

DETECTION OF FEBRILE ILLNESS AND ITS MONITORING USING IOT TECHNOLOGY

**A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of**

MASTER OF SCIENCE

in Biotechnology

**by
Gamze TAMUR KAYA**

**July 2019
İZMİR**

We approve the thesis of **Gamze TAMUR KAYA**


Examining Committee Members:



Asst. Prof. Dr. Hüseyin Cumhuri TEKİN
Department of Bioengineering, İzmir Institute of Technology



Assoc. Prof. Dr. Yusuf Murat ERTEN
Department of Computer Engineering, İzmir Institute of Technology

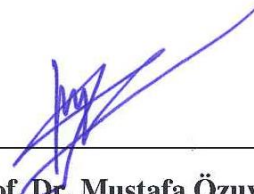


Prof. Dr. Vildan MEVSİM
Internal Medicine Department of Family Medicine, Dokuz Eylül University

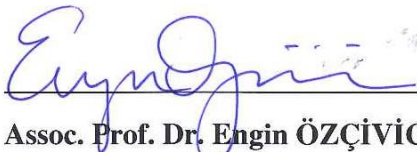
17 July 2019



Asst. Prof. Dr. Hüseyin Cumhuri TEKİN
Supervisor, Department of Bioengineering
İzmir Institute of Technology



Asst. Prof. Dr. Mustafa Özuysal
Co-Supervisor, Department of
Computer Engineering
İzmir Institute of Technology



Assoc. Prof. Dr. Engin ÖZÇİVİCİ
Head of the Department of Biotechnology
and Bioengineering

Prof. Dr. Aysun SOFUOĞLU
Dean of the Graduate School of
Engineering and Sciences

ACKNOWLEDGMENTS

Firstly, I would like to thank my supervisor Asst. Prof. Dr. Hüseyin Cumhuri TEKİN for his leadership, helpful recommendations and open-minded perspective during my master program. He always motivated me every stage of my thesis. Additionally, I am also grateful to members of the Laboratory of Biomedical Micro and Nanosystems for making time enjoyable. Especially, I would like to thank Yekta GÜLMEZ for every moment of my master program.

I would like to thank also İzmir Institute of Technology (2017IYTE61) for the financial support.

I must express my gratitude to my mother and my dad, they are always with me every decision in my life. I would like to thank their endless love. I am so lucky to have such a wonderful sister. Last but not least, I would like to thank to my beloved husband Melih, for being always next to me and for his understanding and making me smile.

ABSTRACT

DETECTION OF FEBRILE ILLNESS AND ITS MONITORING USING IOT TECHNOLOGY

Measuring and monitoring body temperature has importance for disease diagnosis. It could also help following the course of the disease. Extensive number of body temperature monitoring applications are designed and developed. These applications could be named as healthcare applications aiming to provide convenience to the users. The Internet of Things technology is popular in healthcare applications by offering remote and real-time monitoring. In this project, a telemedicine platform facilitating the diagnosis and monitoring of febrile illnesses is designed. A user-friendly platform is implemented using software components. Message Queuing Telemetry Transport (MQTT), which is a communication protocol, is used for the communication between devices and monitoring system. A broker is used to transmit measured body temperature data from a device to the cloud server. The performance of the broker is evaluated with thousands of generated data packets. It is showed that the platform can handle data requests with a high throughput, *i.e.*, up to 40000 packets/s. The monitoring system is designed as an interactive user interface. For this purpose, Telegram is adapted to the requirements of the platform by using an open-source library of Telegram BOT API. Hence, a user is able to access the measurement results and also control the measurements using instant messaging via Telegram interface. Moreover, this proposed platform could eliminate the use of separate healthcare applications for different diseases by using just a single instant messaging program for all different applications.

ÖZET

NESNELERİN İNTERNETİ TEKNOLOJİSİNİ KULLANARAK ATEŞLİ HASTALIKLARIN TEŞHİSİ VE TAKİBİ

Vücut ısısının ölçümü ve anlık takibi hastalık teşhisi için oldukça büyük öneme sahiptir. Ayrıca hastalığın seyrini izlemede yardımcı olmaktadır. Çok sayıda vücut ısısının izlenmesine yardımcı olan uygulamalar tasarlanmış ve geliştirilmiştir. Bu uygulamalar, kullanıcılara kolaylık sağlamayı amaçlayan sağlık uygulamaları olarak adlandırılabilir. Nesnelerin İnterneti teknolojisi, uzaktan ve gerçek zamanlı izleme imkan tanıdığı için sağlık uygulamaları alanında önem kazanmaktadır. Bu projede, ateşli hastalıkların teşhis ve takibini kolaylaştıran bir teletıp platformu tasarlanmıştır. Yazılım bileşenleri kullanılarak kullanıcı dostu bir platform geliştirilmiştir. Nesnelerin İnterneti haberleşme protokolü olarak ise Message Queue Telemetry Transport (MQTT) kullanılarak cihazlar ile sistemin haberleşmesi sağlanmıştır. Cihaz üzerinde ölçülen vücut ısısı verileri bulutta bulunan bir sunucuya iletilmek için bir aracı kullanılmıştır. Aracının performansı yayınlanan binlerce paket ile ölçülmüştür. Platform veri taleplerine yüksek verimlilik ile 40000 paket/sn performansa kadar cevap verebilmiştir. Bu platformun arayüz tasarımı için kullanıcı dostu bir sohbet robotu kullanılmıştır. Bu amaçla açık kaynaklı bir kütüphane olan Telegram BOT API kullanılarak Telegram platform ihtiyacına göre uyarlanmıştır. Böylelikle, kullanıcı Telegram arayüzünü kullanarak anlık mesajlaşma ile ölçüm sonuçlarına erişebilir ve ölçümleri kontrol edebilir. Ayrıca bu önerilen, tüm farklı sağlık uygulamaları için tek bir mesajlaşma programı kullanılmasına olanak sağlayarak, farklı hastalıklar için ayrı bir uygulamanın kullanılması ihtiyacını ortadan kaldırabilir.

TABLE OF CONTENTS

LIST OF FIGURES.....	viii
CHAPTER 1. INTRODUCTION	1
CHAPTER 2. STATE OF ART	4
2.1. Telemedicine	7
2.2. Internet of Things Technology	8
2.2.1. Architecture of Internet of Things Technology.....	11
2.2.2. Internet of Things Communication Protocols	12
2.2.3. Message Queuing Telemetry Transport.....	15
2.3. Chatbots.....	17
2.3.1. Chatbot History	17
CHAPTER 3. MATERIALS	19
3.1. Arduino and Sensor Design	19
3.2. Computer Requirements	20
3.3. Node.js Engine	21
3.4. Remote Dictionary Server.....	21
3.5. Telegram	21
3.6. Mosca MQTT Broker	22
CHAPTER 4. IMPLEMENTATION	23
4.1. Arduino Structure	24
4.2. Mosca MQTT Broker	26
4.3. Redis Database	26
4.4. Telegram Router.....	27
4.4. Telegram User Interface	32
CHAPTER 5. PERFORMANCE EVALUATION.....	34
5.1. Mosca MQTT Broker Performance Evaluation	34

5.2. Redis Database Performance Evaluation.....	37
CHAPTER 6. DISCUSSION.....	39
CHAPTER 7. CONCLUSION.....	42
REFERENCES.....	44

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
Figure 1. Normal body temperature variance in one day at different age ranges.....	2
Figure 2. Working principle behind the Internet of things.....	9
Figure 3. IoT applications market share by 2025	10
Figure 4. Convergence of Internet of Things paradigm scheme.....	11
Figure 5. Architecture of Internet of Things	12
Figure 6. Reference of Internet of Things' protocol stack.....	13
Figure 7. Message queuing telemetry transport protocol model	16
Figure 8. Quality of Service Levels.....	16
Figure 9. Used microcontroller	20
Figure 10. Architecture of the system.....	23
Figure 11. Flowchart of a microcontroller which has a body temperature sensor.....	25
Figure 12. Hash table structure for users in Redis database.	26
Figure 13. The measured data is stored in hash table and detailed data information	27
Figure 14. Visualized sequence diagram includes start measurement operation.	29
Figure 15. A sequence diagram that show a user how to get the results from the system.....	29
Figure 16. Sequence diagram of login operation.	30
Figure 17. A sequence diagram of when user want to find a device	31
Figure 18. Telegram Menu	32
Figure 19. Throughput performance as the broker receive published messages.....	36
Figure 20. Throughput performance of the broker that receives published messages from one Arduino client.....	36
Figure 21. Throughput performance of HSET Query Requests.....	37

CHAPTER 1

INTRODUCTION

The physiological situation of body can be understood with four vital signs which are body temperature, heart rate, blood pressure and respiration rate [3, 4] Generally, the temperature is one of the important indicator and alteration of body temperature affect other vital signs. The mostly used measurement types are oral, axillary, rectal or tympanic and the temperature value depends on gender, age and wherein the body temperature taken [5]. The body temperature is expected to be between 36 and 38°C. Normal body temperature changes, which were taken from people with different age range; 5, 35, 70, are illustrated in Figure 1. As can be seen from the figure, temperature differences were obtained between every hour for 24 hours. The important point in the figure is that the circadian rhythmicity moderately consistent but activities were not utilized. As illustrated; the average body temperature is almost similar, but small differences were observed for different age range.

Elevated body temperature will denature proteins and this will cause death [6]. Increasing or decreasing temperature of body affects the body functions differently. If temperature increases, the body will try to decrease it with blood flow and sweating is observed. On the other hand decreasing of body temperature causes blood vessels to be tight and muscle tremor will start [7].

It can be said that, instability of body temperature is not always a disease, also it could be a symptom of serious illness. The most known febrile illnesses are malaria, pneumonia, typhoid fever, invasive nontyphoidal salmonellosis and influenza's outcome is fever [8]. Alteration of body temperature at serious diseases is followed only by continuous monitoring.

The most common respiratory system diseases are pneumonia. According to World Health Organization (WHO) [9], pneumonia occurs when pus and fluid appear in alveoli that caused breathing becomes painful and oxygen intake decreases. According to researches, approximately one million children dead because of the pneumonia in 2015.

The symptoms of the disease are cough, breathing problems and fever. It is observed that body temperature is a critical symptom if a patient has pneumonia, since it could help to diagnose and monitor of the treatment [11].

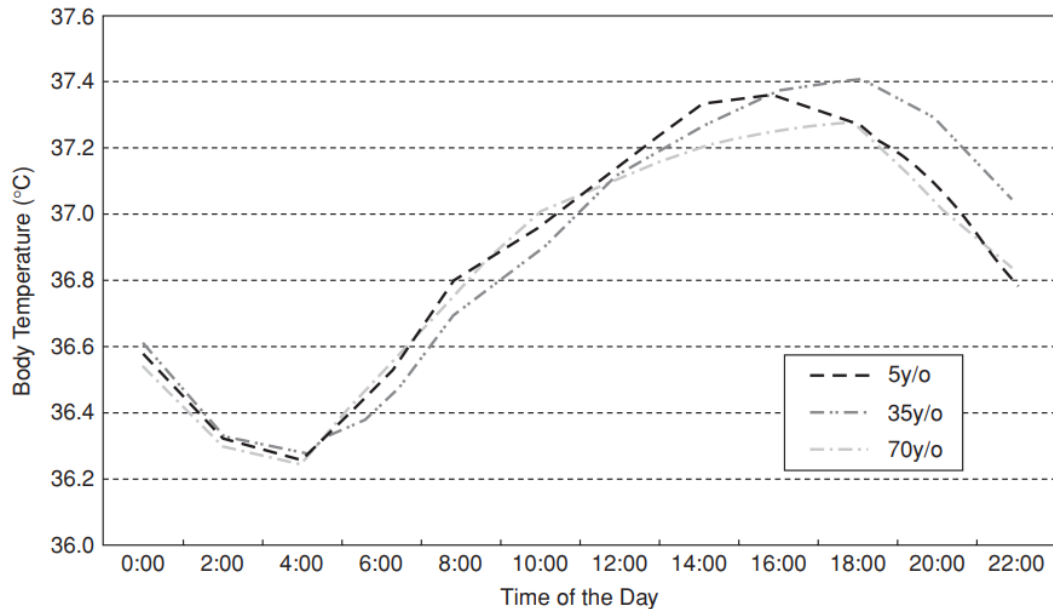


Figure 1. Normal body temperature variance in one day at different age ranges (Source: [10]).

Another febrile illness, malaria, which is infected by *Anopheles* mosquitoes, is an acute illness. It starts with fever, chill and headache. In 2017, about 435 000 people died due to this disease [12]. Meningitis is another example of febrile illness, which starts on the meninges and the brain and spinal cord are covered by illness [13]. In this respect, there are thousands of illness like these examples, a system needs to be developed for detection and monitoring of febrile illnesses in a digital platform.

Detection and monitoring of febrile illnesses should be encouraged by technology. Measuring body temperature rapidly changes; like mercury thermometer to infrared thermometer. Digital industry produces new instruments and measuring body temperature becomes easier than early times. Additionally, by these inventions results are obtained immediately and these instruments have own softwares to check results accurately. Advancements of technologies on measuring body temperatures provide to get patient follow-up by instruments, hardware and software.

Advancements of technologies on measuring body temperatures provide to get patient follow-up by instruments, hardware and software.

Telemedicine is a field that deals with healthcare improvements and new technological developments and they are merged with Internet of Things (IoT). Complex telemedicine systems help patient since doctor or medical staff could observe their illnesses remotely. In this thesis project, a heterogeneous system was developed with IoT technology. Devices are controlled by a broker that works with Message Queueing Telemetry Transport Protocol (MQTT) and published data redirect to the cloud server and a database. The system is integrated with chatbot that is used Telegram BOT API in the cloud server. Main goal of used developed chatbot is being a middleware between user-to-user and user-to-server.

