IoT BASED HEALTH MONITORING SYSTEM

LIM CHEE YUAN

A project report submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering (Honours) Electrical and Electronic Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

May 2019

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	[illegible]
Name	:	LIM CHEE YUAN
ID No.	:	1501917
Date	:	23/4/2020

APPROVAL FOR SUBMISSION

I certify that this project report entitled **“IoT BASED HEALTH MONITORING SYSTEM”** was prepared by **LIM CHEE YUAN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Electrical and Electronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Sue Yuen Ch
Date	:	23/11/2020

Signature	:	
Co-Supervisor	:	
Date	:	

ABSTRACT

Health is wealth. Wealth and happiness are earned by having a healthy mind and body. However, people nowadays do not have much free time to keep track of their health status. Thus, a health monitoring system that automatically tracks and alarm the users about their health status is needed. Rapid improvement of the internet and technology, such as the Internet of Things allows the health monitoring system to be improved. The internet of things allows communication between machines and programmed actions to be triggered automatically, which makes the system to be more efficient. The traditional health monitoring system requires regular visitation of patients to doctors to check their health status. However, with the implementation of the internet of things in the health monitoring system, the health monitoring processes can be automated and helps the patient to save their precious time. Besides, the cloud that revolutionized data changing aids in the efforts of making a better and more reliable health monitoring system. The health data can be stored and visualized in real-time. In this project, a NodeMCU is used as a gateway to collect the health data of the user, and a Raspberry Pi 3 Model B+ broker is used as the central processing unit that processes all the received data. The broker receives health data from the gateway and process the data. The system is capable of tracking the location of the user by using the Google geolocation service. The health data and location of the user are visualized in the Thingsboard visualization platform in real-time. Several experiments and tests, such as accuracy test and error analysis were conducted on the proposed system, and encouraging results were obtained.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Revolution and rapid improvement of the internet, technology such as the Internet of Things has emerged and is snowballing. Internet of Things with cloud computing and edge computing realizes a new and more efficient way of data sharing and transmitting. The Internet of things will remodel the healthcare sector and improves the health and wellbeing of humanity (Rahmani et al., 2018). The traditional healthcare system requires patients to visit the clinic or hospital for medical checkups which is time-consuming and inefficient. The Internet of Things is capable of realizing a real-time health monitoring system that involves sensors to measure heart rate and body temperature of patients and visualize the data in real-time. By such, people can have better control of their health condition. Instead of relying on infrequent visits to clinics or hospitals for various tests, people can access their health data through the internet and start to track their health conditions. The Internet of Things that realizes the connection between devices (Tao et al., 2014) allows activities such as sending an alert email and messages during an emergency to be possible by making use of open source services such as google assistance and IFTTT. Besides, the location of the user can be tracked by using geolocation.

1.2 Problem Statement

In the traditional healthcare system, people are required to visit clinics or medical centres regularly for medical checkups, which is less effective and time-consuming. The high medical cost and long waiting will discourage people from performing medical checkups regularly. A health monitoring system that collects and monitors the health status of the user in real-time will benefits the people by saving their money and time of visiting clinics and medical centres unless there is a need for it.

Besides, the security of the health system is vital to safeguard the privacy of the user. People may avoid healthcare in sensitive areas due to health information privacy concerns (Bansal and Gefen, 2010). Smart wearable gadgets such as Apple Watch and Samsung Galaxy Watch are storing the collected health data in the cloud. Cloud storage allows users to enjoy high-quality services without any burden of storage maintenance (Wang et al., 2011). However, cloud users are more vulnerable to issues such as theft, confidentiality, and information leaked to the third party compared to local storage users (Romero, 2012). Storing confidential health information in the health system itself will help reduce the chance of information leakage, as the health information will only be accessible by authorized users which improves the security and privacy of the system.

The location tracking function plays a vital role in the health monitoring system as it allows people to track the whereabouts of the user. Besides, the coordinate of the user should be recorded as it allows people to trace the whereabouts of the user. The function will come in handy when there is a need to track down a person, such as the COVID-19 outbreak. The whereabouts of the patients are useful to trace the source of the disease. Besides, the information will aid in the prevention and evacuation works that will prevent or reduce the spreading of the disease.

Aim and Objective

The project aims to develop a wearable health monitoring system that tracks the user's location, monitors the user's heart rate and body temperature, and visualizes the data in real-time. Authorized users will have access to the collected data stored in the database to keep track of the patient's health data. The system is capable of sending alert notifications to phones and emails when the person wearing the health system is having abnormal body temperature or sudden changes in heart rate. Thus, the objectives of the project are to propose an Internet of Things (IoT) based health monitoring system that collects and monitors the user's body temperature and heart rate in real-time. To propose a system that visualizes and stores the health data in the database while tracking the location of the user by using geolocation.

1.3 Scope and Limitations

The project focuses on the development of an IoT based health monitoring system that involves both hardware and software. The prototype system consists of sensors and a data processing broker. The prototype allows authorized users to monitor the health data and the location of the health system user through the internet. Besides, the authorized user can easily trace back to the previous health data by accessing to the database that stores all the user's health data.

Since the focus of this project is on the implementation and involvement of IoT in the health monitoring system and due to budget limitation, the accuracy of the sensors used in this system will not be taken into consideration as the sensors used in this system are not medically verified and are not suitable to be used for any serious medical analysis by any means.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter provides the reviews of different types of healthcare system related to this project. The reviews include the smart functions and methods used by researchers in their projects.

2.2 Cloud Computing

Cloud computing delivers computing services such as servers, storage, analytics and, intelligence over the internet or cloud (JoSEP et al., 2010) to provide a faster and flexible data exchange process, which helps to reduce the operating cost and increase the efficiency of the infrastructure. Cloud computing realizes a situation where the infrastructures, data or program that customarily installed in desktop PC and server rooms can now be installed in a cloud (Hayes, 2008). Users can perform different tasks by using services provided by the data centres of the cloud through the internet and access the virtualized resources (hardware and services) provided by the cloud anytime and anywhere as long as there is active internet connectivity.

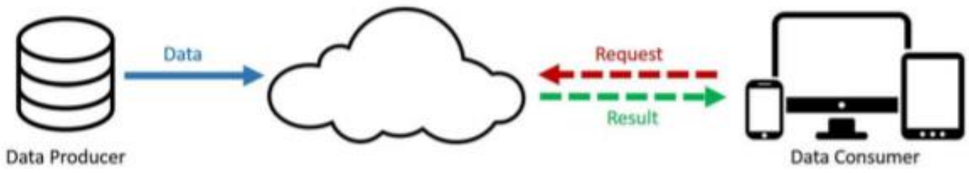


Fig.2.1: Cloud Computing Paradigm (Shi et al., 2016)

Fig.2.1 shows the paradigm of cloud computing. The data consumers are the end devices that send a request for data consuming to the cloud, and the cloud will respond by sending the raw data from data producer to the consumer. Cloud computing services are categorized as software as a service (SaaS), infrastructure as a service (IaaS), and Platform as a service (PaaS) based on the type of capability provided by the clouds.

2.2.1 Software as a Service (SaaS)

SaaS is a type of cloud computing that distributes an application to a wide variety of users through the browser (Knorr and Gruman, 2008), which serves as an alternative to the locally hosted application. SaaS provides access to the software or application through the internet, where license and installation of the software are not required. Customers who pay for the usage fee will have secured access to the SaaS application via the internet. Hosted software is software that "owned" by the company itself, the software can be obtained through licensing, and the software needs to be downloaded and installed in the hosting centre. SaaS is reliable and cost-saving since no licensing and installation of software required. Users could access the software through the internet anytime and anywhere.

2.2.2 Infrastructure as a Service (IaaS)

IaaS is a type of cloud computing that provides infrastructures such as processing, storage, networks, and other fundamental computing resources (Mell and Grance, 2011) in a virtual environment. IaaS can split, assign, and resize these resources to build a system as demanded by users through virtualization (Vaquero et al., 2008). Arbitrary software such as the operating system and infrastructures can be deployed and run by the users.

2.2.3 Platform as a Service (PaaS)

PaaS is a type of cloud computing that provides development environments as a service (Knorr and Gruman, 2008). The users can build their applications by using programming languages, libraries, and tools supported by the provider and run the applications on the infrastructure provided by the PaaS service provider. Multiple authorized users can access the applications from the provider's server through the internet, where the deployed applications and configuration settings for the application-hosting environment are controlled by the developer (Mell and Grance, 2011).

2.3 Edge Computing

Edge computing is a technology that combines cloud computing, grid computing, and IoT, where it acts as an additional layer between end devices and cloud. Edge computing relocates the computational power closer to the end devices (Gezer et al., 2018), where it improves the quality of service (QoS) of applications and reduces the latency of the tasks (Chen et al., 2018). Edge computing aims to make use of computational resources at the edge of the network that keeps increasing and brings data processing closer to where data are generated (Rausch et al., 2018).

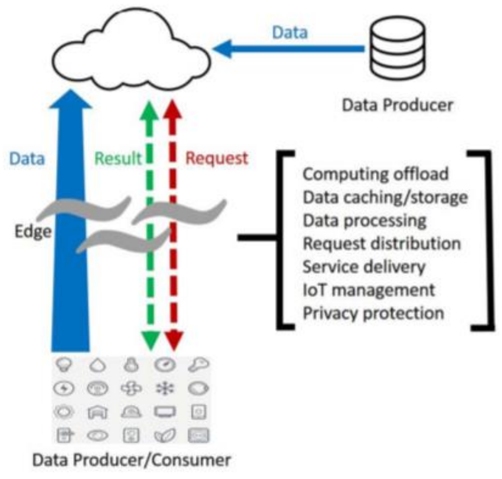


Fig.2.2: Edge Computing Paradigm (Shi et al., 2016)

Fig.2.2 shows the paradigm of edge computing where the edge can act as a data producer and data consumer. The edge can request data from the cloud and perform the computing tasks of the cloud, such as data storing and data caching. With edge computing, the tasks' computing processes can be offloaded to the edge which reduces the response time and increases the efficiency of the system.

2.3.1 Edge Computing and Cloud Computing

Edge computing process data at the edge while the cloud computing process the data in the cloud. The reason edge computing is introduced to the community is to aid the inefficiency of cloud computing in data processing (Shi et al., 2016). Cloud computing is an efficient way of data processing, where it carries out all the computing tasks in the cloud. Cloud computing has superior computing power compared to edge computing. However, when a large number of end devices that generate a large quantity of data are connecting to the cloud, the bandwidth for data transmission between the cloud and end devices will be insufficient and leads to a bottleneck situation. The cloud will need a longer time to process a large amount of data which increases the response time. Edge computing provides private and secured services to the users since the data collected are processed at the edge without sending it to the cloud, which provides users with better privacy protection. Cloud computing reduces the costs of computation by saving the cost of purchasing hardware and provide flexibility, but long-distance between the cloud and end devices will reduce the QoS (Gezer et al., 2018) and increase the latency. The edge that situated between cloud and end devices is closer to the end devices, which can reduce the latency and keep the QoS as high as possible. In cloud computing, the cloud communicates directly with the end devices, while the end devices communicate with intermediate edge components in edge computing as shown in Fig.2.3.

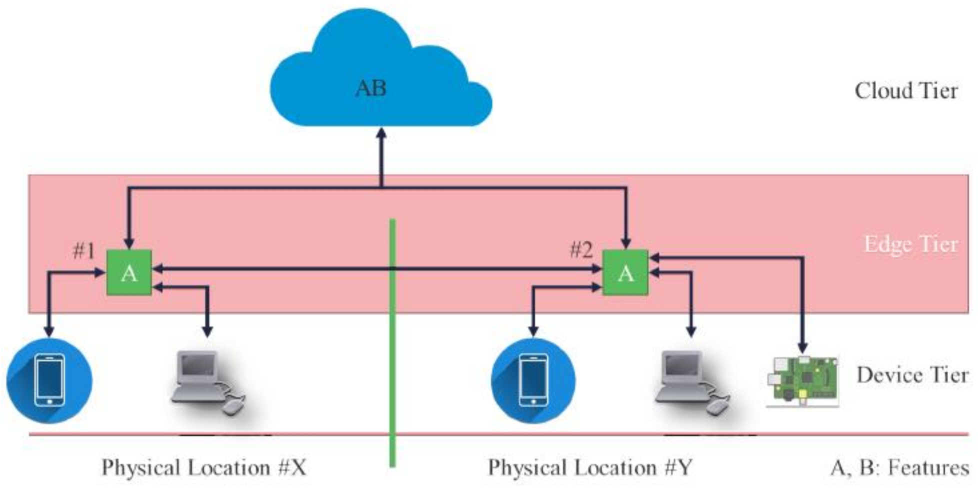
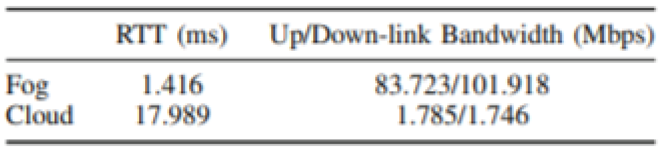


Fig.2.3: Simplified topology of Edge Computing network (Gezer et al., 2018)

2.3.2 Proof of concept that closer distance reduces latency

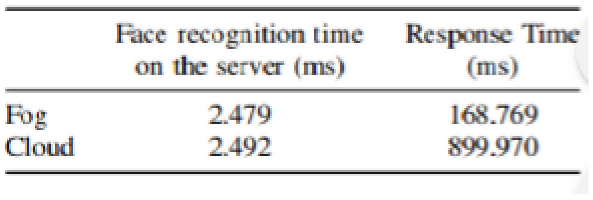
In edge computing, computation and end devices are placed closer to each other. Yi, Hao et al. built a platform to perform a face recognition application. The response time for the application to recognize the face photo from 1521 photos was reduced from 900ms to 169ms by relocating computational to the edge from the cloud. In the project, Yi and Hao compared the latency and bandwidth of fog and cloud and the results obtained are as shown in Fig.2.4.



	RTT (ms)	Up/Down-link Bandwidth (Mbps)
Fog	1.416	83.723/101.918
Cloud	17.989	1.785/1.746

Fig.2.4: Latency and bandwidth comparison (Yi et al., 2015)

A face recognition application was developed to test the performance of cloud and fog. The application installed in a smartphone captures a face photo and transmits the photo to a server located in a fog or cloud. The face photo was matched with 1521 face photos stored in a local database by the remote server. The same task was run on Amazon EC2 cloud and fog. The results obtained are as shown in Fig.2.5.



	Face recognition time on the server (ms)	Response Time (ms)
Fog	2.479	168.769
Cloud	2.492	899.970

Fig.2.5: Fog-based face recognition performance (Yi et al., 2015)

Response time includes the duration from the time the smartphone uploads the face photo until the time it receives the result from the server. The time taken to recognize the face is almost. However, the response time taken by the cloud is 731.201ms longer than the fog due to limited network bandwidth in the cloud.

2.4 Cloud computing on the healthcare system

Rolim et al. (2010) proposed a healthcare system that utilizes cloud computing to collect the patients' data in a healthcare institution. The system automates the process of collecting, distributing, and processing the patient's data which are done manually in the traditional healthcare system. Rolim et al. (2010) suggest that the current manual note-taking approach in the healthcare system as shown in Fig.2.6 is inefficient where medical staff need to collect patient's data and record the data manually. The recorded data are then typed into the data entry terminal that sends the data to the cloud for data processing and storing. The authorized user in the healthcare institution can access the data stored in the database server.

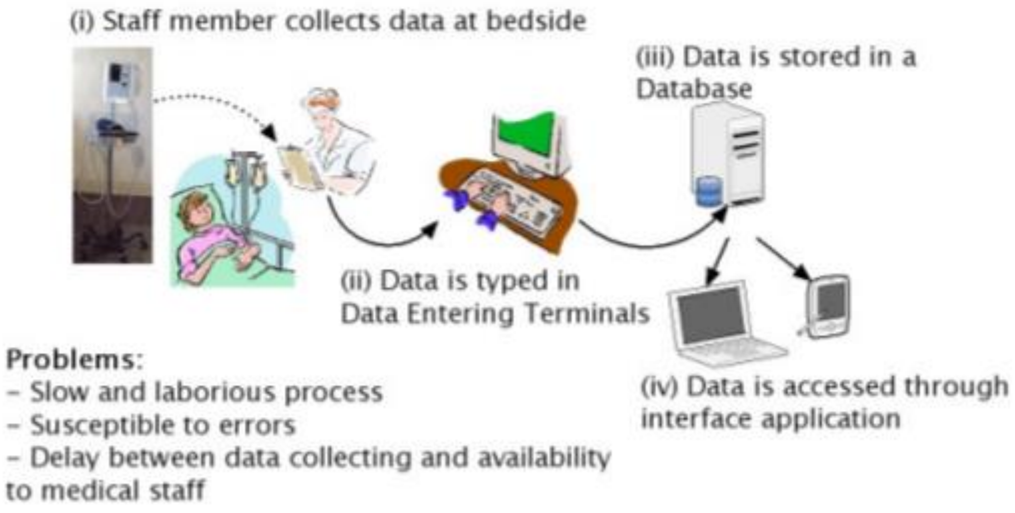

(i) Staff member collects data at bedside
(iii) Data is stored in a Database
(ii) Data is typed in Data Entering Terminals
(iv) Data is accessed through interface application
Problems:
- Slow and laborious process
- Susceptible to errors
- Delay between data collecting and availability to medical staff

Fig.2.6: The typical smart healthcare system's paradigm (Rolim et al., 2010)

The system proposed by Rolim et al. (2010) is capable of collecting data by using mobile sensors connected to medical devices. The collected data and information are delivered to the medical centre's cloud server where the data will be processed, stored, and distributed as shown in Fig.2.7.

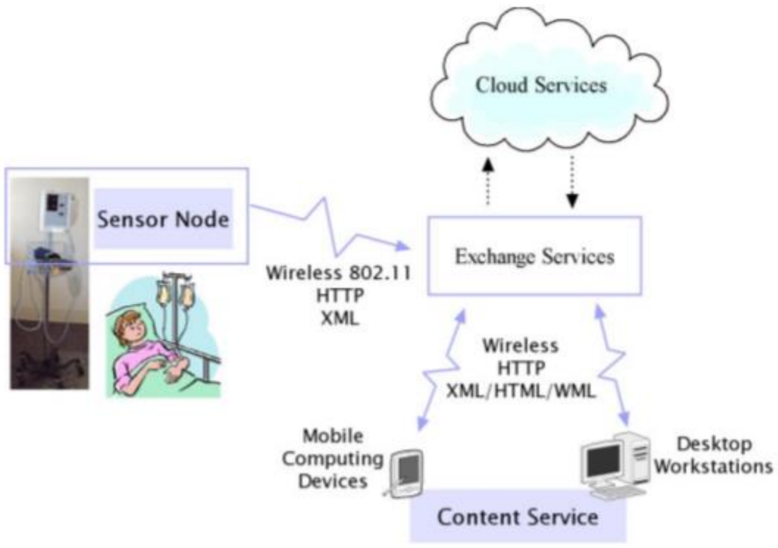


Fig.2.7: Proposed healthcare system's paradigm (Rolim et al., 2010)

In the system, sensor nodes with software installed are placed close to the patients to collect, encode, and transmit data. The data are transmitted via wireless communication channels and stored in the cloud. The system involves sensors to collect data and uses a broker to direct the received data to appropriate storage that hosted on the cloud. Medical staff can retrieve data from cloud service by interacting with the cloud's applications through content service.

Another similarly designed healthcare system that utilizes IoT and cloud computing was proposed by Doukas, C et al. (2012). It is a wearable healthcare system with mobile sensors that capable of collecting and manages sensor data such as heart rate, ECG, and body temperature of users. The data collected can be sent to a smartphone or the cloud infrastructure. The system involves a gateway that capable of communicating with the internet such as a smartphone or Wi-Fi enabled microcontroller. The gateway is used to collect data from the mobile sensors, sends them to the cloud via the internet, and retrieves information from the internet. Cloud will communicate with the gateway with API key provided by the cloud platform and performs data processing, alert management, and billing processes. The system provides real-time data visualization by using a web-based managing application hosted by the cloud which plots the patient's information such as location and activity status in a graph. Fig.2.8 shows the proposed system's architecture.

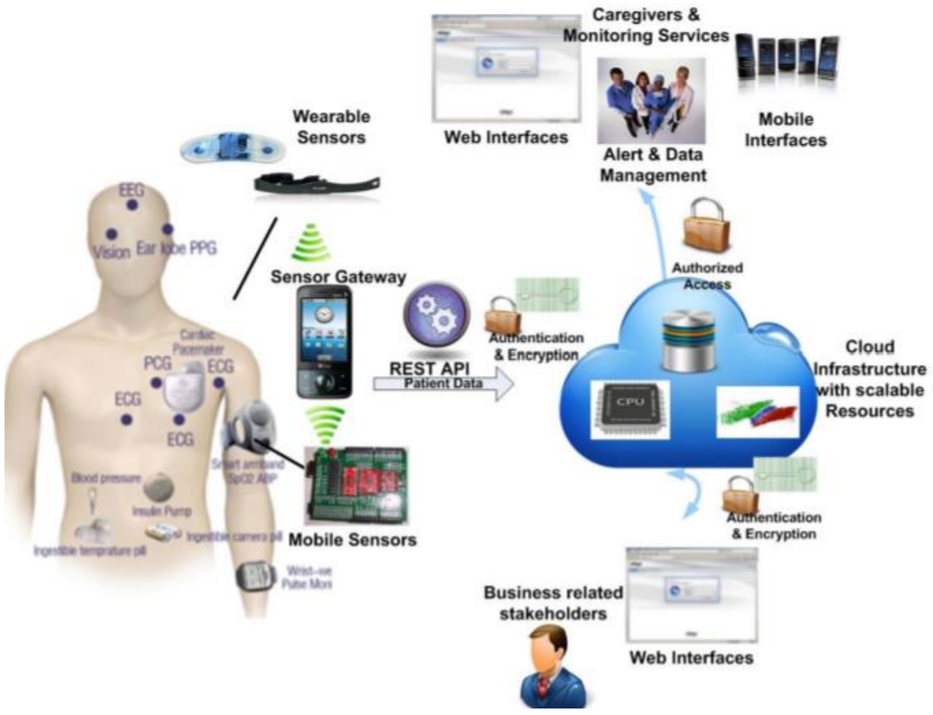


Fig.2.8: The proposed healthcare system architecture (Doukas and Maglogiannis, 2012)

All the data in the healthcare system are secured. Authentication and data encryption are applied where sensors are authenticated by unique id while data are encrypted using symmetric encryption techniques. The system is scalable thanks to the on-demand cloud infrastructure that provides resources based on utilization and demand where the addition of users and sensors will not affect the functionality of the system. In the prototype, microcontrollers and sensors are sewed in a sock where sensors are used to measure user's data while an Android-based smartphone is used as a gateway that sends the data to the cloud as shown in Fig.2.9. A textile version of Arduino, Lilypad and Polar HearRate Module are used. Lilypad collects data from sensor modules and transmits them to an Android-based smartphone via Bluetooth module. The data will then be sent to the cloud by the Android smartphone through an appropriate application developed by Doukas, C et al. (2010) for data analysis.

