DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DESIGN AND IMPLEMENTATION OF IOT GATEWAY FOR MULTI-PURPOSE SENSOR NETWORKS

by
Fatih DİCLE

November, 2019
İZMİR

DESIGN AND IMPLEMENTATION OF IOT GATEWAY FOR MULTI-PURPOSE SENSOR NETWORKS

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Computer Engineering, Computer Engineering Program

by
Fatih DİCLE

November, 2019
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"DESIGN AND IMPLEMENTATION OF IOT GATEWAY FOR MULTI-PURPOSE SENSOR NETWORKS"** completed by **FATİH DİCLE** under supervision of **ASSOC. PROF. DR. SEMİH UTKU** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Semih UTKU

Supervisor

Prof. Dr. Alp Kut

(Jury Member)

Prof. Dr. Emin Anarım

(Jury Member)

Prof.Dr. Kadriye ERTEKİN

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENTS

First and foremost, I would like to thank my thesis advisor Asst. Prof. Dr. Semih UTKU and Asst. Prof. Dr. Mehmet Hilal ÖZCANHAN for their patient, helpful and systematic guidance throughout the writing of this thesis. In addition, I would like to thank my professors and fellow research assistants in Department of Computer Engineering, Faculty of Engineering, Dokuz Eylül University for their support and assistance during my master's degree education.

Secondly, I would like to thank Prof. Dr. Mehmet Ufuk ÇAĞLAYAN and Prof. Dr. Emin ANARIM for their support and guidance during my education and for my thesis.

Thirdly, I would like to thank Gökhan AĞIŞ, Gürcan ALDEMİR, İbrahim ÇÖREKÇİ, Süleyman DÖNMEZ, Güray TURAN and Burcu YANKOVAN for helping me on realization of my thesis project and related materials during their undergraduate studies and internships with our department.

Lastly, I would like to thank my parents, Zafer DİCLE and Aklime SARIKAYA for their unending love and labor they have done to raise me and make me the man I am today and my brother Yiğit DİCLE for being the best brother a person can ask for and supporting me in every way he can.

Fatih DİCLE

DESIGN AND IMPLEMENTATION OF IOT GATEWAY FOR MULTI-PURPOSE SENSOR NETWORKS

ABSTRACT

Today, the Internet of Things (IOT) studies and applications are becoming increasingly common. In these studies and applications, the creation of and giving services to different sectors with these sensor networks are realized. However, as these systems grow in size and number, intermediate layers have become a requirement. It has become a necessity to develop general collection schemes in which data can be collected through the use of intermediate nodes from different sensor networks. In this study, an IoT Gateway that can collect data from different sensor networks in IoT application domains are planned, tested and simulated. It will be ensured that the IoT Gateway that have been developed with this study, will be used in related future work and integrated to sensor networks in different sectors.

IoT (Internet of Things) studies have been examined by conducting a literature review. Then, a simple sensor network has been constructed by creating the necessary sensor infrastructure. In this sensor network system, some structures will be created to be used as Data Collection Point (VTN) and Data Focus Point (VON). With these collection units, it is ensured that the data coming from the sensor network will be collected and transmitted to a central database. A general purpose collection middle layer structure that can collect data from different sensor network structures with a Gateway has been created. This Gateway has been built and implemented on this architecture. By using different methods and the developed Gateway, it is aimed to provide control of the work flow for different situations.

Keywords: Internet of things, IoT, gateway, MQTT, ns-3

ÇOK AMAÇLI SENSÖR AĞLARI İÇİN IOT GEÇİDİ TASARIMI VE UYGULAMASI

ÖZ

Günümüzde IoT (Internet of Things) çalışmaları giderek yaygınlaşmaktadır. Bu çalışmalarda farklı sensör ağlarının oluşturulması ve farklı sektörlere bu ağlar ile hizmetlerin sunulması gerçekleştirilmektedir. Ancak bu sistemler çoğaldıkça ara katmanlara ihtiyaç duyulmaktadır. Ara katmanlar ile farklı sensör ağlarından verilerin toplanabileceği genel toplama yapılarının geliştirilmesi gerekmektedir. Bu çalışmamız ile IoT uygulama alanlarında, farklı sensör ağlarından gelecek verileri toplayabilecek bir IoT Gateway oluşturulması planlanmış, test edilmiş ve simule edilmiştir. Çalışmamız ile geliştirilecek Gateway'in ilerideki çalışmalarımızda kullanılması ve farklı sektörlerdeki sensör ağları yapılarına entegre edilecektir.

Literatür taraması yapılarak IoT (Internet of Things) çalışmaları incelenmiştir. Daha sonra gerekli olan sensör alt yapısı oluşturularak, basit düzeyde bir sensör ağı kurulmuştur. Bu sensör ağı sisteminde, Veri Toplama Noktası (VTN) ve Veri Odak Noktası (VON) olarak kullanılacak yapılar oluşturulmuştur. Bu toplama birimleri ile sensör ağından gelecek olan verilerin toplanması ve merkezi veri tabanına iletilmesi sağlanmıştır. Ara katmanda oluşturulması planlanan Gateway (geçiş yolu) ile farklı sensör ağ yapılarından veri toplayabilecek genel amaçlı bir toplama yapısı oluşturulmuştur. Geliştirilmiş olan Gateway bu mimari üzerine kurulmuş ve uygulaması yapılmıştır. Farklı yöntemlerle, Gateway kullanımı sağlanarak, çalışma yapısının farklı durumlar için kontrolü sağlanmıştır.

Anahtar Kelimeler: Nesnelerin interneti, IoT, geçit, MQTT, ns-3

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

1.1 Internet of Things

The Internet of Things (referred as IoT hereafter) is an emerging technology that was made possible by the miniaturization and cost reduction of embedded devices which allowed them to be widely used in many environments and applications (Buyya & Dastjerdi, 2016). The term “IoT” was coined by Kevin Ashton and was used in a presentation in 1999 for Procter & Gamble (P&G) company (Ashton, 2009).

IoT paradigm can be considered as a natural expansion and evolution of networks and the Internet in general. In this paradigm, each device within this network is given a unique identification number, just like TCP/IP protocol used in Internet, and they exchange data, information and in some cases instructions with each other (Atzori, Iera, & Morabito, 2010).

IoT can be viewed to contain different points of focus, main ones include Internet (the connectivity of included devices with each other and the Internet), things (physical presence and/or capabilities of devices) and semantics (the storage, organization, representation, etc. of the information that produced by the devices) (Atzori et al., 2010).

As an emerging technology, there are many different visions for the future of IoT. First is the standard “Networks of Networks” vision, where IoT connects different networks of devices (e.g. energy, media or other sectors) to increase the reach and power of Internet as a whole (Vermesan et al., 2009). Second is the “Internet of Everything (IoE)” vision where things, people, data, etc. are all unified and connected and working together (Evans, 2012). The third one is “Internet of Nano Things (IoNT)” which is an of IoE to the microscopic realm, with micro sensors and micro computers (Miraz, Ali, Excell, & Picking, 2015).

These innovations have allowed IoT systems, with integration with other Cloud technologies, to be implemented in different work and research areas for the purposes of widespread and automatic data gathering, work and social environment control and smart management of systems (Buyya & Dastjerdi, 2016).

Currently, the number of IoT devices worldwide are estimated to be 7 billion (Lueth, 2018). However, estimates differ based on the definition of IoT devices. Estimates also differ about the possible future number of IoT devices worldwide.

According to (Nordrum, 2016) the projected number of IoT devices in 2020 or 2021 will be between 20 and 30 billion devices worldwide, a significant reduction from previous estimated figure, up to 50 billion devices. Comparison of different estimates are given in Figure 1.1 below.