To understand how the generated server handle thousands published messages, experiment study of this project based on two part: the MQTT broker and the database performance. The purpose of the performance evaluation of the broker is to observe the effect on the broker as the number of consequent messages sent to the system. The instant body temperature could be displayed without any delay when number of published messages are low. Nevertheless, when there is too much message traffic on the broker, the displayed results of the desired moment may be delayed up for a few minutes. At this point, the broker performance is observed with throughput calculations.

For tests, three different size data packets, *i.e.*, 2 bytes, 16 bytes and 128 bytes, were used. Number of packets were increased orderly for each turns and measurement of the broker and database performances were conducted.

Organization of the thesis is as following. State of the art is mentioned in Chapter 2. Materials are given in Chapter 3. Implementation of the proposed platform is explained in Chapter 4. Chapter 5 contains performance evaluation of the platform and discussion. Chapter 6 includes summary and outlook of the thesis.

CHAPTER 2

STATE OF ART

New heterogeneous systems are studied to aid patients and doctors for monitoring and detection of illnesses. For this purpose, different technologies are implemented together, such as microcontrollers (e.g. Arduino boards), IoT devices, IoT protocols (e.g. MQTT), web services (e.g. ThingSpeak), etc..

A wearable sensor could be good example of a complex systems. According to one study [14], the study based on measurement of body temperature and its monitoring. Measurement was done on the baby's foot. In the system, Arduino ESPresso Lite microcontroller was selected to send the measured data to a cloud server through the Wi-Fi connection. Patient's parents must have an Android cell phone to follow up their baby's temperature. Blynk user interface was used in Android cell phone. Interface is provided by paid service.

Another work [15] that used the same temperature sensor and, the data acquisition arranges with Zigbee communication. That communication type helps to send the data to server. The real-time data is analyzed and stored as information in the server. Therefore, medical personnel are informed via the GSM module. A microcontroller is used as a middle layer between temperature sensor and Zigbee modules. Crucial point is distance of the patient device; two Zigbee modules must be closer for better communication.

In another study[16], a healthcare system was produced with IoT technology. The system starts with sensor that could measure two body vital signs; heart rate and body temperature. The data transmission, that came from sensors, is provided with IoT communication protocol which is MQTT. Remote monitoring is provided by a mobile application. A generated web page is also used in the monitoring methods. These procedures is need only Internet access without any special requirements.

The authors [17] developed a patient remote monitoring system with IoT technology. Their proposed system aimed to monitor continuously patient's vital signs

and, send the data to medical staff. In the case of abnormalities of the vital signs, the system will inform the user. Communication of the system used MQTT protocol, microcontrollers and the cloud server are one of MQTT clients. The sensors measure vital signs which are heart rate, pulse, body temperature and body movements (with accelerometer). They connected to Raspberry Pi microcontroller. It communicates Amazon Web Services (AWS) IoT with MQTT communication protocol. AWS IoT was used for a user interface. Interaction between a user and the system was supported by the Android application interface, likewise Short Message Service (SMS) notification exists.

According to another designed system [18], Arduino ESP32 microcontroller and MQTT protocol were used for their smart healthcare system. Pulse rate, the saturation level of oxygen in hemoglobin (SpO₂), temperature and accelerometer for body movements were applied into a microcontroller. Patient's microcontroller connects the MQTT Server via MQTT protocol and results could be seen on the mobile phone or web page. In this study, two communication protocol was compared. First, they measure MQTT protocol performance on their system. Next, they applied HTTP (Hyper Text Protocol) protocol and evaluate its performance. Results of the two protocol performances were compared.

According to one study [19], they developed a healthcare system with IoT technology. It has some sensors to measure body vital signs. Alert system works with e-mail addresses, SMS and cloud server. They used ThingSpeak API application for sensor data, communication was provided by HTTP protocol. But, in this study, they did not evaluate the performance of HTTP requests when thousands of the data was received by ThingSpeak. The benefit of that user interface is that it showed graphics that included patient's body vital signs. However, it does not support flexible menu for user when they want to do other operations. As a result, the user interface is not enough.

Another research of e-health system [20], they developed a smart home system for monitoring of patient's health monitoring but different technologies were implemented to the system. As a gateway, Nokia Smart Home (NSH) was used. It controls whole devices and sensors of the system. For medical alerting, Philips Lifeline technology was applied. It has a user-friendly advantage which is a button for emergency situations. Interesting point is, for prescription, they used Quick Response (QR) codes. It contains JSON, the data was packed into it. Telegram instant messaging application was used for

the endpoint of service, it proved the system could be integrated into modern smart phones.

Additionally, alternative research [21] implemented a platform into their healthcare system which is Ubidots IoT Platforms. Device connection was supported by REST API. Patient, doctors or other related people will receive a warning about temperature, pulse and humidity from the cloud platform. SMS and e-mails are tools for warning to the user.

One of the detailed study [22], they developed SmartBand IoT Device that could connected a smart phone. Data transmission of the device was provided by low-power Bluetooth Interface to the user's smartphone. MQTT communication protocol supported between the cloud server and a smartphone through the Internet. As a broker, HIVE MQ is used but it is not supported by a database connection. The data is stored by a cloud server when the data is subscribed. MQTT packets size were tried to fix minimum bytes to handle the data traffic. Despite complex structure, it does not have any user-friendly user interface.

Structure of mentioned real-time monitoring researches resemble each other; the common point is, some cloud platforms were implemented into monitoring body vital signs systems, but they are not suitable for every situation. For example, a user does not understand how to use the system. That platforms do not have user guide. Another example could be patient-doctor interaction. Sometimes, users would like to interact with their medical personnel. Although cited researches have a lot of advantages, that designs are not enough for user experiences.

IoT platforms deal with a problem that is multiple user interfaces for each new IoT object systems. Android, iOS or web page platforms fit user needs but a lot of applications or using web pages could be confusing for the users. According to study [23], they compared to other notification methods such as e-mails and SMS for interaction with a patient. Instant messaging application WhatsApp Messenger is widely chosen one for user experience. Considering that information, instant messaging services or chatbots can be satisfactory for new system designs.

Another issue is speed of the data transmission. A healthcare system does not receive one data from a device. Countless data is wanted to send by a device and the system must be handled with these use cases. IoT has a lot of communication protocols

for better data transmission performance. However, developers should choose the right protocol for the system architecture.

To sum up, structures of common healthcare projects are same. They have sensors, gateways, communication protocols and a cloud platform. Transmission of the data is supported by IoT protocols and the data is sent to a cloud server. If these studies are wanted to be merged into one topic, telemedicine with IoT technology can be preferred. In this chapter, a few information is given to understand the used terms and technologies.

2.1. Telemedicine

Word of the “telemedicine” derived from two words; “tele” is come by Greek language and it is ‘at a distance, far or remote’. Therefore, it is formed like: “medicine delivered at a distance.”[24]. A simplified description of telemedicine meaning is given medical services via telecommunication protocols [10]. Typical face-to-face consultations between doctor and patient were transferred into another communication channel with telemedicine. Another definition, that was given by the Telemedicine Information Exchange, is “the use of electronic signals to transfer medical data from one site to another via the Internet, telephones, PCs, satellites, or videoconferencing equipment in order to improve access to health care” [25].

Telemedicine has started a long time ago when the telephone was invented; doctors could help their patients via telephone who is at a distance. One study summarized the evaluation of the telemedicine with used technologies by year, 1840 to 1920, telegraph and telephone were the most popular technologies until the radio was invented. The radio was used next thirty years, then television space technologies offered a new way of the telemedicine. In 1990, digital advancements become more useful at telemedicine applications [26].

In recent years, the term is transformed into different meaning because of implemented technologies. Homecare and remote healthcare delivery are used telemedicine tools to determine the patient’s information easily. Another issues, it aims to reduce number of the errors, access easily resources and decrease costs [27, 28]. If

principles of telemedicine are summarized, it must be based on evidence in the practice and risk management must be calculated. Another important thing of the telemedicine is, it should be confirmed for cost effectiveness. Besides, it provides equity in healthcare and professionals can keep in touch with their patients easily [28].

For understanding it a better way, last versions of telemedicine applications have several categories, but major ones are “store and forward”, remote monitoring and real-time interactive services [29]. If these are wanted to describe shortly, first one “store and forward”. When the data arrived from the digital environment, telemedicine application evaluates it and send asynchronously the information from a location to another location [30]. Remote monitoring is watching vital signs (e.g. patients’ heart rate in their daily life) and follow-up patient with technological devices like smart gadgets or medical devices. Furthermore, patient makes contact with doctor when they have distance [31]. Real-time interactive services have a strong relationship between store-and-forward and remote-monitoring.

Advanced technological developments offer to design new systems for healthcare applications in telemedicine perspective. Health resources and healthcare were passed on the electronic environment creates e-health [32]. One definition of e-health [33]:

“E-health is an emerging field in the intersection of medical informatics, public health and business, referring to health services and information delivered or enhanced through the Internet and related technologies.”

Considering behind this term, the recent Internet advancements are encouraged e-health applications to grow up. One of the offered solutions is IoT. It is major technology while developing a new healthcare application. But, it is not easy to implement into developing an application. IoT architecture and its communication protocols should be investigated deeply.

2.2. Internet of Things Technology

Embedded systems and software perspective create alternative ways of the intelligent era. Internet, Bluetooth, ZigBee technologies evaluate these designs and last

recent years, a term was involved by interconnected devices as “Internet of Things”. It can be said virtual world information technology merged with the real world of things [34]. The concept means new communication way between people and physical objects like smartphones, personal computer and devices. They could connect the Internet. IoT creates a new field to interconnected things get in contact with things or human [35, 36].

Anytime-anywhere-anyhow communication type is derived by the new IoT progresses. In Figure 2, it illustrates European Research Cluster of IoT (IERC)’s explanation of IoT which has been defined [37] “The Internet of Things allows people and things to be connected anytime, anyplace, with anything and anyone, ideally using any path/network, and any service”.

In 1999, Kevin Ashton developed an application in logistics with Radio Frequency Identification (RFID) then the IoT term gains popularity [38]. Moreover, years go on and new advancement dramatically increases. One research result was 5.5 million connected devices exist and it increases dramatically in 2016. IoT will become more powerful with estimated 20.8 million devices worldwide in 2020 [39].

It could be used in numerous fields as healthcare, home automation, industries, wireless sensor network, cloud computing, data mining, cellular network and many composite projects [40, 41]. Internet of Things in healthcare will be extremely important than any other categories and it will build \$117 billion in the market in 2020 [42].

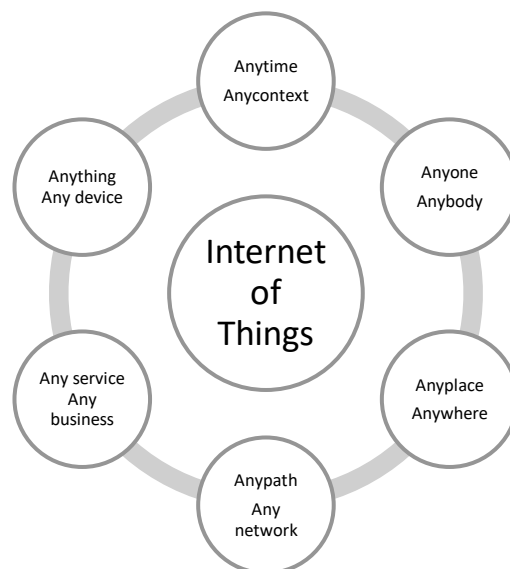


Figure 2. Working principle behind the Internet of Things [37]

In relation to, healthcare applications will cover half of the over market place by 2025, it was showed in Figure 3 [1]. Growing market encourages IoT and its definition become deep day by day.

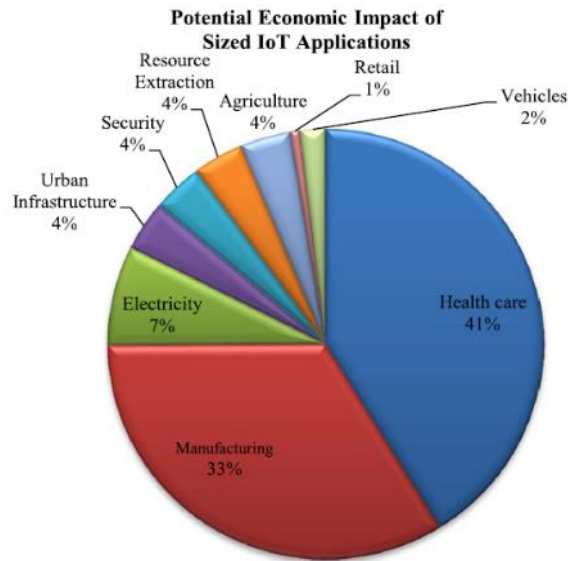


Figure 3. IoT applications market share by 2025 (Source: [1]).

Three paradigms [43, 44] construct IoT: i) Internet-oriented middleware, ii) things-oriented and iii) knowledge-oriented semantics. Figure 4 visualized main concepts that generate IoT. First area, “things-oriented” as known as real objects that are obtained by some technologies, RFID, wireless so on. The second zone, “internet-oriented”, includes middleware, it is responsible for data transmission on interconnected smart devices with interconnectivity protocols (e.g. MQTT, WebSocket). Lastly, “semantic-oriented” reference determines assembled data that came from “object/device/etc.”

Consequently, the point of junction of three paradigms will display the Internet of Things. The paradigms make IoT term understandable when a system is wanted to design with different components.