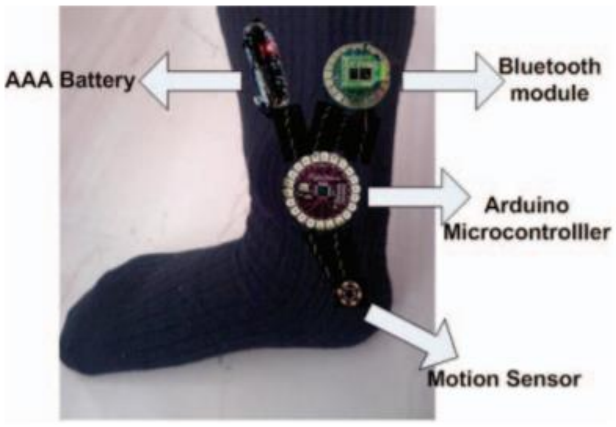


Fig.2.9: The prototype of the cloud-based healthcare system, “CloudSensorSock” (Doukas and Maglogiannis, 2012)

2.5 Edge computing-based system

Edge computing that brings data processing closer to end devices is one way to reduce latencies in the healthcare system. Rausch et al. (2018) proposed a system called EMMA that utilizes MQTT middleware that acts as an intermediate layer between cloud and end devices for edge computing application that improves the QoS of the system. Rausch mentions that many modern IoT systems that utilize cloud-based middleware with pub/sub-model are routing all messages to the cloud, which is impractical in a particular scenario such as the healthcare system due to the latencies. Thus, brokers that are closer to the end devices can be deployed in the edge layer to reduce the latencies and improve the QoS. EMMA is a system that utilizes MQTT protocol in data transmission and is capable of relocating its MQTT clients to a nearby broker based on the situation of clients and resources of brokers to optimize QoS of the system. Proper distribution of edge resources in the system is required since the MQTT clients may leave or enter the system unexpectedly.

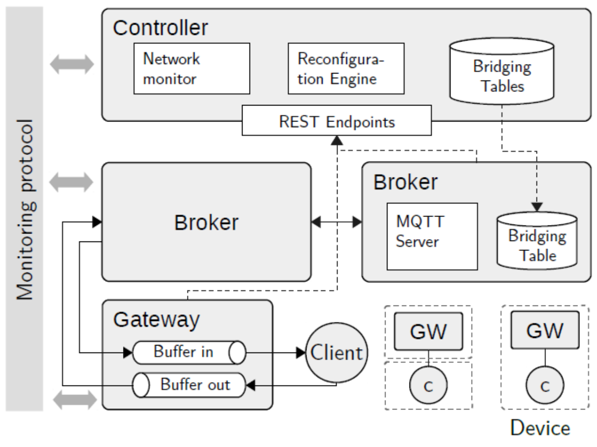


Fig.2.10: Overview of EMMA architecture (Rausch et al., 2018)

EMMA consists of four main components: gateways, broker, controller, and network monitoring protocol as shown in Fig.8. In the system, clients are connected to the gateways which act as a middleman between clients and brokers. It reconfigures the connections between clients and brokers, which allows the clients to move around freely without having to worry about disconnecting from the system. Brokers implement MQTT server protocol that manages sub/pub tasks and acts as topics' bridges that transmit information to other brokers that have the same topics subscription. The controller monitors and distributes edge resources between brokers, and reconfigures gateways and brokers based on network latency. If the latency is high, the controller will direct the gateways to disconnect from the current broker and reconnect to a broker with more resources to improve the QoS of the system as shown in Fig.2.11.

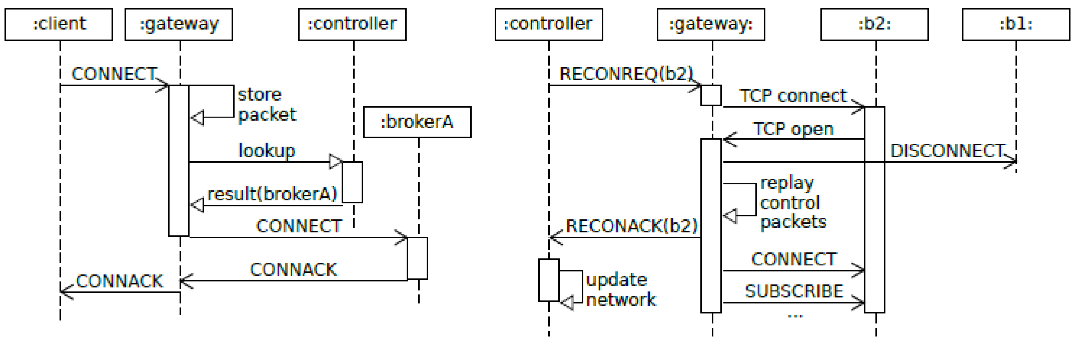


Fig.2.11: Connect and reconnect procedure in EMMA (Rausch et al., 2018)

EMMA architecture realizes a balanced load between brokers, which fits perfectly in the healthcare system where latency is a critically undesired factor. The EMMA architecture is capable of reducing the latency and improve the QoS of the system by effectively separating the load between brokers in the system. By such, paramedics will have access to more accurate data and respond better during an emergency.

On another similar approach, Chen, Li et al. (2018) proposed an edge cognitive computing (ECC) based smart healthcare system. The proposed system is capable of monitoring and analyzing the physical health of users via a cognitive computing network. The system tends to improve the current smart healthcare system by allocating the edge resources in the system according to the health status and risk level of the users, which provides users a better user experience while increasing the survival rates of the user during an emergency. According to Chen, Li et al. (2018), the current smart healthcare system can be divided into three layers, the data collected in the collection layer are sent to the analysis layer through the internet via the gateway or base station in the transmission layer. The health data will be stored and analyzed in the cloud by using machine learning and data mining algorithms. The results that sent back to the system will be analyzed and generate the corresponding medical actions

Chen, Li et al. (2018) mentions that the current smart healthcare system fails to discover the true messages and values that are hiding in a large amount of data because of the inefficiency of traditional machine learning and data mining methods. Besides, the utilization of cloud computing to perform data processing increases the latency, which leads to inaccurate medical analysis during an emergency. Network resources that lack flexibility cause a waste of resources. Thus, cognitive computing can be used in the modern healthcare system. Cognitive computing with smart technologies, such as machine learning, artificial intelligence, and natural language processing (Hou et al., 2016) can determine the relationship and behaviour of the disease from the data. With such technology, medical professionals can learn the disease better and find the cure of the disease faster. Edge computing deploys data processing closer to end devices, which improves the QoS, reduces latency, and improves the reliability of the healthcare system. The proposed healthcare system involves a cognitive data engine and cognitive resource engine which reacts differently during a normal situation and emergency, as shown in Fig.2.12.

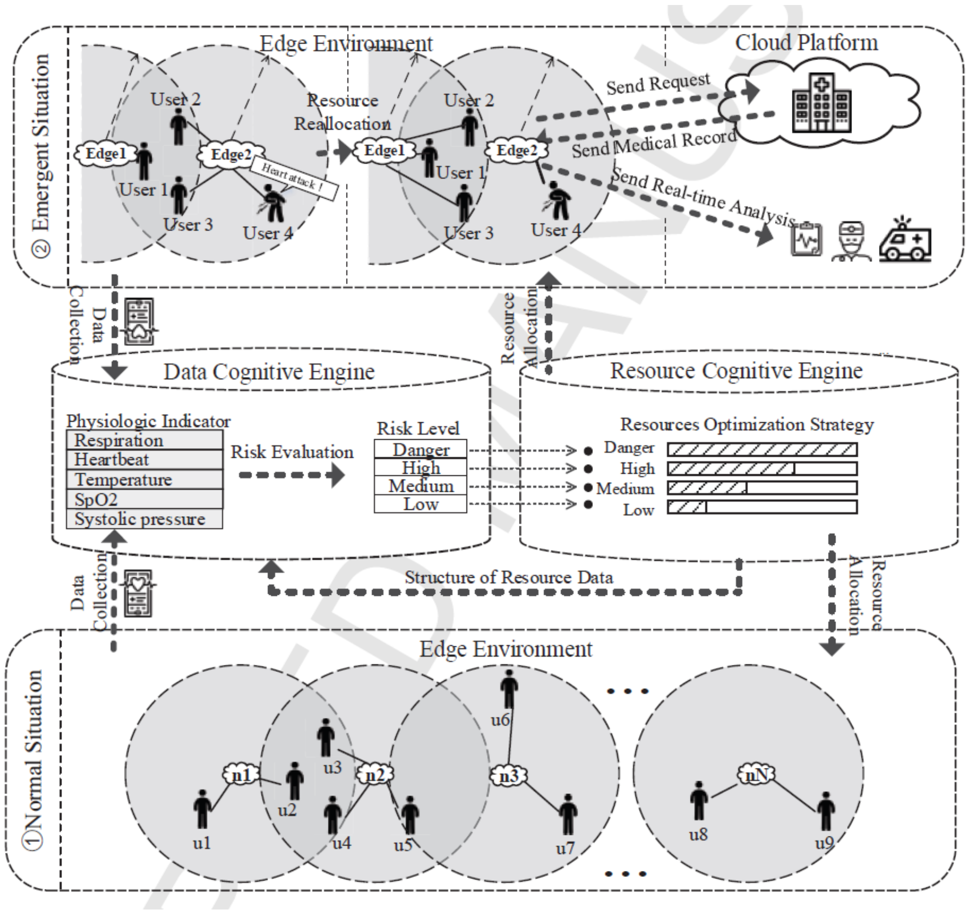


Fig.2.12: ECC-based smart healthcare system (Chen et al., 2018)

The proposed healthcare system is capable of improving itself through its data cognitive engine, where the system learns from resources such as computing and communication resources. Resource cognitive engines will collect and integrate the resources and send them to the cognitive data engine for data analysis and processing via machine learning and deep learning in real-time. The results obtained from data analysis will be sent back to the cognitive resource engine, which used as a guidance in resource allocation and optimization. Besides, the system is capable of relocating the resources according to the priority. Users are categorized based on the risk level faced by users, and the system will automatically relocate the cloud edge resources to serve the user with the highest priority during an emergency.

The system is divided into three major layers, which are the user layer, the cognitive edge layer, and the cloud platform layer. The user layer consists of users, smart clothing, and smartphone, where smart clothing is used to collect the health data of the users such as ECG, body temperature, and blood oxygen saturation in real-time while the smartphone is used to receive health analysis results from edge computing node. On the cognitive edge side, the computing nodes will process and analyze the users' data and distributes the edge resources based on the users' health condition. An alarm will be sent to the hospital by the computing nodes so that medical staff can carry out accurate medical diagnoses. On the cloud platform side, the hospital will manage the cloud that stores user's health data, personal information, and medical history of certain diseases of the user. The cloud plbothatform will call up the user's information and sends them to the edge-computing node for analyzation during an emergency, as shown in Fig.2.13.

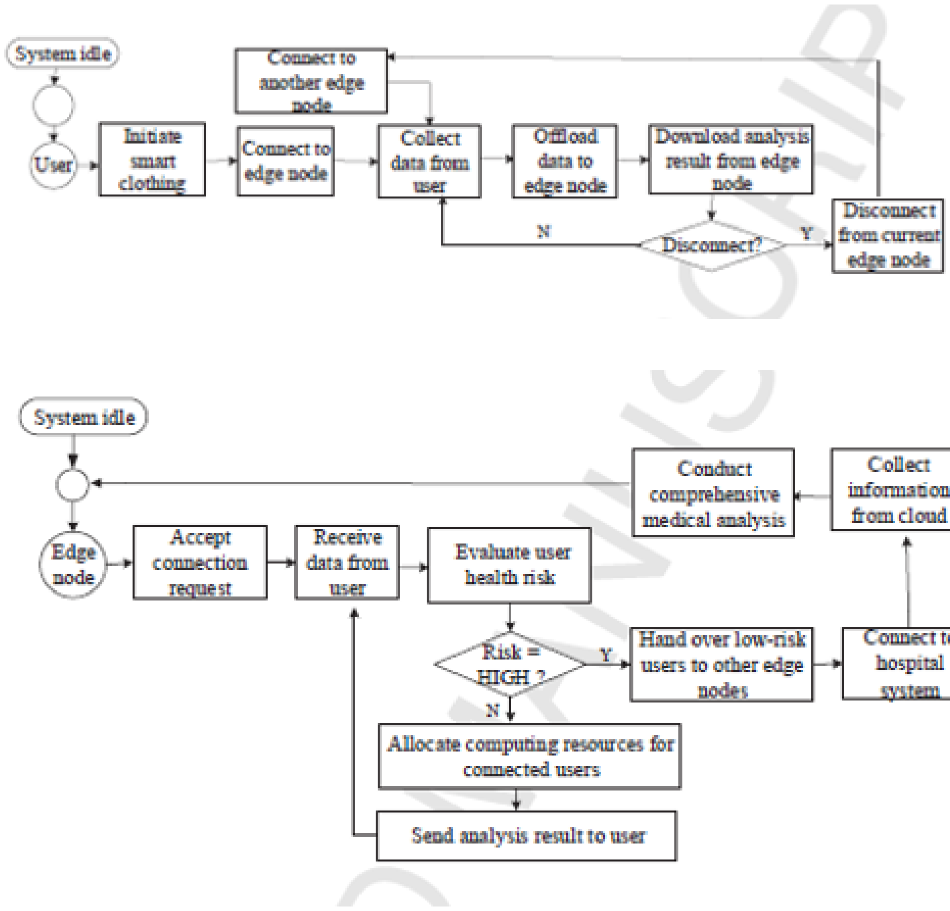

System idle
User
Initiate smart clothing
Connect to edge node
Connect to another edge node
Collect data from user
Offload data to edge node
Download analysis result from edge node
Disconnect?
N
Y
Disconnect from current edge node
System idle
Edge node
Accept connection request
Receive data from user
Evaluate user health risk
Conduct comprehensive medical analysis
Collect information from cloud
Risk = HIGH ?
Y
N
Hand over low-risk users to other edge nodes
Connect to hospital system
Allocate computing resources for connected users
Send analysis result to user

Fig.2.13: Flowchart of the edge node processing healthcare (Rausch et al., 2018)

2.6 Cardiac Cycle

The cardiac cycle is a sequence of the human heart that counts from the beginning of one beat to the beginning of the next beat. The cardiac cycle consists of two periods which are the systole(filling) and diastole(pumping) that form a complete heartbeat(Athanasiou et al., 2017). During the diastole period, the superior and inferior vena cava receives the returning blood from both the upper and lower human body, which then flows into the right atrium. As the blood filling up the right atrium, the pressure in the atrium increases. When the pressure of the right atrium surpasses the pressure of the right ventricle, the tricuspid valve that situated between the right atrium and right ventricle will open, which allows the blood to flow into the right ventricle. At the same time, the oxygenated blood will flow from the lung into the left atrium through the pulmonary veins, which increases the pressure in the left atrium as the blood flowing in. When the pressure in the left atrium surpasses the left ventricle, the mitral valve will open and allows oxygenated blood to flow from the left atrium to the left ventricle.

In the systole period, the blood in the left and right atrium will be forced to flow into the respective ventricle due to the depolarization of the atria as the result of the atrial muscle contraction while the pulmonary valves are closed. During the ventricular systole, the right and left ventricular muscle contracts. The tricuspid valve and mitral valve are closed while the pulmonary and aortic valve will be opened and the high-pressure blood will be pumped out from the heart to the body and lung through the aorta and the pulmonary artery. The heart muscle will then relax and starts the diastole phase again. The diastole and systole will repeat again and again which makes up the cardiac cycle. The Fig.2.14 shows the complete cardiac cycle.

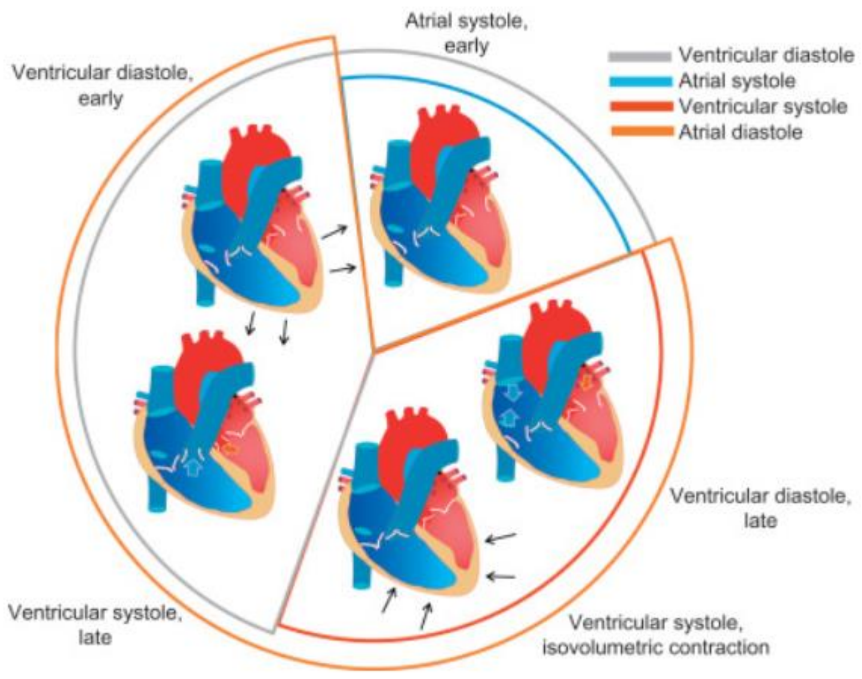
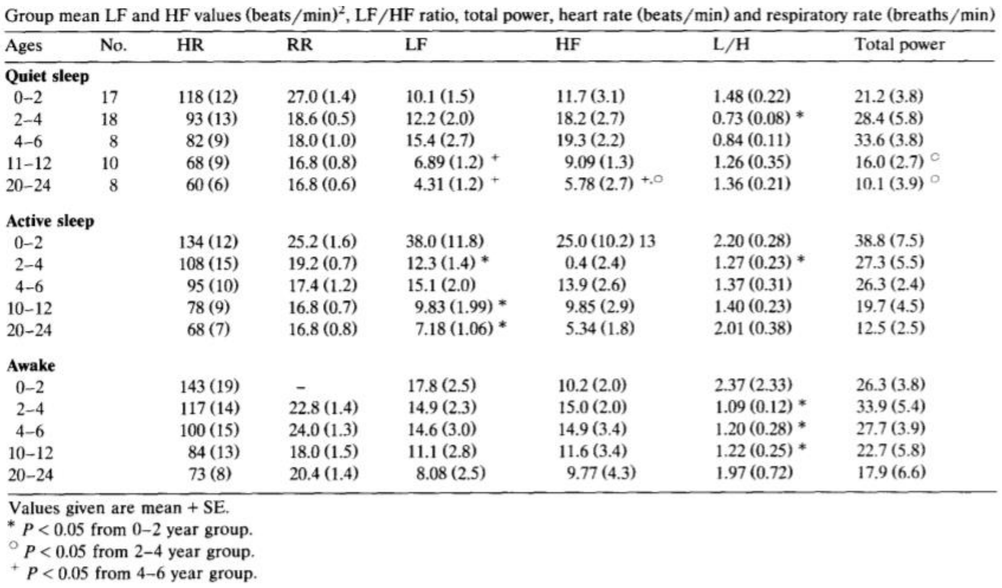

Atrial systole, early
Ventricular diastole, early
Ventricular diastole
Atrial systole
Ventricular systole
Atrial diastole
Ventricular diastole, late
Ventricular systole, late
Ventricular systole, isovolumetric contraction

Fig.2.14: The complete the cardiac cycle(Athanasiou et al., 2017).

2.7 Heart Rate of Human in Different Ages

Heart rate measures the number of human heartbeats per minute. The research to study the variability of heart rate in infants, children, and young adults was done by (Finley, J. P., & Nugent, S. T. 1995). The study found that the heart rate and respiratory rate of humans will be different depending on the age and the activity that is doing by the human as shown in Fig.2.15. From the result, it can be seen that the heart rate and respiratory rate of the human will decrease as the age increases. Besides, the heart rate and the respiratory rate are affected by the activity that is done by the human as the heart rate and respiratory rate are lower in quiet sleep compared to awake. According to Finley and Nugent, the primary finding of the research is that the variability of the heart rate in normal subjects depends on the age. This is proven right in both sleep state and awake. Besides, cardiac volume that changes with the growth of humans is one of the possible influences on the heart rate variability (Finley et al., 1987). The volume of heart chambers changes from infancy to adulthood, where infants may have a smaller heart that is more responsive to respiratory changes. As a result, the heart may experience more significant fluctuation of right atrial stretch and causes a more significant influence on the heart variability compared to the adult heart.