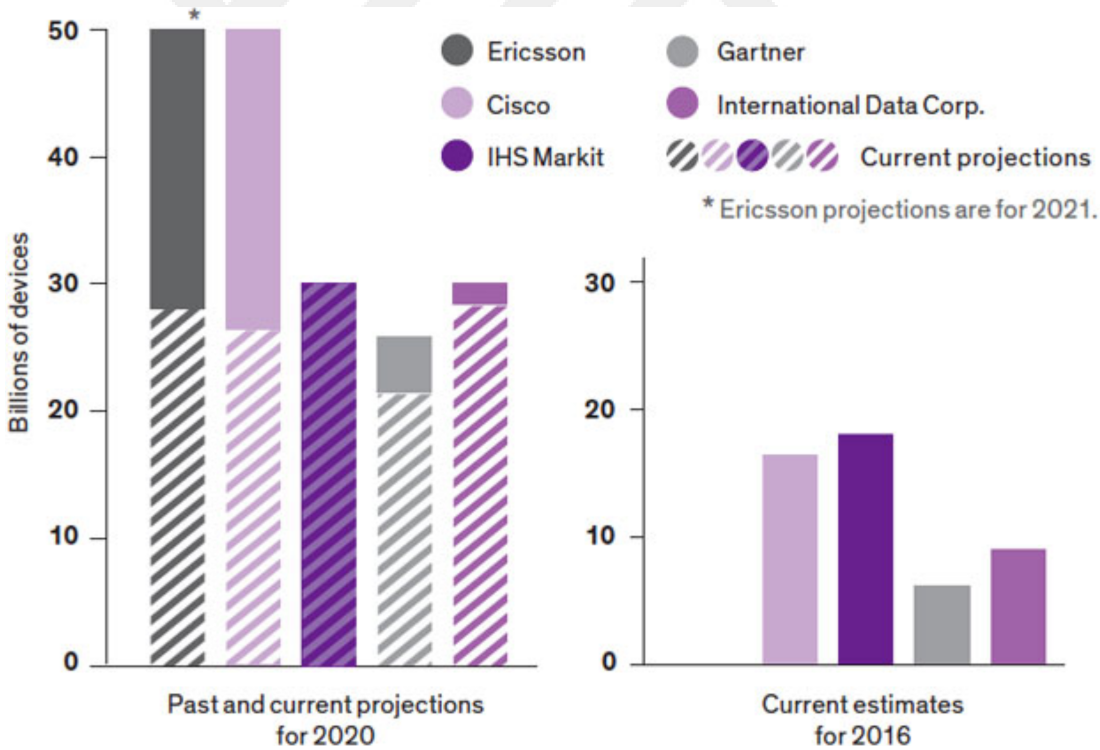


Figure 1.1 Comparison of different estimates of IoT devices worldwide by different companies and organizations (Nordrum, 2016)

In contrast, according to (Lueth, 2018), the prediction for the number of IoT devices worldwide for the same time frame is from 9.9 billion to 11.6 billion. Their estimates for possible numbers of IoT devices in the future is given in Figure 1.2, while their predicted classifications of these possible IoT devices is given in Figure 1.3.

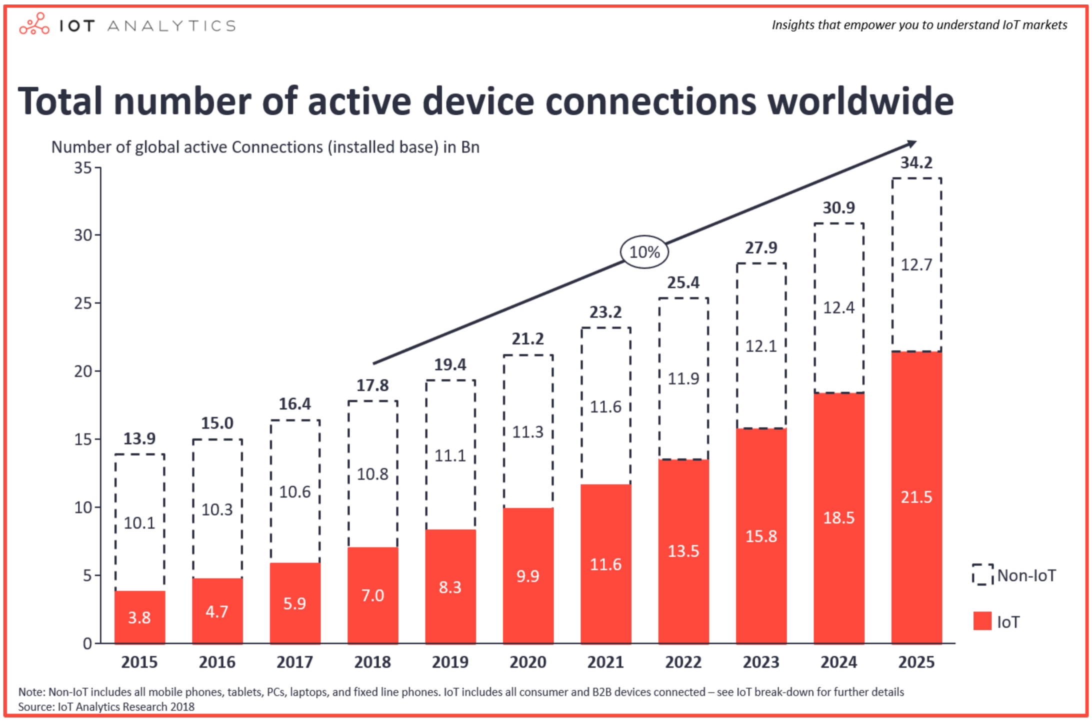


Figure 1.2 Estimates of IoT devices worldwide from 2015 to 2025 according to IoT Analytics website (Lueth, 2018)

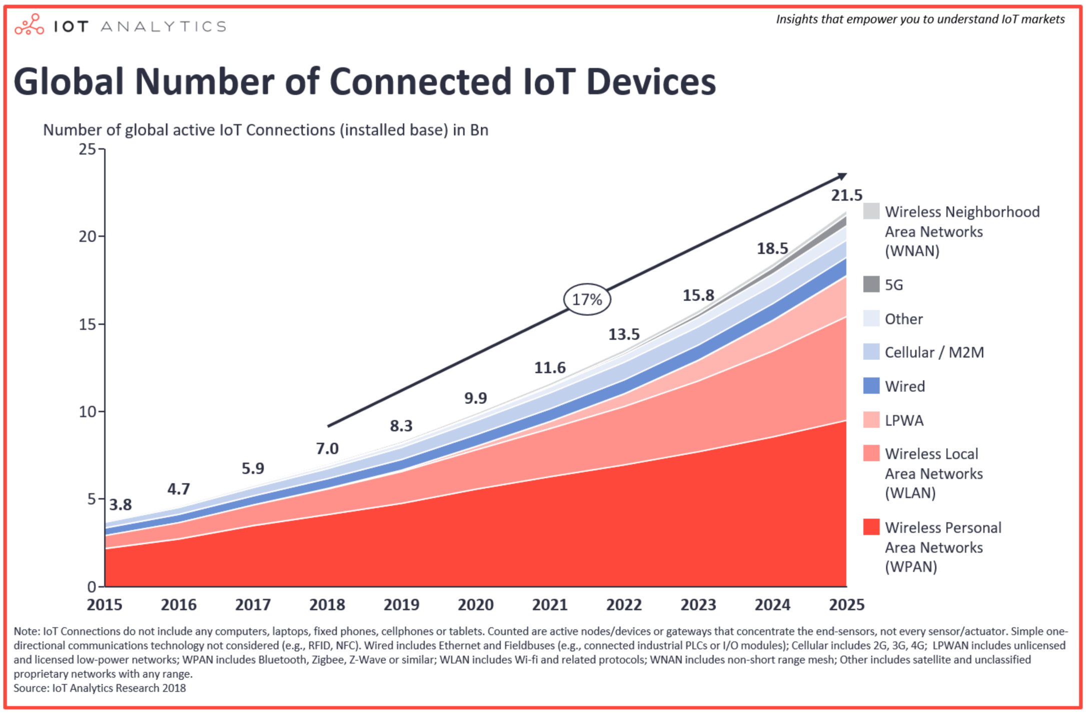


Figure 1.3 Classification of IoT devices worldwide from 2015 to 2025 according to IoT Analytics website (Lueth, 2018)

Also in (Lueth, 2018), the author gave examples for different kinds of IoT devices they considered in their classification scheme and in Figure 3 below. These examples are given in Table 1.1 below.

