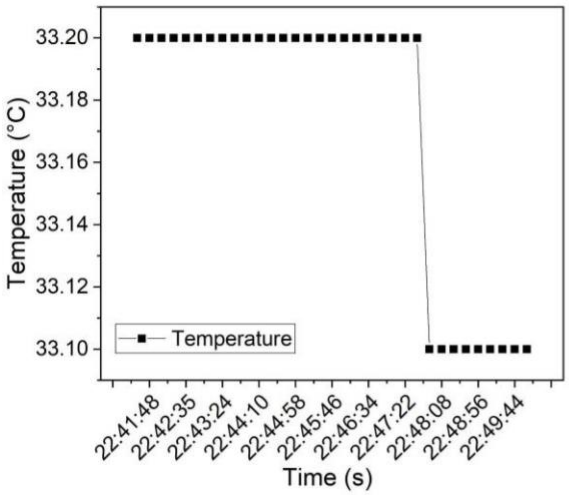

Figure 32: 45-year-old woman Temperature curve

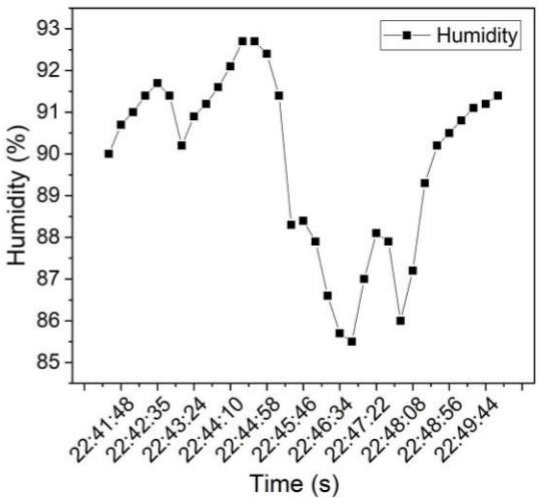

Figure 33: 45-year-old woman Humidity curve

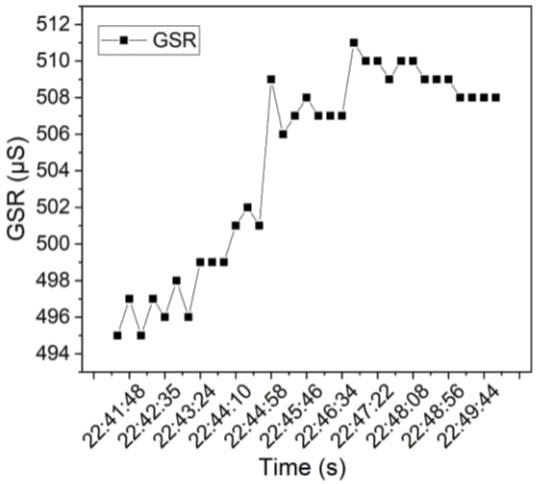

Figure 34: 45-year-old woman GSR curve

The table below shows the heart rate and the blood oxygen saturation data of the 45-year-old woman. We have taken the readings for 45 seconds at an interval of 1 second. The table contains 45 sample data.

Time (HH:MM:SS)	**Heart rate (bpm)**	**SPO2 (%)**
01:05:45	28	0
01:05:46	29.87	0
01:05:47	48.7	94
01:05:48	48.7	94
01:05:49	55.91	94
01:05:50	55.91	94
01:05:51	56.2	94
01:05:52	80.25	94

01:05:53	72.55	94
01:05:54	61.04	94
01:05:55	71.11	94
01:05:56	138.04	94
01:05:57	138.04	94
01:05:59	84.45	94
01:06:00	97.31	97
01:06:01	98.13	97
01:06:02	97.8	97
01:06:03	97.14	97
01:06:04	97.14	97
01:06:05	76.86	97
01:06:06	76.86	97
01:06:07	73.93	97
01:06:08	92.22	97
01:06:09	91.51	97
01:06:10	97.31	97

01:06:11	96.61	97
01:06:12	59.28	97
01:06:13	94.45	97
01:06:14	94.45	97
01:06:15	81.7	97
01:06:16	93.47	97
01:06:17	90.75	97
01:06:18	96.51	97
01:06:19	67.73	97
01:06:20	71.4	95
01:06:21	53.33	95
01:06:22	53.33	95
01:06:23	68.76	97
01:06:24	82.87	97
01:06:25	56.7	97
01:06:26	89.48	97
01:06:27	87.87	97

01:06:28	95.87	97
01:06:29	98.4	97
01:06:30	92.66	97

Table 5: Heart Rate and SpO2 readings for the 45-year-old woman

The heart rate of this individual ranged from 59.28-98.4 bpm which is usual(60-100bpm) and as mentioned before, the values under 59 and above 100 are faulty values given by the sensor. The SpO2 values ranged from 94-97%, which is within the normal range (95-100%) also.

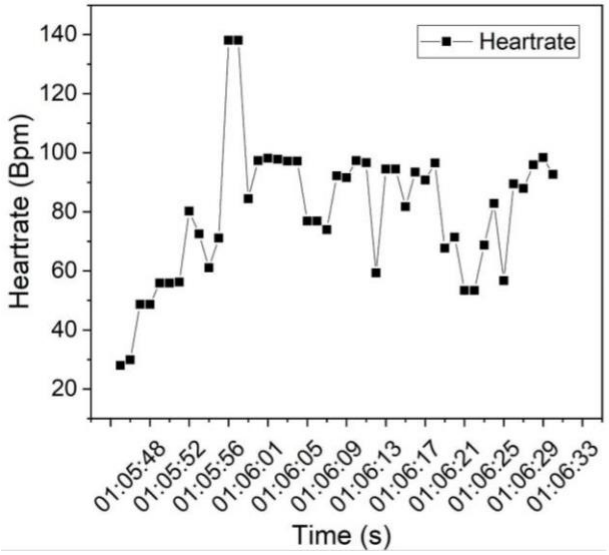

Figure 35: 45-year-old woman heart rate curve

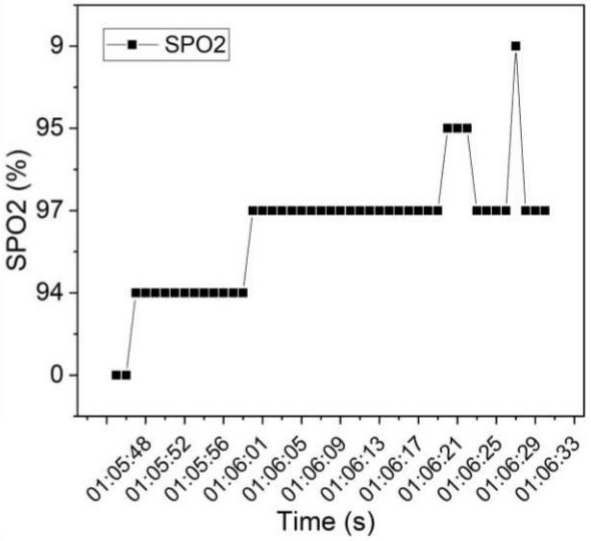

Figure 36: 45-year-old woman SpO2 curve

For this woman, all the sensor values except GSR sensor were within the normal range. The value of GSR sensor was higher than the reference value (350 µS) indicating that this woman is at a risk of developing stress related health problems.

4.2.3 Analysis on a 23-year-old girl

The table below shows the temperature, humidity and the GSR sensors' readings of the 23-year-old girl and we were told that this individual has no particular health problem. We have taken the readings for 8 minutes at an interval of around 15 seconds. The table contains 32 sample data.

Time (HH:MM:SS)	Temperature (°C)	Humidity (%)	GSR (µS)
22:34:58	32.5	87.7	468
22:35:13	32.5	87.4	477
22:35:28	32.5	87.4	467
22:35:43	32.5	87.6	468
22:36:59	32.5	87.8	469
22:36:16	32.5	87.8	470
22:36:32	32.5	88	474
22:36:48	32.5	88	466
22:37:02	32.5	88	465
22:37:19	32.5	88.2	464
22:37:34	32.5	88.3	464
22:37:49	32.5	88.3	466
22:38:04	32.5	87.9	468

22:38:21	32.4	87.6	467
22:38:37	32.5	87.7	464
22:38:52	32.4	87.3	466
22:39:09	32.5	86.7	472
22:39:24	32.5	86.2	462
22:39:40	32.5	86	461
22:39:56	32.5	85.8	461
22:40:12	32.5	84.9	462
22:40:28	32.4	84.5	461
22:40:43	32.5	85.3	461
22:40:59	32.5	86	462
22:41:14	32.4	86.2	464
22:41:30	32.4	85.1	459
22:41:47	32.4	84.3	460
22:42:02	32.4	84.6	463
22:42:18	32.4	85.3	461
22:42:33	32.4	85.6	461

22:42:48	32.4	86.2	466
22:43:03	32.4	86.6	461
22:43:58	32.4	87	463

Table 6: Temperature, Humidity, and GSR readings for the 23-year-old girl

The temperature of this 23-year-old girl is around 32.5°C which is within the normal range. The GSR values of this person are within the range of 460-477µS which is lower than the 45-year-old woman but higher than the 20-year-old pregnant woman and that is reasonable given that, when compared to the stress level of the 20-year-old pregnant woman, this individual has a good amount of academic pressure and thus is expected to be under a lot of stress but this stress does not amount up to the stress of the 45-year-old individual who has to manage a household.

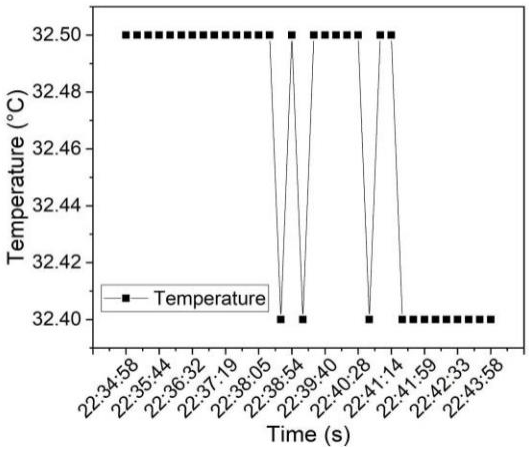

Figure 37: 23-year-old girl Temperature curve

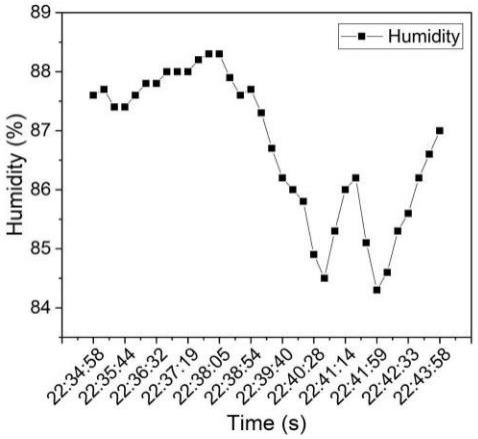

Figure 38: 23-year-old girl Humidity curve

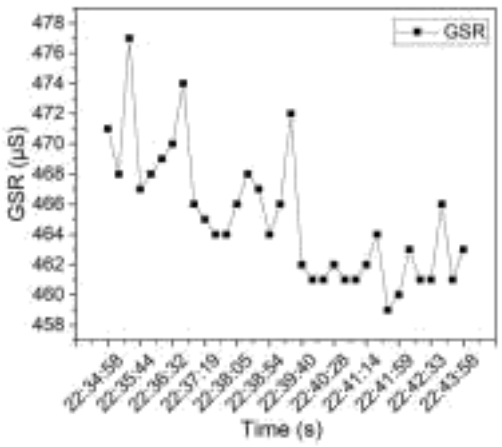

Figure 39: 23-year-old girl GSR curve

The table below shows the heart rate and the blood oxygen saturation data of the 23-year-old girl. We have taken the readings for 45 seconds at an interval of 1 second. The table contains 45 sample data.