Group mean LF and HF values (beats/min)^2, LF/HF ratio, total power, heart rate (beats/min) and respiratory rate (breaths/min)

Ages	No.	HR	RR	LF	HF	L/H	Total power
Quiet sleep							
0–2	17	118 (12)	27.0 (1.4)	10.1 (1.5)	11.7 (3.1)	1.48 (0.22)	21.2 (3.8)
2–4	18	93 (13)	18.6 (0.5)	12.2 (2.0)	18.2 (2.7)	0.73 (0.08) *	28.4 (5.8)
4–6	8	82 (9)	18.0 (1.0)	15.4 (2.7)	19.3 (2.2)	0.84 (0.11)	33.6 (3.8)
11–12	10	68 (9)	16.8 (0.8)	6.89 (1.2) +	9.09 (1.3)	1.26 (0.35)	16.0 (2.7) ○
20–24	8	60 (6)	16.8 (0.6)	4.31 (1.2) +	5.78 (2.7) +,○	1.36 (0.21)	10.1 (3.9) ○
Active sleep							
0–2		134 (12)	25.2 (1.6)	38.0 (11.8)	25.0 (10.2) 13	2.20 (0.28)	38.8 (7.5)
2–4		108 (15)	19.2 (0.7)	12.3 (1.4) *	0.4 (2.4)	1.27 (0.23) *	27.3 (5.5)
4–6		95 (10)	17.4 (1.2)	15.1 (2.0)	13.9 (2.6)	1.37 (0.31)	26.3 (2.4)
10–12		78 (9)	16.8 (0.7)	9.83 (1.99) *	9.85 (2.9)	1.40 (0.23)	19.7 (4.5)
20–24		68 (7)	16.8 (0.8)	7.18 (1.06) *	5.34 (1.8)	2.01 (0.38)	12.5 (2.5)
Awake							
0–2		143 (19)	–	17.8 (2.5)	10.2 (2.0)	2.37 (2.33)	26.3 (3.8)
2–4		117 (14)	22.8 (1.4)	14.9 (2.3)	15.0 (2.0)	1.09 (0.12) *	33.9 (5.4)
4–6		100 (15)	24.0 (1.3)	14.6 (3.0)	14.9 (3.4)	1.20 (0.28) *	27.7 (3.9)
10–12		84 (13)	18.0 (1.5)	11.1 (2.8)	11.6 (3.4)	1.22 (0.25) *	22.7 (5.8)
20–24		73 (8)	20.4 (1.4)	8.08 (2.5)	9.77 (4.3)	1.97 (0.72)	17.9 (6.6)

Values given are mean + SE.
* $P < 0.05$ from 0–2 year group.
○ $P < 0.05$ from 2–4 year group.
+ $P < 0.05$ from 4–6 year group.

Fig.2.15: Typical heart rate for quiet sleep, active sleep and awake state(Finley, J. P., & Nugent, S. T. 1995)

2.8 Room Temperature in Malaysia (Kuala Lumpur and Kuching)

Room temperature is the temperature that is normal inside a building, and it is neither very cold or very hot (Dictionary, 2016). Malaysia is a tropical country with an annual mean temperature of 26.4 degrees Celsius. The average of daily maximum temperature in Malaysia is 34 degrees Celsius, while the average of daily minimum temperature is 23 degrees Celcius (Al-Tamimi & Syed Fadzil, 2011). A research was done by Jamaludin, Mohammed et al. (2015) to investigate the level of the thermal environment of the residential building in Kuala Lumpur, Bayan Lepas, and Kuching. Residential building with typical designed in Malaysia was simulated and the room temperature of the living space in the building was determined. The experiment found that the master bedroom has the highest room temperature of 32.6°C in Kuala Lumpur, and followed Bayan Lepas and Kuching with a room temperature of 31.6°C and 31.1°C. Fig.2.16 shows the average indoor temperature under Kuala Lumpur climate is around 29°C to 33°C.

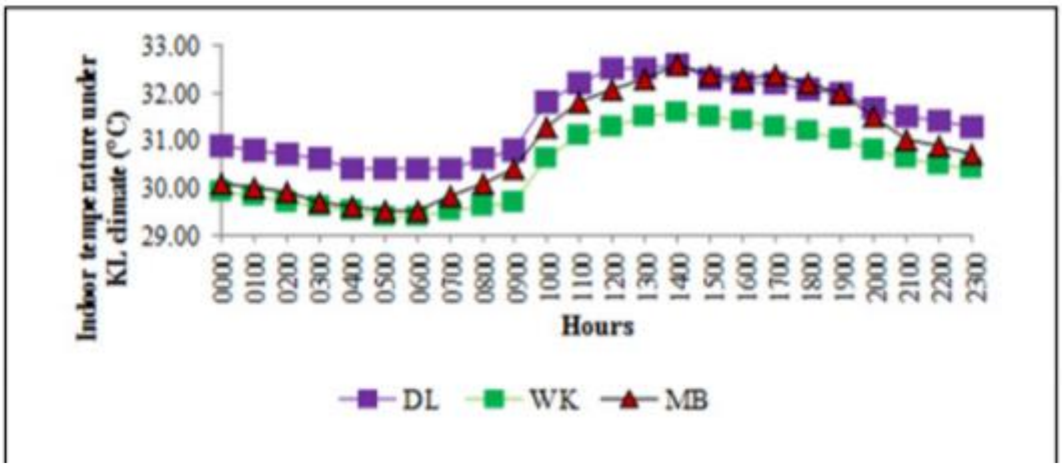


Fig.2.16: Indoor temperature in Kuala Lumpur Climate(Jamaludin, Mohammed et al. 2015)

2.9 Human Body Temperature

Human body temperature is used as an indicator of a person's health condition and illnesses, where abnormal body temperature is often one of the many symptoms of illnesses. For example, people who caught a cold or having fever will have abnormal body temperature. Thus, constant body temperature monitoring is vital to make sure that the person stays healthy. In 1851, the mercurial axillary thermometer was introduced by Wunderlich, where he used to measure the axillary temperature of 25000 people (Sund ‐ Levander et al., 2002). In his book, Wunderlich stated that the average human body temperature to be 37.0°C with a range of around 36.2°C to 37.5°C. He defined temperature above 37.5°C as 'the territory of fever' while the temperature of ≥ 38.0°C as fever (Mackowiak et al., 1992). A study was done by Sund ‐ Levander, Forsberg et al. (2002) to determine the average body temperature of adult men and women. The result found was as shown in the Fig.2.17.

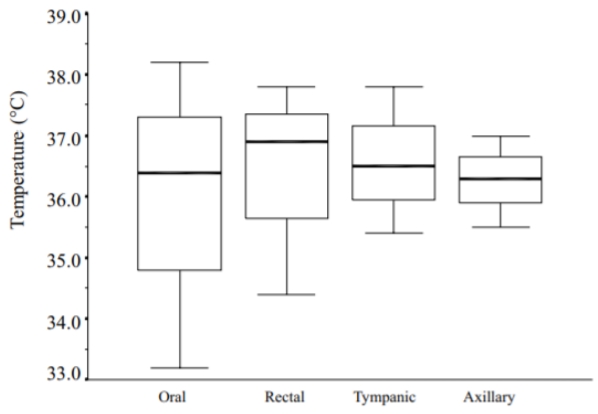
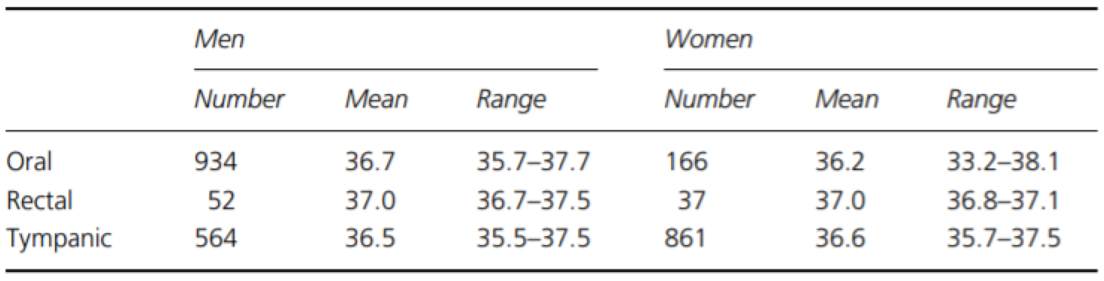


Fig.2.17: The results from 20 studies with strong or fairly strong evidence of normal oral, rectal, and tympanic temperature (°C) in adult men and women. (Sund‑Levander, Forsberg et al. 2002)

The bolded line in Fig.2.17 above shows the mean of the data collected while the thin line represents the range of the data. The mean value of the temperature measured from axillary, oral, rectal, and tympanic was 36.3°C, 36.4°C, 36.9°C, and 36.5°C with ±2 as the range respectively. Sund‑Levander, Forsberg et al. (2002) mentioned that gender to be one of the many factors that will affect body temperature measurement as there was a slight difference between the mean values of adult men and women as shown in Fig.2.18. The axillary temperature was not included in the table due to lacking evidence or fairy strong evidence reported to prove that axillary temperature affected by the gender.



	Men			*Women*		
	Number	*Mean*	*Range*	*Number*	*Mean*	*Range*
Oral	934	36.7	35.7–37.7	166	36.2	33.2–38.1
Rectal	52	37.0	36.7–37.5	37	37.0	36.8–37.1
Tympanic	564	36.5	35.5–37.5	861	36.6	35.7–37.5

Fig.2.18: The mean (m) and range/m ± 2 SD (°C) of studies with strong or fairly strong evidence in normal oral, rectal and tympanic body temperature in men and women (Sund‑Levander, Forsberg et al. 2002)

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter outlines the methods used in this project. The overview, setups, and design of this project are covered in this chapter. Besides, the hardware and software used in this project will be listed and explained. The flowchart, block diagram and communication model of the proposed system are shown in Fig.3.1, Fig.3.2 and Fig.3.3, respectively. The details of Fig.3.1, Fig.3.2 are discussed in section 3.2, while Fig.3.3 are discussed in section 3.3.

3.2 Overview of the System Design

This IoT based health monitoring system consists of three primary layers which are the gateway layer, the broker layer, and the data visualization and storing layer. The gateway layer consists of the LM35 temperature sensor, SEN-11574 pulse sensor, Arduino UNO and ESP8266 NodeMCU gateway, as shown in Fig.3.2. The pulse sensor is connected to the Arduino UNO that sends the heart rate data to ESP8266 via serial communication, and the LM35 temperature sensor that collects the body temperature of the user is connected to the ESP8266 directly. The sensors will collect the body temperature and heart rate of the user and send those data to the ESP8266 gateway. Then, the ESP8266 gateway will send the health data to the broker via Message Queuing Telemetry Transport (MQTT) communication protocol.

The broker layer consists of a broker that acts as the central processing unit in the system. The main processing unit used in this project is a Raspberry Pi 3 Model B+ with Raspbian Lite operating system (OS) running on it. Raspbian Lite is a simpler version of the standard Raspbian operating system that has less software installed in it which consumes fewer operating system resources. Besides, Mosquitto broker, Node-RED and PHPMyAdmin database are installed in the Raspberry Pi 3. Mosquitto broker is an open-source and lightweight message broker that implements MQTT protocol with low power consumption. Node-RED, which is a programming tool, is used to program the health monitoring system and communicate with various services such as IFTTT, Thingsboard, and PHPMyAdmin database. The health data collected by the broker will be visualized on the Node-RED and Thingsboard dashboard in real-time. At the same time, the health data will be stored in the PHPMyAdmin local database that allows the authorized user to keep track of the user's health data. The Raspberry Pi 3 broker will process all the data and direct every action programmed in the Node-Red. Alarm notifications and emails will be sent to assigned users during an emergency (Sudden changes in heart rate or abnormal body temperature) using If This Then That (IFTTT) services. The notification sent will direct the assigned users to the Thingsboard dashboard interface when clicked, and the users can track the health system's user location and health data from the Thingsboard dashboard.

The data visualization and storing layer consists of Thingsboard as the data visualization platform and PHPMyAdmin database. The health data of the user will be visualized in the Thingsboard and updated in real-time. Only authorized users will have access to view the data. At the same time, the health data will be stored in the PHPMyAdmin local database for tracking purposes.

The flowchart of the proposed system is shown in Fig.3.1. The system starts by collecting the health data in the ESP8266 gateway. The gateway then sends the data to the Raspberry Pi 3 broker. Then, the broker will process the data received based on the system's program. The health data will be sent to Thingsboard and PHPMyAdmin database for real-time data visualization and storage. The health data collected will be analyzed by the broker. If there is a sudden change in heart rate or abnormal body temperature (temperature >38°C or < 35°C), the broker will trigger the IFTTT to send the alarm notifications and emails to the assigned users. Then, the whole process will start all over again.

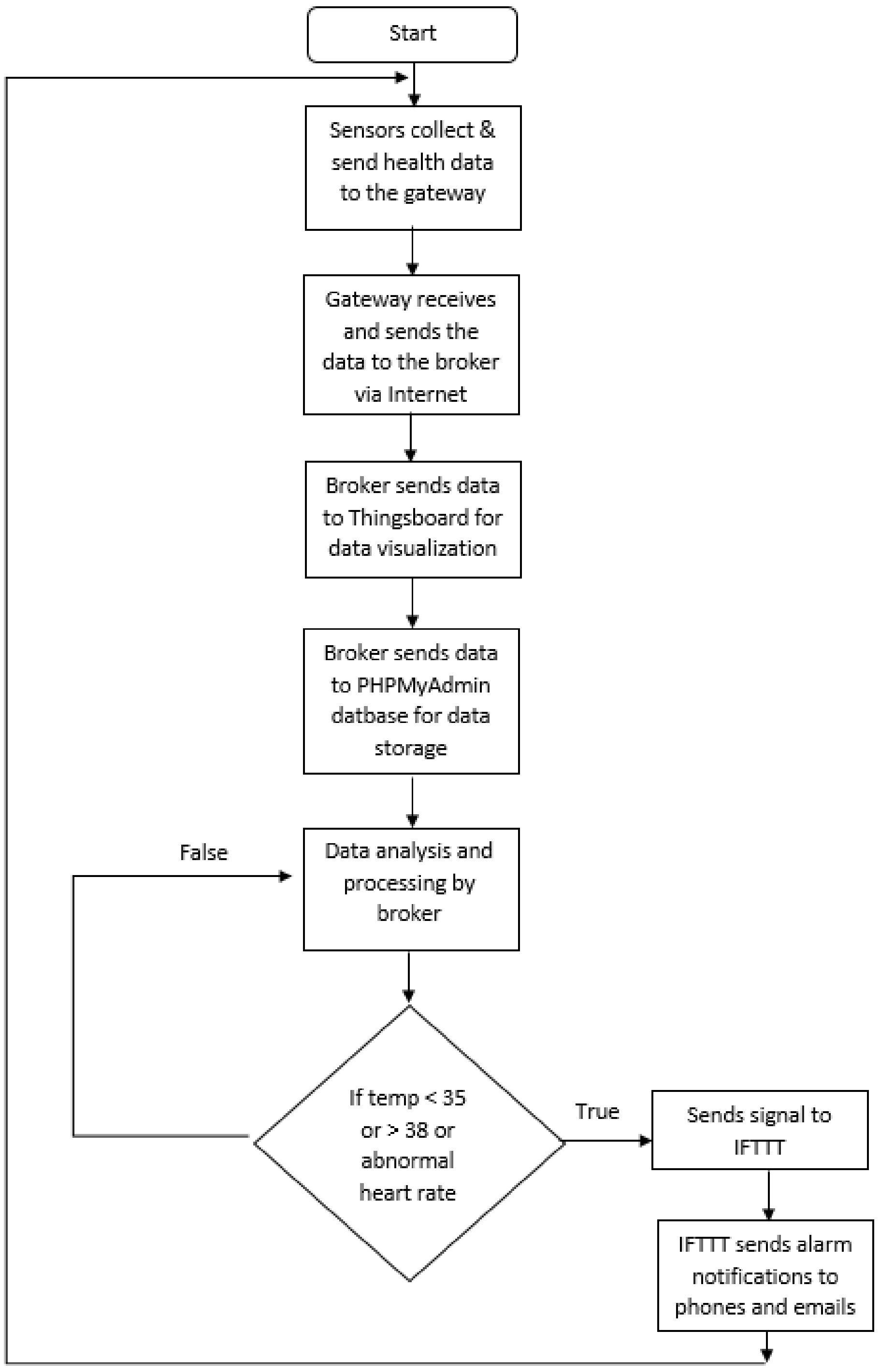

Start
Sensors collect & send health data to the gateway
Gateway receives and sends the data to the broker via Internet
Broker sends data to Thingsboard for data visualization
Broker sends data to PHPMyAdmin datbase for data storage
False
Data analysis and processing by broker
If temp < 35 or > 38 or abnormal heart rate
True
Sends signal to IFTTT
IFTTT sends alarm notifications to phones and emails

Fig.3.1: Flowchart of the proposed system

3.2.1 Block diagram of the proposed system

Fig.3.2 shows the block diagram of the proposed health monitoring system. The figure is discussed in section 3.2.

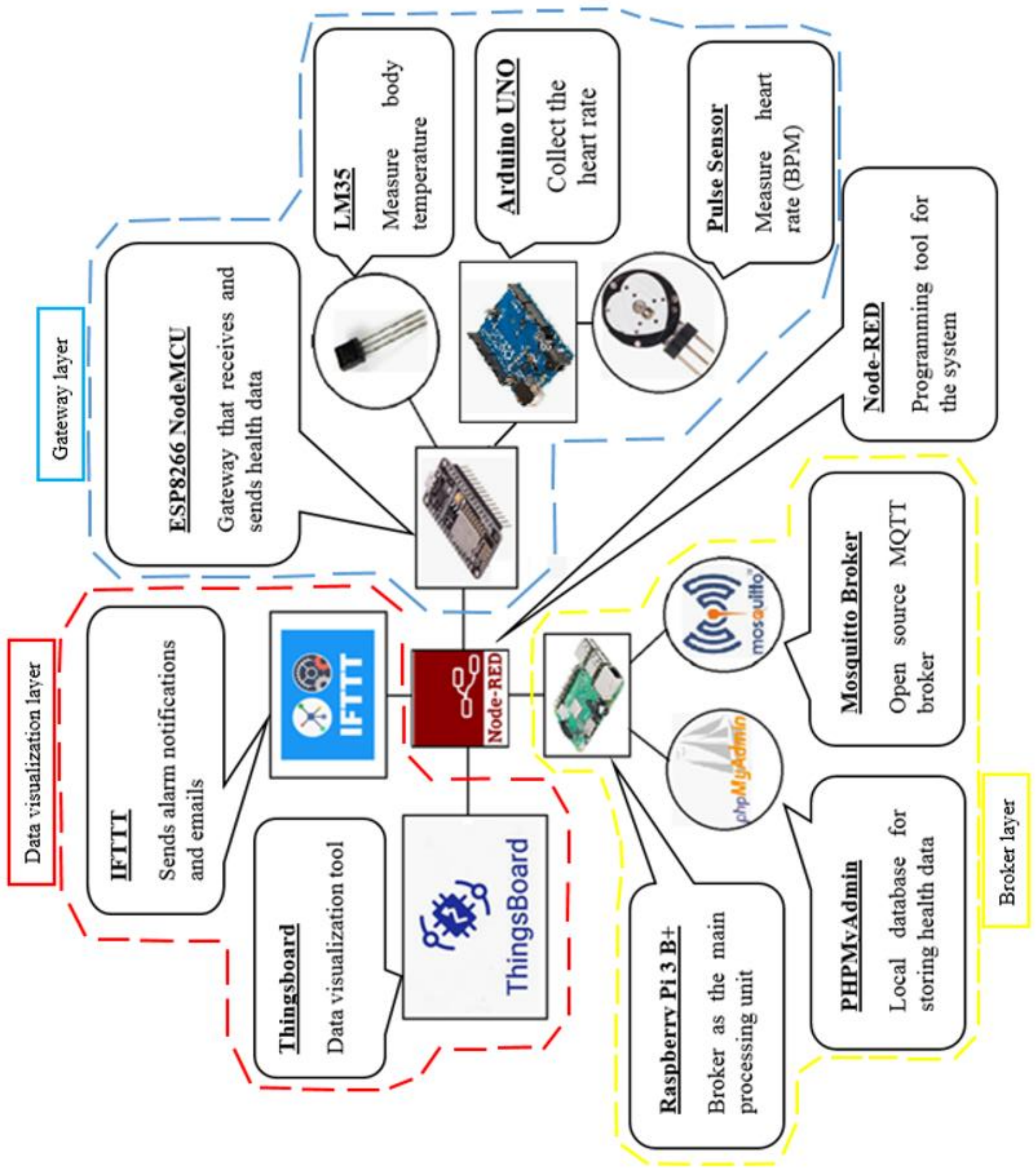


Fig.3.2: Block diagram of the health monitoring system

3.2.2 System communication model

Fig.3.3 shows the communication model of the proposed system. Serial communication and Message Queuing Telemetry Transport (MQTT) communication protocol are used in this system. Heart rate is calculated in Beats per Minute (BPM), and body temperature is measured in degree celsius (°C) in this project. Arduino UNO that receives the BPM from the pulse sensor, sends the data to ESP8266 NodeMCU via serial communication. The BPM and body temperature data will be "publish" to the broker via the MQTT communication protocol. Then, all data in this system (BPM, body temperature, coordinate, accuracy of the coordinate) will be "publish" to Node-RED for data processing. The "publish" process is similar to any data transferring process. Details of the MQTT communication protocol is discussed in section 3.3.3.