Table 1.1 Examples of IoT devices and their classifications (Lueth, 2018)

CATEGORY	CLASSIFICATION	EXAMPLES
Non IoT Devices	No Classification	Mobile Phones Tables Personal Computers Laptops Fixed Line Phones
IoT Devices	Wireless Personal Networks (WPAN)	Bluetooth Devices Headsets Zigbee Z-wave Smoke Alarms Thermostats
	Wireless Local Area Networks (WLAN)	Home Assistants Smart TVs Smart Speakers Factories
	Low-power Wide Area Networks (LPWAN)	Sigfox Lora NB-IoT
	Wired	Industrial Settings Field buses
	Cellular/ M2M	No Example
	5G	No Example
	Wireless Neighborhood Area Networks (WNAN)	Wi-Sun JupiterMesh Utilities Field Area Networks In-Door Metering Systems
	Other	Satellite Unclassified Proprietary Networks

1.2 IoT Challenges

The rapid research and applications in the recent years have revealed many problems and challenges that these systems suffer or may likely to suffer in the future. The category names and brief explanations of these challenges that has been mentioned in literature is available chapter two, Literature Review, of this thesis.

1.3 IoT Gateway

The definition of IoT gateway has been given in many different publications. According to (Zhu, Wang, Chen, Liu, & Qin, 2010), the purpose of IoT gateway is to counter the heterogeneity of different networks (e.g. Internet, Sensor Networks, etc.), to allow them to communicate with each other and allow the management of WSNs over different networks.

The range of IoT gateway could be far-reaching as the number of IoT devices increase as time passes. As stated before, when we have very different number of devices, with very different communication protocols, purposes and architectures, a gateway like device is required to enable communication between these devices and networks, to enable management as well. For this reason, the range of IoT Gateway will continue to expand.

The requirements for IoT gateways could be deduced by the duties they are expected to perform. As stated above, they need to support wide variety of protocols, systems and architectures to enable them to connect and work together, they need to be cheap to manufacture and use (e.g. low energy consumption), they need to be reachable over Internet for monitoring and maintenance of devices that may belong to any of the connected networks that are managed by this gateway. An important factor would be the computation power required by a gateway such as this, and it is the opinion of this thesis that a device capable of supporting an operating system (OS) is required to have a fully functioning gateway. The main reason for this choice is that an OS will allow researchers or developers alike to use and run a lot of different programs (written in different programming languages) simultaneously, unlike other primitive embedded systems (e.g. Arduino, a single thread device that only runs C language programs).

1.4 Purpose and Contribution of the Thesis

The purpose of this thesis is to examine the requirements of an IoT gateway that can be constructed to satisfy the requirements of a multi sensor networks. The requirements we have identified are as follows; the requirement to merge and compress the data that is generated by the sensors and lower level elements of the network, the requirement to encrypt and secure the generated data, the requirement to manage and monitor the lower level elements of the network, to send the data generated by the sensors to the cloud servers of the network with data verification and protection.

After the identification and examination of the requirements, the design and modeling of the gateway and general structure of a multi sensor network will be done and it will be tested and implemented on smaller scale with different configurations.

Lastly, we will conduct a simple simulation to see the feasibility and reliability of our gateway design in ns-3 simulation program.

1.5 Organization and Structure of the Thesis

This thesis is made up of six different chapters and their content and structure are briefly explained at below:

In The first chapter, background information related to Internet of Things and IoT gateway are given. In addition, challenges that IoT and IoT gateway research currently face are mentioned. Lastly, the purpose, contribution and structure of this thesis is given in this chapter.

In the second chapter, a literature review of proposed IoT and IoT gateway systems are shown. Next, a detailed and categorized survey of identified challenges in literature is given and explained. Lastly, a detailed examination and comparison of IoT gateway implementations are shown.

In the third chapter, technologies, software and hardware that was used in this thesis are given. These include MQTT protocol, Mosquitto software and library and comparison of MQTT and MQTT-SN. In addition, ns-3 network simulator, Python programming language and PyCharm IDE is shown and explained. Lastly, hardware that was used in this thesis, which includes Raspberry Pi 3 Model B and D1 Mini single board systems, are shown.

In the fourth chapter, small scale proof of concept prototypes of the proposed system are done and shown, general design of the whole system is shown, message structure that was designed for this system is explained and design diagrams of the proposed system are given.

In the fifth chapter, the message format that was designed in the previous chapter is redesigned and improved for the implemented system in this chapter. Next, a detailed explanation and screenshots of programming codes and user interfaces of the implementation of the designed IoT system and IoT gateway are shown. Lastly, the simulation of previously designed system is done in ns-3 simulation software and the resulting outputs and screenshots are shown.

Lastly, in the sixth chapter, comparison between the original goals and results of the thesis is given, critical discussion about the thesis is done, the role and effect of used technology in this thesis is examined and lastly, possible future development and future work is discussed.

CHAPTER TWO
LITERATURE REVIEW

2.1 Proposed IoT and Gateway Systems in Literature

In (Grønbæk, 2008), the author proposes an architecture of an IoT system that uses radio signals and Host Identity Tag (referred as HIT hereafter) for communication between the networked devices and an API for easier creation and management of such a system. The article also contains a proposed gateway structure for such a network, containing both TCP/IP and radio signal communication capabilities. The author also notes that the proposed architecture could be used as a stepping stone for an IoT architecture that is using TCP/IP communication on all levels, but warns that without cooperation between different organizations and groups in related fields, compatibility will be a problem.

In (Hunkeler, Truong, & Stanford-Clark, 2008), the authors implemented MQTT-S (currently named as MQTT-SN (Piper, 2013; Stanford-Clark & Truong, 2013) protocol in a Wireless Sensor Network (referred as WSN hereafter) they used as a testbed. The authors choose MQTT-SN over MQTT because it can be used with devices that have higher degree constraints to satisfy compared to those who don't (e.g. available electricity power, communication bandwidth, computation speed, etc.). However, they also noted that, using MQTT-SN have revealed many challenges in a WSN system configuration, which they explained in their paper.

In (Chen, Shu, Zhang, Liu, & Sun, 2009), the authors proposed a data aggregation tree structure between network elements to decrease message travel path, time and unnecessary repetition, probabilistic and machine learning models to estimate and predict possible sensor readings from the system, thereby reducing the need for data transference and system packet traffic.

In (Atzori et al., 2010), the authors conducted a comprehensive survey of IoT technologies, research areas and applications. They gave brief definition and usage scenarios of gateways in different IoT network models they examined.

In (Riedel et al., 2010), the authors tried to design a Device Profile for Web Services (referred as DPWS hereafter) gateway that can be used for different IoT products and technologies that are required to have long lifespan in industrial and related applications. They achieve this through model driven code generation and vertical integration of IoT elements, enabling them to control and update the software of gateways for longer usage lifespan.

In (Zhu et al., 2010), the authors have developed a IoT gateway using Zigbee-GPRS hardware, software and protocols. Their work is similar to work that are done in this thesis, albeit with different models.

2.2 IoT Challenges that was Proposed or Identified in Literature

As it was previously briefly mentioned in chapter one, many different challenges have been proposed in literature. These challenges have been organized and given in Table 2.1. The "Category of Challenges" column contains the given categories, if it is given at all, by the referenced publication, explicitly in most cases. The "Named or Identified Challenges" column contains the given challenges of IoT, either by explaining it in detail or just mentioning, by the referenced publication.