Time (HH:MM:SS)	**Heart rate(bpm)**	**SpO2 (%)**
01:09:26	88.81	97
01:09:27	90.22	97
01:09:28	92.91	97
01:09:29	90.02	97
01:09:30	90.06	97
01:09:31	92.43	97
01:09:32	90.93	97
01:09:33	92.03	97
01:09:34	91.52	97

01:09:35	94.59	97
01:09:36	101.55	97
01:09:37	100.01	97
01:09:38	98.97	96
01:09:39	100.38	96
01:09:40	100.35	96
01:09:41	97.56	96
01:09:42	99.13	96
01:09:43	98.86	97
01:09:44	96.54	97
01:09:45	98.78	96
01:09:46	100.65	96
01:09:47	99.7	96
01:09:48	100.79	96
01:09:49	90.5	96
01:09:50	90.25	96
01:09:51	90.3	96

01:09:52	92.17	96
01:09:53	96.62	96
01:09:54	93.95	97
01:09:55	92.96	97
01:09:56	94.33	97
01:09:57	92.42	97
01:09:58	91.63	97
01:09:59	92.8	97
01:10:00	90.25	97
01:10:01	89.35	97
01:10:02	90.23	97
01:10:03	89.34	97
01:10:04	87.79	97
01:10:05	90.45	97
01:10:06	93.03	97
01:10:07	91.74	97
01:10:08.	91.49	97

01:10:09	95.34	97
01:10:10	91.72	97

Table 7: Heart Rate and SpO2 readings for the 23-year-old girl

The heart rate of this individual ranged from 88.81-99.7 bpm which is normal and a few of the values were above 100 which is due to sensor sensitivity issues.

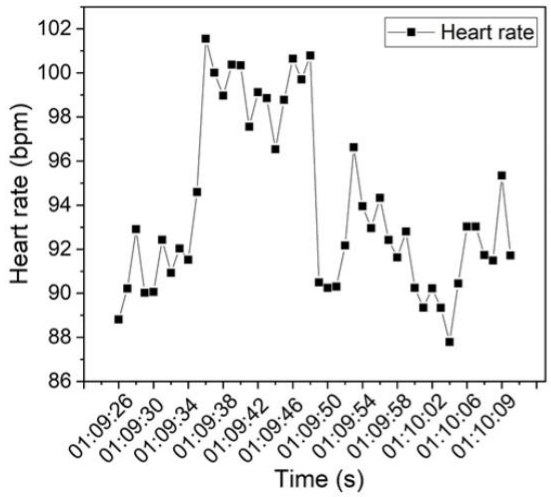

Figure 40: 23-year-old girl heart rate curve

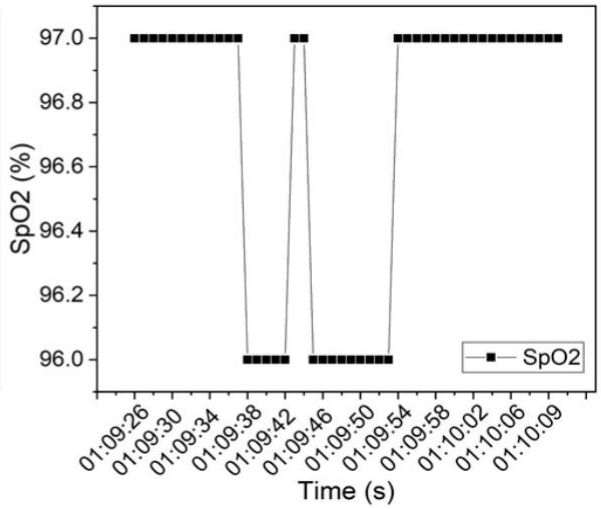

Figure 41: 23-year-old girl SpO2 curve

For this girl, all the sensor values except GSR sensor were within the normal range. The value of GSR sensor was higher than the reference value (350 µS) indicating that this girl is also at a risk of developing stress related health problems.

4.2.4 Analysis on a 55-year-old man

The table below shows the temperature, humidity and the GSR sensors' readings of the 55-year-old man. We were informed that this person suffers from Stress Hyperglycemia sometimes. We have taken the readings for around 8 minutes at an interval of 15 seconds. The table contains 32 sample data.

Time (HH:MM:SS)	Temperature(°C)	Humidity (%)	GSR (µS)
22:52:47	31.4	81.7	601
22:53:02	31.4	81.6	593
22:53:17	31.4	81.6	598
22:53:32	31.4	81.5	604
22:53:47	31.4	81.5	593
22:54:02	31.4	81.6	587
22:54:17	31.6	81.7	603
22:54:32	31.6	81.7	603
22:54:47	31.7	82	604
22:55:02	31.7	82.8	602
22:55:17	31.9	82.8	607
22:55:32	31.9	82.8	608

22:55:47	32	83.4	602
22:56:02	32.1	83	599
22:56:17	32.2	83.6	606
22:56:32	32.3	82.9	586
22:56:47	32.3	82.6	578
22:57:02	32.4	82.4	576
22:57:17	32.5	82.8	575
22:57:32	32.6	83.2	575
22:57:47	32.6	82.5	582
22:58:02	32.6	82.1	585
22:58:17	32.7	82.5	579
22:58:32	32.7	82.7	588
22:58:47	32.8	82.5	591
22:59:02	32.8	82.5	596
22:59:17	32.9	82.6	602
22:59:32	32.9	82.5	603
22:59:47	32.9	82.3	599

22:60:02	32.9	82.3	601
22:60:17	32.9	82.3	604
22:60:32	32.9	82.3	603
22:60:47	32.9	82.3	602

Table 8: Temperature, Humidity, and GSR readings for the 55-year-old man

The temperature of this 55-year-old man is around 33°C which is well within the normal range. This person's GSR readings varied from 582-607µS, indicating high stress, which is understandable considering that a 55-year-old man has a lot of obligations on his shoulders. This has crossed our set GSR threshold of 540µS and as a result, an alert message was sent to the emergency contacts' phone number.

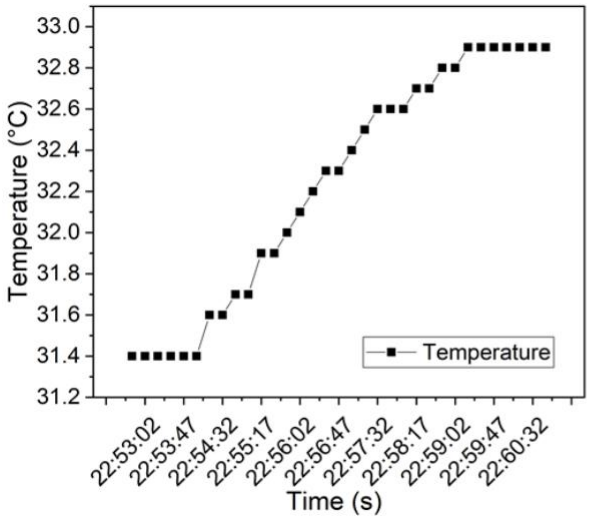

Figure 42: 55-year-old man Temperature curve

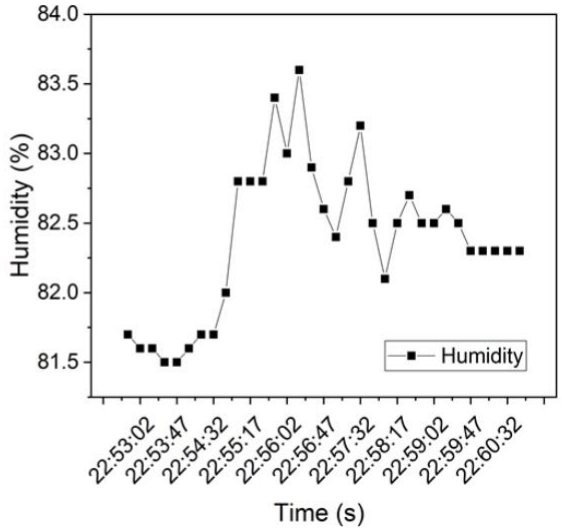

Figure 43: 55-year-old man Humidity curve

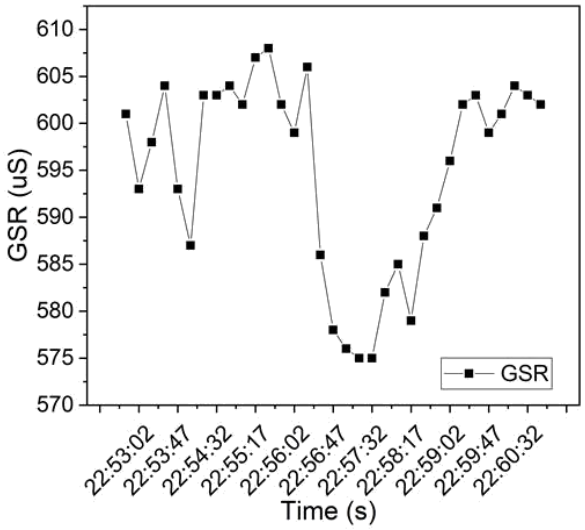

Figure 44: 55-year-old man GSR curve

The table below shows the heart rate and the blood oxygen saturation data of the 55-year-old man. We have taken the readings for 45 seconds at an interval of 1 second. The table contains 45 sample data.

Time (HH:MM:SS)	**Heart Rate (bpm)**	**SpO2 (%)**
01:02:56	91.54	95
01:02:57	96.12	96
01:02:58	96.98	96
01:02:59	98.62	96
01:03:00	99.32	96
01:03:01	96.36	96
01:03:02	96.37	96
01:03:03	97.09	96

01:03:04	96.43	96
01:03:05	95.7	96
01:03:06	94.97	96
01:03:07	94.56	96
01:03:08	95.62	96
01:03:09	94.94	96
01:03:10	94.74	96
01:03:11	98.4	96
01:03:12	89.96	96
01:03:13	97.12	96
01:03:14	103.29	96
01:03:15	94.62	96
01:03:16	98.25	96
01:03:17	60.02	96
01:03:18	90.53	96
01:03:19	94.53	96
01:03:20	101.27	96

01:03:21	101.52	96
01:03:22	102.35	96
01:03:23	101.35	96
01:03:24	99.91	96
01:03:25	99.57	96
01:03:26	100.33	96
01:03:27	99.69	96
01:03:28	98.5	96
01:03:29	95.33	96
01:03:30	99.12	96
01:03:31	95.46	96
01:03:32	101.67	96
01:03:33	102.63	96
01:03:34	100.63	96
01:03:35	101.94	96
01:03:36	102.46	96
01:03:37	97.62	96

01:03:38	99.61	96
01:03:39	100.95	96
01:03:40	104.03	96
01:03:41	96.17	96

Table 9: Heart Rate and SpO2 readings for the 55-year-old man

The heart rate values varied mostly from 91.54-99.32 bpm and the value of SpO2 was 96%. Both of the values were within the normal range.

