الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

MASTER'S DISSERATION

University Stream: Electronics
Speciality: Electronics of Embedded Systems

Presented by

BEKHTI Ramy

&

MANSEUR Mohamed Rafik

Implementation of the MQTT protocol for the development an IoT Based Patient Health Monitoring System.

Proposed by : Bougherira Nadia

College Year 2022-2023

Acknowledgments

- First and foremost, we would like to thank our supervisor Dr. Bougherira Nadia for her support, guidance, and patience throughout our researchjourney. We thank her for all the invaluable suggestions she offered andfor providing us with the necessary tools to succeed. May the AlmightyAllah reward her abundantly for her kindness and hard work.

- We would also like to acknowledge and express our appreciation for themembers of the board of examiners: Dr. Djendi Mohamed and Dr. Naceur Djamila May Allah reward them for their contributions.

Dedication

Bekhti Ramy

- I would like to dedicate my work to my small family: my mom, my dad and my big brother who have supported me throughout this year.

- To my big family that I have been living with for the past two years: Rayan H, Haithem D, Oussama T and finally to my best friend Mouad Issahin.

- To my research partner who has helped me a lot throughout this year.

Dedication

Manseur Mohamed Rafik

I would like to dedicate this work to my beloved and supportive family especially to my mother, who through her guidance and aid, this work has flourished. To my friends that have helped me through my mental struggles and kept me motivated with their positive feedback, I'd also like to extend this dedication to my research partner for giving me the great opportunity to work with him this year.

ملخص:

تتزايد شعبية إنترنت الأشياء بسرعة في مجال أنظمة المراقبة عن بعد. للاستفادة من هذه التكنولوجيا، نقترح في هذا في نظام مراقبة الصحة عن بعد الحالي الذي يكتشف تلقائيًا معدل ضربات القلب MQTT المشروع تنفيذ بروتوكول الاتصال وتشبع الأكسجين ودرجة حرارة الجسم، بالإضافة إلى درجات الحرارة والرطوبة في الغرفة حيث يتم نقل المريض إلى MAX30100 DS18B20 المستشفى على أمل تحسين هذا النظام. لجمع حالات المرضى، سنستخدم أجهزة الاستشعار يتم بعد ذلك نشر البيانات التي تم الحصول عليها عبر بروتوكول النشر/الاشتراك .ESp-32 والسيطرة الدقيقة DHT-11 التي يتم تخزينها وإتاحتها للوصول إلى المشتركين، وتقديم MYSQL وقاعدة بيانات mqtt على وسيط البعوض MQTT .الذي طورناه Android إشعار دفع على الفور للأطباء/الممرضات المسؤولين على واجهة الويب الكاملة و تطبيق

كلمات المفاتيح: إنترنت الأشياء، شبكة الإنترنت، مراقبة الصحة تطبيقاتAndroid،ESP-32 ، بروكر MQTT

Résumé : La popularité de l'Internet des objets augmente rapidement dans le domaine des systèmes de surveillance à distance. Pour tirer parti de cette technologie, nous proposons dans ce projet une mise en œuvre du protocole de communication MQTT dans un système de surveillance de la santé à distance déjà existant qui détecte automatiquement la fréquence cardiaque, la saturation en oxygène et la température corporelle du patient ainsi que les températures et l'humidité de la pièce où le patient est hospitalisé dans l'espoir d'améliorer ce système. Pour recueillir les conditions des patients, nous allons utiliser les capteurs MAX30100, DS18B20 et DHT-11 et un microcontrôleur ESp-32. Les données acquises sont ensuite publiées via le protocole de publication / souscription MQTT sur le broker mosquitto mqtt et une base de données MYSQL qui est ensuite stockée et disponible pour l'accès par les abonnés, et fournissent instantanément une notification push aux médecins / infirmières en charge sur notre interface web complète et une application Android que nous avons développée.

Mots clés : Internet des objets, ESP-32, Android App, Sensors, Web Server, Surveillance de la santé, Broker MQTT

Abstract: The popularity of Internet of Things is rapidly increasing in the area of remote monitoring systems. To take advantage of this technology we are proposing in this project an implementation of the MQTT communication protocol in an already existing remote health monitoring system that automatically detects the patient s heart rate, oxygen saturation and body temperature as well as the temperature and humidity of the room the patient is hospitalized in the hopes of improving this system .

To gather the patient s conditions we are going to use the MAX30100, DS18B20 and DHT-11 sensors and an ESp-32 microcontroller. The acquired data is then published via the MQTT publish/subscribe protocol on to the MQTT mosquito broker and a MYSQL database which is then stored and available for access by the subscribers, and instantly provide push notification to the doctors/nurses in charge on our full stack web interface and an android application that we developed.

Keywords : Internet Of Things, ESP-32, Android App, Sensors , Web Server, Health Monitoring, MQTT broker

List of Acronyms and Abbreviations

A	**ABG**	Arterial Blood Gas
	ADC	Analog-to-Digital Converter
	ALC	Automatic Level Control
	AMBER	America's Missing Broadcast Emergency Response
	AMQP	Advanced Message Queuing Protocol
	API	Application Programming Interface
B	**BPM**	Beats per Minute
C	**CGI**	Computer Generated Imagery
	COAP	Constrained Application Protocol
	COPD	Chronic Obstructive Pulmonary Disease
	CPU	Central Processing Unit
	CSS	Cascading Style Sheets

D	**DAC**	Digital-to-Analog Converter
	DDS	Data Distribution Service
E	**ECG**	Electrocardiogram
	EMR	Electronic Medical Record
F	**FTDI**	Future Technology Devices International
G	**GND**	Ground
	GPS	Global Positioning System
H	**Hb**	Hemoglobin
	HbO2	Oxygenated Hemoglobin
	HTTP	Hypertext Transfer Protocol
	HTML	Hypertext Markup Language
I	**I2C**	Inter-Integrated Circuit
	IDE	Integrated Development Environment
	ID	Identification
	IEEE	the Institute of Electrical and Electronics Engineers
	IFP	Interface Profile
	IOS	iPhone Operating System
	IOT	Internet of Things
	IIOT	Industrial Internet of Things
	IR	Infrared

	ISO	International Organization for Standardization
	IT	Information Technology
K	**KV**	Kivy
L	**LoRaWAN**	Long Range Wide Area Network
M	**M2M**	Machine-to-Machine
	MacOS	Macintosh Operating System
	MCU	Microcontroller Unit
	MQTT	Message Queuing Telemetry Transport
N	**NFC**	Near Field Communication
O	**O2sat**	Oxygen Saturation
	OTP	One-Time Programmable
P	**PHP**	Hypertext Preprocessor
	PHP-GTK	Hypertext Preprocessor GIMP Tool Kit
Q	**QOS**	Quality of Service
R	**RAM**	Random Access Memory
	RFID	Radio Frequency Identification
S	**SaO2**	Arterial Oxygen Saturation
	SpO2	Peripheral Oxygen Saturation
	SOA	Service-Oriented Architecture
	SPI	Serial Peripheral Interface

	SQL	Structured Query Language
	Sub/pub	Subscribe/Publish
T	**TCP/IP**	Transmission Control Protocol/Internet Protocol
U	**UART**	Universal Asynchronous Receiver-Transmitter
	UPnP	Universal Plug and Play
	UUID	Universally Unique Identifier
V	**VS**	Visual Studio
	VDD	Voltage Drain to Drain
W	**WLAN**	Wireless Local Area Network
	WMAN	Wireless Metropolitan Area Network
	WPAN	Wireless Personal Area Network
	WSN	Wireless Sensor Network
	Wi-Fi	Wireless Fidelity
X	**XHTML**	Extensible HyperText Mark up Language

Table of Contents

List of Figures:

List of Tables

General Introduction

The global population is experiencing a significant demographic shift towards an aging society, data from the 'Center for Disease Control's Morbidity and Mortality Weekly Report' indicates a doubling of the population in this age group over a span of 30 years. Looking ahead, it is projected that the proportion of individuals aged 65 years and older will continue to rise in various regions. By 2030, the percentage of this population group is estimated to increase from 12.4% to 19.6% in the USA, from 12.6% to 20.3% in Europe, from 6% to 12% in Asia, from 5.5% to 11.6% in Latin America and the Caribbean, and from 2.9% to 3.7% in Africa. Aging is associated with an increased prevalence of systemic diseases (notably arthritis, cardiovascular diseases, diabetes and stroke). These conditions, coupled with multiple medications like sedatives, are associated with a decline in physiological function (lack of manual dexterity and being prone to injuries from falling) making it harder for older individuals to go to hospitals at any given moment, especially individuals that live in isolated areas.[1]

The shifting demographics mentioned above give rise to a fundamental question: how do we care for the elderly and those in areas with reduced access to providers?

Addressing this question will involve complex changes in healthcare organization and payment systems. However, a potential part of the solution lies in leveraging recent advancements in information technology and related fields. There are currently promising and cost-efficient technologies that have the potential to enhance the capabilities of the healthcare system, expanding its reach into communities, improving diagnostics and monitoring, and promoting engagement among individuals by giving them access to said technologies. In this thesis, we will

thoroughly explore these technologies, with a specific focus on remote monitoring systems utilizing wearable technology. We have chosen to concentrate on these technologies due to significant advancements in wearable sensor systems, which have led to breakthroughs in clinical applications. Wearable sensors offer both diagnostic and monitoring capabilities, making them valuable in the diagnosis and ongoing treatment of various cardiovascular and pulmonary conditions such as hypertension and asthma.[2]

For a continuous monitoring of patients' vital signs, even without their physical presence at a healthcare facility, the deployment of systems equipped with sensors becomes essential, using wireless communication, they can transmit a patient's data to a mobile phone or an access point and relay the information via the Internet.

The concept of the Internet of Things (IoT) plays a significant role in enabling such remote monitoring systems, it represents a network of physical devices that are embedded and constantly gathering and exchanging information, these devices encompass various aspects, such as monitoring blood pressure, electrocardiogram (ECG), and other environmental information, including humidity and temperature, to provide comprehensive and holistic data about the patient's health status. The use of IoT technology enables robust and interactive communication between patients and specialists in remote settings, paving the way for substantial advancements in the industry.[3]

We will present our work in three chapters:

Chapter 1: will talk about the significance of technology in health monitoring, the anatomy and operation of the heart, and the utility of pulse oximetry.

Chapter 2: is dedicated to defining the Internet of Things, its different fields of application, its different architectures and some protocols ensuring its functionality.

Chapter 3: will lastly present our work environment, while also showing steps that were taken to realize our project.

Followed by a general conclusion at the end to draw our thesis to a close.

Chapter 1 Health Monitoring.

1.1 Introduction:

Medical machinery and monitors are superior to health workers because they can gather and process significant amounts of data with speed and precision. Unlike humans, medical machines are not affected by conditions such as fatigue, stress, or emotions, which can lead to variability and errors in their assessments. Moreover, modern medical devices are typically equipped with advanced algorithms and artificial intelligence, which can detect patterns and anomalies that may go unnoticed by human observation. This advanced technology makes it possible for machinery to provide an objective and reliable evaluation of a patient's health status, empowering healthcare providers to make better-informed decisions regarding diagnosis and treatment.

This chapter aims to delve into the significance of technology in health monitoring, the anatomy and operation of the heart, and the utility of pulse oximetry as a tool for monitoring heart health. This chapter's ultimate objective is to provide readers with a comprehensive understanding of pulse oximetry and its role in heart health monitoring.

1.2 The History of Health Monitoring:

In 1625, Santorio of Venice, aided by his friend Galileo, introduced the use of a spirit thermometer to measure body temperature and a pendulum to time pulse rate. However, their discovery was overlooked until Sir John Floyer published "Pulse-Watch" in 1707, which was the first scientific report on pulse rate. Later on, Ludwig Taube plotted the course of fever in a patient and included respiratory rate as a vital sign. The thermometer and clock advancements cemented the monitoring of heart rate,

respiratory rate, and body temperature as standard vital signs. The introduction of the sphygmomanometer in 1896 added arterial blood pressure as the fourth vital sign. In 1903, Willem Einthoven invented the string galvanometer for measuring electrocardiography (ECG), which earned him the 1924 Nobel Prize in physiology. During the 1980s, patient monitoring systems evolved to include bedside arrhythmia analysis. Advancements in display technologies throughout the 1990s and early 2000s made patient monitoring systems more accessible and easier to use.

Nowadays the introduction of portable patient monitoring systems has made it so that it is easier to transport equipment to patients within a facility and enabled doctors and nurses to monitor and report on patient vitals effortlessly.[4]

1.3 Technology in the Field of Healthcare:

The emergence of technology has brought a significant change in the way health monitoring is done, enabling healthcare organizations to keep track of patients' health remotely using various devices such as wearable technology, sensors, and mobile apps.

The recorded data can be analyzed in real-time, allowing healthcare professionals to identify potential health risks or issues proactively. Health monitoring plays a crucial role in the prevention, diagnosis, and management of chronic diseases. By analyzing patients' health patterns using algorithms capable of detecting the early stages of health problems, technology can potentially save lives and enhance health outcomes.

1.3.1 Telemedicine:

Telemedicine refers to the use of advanced combinations of telecommunication technology and information technology to deliver clinical healthcare services from a remote location. This technology presents a promising solution to overcome distance barriers and improve access to medical care for geographically isolated rural communities, as over half of the global rural population

lacks access to healthcare services. Telemedicine applications offer an effective means to provide timely medical care, reduce the likelihood of disease progression, and lower consultation costs, making healthcare more accessible to everyone. The increasing adoption of telemedicine technology solutions is expected to improve health outcomes and enhance recovery from sickness.

This technology is particularly beneficial for rural or inaccessible regions where distance is a significant obstacle to accessing healthcare services. Telemedicine technology solutions can be classified based on their applications, and this chapter discusses the latest research and market developments related to these technologies. It also presents cutting-edge research findings on the development and deployment of these technologies and provides an in-depth analysis of currently open issues for future research and development.

Telemedicine encompasses three primary modes of delivery:

- Save and Forward.
- Tele-meeting.
- Video-conferencing.

a. Save and Forward:

It involves the use of Electronic Medical Records (EMR) software to store a patient's medical history and diagnosis report in a file that can be sent to a medical doctor for advice. This method can help to alleviate waiting times and travel expenses associated with in-person consultations. However, it is not suitable for urgent health problems that require immediate treatment.

b. Tele-meeting:

It involves discussions between different parties about a patient's health using audio communication via the Internet. This method is useful for patient-doctor discussions about medical treatment.

c. Video-conferencing:

This model is the most beneficial and appropriate method for long-distance medical discussions, as it utilizes both audio and video. However, this method requires high bandwidth, and the cost of equipment can be high, particularly when using Radio Frequency (RF) components for audio and video transmission.[5]

1.3.2 Telepresence:

Telepresence can enable a person to appear to be present, to feel as though they were present, or to have an impact remotely via tele robotics at a location other than their actual location, while Telemedicine offers an effective method of passively performing surgery from a different location. In other words, Telemedicine enables doctors and surgeons to successfully perform surgeries from a distance instead of just mentoring another surgeon. In this case, the surgeon can operate on the patient "virtually" with this kind of procedure. This is accomplished by a surgeon console and a remote surgical unit, which consists of two computer-controlled robot arms (as seen on figure 1.1). [6]

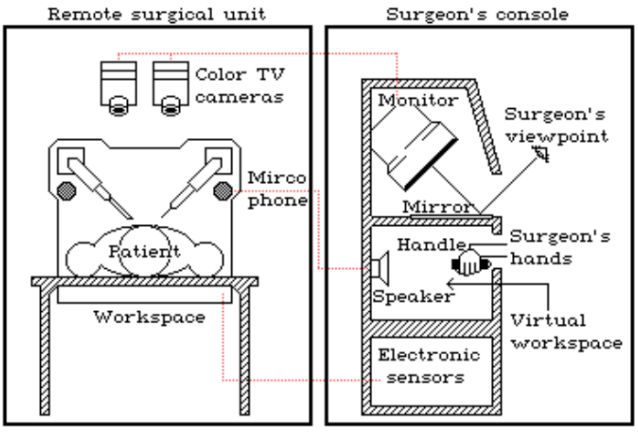

Figure 1. 1 Schematic Representation of the Telepresence Surgery Setup. [6]

1.3.3 Difference between Telepresence and Telemedicine:

While telemedicine uses records and communication to help share knowledge between doctors and monitor patients remotely, telepresence uses tele robotics to simulate medical procedures virtually in real time (figure 1.2 shows the different telehealth applications while highlighting the modes they're done with).