Table 2.1 A collection of IoT challenges from different scientific publications

PUBLICATION ID OR REFERENCE	CATEGORY OF CHALLENGES (IF GIVEN)	NAMED OR IDENTIFIED CHALLENGES
(Atzori et al., 2010)	Open Research Issues	Standards
		Mobility Support
		Naming
		Transport Protocol
		Traffic Characterization and QoS Support
		Authentication
		Data Integrity
		Privacy
		Digital Forgetting
(Ma, 2011)	No Given Category	Data exchange among large-scale heterogeneous network elements.
		Effective integration and interaction adaptation of uncertain information.
		Service adaptation in the dynamic system environment.
(Miorandi, Sicari, De Pellegrini, & Chlamtac, 2012)	Research Challenges	Computing, Communication and Identification Technologies
		Distributed Systems Technology
		Distributed Intelligence
(Chase, 2013)	No Given Category	Connectivity
		Power Management
		Security
		Complexity
		Rapid Evolution
(Gubbi, Buyya, Marusic, & Palaniswami, 2013)	No Given Category	Architecture
		Energy Efficient Sensing
		Secure Reprogrammable Networks and Privacy
		Quality of Service
		New Protocols
		Participatory Sensing
		Data Mining
		GIS Based Visualization

Table 2.1 continues

		Cloud Computing
		International Activities
(Singh, Tripathi, & Jara, 2014)	No Given Category	Communication Mechanism — 6lowpan Challenges
		Data Fusion Mechanism and Challenges
(Hossain, Fotouhi, & Hasan, 2015)	Information Security Requirements	Integrity
		Information Protection
		Anonymity
		Non-Repudiation
		Freshness
	Access Level Security Requirements	Authentication
		Authorization
		Access Control
	Functional Security Requirements	Exception Handling
		Availability
		Resiliency
		Self organization
(Buyya & Dastjerdi, 2016)	Stream Processing in IoT	Scalability
		Robustness
		SLA-Compliance
		Load Balancing
	IoT Robustness and Reliability	Making Service Available to User
		Serviceability of IoT System
		Reliability at Network Level
		Device Level Reliability
	Internet of Vehicles and Applications	Poor network connectivity and stability
		Hard delay constraints
		High reliability requirements
		High scalability requirements
		Security and privacy
		Service sustainability
(Díaz, Martín, & Rubio, 2016)	No Given Category	Security and Privacy
		IPv6
		Fog Computing
		Lambda Architecture
		Interoperability
		Context-Aware Computing

Table 2.1 continues

(Stout & Urias, 2016)	List of the Top 10 IoT Vulnerabilities	Insecure Web Interface
		Insufficient Authentication/Authorization
		Insecure Network Services
		Lack of Transport Encryption
		Privacy Concerns
		Insecure Cloud Interface
		Insecure Mobile Interface
		Insufficient Security Configurability
		Insecure Software/Firmware
		Poor Physical Security
(Thakare, Patil, & Siddiqui, 2016)	No Given Category	Security
		Privacy
		Standards and interoperability
		Scalability
		Low power communication
		Security threats from ubiquitous devices
		Debugging self-diagnosing, and automatic repair
(Abbasi, Memon, Memon, Syed, & Alshboul, 2017)	IoT Data Management	Standardization
		Data storage and management
		Confidentiality and privacy
		Integrity
		Energy constraints
		Device mobility and heterogeneity
		Device security and backup
		Availability
		Internal adversaries
(Zeinab & Elmustafa, 2017)	No Given Category	Scalability
		Self-Organizing
		Data volumes
		Data interpretation
		Interoperability
		Automatic Discovery
		Software complexity
		Security and privacy
		Fault tolerance
		Power supply
		Wireless communications

Table 2.1 continues

(Khalil & Özdemir, 2018)	Explained Challenges	Limited Resources
		Heterogeneous Structure
		Scalability
		Description (Identification)
		Search and Discovery
		Mobility
		Security and Privacy
		Data Confidentiality
		Authentication
		Management of Devices (Things)
	Mentioned Challenges	Lack of Available Internet Service
		Inability to develop sensor systems that are low cost and smart.
		Fault Tolerance
		Management of QoS
(Reyna, Martín, Chen, Soler, & Díaz, 2018)	No Given Category	Storage capacity and scalability
		Security: weaknesses and threats
		Anonymity and data privacy
		Smart contracts
		Legal issues
		Consensus

As it can be seen from Table 2.2, many of the named challenges are identical or very similar in nature. A categorization and organization of these is also has been done in this thesis to merge and simplify all of the proposed challenges in literature. The following topics and their subtopics are considered to be the main areas of challenges of IoT and also, IoT Gateway design and development.

2.2.1 Security

Protection of people, data and devices in a system from threats, both internal and external (Hossain et al., 2015). Current research topics include Authentication, Authorization, Freshness, Non-Repudiation (Hossain et al., 2015), Insecure interfaces, networks, services, software and hardware (Stout & Urias, 2016).

2.2.2 Privacy

Anonymity of the people, data and the devices in the system (Medaglia & Serbanati, 2010). Current research topics include Digital Forgetting (Atzori et al., 2010), to delete data that may be harmful to privacy and Secure Reprogrammable Networks (Gubbi et al., 2013), to keep the identity and data of actors (users and things alike) hidden.

2.2.3 Quality of Service (QoS)

QoS covers a large group of topics which include but not limited to, Context-Aware Computing (Díaz et al., 2016), Poor network connectivity and stability (Buyya & Dastjerdi, 2016) and Service adaptation in the dynamic system environment (Ma, 2011). QoS topics can also be divided into three categories that are given below:

2.2.3.1 Communication

Network messaging or other communication methods used between devices (M2M) to transmit data and information between each other and indirectly people. Current research topics include 6lowpan, Data Fusion (Singh et al., 2014), IPv6 (Díaz et al., 2016), Transport Encryption (Stout & Urias, 2016), Transport Protocol (Atzori et al., 2010), Wireless Communication (Zeinab & Elmustafa, 2017).

2.2.3.2 Reliability

The ability of a system to continue operations while or after experiencing significant crises and having multiple redundancies of every critical services for such events. Current research topics include Availability, Resiliency (Hossain et al., 2015), Data Integrity, Debugging self-diagnosing, and automatic repair, Interoperability and Robustness.

2.2.3.3 Standardization

Like all emerging technologies, IoT also requires standardization to enable easy access for developers and manufacturers to create. Current research topics include Consensus (Reyna et al., 2018), International Activities (Gubbi et al., 2013) and Standards and Interoperability (Thakare et al., 2016).

2.2.4 Scalability

As it has been shown before, IoT devices are increasing and will continue to increase in the future. Every IoT network or system must be designed to accommodate increasing number of devices, to work efficiently under high usage etc. Current research topics include Data exchange among large-scale heterogeneous network elements (Ma, 2011), High scalability requirements (Buyya & Dastjerdi, 2016), Mobility Support (Atzori et al., 2010) and Storage capacity and scalability (Reyna et al., 2018).

2.2.5 Management

Effective control and monitoring of IoT is essential because human machine interaction and communication is a vital part of it. This requires effective management to achieve. Management research covers wide array of topics which are grouped together in three categories below:

2.2.5.1 Information Management

One of the core functionalities of IoT is gathering data and information. How this data will be gathered, stored, analyzed and processed is an active research and engineering area. Research topics in this area include; data interpretation (Zeinab & Elmustafa, 2017), data storage and management (Abbasi et al., 2017), and effective integration and interaction adaptation of uncertain information (Ma, 2011).

2.2.5.2 Resource Management

As was stated earlier, IoT networks contain numerous devices and as this number increase, the resource requirements (e.g. electricity, production costs, computation power, etc.) of these systems also increase. Therefore, an effective management and control of resources used by systems is an important part of IoT research and design. Research topics in this area include; energy constraints (Abbasi et al., 2017), hard delay constraints (Buyya & Dastjerdi, 2016) and low power communication (Thakare et al., 2016).

2.2.5.3 System Management

The management of the whole system from holistic perspective. Previously explained topics can be considered specialized sub categories of this topic. Research topics in this area include; architecture and participatory sensing (Gubbi et al., 2013), automatic discovery and self-organization (Zeinab & Elmustafa, 2017), distributed intelligence and distributed systems technology (Miorandi et al., 2012) and fog computing (Díaz et al., 2016).

2.3 Comparison of IoT Gateway Implementations in Literature

There are different designs and implementations present in literature. A general overview, comparison and examination of them are given in Table 2.2.