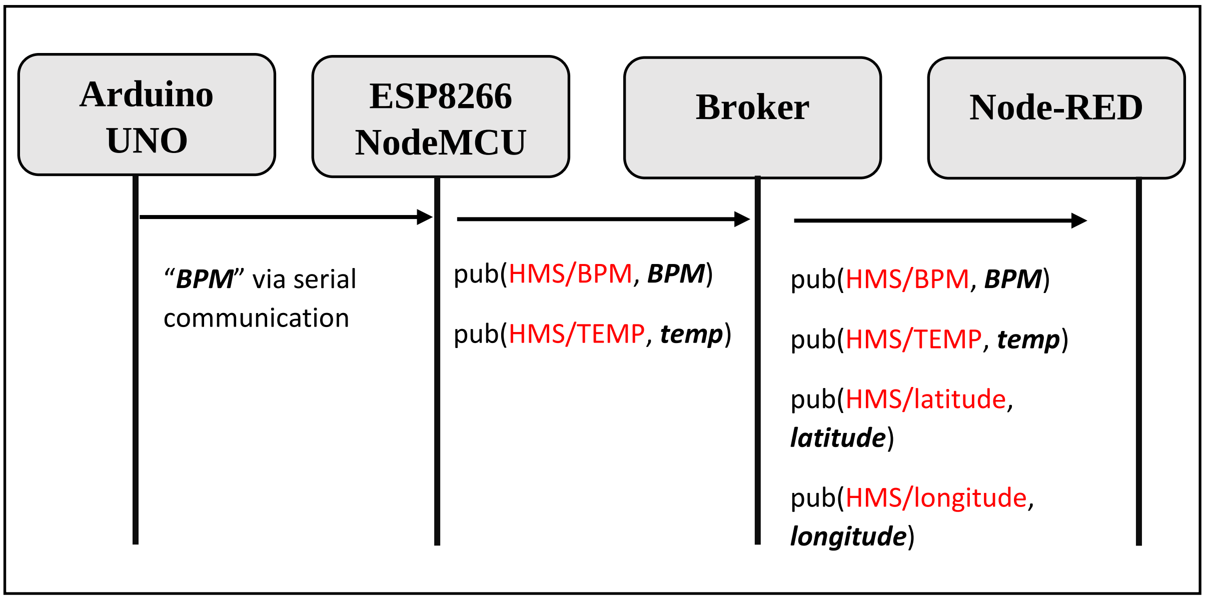


Fig.3.3: System communication model

3.3 Tools

3.3.1 Platform and services

- **Thingsboard**

  Thingsboard is an open-source IoT data visualization platform. The data collected by the broker will be sent to Thingsboard via the internet for real-time visualization. Thingsboard allows users to perform real-time data monitoring through the internet (ThingsBoard Architecture, 2018). The data sent to the Thingsboard will be received by a hub with a unique access token. In this project, the unique access token of the hub will be used to identify the destination of the data sent by the Raspberry Pi 3 broker. Thingsboard consists of customizable dashboards that visualize the data. In this system, body temperature, BPM, and location of the user are displayed in the dashboard.

- **Google Cloud SQL**

  Google Cloud SQL is a SQL based database that stores the users' health data in this project. Google SQL is one of the many services provided in the Google Cloud Platform. Google SQL database allows data to be stored in the cloud and reduces the risk of data loss. The database is automatically scaled up when the limit is near (Cloud SQL: Relational Database Service | Google Cloud, 2020). However, the service is not free of charge where the user will be charged monthly based on the usage of the database.

- **PHPMyAdmin**

  PHPMyAdmin is an open-source administration for MySQL. PHPMyAdmin offers complete web-based management of MySQL servers and data, where it provides a basic MySQL database and table operations and internal relational system that maintain metadata for advanced features (Delisle, 2009). The health data collected by the broker will be stored in the PHPMyAdmin database. PHPMyAdmin acts as the platform to manage and view the data recorded in the database. Besides, PHPMyAdmin can be connected to Google MySQL instances and allow the user to manage the Google MySQL database locally. PHPMyAdmin and MySQL are installed in the raspberry pi.

3.3.2 Software

- **Node-RED**

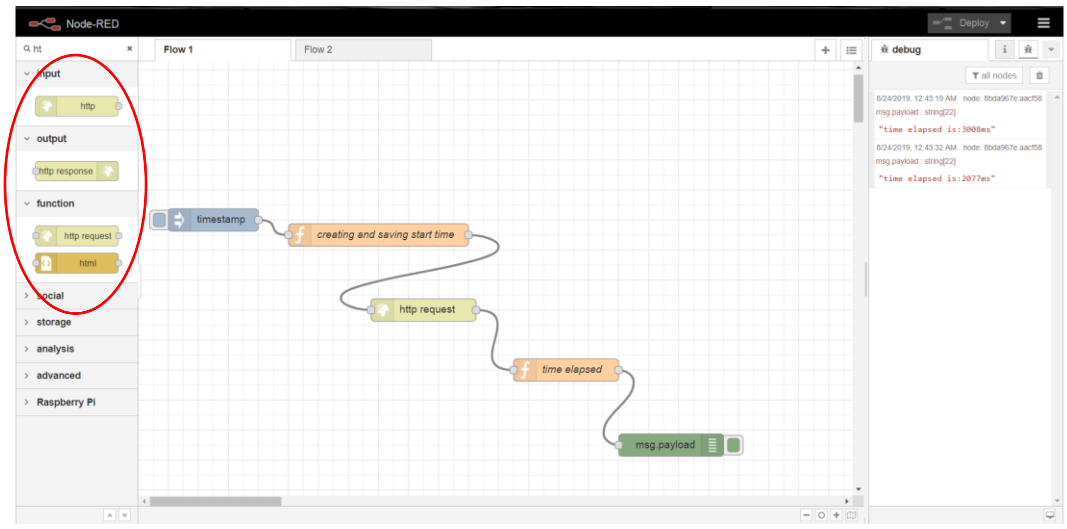
Node-RED is an open-source vision tool that makes the programming process easier by wiring different devices together. It simplifies the programming process by using different nodes without requiring lots of programming knowledge, as shown in the red circle of Fig.3.8. Node-RED consists of two main components as follow:

1. Nodes

   Nodes carry out their process based on their assigned function. Nodes can be installed from the Node-RED library easily. Nodes are written in node.js format.
2. Flow

   A Node-RED flow diagram is made up of the integration of different nodes.

The user needs to configure the required nodes and wire them together to complete a flow. Every node in the Node-RED has its own function and purpose. Node-RED is installed in the Raspberry Pi in this project. The flow editor of Node-RED can be accessed through the browser via the Internet Protocol (IP) address of the Raspberry Pi. The project flows can be deployed by clicking the deploy button on the top right corner of the editor interface.



```
8/24/2019, 12:43:19 AM   node: 8bda967e.aacf58
msg.payload : string[22]
"time elapsed is:3008ms"
8/24/2019, 12:43:32 AM   node: 8bda967e.aacf58
msg.payload : string[22]
"time elapsed is:2077ms"
```

Fig.3.8: Node-RED browser-based flow editor

Besides, Node-RED allows users to make and manage their user interface (UI). The data collected by Node-RED can be displayed on the dashboard as shown in Fig.3.9. The user interface is accessible through the browser which enables the user to monitor and control the system remotely via the internet.

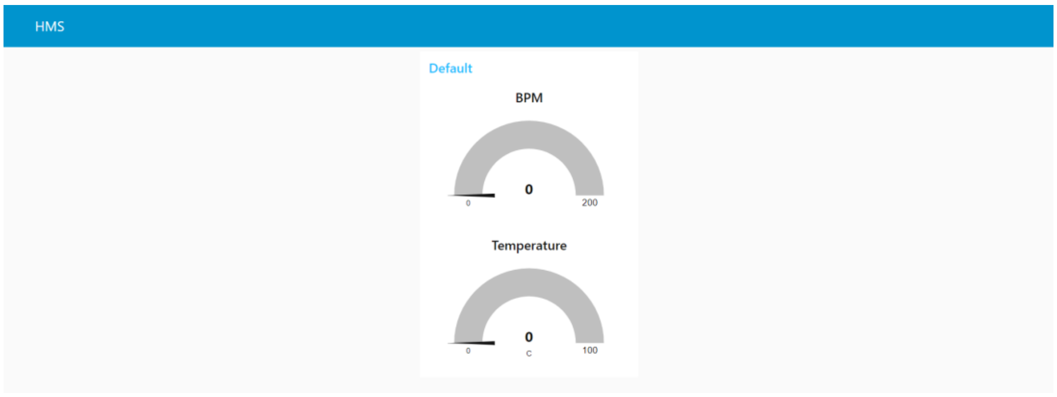


Fig.3.9: Node-RED user interface

- **Mosquitto broker**

Mosquitto is an open-source MQTT broker developed by Eclipse Foundation. Mosquitto broker is suitable to be used in this project since it provides a lightweight method of transmitting the messages with a sub/pub model which helps in power saving. Mosquitto broker is installed in the Raspberry Pi 3 with the communication architecture as shown in Fig.3.11.

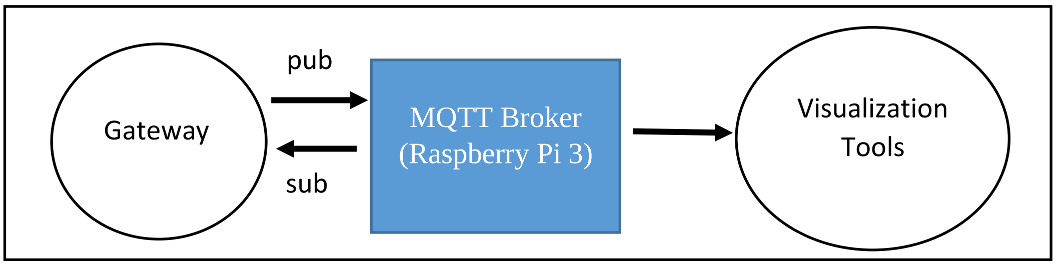
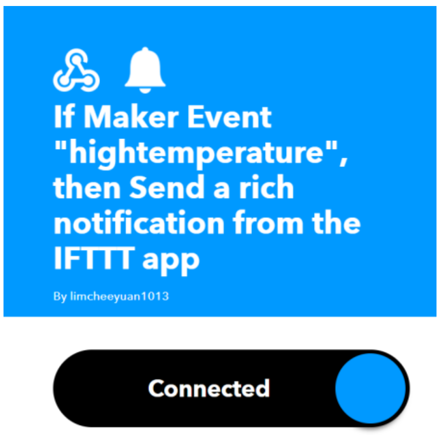


Fig.3.11: Communication architecture

In this project, the Raspberry Pi 3 broker subscribes to the gateway's topics. Once the gateway publishes something to the subscribed topics, the broker will receive the published data. The data received is then sent to the visualization tools for real-time data visualization.

- **IFTTT (If This Then That)**

  IFTTT is an open-source web-based service that allows different conditional statements to be chained together. IFTTT is used to send alarm notifications and emails during an emergency by using Applets that link various services together to perform a task, as shown in Fig.3.13.


If Maker Event "hightemperature", then Send a rich notification from the IFTTT app

By limcheeyuan1013

Fig.3.13: Applets created to send alarm notifications

3.3.3 Messaging Protocols

- **MQTT (Message Queuing Telemetry Transport)**

  MQTT protocol is one of the oldest M2M communication protocol, which is lightweight and energy-saving (MQTT Version 3.1.1, 2020). MQTT has a small data packet which transfers the data without consuming much power. MQTT protocol requires a low amount of bandwidth in data transmitting which decreases the latency of the process. MQTT is suitable to be used in this project since the healthcare system requires reliable real-time data, which can be realized with low latency MQTT protocol. MQTT is a publish/subscribe based protocol. MQTT connection involves MQTT clients which can be either publisher or subscriber. The publishers and subscribers will exchange data based on the topics. Information will be published to the topics by the publisher, and the subscriber that subscribed to that particular topic will receive the information.

3.3.4 Hardware

- **LM35 Temperature Sensor**

  LM35 shown in Fig.3.14 is a temperature sensor used to measure the user's body temperature in this project. The pinout consists of the positive terminal, negative terminal, and digital output pin. The pinout descriptions and specifications of LM35 are shown in Table 3.1 and Table 3.2, respectively.

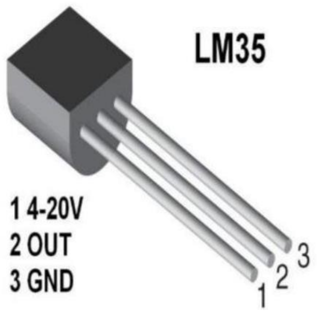


Fig.3.14: LM35 Temperature Sensor

Table 3.1: LM35 Pinout Description

Pin Number	Description
1	Power supply (4V – 20V)
2	Analogue output
3	Connected to the ground

Table 3.2: LM35 Specifications

Specifications	
Operating Voltage	4V- 20V
Operating Current	60µA
Temperature Range	-55°C - 150°C
Accuracy below 25 °C	$\pm$ 0.5°C
Accuracy above 25°C	$\pm$ 1.0°C
Temperature Slope	10mV/°C

- **Pulse Sensor (SEN-11574)**

The pulse sensor shown in Fig.3.15 is used to measure the user's heart rate of the user in Beats per Minute (BPM). The pinout consists of the positive terminal, negative terminal, and digital output pin, as shown in Table 3.3. The pulse sensor consists of a photosensor and an LED that emits green light. The green light and photosensor are used to detect the variation of blood flow caused by the heartbeat. Fig.3.16 shows the working principle of the pulse sensor. The LED on the pulse sensor emits green light while the photosensor detects the reflected green light. The amount of light reflected depends on the volume of the blood vessel in the artery. A high volume of blood vessel will absorb more light and causes a weaker light reflection.

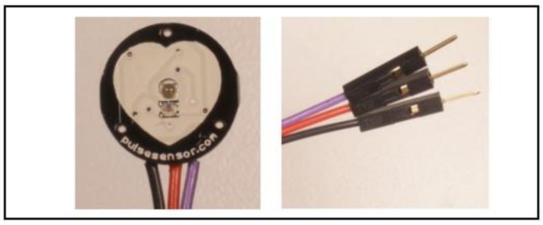


Fig.3.15: SEN-11574 Pulse Sensor

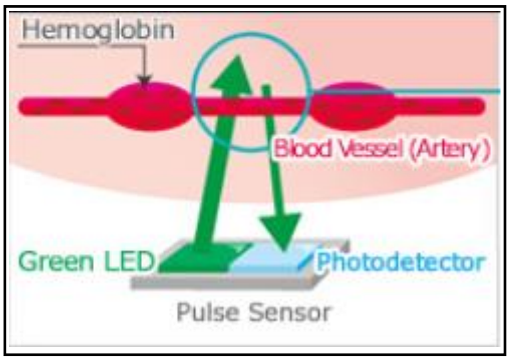


Fig.3.16: Working principle of SEN-11574 Pulse Sensor

Table 3.3: Pulse Sensor Pinout Descriptions

Pin Colour	**Descriptions**
Black	Power supply
Red	Connected to ground
Purple	Output

- **ESP8266 NodeMCU**

Nodemcu is used as a gateway that collects the health data from the connecting sensors in this project. The gateway communicates with the broker and Arduino UNO via MQTT protocol and UART serial communication. It sends the collected data to the broker for data processing. NodeMCU is used to track the user location with geolocation. With the help of geolocation, the GPS tracker is no longer needed as the geolocation function can track the location of the user through the available wifi connection surrounding the user. The pinouts and specifications of the NodeMCU are shown in Fig.3.17 and Table 3.4, respectively.

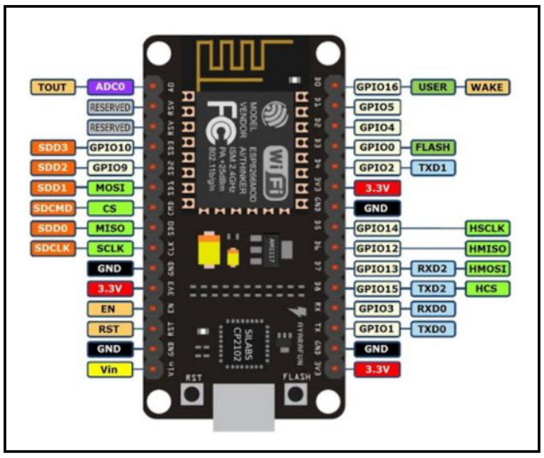


Fig.3.17: ESP8266 NodeMCU Pinouts

Table 3.4: ESP8266 NodeMCU Specifications

Specifications	
Operating Voltage	3.3V
Operating Current	10µA – 170mA
Input Voltage	7V – 12V
Processor	Tensilica L106 32-bit
Processor Speed	80 – 160MHz
GPIO	17
RAM	32K + 80K
ADC Pin	One input with 1024 resolution

- **Raspberry Pi 3 B+**

Raspberry Pi 3 is used as a broker with the mosquito broker installed in it. The broker acts as a central processing unit that directs the flow of the data from the source to the topic's subscriber via the pub/sub-model. The broker will send the health data to the PHPMyAdmin database for storage and Thingsboard for real-time data visualization. The pinouts and specifications of the Raspberry Pi 3 are shown in Fig.3.18 and Table 3.5, respectively.

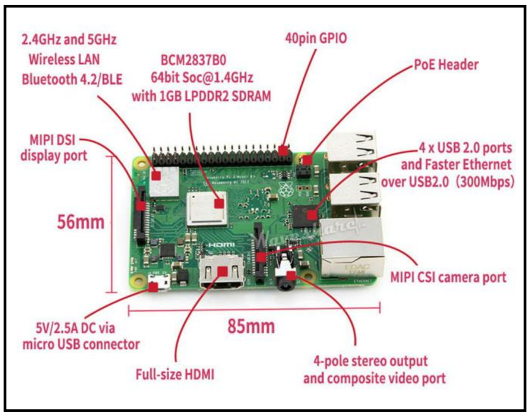

2.4GHz and 5GHz Wireless LAN Bluetooth 4.2/BLE
BCM2837B0 64bit Soc@1.4GHz with 1GB LPDDR2 SDRAM
40pin GPIO
PoE Header
MIPI DSI display port
4 x USB 2.0 ports and Faster Ethernet over USB2.0 (300Mbps)
56mm
MIPI CSI camera port
5V/2.5A DC via micro USB connector
85mm
Full-size HDMI
4-pole stereo output and composite video port

Fig.3.18: Raspberry Pi 3 Model B+ Pinouts

Table 3.5: Raspberry Pi 3 Model B+ Specifications

Specifications	
Operating Voltage	5V
Operating Current	2.5A
Processor	Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit @ 1.4GHz
Networking	Gigabit Ethernet (via USB channel), 2.4GHz and 5GHz 802.11b/g/n/ac Wi-Fi
RAM	1GB LPDDR2 SDRAM
Storage	Mirco-SD
GPIO	40
Dimensions	82mm x 56mm x 19.5mm
Weights	50g

- **Arduino UNO**

Arduino UNO is an open-source microcontroller board developed by Arduino.cc. The board equipped with analogue and digital input/output pins which are used to interact with other circuit boards and components. In this project, Arduino UNO is used to collect the data from the pulse sensor and send the data to the Nodemcu via serial communication. The pinouts and specifications of Arduino UNO are shown in Fig.3.19 and Table 3.6, respectively.

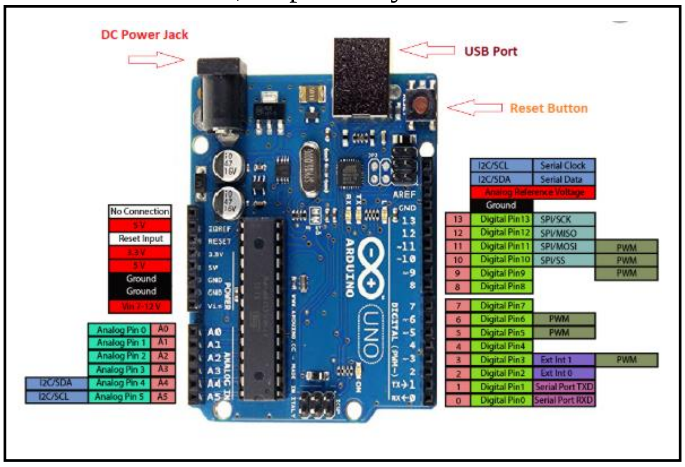


Fig.3.19: Arduino UNO Pinouts

Table 3.6: Arduino UNO specifications

Specifications	
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage	7V – 12V
Digital I/O Pins	14
Analogue I/O Pins	6
DC Current per I/O Pin	40mA
DC Current for 3.3V Pin	50mA
SRAM	2Kb
EEPROM	1Kb
Clock Speed	16MHz

3.4 Implementation Phase

3.4.1 Thingsboard platform setup

Thingsboard allows the user to customize their data visualization environment. The main interface of Thingsboard after logging in is shown in Fig.3.20.