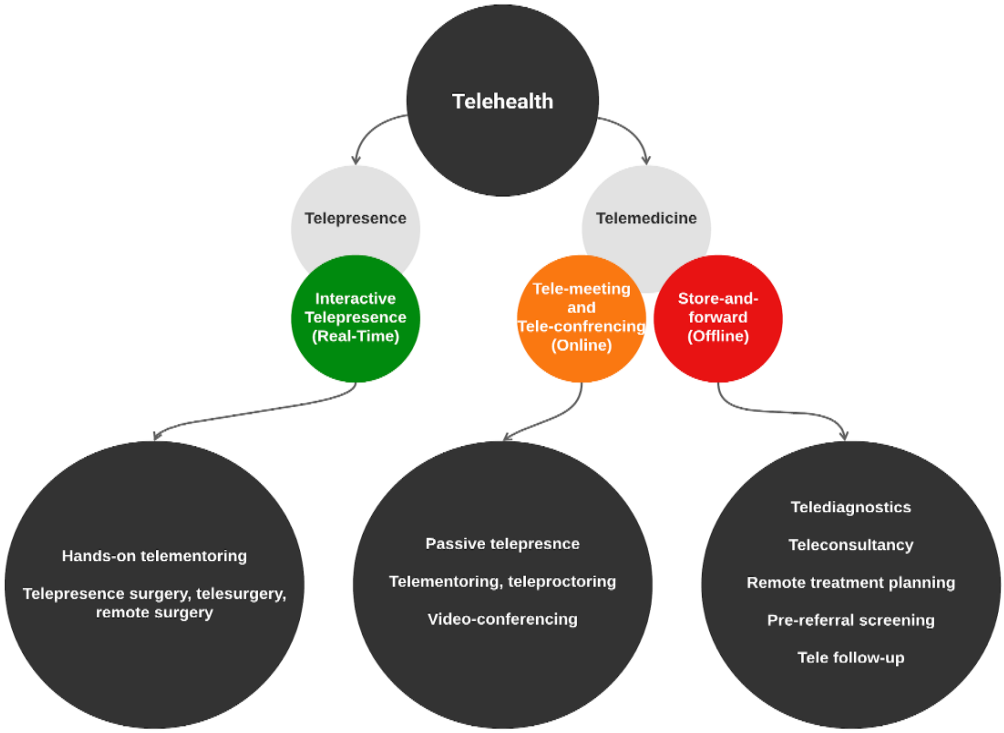

Figure 1. 2 Different Modes of Telehealth (Made with Lucid chart). [6]

1.3.4 IoT in Healthcare:

The Internet of Things (IoT) can enable users in the medical industry to gather enormous amounts of data from numerous embedded systems simultaneously to use them in different applications (see figure 1.3). Each system is specific to a single patient and is equipped with a data acquisition unit, a microcontroller, software, and sensors, some of which are wearable. The system can then transmit the data to the IoT cloud, where it can be saved and retransmitted to other connected devices. Patients'

families may also profit from the system's restricted access while doctors and nurses can examine patients remotely without having to visit them personally.[7]

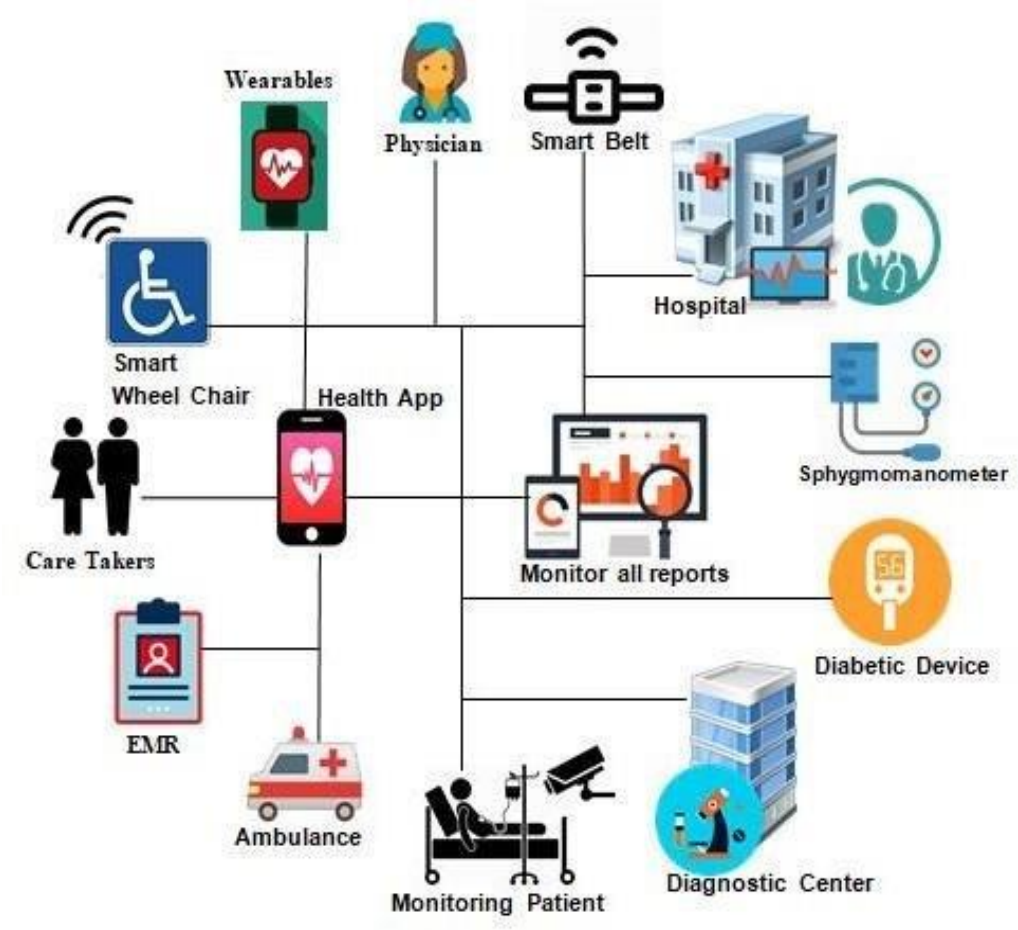

Figure 1. 3 IoMT Devices and Application Examples. [8]

1.4 The Heart:

Among the various factors that contribute to health monitoring, one significant aspect is keeping track of the heart's functionality. The heart serves as a crucial organ responsible for circulating blood throughout the body, which supplies oxygen and essential nutrients to all organs and tissues. Disruptions in the normal functioning of the heart can give rise to grave health complications.

The heart is a significant conical muscular organ situated in the central mediastinum. Its main job is to pump blood to the body's various organs so that it can get the nutrients it needs. It is positioned obliquely beneath the sternum and adjacent portions of the costal cartilage. The atria and ventricles of the heart are connected by a

downward, forward, and leftward blood flow. The heart typically measures around 12 x 9 cm and weighs approximately 300 g in males and 250 g in females.[9]

1.4.1 Heartrate:

Heart rate refers to the number of times the heart contracts within a given period, usually a minute (BPM), in the field of medicine. The pulse can be detected at various locations on the body where an artery is close to the skin, including the wrist, side of the neck, back of the knees, top of the foot, groin, and other areas. A healthy adult's resting heart rate typically ranges between 60 and 100 beats per minute. Measuring the heart rate is a vital tool for assessing an individual's health and it is referred to as the pulse.[10]

The heart rate is classified according to whether it is low, normal, or high:

- Bradycardia: Slow heart rate.
- Normo-Frequent: Normal heart rate.
- Tachycardia: High heart rate.

Numerous variables, such as age and physical condition, affect heart rate. The heart rate while at rest, often known as the "resting pulse" or "resting heart rate," provides details on the health of the heart.

The tables on the following page (table 1.1 and table 1.2) display the average resting pulse values based on gender, age, and physical condition.[11]

Males (Age)	18-25	26-35	36-45	46-55	56-65	65+
Athlete	49-55	49-54	50-56	50-57	51-56	50-55
Excellent	56-61	55-61	57-62	58-63	57-61	56-61
good	62-65	62-65	63-66	64-67	62-67	62-65
Above Average	66-69	66-70	67-70	68-71	68-71	66-69
Average	70-73	71-74	71-75	71-76	72-75	70-73
Below average	74-81	75-81	76-82	77-83	76-81	74-79
Bad	82+	82+	83+	84+	82+	80+

Table 1.1: Resting Pulse in Men in the Correlation of Age and Physical Condition.[11]

Females (Age)	18-25	26-35	36-45	46-55	56-65	65+
Athlete	56-60	54-59	54-59	54-60	54-59	54-59
Excellent	61-65	60-64	60-64	61-65	60-64	60-64
good	66-69	65-68	66-69	66-69	65-68	65-68
Above Average	70-73	69-72	70-73	70-73	69-73	69-72
Average	74-78	73-76	74-78	74-77	74-77	73-76
Below average	79-84	77-82	79-84	78-83	78-83	77-84
Bad	85+	83+	85+	84+	84+	84+

Table 1.2: Resting Pulse in Women in the Correlation of Age and Physical Condition.[11]

1.5 Body Temperature:

The heat that is generated by our bodies determines the body's temperature. We burn carbohydrates and fat even when we are at rest to keep our important organs functioning (especially the brain). Muscle contraction with exercise has two different effects: it increases the person's metabolism and generates body heat.

In addition to the above points, we can also get heat from other sources like the sun or even a warm bath for example. Another point to consider is that since our metabolism produces heat, we maintain a constant temperature of 37 degrees, which can decrease by the same amount on average every 24 hours.

However, there is a chance of heat loss as a consequence of being exposed to a cold environment. In this case, the temperature of the nucleus drops as a result, which is known as hypothermia (Defined as a core temperature of 35 degrees or less). When hypothermia first develops, there is a sensation of cold and shivers (Involuntary contraction of the muscles to increase metabolism and thereby heat production). As hypothermia worsens, various bodily processes slow down, shivers stop, and awareness is lost.

When the core temperature drops below 25–20 degrees, deep hypothermia is usually fatal. Therefore, a core temperature of 36 degrees does not yet imply hypothermia and does not provide a threat. Depending on how it was measured, it can represent a typical circumstance. In actuality, the temperature of the core is underestimated by a measurement taken under an axle. It is preferable to take your temperature at the rectal level or under your tongue.[12]

1.6 The Arterial Pressure:

Arterial pressure, also known as systemic arterial pressure or blood pressure, refers to the pressure within large arteries in the systemic circulation. It is measured in millimeters of mercury and expressed in terms of the systolic pressure over diastolic pressure, which are the maximum and minimum pressures within the arteries. Arterial

pressure is influenced by cardiac output, arterial elasticity, and peripheral vascular resistance. Maintaining normal blood pressure is crucial as it can be easily altered by various activities. Hypertension is categorized into stage 1 and stage 2, with a blood pressure range of 140/80 mmHg to 159/99 mmHg and 160/100 mmHg to 179/109 mmHg, respectively. Hypertensive urgency and emergency refer to extremely high blood pressure that can cause potentially life-threatening symptoms and end-organ damage. Hypotension, on the other hand, is a blood pressure less than 90/60 mmHg. The body needs to adjust to acute changes in blood pressure, and individuals with chronic variations have to receive appropriate medical treatment or lifestyle adjustments. Blood pressure is typically measured using a mercury-tube sphygmomanometer through auscultation.

The role of arterial pressure regulation is to maintain a high enough pressure that allows for proper perfusion of body tissue and organs; but not so high as to cause bodily harm, for that, proper monitoring is very crucial to ensure the safety of patients.[13]

1.7 The Level of Oxygen in the Blood SPO2:

It is a fact that the human body requires a constant supply of oxygen to support vital functions. However, there can be doubts regarding the availability of oxygen at the tissue level. In such situations, blood gas measurements play a critical role in providing vital information concerning oxygenation, ventilation, and acid-base status. It should be noted that these measurements only offer a snapshot of the patient's condition at the time the blood sample is drawn. It is widely known that oxygenation levels can change quickly, and in the absence of continuous monitoring, such changes may go unnoticed until it's too late.

To address this challenge, pulse oximeters have been developed to measure blood oxygen saturation continuously and non-invasively. This measurement indicates the percentage of hemoglobin molecules in arterial blood. Hemoglobin is a protein that is bound to the red blood cells, and the absorption of visible light by a hemoglobin

solution varies with oxygenation. This is because oxidized hemoglobin (HbO2) and reduced hemoglobin (Hb) have different optical spectra (see figure 1.4). Oxygen saturation, often referred to as SaO2 or SpO2, is the ratio of oxyhemoglobin (HbO2) to the total concentration of hemoglobin present in the blood, including both oxyhemoglobin and reduced hemoglobin. The oxygen chemically combined with hemoglobin inside the red blood cells makes up nearly all of the oxygen present in the blood, while there is also a small amount dissolved in the plasma.[14]

The range for normal readings in healthy adults is between 94% and 100%. The term SpO2 is used to denote SaO2 measurements determined by pulse oximetry. However, it should be emphasized that pulse oximetry may give varying readings under certain circumstances, and the use of a different term (SpO2) indicates this. This is a crucial factor to consider when interpreting pulse oximetry readings in clinical settings.[15]

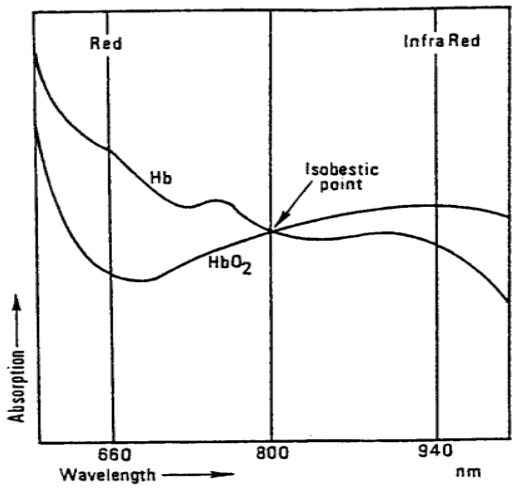

Figure 1. 4 Absorption Spectra of Hb and HbO2. [14]

1.8 Pulse Oximetry:

One of the techniques that medical professionals use to monitor the heart's health is pulse oximetry. It is a non-invasive method of determining the oxygen saturation level in the blood. This is accomplished by employing a small apparatus

called a pulse oximeter (as illustrated on the figure 1.5). The oxygen saturation level, abbreviated as O2sat or SaO2, is the quantity of oxygen the blood is carrying as a percentage of the maximum amount it can hold. It is noteworthy that pulse oximetry does not necessitate the insertion of a needle into the patient's body, making it a painless procedure. Under normal conditions, the percentage of red blood cells carrying oxygen should be greater than 89%. By placing a small device on the fingertip, pulse oximetry can swiftly and accurately determine the pulse rate and the level of oxygen in the bloodstream, which digitally allows us to keep track of the patient's vitals without any discomfort.[16]

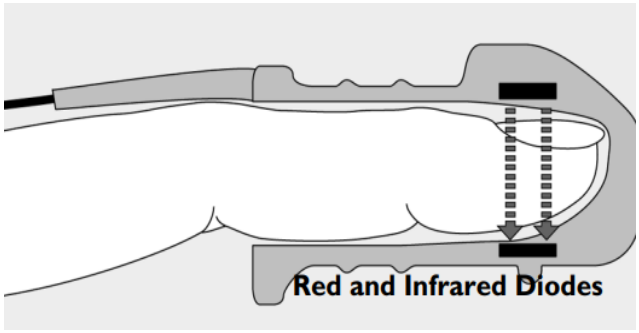

Figure 1. 5 SpO2 Sensor. [17]

1.8.1 Pulse oximeter principles:

A pulse oximeter detects the transmitted light signal after shining two wavelengths of light through a tissue bed, such as an earlobe or finger. The following guidelines govern how the device functions:

- Deoxygenated hemoglobin and oxygenated hemoglobin absorb light differently at the two wavelengths. To be more specific, the collection of linked extinction coefficients for these wavelengths of light absorption is linearly independent, with sufficient variance for proper sensitivity but not enough to make the blood look opaque to either light source. Only oxygenated and deoxygenated hemoglobin are thought to be present in the blood according to this hypothesis.

- Because arterial blood is pulsatile, its waveform in the transmitted signal can be used to distinguish its absorbance effects from those of non-pulsatile venous blood and other bodily tissues. It is possible to achieve a measurement concerning overall tissue absorbance that doesn't require an absolute calibration by employing the quotient of the two effects at different wavelengths. This is a definite benefit of pulse oximeters over earlier oximeter kinds.
- By illuminating enough arterial blood with enough light, scattering in blood and tissue enables the accurate detection of the pulsatile signal. The pulse oximeter must be empirically calibrated due to the scattering effect. However, this phenomenon enables a transmission channel that goes around the finger bone.[17]

1.8.2 Advantages and Limitations of Pulse Oximetry:

Pulse oximeters are useful devices for measuring oxygen saturation levels in the blood, which can indicate if an individual is experiencing low oxygen levels. They are reasonably accurate and provide readings within 2% of the levels measured via arterial blood gas (SaO2). However, numerous factors may influence the accuracy of a pulse oximeter reading. Additionally, compared to arterial blood gas they may fall short in some aspects making them less reliable.

a. Advantages of Pulse Oximetry:

For most individuals, maintaining an oxygen saturation level of at least 89% is required to keep their cells healthy. Brief episodes of low oxygen levels are not believed to result in any harm. However, if such occurrences persist, the cells may be strained or damaged. In the event of a low oxygen level in room air, supplemental oxygen may be prescribed. An oximeter can be employed to determine the required quantity of oxygen and the appropriate time to administer it. For instance, some individuals may require more oxygen while sleeping than while awake, and some may need more oxygen during physical activity than when at rest.

The measurement of oxygen saturation levels by pulse oximeters is reasonably accurate, with most devices providing readings that are within 2% of the saturation levels measured via arterial blood gas. For example, if the pulse oximeter reading shows 92% oxygen saturation, the actual saturation levels could range from 90 to 94%. It is essential to allow the pulse oximeter a few seconds to capture your pulsations to obtain an accurate reading.

If an individual suffers from a lung disorder, their blood oxygen level may be below the standard range. This information is of utmost significance because when the oxygen level in the bloodstream is low, the cells in the body may experience difficulties functioning correctly. Oxygen is the essential "fuel" that powers the human body, and if there is an insufficient supply of it, the body's functions can be disrupted. Additionally, a significantly reduced blood oxygen level may exert stress on the heart and brain.

b. Limitations of Pulse Oximetry:

An oximeter measures the oxygen level in your blood indirectly, whereas an arterial blood gas (ABG) measures both the oxygen and carbon dioxide levels directly by taking blood from an artery, typically from the wrist. ABG is a painful procedure while oximetry is painless but is less accurate than ABG. It is important to note that pulse oximetry cannot measure the level of carbon dioxide in your blood.