2.2.1. Architecture of Internet of Things Technology

Millions of devices connected with wired or wirelessly. Since network traffic is

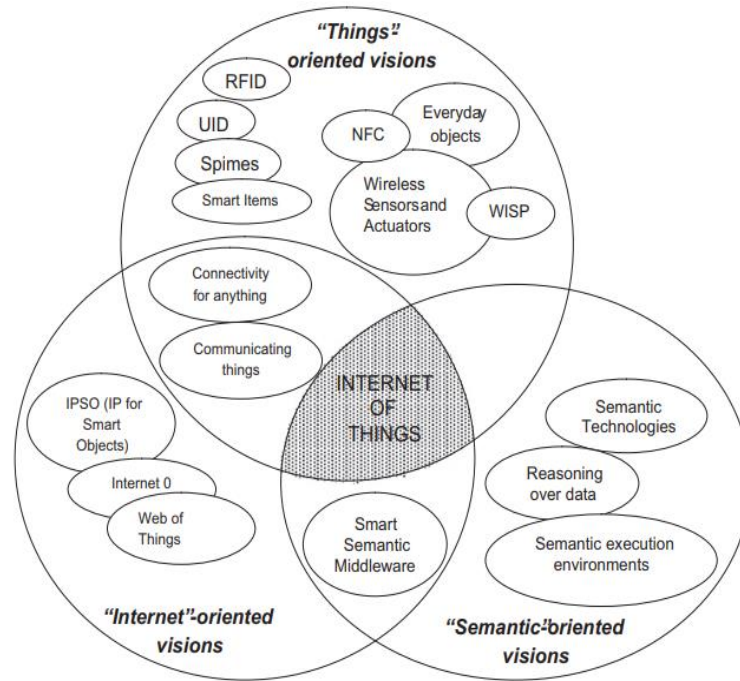


Figure 4. Convergence of Internet of Things paradigm scheme (Source : [44]).

growing exponentially, it might to caused security, data loss, reliability, etc. problems. A heterogeneous network, that consists of sensors, actuators, cloud servers etc., needs reference architecture to prevent problems. In a computer network, it has Open Systems Interconnection (OSI) layers and IoT has same layers. In Figure 5, it summarizes three main architectures of IoT: i) Service-Oriented Architecture (SOA) ii) 3-layer and iii) 5-layer based.

SOA [45, 46] was generated by the usage of services. The main goal is dedicated to being simple for design workflow and it fits the heterogonous system of IoT. It is usually used in wireless sensor networks. In the object layer, it includes hardware objects. Next layer that is middleware layer assists wireless and wired connections, then service layer user or applications handle with services for creating and managing. It includes service composition, service management and object abstraction. Lastly, the application layer handles with methods.

3-layer IoT architecture [47] is a basic model. It has perception, network and application layer. The first layer, that is named perception layer, is usually famous as the sensor layer. It based on the physical devices and components (e.g. smart devices, sensor). Its goal is to connect, measure, and evaluate physical devices into IoT network. Network layer could be called as a transmission layer, where existed communication protocols and

technologies are used for applications or things at heterogeneous networks. Interfaces and gateways are applied. The top layer of the architecture is the application layer, which accepts data transmission coming from the network layer. 5-layer architecture IoT [48-50] is composed of business layer, application layer, middleware layer, network layer and perception layer. The architecture starts with the perception layer and goes to upper layers. Next layer (network layer) helps to do the transmission of the data with 3G, 4G, LTE, WiFi, Bluetooth, etc. Middleware layer includes management of services. Used platforms and frameworks in IoT applications are involved in this layer. Its processes are based on the middleware layer transactions. Monitoring and management of four layers are managed in the business layer. Results, analysis, future works and business strategies are determined in it. The architectures of IoT technology look similar but they have differences in their working style, which depend on the project's structure.

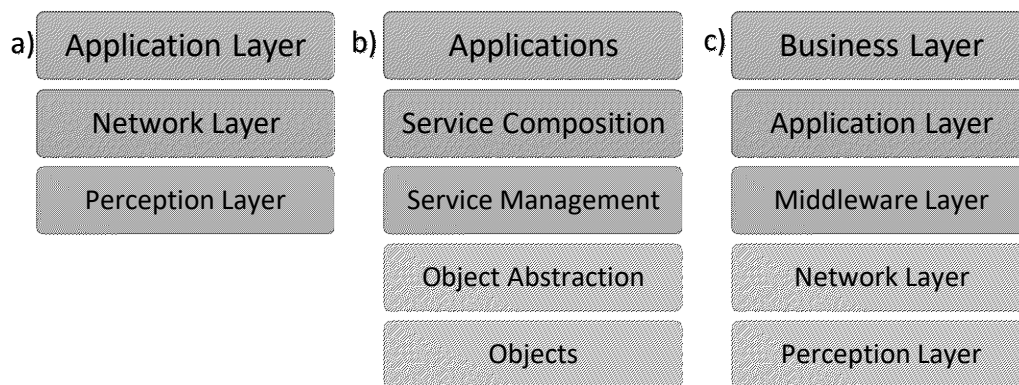


Figure 5. Architecture of IoT, a) 3-layer architecture b) Service Oriented Architecture based model and c) 5-layer Architecture

2.2.2. Internet of Things Communication Protocols

Common IoT protocols are CoAP (Constrained Application Protocol), XMPP (Extensible Messaging and Presence Protocol), AMQP (Advanced Message Queuing Protocol), MQTT (Message Queue Telemetry Transport) and WebSockets (Figure 6).

CoAP has two sublayers, messaging sublayer and request/response sublayer [51]. The first sublayer, messaging sublayer, uses UDP transport protocol for reliable communication and duplications are detected by it. Request/response sublayer goal is to manage individual nodes' REST communication.

XMPP protocol provides for request/response and publish/subscribe. Bi-directional communication is supported by request/response. Together with publish/subscribe model ensures multi-directional communication [52]. The way of conversation is established like a client-server concept; likewise, server-to-server communication could be coming true as wished. The data is formed by "XML stanzas" that are small XML (Extensible Markup Language) based texts [53]. Common IoT protocols are CoAP (Constrained Application Protocol), XMPP (Extensible Messaging

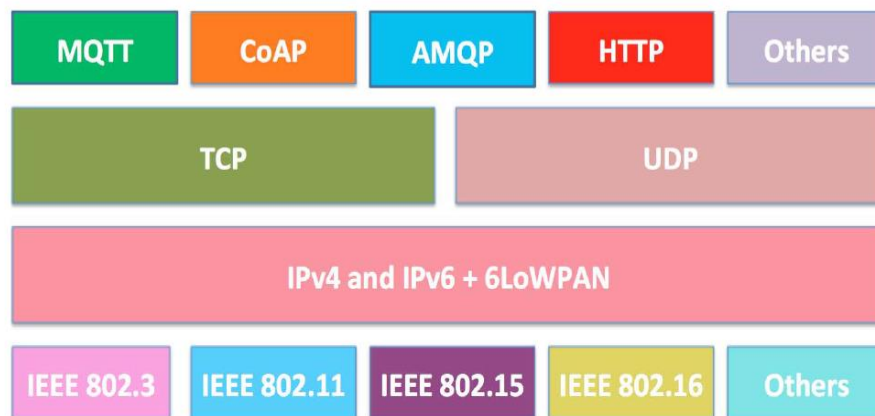


Figure 6. Reference of Internet of Things' protocol stack (Source: [49]). Model based On Open Systems Interconnection (OSI) layers, lower to upper layers, Link Layer, Internet Layer, Transport Layer and Application Layer.

and Presence Protocol), AMQP (Advanced Message Queueing Protocol), MQTT (Message Queue Telemetry Transport) and WebSockets (Figure 6).

CoAP has two sublayers, messaging sublayer and request/response sublayer [51]. The first sublayer, messaging sublayer, uses UDP transport protocol for reliable communication and duplications are detected by it. Request/response sublayer goal is to manage individual nodes' REST communication.

XMPP protocol provides for request/response and publish/subscribe. Bi-directional communication is supported by request/response. Together with

publish/subscribe model ensures multi-directional communication [52]. The way of conservation is established like a client-server concept; likewise, server-to-server communication could be coming true as wished. The data is formed by “XML stanzas” that are small XML (Extensible Markup Language) based texts [53].

Communication between different devices that could be generated by different languages has two characteristic points: heterogeneity and interoperability. AMQP protocol supports the structure of the communication [52]. Reliability, security and performance are aimed to use as a base by the protocol. The protocol uses publish/subscribe model with TCP session, and it is made up two parts: exchange queue and message queue. Exchange queue based on routing messages into the queue within specified rules. Other queue put aside the unsent messages in queue [51].

Websocket does not fit request/response or publish/subscribe methods. It works with a handshake during the time that single TCP channel and duplex communication ways are included [52]. Although the plan of the Websocket was not created for embedded systems such as constrained devices, Websocket is useful for real-time communication. Security is high, overhead is minimized by its and efficient messaging occurs [54].

Architecture of MQTT (Message Queue Telemetry Transport) has two main parts, broker (server) and clients. It maintains publish-subscribe messaging. A device which uses MQTT protocol consume low power, memory and bandwidth; it assures delivery of packets [55]. Communication will start when one client publishes a message with specific topic to broker. If one connected MQTT client waits that specific topic, message will be subscribed [56].

According to one research [57], consideration of these protocols was explained and compared. MQTT, CoAP, AMQP and XMPP have been included by the assessment. CoAP has high latency and low throughput than MQTT protocol. Quality of services (it could be generalized as determined network performance [58]) is weak for message deliveries and, its libraries and tools are poor. The protocol does not have enough open-source libraries. AMQP protocol does not fit real time applications and it does not provide bandwidth. Moreover, it does not have any quality of service. Lastly, the most important disadvantage of that protocol is deficiency of its open-source libraries for constrained devices. Disadvantages of XMPP protocols are quite different. Usage of CPU is high and, utilization of bandwidth little more than other protocols.

MQTT communication protocol was chosen to use in this thesis project's server system. Benefits of the protocol are low memory and CPU usage during communication. Moreover, this protocol has low complexity and low power consumption. MQTT protocol can be also implemented easily in embedded devices and mobile platforms. Because of these reasons, this protocol was chosen to be used in this thesis.

2.2.3. Message Queuing Telemetry Transport

IBM introduced MQTT in 1999 as publish/subscribe protocol that was aimed to send data with short network delays and high bandwidth [59].

The architecture of the MQTT has involved two components, clients and brokers. Clients roles could be a publisher or a subscriber. One broker must exist for starting communication. In Figure 7, it shows TCP/IP protocol must be provided for clients and broker. For classification of the roles' perspective [60], client publish/subscribe a message and the broker subscribe it. If a subscription is wanted, firstly broker collects subscription by the clients.

Provided Quality of Service (QoS) [2, 59] has three types. In Figure 8, QoS levels communication was visualized. First one, QoS=0, a message is sent by publisher and resend of the message does not exist. The other name of this QoS level is "fire and forget" and it might cause the message to be lost. Second type, which is QoS=1, "at least once", message delivery was guaranteed by the service. PUBACK message is important for it because, in case of a message is lost, packet will be wanted to send again as far as PUBACK message arrived. Lastly, QoS=2, "exactly once", performs three extra message type with normal PUBLISH message for exact message delivery. Delay could be longer than others, but message delivery performance is perfect, message loss could not happen when it applied.

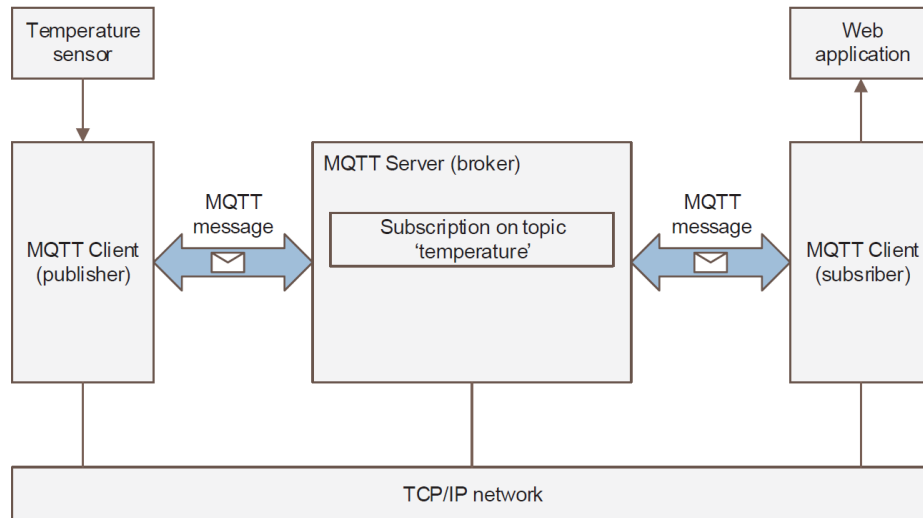


Figure 7. Message queuing telemetry transport protocol model (Source: [2]).