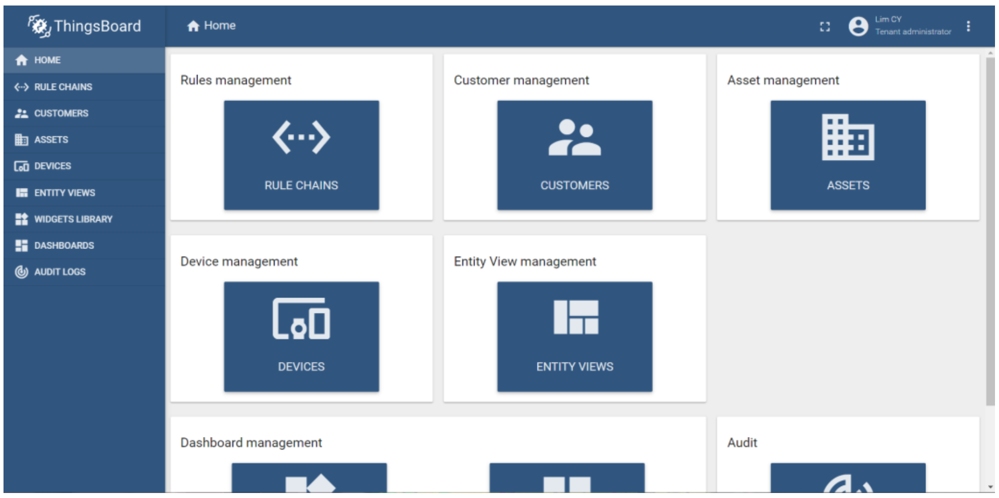
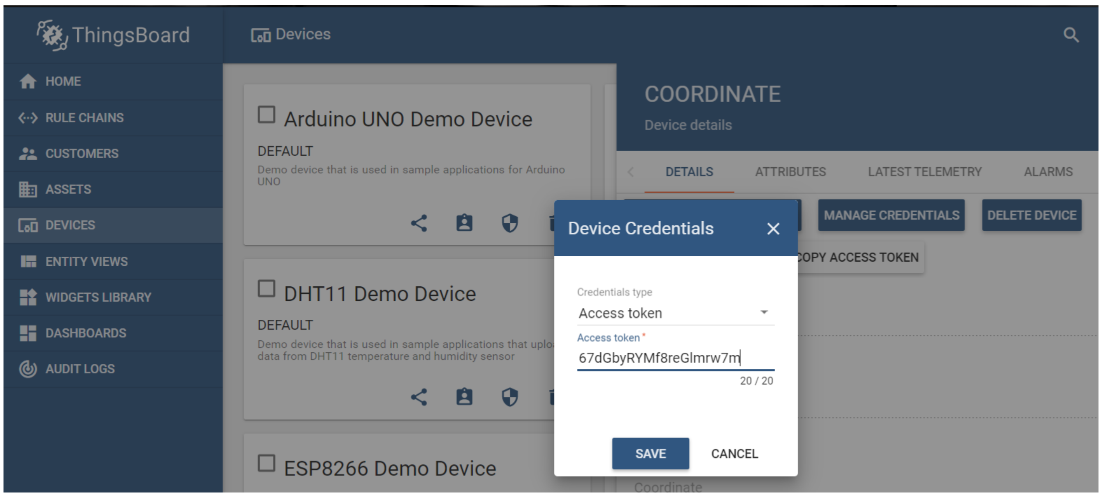


Fig.3.20: Thingsboard main interface

Users can set up hubs that receive all the data sent to the Thingsboard by clicking the "devices" button on the left side of the interface. "Devices" can be created and named based on user preferences. Each of the devices has a unique key known as the access token for identification purposes, as shown in the Fig.3.21. The data received by the Thingsboard will be distributed based on the access token.



Arduino UNO Demo Device
DEFAULT
Demo device that is used in sample applications for Arduino UNO

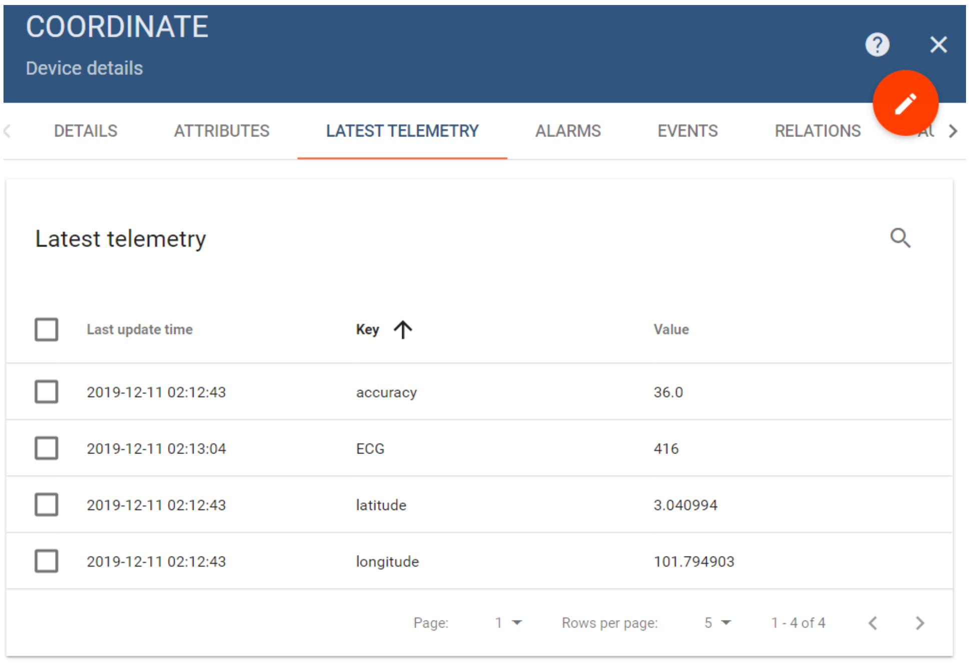
DHT11 Demo Device
DEFAULT
Demo device that is used in sample applications that uplo
data from DHT11 temperature and humidity sensor

ESP8266 Demo Device

Credentials type: Access token
Access token *: 67dGbyRYMf8reGlmrw7m
20 / 20

Fig.3.21: "Devices" in the Thingsboard platform with the unique access token

Fig.3.22 shows a hub with the name of "coordinate". The hub receives the coordinate data sent by the Raspberry Pi 3 broker in real-time. The coordinate data consist of longitude, latitude, and accuracy of the coordinate, as shown in Fig.3.22.



Latest telemetry

Last update time	Key	Value
2019-12-11 02:12:43	accuracy	36.0
2019-12-11 02:13:04	ECG	416
2019-12-11 02:12:43	latitude	3.040994
2019-12-11 02:12:43	longitude	101.794903

Page: 1 Rows per page: 5 1 - 4 of 4

Fig.3.22: The data received by the "coordinate" device.

In this project, data collected by the sensors will be sent from the broker to Thingsboard for real-time data visualization. Each of the data will be sent to different hubs with a different name and access token.

The user's health data and location are displayed on the dashboard interface that is customizable by the user. Fig.3.23 shows the dashboard that visualizes the temperature, BPM, and location of the user.

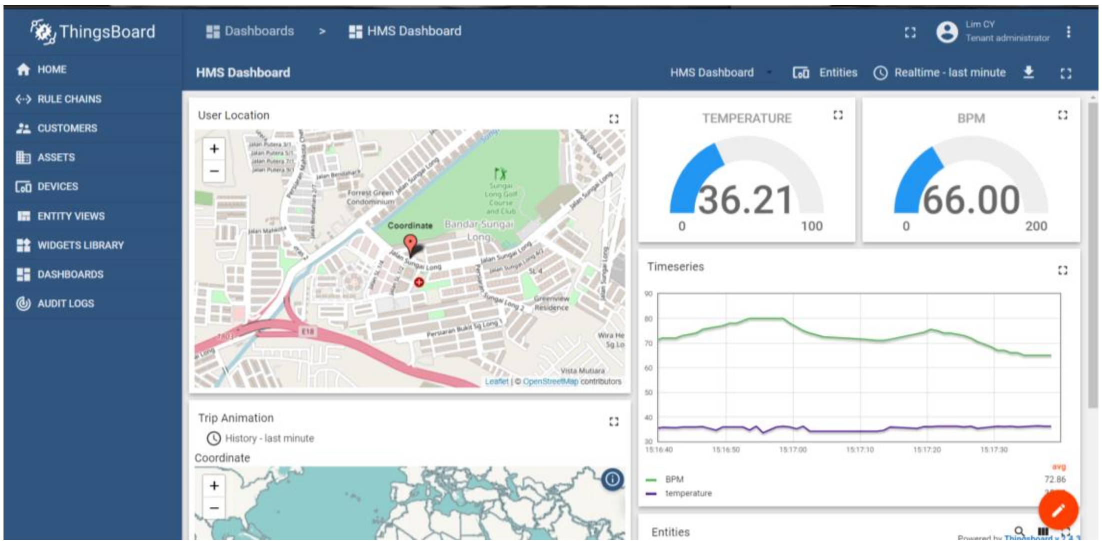


Fig.3.23: Dashboard of the health monitoring system

The admin can assign the authorization to view the dashboard to other users via email by clicking the “customer” on the left side of the Thingsboard main interface, as shown in Fig.3.20. The authorized users can log in to the dashboard to keep track of the user's condition.

3.4.2 Node-Red setup

Node-Red is installed in the broker that serves as an intermediate layer between the end devices and the cloud. Node-Red is used to connects several processes to form a complete system. Different nodes are wired together to form a complete “flow,” as shown in the Fig.3.24.

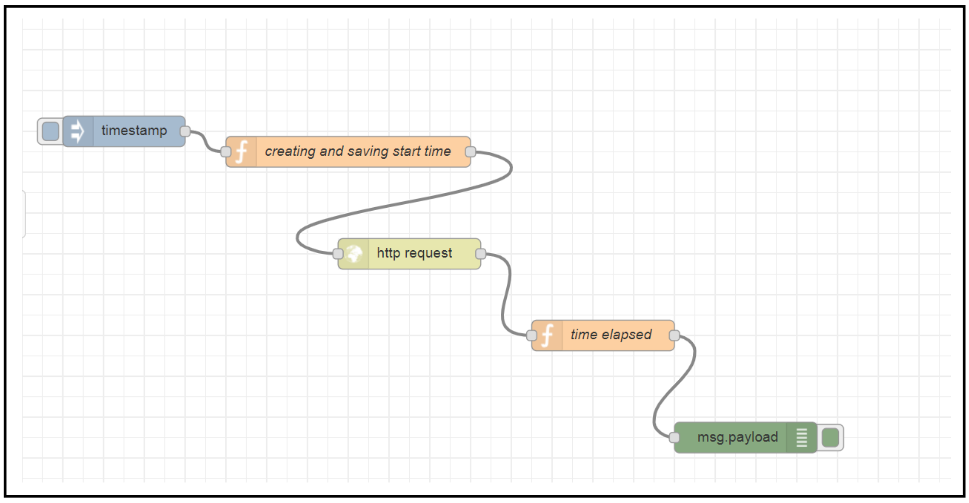


Fig.3.24: Sample flow to measure the time elapsed for a process

The sample flow shown in Fig.3.24 is used to measure the time elapsed for the "*Http* request" node. The average time for the "*Http* request" node to complete the request process is around 200ms. The time elapsed function measures the time elapsed by counting the start and stop time of the "*Http* request" node. The time elapsed is the interval between the start time and the stop time.

Node-Red consists of different types of nodes that are available on the left side of the Node-Red interface, as shown in Fig.3.8. The nodes can be linked together to form a process. Since the default nodes that available after installation of the Node-Red is often not enough to form a complicated flow or process, thus some extra nodes can be installed manually by clicking the "manage palette" button in the setting as shown in the Fig.3.25.

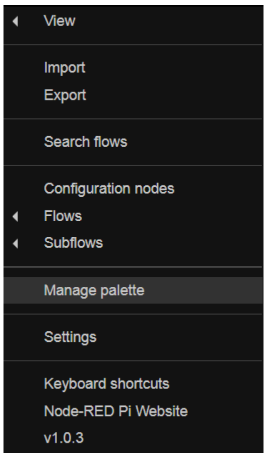


Fig.3.25: "Manage palette" button to configure the nodes in Node-Red

With the extra nodes installed, the user can link different nodes together to form several flows that carry out different actions. In this health system, function node with conditions statements will connect to the IFTTT nodes to trigger actions such as messaging, email, and calling the related personnel during an emergency. When the user experience sudden changes in heart rate or the body temperature falls below 35°C or exceeds 38°C, the broker will send a signal to inform IFTTT to carry out the actions.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Results of the prototype system

The results of the prototype system will be discussed in the following sections.

4.1.1 Real-time data visualization in Thingsboard

Health data receives by the system are displayed and updated on the Thingsboard in real-time, as shown in Fig.4.1. The data displayed are the body temperature, BPM, and the location of the user.

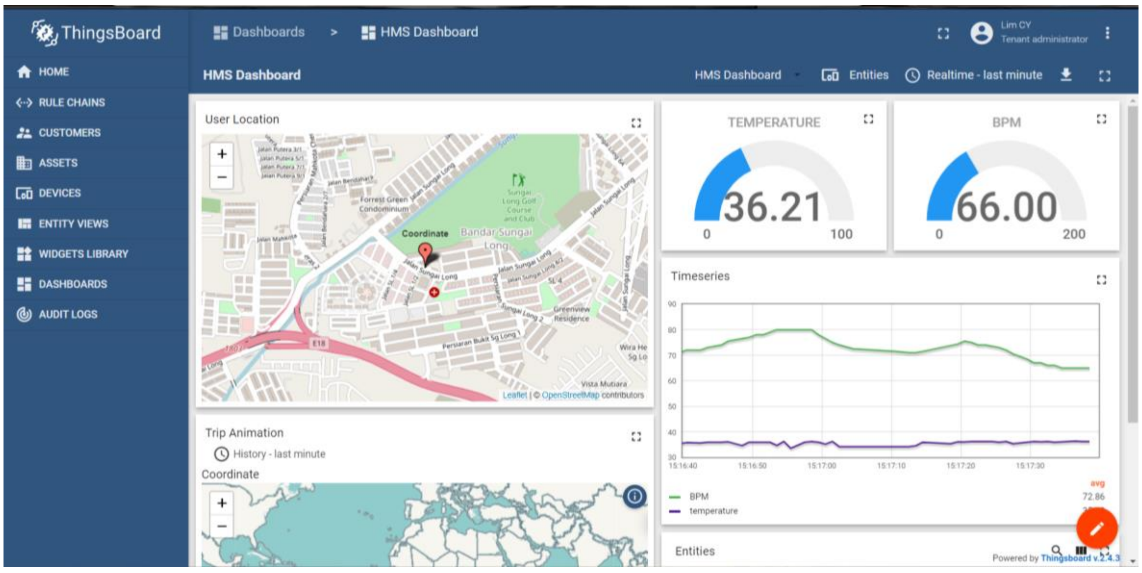


Fig.4.1: Thingsboard’s dashboard with health data

The health system is capable of tracking the user location through geolocation service, and the coordinate of the user is displayed on the Thingsboard. The interface shown in Fig.4.1 is in the administrator mode, where the admin has access to make any changes in the Thingsboard. Besides, “view only” access is assigned to other users through customer service and email addresses. The assigned users can view the data assigned by the admin. The customer service provided by Thingsboard enables the assigned users to keep track of the user's health condition. Fig.4.2 shows the email address of the assigned user, Fig.4.3 shows the login interface of the assigned user and Fig.4.4 shows the assigned user's "view-only" interface.

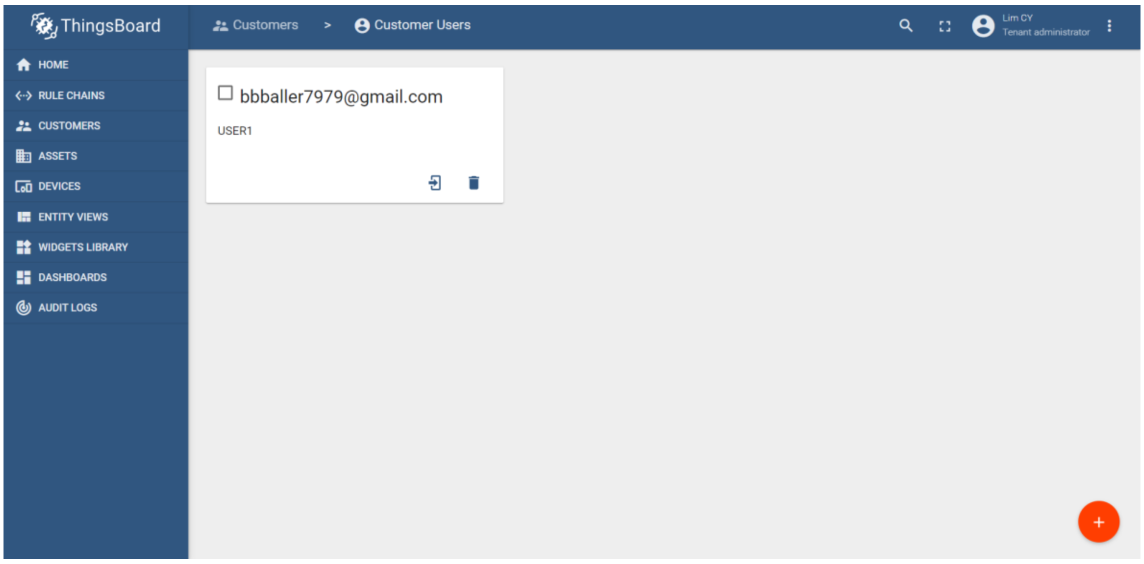


Fig.4.2: Assigned user with "view only" access

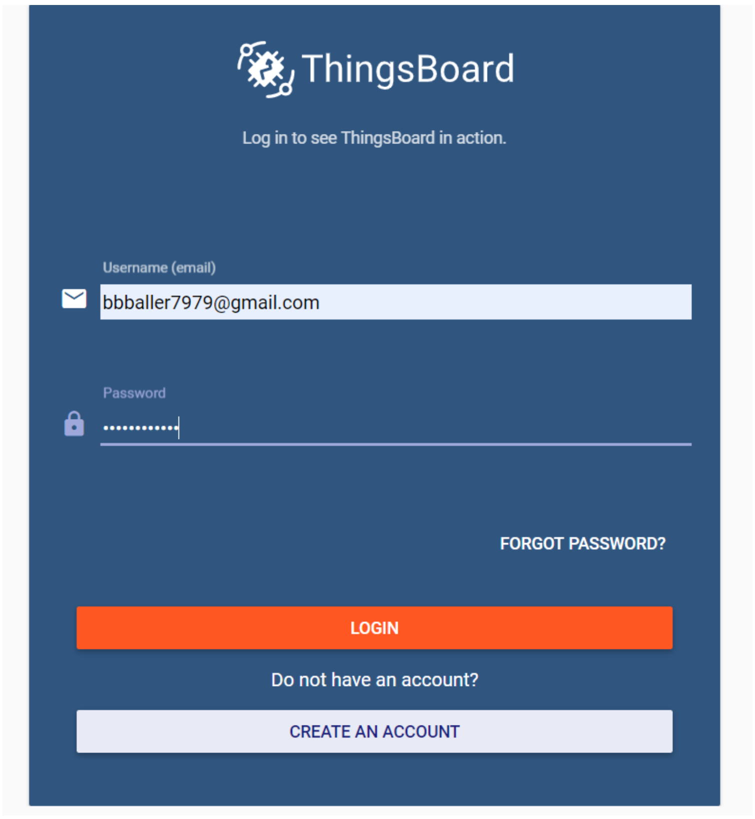


Fig.4.3: Login with the assigned user account

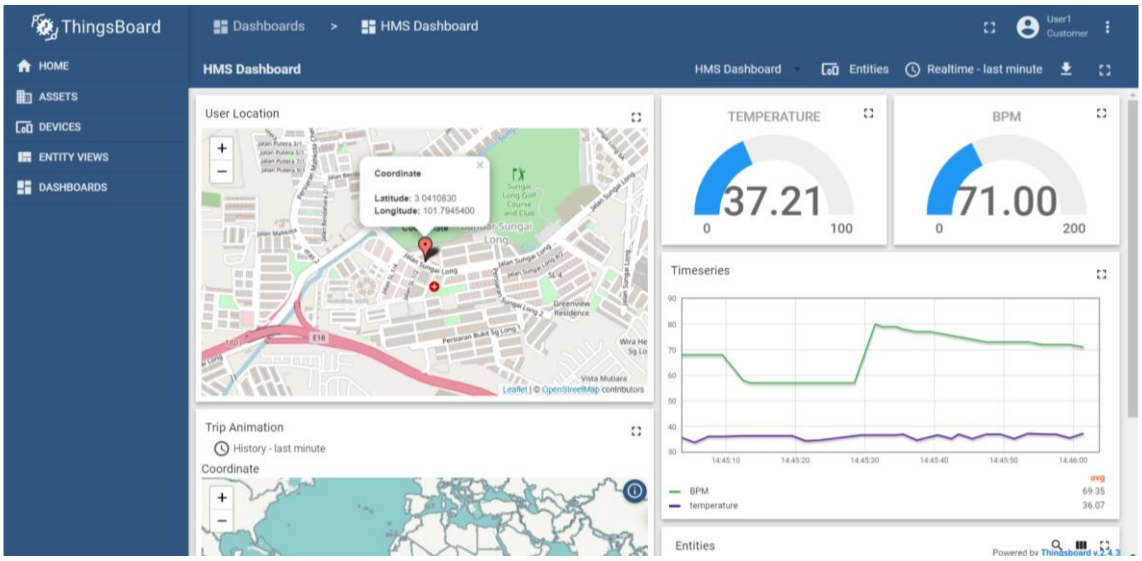


Fig.4.4: Assigned user's "view-only" Thingsboard interface.