Numerous factors may influence the accuracy of a pulse oximeter reading, including cold hands, movement, nail polish (particularly black, blue, or green), artificial nails, very low oxygen saturation levels (below 80%), thicker-than-normal skin, and skin pigmentation. Recent research indicates that pulse oximetry may miss below-normal oxygen saturations in individuals with darker skin pigments. Thus, individuals with darker skin tones should inquire with their healthcare providers about the accuracy of their oximeter readings. Smoking can also influence the accuracy of the oximeter reading, as smoking raises carbon monoxide levels in the bloodstream, which the oximeter cannot differentiate from oxygen. [15]

1.8.3 Applications of Pulse Oximetry:

Pulse oximetry is a diagnostic tool that can be employed to assess the level of oxygen in a person's blood, which is valuable in various scenarios. It may be used to evaluate oxygen levels during or after surgeries or procedures that involve sedation, monitor the efficacy of lung medications, assess an individual's capacity to tolerate increased physical activity, determine the need for or effectiveness of a ventilator for breathing support, and identify instances of sleep apnea characterized by breathing pauses during sleep.

In addition, pulse oximetry is utilized to monitor the oxygenation status of individuals with medical conditions that can impact their blood oxygen levels, such as heart attack, heart failure, chronic obstructive pulmonary disease (COPD), anemia, lung cancer, asthma, and pneumonia.[18]

1.9 Conclusion:

In this chapter, we have introduced a comprehensive analysis of what health monitoring is, the impact technology has on it, and the way technological advancements in the medical field have reshaped the way healthcare is being offered. We have also looked at how health monitoring can be done using different devices while highlighting the importance of heart monitoring by exploring the practicality and concept of a pulse oximeter, as well as its limitations.

In the next chapter, we will talk about the IoT, its importance, the domains that we can encounter it in, and the different types of its architectures.

Chapter 2 Internet of Things (IoT).

2.1 Introduction:

The next paradigm shift is the Internet of Things (IoT), where sensors are linked to the Internet and gather data for analysis to make our world more instrumented, interconnected, and intelligent. Nowadays, the average person carries one or two mobile devices. Therefore, it is possible to considerably lower the cost of equipment in various businesses by taking advantage of the growing use of mobile devices.[19]

There are billions of smart devices or things in today's Internet of Things (IoT) ecosystems, including worldwide homes, hospitals, companies, and vehicles. The number of connected devices as a result is consistently and quickly increasing. For the transmission of sensor or event data, these devices communicate with one another and with other services utilizing a variety of communication protocols.

Applications can use these protocols to gather, store, process, describe, and analyze data to address a range of issues. In addition, IoT seeks to provide secure, reciprocal communication between linked devices, such as microcontrollers, sensors, actuators, or smart appliances, and related cloud services. [20]

2.2 Internet of Things:

For a better understanding of the IoT concept, we need to ask two questions:

- What is IoT?
- Why does IoT matter?

2.2.1 What is IoT?

In 1999, Kevin Ashton proposed the concept of IoT, referring to it as uniquely identifiable interoperable connected objects using radio-frequency identification (RFID) technology. However, the precise definition of IoT is still being developed and is subject to various perspectives.

In general, the Internet of Things (IoT) was described as a "dynamic" global network infrastructure with self-configuring capabilities based on standards and interoperable communication protocols, physical and virtual "things" in an IoT have identities and attributes, are capable of using intelligent interfaces, and can be integrated as an “information network."

Essentially, the Internet of Things (IoT) can be thought of as a superset of connecting devices that are uniquely identifiable by existing near-field communication (NFC) techniques. The terms "Internet" and "Things" refer to a globally interconnected network based on sensory, communication, networking, and information processing technologies, which could be the next generation of information and communications technology.[21]

2.2.2 Why does IoT matter?

Confusion arises not because the concept is too narrow and too tightly defined, but because it is too broad and too loosely defined. When there are so many examples and possibilities in IoT, it can be difficult to grasp the concept.

To help clarify, it is critical to comprehend the advantages of connecting things to the internet. So, why would we want to connect everything to the internet in the first place? Simply put, when something is connected to the internet, it means that it can send or receive data, or both. Things become "smart" when they can send and/or receive information.

A thing does not need to be smart if it has super storage or a supercomputer inside of it. All that is required is for something to connect to super storage or a supercomputer.

All things connected to the Internet in the Internet of Things can be classified into three types:

- Things that collect and send information. (Smart thermostats, fitness trackers, environmental Sensors, etc...)
- Things that receive and act on information. (Smart locks, robotic vacuums, automated sprinklers, etc...)
- Things that can do both.[22]

2.3 IoT applications:

Only a very small portion of the applications that could be developed thanks to the IoT's potential are currently made available to our society.

New applications would probably enhance our quality of life in many areas and settings, such as at home, when traveling, while we are ill, at work, while jogging, and in the gym, to name a few. These environments increasingly contain things with rudimentary intelligence, frequently without the ability to communicate. Giving these things the ability to speak with one another and to elaborate on the information they gather from their surroundings necessitates having various contexts in which a highly diverse variety of applications can be used.[23]

These variety of applications fall into the following categories:

- Transportation and logistics.
- Industrial and manufacturing.
- Energy.
- Smart cities.
- Healthcare

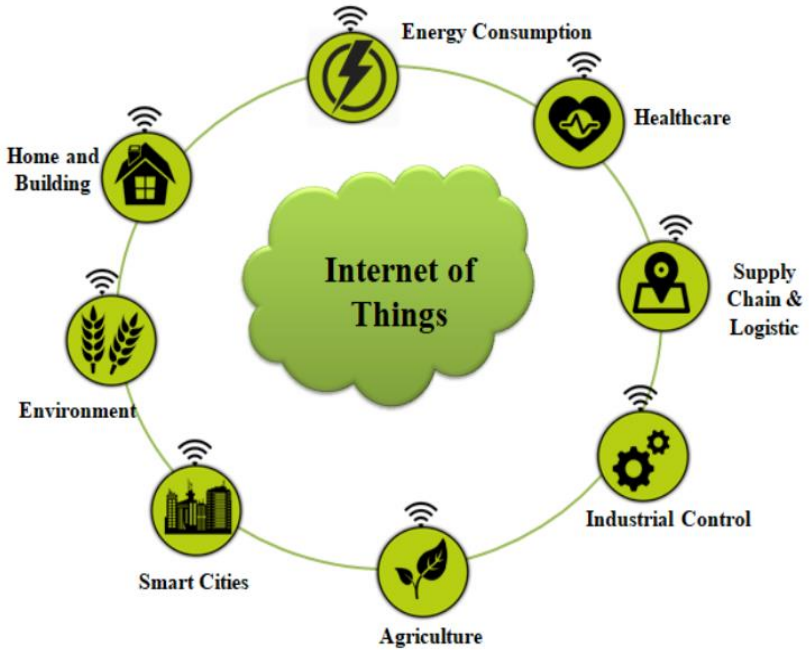

Figure 2. 1 Applications of the Internet of Things.[24]

2.3.1 Transportation and logistics:

The IoT will be significantly influenced by transportation and logistics, if not the main engine. The use cases of IoT in this field are the following:

- Fleet tracking and location awareness.
- Rail car identification and tracking.
- Asset and package tracking within fleets.
- Preventative maintenance of vehicles on the road.[25]

2.3.2 Industrial and manufacturing:

According to the number of connected objects and the value those services provide for manufacturing and factory automation, industrial IoT (IIoT) is one of the fastest-growing and largest categories in the total IoT space. The world of operations technology has typically been included in this area (OT). To monitor physical devices, hardware, and software tools are required. Traditional IT responsibilities have been

handled differently than OT roles in the past. Yield metrics, uptime, real-time data gathering and reaction, and system safety will all be of interest to OT. The focus of the IT job will be on services, data delivery, groups, and security.

As the IoT spreads throughout manufacturing and industry, thousands of factory and production machines will join with predictive maintenance to give an unprecedented amount of data to private and public cloud infrastructure.

Some of the application examples for industrial and manufacturing IoT include the following:

- Preventative maintenance on both new and old factory equipment.
- Real-time demand increases throughout.
- Power savings.
- Safety mechanisms including temperature, pressure, and gas leak detection.
- Expert systems on the shop floor.[25]

2.3.3 Energy:

The energy sector covers the tracking of the flow of energy from the point of production to the client. Consumer and commercial energy monitors, such as smart electric meters that communicate through low-power and long-range protocols to show real-time energy usage, have received a lot of attention in research and development.

Some of the application examples for energy IoT include the following:

- Analysis of thousands of sensors and data points on an oil rig to find efficiency improvements.
- Remote solar panel maintenance and monitoring.
- Risk assessment for nuclear facilities.
- A citywide rollout of smart electric meters to track energy demand and usage.
- Real-time weather-based blade changes on distant wind turbines.[25]

2.3.4 Smart cities:

The term "smart city" refers to bringing intelligence to a previously disconnected world. One of the fastest-growing markets is smart cities, which have good cost-benefit ratios, especially when considering tax revenues. Smart cities also have an impact on inhabitants' lives through convenience, security, and safety.

The following are a few IoT application cases for smart cities:

- Environmental sensing for pollution regulation and control.
- Forecasting the microclimate with citywide sensor networks.
- Cost savings and efficiency improvements via demand-driven waste management.
- Smart traffic signal control and patterning improve traffic flow and fuel efficiency.
- Demand-driven city lighting's energy efficiency.
- Intelligent snow removal based on current traffic conditions, the weather, and nearby plows.
- Adaptive irrigation for parks and public areas based on the climate and usage.
- Real-time automatic AMBER Alerts and smart cameras to monitor for criminal activity.
- Automated parking lots with the greatest spaces for on-demand parking.
- Monitoring wear and usage of infrastructure (bridges, streets, etc.) to increase durability and service.[25]

2.3.5 Health care:

One of the most noticeable uses of IoT technology is in the medical or healthcare industry. The Internet of Things (IoT) technology aids in managing or monitoring remote health services, at-home personal fitness care, chronic diseases like arthritis, asthma, cancer, and diabetes, as well as care for elderly people like heart attack detection and activity and movement recognition for the elderly.

The combination of RFID (radio frequency identification system), Bluetooth, and Wi-Fi technologies can quickly lead to a person's health. By combining RFID and other technologies with sensors, we may learn details about the human body's temperature, heart rate, blood pressure, and other parameters. In the event of an emergency, the patient's doctor is informed of all the data gathered by the use of sensors.

There are numerous health-related services in which the Internet of Things plays a significant role (see Figure 2).[24]

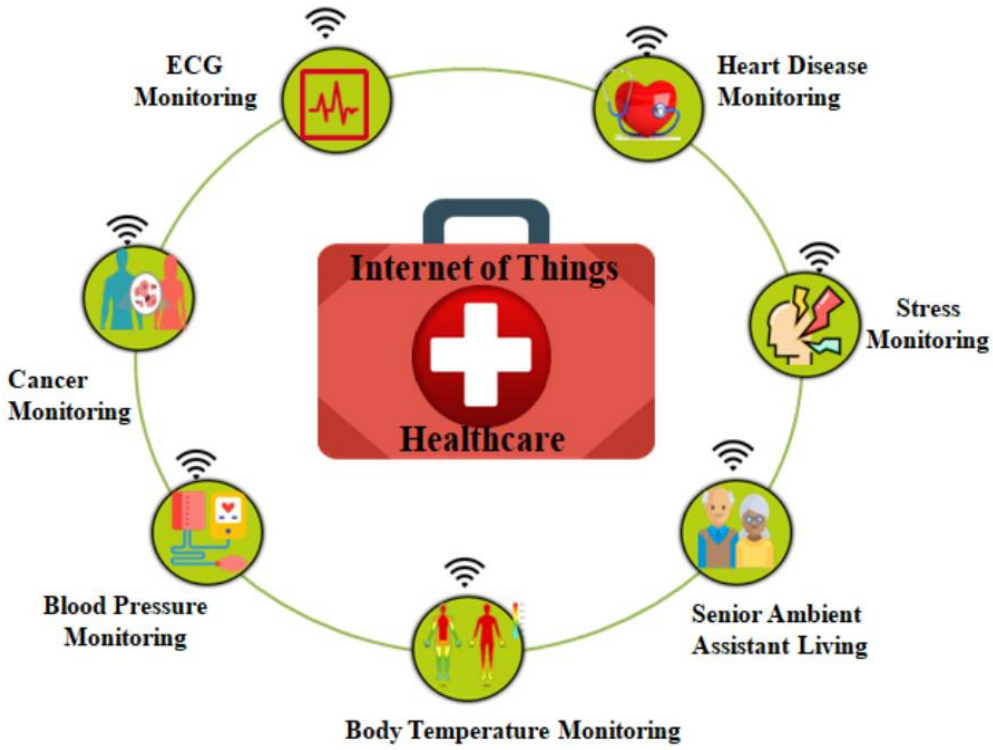

Figure 2. 2 Internet of Things in the Field of Healthcare. [6]

a. Health Monitoring System:

The system that we improved in our research is specific to a single patient and is equipped with a data acquisition unit (medical sensors) which is comprised by a MAX30100, DS18b20 and a DHT11 accompanied by a ESP32 microcontroller (view Figure 2.3).

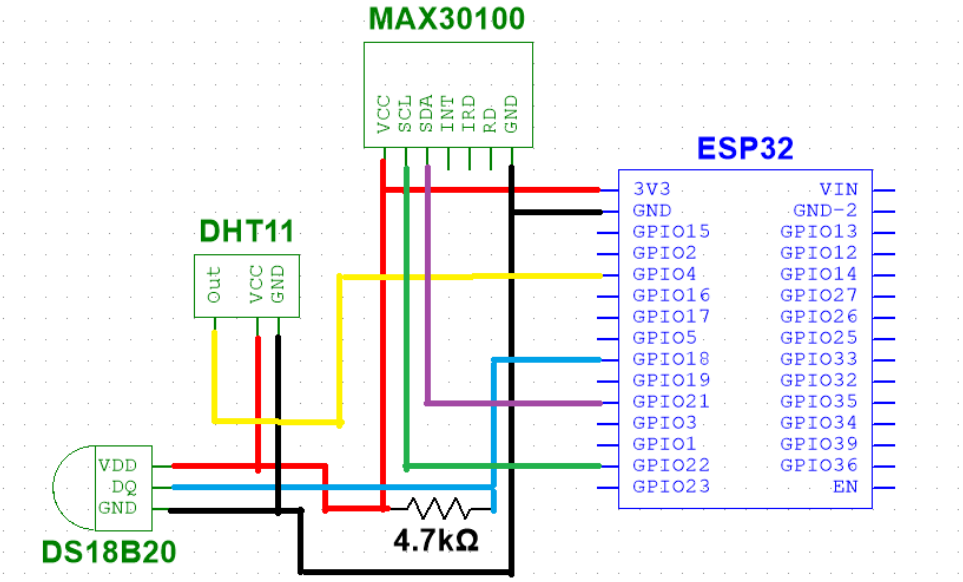

Figure 2. 3 Circuit diagram of the IoT health monitoring system.

- **MAX30100 Sensor :**

The MAX30100 is a sensor system with integrated pulse oximetry and heart rate monitoring. To detect pulse oximetry and heart rate signals, it incorporates two LEDs, a photo detector, improved optics, and low-noise analog signal processing. It runs on 1.8V and 3.3V power supplies and may be switched off by software with barely any standby current, allowing the power supply to always be connected.

Ambient light cancellation (ALC), a 16-bit sigma delta ADC, and a custom discrete temporal filter make up the SpO2 subsystem of the MAX30100.A continuous time oversampling sigma delta converter with a maximum resolution of 16 bits is the SpO2 ADC. The ADC output data rate ranges from 50Hz to 1 kHz and is programmable. The MAX30100 has a customized discrete temporal filter that eliminates interference at 50Hz and 60Hz as well as low-frequency residual ambient noise, (view Figure2.4).[26]

Figure 2. 4 MAX30100 heart rate sensor.

- **DHT11 Sensor:**

A temperature and humidity sensor complex with a calibrated digital signal output is part of the DHT11 Temperature & Humidity Sensor. It guarantees high dependability and outstanding long-term stability by utilizing the unique digital-signal-acquisition technique and temperature & humidity sensing technology. This sensor links to a high-performance 8-bit microcontroller and combines a resistive-type humidity measurement component with an NTC temperature measurement component to provide great quality, quick response, interference resistance, and cost effectiveness (view Figure 2.5).

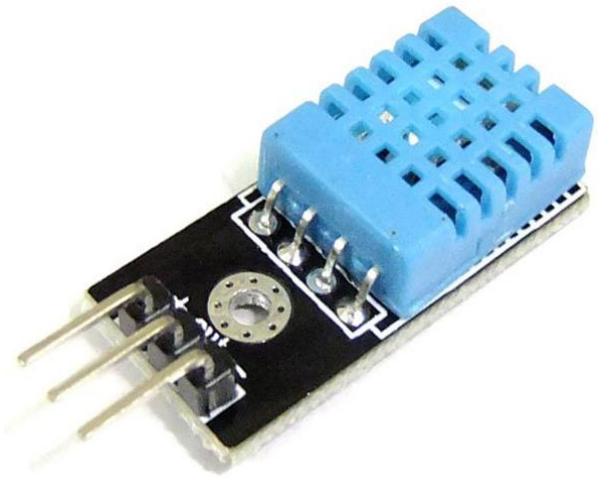

Figure 2. 5 DHT11 Temperature and Humidity sensor.