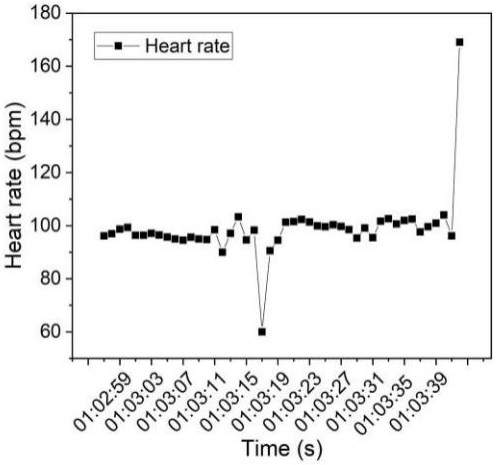

Figure 45: 55-year-old man heart rate

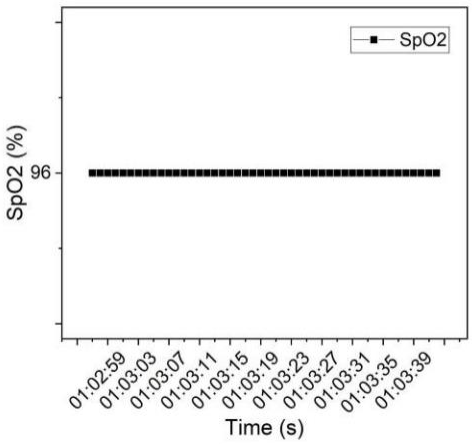

Figure 46: 55-year-old man SpO2

This person has very high stress levels, which means that he is at a greater risk of having high blood pressure or heart disease. Moreover, he has already suffered from Stress Hyperglycemia a couple of times so there is a high chance that he can develop diabetes in the future.

4.2.5 Analysis on a 10-year-old boy

The table below shows the temperature, humidity and the GSR sensors' readings of the 10-year-old boy. We were informed that the boy has no serious health issues. We have taken the readings for around 8 minutes at an interval of 15 seconds. The table contains 32 sample data.

Time (HH:MM:SS)	Temperature (°C)	Humidity (%)	GSR (µS)
22:20:59	33	92	382
22:21:14	32.9	92.1	384
22:21:29	32.9	92.1	383
22:21:44	33	92.1	386
22:21:59	33	92.1	385
22:22:14	33	92	386
22:22:29	33	92	387
22:22:44	33	92	388
22:22:59	33	92.1	388
22:23:14	33	92.2	390
22:23:29	33	92.2	389
22:23:44	33	92.2	362
22:23:59	33	92.2	358

22:24:14	33	92.1	356
22:24:29	33	92.1	359
22:24:44	33	91.8	371
22:24:59	33	91.9	380
22:25:14	33	91.7	388
22:25:29	33	91.7	386
22:25:44	33.1	91.7	386
22:25:59	33.1	91.8	387
22:26:14	33.1	91.9	390
22:26:29	33.1	91.9	391
22:26:44	33.1	91.9	391
22:26:59	33.1	91.6	397
22:27:14	33.1	91.3	400
22:27:29	33.1	91.1	398
22:27:44	33.1	90.5	398
22:27:59	33.1	90.3	401
22:28:14	33.1	90.1	400

22:28:29	33.1	90.7	399
22:28:44	33.1	90.8	396
22:28:59	33.1	91.1	397

Table 10: Temperature, Humidity, and GSR readings for the 10-year-old boy

The temperature of this 10-year-old boy is around 33°C, which is within the normal range. He has GSR sensor values ranging from 356-401µS which is low and that is reasonable considering the fact that this individual is at a much younger age and has very little to be stressed about.

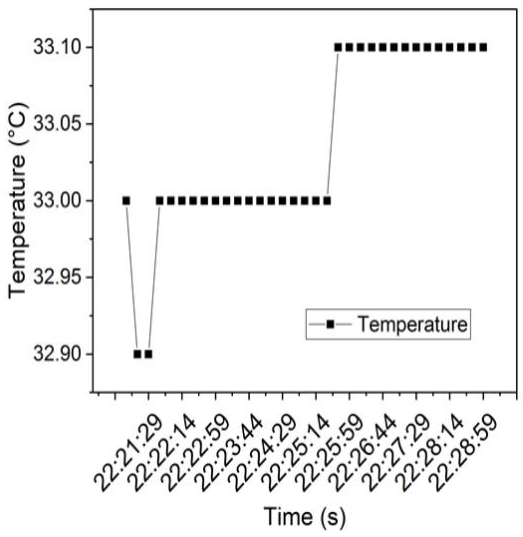

Figure 47: 10-year-old boy Temperature curve

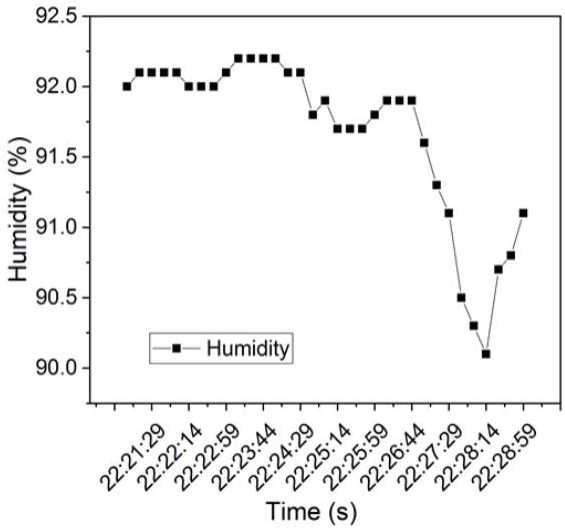

Figure 48: 10-year-old boy Humidity curve

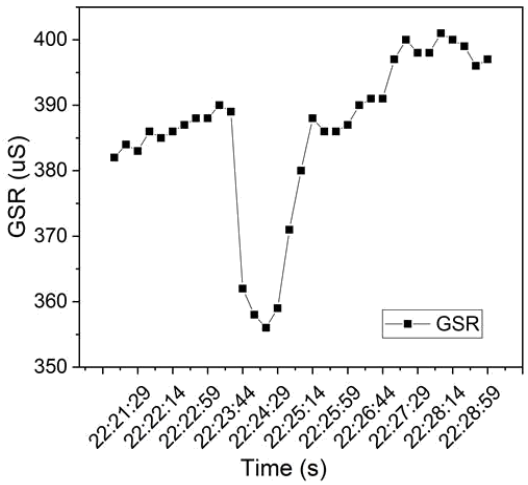

Figure 49: 10-year-old boy GSR curve

The table below shows the heart rate and the blood oxygen saturation data of the 10-year-old boy. We have taken the readings for 45 seconds at an interval of 1 second. The table contains 45 sample data.

Time (HH:MM:SS)	**Heart rate (bpm)**	**SpO2 (%)**
00:54:12	60.37	238
00:54:13	67.04	95
00:54:14	58.77	95
00:54:15	98.15	95
00:54:16	98.15	95
00:54:17	90.53	95
00:54:18	146.73	94
00:54:19	106.77	94

00:54:20	120.39	94
00:54:21	82.02	94
00:54:22	59.04	96
00:54:23	72.92	96
00:54:24	72.92	96
00:54:25	68.86	93
00:54:26	98.92	93
00:54:27	86.36	93
00:54:28	85.38	93
00:54:29	58.26	94
00:54:30	78.33	94
00:54:31	54.06	94
00:54:32	54.06	94
00:54:33	37.75	96
00:54:34	67.62	96
00:54:35	126.9	95
00:54:36	119.29	94

00:54:37	80.76	94
00:54:38	105.34	94
00:54:39	75.55	95
00:54:40	97.83	95
00:54:41	97.83	95
00:54:42	100.04	95
00:54:43	97.23	94
00:54:44	71.05	94
00:54:45	65.36	95
00:54:46	65.36	95
00:54:47	0	0
00:54:48	44.74	0
00:54:49	43.31	0
00:54:50	79	94
00:54:51	98.78	94
00:54:52	95.7	93
00:54:53	100.28	97

00:54:54	94.71	97
00:54:55	73.67	97
00:54:56	47.73	97
00:54:57	67.21	97

Table 11: Heart Rate and SpO2 readings for the 10-year-old boy

The heart rate of this boy ranged from 59.04-100.04 bpm which is acceptable. The SpO2 values fluctuated from 93-96%.

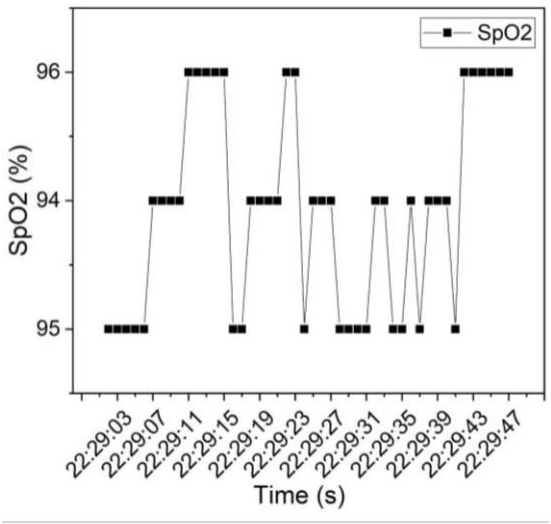

Figure 50: 10-year-old boy SpO2 curve

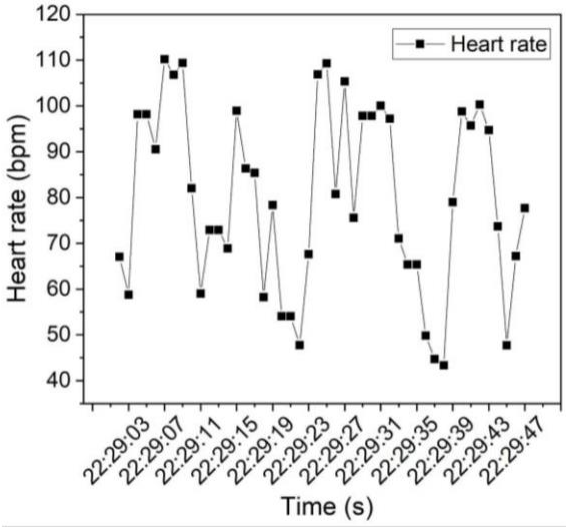

Figure 51: 10-year-old boy heart rate curve

For a child of his age, all of his sensor values were well within the usual range. We may deduce from this that the child is in generally good health.

From all the above analysis, we can conclude that the temperature, heart rate and SpO2 values that we received from the sensors were as predicted, showing that the sensors are functioning well. The GSR values determine the stress level in a person. The higher the GSR value, the higher the stress and this thing was also verified by these 5 different age groups of people. Alert messages are sent to the emergency contact number when a threshold is crossed. Some of them are shown below.