4.1.2 Flow of the health system in Node-Red

In this system, Node-Red is used to program and connect all the processes in the system. Node-Red is installed in the Raspberry Pi 3 broker in the system. The broker receives data from the gateway and process those data locally. The complete flow of the proposed health system in the Node-Red is as shown in Fig.4.5

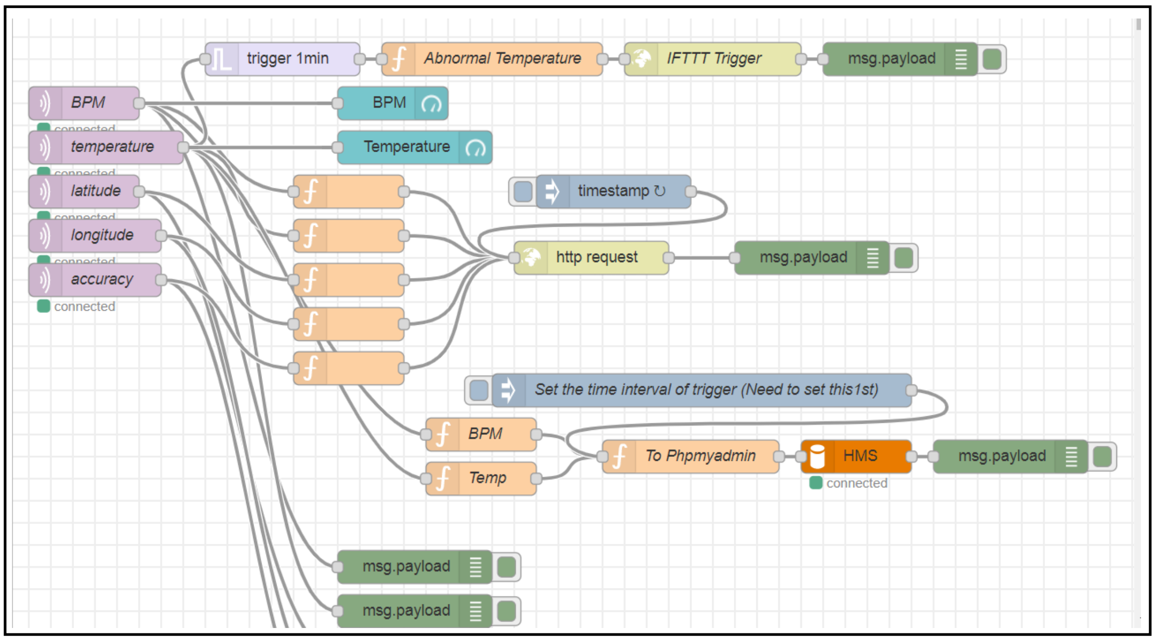


Fig.4.5: Complete flow of the proposed health system in Node-Red

The nodes in the Node-Red links and integrates all the processes and form a complete system. The function nodes in this system consist of simple JSON programming with if-else statements and conditions that trigger different types of actions such as sending data to Thingsboard, storing data to the database, and sending alarm notification to the user’s phone and email. Besides, the admin can track the health data in real-time through the Node-Red User Interface (UI) as shown in Fig.4.6. . Details of the flows are discussed in section 4.2.

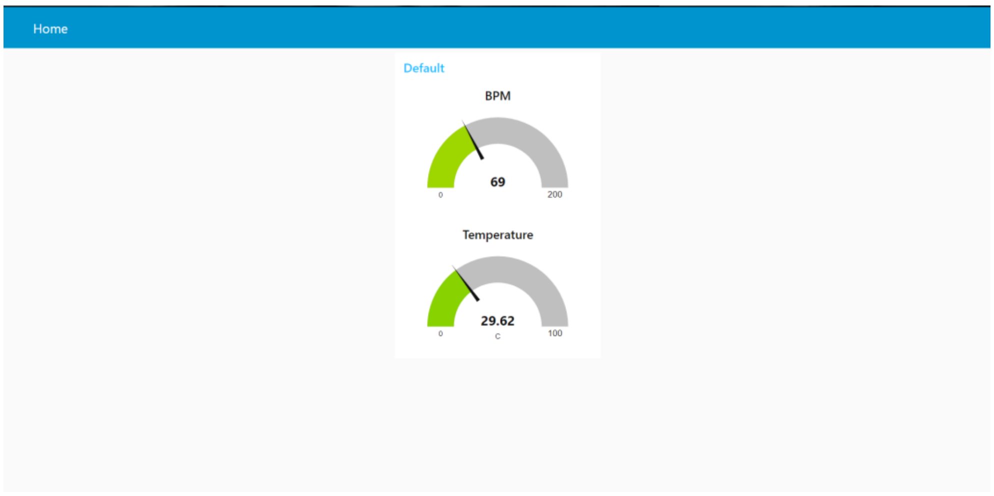
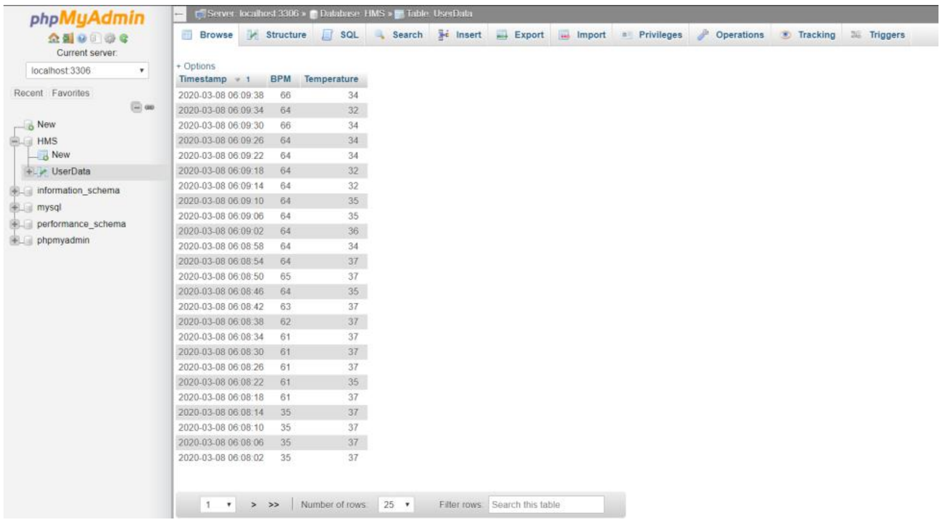


Fig.4.6: The Node-Red dashboard displaying the data

4.1.3 Data storing in PHPMyAdmin database

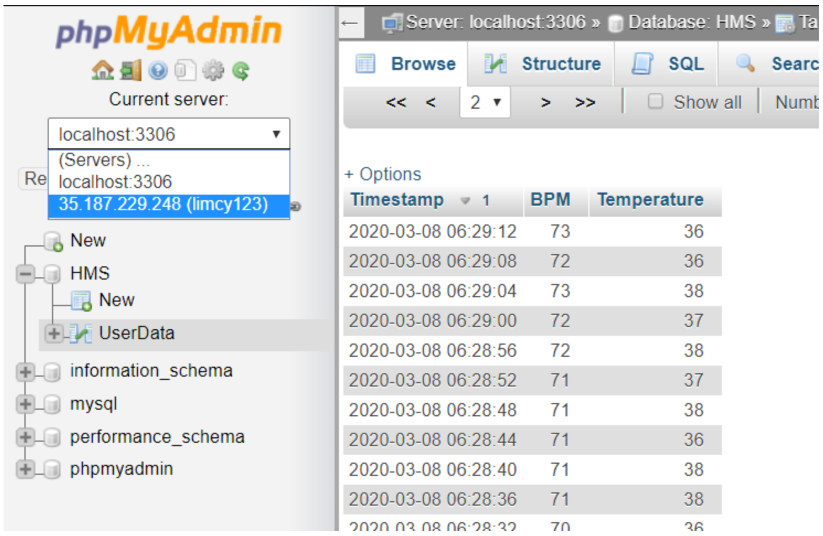
The health system allows the system’s admin to keep track of the health record of the users, where the health data receives by the broker will be saved in PHPMyAdmin local database. Storing the health data in the local database will reduce the risk of having the health data exposed and leaked, which increases the privacy and security of the health system. The health data are stored in the SQL format, as shown in Fig.4.7.



Timestamp	BPM	Temperature
2020-03-08 06:09:38	66	34
2020-03-08 06:09:34	64	32
2020-03-08 06:09:30	66	34
2020-03-08 06:09:26	64	34
2020-03-08 06:09:22	64	34
2020-03-08 06:09:18	64	32
2020-03-08 06:09:14	64	32
2020-03-08 06:09:10	64	35
2020-03-08 06:09:06	64	35
2020-03-08 06:09:02	64	36
2020-03-08 06:08:58	64	34
2020-03-08 06:08:54	64	37
2020-03-08 06:08:50	65	37
2020-03-08 06:08:46	64	35
2020-03-08 06:08:42	63	37
2020-03-08 06:08:38	62	37
2020-03-08 06:08:34	61	37
2020-03-08 06:08:30	61	37
2020-03-08 06:08:26	61	37
2020-03-08 06:08:22	61	35
2020-03-08 06:08:18	61	37
2020-03-08 06:08:14	35	37
2020-03-08 06:08:10	35	37
2020-03-08 06:08:06	35	37
2020-03-08 06:08:02	35	37

Fig.4.7: PHPMyAdmin local database with the data stored in SQL format

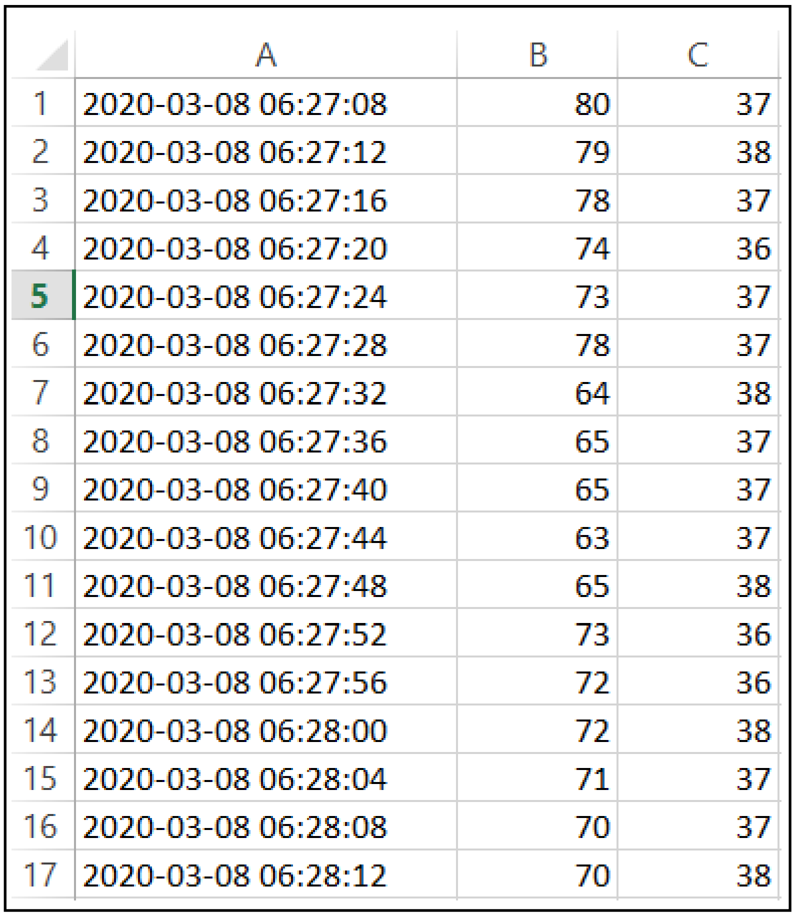
If the system's admin wishes to store the health data in the cloud, such as the google cloud platform that provides various types of database services, the PHPMyAdmin database can act as a platform to view the health data. One of the many services provided by the google cloud platform is the MySQL database that stores data in SQL format. PHPMyAdmin is capable of connecting itself to the google MySQL database where the system's admin can easily switch between the local database and the google MySQL database, as shown in Fig.4.8. The database can switch between the localhost database and the Google MySQL database with an IP-address of 35.187.229.248.



Timestamp 1	BPM	Temperature
2020-03-08 06:29:12	73	36
2020-03-08 06:29:08	72	36
2020-03-08 06:29:04	73	38
2020-03-08 06:29:00	72	37
2020-03-08 06:28:56	72	38
2020-03-08 06:28:52	71	37
2020-03-08 06:28:48	71	38
2020-03-08 06:28:44	71	36
2020-03-08 06:28:40	71	38
2020-03-08 06:28:36	71	38
2020-03-08 06:28:32	70	36

Fig.4.8: Switching between the local database and google MySQL database in PHPMyAdmin

The system's admin can easily export the data stored in the database through the PHPMyAdmin database. The PHPMyAdmin is capable of exporting the data in several formats such as CSV, PDF, SQL, and OpenDocument Spreadsheet. Fig.4.9 shows an example of data exported from the database in the OpenDocument Spreadsheet format and viewed in Microsoft excel.

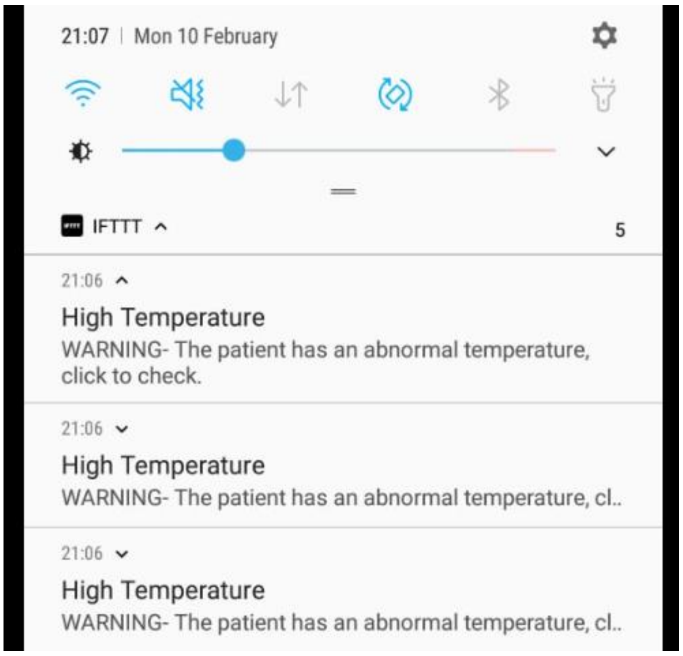


	A	B	C
1	2020-03-08 06:27:08	80	37
2	2020-03-08 06:27:12	79	38
3	2020-03-08 06:27:16	78	37
4	2020-03-08 06:27:20	74	36
5	2020-03-08 06:27:24	73	37
6	2020-03-08 06:27:28	78	37
7	2020-03-08 06:27:32	64	38
8	2020-03-08 06:27:36	65	37
9	2020-03-08 06:27:40	65	37
10	2020-03-08 06:27:44	63	37
11	2020-03-08 06:27:48	65	38
12	2020-03-08 06:27:52	73	36
13	2020-03-08 06:27:56	72	36
14	2020-03-08 06:28:00	72	38
15	2020-03-08 06:28:04	71	37
16	2020-03-08 06:28:08	70	37
17	2020-03-08 06:28:12	70	38

Fig.4.9: Data exported from the PHPMyAdmin database

4.1.4 Action triggered during emergency

When there are abnormal heart rate and body temperature, the health system will send alarm notifications to the assigned phone number and email addresses through IFTTT. The notification consists of alarm messages. The alarm notification will direct the user to the Thingsboard interface shown in Fig.4.11 when clicked. Fig.4.10 shows the alarm notifications sent to the phone when the user has an abnormal body temperature.


21:07 | Mon 10 February

IFTTT 5

21:06

High Temperature

WARNING- The patient has an abnormal temperature, click to check.

21:06

High Temperature

WARNING- The patient has an abnormal temperature, cl..

21:06

High Temperature

WARNING- The patient has an abnormal temperature, cl..

Fig.4.10: Alarm notification sent to the phone

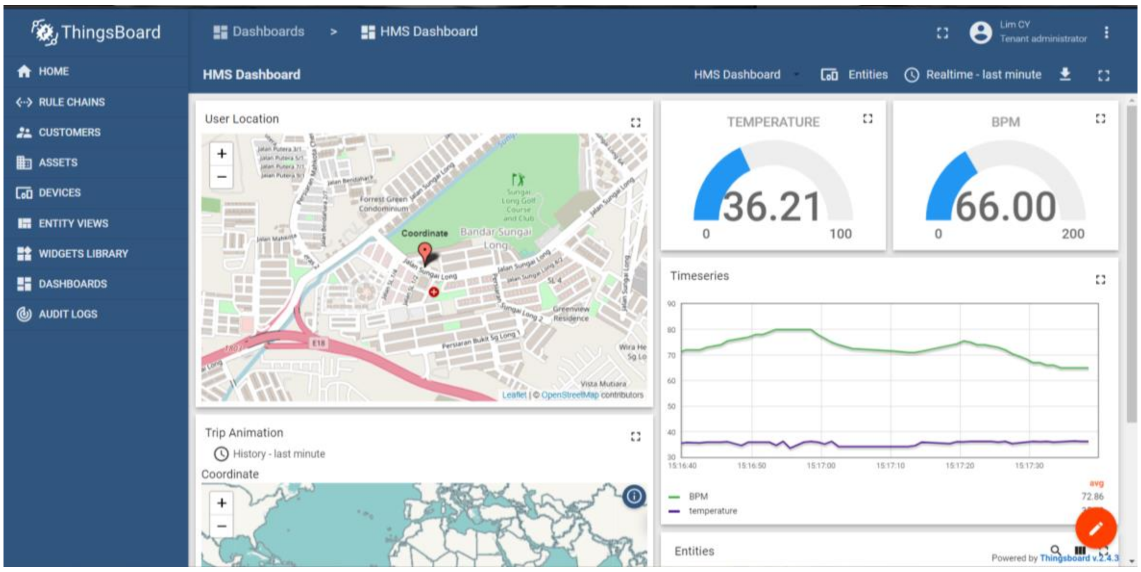


Fig.4.11: Thingsboard dashboard interface

4.2 Discussion

4.2.1 Thingsboard as the data visualization platform

Thingsboard meets the needs of this health system by visualizing the health data in real-time with a very short delay. As soon as the Node-Red receives the health data via MQTT from the gateway and sensors, the function nodes that send the health data to the Thingsboard in the Node-Red will be triggered, and the data displayed on the Thingsboard will be updated immediately. The time taken to update the data in the Thingsboard is measured using the flow, as shown in Fig.4.11:

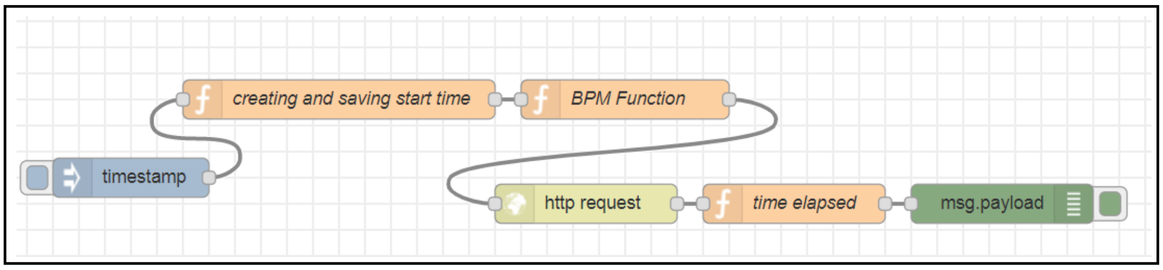


Fig.4.11: Flow used to calculate the time interval

The timestamp node will act as the MQTT node that outputs the health data in this sample flow. The flow starts measuring the time elapsed as soon as the health data receives by the MQTT nodes and stops after the health data is sent to the Thingsboard. Twenty reading was taken, and the average time calculated is 563ms.

4.2.2 Node-Red

Node-Red allows this health monitoring system to be programmed without requiring any complicated coding. Different types of nodes are used in the health system to form different flows that carry out different processes and actions. All the flows are then linked together to form a complete system.

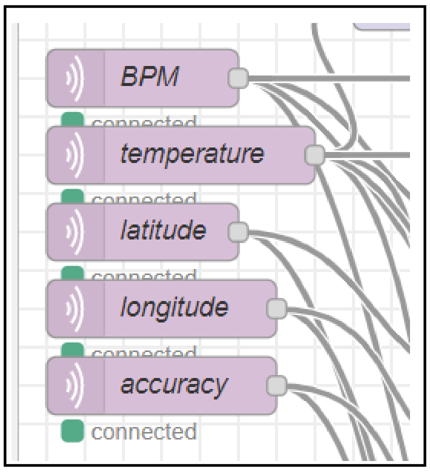


Fig.4.12: MQTT input nodes that receive data from esp8266 gateway

The MQTT input nodes shown in the Fig.4.12 are the MQTT nodes that receive the health data sent by the gateway. Each of the MQTT nodes will subscribe to a different topic, and the topic will determine the type of data that the node will receive. After receiving the data from the gateway, the MQTT nodes will pass the data to the next node.

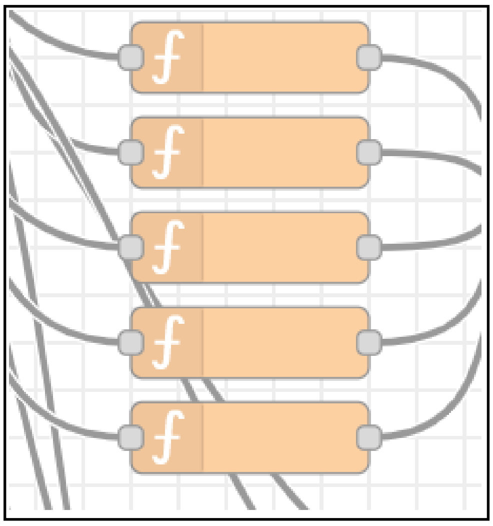
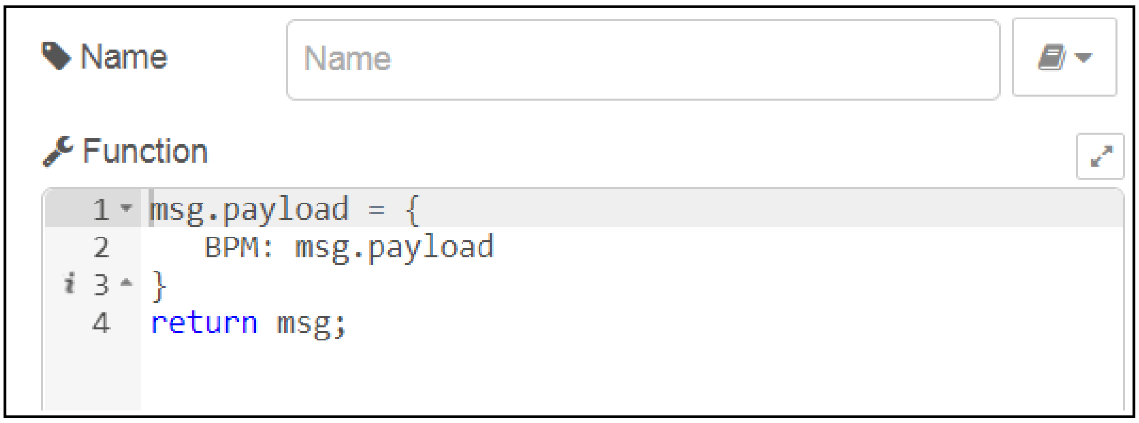


Fig.4.13: Function nodes that execute the code and assign the data into variables

The function nodes shown in Fig.4.13 are the nodes that will execute the function when they receive the data from the previous MQTT nodes. The function nodes are used to run the JavaScript code and process the receiving data. Function nodes are useful, especially when the user tries to include some conditions or if-else statements into the system to control the output of the nodes or to determine the next actions that will be taken by the system.