Each DHT11 component is meticulously calibrated in a humidity calibration lab that is exceptionally accurate. The calibration coefficients are

used by the sensor's internal signal detecting process and are stored as programs in the OTP memory. System integration is quick and simple thanks to the single-wire serial interface. It is the finest option for a variety of applications, including those that are the most demanding, because to its small size, low power consumption, and up to 20 meter signal transmission. The component is packaged with a single row of four pins. The ability to connect is simple, and users can request certain packages. [27]

- **DS18B20 Sensor:**

The 9-bit to 12-bit Celsius temperature measurements provided by the DS18B20 digital thermometer include an alert function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates with a central microprocessor via a 1-Wire bus, which by definition only needs one data line (and ground) for communication. Furthermore, the DS18B20 may obtain power directly from the data line (sometimes known as "parasite power"), doing away with the requirement for an additional power source (view Figure 2.6).[28]

Figure 2. 6 DS18B20 temperature sensor.

- **ESP32 Microcontroller :**

The ESP32 is a dual-core, low-power microcontroller (MCU) board with built-in Wi-Fi and Bluetooth. It is also inexpensive. The first is a potent processor, such as an Xtensa LX6 (240 MHz) with 512 KiB of memory, and the second is an ultra-low coprocessor (ULP) with only 8 KiB of memory that is intended to operate when the ESP32 is in deep sleep mode(view Figure2.7 for pinout).[29]

When it comes to the ESP32 chip specifications, you will find that:

- The ESP32 is dual-core, which means it has two processors.
- It has Wi-Fi and Bluetooth built-in.
- It runs 32-bit programs.
- The clock frequency can go up to 240MHz and it has 512 kB RAM.
- It also has a wide variety of peripherals available, like capacitive touch, ADCs, DACs, UART, SPI, I2C, and much more. [30]

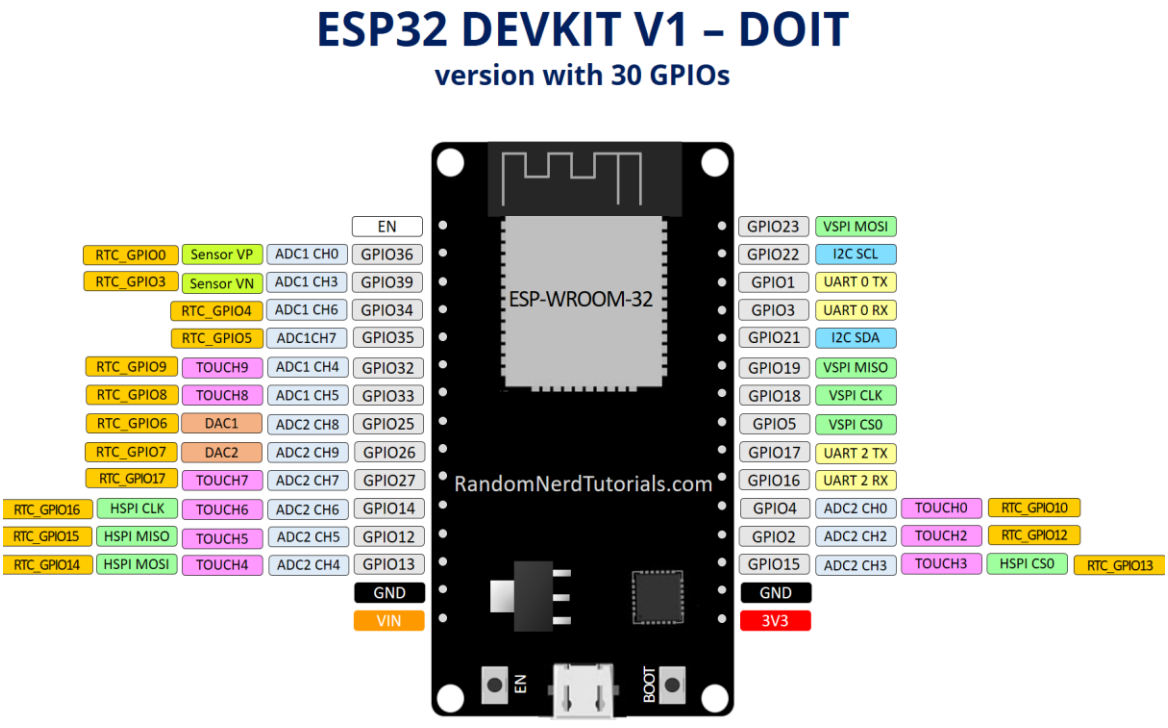

Figure 2. 7 pin layout of the ESP32 Board. [30]

2.4 IoT architecture:

The interconnection of the objects in the network is a crucial necessity for IoT. IoT operations, which connect the physical and virtual worlds, must be ensured by IoT system architecture. Several variables, including networking, communication, business models and procedures, and security, go into the design of IoT architecture. The extensibility, scalability, and interoperability of heterogeneous devices and their business models should be taken into account while creating the IoT architecture.

The IoT architecture should be adaptable to let devices engage with other things dynamically and facilitate the clear transmission of events because objects may move geographically and need to interact with others in real-time mode. In addition, the IoT should be heterogeneous and decentralized.

There is no single model utilized for IoT engineering due to the absence of standards. There are benefits and drawbacks to using different architecture applications. Many scientists have offered several models.[31]

In this segment, we are going to focus on the following IoT architectures:

- Four-layer architecture.
- Seven-layer architecture.

2.4.1 Four-layer architecture:

Service-oriented architecture (SOA) may be essential for customers and service providers in the Internet of Things. In numerous ways, SoA ensures the interoperability of heterogeneous devices. A generic SoA is shown in Figure 3, which has four layers with distinct functionality.

A complex system is viewed by the SoA as a collection of well-defined simple objects or subsystems. The software and hardware parts of an IoT can be efficiently upgraded and reused since those items or subsystems can be maintained and reused separately. SoA has been widely used as a standard architecture for wireless sensor networks because of these benefits. In the context of the Internet of Things, SoA is

intended to provide flexibility, scalability, modularity, and interoperability across heterogeneous objects. In addition, the functionalities and capacities are abstracted into a common set of services.

The SoA proposed for IoT is illustrated in Figure 3, and the specifics of its components are detailed below.[3]

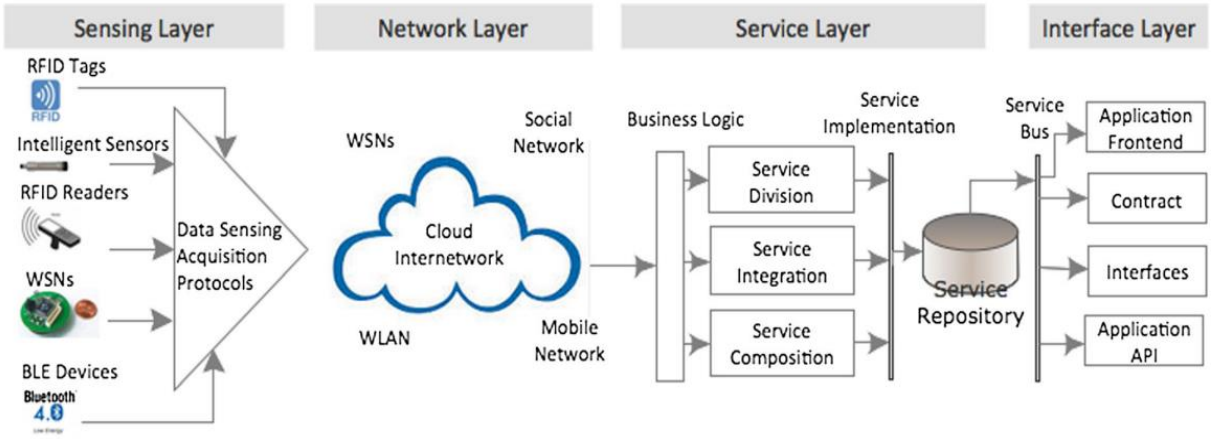

Figure 2. 8 Service-oriented architecture for IoT.[3]

a. Sensing layer:

The primary responsibility of the sensing layer is to sense both the outside environment and the internal status of the object. The physical layer is in charge of collecting important raw data from sensors and other intelligent things. In this layer, many sensing technologies are employed (e.g. GPS, IR, RFID, WSN, Gyro, etc.). This layer makes it convenient for network transmission by converting the physical values into digital signals. [8]

The IoT has significantly expanded in recent years thanks to advances in sensing and communication technologies that have made objects with RFID or sensors more adaptable and accessible. This is because objects can now be uniquely identified and the environment around them can be monitored for a variety of uses. Every IoT object has a digital identification that makes it simple to trace it online. A universal unique identifier (UUID) is a method of giving a thing a distinct identification (UUID). On an

enormous network like the IoT, UUID in particular is essential for successful service rollout. The IDs could be addresses or names.

The following factors should be taken into account when choosing an IoT's sensing layer:

- **Cost, dimensions, resource use, and energy usage**: The items could have sensor nodes and RFID tags as well as other sensing equipment. Due to the numerous sensors used in complicated system applications, intelligent devices should be made with the least amount of resources and money possible.
- **Deployment:** Depending on the needs of applications, the sensing devices (such as RFID tags, sensors, etc.) can be deployed all at once, progressively, or randomly.
- **Heterogeneity**: The Internet of Things (IoT) can be quite heterogeneous due to a range of devices with various characteristics.
- **Communication**: For objects to be retrievable and accessible, sensors must be able to communicate.
- **Network**: They are set up as ad hoc, multi-hop, or mesh networks.[3]

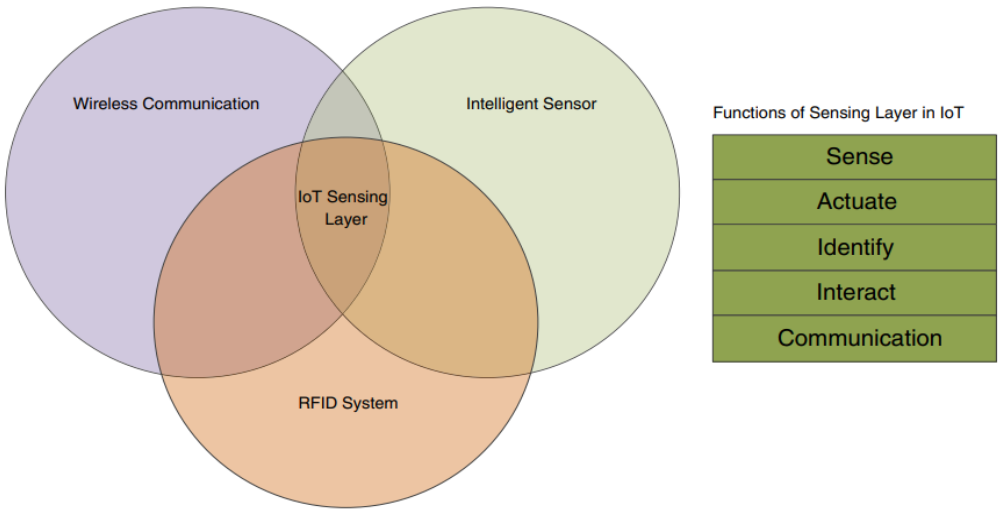

Figure 2. 9 Functions of the sensing layer in IoT.[3]

b. Network layer:

IoT connects all objects and enables them to be aware of their surroundings thanks to the network layer. To manage and process intelligent events in the Internet of Things, things must be able to share data with connected things via the network layer. Also, the networking layer can aggregate data from current IT infrastructures, which may subsequently be sent to departments that make decisions for very complicated services.

In an SoA, the things that are deployed in a heterogeneous network always provide the services. Using the service Internet, pertinent stuff can also be integrated. Quality of Service (QoS) may be used in network communication to ensure dependable services for various users or applications.

On the other hand, a network must be able to automatically find and map objects within the network. To deploy, manage, and plan the behaviors of things, as well as to be able to transition between any roles whenever necessary, things need to be automatically given roles. This makes it possible for devices to work together on activities.

The following problems need to be solved at the networking layer:

- Network management technologies including managing fixed, wireless, and mobile networks.
- Network energy efficiency.
- Requirements of QoS.
- Technologies for mining and searching.
- Data and signal processing.
- Security and privacy.[3]

c. Service layer:

The middleware technology that is a crucial enabler of IoT services and applications is used by the service layer. Hardware and software platforms can be reused thanks to middleware technology's affordable platform. Activities required by

the middle service specifications are included in the service layer. The network is directly used by the services at the service layer to efficiently find new services for applications and retrieve dynamic metadata about the latter. The majority of specifications are handled by numerous standards created by various organizations. For the Internet of Things, a widely acknowledged service layer is crucial. Applications, application programming interfaces (APIs), and protocols enabling necessary applications and services make up the minimal set of viable service layers.

In the service layer, all service-oriented tasks are carried out, including information exchange and storage, data management, ontology database, search engines, and communication. The activities are conducted by the following components:

- **Service discovery** identifies items that can successfully supply the needed service and information.
- **Service composition** facilitates communication between connected objects. In order to find the needed service, discovery uses relationships between objects as a tool. To get the most dependable services, service composition arranges or builds new, better-suited services.
- **Trustworthiness management** strives to comprehend how the data supplied by other services must be processed.
- **Service APIs** provide the user-needed interactions between services.[3]

d. Interface layer:

IoT involves a vast number of devices, many of which may come from different suppliers and hence may not necessarily comply with the same standards. For there to be interactions between things, the issue of compatibility between heterogeneous items needs to be addressed. Exchange of information, communication, and event processing are all part of compatibility. An efficient interface mechanism is desperately needed to make managing and connecting items easier.

A subset of service standards known as an interface profile (IFP) allows for the least amount of interaction with applications running on application layers. The specifications between applications and services are described in the interface profiles. The implementation of Universal Plug and Play (UPnP), which outlines a protocol for seamless interactions among diverse objects, serves as an example Seven-layer.[3]

2.4.2 Seven-layer architecture:

Things in an IoT system can refer to any sensor or device, from extremely commonplace small items to enormous live organisms. Actuators serve as output devices to control or move something by the instructions received from the Internet. Microcontrollers process and send the data produced by or collected by sensors in objects. The number of connections between individuals and objects is growing daily, and the volume of data produced every second is staggering.

Data produced by various devices can be processed in a variety of ways, transported to distant locations, and then used by applications. In 2014, Cisco developed a seven-layer Internet of Things paradigm to develop a uniform worldwide reference framework(see Figure 5).[20]

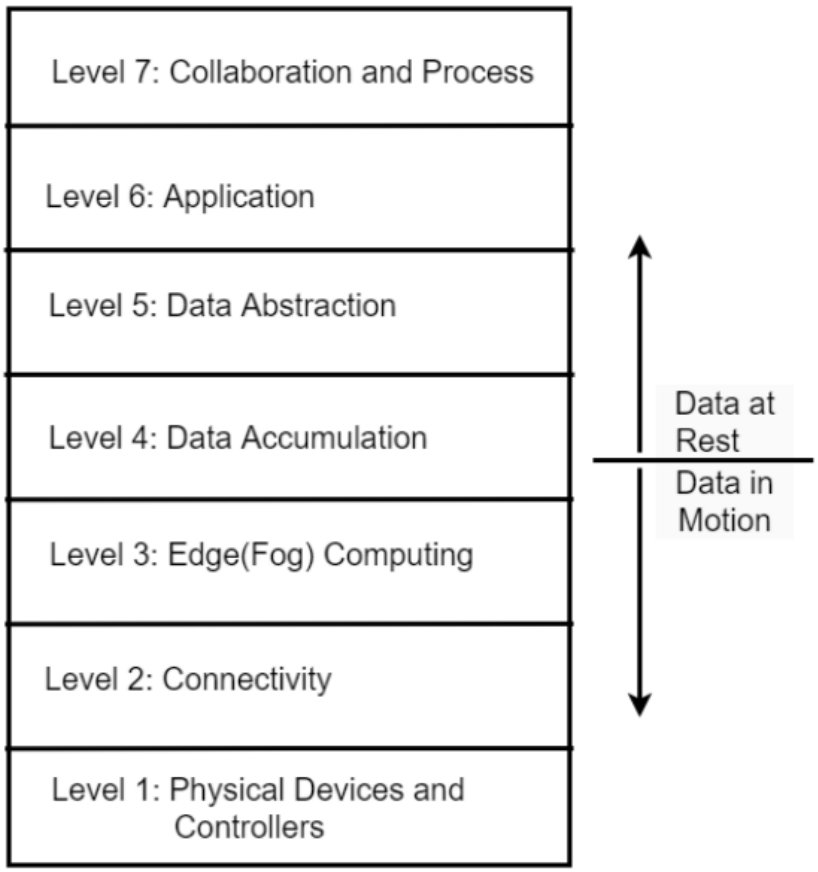

Figure 2. 10 The seven layers of the IoT Reference Model. [20]

a. Physical devices and controllers:

In the Internet of Things, we refer to physical devices and device controllers as things. These items are many since they come in a variety of sizes, shapes, locations, and origins.