Table 2.2 Examination and comparison of IoT gateway implementations in scientific literature

PUBLICATION ID OR REFERENCE	SENSOR AND / OR DEVICE CAPACITY	IMPLEMENTATION DONE? (HOW IT IS ACCOMPLISHED?)	TESTING DONE? (HOW IT IS ACCOMPLISHED?)	ADVANTAGES OF IMPLEMENTATION	DISADVANTAGES OF IMPLEMENTATION
(Emara, Abdeen, & Hashem, 2009)	They are using a VIP (Virtual IP Address) system to designate sensor nodes in a WSN. The device limit in this implementation depends on the chosen IP address range for sensors.	No implementation has been done, only a theoretical model has been proposed.	No testing has been done.	VIP system they are using allow dynamic increase or decrease of sensors in the system.	No details where given about the communication method of sensors in WSN. Not enough details are given about the system as a whole.
(Min, Xiao, Sheng, & Shiya, 2012)	32 devices per gateway (a limitation of RS485)	Using 1 STM32F107 as a gateway and 8 STC8051 to represent connected devices. Only simple message send and received is done.	No testing done other than simple experimental implementation.	It can be used with older hardware that have DB-9 port or other RS485 usable ports. It does have encryption for serial communication.	It uses older communication technologies that depends on physical layer, compared to TCP/IP and network protocols. Also, because it is older technology, the hardware to use it will be expensive and rare.
(Guoqiang, Yanming, Chao, & Yanxu, 2013)	According to their hardware design, their gateway support up to 8 user cards to connect other hardware. According to their communication protocol design, up to 2^{16} addresses are supported. Also, 2^{8} sensors are supported per device, which makes it 2^{24} in total.	No given implementation in paper. Assuming it was not done.	No given testing in paper. Assuming it was not done.	Through the use of User Cards, different communication standards are supported. Designed a data package format for communication between devices.	Their design seems to be incomplete or not very usable.
(Min, Xiao, Sheng, Quanyong, & Xuwei, 2014)	Their proposed gateway uses Bluetooth, CAN, Ethernet, GSM/GPRS, RS485 and Zigbee, therefore, these technologies connection limits apply.	No implementation has been, only a proposal and design is given.	A test has been done on task complementation times of different scheduling times, in MATLAB, however no algorithm is given about the calculations.	Their gateway design supports different number of communication and management technologies.	No implementation of their gateway has been given in the paper. Testing has been done in MATLAB instead of ns-3 or other network simulation software.
(Kim, Choi, & Rhee, 2015)	Their device ID field in JSON format contains 16 alphanumeric characters, however no specific explanation is given regarding device limit. Their proposed gateway uses Bluetooth, DPWS, MQTT and Zigbee, therefore, these technologies connection limits apply.	Done implementation of MQTT with an Arduino board (simple Publish and Subscribe example), DPWS with DPWSim (assuming only simulation) and REST on gateway device (to show received sensor data from devices).	No testing has been mentioned in the paper.	Their gateway design supports different number of communication and management technologies.	No full implementation of their gateway has been given in the paper, only partial implementations of proposed technologies.

Table 2.2 continues

(Glória, Cercas, & Souto, 2017)	The communication methods they used are USB and Ethernet. Capacity of Ethernet is dependent to the used address space and USB has a connection limit of 127 different devices, also 6 USB hubs, per USB host (Axelson, 2015).	They did a basic prototype implementation of their system, using a Raspberry Pi and an Arduino board. Raspberry Pi is connected to the server via Ethernet (and Internet if possible) and connected to the Arduino board with a USB port.	Aside from their implementation and its execution, no other testing has been done.	Their system is simple and well-designed. It is very similar to the author's design in this thesis.	No simulations have been done for very large IoT networks. Their prototype should have been explained in more detail.

With Table 2.2, a detailed literature review, general IoT challenges and examination and comparison of IoT gateway implementations has been finished. These research and examination results will guide future studies of researchers in the future.

CHAPTER THREE
INFRASTRUCTURE

To be able to realize the proposed system of this thesis, the following technologies, protocols, specialized software and hardware will be used. The explanations of these tools are given in this chapter.

3.1 MQTT (Message Query Telemetry Protocol)

In order to enable IoT systems to reach the maximum possible communication quality, a low energy and computation power consuming protocol is required, a very good example is MQTT (Vermesan & Friess, 2013).

MQTT is a protocol that works in Application layer (over TCP/IP protocol) and enables low bandwidth and low cost communication between devices that uses this protocol (Banks & Gupta, 2014). It is an open source, asynchronous protocol that supports hierarchical structure in the topic name of the shared information (e.g. "root_topic/sub_topic/another_sub_topic") (Vermesan & Friess, 2013).

The basic methods or commands of MQTT protocol as explained in (Karhula, 2016):

CONNECT: connecting from a client to a broker or from broker to client.
DISCONNECT: severing an established connecting by any device.
SUBSCRIBE: subscribing a topic in a broker by a client.
UNSUBSCRIBE: unsubscribing a topic by a client.
PUBLISH: sending data from client to broker or vice versa.

The protocol requires two types of devices to work, clients and brokers. The clients connect to brokers and subscribe to a topic name (and identifier for data) and send or receive data over it. Brokers act as central communication hubs and monitor the connected devices and their transmissions. In this use model, IoT Gateways become the best possible candidate as a broker because all devices from connected networks (sensors or servers or etc.) can connect and communicate with it. A simple representation of MQTT communication between devices are shown in the Figure below:

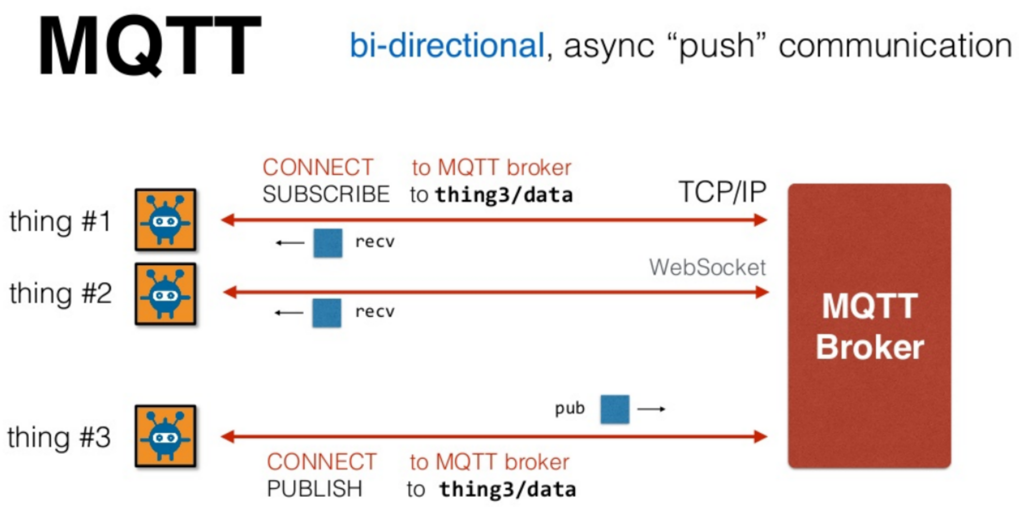


Figure 3.1 A simple representation of MQTT communication between a broker and client devices. Note that a client can subscribe to a topic without publishing any data and can publish data to a topic that it is not subscribed to (Boyd, 2014)

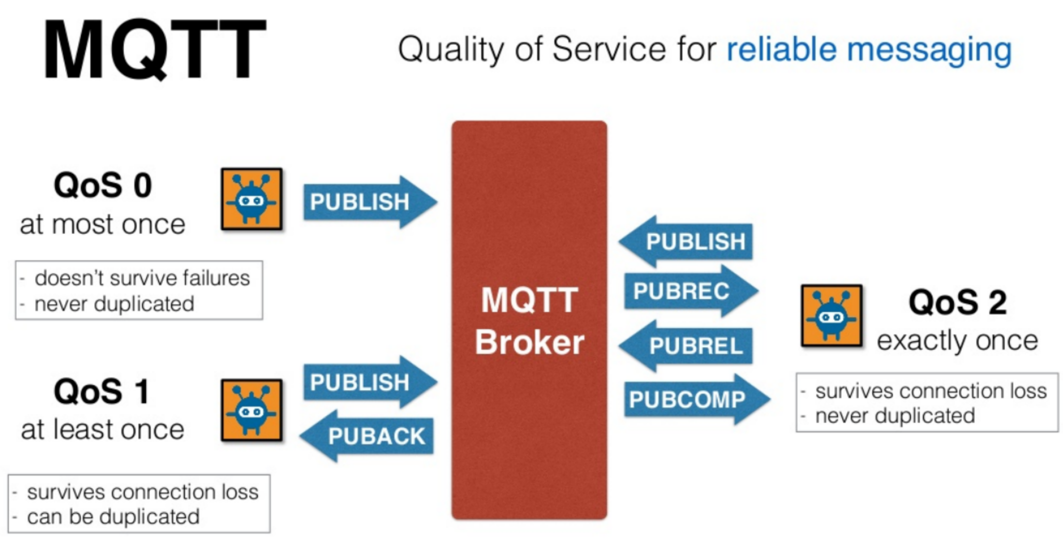


Figure 3.2 A simple representation of supported Quality of Service levels, operations and their differences in execution (Boyd, 2014)