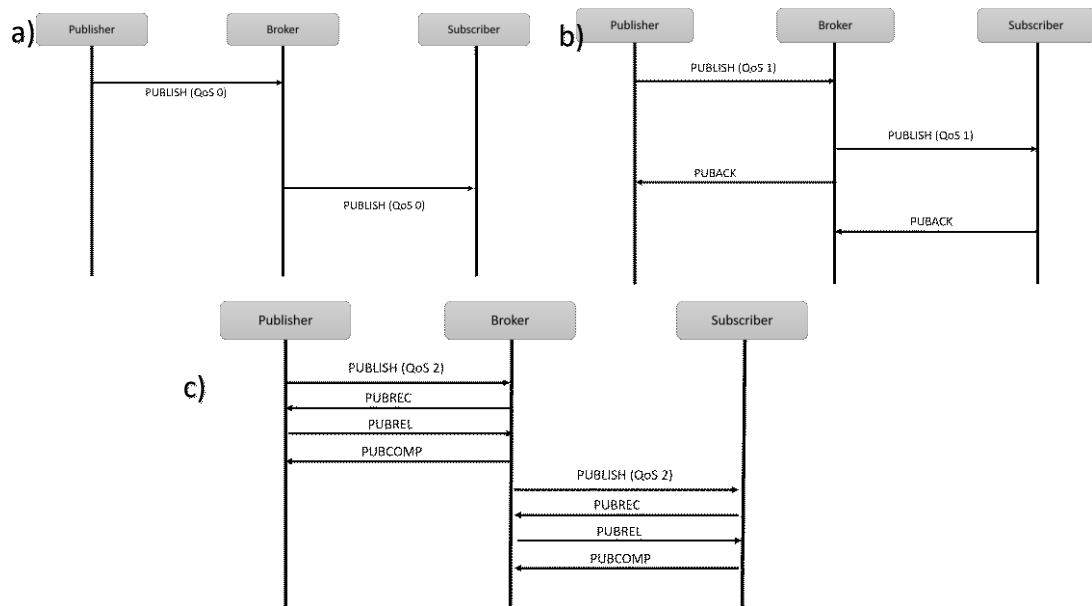


Figure 8. Quality of Service Levels a) Quality of service equals to zero “fire and forget”, b) Quality of service equals to 1 “at least once”, c) Quality of service equals to 2, “exactly once”.

2.3. Chatbots

Development of the applications creates new requirements at human-computer interaction perspective. Users prefer to get in contact with the system more than a menu. This necessity generates an advancement that is called as “chatbot”. It looks like a simulation of human conversation with an architecture which includes a language model and computational algorithms [61]. In addition, that term is also introduced as conversational agents or systems, chatterbots, chat robots [62, 63]. That communication affects the dynamic of user and digital technology interaction and interface.

If standard chatbots are wanted to compare with standard mobile applications, several advantages of the chatbots could be said. Users do not need extra download and install the redundant application; a messaging application will be enough, and it is one of the best solutions for supporting at multi-platform. Another benefit are security, messaging protocols and performance will be done by the messaging application. A chatbot will not be done additional operations for these issues [64].

Chatbots are divided into two types, rule-based and Artificial Intelligence (AI) - based. Rule-based chatbots have defined scripts and commands to lead flow and conversation. The second one, AI-based is different than rule-based chatbots. They are more complex which have rich dialogs and use higher technologies such as machine learning, natural language processing (NLP), etc. [65].

2.3.1. Chatbot History

Chatbot is not just a popular term in recent years and it has a history. Ancestor of chatbots was named ELIZA by Joseph Weizenbaum from Massachusetts Institute of Technology, it was designed as “pattern matching” technique that used determined scripts [66]. According to Nadelson [67], it has a basic aim to “demonstrate that the communication between man and machine was superficial”.

In 1995, another chatbot was developed and it is called as ALICE (Artificial Linguistic Internet Computer Entity) by Dr. Richard Wallace. The tool's language is based on Extensible Markup Language (XML) [68]. It seems like Weizenbaums's ELIZA chatbots about user-machine conversation with "pattern matching". Additional well-known chatbots are MegaHAL, Parry and Converse [69] whom helped to develop chatbot structure.

Encouraged improvements of chatbot technologies that were established by the ancestors, it provides a new line of vision. AI was added and whole engineering methods were changed. It showed up as stand-alone services (e.g. Amazon Alexa, Microsoft's Cortana, Apple Siri) or it could be found like in embedded instant messaging programs [70]. Additionally, Natural Language Processing (NLP) was included in modernized ones. Union of the ancestor of chatbots, AI and NLP build well-designed architecture and interaction between user and machine will be effective in daily life.

It is accepted by everybody to chatbots in healthcare is a new area of research. Either AI or rule-based chatbots, they help to people about their health. Therefore, one specialized chatbot [71] was developed for weight control with AI-powered. The interactive script offered daily weight control, physical activity, etc.. Another study MamaBot [64], this AI-based chatbot provided to increase user interaction and experiences, but the important thing is in this project, chatbot was developed with the Telegram open-source API. By the way, chatbots become more popular than their ancestors in different application areas.

CHAPTER 3

MATERIALS

The proposed system is aimed at the Internet of Things technology about to help patients and medical consultants. Due to the fact that, performance of the system and cost of used components become more important circumstances than other goals of the project. Selected technologies were evaluated carefully to how to be used in it. While designing architecture, five important advancements were used: Telegram, Node.js server-side environment, Redis Database, Mosca MQTT Broker, Arduino microcontroller boards. All the developed of technologies and applications are open source that means libraries open for everybody to develop new projects.

To build a new system, cloud server environment steps forward to develop precisely. Node.js was selected because its engine provides a lot of free libraries for server-side programming. Other current technological advancements that are Redis, Mosca and Telegram router, could be integrated without any difficulty in Node.js environment. One Node.js server might support all components at in different ports.

Telegram is a secure, open-source and free instant messaging application but application suggests an idea to develop own chatbots. Another component, Mosca, is written with Node.js programming language as MQTT Broker and it is free too. Therefore, Arduino is a microcontroller board and it is open-source platform.

That embedded system elements could talk to each other. Additionally, user-friendly interface is offered by existed chatbots API, it gives good user experiences and does not need to download or develop a new application for a user.

3.1. Arduino and Sensor Design

Open-source platforms a catchier title for developers because their open source libraries.

In 2005, one of the open-source projects was started which is Arduino in Italy[72]. Arduino[73] could be defined as a microcontroller which is easy-to-use and open-source hardware and software platform. Simple structure offers to build new projects such as wiring sensor, IoT embedded systems etc.

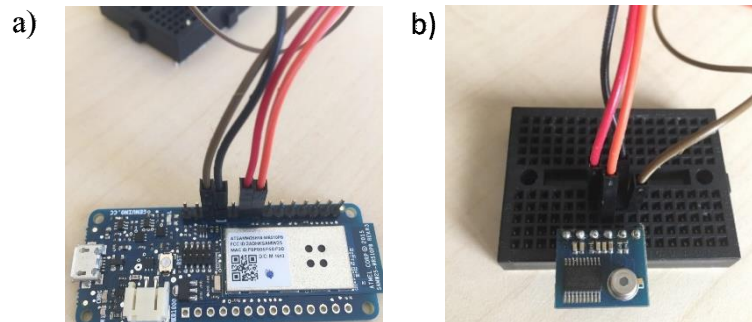


Figure 9. Used microcontroller a) Arduino MKR1000 microcontroller and b) infrared temperature sensor, MLX 90614

New Arduino boards are produced and one of them is MKR1000 (Figure 9a). Practical and low cost properties with Wi-Fi connection [74] make preferable for IoT projects. The board is 32-bit and Wi-Fi was developed with 2.4GHz IEEE 802.11 b/g/n [75]. Therefore, it can publish/subscribe library [76] provided by MQTT broker connection and it is suitable the project's implementation.

In this project, infrared body temperature sensor, MLX90614, (Figure 9b) was used [77] and it was wired on Arduino board to transfer measured data.

3.2. Computer Requirements

The server was built on a computer that has 16 GB RAM and Intel Core i7 chip clocked at 1.99 GHz and 64-bit processor. The system runs on the Ubuntu version 18.04 LTS operating system. Engine of the Node.js limited as 8GB at RAM.

3.3. Node.js Engine

Node.js was written in JavaScript environment and it is developed for server-side applications. Its design based on the asynchronous event-runtime is good for scalable network [78]. Low memory consumption, providing long-running server processes and high performance are the key properties of the Node.js [79].

3.4. Remote Dictionary Server

Redis (Remote Dictionary Server) is remote database stored as key-value. It is also used as cache and message broker. The conception of operation starts as all the data is stack into memory and its operations are done in it. In the following step, stacked data will be transferred to hard disk asynchronously from the memory [80]. On a normal computer, Redis can deal with 100 thousand read/write requests in a second [81]. The comparison of performance is similar as Memcached [82] but data structure types are different. These are strings, lists, sets, sorted sets and hashes.

3.5. Telegram

A great number of mobile applications are generated day by day, which is over 2 million in 3rd quarter of 2018[83]. Majority of the companies develop their own applications and it makes overcrowd. Users usually prefer minimum number of applications and this provision has positive effect on developing chatbot applications. Facebook Messenger, Telegram, Slack, Kik, and Viber are good examples that they could be used as developing chatbots [84].

Telegram is a fast and simple messaging app that is completely free. It is cloud-based and works synchronously at all devices where they are opened[85].

Content of the Telegram is an instant messaging application. Users could send messages as other messaging application could do. Nevertheless, communication is not remained limited between user to user. A receiver could be user, group or channel. If one message is wanted to send to the receiver it does not need a relationship between the users [86]. Moreover, all private conversation and personal data are protected from third parties, marketers, advertisers, etc.. One of the best properties of the messenger is it pays attention to security; secret chats are protected by end-to-end encryption.

Telegram Messenger is open-source and it is open to all developers for creating chatbots. Telegram bot [87] can be defined as a machine user, but it is different than a normal user and its commands are managed by a programmed machine.

If a chatbot is wanted to develop, Telegram Bot API framework is provided by some platforms. According to one investigation, they researched repositories in the most popular repository hosting platforms such as Github, Gitlab, Sourceforge, etc. and they found most common languages are Python, Javascript, Java, Go, PHP for developing Telegram Bot [88].

Telegram Bot token key must be taken from the BotFather before the chatbot is developed. Creating a new chatbot does not require any GSM number, the only a token key would provide security issues.

3.6. Mosca MQTT Broker

Used library as MQTT broker is Mosca[89] that is written in Node.js language. It supports MQTT 3.1, provided quality of services are QoS 0 and QoS 1. Mosca could be integrated over Redis, Kafka, WebSocket, MongoDB, etc.

CHAPTER 4

IMPLEMENTATION

Architecture has four components as the main server, database, MQTT broker and devices. Before the evaluation of the user interaction between the system, it must be defined workflow of the system. In Figure 9, architecture and workflow were described, main server (Telegram Router), database and broker; they are not put into separated cloud servers, one server covered all these architectures. In order to, Telegram Router, Mosca MQTT Broker and Redis Database have own unique ports at the cloud server. Another component is a device that created with a microcontroller and a body temperature sensor.

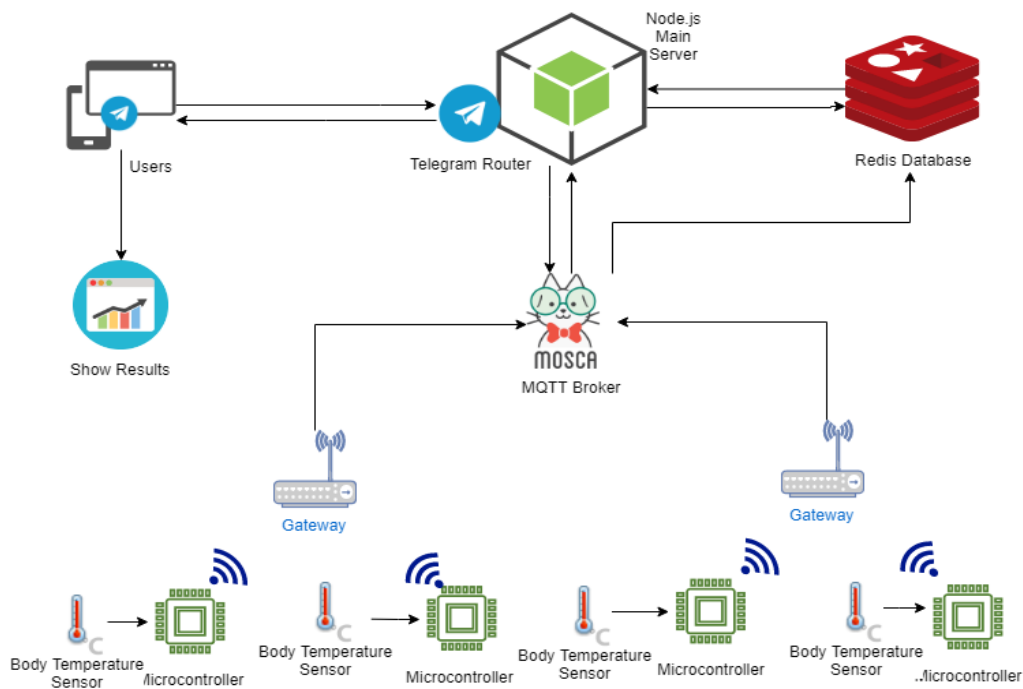


Figure 10. Architecture of the system.

That microcontroller could connect the wireless network and send measured data to the server through MQTT broker. These parts are automated, and users are not allowed to interrupt that flow. Users have pre-selected options for the system

communication, only these options could be done by the users such as monitoring temperature graph, find a device, etc.

4.1. Arduino Structure

In this project, an infrared body temperature sensor is connected on Arduino microcontroller. Used microcontroller can make a connection to the Wi-Fi easily and securely with WiFi101 library.

When body temperature is measured, the sensor sends the data to its wired microcontroller which could pack the measured data into JSON data format. Successfully packed JSON data will be ready to publish. Microcontroller uses a library to connect MQTT broker. It will wait any a new message like “are you alive?” or “start measurement” from this broker.

In Figure 11, the flowchart begins with WiFi101 library. When a device is turned on, it will try to connect Wi-Fi with this library. The information of the wireless connection must be defined in the device. If a device is wanted to connect wireless access point exists, it easily connects. In case of no wireless network in the environment, it will be wait until the desired situation is available. Next step is the connection of the sensor. A sensor must be wired over the microcontroller. But if any problem exists, the user should get in contact with responsible medical personnel. If everything goes well, flow will hop next condition. Microcontroller must check MQTT broker current statue. If it is not connected, it will try to connect to the broker. If there is a broker exists, that broker will accept the connection request and it will send connection acknowledgement. When the acknowledgement arrived, in this step, the microcontroller will wait subscription of the message with specific defined topics.