Name

Function

```
msg.payload = {
    BPM: msg.payload
}
return msg;
```

Fig.4.14: Sample code of function node

Fig.4.14 shows the simple coding of a function node that receives the heart rate value from the MQTT node. This function node will execute the code that sends the payload it receives from the MQTT node to the next node as output. The heart rate values will be stored in a BPM variable that will be received by the Thingsboard. The other four function nodes consist of similar coding with different variables. The reason for using these function nodes is to assign different data into separate variables so that the Thingsboard can recognize the data easily.

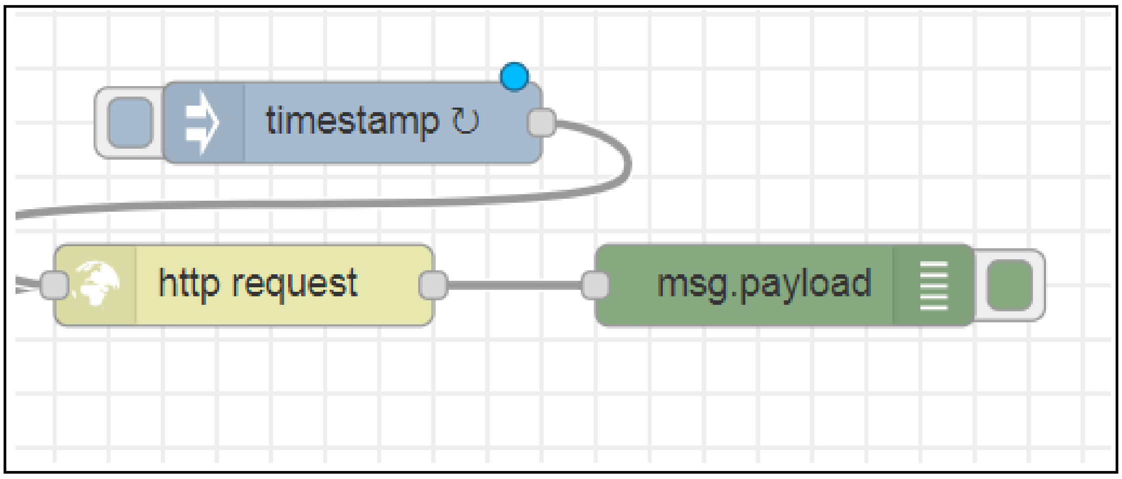
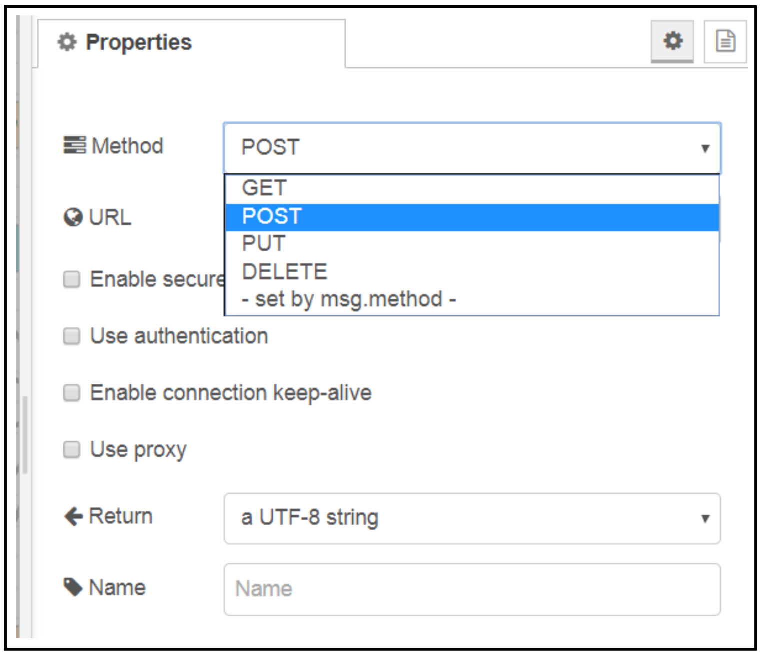


Fig.4.15: "*HTTP* request" node that connects to the Thingsboard

Fig.4.15 shows the flow with the timestamp node, *HTTP* request node, and the msg.payload node. The flow will send the payload that the *HTTP* request node receives to the Thingsboard. The *HTTP* request node consists of several types of methods as shown in the Fig.4.16. The "POST" mode is selected in this health system, where the node sends *HTTP* requests based on the URL provided and returns the response by sending the health data it receives from the previous node to the Thingsboard. The URL used in this node consists of a unique API key provided by the Thingsboard.


Properties

Method: POST

GET
POST
PUT
DELETE
- set by msg.method -

URL

☐ Enable secure

☐ Use authentication

☐ Enable connection keep-alive

☐ Use proxy

Return: a UTF-8 string

Name: Name

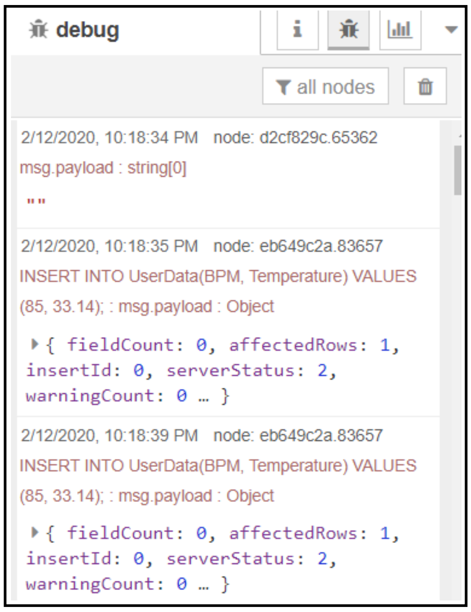
Fig.4.16: "*HTTP* request" node configuration

The *HTTP* request node can be used to obtain the data from the Thingsboard as well. This can be done by swapping the method to the "GET" method, and the node will make requests to the URL provided and get the data based on the API key that included in the URL. The URL requires the correct format to works properly where the format will be different for different websites and services. Table 4.1 shows the URL format of the Thingsboard and the IFTTT that are different from each other. Thus, it is crucial to obtain the correct format from the website for the node to works.

Table 4.1: Sample URL of Thingsboard and IFTTT

Service	Sample URL
Thingsboard	http://demo.thingsboard.io:80/api/v1/67dGbyRYMf8reGlmrw7m/telemetry
IFTTT	https://maker.ifttt.com/trigger/(Bansal and Gefen)}/with/key/b0_aKoThVlTQ6XnLvDNvyuUDIU5FNlDlrEBkMjooxUL

The msg.payload node is the node that shows the result of the flow after debugging in the Node-Red. Fig.4.17 shows the debug messages in the Node-Red.



```
2/12/2020, 10:18:34 PM   node: d2cf829c.65362
msg.payload : string[0]
""

2/12/2020, 10:18:35 PM   node: eb649c2a.83657
INSERT INTO UserData(BPM, Temperature) VALUES
(85, 33.14); : msg.payload : Object
▸{ fieldCount: 0, affectedRows: 1,
insertId: 0, serverStatus: 2,
warningCount: 0 … }

2/12/2020, 10:18:39 PM   node: eb649c2a.83657
INSERT INTO UserData(BPM, Temperature) VALUES
(85, 33.14); : msg.payload : Object
▸{ fieldCount: 0, affectedRows: 1,
insertId: 0, serverStatus: 2,
warningCount: 0 … }
```

Fig.4.17: Debug messages in Node-Red

Fig.4.18 shows the IFTTT flow that triggers the alarm notification during abnormal temperatures. The function node in this flow consists of if-else statements that determine the output of the function node. If the temperature is abnormal, the function node will output a msg.event that triggers the IFTTT to send the alarm notification to the user. The flow is set to be triggered every one minute to avoid the IFTTT from spamming the user with alarm notifications. Aside from sending alarm notification, the IFTTT provides many more other services such as sending emails, calling the user, and many more. Thus, the system's admin has lots of rooms to further improve the system by adding more services and actions.

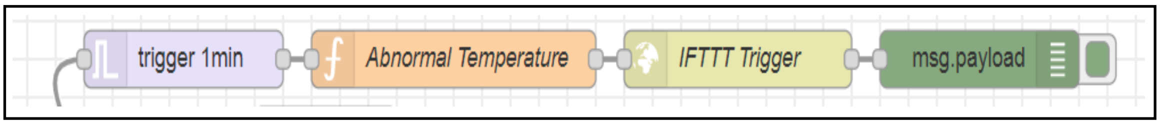


Fig.4.18: IFTTT flow that triggers alarm notification

Fig.4.19 shows the flow that sends BPM and temperature values to the PHPMyAdmin database every ten seconds. The BPM and Temp function nodes convert the BPM and temperature values into global variables that make them accessible by every flow in Node-Red. The BPM and temperature values are sent to the next function node that saves the values into the database. The database node requires credential information such as the database IP address, name of the database, username, and password to works. Since the local database is used in this system, the IP address used is 127.0.0.1 with a 3306 port. Aside from saving the data in the local database, a similar flow as shown in the Fig.4.19 can be used to save the data into the cloud, such as the Google cloud platform that provides various database services. However, the user will be charged according to the usage of the Google cloud platform services. The user requires to change the credential information to connect to the Google cloud database.

The advantages of saving the data into the cloud are high data accessibility and backup purposes. The user can access their data anywhere and anytime through the Google cloud platform without worrying about the data loss. If the user wishes to develop their health system in the Google environment, the data stored in the Google cloud database can be easily transferred between Google services as well. This will make the user's life easier by skipping the data importing process in the Google cloud.

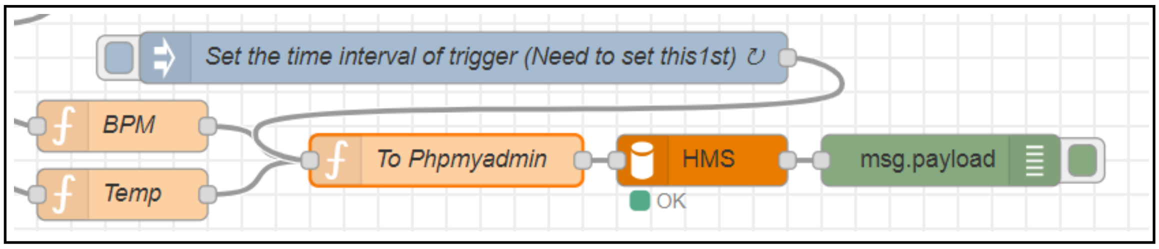


Fig.4.19: Flow that sends data to the PHPMyAdmin database.

4.3 Experiment

The functions of the health monitoring system were tested. The details of the experiments are discussed in the following sections. LM35 temperature sensor and SEN-11574 pulse sensor were tested. A 23 years old male was chosen as the test subject in the experiments. The formula that converts the output of the LM35 to degree Celsius was tested and discussed in section 4.2.1. Besides, error analysis to determine the margin error of the LM35 was carried out and discussed in section 4.2.2. Data comparison between the LM35 temperature sensor and thermometer was carried out, and the results are discussed in section 4.2.3.

An experiment to test the functionality of the pulse sensor was done and discussed in section 4.2.4. The experiment was carried out in a resting condition and active condition. The heart rate is measured in beats per minute (BPM). Besides, the accuracy test of the pulse sensor was carried out by comparing the BPM between the pulse sensor and manual measurement. The results are discussed in section 4.2.5.

4.3.1 LM35 temperature sensor

The body temperature of the system user is measured by using the LM35 temperature sensor. The formula used to obtain the temperature in Celsius (°C) is as follow:

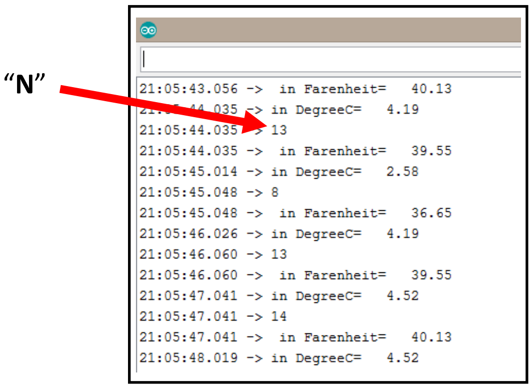
$$\frac{N}{1024} \times LM35\ Input\ Voltage = LM35\ Output\ Voltage\ (Input\ Voltage\ of\ A0)$$

N = “ADC Value” as shown in Fig.4.20

The value of “N” is determined by the resolution of the Analog-Digital Converter (ADC) of the NodeMCU. According to the NodeMCU datasheet, the resolution of the A0 pin of NodeMCU is 10 bit or 1024 steps. In the system setup, the input voltage of the LM35 is supplied by the 3.3V pin of the NodeMCU. In order to find out the input voltage of the A0 pin of NodeMCU, the “N” value is obtained from the Arduino IDE, and the LM35 input voltage is assigned to 3.3V as follow:

$$\frac{N\ from\ Arduino\ IDE}{1024} \times 3.3V = LM35\ Output\ Voltage\ (Input\ Voltage\ of\ A0)$$

Assuming the value of "N" is 14, the input voltage of the A0 pin in NodeMCU will be equal to 0.045117V after calculation. According to the LM35 datasheet, each increment of 10mV equals one degree Celsius (°C) which makes the 0.045117V (45.117mV) equals to 4.5117 degrees Celsius (°C).



```
21:05:43.056 ->  in Farenheit=   40.13
21:05:44.035 -> in DegreeC=   4.19
21:05:44.035 -> 13
21:05:44.035 ->  in Farenheit=   39.55
21:05:45.014 -> in DegreeC=   2.58
21:05:45.048 -> 8
21:05:45.048 ->  in Farenheit=   36.65
21:05:46.026 -> in DegreeC=   4.19
21:05:46.060 -> 13
21:05:46.060 ->  in Farenheit=   39.55
21:05:47.041 -> in DegreeC=   4.52
21:05:47.041 -> 14
21:05:47.041 ->  in Farenheit=   40.13
21:05:48.019 -> in DegreeC=   4.52
```

Fig.4.20: Outputs of LM35 temperature sensor in Arduino IDE

4.3.2 LM35 error analysis

Ideally, the input voltage of LM35 supplied by the NodeMCU should be fixed at 3.3V. However, there are fluctuations in the voltage supply that fluctuates the value of the temperature in practical cases. The voltage fluctuation may occur due to the long jumper wire that may introduce a small number of resistances into the system. As a result, the accuracy of the results is affected.

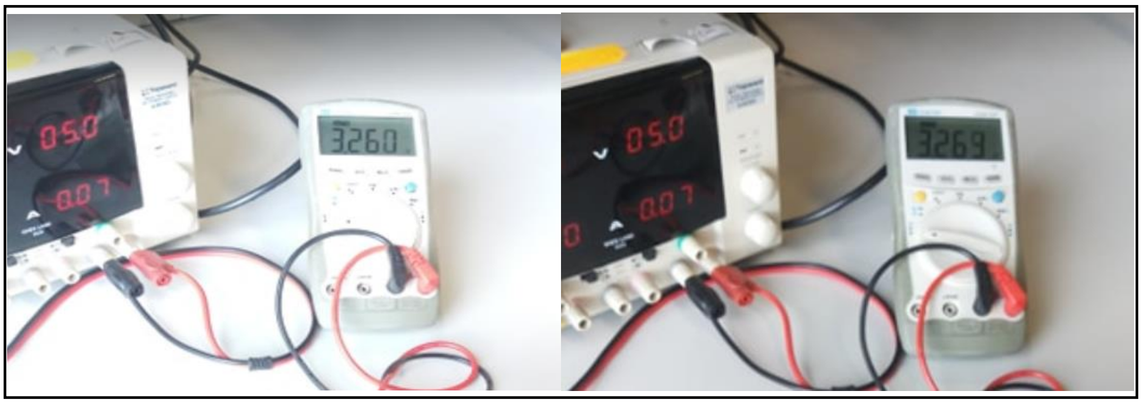
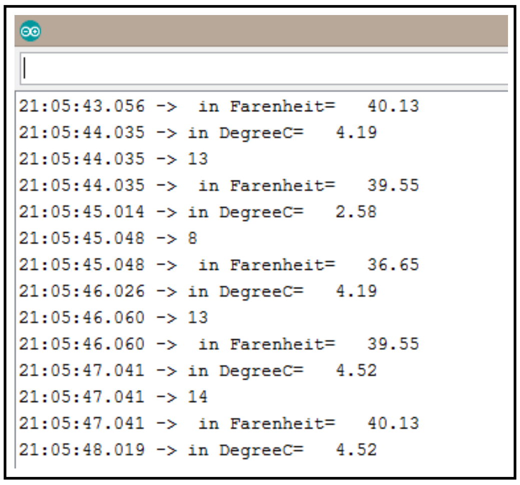


Fig.4.20: Measured ESP8266 3.3v pin output voltage

Besides, there is a margin error in the LM35 temperature sensor. Ideally, the temperature reading obtained in the Arduino IDE should be zero when the LM35 temperature sensor is disconnected from the system. However, some voltages are received by the NodeMCU A0 pin when the LM35 temperature sensor is disconnected from the system. The results of the Arduino IDE are as shown in the Fig.4.22.



```
21:05:43.056 ->  in Farenheit=   40.13
21:05:44.035 -> in DegreeC=   4.19
21:05:44.035 -> 13
21:05:44.035 ->  in Farenheit=   39.55
21:05:45.014 -> in DegreeC=   2.58
21:05:45.048 -> 8
21:05:45.048 ->  in Farenheit=   36.65
21:05:46.026 -> in DegreeC=   4.19
21:05:46.060 -> 13
21:05:46.060 ->  in Farenheit=   39.55
21:05:47.041 -> in DegreeC=   4.52
21:05:47.041 -> 14
21:05:47.041 ->  in Farenheit=   40.13
21:05:48.019 -> in DegreeC=   4.52
```

Fig.4.22: Temperature displayed in Arduino IDE with LM35 disconnected

In order to determine the average margin error of the LM35 temperature sensor, ten readings are sampled, and the average of those readings is calculated. The results obtained are shown in table 4.2.

Table 4.2: Ten sampled LM35 temperature sensor readings.

No.	“ADC” values	The temperature in Celsius (°C)
1	13	4.19
2	14	4.52
3	14	4.52
4	13	4.19
5	7	2.26
6	7	2.26
7	14	4.52
8	13	4.19
9	14	4.52
10	14	4.52
Average Value		3.97

The average margin error of the LM35 temperature sensor found is 3.97 degrees Celsius (°C). Thus, the outputs of the LM35 temperature sensor should be deducted by the margin error for better results.

4.3.3 Body temperature measurement by LM35 and thermometer

The LM35 temperature sensor and a thermometer are used to determine the axillary temperature. Both of the LM35 temperature sensor and the thermometer is placed under the armpit for temperature measurement. In this experiment, a 23 years old male was chosen as the test subject.

The body temperature of the subject was measured. The measurement was repeated three times, and the average of the three data was calculated. Ten readings were taken, and the results obtained are as shown in Table 4.3 and Table 4.4.

Table 4.3: Body temperature measured by the LM35 temperature sensor

No.	Temperature reading in Celsius (°C)			Average
1	36.0	34.3	36.3	35.5
2	36.3	36.3	36.3	36.3
3	36.3	36.3	36.3	36.3
4	36.6	36.6	36.6	36.6
5	36.9	35.2	36.6	36.2
6	36.9	36.9	35.2	36.3
7	36.9	35.5	34.6	35.8
8	36.0	36.0	36.2	36.1
9	36.5	36.5	36.5	36.5
10	36.3	36.6	36.6	36.5
Total Average				36.2

The measuring processes were repeated on the same subject by using a thermometer. The obtained results are as shown in Table 4.4.

Table 4.4: Body temperature measured by a thermometer

No.	Temperature reading in Celsius (°C)			Average
1	35.8	36.1	35.9	35.9
2	36.2	36.2	36.3	36.2
3	36.2	36.1	36.1	36.1
4	36.2	36.2	36.1	36.2
5	36.1	36.3	36.2	36.2
6	36.2	36.2	36.2	36.2
7	36.4	36.4	36.3	36.4
8	36.1	36.1	36.3	36.2
9	36.2	36.1	36.3	36.2
10	36.2	36.4	36.4	36.3
Total Average				36.2

The graph of Table 4.3 and 4.4 are plotted, as shown in Fig.4.22.