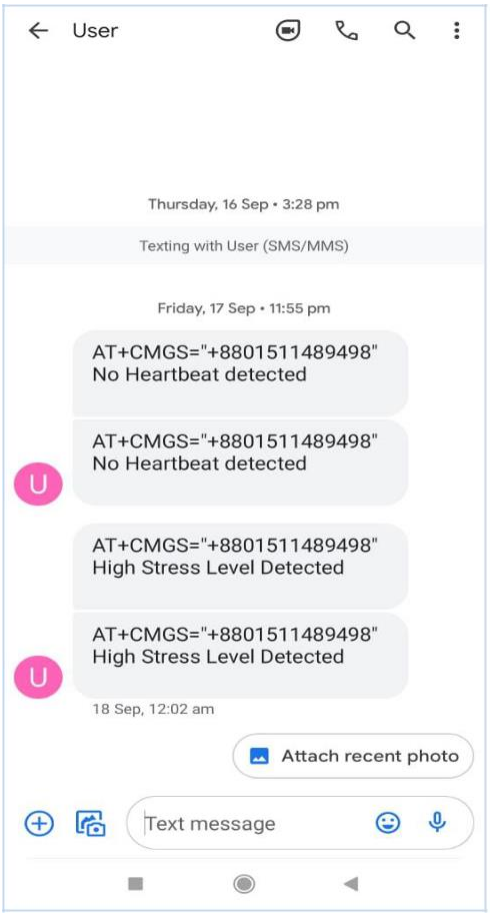

Figure 52: Fetal heartbeat and stress level alert messages

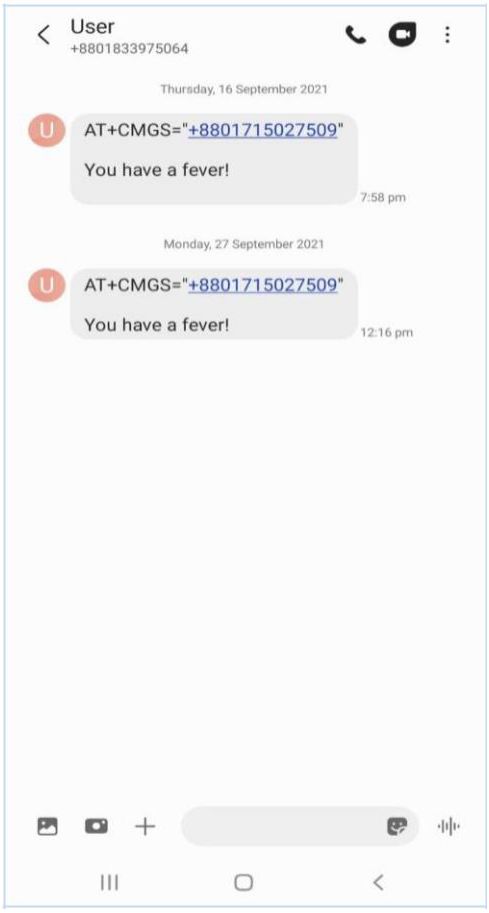

Figure 53: Temperature alert massage

4.2.6 Response characteristics of the sensors

The data we have collected from all the five individuals of different ages are shown in the above tables with graphical representation. But we cannot determine the overall health condition or whether the person is at a risk or not directly from the data or graphs. To be able to do that, we have analyzed the data and studied the graphs to decide on a range for normal, low risk and high risk according to the response of the sensors.

Temperature Data:

The temperature data we have obtained from DHT22 gave us a value a few degrees less than the usual value. Usually, the normal body temperature ranges from 36.1°C-37.2°C while in a pregnant woman it is already elevated around 0.4 degrees. Based on the pattern of the values we have collected; we have come to a conclusion to keep the normal range within 32°C-37°C and any value above or below that range to impose a risk for the person.

Stress Data:

The stress data we have got from GSR varied for the 5 test cases. According to the pattern of the data obtained, we have set a reference value for the GSR sensor, and the threshold ranges are set for normal, low risk and high risk. The normal range is set as 300-400µS. We have seen that an adult person's stress level usually goes above 400µS when she is busy with her daily working schedule. Therefore, we have considered low risk from 400-540µS. Next, we took the stress value from a 55 years old man suffering from Stress Hyperglycemia, which is a stress related problem, and his value came to be higher than 540µS. Thus, we set the range for high risk above 540.

Heart Rate Data:

The heart rate data obtained from MAX30100 were quite accurate compared to the reference values. The normal heart rate of a person varies between 60-100bpm, whereas the heart rate of a pregnant woman goes up to 113bpm in the third trimester. Accordingly, we have set the low-risk range from 101-110bpm and any value beyond this range signifies a high risk for the person.

Blood Oxygen Saturation Data:

The blood oxygen saturation of a normal person varies from 95-100%. So, based on studies and a few analyses, a range of 85-94% implies a low risk for the individual, while any result below this indicates a high risk.

Table 12 shows the level of risk imposed on individuals in respect to a normal range

Sensor	Status	Range
Temperature (°C)	Normal High Risk	32-37 Temp<32 or Temp>37
Stress (µS)	Normal Low Risk High Risk	300-400 400-540 Stress>540
Heart Rate (bpm)	Normal Low Risk High Risk	60-100 101-110 bpm>110
Blood Oxygen Saturation (%)	Normal Low Risk High Risk	95-100 85-94 SpO2<85

Table 12: Health monitoring device threshold value chart

The table below shows the summarization of all the analysis done above along with the overall health status. The recorded values given here are the average of the samples taken from each of the parameters.

	Test Case	Temperature (°C) (T)	Humidity (%) (H)	GSR (µS) (G)	Heart rate (bpm) (HR)	SpO2 (%) (S)	AD8232(mV) (A)	Health Status
1	20-year-old pregnant woman	Recorded: 35.1 Reference: 35	Recorded: 88.07 Reference: 85	Recorded: 323.53 Reference: 350	Recorded: 99 Reference: 95	Recorded: 102.2 Reference: 97	Fetal Heart rate detected: Yes	T-Normal H-Normal G-Normal HR-Normal S-Normal
2	45-year-old woman	Recorded: 33.5 Reference: 35	Recorded: 89.8 Reference: 85	Recorded: 504.4 Reference: 350	Recorded: 92 Reference: 95	Recorded: 96 Reference: 97	Nil	T-Normal H-Normal G-Low Risk HR- Moderate S-Normal
3	23-year-old girl	Recorded: 33.2 Reference: 35	Recorded: 86.7 Reference: 85	Recorded: 465 Reference: 350	Recorded: 94 Reference: 95	Recorded: 97 Reference: 97	Nil	T-Normal H-Normal G-Low Risk HR-Normal S-Normal
4	55-year-old man	Recorded: 32.2 Reference: 35	Recorded: 82.4 Reference: 85	Recorded: 595 Reference: 350	Recorded: 97 Reference: 95	Recorded: 96 Reference: 97	Nil	T-Normal H-Normal G-High Risk HR-Normal S-Normal
5	10-year-old boy	Recorded: 33 Reference: 35	Recorded: 91.7 Reference: 85	Recorded: 362 Reference: 350	Recorded: 94 Reference: 95	Recorded: 98 Reference: 97	Nil	T-Normal HNormal G-Normal HR-Normal S-Normal

Table 13: Health status summarization table

4.3 Device 2 Data Analysis

In this section, we will be examining the fall detection system by analyzing different situations that can make pregnant women collapse. Besides this, we will also be seeing the functionality of the vaccine alert system.

4.3.1 Analyzing Fall Detection System

In order to examine our fall detection system, we have run three test cases:

Case 1: When the woman is in a sitting or standing position

Case 2: When the woman has fallen and is unable to get up or recover from the fall

Case 3: When the woman has fallen and is able to get up or recover from the fall

For case 1, the woman is in a sitting or standing position and at that position we get the accelerometer value (Amp) to be 9 and gyroscope value (angleChange), which denotes the change in orientation, to be 6 degrees. The figure below shows the values for this case.

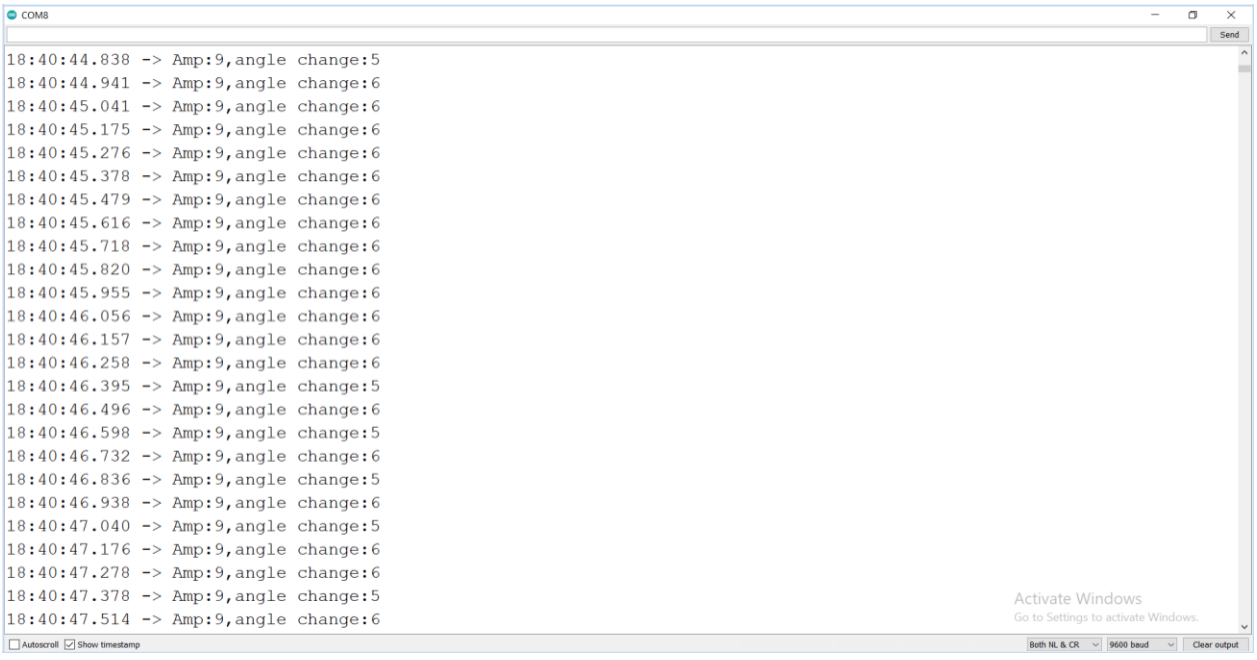

Figure 54: Case 1 readings in Serial monitor

For case 2, when the woman falls, the accelerometer value, Amp, first decreases and then increases. So, when the Amp value decreased below 2, Trigger 1 was activated indicating that the person might have started to fall. After that as the Amp value increased above 11, Trigger 2 was activated and we can also see that the angleChange had suddenly increased, denoting an abrupt change in orientation. While Trigger 2 remained activated, since the value of angleChange was between 30°-120°, Trigger 3 got activated. Next, the program checked whether the angleChange value was between 0°-10° for some time. This is done to ensure whether the person who has fallen has gotten up or not within that specific time. Thus, when the value of angleChange came to be under 10°, it meant that less movement was detected, therefore ensuring that the person had fallen and was not able to get up. Hence the serial monitor showed "Fall Detected" and an alert message was sent to the emergency contact number. The figures below show these results and the sent alert message respectively