3.1.1 Mosquitto

MQTT is a protocol that is well-defined and standardized. However, this project needs an implementation of this protocol to use in our system, otherwise, the author had to do a usable implementation too. The implementation that has been used in this project is Mosquitto (A Light, 2017; Eclipse Mosquitto, 2009). Compared to other known MQTT implementations, Mosquitto is the only one which supports both client and broker operations and that is published under a free and open source software license ("Comparison of MQTT implementations," 2019).

3.1.2 Comparison with MQTT-SN

MQTT-SN (Message Query Telemetry Protocol – Sensor Networks) is a lightweight version of MQTT, designed for IoT networks that have even more extreme limitations with communication and processing operations. Their incomplete list of differences are shown below as explained in (Cope, 2017; Kumar, 2017; Piper, 2013; Stanford-Clark & Truong, 2013):

TopicName in MQTT is replaced by TopicID (a 16-bit integer) in MQTT-SN to lower message and general communication size.

MQTT-SN have 3 different CONNECT messages compared to 1 in MQTT, extra messages are related to WILL option.

MQTT-SN have gateway discovery functionality for automatic translation between MQTT-SN and MQTT for compatibility with MQTT.

MQTT-SN is over UDP while MQTT is over TCP/IP to lower communication bandwidth.

MQTT-SN using devices can SLEEP to stop receiving messages and RESUME to get current and previously unsent messages and restart communication with gateway to save battery power etc.

3.2 ns-3 Network Simulator

ns-3 is one of the available network simulators with advanced configuration and free software license (ns-3, 2008). It has been used by very different engineering and research projects worldwide. In this thesis, it is mainly used to simulate and test a very large scale implementation of IoT Gateway model that has been designed. A general structure of ns-3 software environment is given at the figure below:

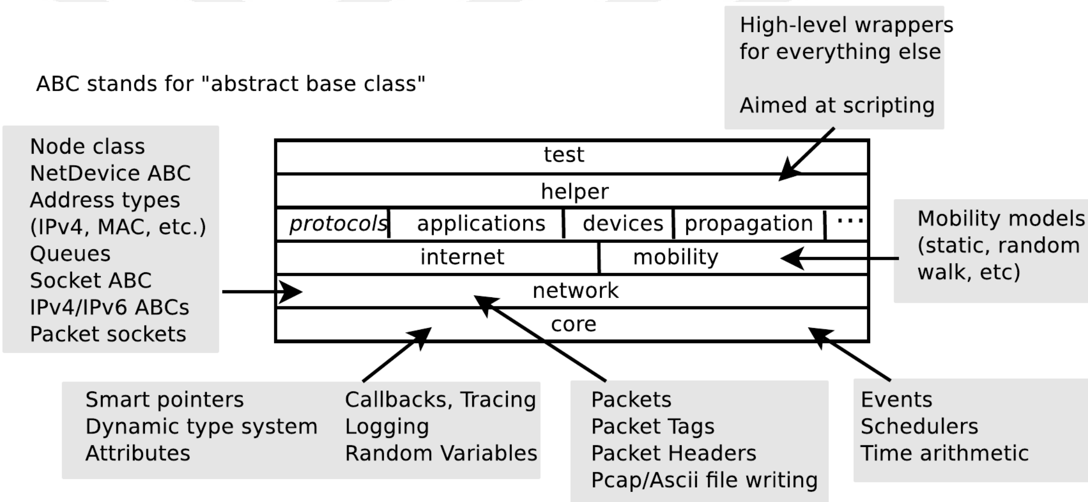


Figure 3.3 A representation of ns-3 software structure, showing programming objects, tools and their relationship with each other in hierarchical structure (“ns3 Manual,” 2019)

3.3 Python Programming Language

Python programming language has been used in this project for MQTT protocol and Mosquitto application on Raspberry Pi and PC. It has been chosen for its simplistic syntax, user-friendliness and wide range of compatibility with other software applications (Python, 1991).

3.3.1 PyCharm IDE

To develop and write the python software that has been used on this project, PyCharm IDE program has been used because of its user-friendly functionality and design (PyCharm, 2010).

3.4 Hardware Used in Project

For our testing and implementation, we have used two different hardware devices. First is Raspberry Pi 3 Model B (Raspberry Pi 3 Model B, 2016), a popular single board computer, capable of running a Linux OS that allow us to execute different software and programming language codes on a single device. Second is D1 Mini (D1 mini, 2017), a single board micro controller that can supports Wi-Fi communication technology and can be programmed via Arduino IDE (Arduino Software IDE, 2005) using C programming language.

CHAPTER FOUR
METHODOLOGY AND IMPLEMENTATION

4.1 Proof of Concept Prototyping

For this project, three different setups have been created to test and simulate IoT gateway. Used systems for this project include MQTT (MQTT, 1999), an M2M protocol and MQTT Dashboard (MQTT Dashboard, n.d.), a public Online MQTT broker to enable communication between devices without the requirement to install and configure a MQTT broker server yourself.

Firstly, D1 Mini (D1 mini, 2017), an embedded device has been connected to MQTT Dashboard to "PUBLISH" sensor data to a topic and another device has been connected to MQTT Dashboard too, to "SUBSCRIBE" to the same topic to receive sensor data published by D1 Mini. D1 Mini uses Arduino IDE (Arduino Software IDE, 2005), to program in C programming language and MQTT protocol is used via a library. The subscriber computer is using MQTT via Python, a scripting language with more flexibility than C programming language.

Figure 4.1 D1 Mini (D1 Mini Image, n.d.), is publishing a data packet to a topic in MQTT Dashboard (HiveMQ Logo, n.d.). Simultaneously, a computer (Desktop PC Image, n.d.) has subscribed to the same topic that D1 Mini is sending data and receives this data from MQTT Dashboard

After this connection is established, sensors are connected to D1 Mini, which in turn reads data from these sensors and sends it to another device (which in this case, is a desktop computer) to store or process it.

Secondly, Raspberry Pi 3 Model B (Raspberry Pi 3 Model B, 2016), a single board computer has been used to connect to MQTT Dashboard to “PUBLISH” data to a topic and, same as before, another device has been connected to MQTT Dashboard too, to “SUBSCRIBE” to the same topic.

Unlike D1 Mini, Raspberry Pi 3 can support an operating system and this enables users to use a wide range of tools and programming languages compared to D1 Mini. One of the most popular operating system for Raspberry Pi 3 is Raspbian (Raspbian, 2013), therefore is has been used as an OS for this project.

Python programming language has been used via PyCharm IDE to use MQTT protocol in both of devices, in a very similar method that has been used by the computer in previous experiment.

Figure 4.2 Raspberry Pi 3 (Raspberry Pi 3 Image, n.d.), is publishing a data packet to a topic in MQTT Dashboard (HiveMQ Logo, n.d.). Simultaneously, a computer (Desktop PC Image, n.d.) has subscribed to the same topic that Raspberry Pi 3 is sending data and receives this data from MQTT Dashboard, same operation as before

Same as it was done before, after this connection is established, sensors are connected to Raspberry Pi 3, which in turn reads data from these sensors and sends it to another device (which in this case, is a desktop computer) to store or process it.