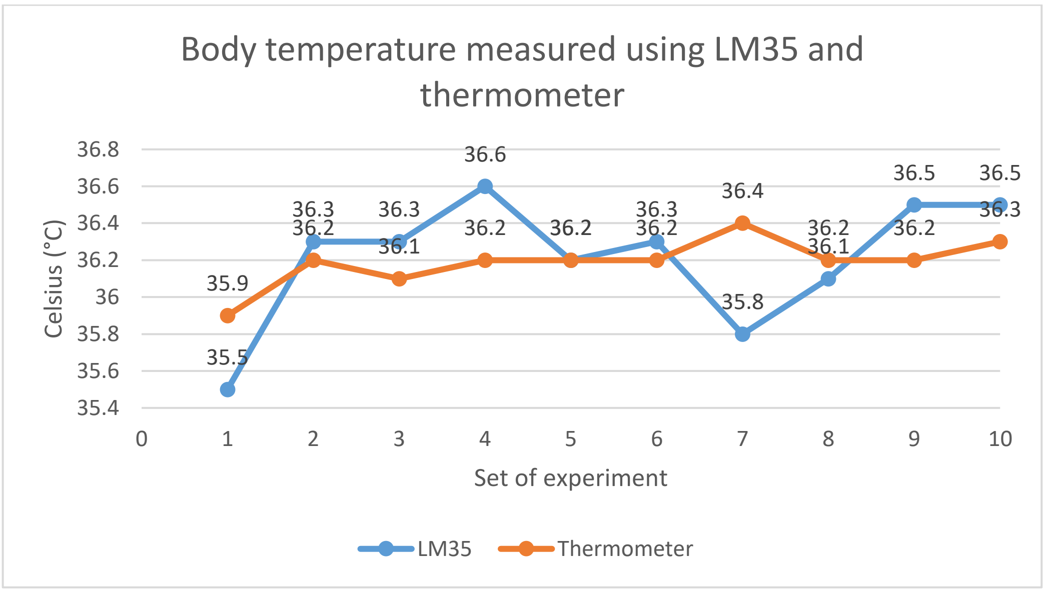


Fig.4.23: Comparison of average temperature measured using LM35 and thermometer

The body temperature measured by a thermometer is stable, while the body temperature measured by the LM35 shows some fluctuation in the graph shown in Fig.4.23. The inaccuracy of the LM35 mentioned in section 4.2.2 causes the fluctuation in the body temperature reading. The temperature readings are based on the voltage output of the LM35. The temperature reading obtained will fluctuate if there is fluctuation in the output voltage of the LM35. As a result, false alarms may be triggered from time to time due to the inaccuracy of the body temperature sensor. Thus, a better temperature sensor with IoT capabilities can be used in the future to improve the reliability of this system. From Fig.4.23, the margin error of the body temperature measured using LM35 is around ± 3 °C, with a percentage error of 7.69%.

4.3.4 SEN-11574 pulse sensor experiment

The heart rate is measured in beats per minute (BPM) in this project. BPM of the user is measured by using the SEN-11574 pulse sensor. The pulse sensor can be clipped on the user's fingertips or earlobe.

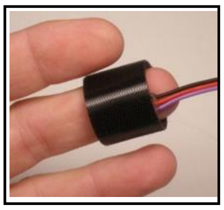


Fig.4.24: Pulse Sensor clipped on the fingertip

The SEN-11574 pulse sensor was clipped on the fingertip of the test subject during the experiment, as shown in Fig.4.24. The resting BPM of the test subject was measured. The measuring process was repeated three times, and the average of the three data is calculated. Ten sets of readings were taken. The obtained results are as shown in Table 4.5.

Table 4.5: Resting heart rate measured using the pulse sensor

No.	Resting heart rate (Beats Per Minute)			Average
1	66	65	66	65.7
2	67	66	65	66.0
3	68	67	65	66.7
4	67	65	66	66.0
5	66	67	66	66.3
6	66	68	65	66.3
7	65	64	66	65.0
8	65	66	67	66.0
9	66	68	65	66.3
10	66	67	66	66.3
Total Average				66.1

The measuring process was repeated on the same subject for active heart measurement. BPM of the test subject after performing twenty push-ups was measured. Each set of readings was taken with ten minutes resting interval. The results obtained are as shown in Table 4.6.

Table 4.6: Active heart rate after twenty push-ups measured using the pulse sensor

No.	Active heart rate after 20 push-ups (Beats Per Minute)			Average
1	115	115	115	115.0
2	114	114	115	114.3
3	113	113	114	113.3
4	113	114	115	114.0
5	113	114	113	113.3
6	112	113	113	112.7
7	113	113	114	113.3
8	113	113	113	113.0
9	112	114	114	113.3
10	113	114	115	114.0
Total Average				113.6

The heart rate of the test subject shows a significant increase after performing twenty push-ups. The results obtained in Table 4.5 and Table 4.6 shows that the SEN-11574 heart rate sensor is capable of detecting the rise of the heart rate. Small fluctuations in the BPM obtained was caused by the pressure applied by the fingertip on the pulse sensor. The pulse sensor uses the LED light and photosensor to detect the variation of the blood flow in the fingertip to measure the BPM. Pressure applied by the fingertip onto the pulse sensor will affect the blood flow and causes the fluctuation in the BPM measurement. The fluctuation obtained in Table 4.5 and Table 4.6 was around ± 2 beats.

4.3.5 Accuracy of SEN-11574 Pulse Sensor

The accuracy of the pulse sensor was determined by comparison with the manual pulse rate measurement. The manual measurement was performed by pressing index and middle fingers on the opposite wrist, as shown in Fig.4.27. The BPM of the pulse sensor was compared to the BPM measured from the wrist.

Fig.4.27: BPM measured from the wrist

The BPM measurement process was repeated three times, and the average of the values was calculated. Ten sets of readings were taken. The experiment was done in both active and resting conditions. The obtained results are shown in Table 4.7 and Table 4.8.

Table 4.7: Resting heart rate measured from the wrist

No.	Resting heart rate (Beats Per Minute)			Average
1	67	66	64	65.7
2	66	65	66	65.7
3	66	66	65	65.7
4	65	65	64	64.7
5	64	66	65	65.0
6	64	66	65	65.0
7	65	66	65	65.3
8	66	67	65	66.0
9	64	66	65	65.0
10	66	67	66	66.3
Total Average				65.44

The measuring process was repeated on the same subject for active heart rate measurement. BPM of the test subject after performing twenty push-ups was measured. Each set of readings was taken with ten minutes resting interval. The results obtained are as shown in Table 4.8.

Table 4.8: Active heart rate after performing 20 push-ups measured from the wrist

No.	Active heart rate after 20 push-ups (Beats Per Minute)			Average
1	115	114	114	114.3
2	115	115	113	114.3
3	113	114	115	114.0
4	114	115	113	114.0
5	114	115	114	114.3
6	115	115	113	114.3
7	113	114	114	113.7
8	114	115	113	114.0
9	115	116	115	115.3
10	114	113	113	113.3
Total Average				114.2

A comparison between Table 4.5 and Table 4.7 is plotted in a graph as shown in Fig.4.27

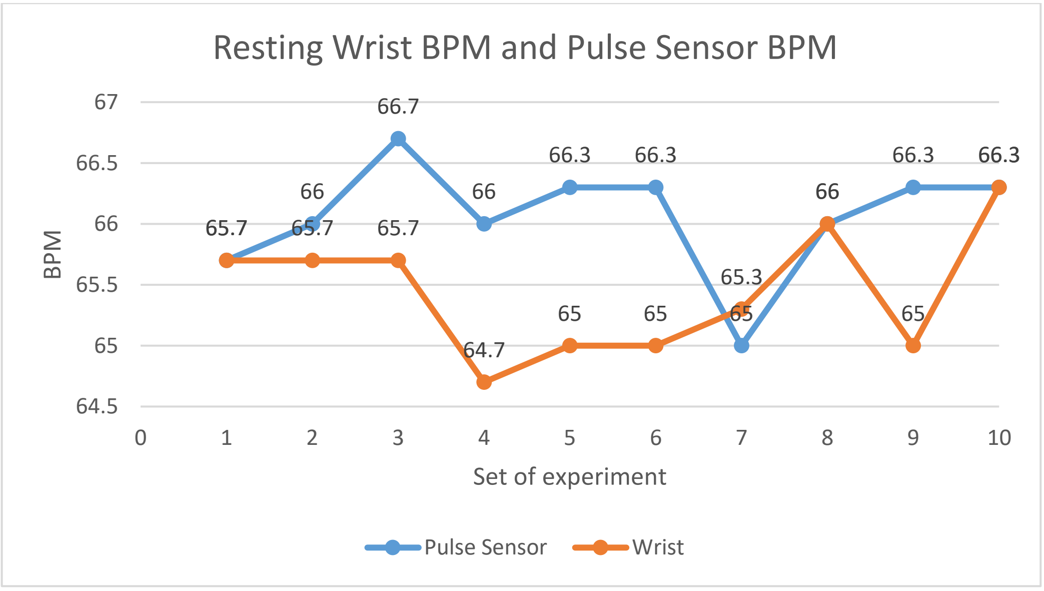


Fig.4.28: Comparison between resting wrist BPM and pulse sensor BPM

There were ±2 beats fluctuation in the measured pulse sensor and wrist BPM, as shown in Fig.4.28. Any movement of the test subject will affect the BPM and led to the fluctuations. The results of the experiment show that the SEN-11574 pulse sensor is capable of measuring the heart rate with high accuracy since the fluctuations obtained were small and insignificant.

A comparison between Table 4.6 and Table 4.8 is plotted in a graph, as shown in Fig.2.29.

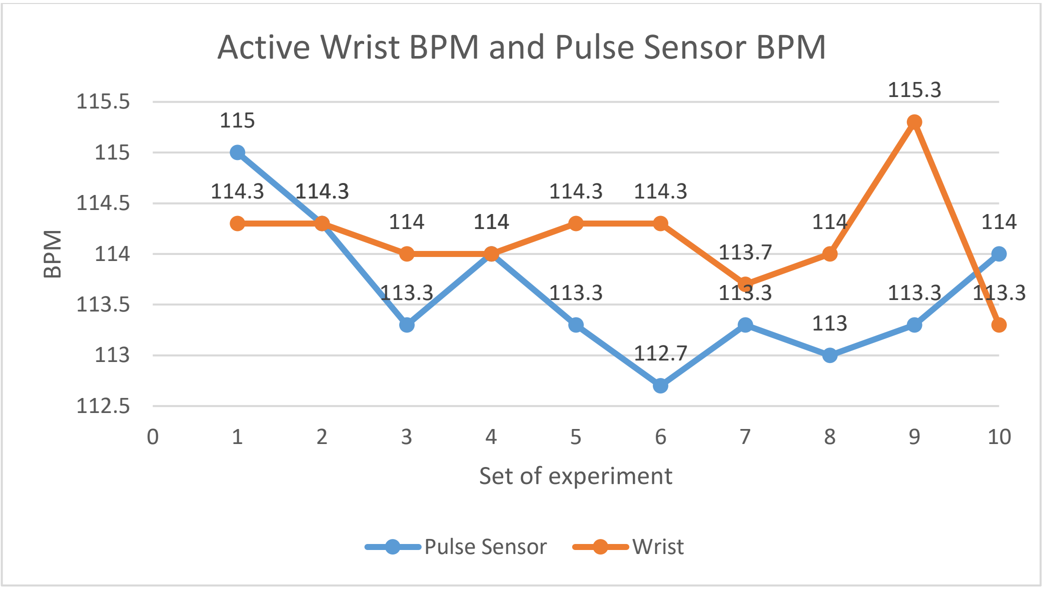


Fig.4.29: Comparison between active wrist BPM and pulse sensor BPM

4.4 Schematic diagram of the proposed system

Fig.4.30 shows the schematic diagram of the proposed health system. The diagram consists of the Arduino UNO, ESP8266 NodeMCU gateway, LM35 temperature sensor, and SEN-11574 pulse sensor. Pin D5 and D6 of both Arduino UNO and ESP8266 NodeMCU are connected for serial communication between each other. The output of the LM35 temperature sensor will be sent to the ESP8266 NodeMCU via serial communication. The pulse sensor is connected to 3.3V, ground, and A0 ADC pin of Arduino UNO. The LM35 temperature sensor is connected to 3.3V, ground, and A0 ADC pin of ESP8266 NodeMCU.

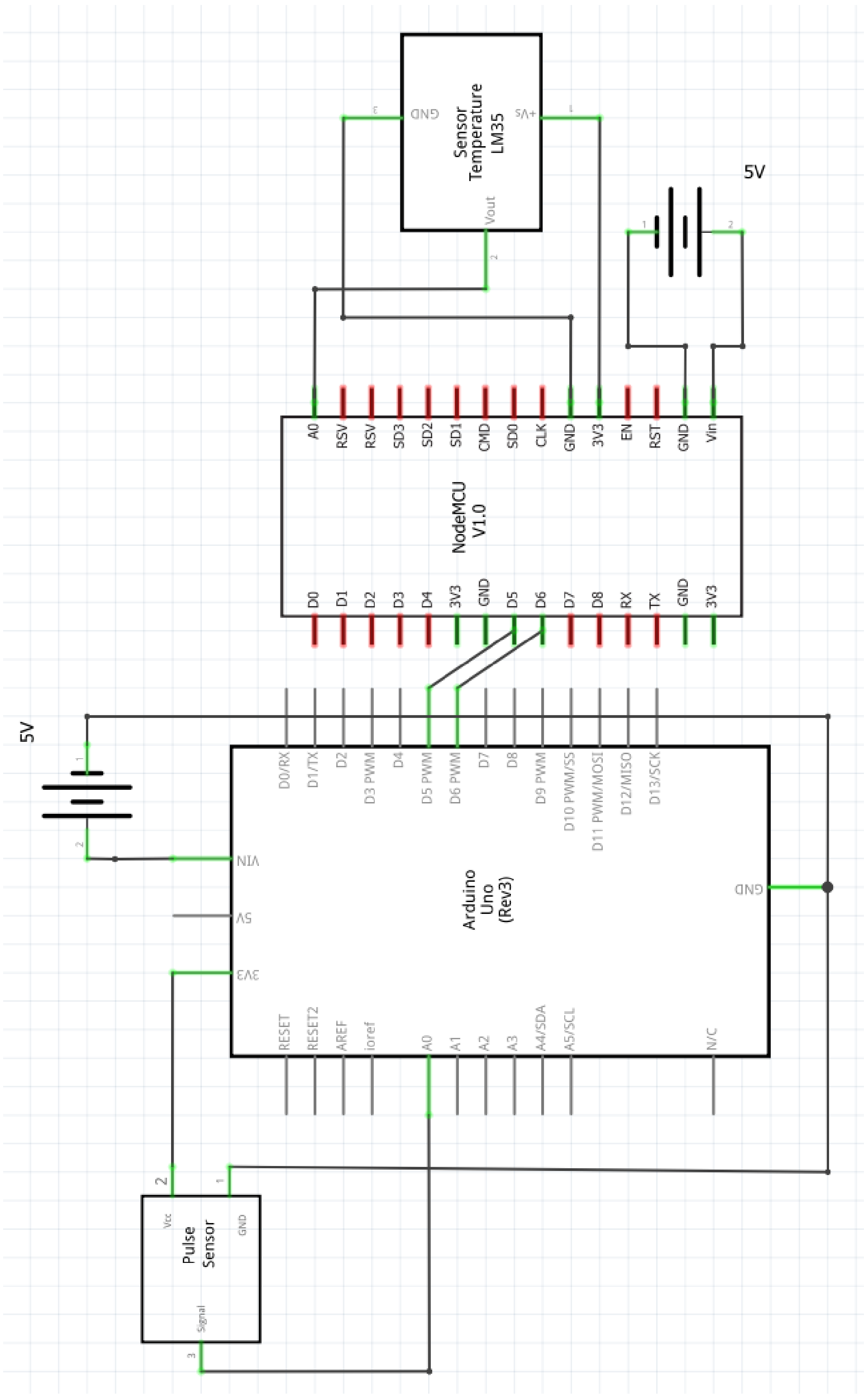


Fig.4.30: Schematic diagram of the proposed system

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusion

The primary objective of this project is to develop an IoT based health monitoring system that able to provide real-time health data monitoring of the user. The proposed system is able to track the user's location, send the health data to the visualization platform in real-time, and sends alarm notifications to phones and emails. The location of the user is tracked by using geolocation service while the real-time health data visualization is displayed on the Thingsboard. The coordinate of the user location is very useful to track down the user during an emergency. User's relatives or authorized users are able to view the health data of the user through the Thingsboard. The alarm notifications can be sent to any related personnel when there are sudden changes in heart rate and abnormal body temperature. The health data are stored in a local and cloud database, which allow the user to keep track of their health condition by tracking the health data. Experiments are conducted to prove the functionality of the sensors and the system.

5.2 Recommendation for Future Work

The LM35 temperature sensor should be replaced with much reliable and accurate temperature sensors. The temperature output of the LM35 temperature sensor is not reliable since the outputs of the sensor fluctuates a lot during the experiment. As a result, false alarms may be triggered from time to time which reduces the reliability and efficiency of the health monitoring system.

Besides, other medical sensors such as ECG sensors and blood pressure sensors can be added into the system to improve the functionality of the system. The users will be able to track their health conditions better if the system is capable of tracking more other health data accurately.

REFERENCES

1. European Commission Information Society, Internet of things in 2020: a roadmap for the future, 2008. http://www.iot-visitthefuture.eu.
2. Rolim, C. O., Koch, F. L., Westphall, C. B., Werner, J., Fracalossi, A., & Salvador, G. S. (2010). *A Cloud Computing Solution for Patient's Data Collection in Health Care Institutions. 2010 Second International Conference on eHealth, Telemedicine, and Social Medicine.*
3. Doukas, C., & Maglogiannis, I. (2012). *Bringing IoT and Cloud Computing towards Pervasive Healthcare. 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing.*doi:10.1109/imis.2012.26
4. Chen, M., Li, W., Hao, Y., Qian, Y., & Humar, I. (2018). *Edge cognitive computing based smart healthcare system. Future Generation Computer Systems, 86, 403–411.*doi:10.1016/j.future.2018.03.054
5. Naik, N. (2017). *Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. 2017 IEEE International Systems Engineering Symposium (ISSE).*doi:10.1109/syseng.2017.8088251
6. Gezer, V., et al. (2018). "An introduction to edge computing and a real-time capable server architecture." Int. J. Adv. Intell. Syst.(IARIA) **11**(7): 105-114.
7. Hayes, B. (2008). "Cloud computing." Communications of the ACM **51**(7): 9-11.
8. Hou, X., et al. (2016). "Vehicular fog computing: A viewpoint of vehicles as the infrastructures." IEEE Transactions on Vehicular Technology **65**(6): 3860-3873.
9. JoSEP, A. D., et al. (2010). "A view of cloud computing." Communications of the ACM **53**(4).
10. Knorr, E. and G. Gruman (2008). "What cloud computing really means." InfoWorld **7**: 20-20.
11. Mell, P. and T. Grance (2011). "The NIST definition of cloud computing."

12. Rahmani, A. M., et al. (2018). "Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach." Future Generation Computer Systems **78**: 641-658.

13. Rausch, T., et al. (2018). Emma: Distributed qos-aware mqtt middleware for edge computing applications. 2018 IEEE International Conference on Cloud Engineering (IC2E), IEEE.

14. Shi, W., et al. (2016). "Edge computing: Vision and challenges." IEEE Internet of Things Journal **3**(5): 637-646.

15. Tao, F., et al. (2014). "CCIoT-CMfg: cloud computing and internet of things-based cloud manufacturing service system." IEEE Transactions on Industrial Informatics **10**(2): 1435-1442.

16. Vaquero, L. M., et al. (2008). "A break in the clouds: towards a cloud definition." ACM SIGCOMM Computer Communication Review **39**(1): 50-55.

17. Athanasiou, L. S., et al. (2017). Atherosclerotic Plaque Characterization Methods Based on Coronary Imaging, Academic Press.

18. Dictionary, o. (2016). *occupation Meaning in the Cambridge English Dictionary*. [online] Dictionary.cambridge.org. Available at: http://dictionary.cambridge.org/dictionary/english/occupation [Accessed 20 Jan. 2020].

19. Jamaludin, N., et al. (2015). "Thermal comfort of residential building in Malaysia at different micro-climates." Procedia-Social and Behavioral Sciences **170**: 613-623.

20. Sund-Levander, M., et al. (2002). "Normal oral, rectal, tympanic and axillary body temperature in adult men and women: a systematic literature review." Scandinavian journal of caring sciences **16**(2): 122-128.

21. Yi, S., et al. (2015). Fog computing: Platform and applications. 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb), IEEE.

22. Finley, J. P., et al. (1987). "Heart-rate variability in children. Spectral analysis of developmental changes between 5 and 24 years." Canadian journal of physiology and pharmacology **65**(10): 2048-2052.

23. Mackowiak, P. A., et al. (1992). "A critical appraisal of 98.6 F, the upper limit of the normal body temperature, and other legacies of Carl Reinhold August Wunderlich." Jama **268**(12): 1578-1580.

24. ThingsBoard. n.d. *Thingsboard Architecture*. [online] Available at: <https://thingsboard.io/docs/reference/architecture/> [Accessed 17 March 2020].

25. Google Cloud. 2020. *Cloud SQL: Relational Database Service | Google Cloud*. [online] Available at: <https://cloud.google.com/sql> [Accessed 17 March 2020].

26. Docs.oasis-open.org. 2020. *MQTT Version 3.1.1*. [online] Available at: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html> [Accessed 19 March 2020].

27. DELISLE, M. 2009. *Mastering phpMyAdmin 3.1 for effective MySQL management*, Packt Publishing Ltd.

28. WANG, C., CHOW, S. S., WANG, Q., REN, K. & LOU, W. 2011. Privacy-preserving public auditing for secure cloud storage. *IEEE transactions on computers,* 62**,** 362-375.

29. BANSAL, G. & GEFEN, D. 2010. The impact of personal dispositions on information sensitivity, privacy concern and trust in disclosing health information online. *Decision support systems,* 49**,** 138-150.

30. ROMERO, N. L. 2012. "Cloud computing" in library automation: benefits and drawbacks. *The Bottom Line*.