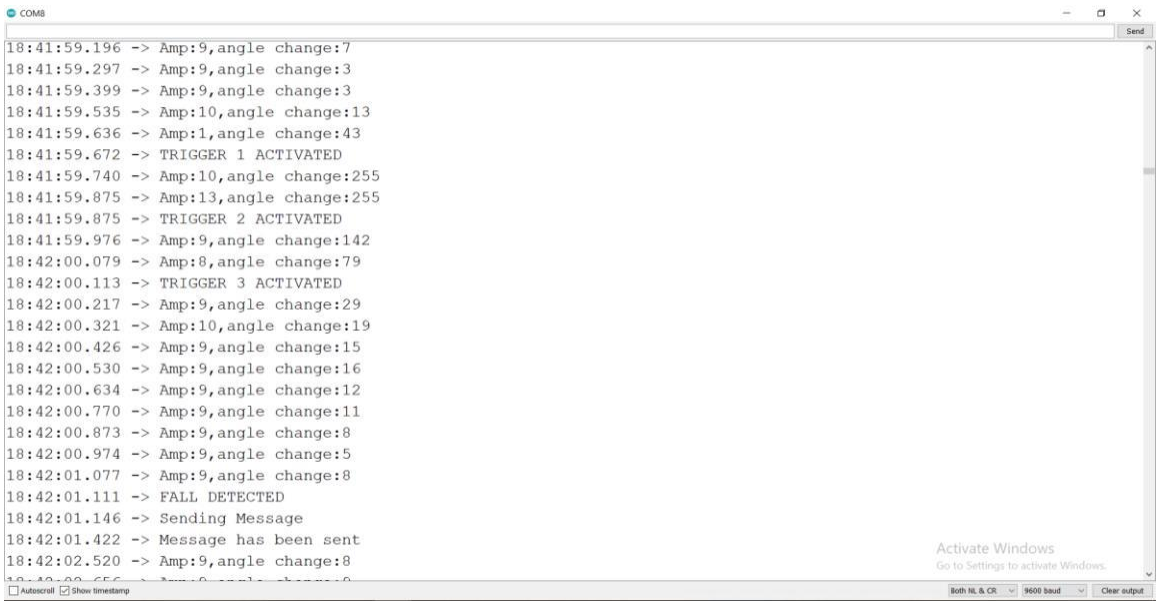

Figure 55: Case 2 readings in Serial monitor

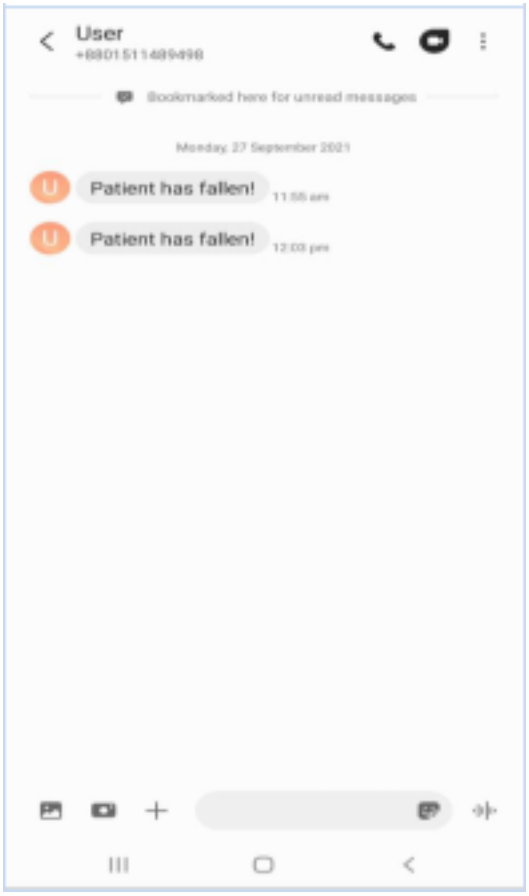

Figure 56: Alert message when Fall Detected

For Case 3, as Trigger 3 got activated, the program checked whether the angleChange value was between 0°-10° for some time. Unlike the previous case, the value of angleChange, in this case, came to be over 10°, meaning that the person who had fallen was able to get up. Hence, Trigger 3 got deactivated which can be seen in the figure below and no alert message was sent to the emergency contact number.

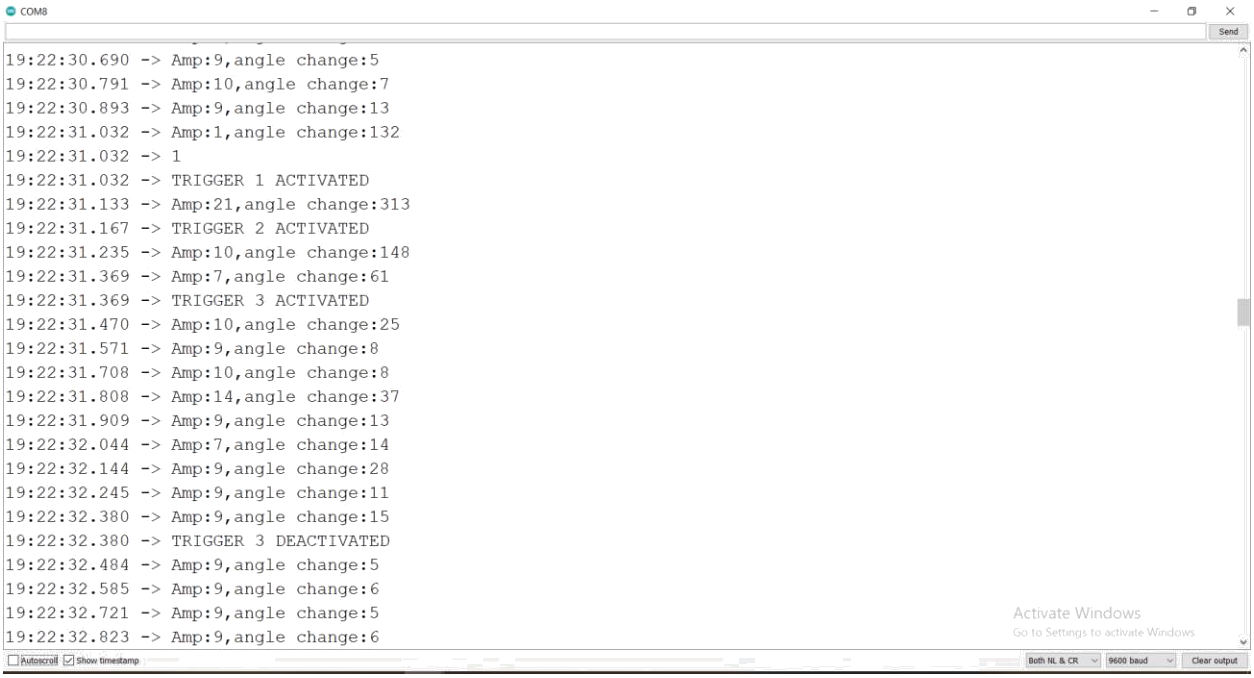

Figure 57: Case 3 readings in Serial monitor

The table below shows the summarization of the fall detection system.

Cases	Situation	Trigger status	Fall Detection status	Notification Status
1	When the woman is in a sitting or standing position	No trigger activated (No risk)	No fall detected	No message sent
2	When the woman has fallen and is unable to get up or recover from the fall	Trigger 1 activated Trigger 2 activated Trigger 3 activated (High Risk)	Fall detected	Message sent
3	When the woman has fallen and is able to get up or recover from the fall	Trigger 1 activated Trigger 2 activated Trigger 3 activated Trigger 3 deactivated (No risk)	No fall detected	No message sent

Table 14: Fall detection system summarization table

4.3.2 Outcome from Vaccination System

The confirmation of the delivered messages, and the 8 messages received by the emergency contact number, of all the necessary vaccine schedules for the newborn child, are shown in the two figures below. We have incorporated the vaccine schedule for a child of up to two years and if we want to show this through messages it will require the continuous running of our device for that entire period of time. Therefore, it was not feasible to show it in real life which is why, for demonstration purposes, the messages were sent in a single day at an interval of 5 seconds. But it can be modified in the future to make it work according to the actual scheduled date and time of the vaccines.

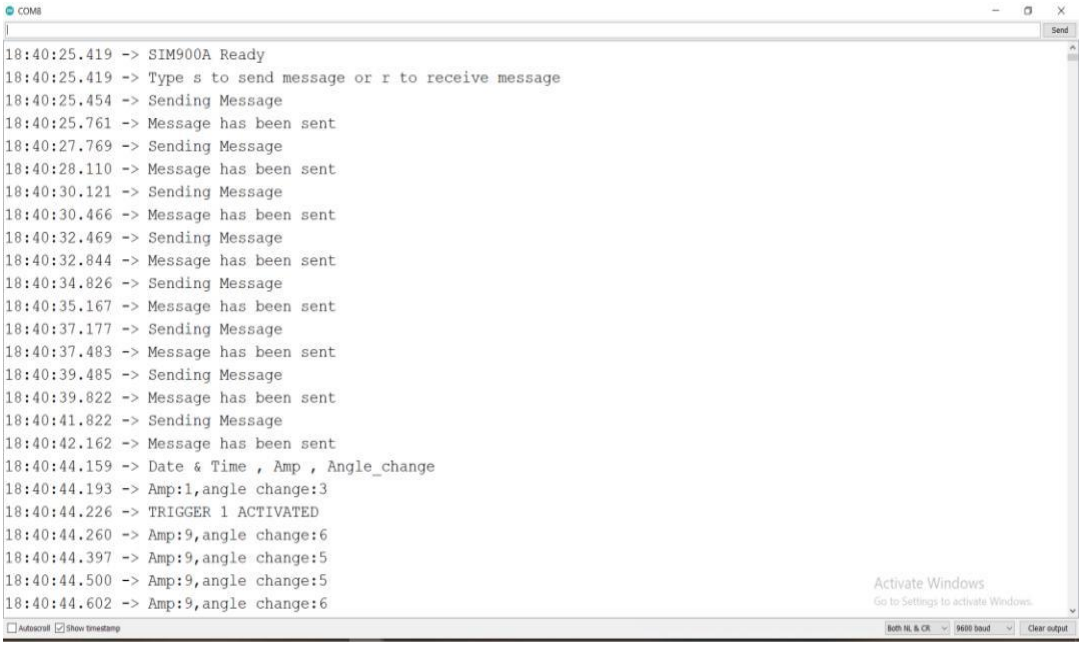

Figure 58: Serial monitor when vaccination reminder messages are sent

Figure 59: Vaccine reminder messages

The table below shows the summarization of the vaccination notification system

	Time of Dose	Recommended Vaccine	Notification Status
1	After Birth	BCG [A], Oral Polio(1^{st} dose) [B]	Notified for A & B
2	6 weeks	Pentavalent(1^{st} dose) [C], Oral Polio(2^{nd} dose) [D]	Notified for C & D
3	10 weeks	Pentavalent(2^{nd} dose) [E], Oral Polio(3^{rd} dose) [F]	Notified for E & F
4	14 weeks	Pentavalent(3^{rd} dose) [G], Oral Polio(4^{th} dose) [H]	Notified for G & H
5	9 months	MR(Measles and Rubella) [I]	Notified for I
6	1 year	Varilrix [J]	Notified for J
7	15 months	Oral Polio(5^{th} dose) [K], Measles [L]	Notified for K & L
8	2 years	Typhoid [M]	Notified for M

Table 15: Vaccination notification system summarization table

The figure below shows a systematic flowchart of the vaccine reminder system giving us a clear idea of its working mechanism.