An item can range in size from as small as a silicon chip to as big as a car. These items are endpoint devices that can generate data, send and receive information, and convert analog signals into digital signals in response to control commands they receive via the internet from physical devices and controllers.[20]

b. Connectivity:

An IoT system has many levels in addition to the data communications network. Conventional data communication networks are based on the (ISO) 7-layer reference architecture and feature numerous functions based on it. There is no need to build a

new network because the sole goal of an IoT reference model is to conduct M2M communications over the existing networks. It functions flawlessly on current networks thanks to its design principles. Communication with and between level 1 devices, dependable delivery across networks, application of various protocols, switching and routing, protocol translation, security at the network level, and (self-earned) networking analytics are all aspects of connectivity.[20]

c. Edge (Fog) Computing:

Level 3 is concerned with transforming network data flows into information that can be stored and processed further at Level 4. It focuses on the analysis and transformation of large volumes of data. The core idea behind the IoT reference model is that intelligent systems should start processing data as soon as feasible, as close to the network's edge as possible, rather than relying solely on the cloud to handle everything. Edge or fog computing are common names for It.

Data generated by things are often transferred to the connectivity level's networking devices, hence level 3 comprises little packet-by-packet processing without session-based transactions. Data filtering, cleanup, aggregation, packet content inspection, a mix of network and data level analytics, and thresholding are some of the functions of level 3 data elements. Event generation and processing functions consist of data reformatting for consistent higher-level processing, expanding, or decoding, which refers to including additional information such as the source of the data, distillation/reduction, which refers to reducing the impact of data and traffic on the network and higher-level processing systems, and assessment, which involves evaluating threshold or alert, which process may redirect data to different locations, are all examples of data processing techniques.[20]

d. Data accumulation or storage:

Mechanisms are put in place at this level to enable apps to use network data. Data in motion must be transformed into data at rest, network packets must be

formatted into relational database tables, event-based computing must be made possible, query-based computing must be made possible, and data must be reduced through filtering and selective storing. In a nutshell, level 4 collects and stores data for later use by non-real-time applications. By acting as a barrier between event-based data generation and upper-level query-based data consumption, it bridges the gap between the real-time networking world and the non-real-time application world.[20]

e. Data Abstraction:

IoT solutions must scale to meet demand, whether it be at the corporate or global level. Several storage solutions are needed to scale the massive volume of data that is received from IoT and non-IoT equipment. As a result, information integration from diverse data sources becomes necessary. Hence the level of data abstraction is always changing. It resolves the discrepancy between various data formats from various sources and ensures that the semantics of the data are consistent across all sources.

Additionally, it makes sure that the data is complete enough to be used by any higher-level applications, merges the data into a single location, or uses data virtualization to provide access to many data stores. Moreover, it makes data protection easier by enabling appropriate permission and authentication, normalizing or denormalizing, and using indexing to speed up application access.[20]

f. Application:

Applications at Level 6 can communicate with data at rest and with Level 5. Thus, network speeds are not necessary for this level to function. Monitoring device data is one of the crucial features of this level. In addition to controlling devices, fusing data from devices and other sources, evaluating data, and reporting.[20]

g. Collaboration and Process:

Beyond the technical model, this level functions. Both people and business procedures are part of it. Without intelligent analytics, the information generated by various IoT technologies is of very little value. A wise choice and suitable action frequently include people and processes. To extract relevant information from IoT data, people employ a variety of applications and techniques. Usually, it involves more than one person. To make the most of IoT data and make the best business decisions possible at the right moment, people must interact and communicate with one another. Hence, Level 7 denotes a level involving people greater than a single application.

2.5 IoT communication Protocols:

The reference model for the Internet of Things systems was introduced in the part before, and its implementation called for a trustworthy communication protocol. This part covers IoT protocols in general, introduces the MQTT protocol in depth, and defines the fundamentals of a 'publish/subscribe' (pub/sub) service.

For communications at the lowest level, IoT networks use a variety of radio technologies, including RFID, WLAN (IEEE 802.11), WPAN (IEEE 802.15), and WMAN (IEEE 802.16), among others. LoRaWAN and SigFox fall into the long-range (km)-low data rate (bps-kbps) category, Cellular/4G/5G into the long-range (km)-high data rate (Mbps) category, ZigBee, and Zwave into the medium-range (m)-medium-data rate (kbps) category, Wi-Fi into the medium-range (m)-high-data-rate (Mbps) category, and NFC into the short-range (cm). All independent data-generating end devices must make their data available to the internet for further processing and send control information back, regardless of the radio technology employed to build an IoT network. The effectiveness of M2M communication inside IoT applications significantly depends on the unique messaging protocols created for M2M communication. The web employs the one standard communications protocol HTTP, whereas IoT cannot adhere to the "one protocol fits all" tenet due to its excessive diversity in

characteristics. As a result, there are now numerous communications protocols available to choose from for different IoT system requirements.

The four established and growing messaging protocols for IoT systems are MQTT, CoAP, AMQP, and HTTP. The TCP/IP protocol stack for IoT systems is depicted in Figure 6.[20]

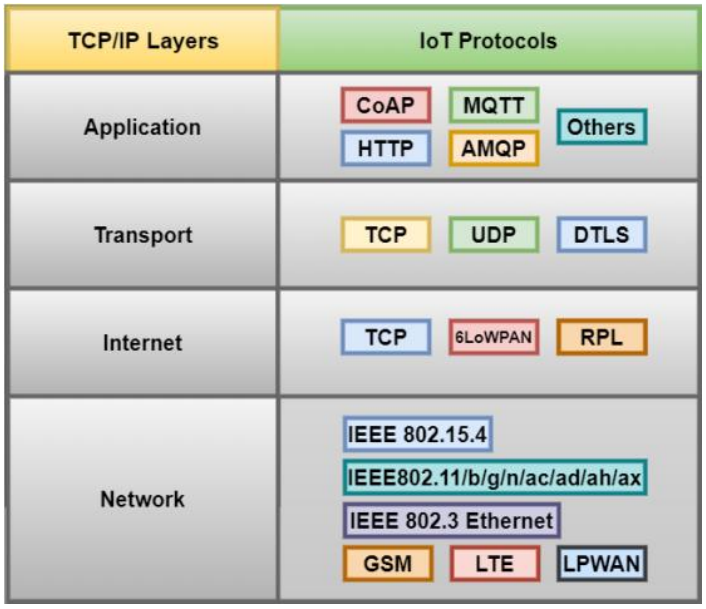

Figure 2. 11 TCP/IP Model-Main IoT Protocols.[20]

2.5.1 Publish/subscribe protocols:

"Message" refers to the data that passes through a pub/sub messaging service, "topic" is a named entity that signifies a feed of messages, and "subscription" denotes an interest in receiving messages on a specific topic. A device or software that produces messages and publishes them to the messaging service on a specific topic is referred to as a "publisher or producer," and a device or program that consumes messages is referred to as a "subscriber or consumer". The protocols that are based on this service are the following:

- AMQP.
- DDS.
- MQTT.[20]

a. Advanced Message Queuing Protocol (AMQP):

It is defined as an application layer protocol for the message-oriented environment that is open and the standard is called AMQP. Its objective is to make it possible for a variety of various applications and systems to collaborate to improve interoperability. Moreover, the protocol is open-source and asynchronous. A message exchange mechanism, such as different queues for each subscriber, is included in the AMQP architectural scheme, which is the same as MQTT's publishing, broker, and subscriber strategy. Fig. 7 depicts the same thing.

Figure 2. 12 Mechanism of AMQP Protocol.[32]

As shown in Fig. 7, this exchange architecture accepts messages from the publisher and routes them to queues based on predetermined standards. The message is examined using a function and instances, and then it is forwarded to the appropriate queue using a key, which is a virtual address.[32]

b. The Data Distribution Service (DDS):

In contrast to the other, publish/subscribe models, the Data Distribution Service (DDS) focuses on devices that use device data directly. For instance, DDS is regarded as a device-to-device protocol since it distributes data to other devices on a bus as opposed to being a device-to-server protocol, where servers harvest data from devices in a point-to-point or star topology, as with MTTQ. While DDS was designed to interface with other devices, which entails quick communication and teamwork

amongst devices on the same segment, the protocol also facilitates device-to-server communication.

The idea behind how DDS manages the publish/subscribe approach is that since devices are quick, they need data immediately. In this setting, "real-time" is frequently expressed in terms of microseconds. This occurs frequently because time-sensitive applications generally require complicated device-to-device communication in IIoT environments. As a result, the point-to-point connection-oriented data streams of TCP/IP are incredibly slow and constrained. Instead, DDS provides multicast, customizable dependability, extensive redundancy, and precise quality-of-service (QoS) control. Without the time delay of the broker model, DDS also offers means to filter and publish to certain subscribers, who may be thousands of simultaneous destinations. DDS can also allow simplified variations of the protocol that operate in limited circumstances, such as on low-power devices.[33]

c. Message Queuing Telemetry Transport (MQTT):

The lightweight MQTT protocol is based on binary and utilizes little battery power and bandwidth. Given that it uses an acknowledgment system for all formats, it is a more dependable protocol. The asynchronous message queuing protocol described here is open-source. For machine-to-machine communication in a low bandwidth environment, it uses publish or subscribe architecture and runs on TCP. Three components are offered by the MQTT specification: connection semantics, routing, and endpoint(see Figure 8).[32]

There are three parts to the protocol:

- Publisher or Producer (An MQTT Client).
- Consumer/ Subscriber (An MQTT Client).
- A broker (An MQTT Server).

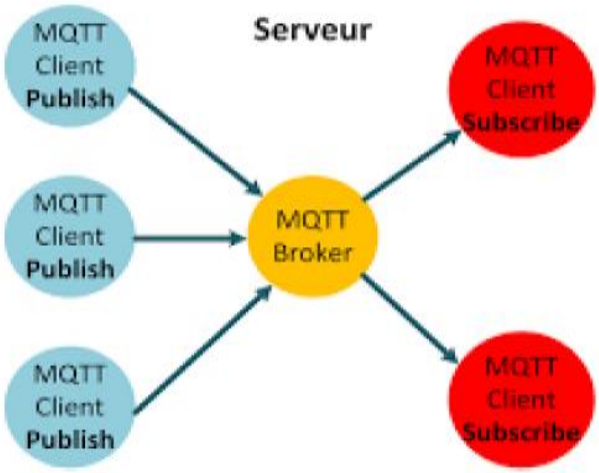

Figure 2. 13 MQTT Architecture.

Opening a network connection to the server, creating messages to be published, publishing application messages to the server, subscribing to request application messages that it is interested in receiving, unsubscribing to remove a request for application messages, and closing the network connection to the server are all tasks that fall under the purview of the client. The data sent over the network using the MQTT protocol for the application is referred to as the message's application. Each application message sent over MQTT has a topic name, a QoS, a set of properties, and payload data.

A program or device based on MQTT known as an MQTT server serves as a post office between publishers and subscribers. An MQTT broker is in charge of accepting network connections from clients, accepting application messages published by clients, processing client requests for subscription and unsubscription, sending application messages to clients by their subscriptions, and closing the client's network connection.[20]

2.6 Conclusion:

In this chapter, we have presented a detailed study about what the Internet of Things is, its importance, the domains that we can encounter it in, and different types of architectures for a functioning IoT system as well as protocols that ensure said functionality.

The Internet of Things is a groundbreaking technology that connects physical objects to share data with other devices and systems with minimal human intervention, making our world more interconnected than ever before. In the next chapter, we will talk in detail about how we can create an IoT-based health monitoring system.

Chapter 3 Design And Implementation

3.1 Introduction:

In this chapter, we will present two main sections: the first talks about the working environment, and the second details the steps made to accomplish this project, these will be our most essential elements for creating our system.

3.2 Working environment :

3.2.1 Text editors and software:

Many softwares have been used to realize this project, most notably VS code, Arduino IDE, phpMyAdmin, and the mosquitto MQTT Broker.

a. VS code:

Microsoft's premier Integrated Development Environment (IDE) for creating, testing, and deploying desktop, web, and mobile apps is called Visual Studio (VS). For simple application development, Visual Studio includes a wide variety of features like auto complete, server setup, database integrations, and source control.

Since Visual Studio Code is so extensible, you can add extensions for almost every major computer language. This implies that you do not need to acquire overly complex or bloated IDEs like Visual Studio.

Numerous computer languages are supported by Visual Studio Code, including Python, JavaScript, HTML, CSS, Typescript, C++, Java, PHP, Go, C#, PHP, SQL, Ruby, Objective-C, and many others.

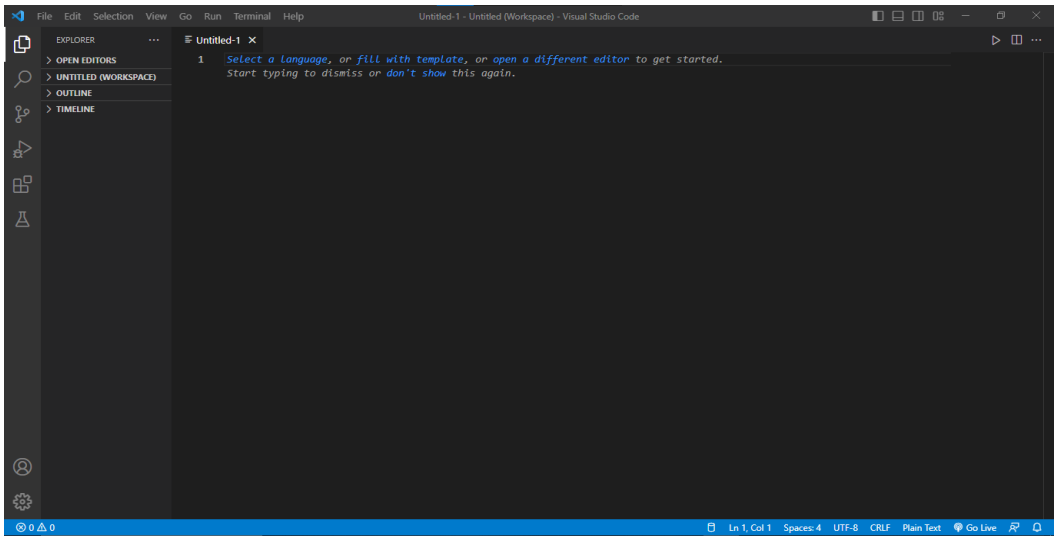

Figure 3. 1 VS code interface.[34]

b. Arduino IDE:

In addition to a text editor for writing code, a message area, a text console, a toolbar with icons for frequently used operations, and several menus, the Arduino Integrated Development Environment, also known as the Arduino Software (IDE), is also available. To upload programs and interact with them, it connects to the Arduino hardware.

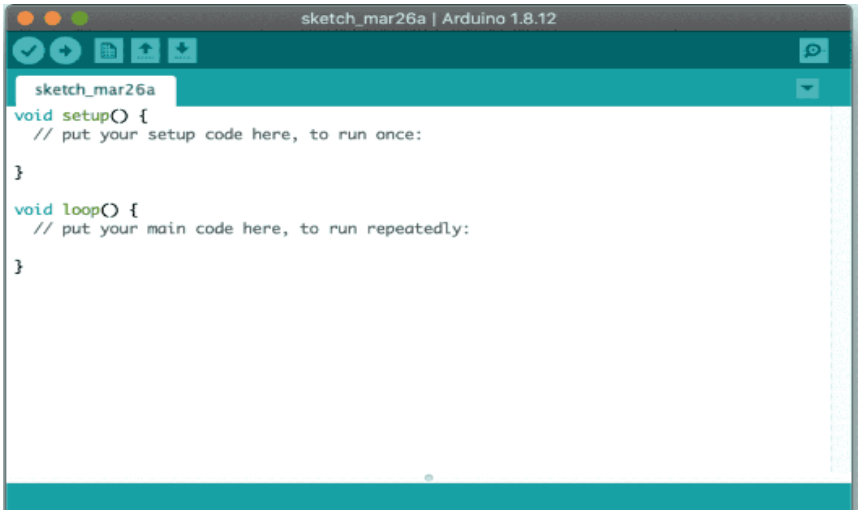

Figure 3. 2 Arduino IDE interface. [35]

Sketches are computer programs created using Arduino Software (IDE). These drawings are created in a text editor and stored as files with the ".ino" extension. The editor has functions for text replacement and text scanning. When saving and exporting, the message section provides input and shows errors. The console shows text generated by the Arduino Software (IDE), including error notices in their entirety and other data. The configured board and serial port are visible in the window's lower right corner. You can create, view, and save sketches, validate and upload programs, open the serial monitor, and more using the toolbar buttons.[35]

c. Kivy:

In order to create cross-platform graphical user interfaces and applications that require touch or multi-touch interfaces, such as mobile or tablet applications, Kivy acts as a free and open-source Python framework. It works with a variety of operating systems, including Windows, macOS, Linux, Android, and iOS.

For creating user interfaces, it employs the KV language, a special declarative language. It aids in the definition of the interface's design and functionality utilizing a straightforward and understandable syntax independent of the application logic. Button, label, slider, and text input widgets are just a few of the many pre-made tools and widgets that come with Kivy and may be easily modified and expanded. [36]

d. phpMyAdmin:

PHP was used to create the web program phpMyAdmin, which also includes client code for XHTML, CSS, and JavaScript (like the majority of web apps). It offers a comprehensive online interface for managing MySQL databases and is widely regarded as the industry standard.