Some operations are assigned into the device, such as discovery messages, start-stop measurement and assignment of the device. In the first case, if a user or cloud server wants to find a specific device, the server will publish discovery message through the broker. Secondly, defined and described topics will wait to the subscription. If broker subscribes intended topics, it will publish through the broker and the device will subscribe

it. The topic must have device ID and two commands. For example, if a message contains “start measurement command”, it will be looked as “devID/temperature/startmeasurement”. Slashes divide topic into different level for wildcards and security. The device could only subscribe defined topics with their wildcards. Last issue is, defining a new device command. When the device assigned to another patient, the information of device and patient must be attached inside.

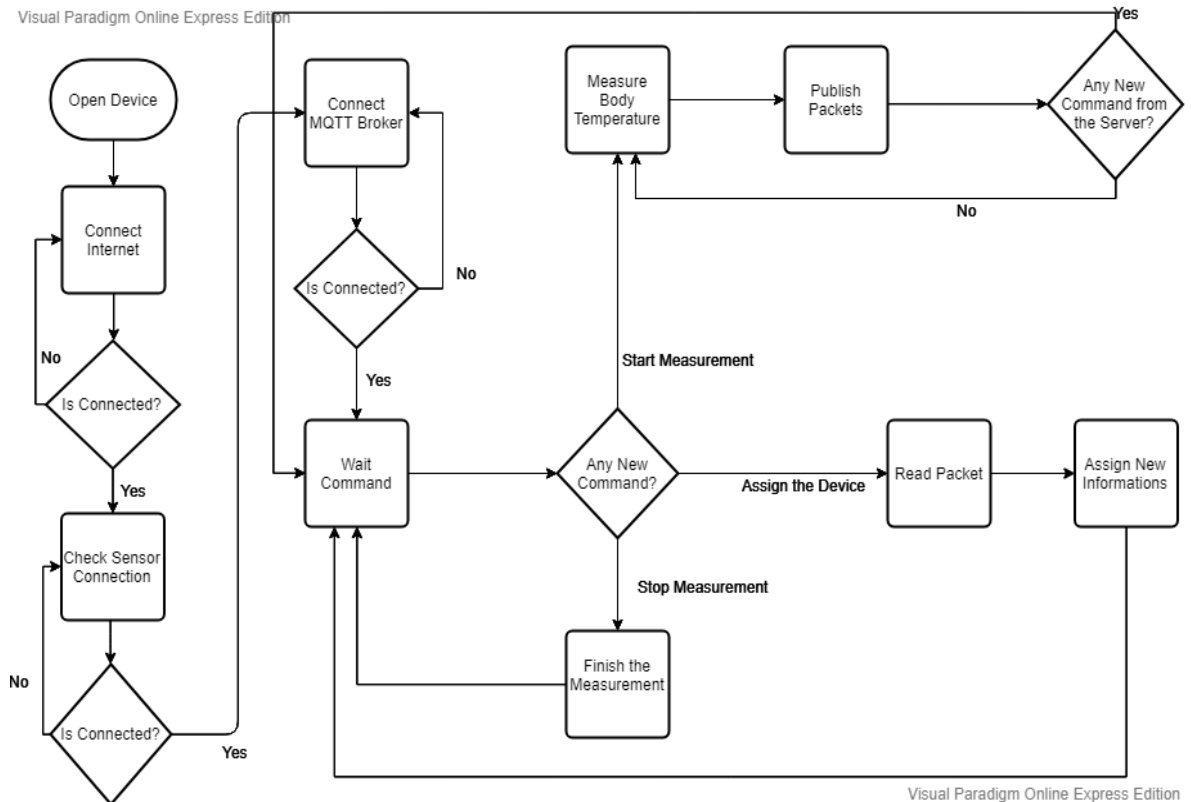


Figure 11. Flowchart of a microcontroller which has a body temperature sensor.

Moreover, when the device subscribes command of start measurement, it will measure body temperature with its sensor then the data of measurement will pack into JSON file format with some device information. The packet includes the time of the measurement, body temperature degree, device and patient ID. Total size of JSON packet is about 60 Bytes. Packages publish until the stop command subscribes from the broker.

4.2. Mosca MQTT Broker

Used MQTT Broker is Mosca, which was written in Node.js programming language. The broker helps to communication between user-device and server-device in the project. The communication protocol is MQTT and used quality of service (QoS) is equals to zero that based on “fire and forget” aspect. Port of the Mosca in the server was assigned as 1880 and clients must know broker’s specific password to connect the broker. Moreover, arrived measurement data is send to Redis database via the broker using the port of the database. Database queries, that are HSET, are used to store the data.

4.3. Redis Database

Port of the database is 6379 at the server and it exists in the cloud server between broker and Telegram router. Database structure could be explained as it is made by hashes and lists. The information of the users and the data of measurements are stored in hash tables. Moreover, defined and assigned devices are kept in the linked lists. Redis database is made by data structures like linked lists, sets, hash tables, etc.. It provides flexibility for storing data. In Figure 12, user data put into hash tables with their

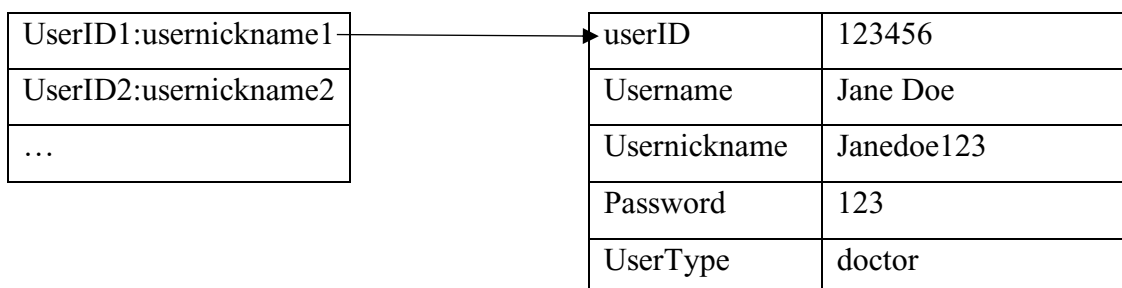


Figure 12. Hash table structure for users in Redis database. Users’ identity numbers and nicknames are stored by a hash table and a user information attached at the userID.

information and nested hash tables created for the storage are shown. Hash keys were identified by the Telegram as chat identity numbers.

A measured data is stored in the user's folder of the database (Figure 13). Keys of the hash table are date and time of measurement and degree of temperature. Detailed measurement data including time, degree, device id and patient id in JSON file structure was stored in a hash table.

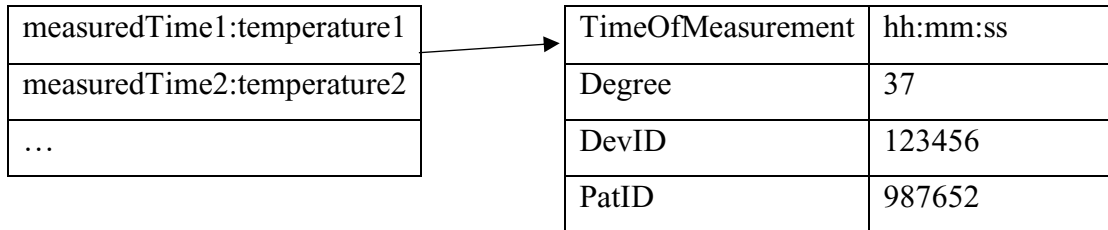


Figure 13. The measured data is stored in hash table and detailed data information keep in another hash table.

4.4. Telegram Router

Telegram chatbot was designed with Telegram BOT API using Node.js engine. Before creation, the token key was obtained from the Telegram Botfather. Additionally, to prevent crash of the communication and other unexpected problems, Telegram chatbot was assigned to a new port in the cloud server. Polling was used instead of setWebhooks for routing. Node.js libraries are available for Telegram BOT API, and especially “telegram-node-operation-manager” [90] has importance at menu creation. Moreover, the term of Telegram router means that it manages users and chatbot communication, functions etc.. It stores chatbot's information at the cloud server.

For understanding conversation between the chatbot and user, sequence diagrams were illustrated. In Figure 16, a sequence diagram of login operation was drawn for registered users. Before the logging operation, a Telegram user has to write a start message which was selected as “/hello”. When a user sends this message, if Telegram router is open, it will accept that start message. Then, automatically Redis database will start. Furthermore, the start message includes a lot of information such as Telegram

User's ChatID, username, etc.. They might help to operate some functions while Telegram router is running. For example, at the logging operations, the message's information will start the login function. Telegram ChatID is got from the start message by the router and server send it with HGET query. Database transfer data through an object and system control the object. If the object is null, the system will warn the user for sign up. Otherwise, the system sends a message to user and it will wait for password of the user. Entered password is checked from the database by the server and feedback of result is forward to user. In case of password is correct, main menu will open to the user. But if the password is incorrect, user should restart the system with send a start message again. The chatbot will offer a couple of operations for the system. After the system login operations, the system will wait the response of the user from the menu. In Figure 17, a sequence diagram for finding a device was shown. For this case, user type was selected as a doctor and he/she logged in the system then user select "Find A Device" options. When TelegramBot receives the chosen option, Telegram router must check devices that are assigned to the user. A database query will be sent by the server and object of results will be shown on screen of the user as a keyboard. Next expected operation is user's device selection. In that point, an asynchronous function will start. The identity number of the selected device is published through Mosca broker and the system waits for a second for an answer from the device. If the device was turned on, it will subscribe that message from the broker and publish its own message to indicate that it is alive. One second waiting time is set to receive the answer. If it takes a longer time, the user will be notified for this situation and a new menu will open to the next operation. After alive message is come from the device, the user will receive a message of the device statue and new menu will open for other operations.

One of the opened menus includes "start measurement" operation was explained in Figure 14 with a sequence diagram described the communication between the components of the system. When the user selects the start measurement option, the server will directly publish that message through the broker and wait for feedback of the operation. Subsequently, the selected device connected to the broker and the device will subscribe to the message having a specific topic. Before the measurement operation, the device will publish a message and the measurement of the body temperature will be started. Next step is to pack the result of measurements into JSON format and the results

will publish with MQTT communication protocol to the broker. At this point, the database will only get data from the broker with queries.

When the user wants to observe measured body temperature results in a continuous graph, “Get the Results” option must be selected. In Figure 15, Telegram router takes that request, creates a query that is based on HGETALL and waits for results. The results will be added into an object for evaluation. Then, the server uses Chart.js [91] library and append the results to a selected graph. One link will be sent to the user by the cloud server to show up the graph in HTML file.

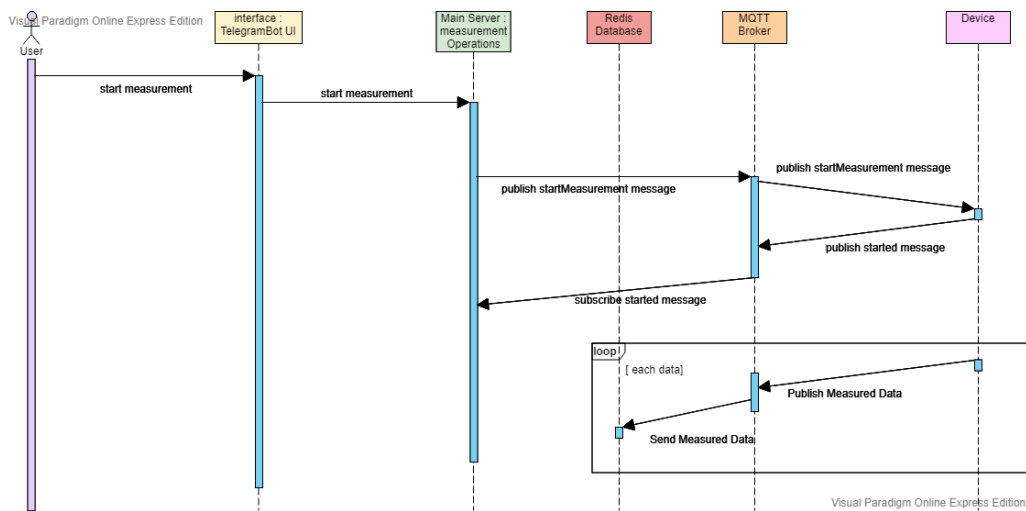


Figure 14. Visualized sequence diagram includes start measurement operation.

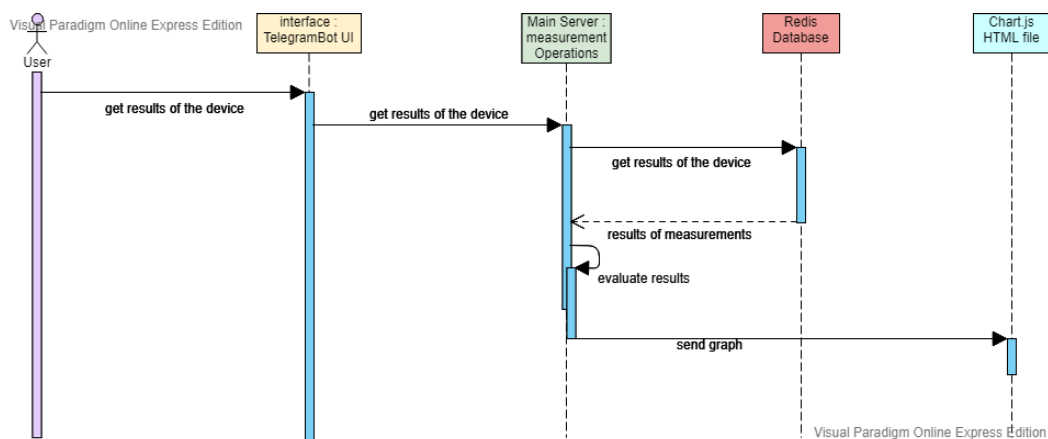


Figure 15. A sequence diagram that show a user how to get the results from the system.