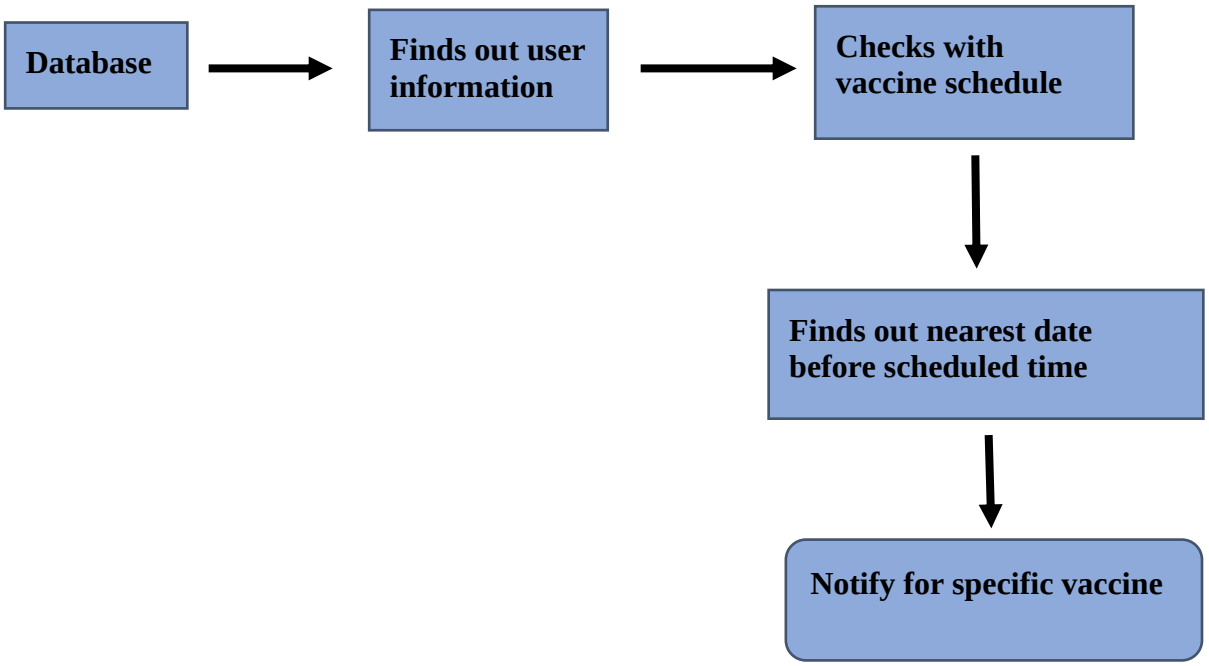

Figure 60: Flowchart of Vaccine reminder system

4.4 Conclusion

In this chapter, we have shown our data collection and analysis from Device 1 and MAX30100 from Device 2, based on which we have stated the health status of a 20-year-old pregnant woman, a 45-year-old-woman, a 23-year-old girl, a 55-year-old man, and a 10-year-old boy. We have also given graphs for each health metric for every person we analyzed so that anyone may get a sense of how that person's health is doing. The analysis of Fall Detection from Device 2 has been explained and displayed for three specific cases. Finally, the outcome of the vaccination system has been discussed.

Chapter 5

To sum up our project, in the last chapter we have discussed a few system limitations and drawbacks, which we could not overcome during our limited time frame, following some possible future work that will make the entire system worthy of manufacturing in the market.

5.1 System Limitations and Drawbacks

This section covers all the limitations and drawbacks of our whole system.

5.1.1 System Limitations

The health parameters of a pregnant woman that we have monitored can vary due to many reasons including her lifestyle, mental health, age, and weight so we cannot come to a proper conclusion only from the data that we have received without knowing these factors. We need to have proper knowledge of any small details that can affect any of these parameters. If we want to take a large amount of data or want to monitor someone for a long period of time, it might not always be possible to keep track of every significant change happening in her life. As a result, the device may not always provide us with an accurate or correct analysis.

We were unable to collect data from several pregnant women because of the Covid-19 situation. Otherwise, we may have collected a wider range of data for our investigation.

5.1.2 Drawbacks

- If the provided emergency contact number is out of reach for any reason, our system will not be able to alert the emergency contact person, so it is critical that the number we provide is always operational.

- The ESP8266 needs to establish a secure and stable connection, so that the MCU and cloud server maintain proper communication. If that network connectivity is lost, the device will not be able to send the data to the server.

- If any of the sensors get damaged during taking readings from the mother, the only option is to replace it with a new or better one. As a result, the system has to be rectified once more with a new sensor.

- It may be difficult to take readings in rural areas with poor internet connectivity, and because our system is built on the Internet of Things, we may not be able to effectively access the health problems of pregnant women in those locations.

5.2 Future Work

Our project still has a lot of space for improvement and possibilities in the future like any other project.

- The sensor data of the pregnant women needs to be sent to the nearby hospital via the Quectel GPS module so that the user can receive further support in the form of quick action. This is much needed because during any unfavorable condition, the mother might collapse, and taking any action by herself would not be possible.

- The whole system can be implemented using machine learning for advanced system management.

- In this project, we have utilized Arduino as a source for collecting data from the sensors but this can be replaced by Raspberry Pi. The additional benefit that we will get from using Raspberry Pi is that it can take data simultaneously from a large number of people e.g. 1000 people at a time since it is faster and has more RAM. For that, we need ML to implement with Raspberry Pi.

- The vaccination time and date for the child, which is to be reminded to the parents through text message, must be executed in real-time. That is the vaccine's exact time and date.

- We can improve the accuracy of the data by using more high-end sensors, i.e. a better version of the sensors compared to that we have used in this system for each measurement parameter.

- To solve the problem of internet connection in rural areas, we can use portable Wi-Fi modules and wireless routers, which will be able to provide Internet connectivity up to a certain range of regions. We will then use as many such Wi-Fi devices to extend the connectivity to an expanded range of areas. Besides this, use of Wi-Fi repeaters will be beneficial in case of poor network regions. The Wi-Fi repeater operates by connecting to the portable Wi-Fi network and rebroadcasting the signal to a larger area.

- The prototype can be converted into any type of wearable device that is easy to carry and can monitor the mother for a longer period of time.

- During emergencies, e.g. when the threshold value is crossed, the message is sent to a certain number but if that is inactive as mentioned in the first drawback, we need to have access to the phone numbers of other family members or close relatives.

5.3 Conclusion

Our IoT based project is grounded on the notation that physical distance between a pregnant woman and health care providers should not be the source of any harm to the pregnant woman's health. As a result, our ongoing project has attempted to fill this void. The mentioned statistics clearly state the mortality rates of women who die due to difficulties during childbirth every day. This IoT built system is reliable and capable of keeping track of the health care factors of a pregnant woman and alerting her and any emergency contact whenever a difficult situation needs to be addressed. The existence of IoT in the system allows for a live stream view of the pregnant woman's data with very little human participation. We used the private channel of the ThingSpeak server to keep data private; nonetheless, this data can be accessed by anyone from anywhere in the world with whom the user shares the Account ID and password. In this constantly developing world with cutting-edge technology, the lives of pregnant women should not be lost because of difficulties that can be monitored, handled, and prevented in a simple yet complex manner.

References

1. “Pregnancy-related complications.” https://www.who.int/teams/integrated-health-services/clinical-services-and-systems/surgical-care/pregnancy-related-complications (accessed Sep. 22, 2021).

2. “Monitoring childbirth in a new era for maternal health.” https://www.who.int/news/item/15-12-2020-monitoring-childbirth-in-a-new-era-for-maternal-health (accessed Sep. 22, 2021).

3. “Antenatal care,” *UNICEF DATA*. https://data.unicef.org/topic/maternal-health/antenatal-care/ (accessed Sep. 23, 2021).

4. “Maternal health,” *UNFPA Bangladesh*, May 23, 2017. https://bangladesh.unfpa.org/en/topics/maternal-health-4 (accessed Sep. 23, 2021).

5. “Bangladesh Maternal mortality ratio, 1960-2020 - knoema.com.” https://knoema.com/atlas/Bangladesh/Maternal-mortality-ratio (accessed Sep. 23, 2021).

6. “Body temperature norms: MedlinePlus Medical Encyclopedia.” https://medlineplus.gov/ency/article/001982.htm (accessed Sep. 22, 2021).

7. “Fever: Symptoms, treatments, types, and causes,” May 05, 2020. https://www.medicalnewstoday.com/articles/168266 (accessed Sep. 22, 2021).

8. R. Ansorge, “High Temperature? Find Out What Causes a Fever,” *WebMD*. https://www.webmd.com/first-aid/fevers-causes-symptoms-treatments (accessed Sep. 22, 2021).

9. M. Angst, “8 ways freezing temperatures can affect your body,” *Insider*. https://www.insider.com/cold-temperatures-affect-body-2019-1 (accessed Sep. 23, 2021).

10. “Is Too Much Humidity Hurting Your Health?,” *Arista*, May 07, 2020. https://aristair.com/blog/is-too-much-humidity-hurting-your-health/ (accessed Sep. 23, 2021).

11. “Stress and pregnancy.” https://www.marchofdimes.org/complications/stress-and-pregnancy.aspx (accessed Sep. 22, 2021).

12. “Tachycardia: Causes, symptoms, diagnosis, and treatments,” Aug. 18, 2020. https://www.medicalnewstoday.com/articles/175241 (accessed Sep. 23, 2021).

13. “Tachycardia - Symptoms and causes,” *Mayo Clinic*. https://www.mayoclinic.org/diseases-conditions/tachycardia/symptoms-causes/syc-20355127 (accessed Sep. 23, 2021).

14. “Articles,” *Cedars-Sinai*. https://www.cedars-sinai.org/health-library/diseases-and-conditions/b/bradycardia.html (accessed Sep. 22, 2021).

15. X. Zhao, X. Zeng, L. Koehl, G. Tartare, J. de Jonckheere, and K. Song, "An IoT-based wearable system using accelerometers and machine learning for fetal movement monitoring," in *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, 2019, pp. 299-304: IEEE.

16. L. Hema and A. Anil, "Pregnant Women Health Monitoring System Using Embedded System," in *IOP Conference Series: Materials Science and Engineering*, 2020, vol. 993, no. 1, p. 012078: IOP Publishing.

17. S. Santhi, A. Gandhi, M. Geetha, and K. J. B. I. J. Nirmala, "Smart Maternal Healthcare Monitoring System using Wireless Sensors Network," vol. 12, no. 12, 2016.