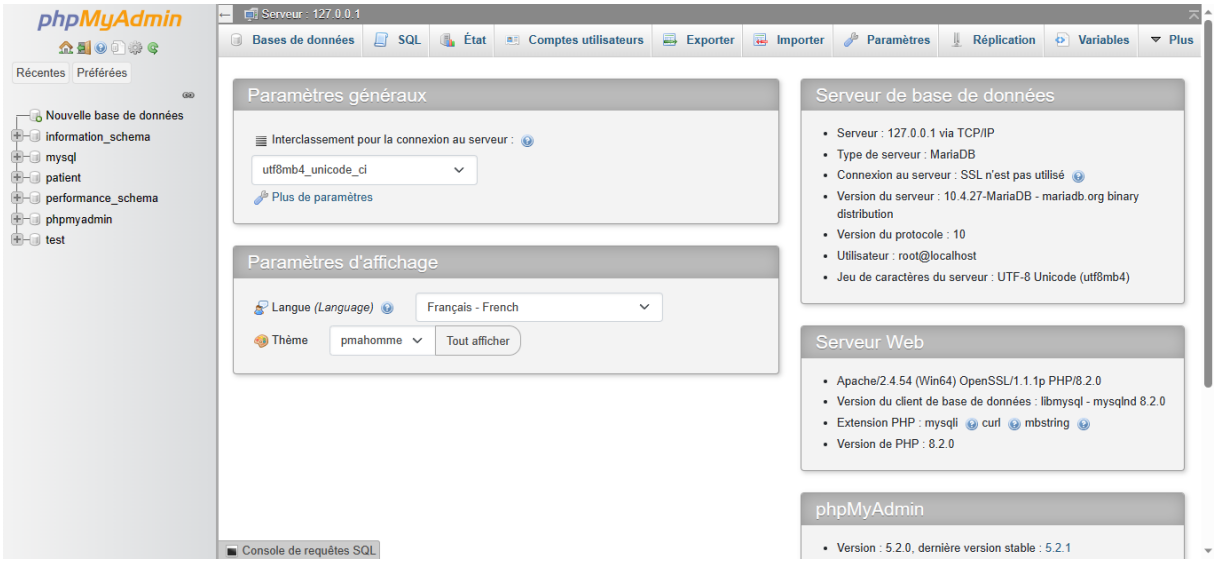

Figure 3. 3 phpMyAdmin Home page. [37]

e. Mosquito MQTT Broker:

Receiving network connections from the client, managing the client's Subscribe/Unsubscribe and Publish requests, and forwarding the messages the client publishes to other subscribers are all tasks that fall under the purview of the MQTT Broker.

The lightweight open-source message broker, known as Mosquito, implements MQTT versions 3.1.0, 3.1.1, and 5.0. Roger Light has written it in C. It is an Eclipse project, free to download, and compatible with Windows and Linux. A publish/subscribe architecture is used by the MQTT protocol to provide a lightweight messaging solution. As a result, it is appropriate for Internet of Things messaging, such as with low-power sensors or mobile devices like phones, embedded computers, or microcontrollers. [38]

3.2.2 Programming languages:

Our work required us to use different coding languages in order to exploit the strengths and features of each language, improve performance, and seamlessly combine components to create robust and efficient systems.

a. HTML (Hypertext Markup Language):

The most fundamental component of the Web is HTML (Hypertext Markup Language). It describes the purpose and organization of web content. Links that join online pages together, either within a single website or between websites, are referred to as "hypertext."

At its core, HTML is a language made up of elements that may be used to give the text in a document distinct meaning, organize a document into logical sections, and incorporate material like photos and videos into a page.[39]

b. CSS (Cascading Style Sheets):

CSS is a style sheet language that is used to describe how a document written in HTML or XML is presented (Including XML dialects such as SVG, MathML, or XHTML). CSS specifies how items should be shown in various media, including speech, paper, screens, and other media.[40]

c. PHP (Hypertext processor):

A popular general-purpose open-source scripting language that is ideal for web development and can be integrated into HTML is PHP.

With PHP, which primarily focuses on server-side scripting, you may do any CGI program's functions, including collecting data forms, creating dynamic page content and sending and receiving cookies. But PHP is considerably more versatile as PHP scripts are mostly utilized in three different contexts:

- **Server-Side scripting:** This is PHP's most established and primary application area. The PHP parser (CGI or server module), a web server, and a web browser are required for this to function. Running the web server and connecting the PHP installation are both required. By accessing the PHP page via the server, you can access the output of the PHP program using a web browser. If you're simply dabbling with PHP programming, you can run all of these on your home computer.
- **Command line scripting**: A PHP script can be created to run it independently of a server or browser. To utilize it in this manner, you only need the PHP parser. This method of use is appropriate for scripts that are regularly run by Task Scheduler or cron (on *nix or Linux) (on Windows). Simple text-processing tasks can also be accomplished with these programs.
- **Writing desktop applications:** The best language for writing desktop apps with a graphical user interface is probably not PHP, but PHP-GTK, it can be used to construct such programs if you are highly familiar with PHP and want to employ some advanced PHP features in your client-side applications. Additionally, with this approach, you may create cross-platform applications.[41]

d. Python:

Programmers create programming languages for specialized purposes. Their objectives influence the language's development, adaptability, and usability. Python was developed as a language that programmers could use to write effective and productive codes. Some of Python’s uses are the following:

- Scripting browser-based applications.
- Creating better user interfaces.
- Designing scientific, mathematical, and engineering applications.
- Interacting with databases.[42]

e. MySQL:

The MySQL software provides a very quick, multi-threaded, multi-user, and reliable SQL (Structured Query Language) database server. Both embedded in widely used software and mission-critical, high-load production systems are intended uses for MySQL Server. MySQL has two licenses. Users have the option of purchasing a normal commercial license from MySQL AB or using MySQL software as an Open Source product under the terms of the GNU General Public License.[43]

3.3 System development:

3.3.1 System diagram:

To enhance the monitoring and data management of the system, we added a MQTT broker and a MYSQL database to an already-existing health monitoring system.

The system uses the MQTT protocol for communication, in which the ESP32 publishes the data it has gathered from the medical sensors to the MQTT broker, who then stores it as topics and grants access to the subscribers (see Figures 3.4 and 3.5).

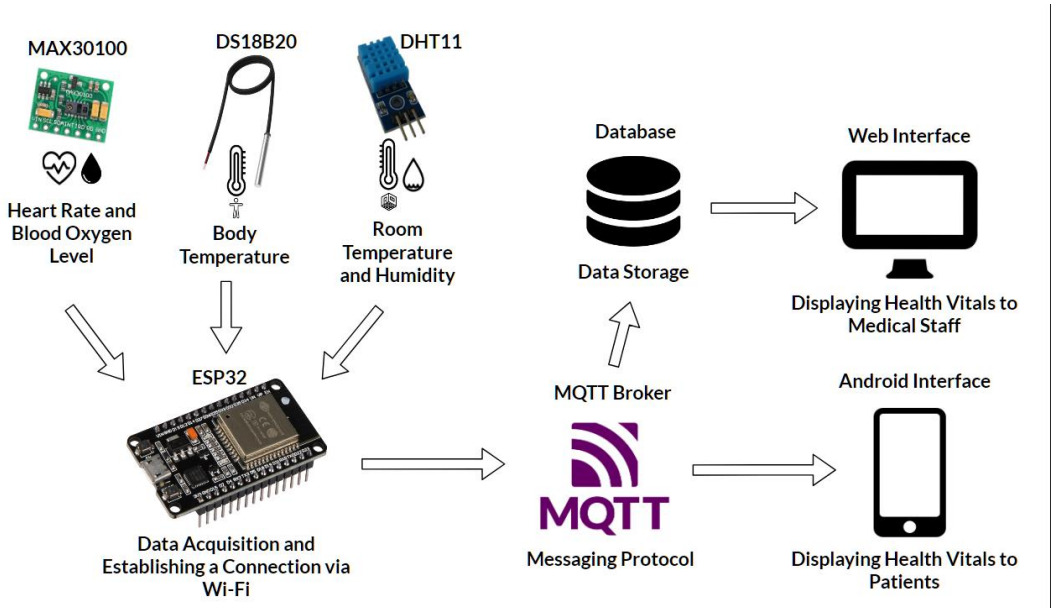

Figure 3. 4 Synoptic diagram of the IoT health monitoring system.

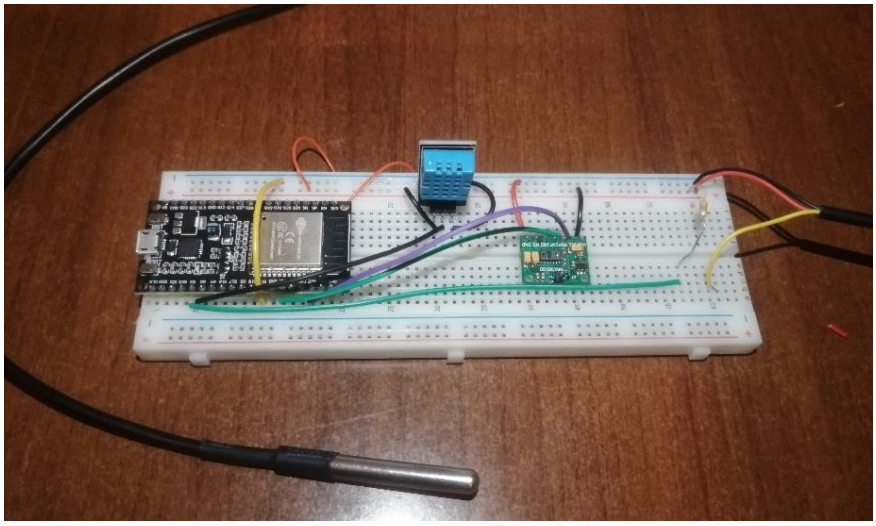

Figure 3. 5 IoT health monitoring system.

3.3.2 Software development:

a. Setting up the MQTT broker:

We start by defining the MQTT topics which is the data collected by the sensors followed by setting up the Wi-Fi and the broker credentials (view Figure 3.6).

```
// MQTT topics
#define BPM_TOPIC "sensor/BPM"
#define SPO2_TOPIC "sensor/SPO2"
#define BODY_TEMPERATURE_TOPIC "sensor/body_temperature"
#define HUMIDITY_TOPIC "sensor/humidity"
#define TEMPERATURE_TOPIC "sensor/temperature"
#define REPORTING_PERIOD_MS     1000
#define ID_TOPIC "patient"

// WiFi and MQTT configuration
const char* ssid = "Honor 8X";
const char* password = "onepiece";
const char* mqtt_server = "192.168.43.56";
const int mqtt_port = 1883;
WiFiClient espClient;
PubSubClient client(espClient);
```

Figure 3. 6 MQTT Broker set up.

b. Publishing the Data:

After the data is collected the ESP32 will publish the sensor data onto the MQTT broker where it will be stored (view Figure 3.7 and Figure 3.8).

```
  // Publish sensor data to MQTT topics
char buffer[8];

dtostrf(bpm, 6, 2, buffer);
client.publish(BPM_TOPIC, buffer, true);

dtostrf(spo2, 6, 2, buffer);
client.publish(SPO2_TOPIC, buffer, true);

dtostrf(body_Temperature, 6, 2, buffer);
client.publish(BODY_TEMPERATURE_TOPIC, buffer, true);

dtostrf(humidity, 6, 2, buffer);
client.publish(HUMIDITY_TOPIC, buffer, true);

dtostrf(temperature, 6, 2,buffer);
client.publish(TEMPERATURE_TOPIC,buffer,true);

dtostrf(id, 6, 2, buffer);
client.publish(ID_TOPIC, buffer, true);
```

Figure 3. 7 code for publishing data to MQTT broker.

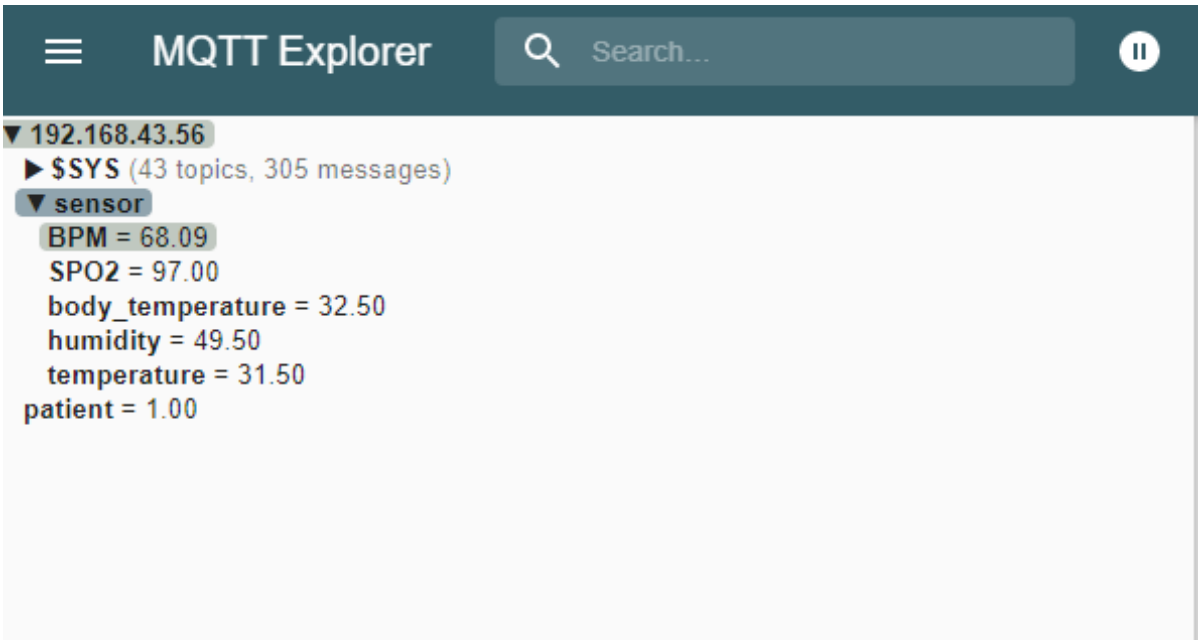

Figure 3. 8 MQTT broker interface.

c. Setting up the data base connection:

we wrote a python to establish the connection between the database and the MQTT broker, which is then followed by the subscription to the desired topics (see Figure 3.9).

```
# MQTT broker settings
broker_address = "192.168.43.56"
broker_port = 1883
BPM_TOPIC = "sensor/BPM"
SPO2_TOPIC = "sensor/SPO2"
BODY_TEMPERATURE_TOPIC = "sensor/body_temperature"
HUMIDITY_TOPIC = "sensor/humidity"
TEMPERATURE_TOPIC = "sensor/temperature"

# MySQL database settings
host = "localhost"
user = "root"
password = ""
database = "patient"
# Create MQTT client instance
client = mqtt.Client()
# Subscribe to the topics
client.subscribe(BPM_TOPIC)
client.subscribe(SPO2_TOPIC)
client.subscribe(BODY_TEMPERATURE_TOPIC)
client.subscribe(HUMIDITY_TOPIC)
client.subscribe(TEMPERATURE_TOPIC)
```

Figure 3. 9 Python code for setiing up the database connection.

d.Arduino code:

The following Figure depicts the Arduino code as a flowchart and will further elaborate it:

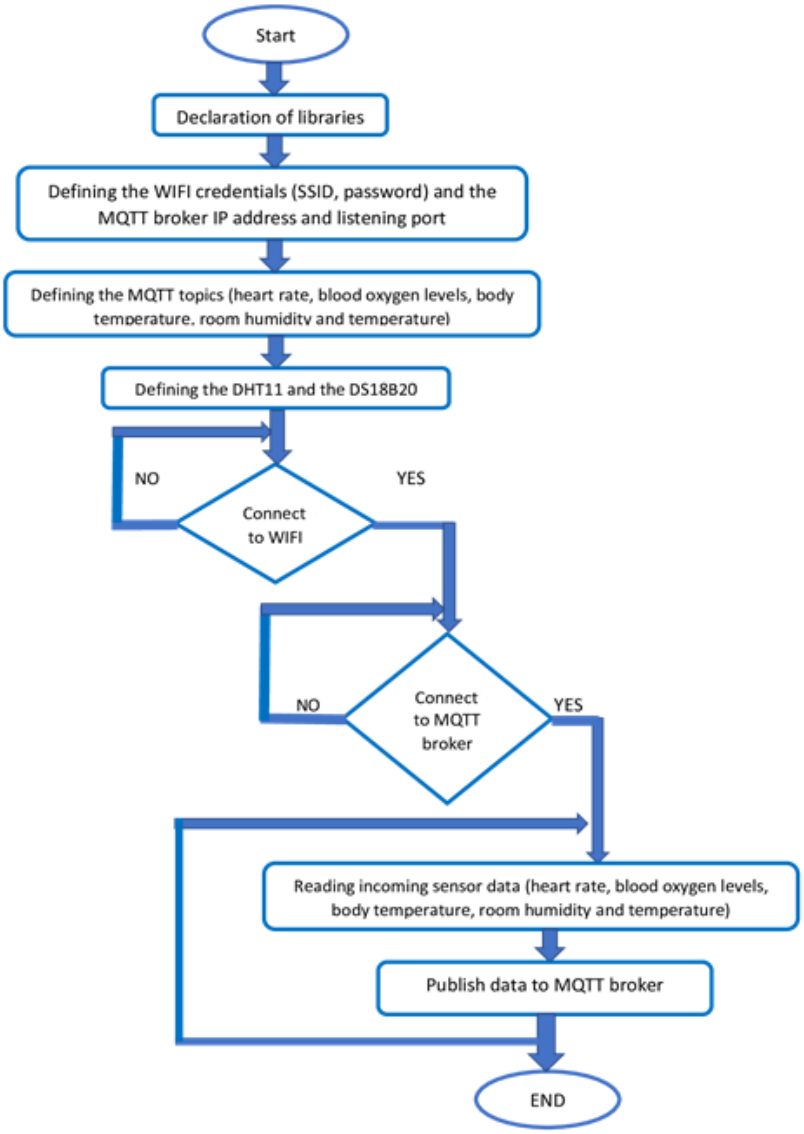

Figure 3. 10 Arduino code flow chart.