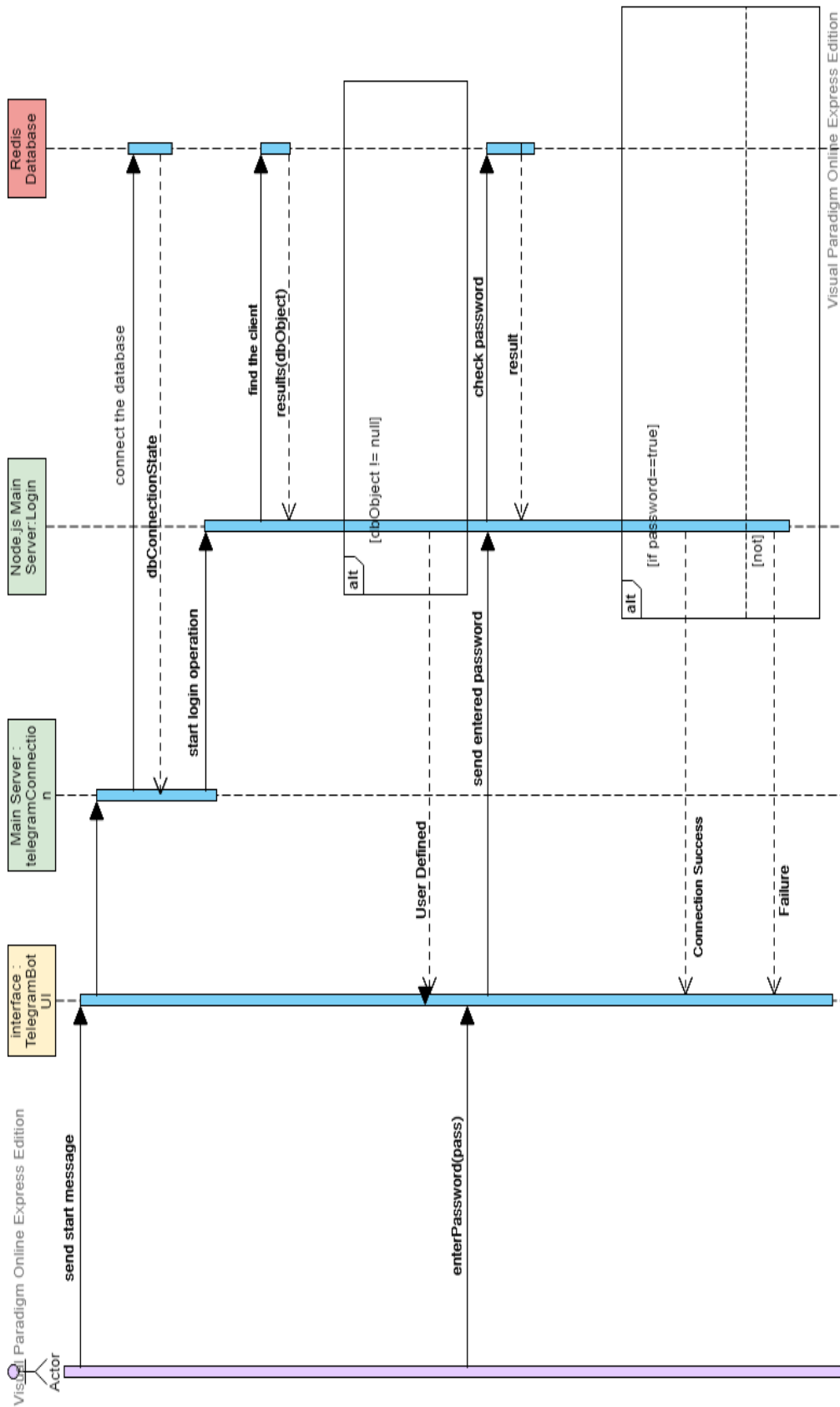


Figure 16. Sequence diagram of login operation.

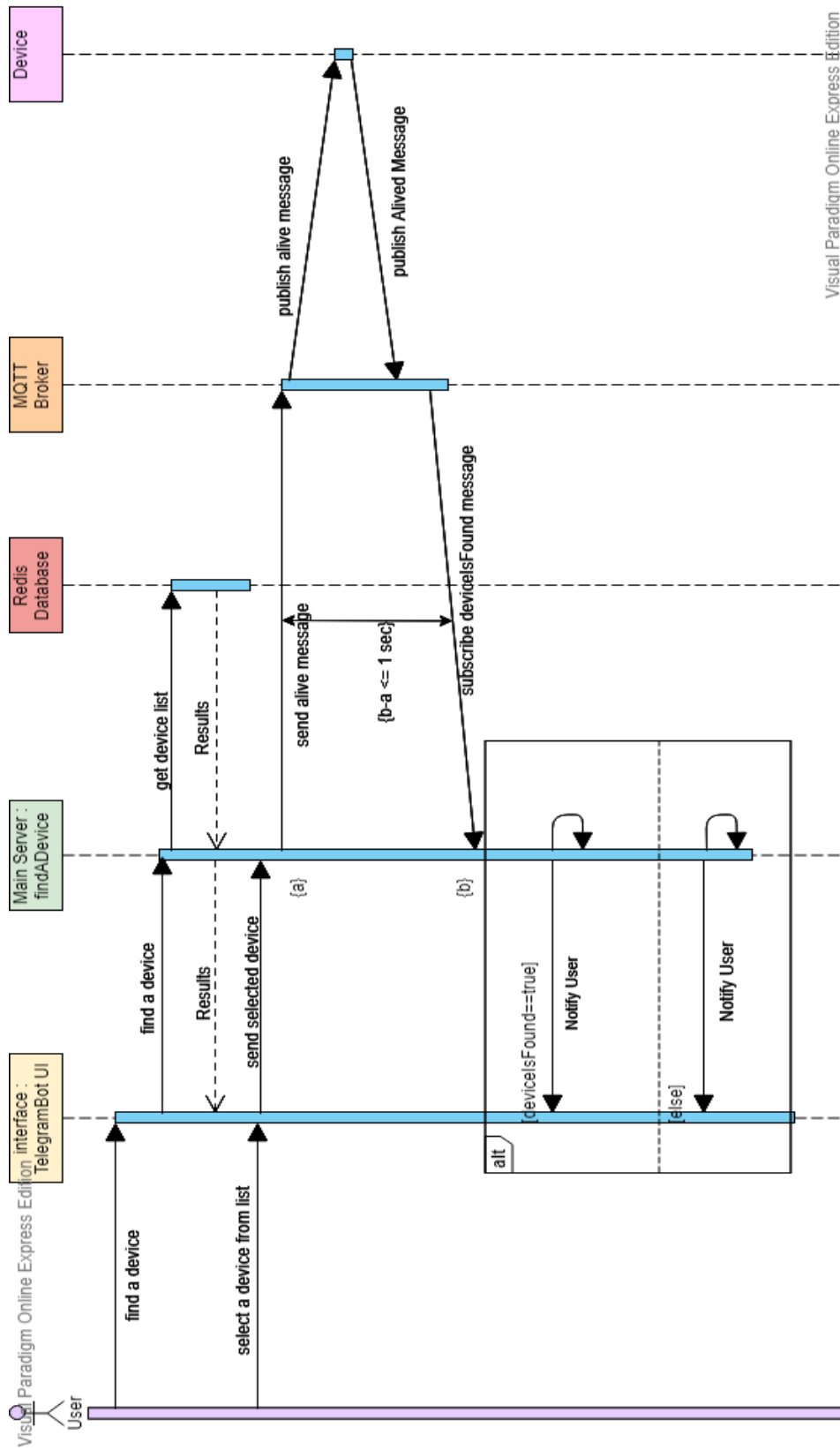


Figure 17. A sequence diagram for finding a device

4.4. Telegram User Interface

In Figure 18, screenshots of Telegram chat user interface were given for real-time conversation. First, the user should write start message to initialize conversation. After the start message, the system checks user registration. If the user exists, one menu will appear at the keyboard (Figure 18.a.) and it includes two options which are Login and Help. In Figure 18.b., the user selected Login operation and the system is waiting for user's password for next menu.

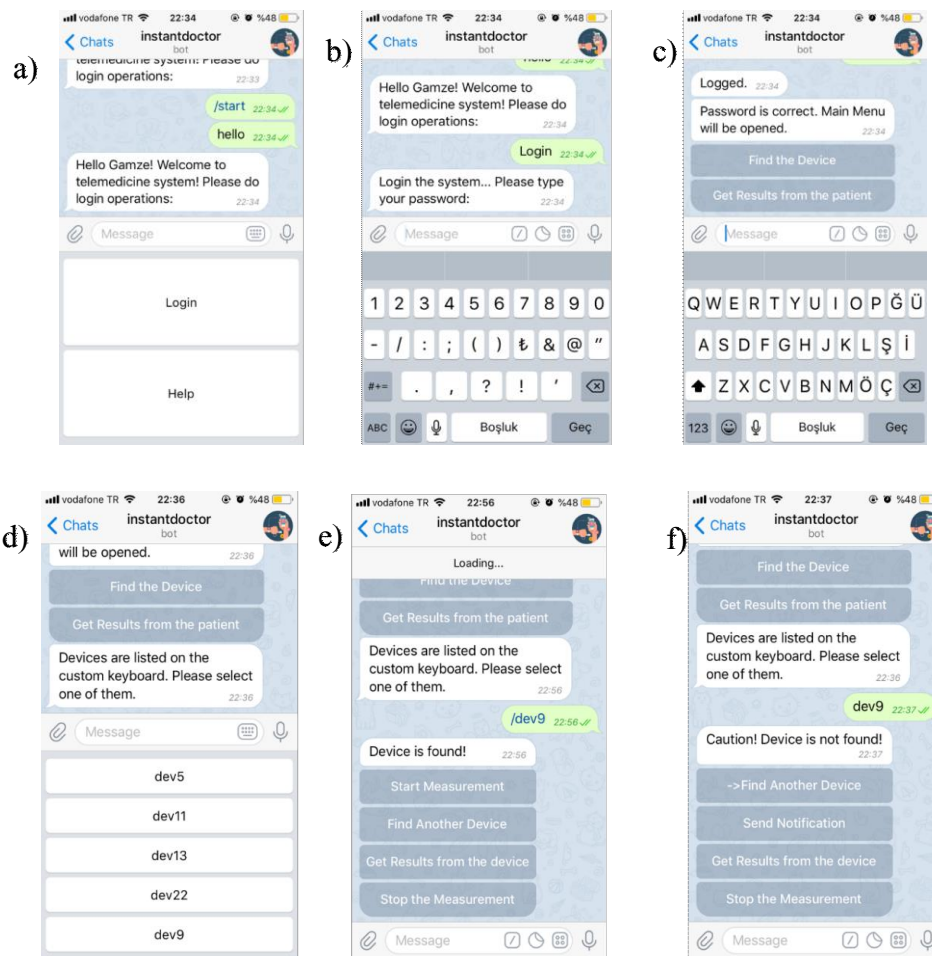


Figure 18. Telegram Menu a) Login operation keyboard b) The system waits password from the user c) Find a device and get results from the patient inline keyboard appears after the login d) If device is found next inline keyboard appears e) Keyboard of device is found f) When device is not found.

After entering the password, in Figure 18.c., inline keyboard comes out and it has two options “Find a Device” and “Get the Results”. The user interface of finding a device starts with Figure 18.d. that contains the devices list at the keyboard. It is opened for the user to select. In Figure 18.e., if the device is not found, the user will be warned by the system with a new menu. In case that, the device is found, a different new menu will open as in Figure 18.f.

CHAPTER 5

PERFORMANCE EVALUATION

In this section, the throughputs of published packets through Mosca MQTT broker and the database performance are evaluated.

Performance of the broker was measured with changing number of packets and using different packet sizes. The topic of published packets is same but the message payload different at different cases. Payloads were generated by Node.js's crypto library which creates random characters with stated packet size. Performance evaluation of the broker started with 1000 messages and number of the packets were increased 10000 at each test. While testing on a computer which provides requirements of experimental evaluation, throughputs and published times were examined.

Database evaluation was done by HSET query requests. Tests were run at different states on the server. First, requests were sent when the server has no operations. Requests were also delivered while Mosca MQTT Broker was running. This different test environment shows performance of the Redis database at that server.

5.1. Mosca MQTT Broker Performance Evaluation

Mosca MQTT Broker was described in Materials section and, as mentioned previously, it was written in Node.js programming language. In the computer, which is used as a cloud server for the system, 8 GB RAM space was given to Node.js environment for running Mosca MQTT broker. It helped to handle published packets through the broker.

The simulation started with two virtual clients on the computer: publisher and subscriber, which were written in JavaScript programming language. The communication

protocol between clients and broker was MQTT and, QoS level equals to zero. Performance evaluation based on three test cases that were generated different packet sizes in payloads: 2 bytes, 16 bytes, and 128 bytes. A packet was filled with JavaScript's crypto library, which uses random bytes.

Benchmark of the broker is developed, and it has a basic structure for measuring message throughputs. It based on publish a packet at specified time intervals in nanoseconds. Desired performance is to send the predefined number of packets at a second, but either the Internet performance or the broker traffic lowers the throughput. The formula of throughput could be written as:

$$\text{Throughput (msg/sec)} = \frac{\text{Aimed Number of Packets}}{\text{Sum of time during publication}}$$

As mentioned, the first test starts with 1000 packets that are wanted to send at one second. Each test increased the number of packets, which is planned to send. An increment will be done as add 10000 packets at the current number of packets. For example, in the second test, the number of messages to be sent in one second is increased to 11000. Nevertheless, not all of the packets could be published to the broker at one second. Delays can be occurred due to the broker performance. For finding standard deviation of throughput for each test, performance evaluation was performed as 3 replicates.