4.2 IoT System Design

In this Project, we mainly focus on development of two devices, VTN (Turkish: **V**eri **T**oplama **N**oktası, English: Data Collection Point) and VON (Turkish: **V**eri **O**dak **N**oktası, English: Data Focus Point). As the name suggests, VTN is a lower level data gathering point directly from sensors or boards that are running the sensors. While VON, is a higher level data gathering and processing device compared to VTN and allows lower level devices to be connected to and reached from other devices in the network over Internet.

In Figure 4.3, you can see an illustration of the proposed IoT system and gateway. The reason for this multi layered approach is to increase the range and effective area of this system, without increasing the costs. Even though in the figure, lowest level elements are named Arduino, because they are one of the most popular and cheaply available devices that can be used for this function, they do not have to be used. In addition, even though sensors that are connected to Arduino devices are represented as independent devices, most of them require a connected embedded device to work and function.

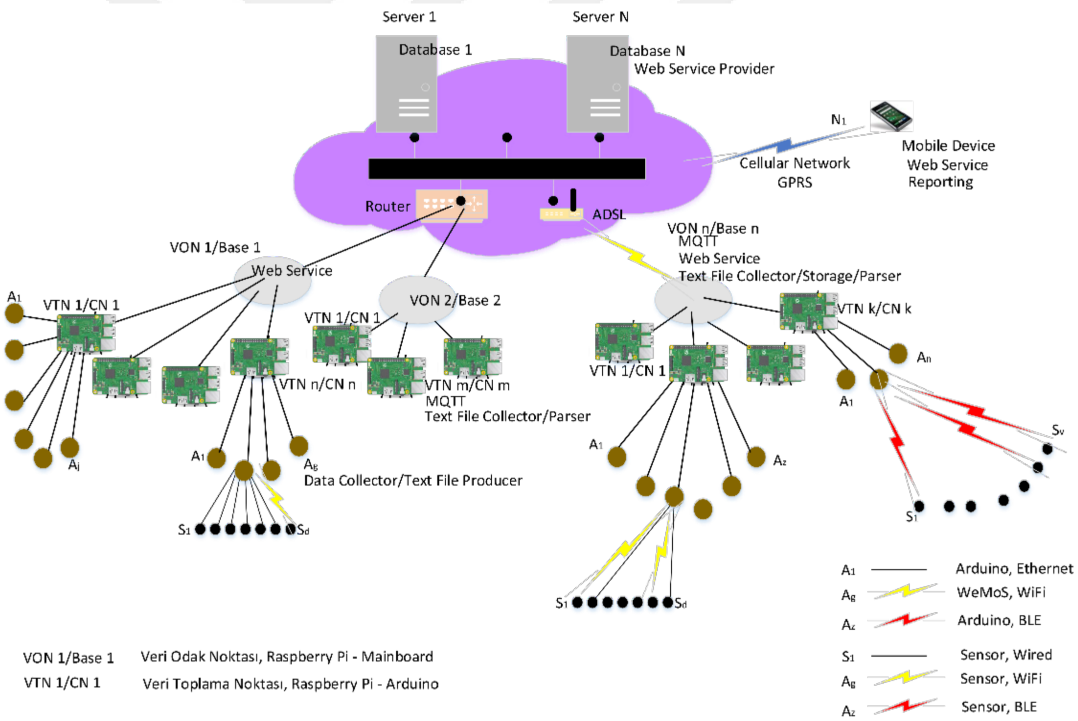


Figure 4.3 An illustration and structure of proposed IoT system and gateway in this thesis

In Figure 4.3, the system works by gathering data from sensors and Arduino, then sending and processing it to the devices in higher hierarchy, a mobile application can be used to monitor the IoT network, but not for control. It is not represented here, however, management and control operations are done by connecting to servers via web service. It is important to note that flow of information or communication is bidirectional, not unidirectional.

For the gateway to function in this IoT network, it must has a list of basic functions to work properly. In Table 4.1 below, a list of basic functions of IoT network devices is given with explanation.

Table 4.1 A list of basic functions of IoT gateway project

DEVICE NAME	FUNCTION NAME	INPUT	OUTPUT	EXPLANATION
BOARD	Read Sensor	Sensor ID	Sensor Data	Reading a sensor by a Board
VTN	Read Sensor	Sensor ID, Board ID	Sensor Data	Reading a sensor of a board by a VTN
VTN	Read Board	Board ID	A set of Sensor Data, Sensor ID	Reading all sensors, of a board by a VTN
VTN	Write Sensor Data	Sensor Data, Sensor ID, Board ID	Text File Record	Writing a record of a sensor of a board by a VTN to a text file
VTN	Read Text File	Previously Written Text File	A set of Sensor Data, Sensor ID, Board ID	Reading all previously written text file records by VTN
VON	Read Sensor	Sensor ID, Board ID, VTN ID	Sensor Data	Reading a sensor of a board, of a VTN by a VON
VON	Read Board	Board ID, VTN ID	A set of Sensor Data, Sensor ID	Reading all sensors, of a board, of a VTN by a VON
VON	Read VTN	VTN ID	A set of Sensor Data, Sensor ID	Reading all sensors, of all boards, of a VTN by a VON
VON	Write Sensor Data	Sensor Data, Sensor ID, Board ID, VTN ID	A Record in a Text File	Writing a record of a sensor of a board, of a VTN by a VON to a text file

Table 4.1 continues

VON	Read Text File	Previously Written Text File	A set of Sensor Data, Sensor ID, Board ID, VTN ID	Reading all previously written text file records by VON
SERVER	Read Sensor	Sensor ID, Board ID, VTN ID, VON ID	Sensor Data	Reading a sensor of a board, of a VTN, of a VON by a SERVER
SERVER	Read Board	Board ID, VTN ID, VON ID	A set of Sensor Data, Sensor ID	Reading all sensors, of a board, of a VTN, of a VON by a SERVER
SERVER	Read VTN	VTN ID, VON ID	A set of Sensor Data, Sensor ID, Board ID	Reading all sensors, of all boards, of a VTN, of a VON by a SERVER
SERVER	Read VON	VON ID	A set of Sensor Data, Sensor ID, Board ID, VTN ID	Reading all sensors, of all boards, of all VTN, of a VON by a SERVER
SERVER	Insert Record to Database	Sensor Data, Sensor ID, Board ID, VTN ID, VON ID	A Record in a Database	Writing a record of a sensor of a board, of a VTN, of a VON by a SERVER to a database
SERVER	Read Data from Database	One or a set of Search Parameters	One or a set of Records from Database	Reading one or a set of records from database by SERVER

Table 4.1 continues

MOBILE	Read Data from Server	One or a set of Search Parameters	One or a set of Records from Database	Retrieving one or a set of records from SERVER for analysis and monitoring purposes by MOBILE

4.3 Designed Message Structure

For this system, a message format that can be used by all devices in the IoT network has been developed. This format has been shown in Table 4.2 and detailed explanation is given below.

Table 4.2 A representation of the package format and possible system capacity

PACKET STRUCTURE	PACKET BIT ORDER	PACKET BIT SIZE	MAXIMUM PER PARENT	MAXIMUM PER SYSTEM	DEFAULT MESSAGE DIRECTION
VON PARITY	0	1	2^7 (128)	2^7 (128)	FROM VON TO SERVER
VON ID	1…7	7			
VTN PARITY	8	1	2^7 (128)	2^{14} (16 384)	FROM VTN TO VON
VTN ID	9…15	7			
BOARD PARITY	16	1	2^{11} (2 048)	2^{25} (33 554 432)	FROM BOARD TO VTN
BOARD ID	17…27	11			
SENSOR ID	28…31	4	2^4 (16)	2^{29} (536 870 912)	FROM SENSOR TO BOARD
SENSOR DATA	32…39	8			

In Table 4.2, “PACKET STRUCTURE” column represents the data related to the elements of the system and their hierarchy within the system, whereas Sensors are the lowest in the hierarchy and VON devices are the highest. The “SIZE” column represents the number of bits allocated to that specific packet element.

The “MAXIMUM PER PARENT” column represents the theoretical maximum number of lower hierarchical elements (e.g. Board), a higher hierarchical element (e.g. VTN) of the system that can contain, both as power of two and corresponding decimal value.