18. G. K. Endo, I. Oluwayomi, V. Alexandra, Y. Athavale, and S. Krishnan, "Technology for continuous long-term monitoring of pregnant women for safe childbirth," in *2017 IEEE Canada International Humanitarian Technology Conference (IHTC)*, 2017, pp. 6-10: IEEE.

19. L. M. Borges, N. Barroca, F. J. Velez, and A. S. Lebres, "Smart-clothing wireless flex sensor belt network for foetal health monitoring," in *2009 3rd International Conference on Pervasive Computing Technologies for Healthcare*, 2009, pp. 1-4: IEEE.

20. R. Hiyama, M. Saito, Y. Nakanishi, Y. Hirose, and S. Arisumi, "BabyBumper: protector/communication wearable device for pregnant women," in *Adjunct Proceedings*

of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers, 2015, pp. 173-176.

21. F. Sarhaddi *et al.*, "Long-Term IoT-Based Maternal Monitoring: System Design and Evaluation," vol. 21, no. 7, p. 2281, 2021.

22. T. G. Troyee, M. K. Raihan, and M. S. Arefin, "Health Monitoring of Expecting Mothers using Multiple Sensor Approach: “Preg Care”," in *2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT)*, 2020, pp. 77-82.

23. N. M. Bahbouh, A. B. Alkhodre, A. A. A. Sen, A. Namoun, and S. S. Albouq, "A Cost Effective IoT-based System for Monitoring Baby Incidents by Deaf Parents," in *2019 International Conference on Advances in the Emerging Computing Technologies (AECT)*, 2020, pp. 1-6.

24. T. M. Kadarina and R. Priambodo, "Monitoring heart rate and SpO2 using Thingsboard IoT platform for mother and child preventive healthcare," *IOP Conference Series: Materials Science and Engineering*, vol. 453, p. 012028, 2018/11/29 2018.

25. P.Poonkuzhlai,R.Aarthi,Yaazhini.V.M, Yuvashri.S, Vidhyalakshmi.G, “Child Monitoring and Safety System Using Wsn and Iot Technology”, *Annals of RSCB*, pp. 10839–10847, Apr. 2021.

26. M. K. Saini and R. K. Saini, “Internet of Things (IoT) Applications and Security Challenges: A Review,” *International Journal of Engineering Research & Technology*, vol. 7, no. 12, Dec. 2019, Accessed: Sep. 23, 2021. [Online]. Available: https://www.ijert.org/research/internet-of-things-iot-applications-and-security-challenges-a-review-IJERTCONV7IS12028.pdf, https://www.ijert.org/internet-of-things-iot-applications-and-security-challenges-a-review

27. “What is IoT (Internet of Things) and How Does it Work?,” *IoT Agenda*. https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT (accessed Sep. 23, 2021).

28. S. Ranger, “What is the IoT? Everything you need to know about the Internet of Things right now,” *ZDNet*. https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/ (accessed Sep. 23, 2021).

29. “Arduino MEGA 2560 Microcontroller Rev 3 - Boxed Original.” http://www.robotpark.com/Arduino-MEGA-2560-En (accessed Sep. 28, 2021).

30. “Which Arduino for Beginners | Choosing an Arduino.” https://startingelectronics.org/articles/arduino/which-arduino-for-beginners/ (accessed Sep. 28, 2021).

31. “DHT22 Digital Temperature Humidity Sensor Module.” https://www.electronics.com.bd/dht22-temperature-humidity-sensor (accessed Sep. 28, 2021).

32. “Zkee Shop MAX30100 Pulse Oximeter Heart Rate Sensor Module Compatible for Arduino- Buy Online in Bangladesh at Desertcart - 48388763.” https://bangladesh.desertcart.com/products/48388763-max30100-pulse-oximeter-heart-rate-sensor-module-compatible-for-arduino?fbclid=IwAR2_WQ-qS91J7PVtKdgckMRXm10SW-EF38VJsvE9LRKsPwgqt0LaqOwKm9Y (accessed Sep. 28, 2021).

33. “MPU6050 GY521 Cheap 6DOF IMU (Accelerometer & Gyro) IMUs | JSumo.com,” *https://www.jsumo.com/*. https://www.jsumo.com/mpu6050-gy521-cheap-6dof-imu-accelerometer-gyro (accessed Sep. 28, 2021).

34. “Kit Wireless Extension Module Sim900a Sim900 V4.0 Board New - AliExpress.” https://www.aliexpress.com/i/32308127622.html (accessed Sep. 28, 2021).]

35. “Grove Gsr Sensor Skin Current Sensor Skin Measuring Sensor - Buy Gsr Sensor,Skin Current Sensor,Skin Measuring Sensor Product on Alibaba.com.” https://www.alibaba.com/product-detail/Grove-GSR-Sensor-Skin-Current-Sensor_60816828804.html (accessed Sep. 28, 2021).

36. “AD8232 ECG Sensor : Pin Configuration, Features and Its Applications,” *ElProCus - Electronic Projects for Engineering Students*, Dec. 19, 2019. https://www.elprocus.com/ad8232-ecg-sensor-working-and-its-applications/ (accessed Sep. 28, 2021).

37. “WiFi Module - ESP8266 (4MB Flash) - WRL-17146 - SparkFun Electronics.” https://www.sparkfun.com/products/17146/ (accessed Sep. 28, 2021).

38. “Incepta Vaccine.” http://inceptavaccine.com/immunization-schedule.php?fbclid=IwAR269tN81i5UQUE1rXLeKw_5sfw7HbAwHHkzRHiYcMad7Qppydm2GcvND54 (accessed Sep. 23, 2021).

39. “Latest EPI Child Vaccination Schedule 2018 Know About Vaccine,” *Pediatric Medicine*, Sep. 05, 2018. https://pedimedicine.com/vaccine-preventable-diseases-and-latest-epi-child-vaccination-schedule-2018/ (accessed Sep. 23, 2021).

40. “Heart Palpitations During Pregnancy: Should I Worry?,” *Healthline*, Aug. 11, 2017. https://www.healthline.com/health/pregnancy/heart-palpitations (accessed Sep. 22, 2021).

41. “Normal Vital Signs in Pregnancy.” http://www.perinatology.com/Reference/Reference%20Ranges/Vital%20Signs.htm (accessed Sep. 22, 2021).

Appendix A

We implemented the codes below for our two devices to perform their specified tasks.

DEVICE 1

```
#include <dht.h>
#include <LiquidCrystal.h>

dht DHT;
#define DHT11_PIN 12
LiquidCrystal lcd(36,35,34,33,32,31);

const int GSR=A2;
int GSRsensorValue=0;
int gsr_average=0;
float high_temp_threshold = 37;
float low_temp_threshold = 31;
float heart_rate=0;
float gsr_threshold = 540;
boolean noheartrate= false;
byte noheartratecount=0;

String AP = "Azharul98";       // AP NAME
String PASS = "reputation"; // AP PASSWORD
String API = "8VLQ0DVB4LYWF0MB"; // Write API KEY
String HOST = "api.thingspeak.com"; String PORT = "80";

int countTrueCommand;
int countTimeCommand;
boolean found = false;

void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.print("Initializing");
  lcd.clear();
  delay(100);
  Serial.println("Date & Time , Temperature , Humidity , GSR , AD8232");
  Serial3.begin(115200);
  sendCommand("AT",5,"OK");
  sendCommand("AT+CWMODE=1",5,"OK");
  sendCommand("AT+CWJAP=\""+ AP +"\",\""+ PASS +"\"",20,"OK");
  AD8232Setup();
  sendMessageSetup();
```

```
}

void loop() {

 printData();
 notify();
 sendtoServer();

}


void AD8232Setup(){
  pinMode(10, INPUT); // Setup for leads off detection LO +
  pinMode(11, INPUT); // Setup for leads off detection LO -

}

void sendMessageSetup(){
  Serial2.begin(9600);
  Serial.println ("SIM900A Ready");
  Serial.println ("Type s to send message or r to receive message");
}

String getTemperatureValue(){
  int chk = DHT.read22(DHT11_PIN);
  float t= DHT.temperature;
  delay(50);
  return String (t);
}

String getHumidityValue(){
  float h= DHT.humidity;
  delay(50);
  return String (h);
}

String getGSRValue(){
  long sum=0;
  for(int i=0;i<10;i++)          //Average the 10 measurements to remove the glitch
      {
      GSRsensorValue=analogRead(GSR);
      sum += GSRsensorValue;
      delay(5);
      }
   gsr_average = sum/10;
   return String(gsr_average);
```

```
}

void printData(){

   Serial.print(",");
   Serial.print(DHT.temperature);
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("Temp:");
   lcd.print(DHT.temperature);
   Serial.print(",");
   Serial.print(DHT.humidity);
   Serial.print(",");
   Serial.print(gsr_average);
   lcd.setCursor(9, 2);
   lcd.print("GSR:");
   lcd.print(gsr_average);
   Serial.print(",");
   if((digitalRead(10) == 1)||(digitalRead(11) ==
     1)){ Serial.print('!');
     lcd.setCursor(0, 2);
     lcd.print("FH:!");

     noheartratecount++;
     Serial.print(", noheartratecount:");
     Serial.println(noheartratecount);
     if (noheartratecount>=5 && noheartrate== false
       ){ noheartrate= true;
       Serial.println("No Heartbeat detected");
       SendMessage3();
       lcd.clear();
       lcd.setCursor(0, 0);
       lcd.print("No Heartbeat");
       lcd.setCursor(0, 2);
       lcd.print("Detected");
       delay(1000);
       noheartratecount=0;
       noheartrate= false;

     }
   }
   else{
     noheartratecount=0;

     float sum=0;
     for (int i = 0; i < 10; i++){
```

```
      float heartrate= analogRead(A0);
      sum=sum+heartrate ;
     }
   float heart_rate= (sum/10);
   Serial.println(heart_rate);

   lcd.setCursor(0, 2);
   lcd.print("FH:");
   lcd.print(heart_rate);

  }

}

void notify(){

  if ((((DHT.temperature) > high_temp_threshold))|| (((DHT.temperature)
< low_temp_threshold))) {
   SendMessage1();
  }

  else if (gsr_average>gsr_threshold){
   SendMessage2();
  }

}

void SendMessage1(){

    Serial.println ("Sending Message");
   //Serial2.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   Serial2.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to
send message
   delay(110);
   Serial2.println("You have a fever!");// Messsage
   content Serial2.println((char)26);// ASCII code of
   CTRL+Z Serial.println ("Message has been sent");
}

void SendMessage2(){

   Serial.println ("Sending Message");
   //Serial2.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
     Serial2.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to send
```

```
message
   delay(110);
   Serial2.println("High Stress Level Detected");// Messsage
   content Serial2.println((char)26);// ASCII code of CTRL+Z
   Serial.println ("Message has been sent");
}

void SendMessage3(){

   Serial.println ("Sending Message");
   //Serial2.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   Serial2.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to
send message
   delay(110);
   Serial2.println("No Heartbeat detected");// Messsage
   content Serial2.println((char)26);// ASCII code of CTRL+Z
   Serial.println ("Message has been sent");
}

void sendtoServer( ){
  String getData = "GET /update?api_key="+ API +"&field1="+ getGSRValue()+ "&field2="+
getTemperatureValue()+ "&field3="+ getHumidityValue();
 sendCommand("AT+CIPMUX=1",5,"OK");
 sendCommand("AT+CIPSTART=0,\"TCP\",\""+ HOST +"\","+ PORT,15,"OK");
 sendCommand("AT+CIPSEND=0," +String(getData.length()+4),4,">");
 Serial3.println(getData);
 countTrueCommand++;
 sendCommand("AT+CIPCLOSE=0",5,"OK");

}

void sendCommand(String command, int maxTime, char readReplay[]) {

  while(countTimeCommand < (maxTime*1))
  {
   Serial3.println( command);//at+cipsend
    if(Serial3.find(readReplay))//ok
    {
      found = true;
      break;
    }

    countTimeCommand++;
  }
```

```
 if(found == true)
 {
  countTrueCommand++;
  countTimeCommand = 0;
 }

 if(found == false)
 {
  countTrueCommand = 0;
  countTimeCommand = 0;
 }

 found = false;
}
```