Figure 19 includes a plot that shows throughput performance for different number of messages. Each line displays a packet (payload) size that was published by the publisher to local Mosca Broker. It is observed that throughput performance for 2 bytes and 16 bytes packets are nearly same. The saturation of throughput began at the 4×10^4 msg/s. On the other hand, when payload size is assigned as 128 bytes, throughput is decreased then nearly half of other performance tests. In addition, the throughput was saturated at 2×10^4 msg/s. It proved importance of payload size when a message is wanted to publish from a device or from a server. The most important thing in these cases is that broker can handle publish all coming packets without any loses for different number and size of packets.

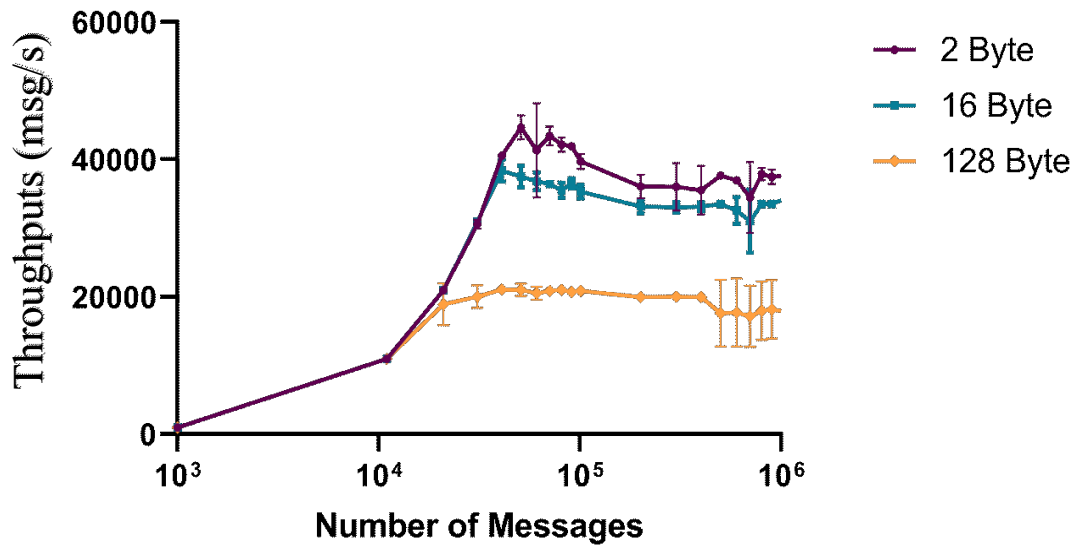


Figure 19. Throughput performance as the broker receive published messages.

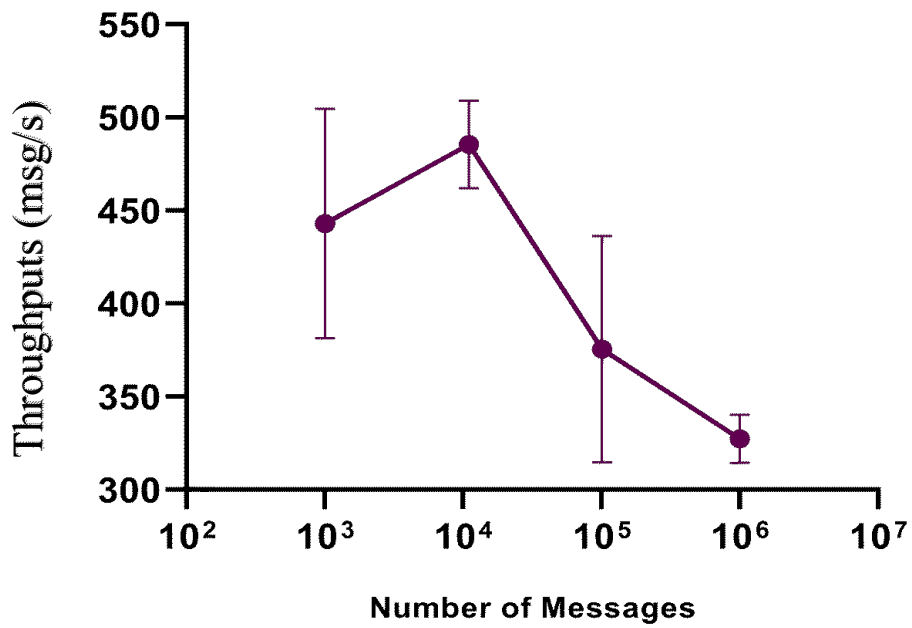


Figure 20. Throughput performance of the broker that receives published messages from one Arduino client.

5.2. Redis Database Performance Evaluation

Database performance was evaluated under two conditions: the broker does not have any workload and the broker gets a new published packet. The dataflow is operated by the broker. When a packet is arrived, the broker sends it to database. Throughput performance calculation depends on number of sent HSET query requests and total time necessary to publish all requests as in the following equation:

$$\text{Throughput (msg/sec)} = \frac{\text{Aimed Number of HSET Query Request}}{\text{Sum of time during publication}}$$

For understanding how broker operations affect the database operations, a test was started with the broker without any MQTT protocol operations. HSET query requests were investigated to find a normal performance. Selected number of requests are 1000, 10000, 50000, 100000, 500000 and 1000000. Each test was done three times then the results were used for standard deviation calculations.

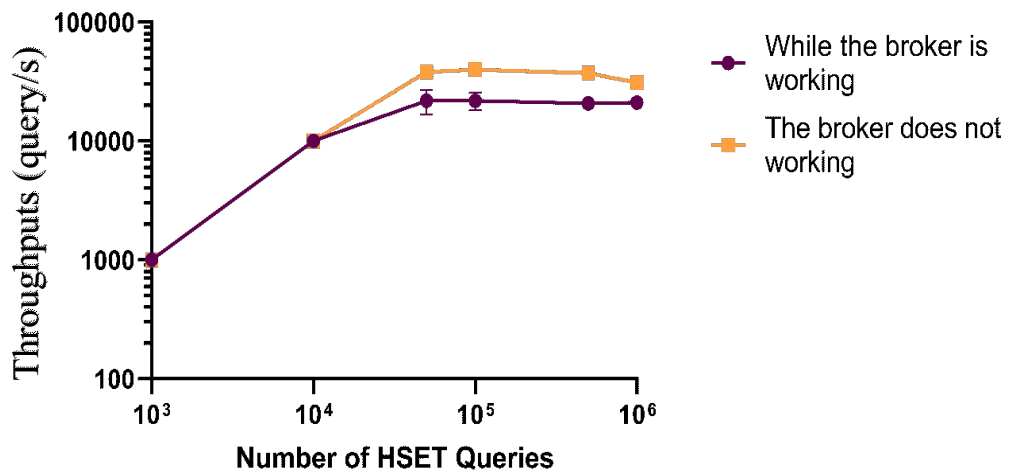


Figure 21. Throughput performance of HSET Query Requests

In Figure 21, the performance of the HSET operation was shown. Throughput was saturated after 10000 HSET queries. The performance evaluation was investigated in 2 test cases: the broker handled with messages and the broker does not have any a message. For the second case, better throughput can be observed for HSET queries compared to the first case. Moreover throughput is saturated around 20000 query/s and around 40000 query/s for first and second cases, respectively.

CHAPTER 6

DISCUSSION

There are a lot of studies exists for monitoring body vital signs rise and the proposed system is compared other developed systems. A weak part of the studies mentioned in Introduction Section is that they do not have an interactive user interface. In this project, the system design and implementation have been focused on user-friendly perspective. The proposed system has three important components: a device that is controlled by a microcontroller, the MQTT broker and chatbot. The whole system is controlled by a server that was written in Node.js engine. The architecture of the project same as another mentioned studies but its components are not similar. For example, MQTT protocol and its broker are common. But other projects used an existing platform for the broker. Existing platforms could be expensive so that the cost of the system can be increased. These platforms do not support custom designs for developers. Mosca MQTT broker was integrated into the system and a device publishes a message with QoS 0 to the broker. In this project, specified MQTT dependencies were decided for the speed of the data transmission and cost of the platform. Open source broker provides low cost solution and its performance is satisfactory.

The MQTT broker performance was calculated by a written benchmark. At the beginning of the performance evaluation, firstly, the broker was set up in the system and one virtual client was generated. A publisher has published a message and broker accepted. This benchmark step is same as other studies, but the main goal of the system is to measure and evaluate the performance of the system. Due to a lot of packet arrivals, the broker must handle whole published messages. The performance was evaluated using throughput, which depends on number of messages published in one-second interval. A result of measurement is put into a JSON packet that is generated by device. Then that packet is encapsulated into MQTT packet. Performance was calculated when MQTT packets arrived to the broker. As a result, the server tried to handle thousand messages in one second. Saturation started at a throughput value of 2×10^4 messages per second.

Increased packet (payload) size diminishes throughput. Packets size of 2 bytes and 16 bytes demonstrated similar performance, but the throughput was fallen dramatically to half when packet size was increased to 128 bytes. This shows that packet size important to publish a packet in high throughput. Current packet size in the proposed system is 127 byte and it could be decrease into 16 bytes by including only time and measured body temperature value in the packet structure to increase performance.

Connection of the database was provided in the Mosca broker. Published and arrived data is sent to the database with Redis query that is HSET. When number of received messages increases, the database's throughput will change. It was investigated the effect of thousands queries on the throughput. HSET query requests were used for evaluation. Test was analyzed for two cases: normal performance where the broker was working, and standalone performance where the broker was not working. Overall, throughput was saturated by increasing HSET queries and it can reach up to 40000 query/s.

Additionally, after a device-server management had been done, a user interface experiences were investigated. As mentioned before, various platforms were used in different projects. In the proposed system, a chatbot was used for efficient user interaction. It was integrated with Telegram BOT API and a rule-based chatbot was designed. It provides a lot of advantages. For example, it is a middleware between patient-doctor and user-server. A patient or doctor could not interrupt between each other at an unearthly hour. The users cannot disturb server or device operations. They are allowed talk to the system and control what they want in a rule-based script frame. Due to the advantages of chatbots, user does not need to install a secondary application to control the proposed system. Developing new application is time consuming and costly, because application should be checked frequently any structural bugs and it had to be implemented all platforms such as iOS and Android. One instant messaging application supplies the needs of server and user interaction. Monitoring body temperature could be also followed by chatbot instant warnings or real-time plots created using the measured data stored in the database.

The project offers a different perspective to literature by developing healthcare platform utilizing MQTT broker for communication between device-server and user-server, and chatbot as an user-interface to monitor and control the measurements. A user could reach defined menus in the chatbot and he/she choose one of them. Other parts of

the system work automatically, user does not have to handle with device operations, broker configuration and database storage problems. Cloud server which was written in Node.js programming language deals whole components the broker, devices and the database operations. The advantage of the system is the chatbot providing a familiar and user-friendly interface to the user. Moreover, users do not have to deal with other components of system. Proposed platform can handle one million concurrent messages with high throughput and without any loss. The length of the sent data could be decreased for better throughput performance.

CHAPTER 7

CONCLUSION

Detection and monitoring of body temperature is important for different diseases. Body temperature monitoring could give physiological situation of the patient. In this thesis, a system is developed to detect and monitor body temperature for febrile illnesses. New technological advancements like IoT technology is integrated in the system. This technology is helpful for controlling machine-to-machine communication via Internet without user interruption. Although the most of healthcare application which includes IoT has its user interfaces to inform users, these healthcare applications do not support interactive user interface for patients and doctors.

In this project, developed system was created with some principles. First, it must have interactive user interface. Secondly, cost of the used platforms should be low. Lastly, the system should be reliable.

With a literature survey, MQTT communication protocol and Telegram BOT API were chosen to be used for reliable, low-cost and interactive system. A main server designed with its components: Telegram chatbot router, Redis database connection and the MQTT broker connection. When the main server is started, the database, Telegram chatbot router and the broker will be ready for communication. The server was written in Node.js programming language.

The broker performance was measured while it was receiving published messages that were sending by a device. While number of published messages increase, the broker could handle the coming packets with a throughput up to 2×10^4 msg/s and without any loss. Moreover, database can handle incoming queries with high throughput also.

Rule-based chatbot design was used in the proposed system. For the future, state-machine should be applied for deep menu structure and answers of the chatbot. Although state-machine structure of menu is adequate for current state of the project, added artificial intelligence feature will increase user experience performance. The broker performance was studied over one publisher. However, multiple publishers can be utilized to evaluate system performance further. Although the broker's reserved RAM

size is enough, maximum space size for Node.js may be enlarged to enhance performance. Redis is used database for its speed of operations. When number of the data increase, the database performance will fall. In future work, Redis could be used as a cache and another powerful NoSQL database can be implemented into the Redis. Mosca library is used for the MQTT broker. In some cases, devices do not connect the MQTT broker due to unexpected situation. Redis has a feature to be used as a broker. Just in case, this feature of Redis can be integrated into the main server. By doing so, when a device could not connect the broker, it will join to Redis' broker without a problem.