3.3.3 Health monitoring Web site:

Our project offers a three-layer architecture (client, web server, and database) IoT platform solution for accessing the patient's vitals. This method not only enables us to examine the data in each instance, but also saves it in a MySQL database for later review and analysis. In order to increase security, we developed a login page that only allows access to medical professionals, as seen below (see Figure 3.11).

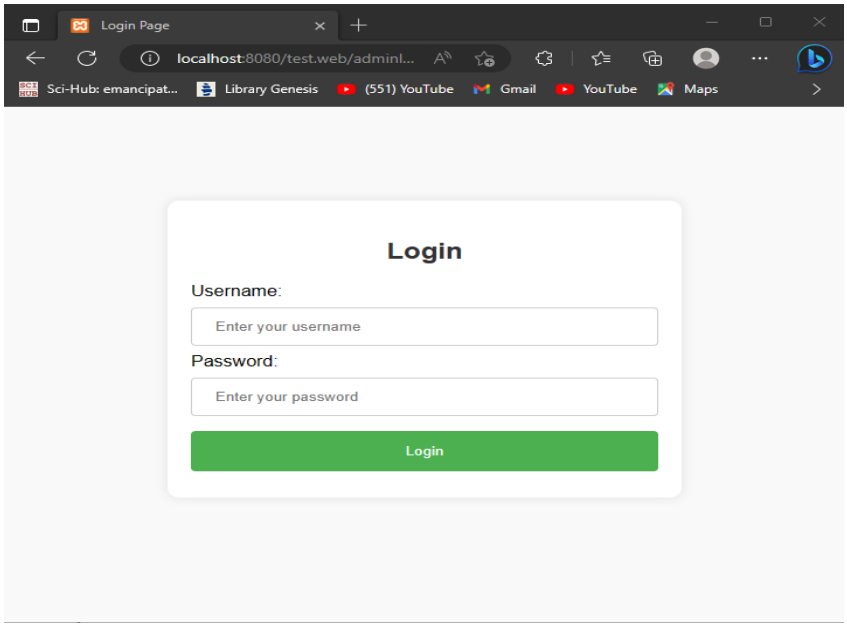

Figure 3. 11 IoT health monitoring login page.

When we click the "Login" button, it will either take us to the home page (see Figure 3.12) or display an error message and request a new login entry if authentication was unsuccessful (see Figure 3.13).

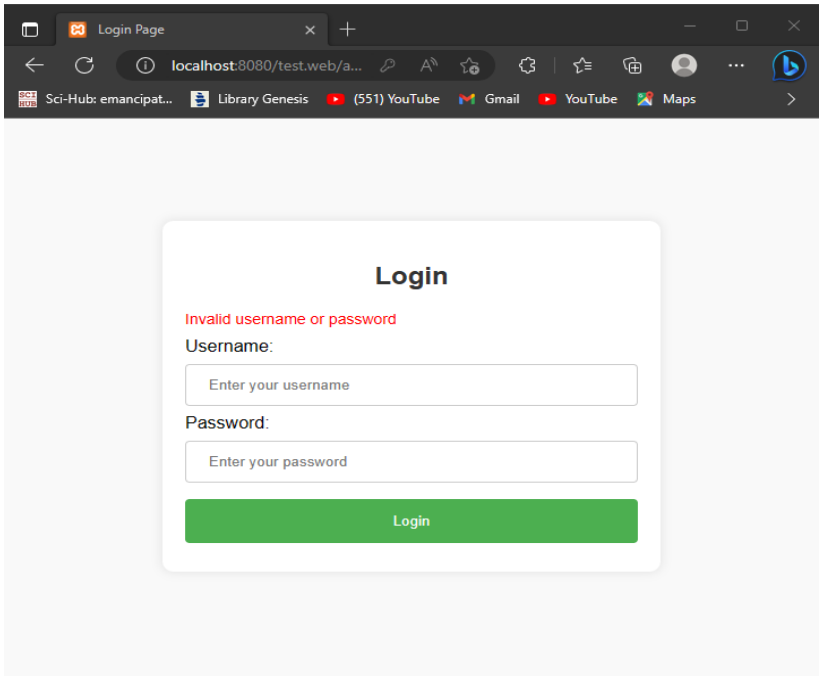

Figure 3. 12 Failed login page.

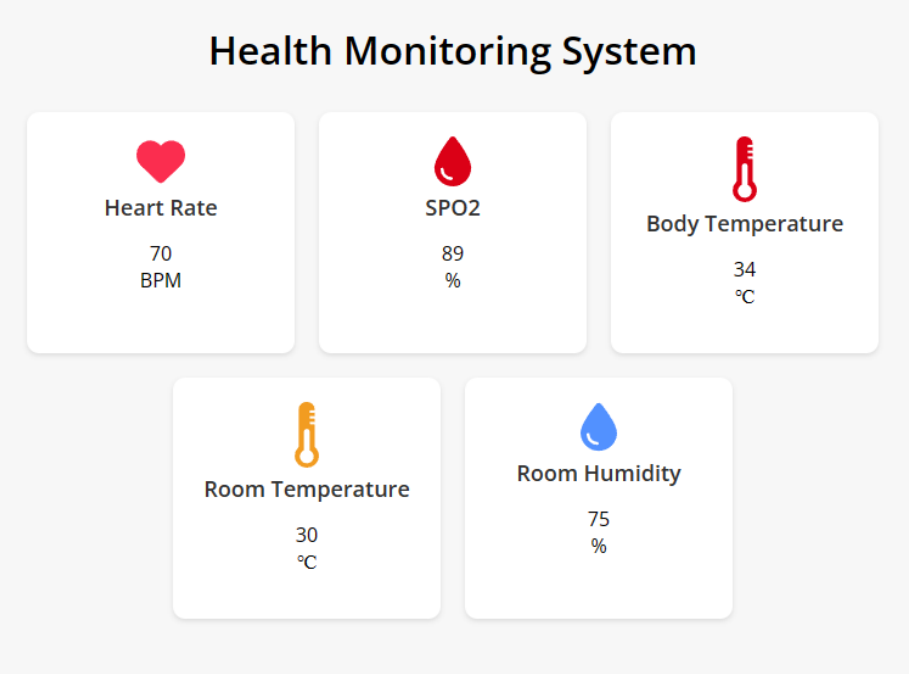

Figure 3. 13 IoT health monitoring main page.

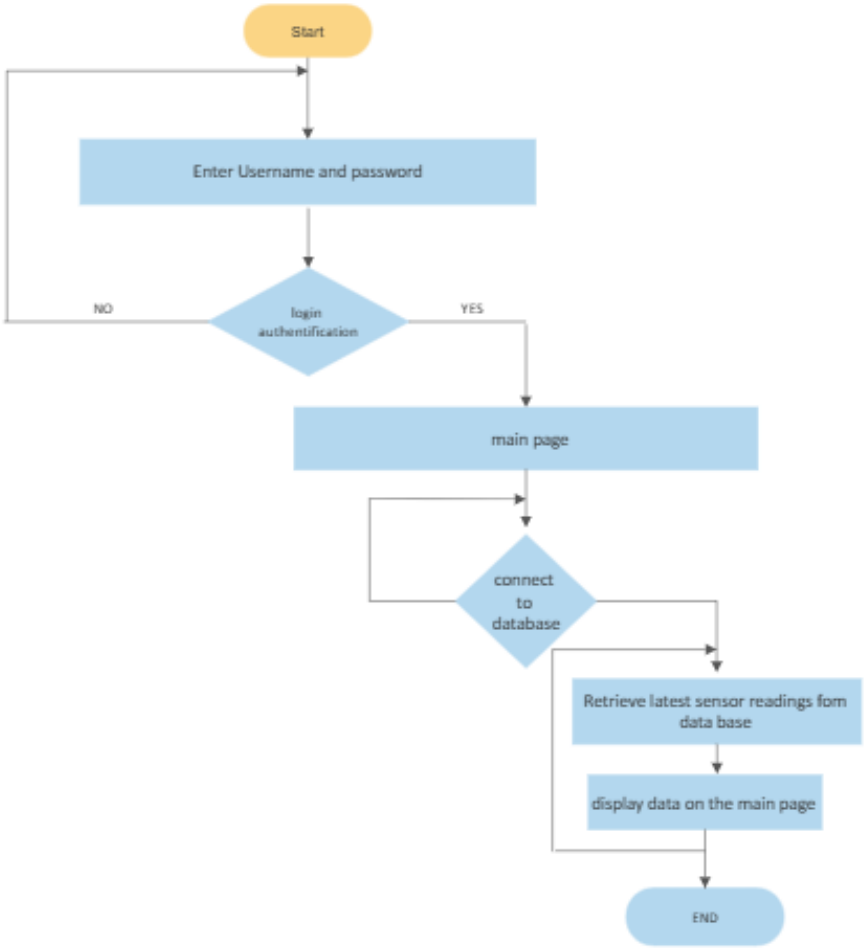

Figure 3. 14 Login authentication and data display flow chart.

To retrieve the data from MQTT broker we made a separate Python script that will be explained in the following flow chart (see Figur3.15).

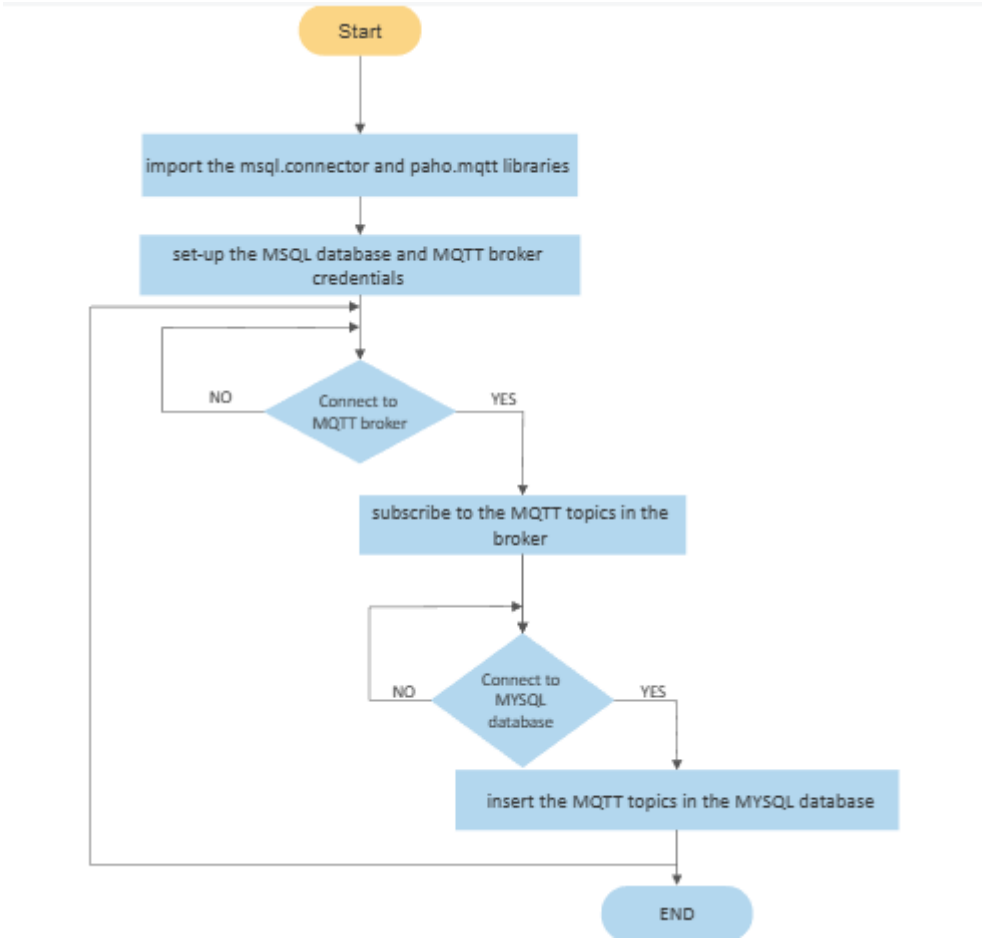

Figure 3. 15 Flow chart for MQTT broker and MYSQL database connection.

3.3.4 Creating a Health monitoring android app:

Employing an android app to connect IoT devices, allows users access to shared information between said devices using a gadget that has become available to most people, smartphones grant us the means of connecting to an embedded device through wireless technologies (Bluetooth or Wi-Fi) making for a cost efficient display or alerting device for our system.

That is why our project also proposes a back-up option for monitoring our patients just in case the web site goes down.

a. Explanation of Kivy framework and its features:

Kivy serves as a free and open-source Python framework that enables the ability of developing cross-platform graphical user interfaces and applications that require touch or multi-touch interfaces, such as mobile or tablet applications. It supports multiple platforms, including Android, iOS, Windows, macOS, and Linux.

It uses a unique declarative language for building user interfaces called KV language. It helps you define the layout and behavior of the interface using a simple and intuitive syntax that is separate from the application logic. Kivy comes with a large set of pre-built widgets and tools, including buttons, labels, sliders, and text inputs, which can be easily customized and extended.

b. Setting up the Development Environment:

- **Installation of necessary tools and software:**

To set up a development environment for Kivy, we need to install Python, and choose an IDE such as PyCharm or Visual Studio Code. Once done we can easily install Kivy by following a tutorial on the official website for Kivy, not all versions of Kivy and Python are compatible with each other so that should be kept in mind.

- **Creating a Kivy project:**

To create a new Kivy project, we create a new folder, using your IDE. We then create a new Python file with the extension .py and a Kivy file by creating a file with the extension .kv

c. Creating User Interface for the IoT App:

- **Designing the app interface using Kivy's widgets:**

Kivy has a variety of pre-built widgets, such as buttons, labels, text inputs, and sliders, these elements can be customized and styled according to your app's needs, these widgets can be placed like building blocks on different layouts, for our project we'll be using the 'GridLayout' which lets up place these different elements on a grid

with a preset size, we'll use two different screens (also called instances) named a 'LoginWindow' and a 'MainWindow' (as seen on the right half of figure 3.17).

.py

```python
class MainWindow(Screen):

    def on_enter(self, *args):
        # instructions to run when this screen is entered
    def callback(self, instance):
        # instructions to run when the button is pressed

class LoginWindow(Screen):

    def callback(self, instance):
        # instructions to run when the button is pressed

class WindowManager(ScreenManager):

    pass

kv = Builder.load_file("my.kv")

class MyMainApp(App):

    #this is where our main app runs

    def build(self):
        return kv

if __name__ == "__main__":
    MyMainApp().run()
```

.kv

```
WindowManager:
    LoginWindow:
    MainWindow:

<MainWindow>:
    name: "main"
    GridLayout:
        cols:2

    # button widget
        Button:
            text: "go to LoginWindow"
            on_press:
                root.callback(self)
            on_release:
                app.root.current = "login"
                root.manager.transition.direction = "right"

<LoginWindow>:
    name: "login"
    GridLayout:
        cols:1

    # button widget
        Button:
            text: "Go to MainWindow"
            on_press:
                root.callback(self)
            on_release:
                app.root.current = "main"
                root.manager.transition.direction = "left"
```

Figure 3. 16 An example of a simplified code for Python on the left and Kivy on the right.

We define three classes 'MainWindow', 'LoginWindow ', and 'WindowManager', which inherit from the 'Screen' and 'ScreenManager' classes provided by the Kivy framework (as seen on the left half of figure 3.x). These classes permit our app to have two separate screens. We then load a Kivy language file using the Builder.load_file() method, and create an instance of the 'MyMainApp' class, which inherits from the 'App' class provided by Kivy. The 'MyMainApp' class has a build method that returns the loaded Kivy language file. Finally, the `if __name__ == '__main__'` statement checks if the script is being run as the main program and runs 'MyMainApp()'.

- **Implementing user interactions and events:**

In the Kivy file we introduce the two screens: 'MainWindow' and 'LoginWindow', both contained within a 'WindowManager' screen manager.

The ‘MainWindow’ screen contains a GridLayout with two columns, the columns in the GridLayout tell us how many widgets (labels, images, buttons…) can be put in a single line. In this layout we have a button widget that can be pressed, it calls the 'callback' function in the ‘MainWindow’ class, and then sets the current screen to 'login' and applies a rightward transition (see the button widgets on the right side of figure 3.x).

The ‘LoginWindow’ screen contains a GridLayout with one column, and a button widget that when pressed, calls the 'callback' method in the ‘LoginWindow’ class, then sets the current screen to 'main' and applies a leftward transition.

d. Integrating IoT Functionality:

- **Implementing IoT protocols (MQTT Broker):**

For our project we need to implement an MQTT broker, we can use the ‘paho.mqtt.client’ library, to install it we simply run the command 'pip install paho-mqtt' in the terminal, to connect the client we first import the Paho MQTT client:

```
import paho.mqtt.client as mqtt
```

We then set up the IP address and port finally connecting to the broker and starting the client with ‘client.loop_start()’:

```
mqtt_broker = 'ip_address'

mqtt_port = port

client.connect(mqtt_broker, mqtt_port)

client.loop_start()
```

- **Sending and receiving data:**

After connecting to our broker we can set up the functions for when we connect to the MQTT broker and when we receive messages:

```
client.on_connect = on_connect

client.on_message = on_message
```

The 'on_connect' function is used to subscribe to MQTT topics:

```
def on_connect(client, userdata, flags, rc):

      # Subscribe to MQTT topics

      client.subscribe('topic_name')
```

While the 'on_message' function it's mainly used to receive messages:

```
def on_message(client, userdata, message):

      topic = message.topic

      payload = message.payload.decode('utf-8')

      print('Received message from topic' + str(topic) ':' +
str(payload))
```

- **Displaying data in the app interface:**

For our Kivy app, some changes need to be done to display data in the app interface, to connect to the MQTT broker, we need to start the connection inside our build function:

```
class MyMainApp(App):

def build(self):

      self.client.connect(mqtt_broker, mqtt_port)

      self.client.loop_start()
```

```
        self.client.on_connect = self.on_connect
        self.client.on_message = self.on_message
```

The ‘on_connect’ function to subscribe to MQTT topics becomes as follows:

```
def on_connect(self, client, userdata, flags, rc):
        # Subscribe to MQTT topics
        self.client.subscribe('topic_name')
```

The ‘on_message’ function now will be used to update labels, to update a label having an id ‘label1’ that was set up in the .kv file, we use the variable ‘main_screen.ids.label1.text’ which updates the label with the id ‘label1’ in the screen with the name ‘main’ :

```
def on_message(self, client, userdata, message):
        #to access labels in the main page we use root.get_screen:
        main_screen = self.root.get_screen('main')
        payload = message.payload.decode('utf-8')
        main_screen.ids.label1.text = str(payload)
```

e. Deploying the IoT App to Android:

- **Creating an APK file for the app:**

To create a package for android using the python-for-android project, the Buildozer tool is used, it also exists to simplify and automate the entire process. Buildozer is recommended as it is one of the easiest ways to make a full APK (Android

Package), once done, Kivy applications can be released on an Android market such as the Play store.