DEVICE 2

```
#include <Wire.h>
#include<SoftwareSerial.h>
#include <LiquidCrystal.h>
#define vcc2 12
SoftwareSerial sim900(8,9);
LiquidCrystal lcd(2,3,4,5,6,7);
const int MPU_addr=0x68; // I2C address of the MPU-6050
int16_t AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
float ax=0, ay=0, az=0, gx=0, gy=0, gz=0;
boolean fall = false; //stores if a fall has occurred
boolean trigger1=false; //stores if first trigger (lower threshold) has occurred
boolean trigger2=false; //stores if second trigger (upper threshold) has occurred
boolean trigger3=false; //stores if third trigger (orientation change) has occurred
byte trigger1count=0; //stores the counts past since trigger 1 was set true
byte trigger2count=0; //stores the counts past since trigger 2 was set true
byte trigger3count=0; //stores the counts past since trigger 3 was set true
int angleChange=0;
char *myStrings[] =
{"AfterBirth","6weeks","10weeks","14weeks","9months","1year","15months","2years"};

void setup(){
  pinMode (vcc2,OUTPUT); // Setting pin 12 as output
  digitalWrite(vcc2,HIGH); // Setting vcc2 as second voltage source
  Serial.begin(9600);
```

```
  sim900.begin(9600);
  Serial.println ("SIM900A Ready");
  Serial.println ("Type s to send message or r to receive message");
  lcd.begin(16, 2);
  lcd.print("Initializing");
  delay(100);
  lcd.clear();
  for (int i = 0; i < 8; i++) {
    myStrings[i];
       if (myStrings[i]==myStrings[0]){
         SendMessage1();
       }
       else if (myStrings[i]==myStrings[1]){
         SendMessage2();
       }
       else if (myStrings[i]==myStrings[2]){
         SendMessage3();
       }
       else if (myStrings[i]==myStrings[3]){
         SendMessage4();
       }
       else if (myStrings[i]==myStrings[4]){
         SendMessage5();
       }
       else if (myStrings[i]==myStrings[5]){
         SendMessage6();
       }
       else if (myStrings[i]==myStrings[6]){
         SendMessage7();
       }
       else{
          SendMessage8();
       }
       delay(2000);

   }
   Serial.println("Date & Time , Amp , Angle_change");
  Wire.begin();
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B); // PWR_MGMT_1 register
  Wire.write(0);    // set to zero (wakes up the MPU-6050)
  Wire.endTransmission(true);
}

void loop(){
```

```
  mpu_read();
  ax = (AcX-2050)/16384.00;
  ay = (AcY-77)/16384.00;
  az = (AcZ-1947)/16384.00;
  gx = (GyX+270)/131.07;
  gy = (GyY-351)/131.07;
  gz = (GyZ+136)/131.07;
  // calculating Amplitute vactor for 3 axis
  float Raw_Amp = pow(pow(ax,2)+pow(ay,2)+pow(az,2),0.5);
  angleChange = pow(pow(gx,2)+pow(gy,2)+pow(gz,2),0.5);

  int Amp = Raw_Amp * 10;  // Multiplied by 10 because values are between 0 to 1
  //Serial.print(",");
  Serial.print("Amp:");
  Serial.print(Amp);
  Serial.print(",");
  Serial.print("angle change:");
  Serial.println(angleChange);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Amp:");
  lcd.print(Amp);
  lcd.setCursor(0, 4);
  lcd.print("angleChange:");
  lcd.print(angleChange);

  if (Amp<=2 && trigger2==false){ //if system breaks lower
    threshold trigger1=true;
    Serial.println(Amp);
    Serial.println("TRIGGER 1 ACTIVATED");
    }
  if (trigger1==true){
    trigger1count++;
    if (Amp>=11){ //if system breaks upper
      threshold trigger2=true;
      Serial.println("TRIGGER 2 ACTIVATED");
      trigger1=false; trigger1count=0;
      }
  }
  if (trigger2==true){
    trigger2count++;
    angleChange = pow(pow(gx,2)+pow(gy,2)+pow(gz,2),0.5);
    //Serial.println(angleChange);
    if (angleChange>=30 && angleChange<=120){ //if orientation changes between 30-
120 degrees
      trigger3=true; trigger2=false; trigger2count=0;
```

```
      //Serial.println(angleChange);
      Serial.println("TRIGGER 3 ACTIVATED");

        }
     }
   if (trigger3==true){
     trigger3count++;
     if (trigger3count>=10){
       angleChange = pow(pow(gx,2)+pow(gy,2)+pow(gz,2),0.5);
       delay(10);
       //Serial.println(angleChange);
       if ((angleChange>=0) && (angleChange<=10)){ //if orientation change remains
between 0-10 degrees
           fall=true; trigger3=false; trigger3count=0;
           //Serial.println(angleChange);
             }
       else{ //user regained normal orientation
         trigger3=false; trigger3count=0;
         Serial.println("TRIGGER 3 DEACTIVATED");
       }
      }
    }
   if (fall==true){ //in case of a fall detection
     Serial.println("FALL DETECTED");
     lcd.clear();
     lcd.setCursor(0, 0);
     lcd.print("Fall Detected!");
     SendMessage();
     delay(1000);

     fall=false;
     }
   if (trigger2count>=6){ //allow some time for orientation
     change trigger2=false; trigger2count=0;
     Serial.println("TRIGGER 2 DEACTIVATED");
     }
   if (trigger1count>=6){ //allow some time for system to break upper
     threshold trigger1=false; trigger1count=0;
     Serial.println("TRIGGER 1 DEACTIVATED");
     }
     delay(100);
 }

void mpu_read(){

   Wire.beginTransmission(MPU_addr);
```

```
  Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
  Wire.endTransmission(false);
  Wire.requestFrom(MPU_addr,14,true); // request a total of 14 registers
  AcX=Wire.read()<<8|Wire.read(); // 0x3B (ACCEL_XOUT_H) & 0x3C
(ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read(); // 0x3D (ACCEL_YOUT_H) & 0x3E
(ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read(); // 0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp=Wire.read()<<8|Wire.read(); // 0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX=Wire.read()<<8|Wire.read(); // 0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read(); // 0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read(); // 0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
 }

 void SendMessage()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to
send message
   delay(110);
   sim900.println("Patient has fallen!");// Message
   content sim900.println((char)26);// ASCII code of
   CTRL+Z Serial.println ("Message has been sent");
}

void SendMessage1()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to
send message
   delay(110);
   sim900.println("Time for vaccine: BCG, Oral Polio(1st dose)");// Message
   content sim900.println((char)26);// ASCII code of CTRL+Z Serial.println
   ("Message has been sent");
}

void SendMessage2()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode delay(110);
   //Serial.println ("Set SMS Number");
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to send
```

```
message
   delay(110);
   sim900.println("Time for vaccine: Pentavalent(1st dose), Oral Polio(2nd dose)");//
Message content
   sim900.println((char)26);// ASCII code of
   CTRL+Z Serial.println ("Message has been sent");
}

void SendMessage3()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode delay(110);
   //Serial.println ("Set SMS Number");
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to send
message
   delay(110);
   sim900.println("Time for vaccine: Pentavalent(2nd dose), Oral Polio(3rd dose)");//
Message content
   sim900.println((char)26);// ASCII code of
   CTRL+Z Serial.println ("Message has been sent");
}

void SendMessage4()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to
send message
   delay(110);
   sim900.println("Time for vaccine: Pentavalent(3rd dose), Oral Polio(4th dose)");//
Message content
   sim900.println((char)26);// ASCII code of
   CTRL+Z Serial.println ("Message has been sent");
}

void SendMessage5()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to
send message
   delay(110);
   sim900.println("Time for vaccine: MR(Measles and Rubella)");// Message
   content sim900.println((char)26);// ASCII code of CTRL+Z
```

```
   Serial.println ("Message has been sent");
}

void SendMessage6()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to
send message
   delay(110);
   sim900.println("Time for vaccine: Verilix");// Message
   content sim900.println((char)26);// ASCII code of CTRL+Z
   Serial.println ("Message has been sent");
}

void SendMessage7()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to
send message
   delay(110);
   sim900.println("Time for vaccine: Oral Polio(5th dose), Measles");// Message
   content sim900.println((char)26);// ASCII code of CTRL+Z Serial.println ("Message
   has been sent");
}

void SendMessage8()
{
   Serial.println ("Sending Message");
   sim900.println("AT+CMGF=1"); //Sets the GSM Module in Text Mode
   delay(110);
   sim900.println("AT+CMGS=\"+8801511489498\"\r"); //Mobile phone number to
send message
   delay(110);
   sim900.println("Time for vaccine: Typhoid");// Message
   content sim900.println((char)26);// ASCII code of CTRL+Z
   Serial.println ("Message has been sent");
}
```

Code for MAX30100

```
#include <Wire.h>
#include "MAX30100_PulseOximeter.h"
#include <LiquidCrystal.h>
#define REPORTING_PERIOD_MS     1000
LiquidCrystal lcd(36,35,34,33,32,31);
PulseOximeter pox;
uint32_t tsLastReport = 0;
void onBeatDetected()
{
    Serial.println("Beat!");
}
void setup()
{
    Serial.begin(9600);
    lcd.begin(16, 2);
    lcd.print("Initializing");
    lcd.clear();
    delay(100);
    Serial.print("Initializing pulse oximeter..");
    Serial.println("HeartRate , SpO2");
    if (!pox.begin()) {
        Serial.println("FAILED");
        for(;;);
```

```
    } else {
        Serial.println("SUCCESS");
    }
    pox.setOnBeatDetectedCallback(onBeatDetected);
}
void loop()
{
    pox.update();
    if (millis() - tsLastReport > REPORTING_PERIOD_MS)
        { Serial.print(",");
        Serial.print(pox.getHeartRate()); // printing heart rate value in serial monitor
        Serial.print(",");
        Serial.print(pox.getSpO2()); // printing SpO2 value in serial monitor
        Serial.println("%");
        tsLastReport = millis(); // readings taken at 1 second interval
    }
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("HR:");
    lcd.print(pox.getHeartRate());
    lcd.setCursor(0, 2);
    lcd.print("SpO2:");
    lcd.print(pox.getSpO2());
    }
```