REFERENCES

1. Al-Fuqaha, A., et al., *Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications*. IEEE Communications Surveys & Tutorials, 2015. **17**(4): p. 2347-2376.
2. Grgić, K., I. Špeh, and I. Heđi. *A web-based IoT solution for monitoring data using MQTT protocol*. in *2016 International Conference on Smart Systems and Technologies (SST)*. 2016.
3. Priya, L., R. Hariprasad, and R. Raghul. *Real time monitoring of vital signs using wireless technique*. in *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*. 2014.
4. Drews, F. and A. Doig, *Evaluation of a Configural Vital Signs Display for Intensive Care Unit Nurses*. Vol. 56. 2014. 569-80.
5. Sund-Levander, M., C. Forsberg, and L.K. Wahren, *Normal oral, rectal, tympanic and axillary body temperature in adult men and women: A systematic literature review*. Scandinavian Journal of Caring Sciences, 2002. **16**(2): p. 122-128.
6. Sinitsyn, A., E. Filippova, and O. Derevianko. *Development of a New Device for Human Body Temperature Regulation*. in *2018 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*. 2018.
7. Miah, M.A., et al. *Continuous heart rate and body temperature monitoring system using Arduino UNO and Android device*. in *2015 2nd International Conference on Electrical Information and Communication Technologies (EICT)*. 2015.
8. Salerno, J., et al., *Human–Wildlife Interactions Predict Febrile Illness in Park Landscapes of Western Uganda*. EcoHealth, 2017. **14**(4): p. 675-690.
9. Organization, W.H. *Pneumonia*. 2019 [cited April 2, 2019; Available from: <https://www.who.int/news-room/fact-sheets/detail/pneumonia>].

10. Hickey, L., *Review of Telemedicine Technologies: Information Technologies in Medicine and Telehealth* by Bernard Fong, A.C.M. Fong, and C.K. Li. Vol. 19. 2013.
11. Mala, K., et al. *A wearable diagnostic device to combat children's pneumonia*. in *2016 IEEE Global Humanitarian Technology Conference (GHTC)*. 2016.
12. Organization, W.H. *Malaria*. 2019 April 4, 2019]; Available from: <https://www.who.int/news-room/fact-sheets/detail/malaria>.
13. Shmaefsky, B.R., *Panic—The Silent Epidemic*, in *Meningitis*. 2005.
14. Zakaria, N.A., F.N.B.M. Saleh, and M.A.A. Razak. *IoT (Internet of Things) Based Infant Body Temperature Monitoring*. in *2018 2nd International Conference on BioSignal Analysis, Processing and Systems (ICBAPS)*. 2018.
15. Sali, S. and C.S. Parvathi. *Integrated wireless instrument for heart rate and body temperature measurement*. in *2017 2nd International Conference for Convergence in Technology (I2CT)*. 2017.
16. Sudevan, S. and M. Joseph. *Internet of Things: Incorporation into Healthcare Monitoring*. in *2019 4th MEC International Conference on Big Data and Smart City (ICBDSC)*. 2019.
17. usha rani, S., et al., *Iot Patient Health Monitoring System*. Vol. 8. 2017. 1329.
18. Sarierao, B.S. and A. Prakasarao. *Smart Healthcare Monitoring System Using MQTT Protocol*. in *2018 3rd International Conference for Convergence in Technology (I2CT)*. 2018.
19. Arora, J. and P.M. Yomsi, *Wearable Sensors Based Remote Patient Monitoring using IoT and Data Analytics*. U. Porto Journal of Engineering, 2019. **5**(1): p. 34-45.
20. Ramljak, M. *Smart home medication reminder system*. in *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. 2017.

21. Maria, A.R., et al. *MIoT Applications for Wearable Technologies Used for Health Monitoring*. in *2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. 2018.
22. Santamaria, A.F., et al., *A real IoT device deployment for e-Health applications under lightweight communication protocols, activity classifier and edge data filtering*. *Computer Communications*, 2018. **128**: p. 60-73.
23. Krynski, L., G. Goldfarb, and I. Maglio, *Technology-mediated communication with patients: WhatsApp Messenger, e-mail, patient portals. A challenge for pediatricians in the digital era*. *Arch Argent Pediatr*, 2018. **116**(4): p. e554-e559.
24. Coiera, E., *Essentials of Telemedicine and Telecare*. *BMJ*, 2002. **324**(7345): p. 1104.
25. Brown, N., *A brief history of telemedicine*. *Telemedicine Information Exchange*, 1995. **105**: p. 833-835.
26. Akhlaghi, H. and H. Asadi, *Essentials of telemedicine and telecare*. 2002: Chichester: Wiley.
27. Mohammadzadeh, N. and R. Safdari, *Patient monitoring in mobile health: opportunities and challenges*. *Medical archives (Sarajevo, Bosnia and Herzegovina)*, 2014. **68**(1): p. 57-60.
28. Wootton, R., *Telemedicine: a cautious welcome*. *BMJ*, 1996. **313**(7069): p. 1375.
29. Rubin, J.S. and R.T. Sataloff, *Telemedicine: Part I [Editorial]*. *Ear, Nose, & Throat Journal*, 2018. **97**(7): p. 186-187.
30. Nelson, C.A., et al., *Impact of store-And-forward (SAF) teledermatology on outpatient dermatologic care: A prospective study in an underserved urban primary care setting*. *Journal of the American Academy of Dermatology*, 2016. **74**(3): p. 484-490.e1.
31. Sasikala, S., K. Indhira, and V.M. Chandrasekaran, *Performance prediction of interactive telemedicine*. *Informatics in Medicine Unlocked*, 2018. **11**: p. 87-94.

32. Antunes, M., C. Silva, and J. Barranca, *A Telemedicine Application Using WebRTC*. *Procedia Computer Science*, 2016. **100**: p. 414-420.
33. Eysenbach, G., *What is e-health?* *Journal of medical Internet research*, 2001. **3**(2): p. E20-E20.
34. Uckelmann, D., M. Harrison, and F. Michahelles, *An architectural approach towards the future internet of things*, in *Architecting the internet of things*. 2011, Springer. p. 1-24.
35. Prokofiev, A.O., Y.S. Smirnova, and V.A. Surov. *A method to detect Internet of Things botnets*. in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIconRus)*. 2018.
36. Wu, G., et al., *M2M: From mobile to embedded internet*. *IEEE Communications Magazine*, 2011. **49**(4): p. 36-43.
37. Sundmaeker, H., et al., *Vision and challenges for realising the Internet of Things*. Cluster of European Research Projects on the Internet of Things, European Commission, 2010. **3**(3): p. 34-36.
38. Palattella, M.R., et al., *Standardized Protocol Stack for the Internet of (Important) Things*. *IEEE Communications Surveys & Tutorials*, 2013. **15**(3): p. 1389-1406.
39. Weber, R.M., *Internet of Things Becomes Next Big Thing*. *Journal of Financial Service Professionals*, 2016. **70**(6).
40. Mehaseb, M.A., et al., *Classification of LTE Uplink Scheduling Techniques: An M2M Perspective*. *IEEE Communications Surveys & Tutorials*, 2016. **18**(2): p. 1310-1335.
41. Din, I.U., et al., *The Internet of Things: A Review of Enabled Technologies and Future Challenges*. *IEEE Access*, 2019. **7**: p. 7606-7640.
42. Dimitrov, D.V., *Medical Internet of Things and Big Data in Healthcare*. *Health Inform Res*, 2016. **22**(3): p. 156-163.

43. Gómez, J., B. Oviedo, and E. Zhuma, *Patient Monitoring System Based on Internet of Things*. *Procedia Computer Science*, 2016. **83**: p. 90-97.
44. Atzori, L., A. Iera, and G. Morabito, *The Internet of Things: A survey*. *Computer Networks*, 2010. **54**(15): p. 2787-2805.
45. Ray, P.P., *A survey on Internet of Things architectures*. *Journal of King Saud University - Computer and Information Sciences*, 2018. **30**(3): p. 291-319.
46. Buyya, R. and A.V. Dastjerdi, *Internet of Things: Principles and paradigms*. 2016: Elsevier.
47. Lin, J., et al., *A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications*. *IEEE Internet of Things Journal*, 2017. **4**(5): p. 1125-1142.
48. Patra, L. and U.P. Rao. *Internet of Things — Architecture, applications, security and other major challenges*. in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2016.
49. Khan, R., et al. *Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges*. in *2012 10th International Conference on Frontiers of Information Technology*. 2012.
50. Kraijak, S. and P. Tuwanut. *A survey on internet of things architecture, protocols, possible applications, security, privacy, real-world implementation and future trends*. in *2015 IEEE 16th International Conference on Communication Technology (ICCT)*. 2015.
51. Masek, P., et al., *Implementation of True IoT Vision: Survey on Enabling Protocols and Hands-On Experience*. *International Journal of Distributed Sensor Networks*, 2016. **12**(4): p. 8160282.
52. Yassein, M.B., M.Q. Shatnawi, and D. Al-zoubi. *Application layer protocols for the Internet of Things: A survey*. in *2016 International Conference on Engineering & MIS (ICEMIS)*. 2016.

53. Nastase, L. *Security in the Internet of Things: A Survey on Application Layer Protocols*. in *2017 21st International Conference on Control Systems and Computer Science (CSCS)*. 2017.
54. Karagiannis, V., et al., *A survey on application layer protocols for the Internet of Things*. Vol. 3. 2015. 11-17.
55. Saritha, S. and V. Sarasvathi. *A study on application layer protocols used in IoT*. in *2017 International Conference on Circuits, Controls, and Communications (CCUBE)*. 2017.
56. Naik, N. *Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP*. in *2017 IEEE International Systems Engineering Symposium (ISSE)*. 2017.
57. Talaminos-Barroso, A., et al., *A Machine-to-Machine protocol benchmark for eHealth applications – Use case: Respiratory rehabilitation*. *Computer Methods and Programs in Biomedicine*, 2016. **129**: p. 1-11.
58. Duan, R., X. Chen, and T. Xing. *A QoS Architecture for IOT*. in *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*. 2011.
59. Soni, D. and A. Makwana, *A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT)*. 2017.
60. Alam, K.M. and A. Akram, *A Survey on MQTT Protocol for the Internet of Things*. Khulna University, Dept. of Computer Science and Engineering (CSE), 2016.
61. Madhu, D., et al. *A novel approach for medical assistance using trained chatbot*. in *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*. 2017.
62. Shawar, B.A. and E. Atwell. *Chatbots: are they really useful?*
63. Al-Zubaide, H. and A.A. Issa. *OntBot: Ontology based chatbot*. in *International Symposium on Innovations in Information and Communications Technology*. 2011.

64. Vaira, L., et al. *MamaBot: a System based on ML and NLP for supporting Women and Families during Pregnancy*. ACM.
65. Mostaço, G., et al., *AgronomoBot: a smart answering Chatbot applied to agricultural sensor networks*. 2018.
66. Weizenbaum, J., *ELIZA---a computer program for the study of natural language communication between man and machine*. Communications of the ACM, 1966. **9**(1): p. 36-45.
67. Nadelson, T., *The Inhuman Computer / the Too-Human Psychotherapist*. American Journal of Psychotherapy, 1987. **41**(4): p. 489-498.
68. McNeal, M.L. and D. Newyear, *Introducing chatbots in libraries*. Library technology reports, 2013. **49**(8): p. 5-10.
69. Batacharia, B., et al., *CONVERSE: A conversational companion*. 1999.
70. Yan, M., et al. *Building a chatbot with serverless computing*. ACM.
71. Huang, C., et al. *A Chatbot-supported Smart Wireless Interactive Healthcare System for Weight Control and Health Promotion*. in *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. 2018.
72. Nayyar, A. and V. Puri. *A review of Arduino board's, Lilypad's & Arduino shields*. in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2016.
73. *What is Arduino?* [cited 2019 May]; Available from: <https://www.arduino.cc/en/Guide/Introduction>.
74. *Getting started with the Arduino/Genuino MKR1000*. [cited 2019 May]; Available from: <https://www.arduino.cc/en/Guide/MKR1000>.

75. *MKR1000 Overview*. [cited 2019 May]; Available from: https://www.arduino.cc/en/Main/ArduinoMKR1000?s_tact=C3970CMW.
76. O'Leary, N. *PubSubClient*. PubSubClient-2.7.0:[Available from: <https://www.arduinolibraries.info/libraries/pub-sub-client>.
77. *Digital plug & play infrared thermometer in a TO-can MLX90614*. [cited 2019 May]; Available from: <https://www.melexis.com/en/product/MLX90614/Digital-Plug-Play-Infrared-Thermometer-TO-Can>.
78. *About Node.js*. [cited 2019 April]; Available from: <https://nodejs.org/en/about/>.
79. Tilkov, S. and S. Vinoski, *Node.js: Using JavaScript to Build High-Performance Network Programs*. IEEE Internet Computing, 2010. **14**(6): p. 80-83
80. Jing, H., et al. *Survey on NoSQL database*. in *2011 6th International Conference on Pervasive Computing and Applications*. 2011.
81. Li, S., H. Jiang, and M. Shi. *Redis-based web server cluster session maintaining technology*. in *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. 2017.
82. Carlson, J.L., *Redis in action*. 2013: Manning Publications Co.
83. *Number of apps available in leading app stores as of 3rd quarter 2018*. 2019 [cited 2019 April]; Available from: <https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/>.
84. Brandtzaeg, P. and A. Følstad, *Chatbots*. interactions, 2018. **25**(5): p. 38-43.
85. *Telegram FAQ*. [cited 2019 1 May]; Available from: <https://telegram.org/faq#q-what-is-telegram-what-do-i-do-here>.
86. Dargahi Nobari, A., N. Reshadatmand, and M. Neshati. *Analysis of Telegram, An Instant Messaging Service*. ACM.

87. Oliveira, J.C.d., D.H. Santos, and M.P. Neto. *Chatting with Arduino platform through Telegram Bot*. in *2016 IEEE International Symposium on Consumer Electronics (ISCE)*. 2016.
88. Assenmacher, D., et al., *Openbots - An Empirical Study on Automated Programs in Social Media*. 2019.
89. Collina, M. *Mosca*. [cited 2019 May]; MQTT Broker as a module]. Available from: <http://www.mosca.io/>.
90. *Node-telegram-operation-manager*. [cited 2019 May]; Available from: <https://github.com/alexandercerutti/node-telegram-operation-manager>.
91. *Chart.js*. [cited 2019 May]; Simple yet flexible JavaScript charting for designers & developers]. Available from: <https://www.chartjs.org/>.