Buildozer builds an android app from the Kivy file automatically, it downloads and sets up all the prerequisites for python-for-android, it then builds an APK that can be installed on androids. Buildozer works only in Linux, macOS or in Windows using a Linux emulator. To install Buildozer one can follow the instructions provided in the official Buildozer website that require you to copy and paste command lines one by one until the installation is complete. Detailed guides can easily be found online.

To use Buildozer, we need to make sure to first install all of its dependencies, then we can start by configuring the buildozer.spec file. First, we need to create the buildozer.spec file in the root directory of our Kivy project, to do that we simply need to open the root directory of our Kivy project with the Linux terminal then enter the command 'buildozer init', after the buildozer.spec file is created, we can then edit it to include information such as the name and version of the app, minimum Android SDK version, and any additional Python packages required, such as the ‘paho.mqtt.client’ library to connect to the MQTT broker in our case. Once done, you we can run the command 'buildozer -v android debug', this stage can take up to an hour, when the building is done, you can find your .apk file in a bin folder in the directory of your Kivy project, which you can then install on your Android device.

the following flow chart will show the logic behind the development of our app (see Figure 3.17).

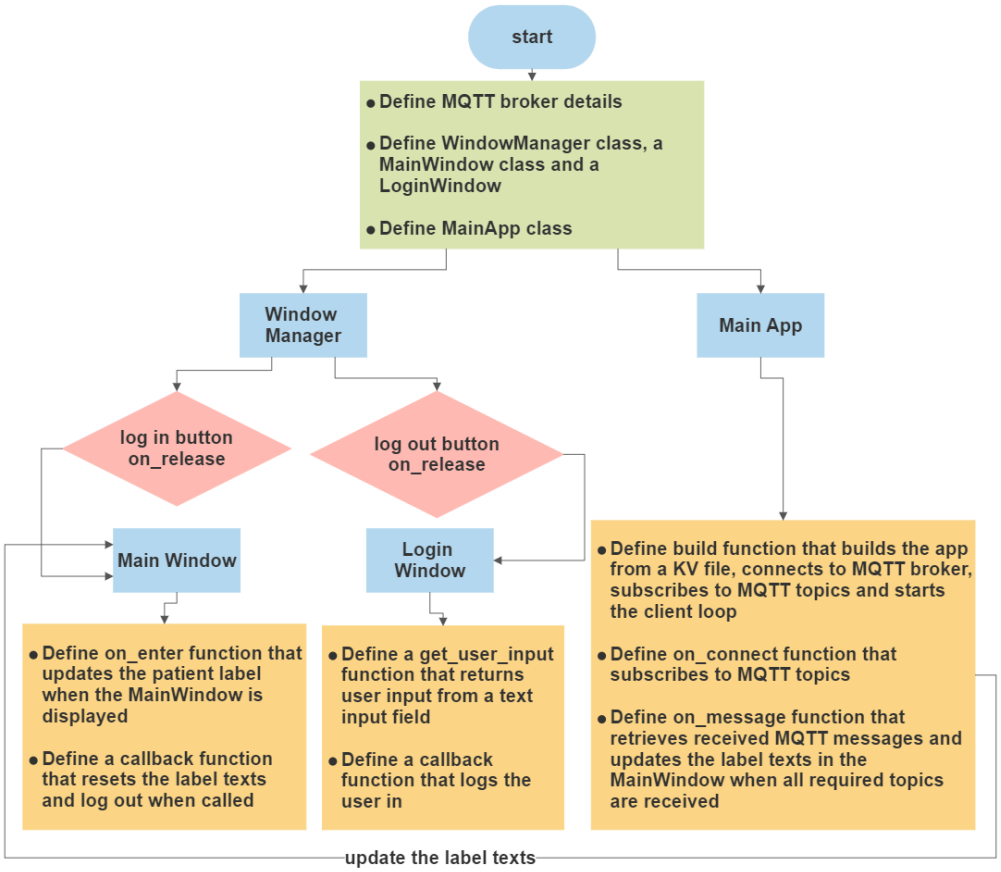

Figure 3. 17 android app flowchart.

Once the Kivy app properly works as intended, it can then be exported as an android package file. This can easily be done using the Buildozer tool, which is a Linux tool that helps you set up the name and version of your app, minimum Android version, and any additional Python packages required, such as the library required to connect to the MQTT broker in our case.

Upon opening the app, users will be prompted to enter an ID provided by the hospital to have access to the patient's data.

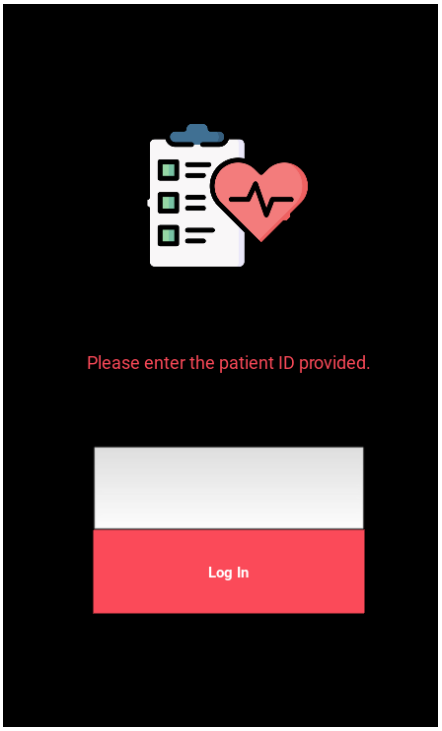

Figure 3. 18 IoT health monitoring android app login page.

Once logged in, using the ID. The main page opens, showing the different sensor readings.

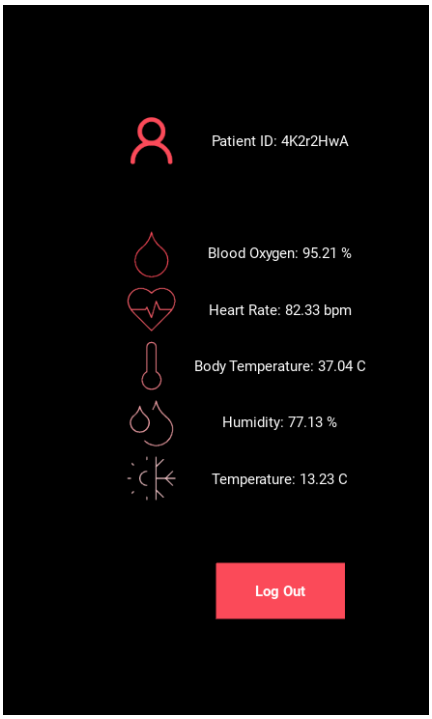

Figure 3. 19 IoT health monitoring android app main page.

3.4 Conclusion:

This chapter summarizes all the steps we did to develop our final project, which is a health monitoring system based on IoT. The approach we used was to take on each part by itself, by first creating our health monitoring system with the different sensors (MAX30100, DHT11 and DS18B20) and the Esp32 microcontroller, and then we implemented the IoT side of things by publishing the data to the MQTT broker and developed the android app with the website.

General Conclusion

Throughout the course of this dissertation, we presented the different steps of the design and implementation of an IoT based health-monitoring system.

The first chapter is dedicated to the study of the human cardiovascular system and pulse oximetry. We began the chapter by discussing a general understanding of health monitoring, the human heart, and the many vital signs (heart rate, blood oxygen level, and body temperature), and we concluded by discussing a general understanding of the notion of pulse oximetry.

The second chapter is devoted to the general understanding of the concept of “Internet of Things”. We began the chapter with an overview of the Internet of Things and its history, followed by a discussion of the various architectures and communication protocols. Finally, we focused on the MQTT protocol, which we utilized in our project, to wrap up the chapter.

The third chapter discusses how our project was carried out. We started by providing a general overview of the various text editors and softwares we used for coding. Then, we covered the various programming languages required for our project. Finally, we presented the microcontroller and sensors we used, along with flowcharts that explained the system's logic, to wrap up the chapter.

We gained a lot of knowledge via this project, including full stack web development, by building a website for our health monitoring system utilizing PHP and MySQL for the back end, HTML, and CSS for the front end. In addition, we gained Python skills by creating an Android app. Last but not least, we hope that our project will be able to reduce the amount of work that nurses and doctors have to do across various hospitals throughout the world.

References

[1] E. Kanasi, S. Ayilavarapu, and J. Jones, “The aging population: demographics and the biology of aging,” *Periodontol. 2000*, vol. 72, no. 1, pp. 13–18, 2016, doi: 10.1111/prd.12126.

[2] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers, “A review of wearable sensors and systems with application in rehabilitation.,” *J. Neuroengineering Rehabil.*, pp. 1–17, 2012.

[3] M. Talal *et al.*, “Smart Home-based IoT for Real-time and Secure Remote Health Monitoring of Triage and Priority System using Body Sensors: Multi-driven Systematic Review,” *J. Med. Syst.*, vol. 43, no. 3, 2019, doi: 10.1007/s10916-019-1158-z.

[4] Glenmed Medical Solutions, “The history of patient monitoring systems.” https://www.glenmedsolutions.com/2017/09/21/history-patient-monitoring-systems/

[5] D. Le, C. Van Le, J. G. Tromp, and G. N. Nguyen, Eds., *Emerging Technologies for Health and Medicine*. Wiley, 2018. doi: 10.1002/9781119509875.

[6] N. I. Mathi and J. J. Sree, “TELEPRESENCE IN MEDICINE – AN OVER VIEW,” vol. 4, no. 3, pp. 64–68, 2016.

[7] S. K., P. A. S., P. U., P. S., and S. P. V., “Patient health monitoring system using IoT,” *Mater. Today Proc.*, Jun. 2021, doi: 10.1016/j.matpr.2021.06.188.

[8] N. Dilawar, M. Rizwan, F. Ahmad, and S. Akram, “Blockchain: Securing internet of medical things (IoMT),” *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 1, pp. 82–89, 2019, doi: 10.14569/IJACSA.2019.0100110.

[9] C. BD, *Human Anatomy, Volume 1: Regional and Applied Dissection and Clinical: Upper Limb and Thorax*, 6th ed. jaypee Brothers Medical Publishers, 2014.

[10] N. C. I. (n.d.), "Heart rate," *NCI Dictionary of Cancer Terms*. https://www.cancer.gov/publications/dictionaries/cancer-terms/def/heart-rate

[11] C. (n.d.), "Cœur en bonne santé." https://www.cardiosecur.com/fr/magazine/articles-specialises/notions-de-base-sur-le-coeur/coeur-en-bonne-sante

[12] G. M. Fouad and I. Abdelrahim, "Système de télésurveillance du rythme cardiaque et de la température corporelled'un patient," 2020.

[13] N. R. T. S. J. S. S. Aeddula, *Physiology, Arterial Pressure Regulation.* StatPearls Publishing, 2022.

[14] N. Zoremba, C. Brãlls, V. Thiel, A. Rãhl, and R. Rossaint, "Pulse oximetry during intraaortic balloon pump application," *Acta Anaesthesiol. Scand.*, vol. 55, no. 3, pp. 322–327, 2011, doi: 10.1111/j.1399-6576.2010.02388.x.

[15] J. Smith, "Understanding pulse oximetry.," *Anaesth. Intensive Care*, vol. 20, no. 2, pp. 255–256, 1992.

[16] V. G. Nasr and J. A. DiNardo, "Pulse oximetry," *Pediatr. Rev.*, vol. 40, no. 11, pp. 605–608, 2019, doi: 10.1542/pir.2018-0123.

[17] P. King, "Design Of Pulse Oximeters," *IEEE Eng. Med. Biol. Mag.*, vol. 17, no. 3, pp. 117–117, 2005, doi: 10.1109/memb.1998.677180.

[18] J. H. M. (n.d.), "Pulse Oximetry." https://www.hopkinsmedicine.org/health/treatment-tests-and-therapies/pulse-oximetry

[19] J. Mohammed, C. H. Lung, A. Ocneanu, A. Thakral, C. Jones, and A. Adler, "Internet of things: Remote patient monitoring using web services and cloud computing," *Proc. - 2014 IEEE Int. Conf. Internet Things, iThings 2014, 2014 IEEE Int. Conf. Green Comput. Commun. GreenCom 2014 2014 IEEE Int. Conf. Cyber-*

Physical-Social Comput. CPS 20, no. iThings, pp. 256–263, 2014, doi: 10.1109/iThings.2014.45.

[20] B. Mishra and A. Kertesz, "The use of MQTT in M2M and IoT systems: A survey," *IEEE Access*, vol. 8, pp. 201071–201086, 2020, doi: 10.1109/ACCESS.2020.3035849.

[21] S. Li, L. Da Xu, and S. Zhao, "The internet of things: a survey," *Inf. Syst. Front.*, vol. 17, no. 2, pp. 243–259, 2015, doi: 10.1007/s10796-014-9492-7.

[22] T. Power, "Leverege_Intro_to_IoT_eBook".

[23] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010, doi: 10.1016/j.comnet.2010.05.010.

[24] A. Bansal, M. K. Ahirwar, and P. K. Shukla, "A Survey on Classification Algorithms Used in Healthcare Environment of the Internet of Things," *Int. J. Comput. Sci. Eng.*, vol. 6, no. 7, pp. 883–887, Jul. 2018, doi: 10.26438/ijcse/v6i7.883887.

[25] P. Dow, Colin;Lea, *Mastering IOT: Build modern IoT solutions that secure and monitor your IoT infrastructure*. Packt Publishing, 2019.

[26] Maxim Integrated, "MAX30100 Pulse Oximeter and Heart-Rate Sensor IC for Wearable Health [Data sheet].," pp. 1–29, [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX30100.pdf

[27] Aosong Electronics Co. Ltd. (n.d.)., "DHT11 Technical Data Sheet [Data sheet]," *Mouser Electron.*, [Online]. Available: https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

[28] Maxim Integrated, "DS18B20 Programmable Resolution 1-Wire Digital Thermometer [Data sheet].", [Online]. Available: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

[29] IoT Tech Trends, "What is ESP32?" https://www.iottechtrends.com/what-is-

esp32/.

[30] Rui Santos and S. Santos, “ESP32 Web Server with Arduino IDE,” pp. 6–13, 2018.

[31] D. V and A. Kumar Singh, “Internet of Things: a Survey of the Advancements,” *Int. J. Eng. Technol.*, vol. 7, no. 3.12, p. 255, 2018, doi: 10.14419/ijet.v7i3.12.16036.

[32] A. D. Pathaka and J. V. Tembhurne, “Internet of Things: A Survey on IoT Protocols,” *SSRN Electron. J.*, pp. 483–487, 2018, doi: 10.2139/ssrn.3168575.

[33] M. Guizani, *The industrial internet of things*, vol. 33, no. 5. 2019. doi: 10.1109/MNET.2019.8863716.

[34] A. Del Sole, *Visual Studio Code Distilled*. Apress, 2021. doi: 10.1007/978-1-4842-6901-5.

[35] Arduino. (n.d.). Environment, “Arduino Software (IDE) V1.” https://docs.arduino.cc/software/ide-v1/tutorials/Environment

[36] Kivy. (n.d.)., “Kivy documentation - Stable version.” https://kivy.org/doc/stable/

[37] M. Delisle, *Effective MySQL Management*.

[38] IoT for Beginners. (n.d.), “Mosquitto MQTT Broker Introduction.” https://iot4beginners.com/mosquitto-mqtt-broker-introduction/#:~:text=Mosquitto is a lightweight open source message broker,of carrying out messaging using a publish%2Fsubscribe model.

[39] Mozilla. (n.d.)., “Introduction to HTML,” *Mozilla Developer Network*. https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML

[40] Mozilla. (n.d.), “CSS (Cascading Style Sheets),” *Mozilla Developer Network*. https://developer.mozilla.org/en-US/docs/Web/CSS

[41] The PHP Group. (n.d.), “PHP: Hypertext Preprocessor,” . *PHP Manual*. https://www.php.net/manual/en/

[42] L.-S. LIFE-STYLE ACADEMY & Python [ACADEMY, *Python: The Complete Python*

Quickstart Guide (For Beginners) (Python, Python Programming, Python for Dummies, Python for Beginners). 2015.

[43] MySQL AB, *MySQL administrator's guide and language reference*. MySQL Press, 2006.