The “MAXIMUM PER SYSTEM” column represents the theoretical maximum number of elements that this system can contain by using this package format, both as power of two and corresponding decimal value.

The “MESSAGE DIRECTION” column shows the hierarchy of data transfer from the lowest level elements (e.g. Sensor) to the highest level elements (e.g. VON). At each level of hierarchy, the corresponding level of element ID and parity bit is added to the front of the message.

It has been shown in Table 4.2 that this message format is 40 bits long in total. The first element of this format is “SENSOR ID” and data. Sensor ID represents the identification number of individual sensors that are connected to a single board and these sensors can send a number between 0 and 256 as data that has been read from the sensor itself. Next in the hierarchy, are “BOARD ID” and parity bit. It identifies the specific Board that this data belongs to and parity bit is used for very simple Odd/Even Parity Bit data verification. This structure is repeated and used the same way for all the remaining elements of this system, up to the VON.

4.4 Design Diagrams

The design diagrams of this project has been prepared to comply and use UML diagrams (“UML Tutorial,” n.d.). They are given at below sections.

4.4.1 Use Case Diagrams

Use case diagram shows the interaction between the user and the system in the simplest way possible. It is a methodology used in system analysis to identify, clarify, and organize system requirements. Use case diagrams are used to collect the requirements of a system including internal and external impacts. These requirements are usually design requirements. Figure 4.4 shows interaction between the user and the system while Figure 4.5 shows the interaction between elements (e.g. VTN, VON, etc.) of the system.

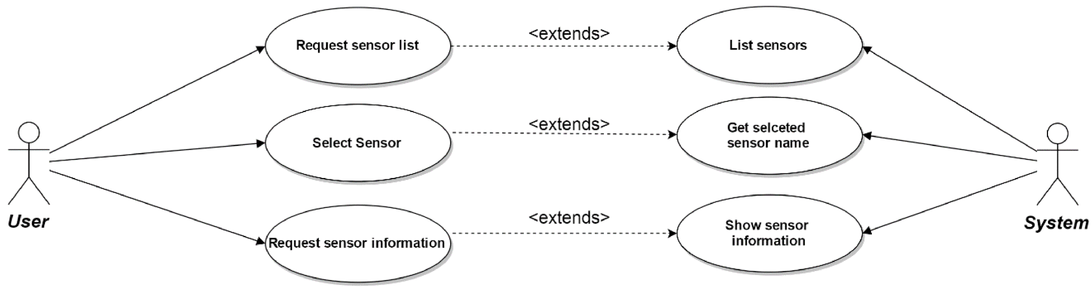


Figure 4.4 Use case diagram of user-system interaction

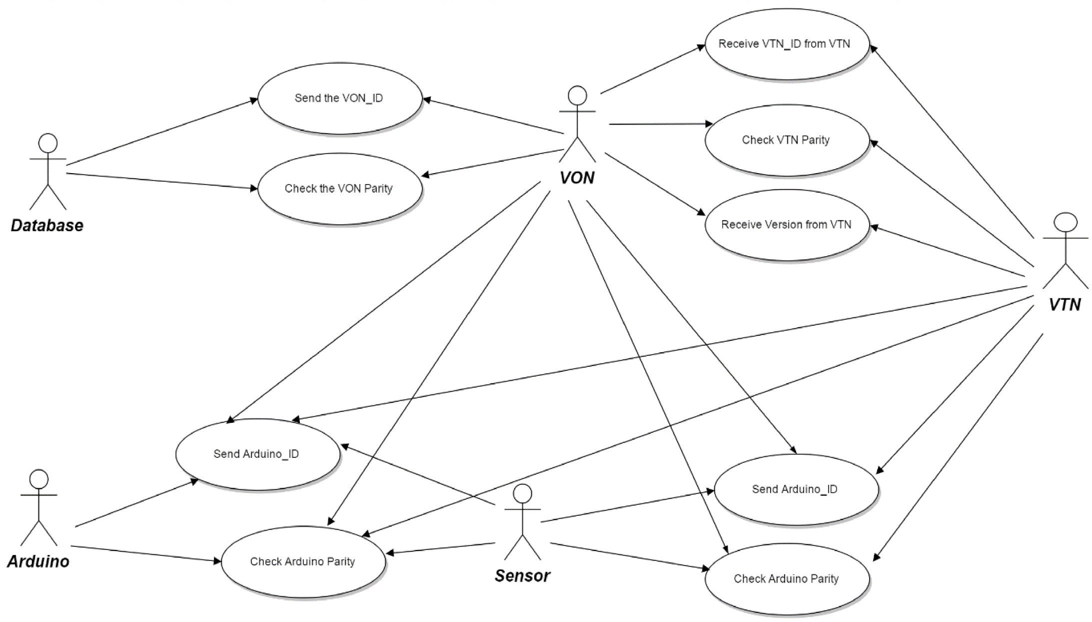


Figure 4.5 Use case diagram of system elements' interaction with each other

4.4.2 Sequence Diagram

A sequence diagram shows how the elements of system interact with each other in an orderly and sequential manner. Sequence diagrams are useful for visualizing different, often time critical, runtime scenarios. These scenarios can help developers to predict how a system will behave and help them discover timing related bugs in system design. After designing possible behaviors of the system, we can easily discover the responsibilities, actions and reactions a class may required to have in the process of creating a new system.

The basic sequence of interactions between the elements of this project is given in Figure 4.6 at below.

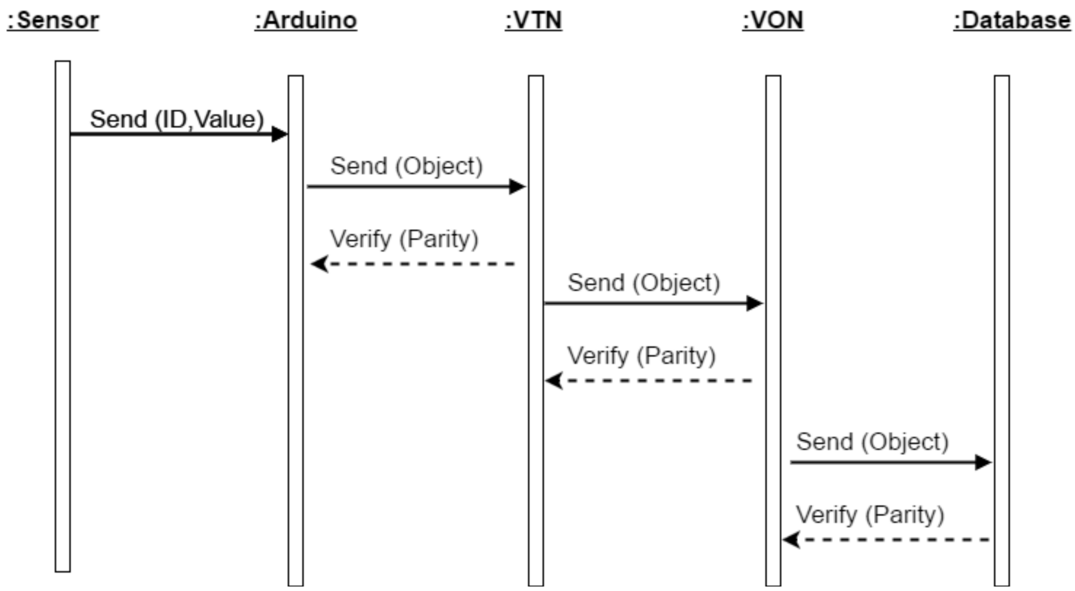


Figure 4.6 The basic sequence of interactions between the elements of this project

4.4.3 Activity Diagram

An activity diagram is a graphical illustration of a system work flow between its elements. The difference between sequence diagrams and activity diagrams can be summarized as sequence diagrams show the communication and messages between different devices in a system whereas activity diagrams show the behavior and execution steps of system operations with system elements roles and participation.

The control flow of activity diagram is drawn from one operation to another. This flow can also be sequential. The basic work flow and work load of elements of this project is very straight forward and it is given in Figure 4.7 below.

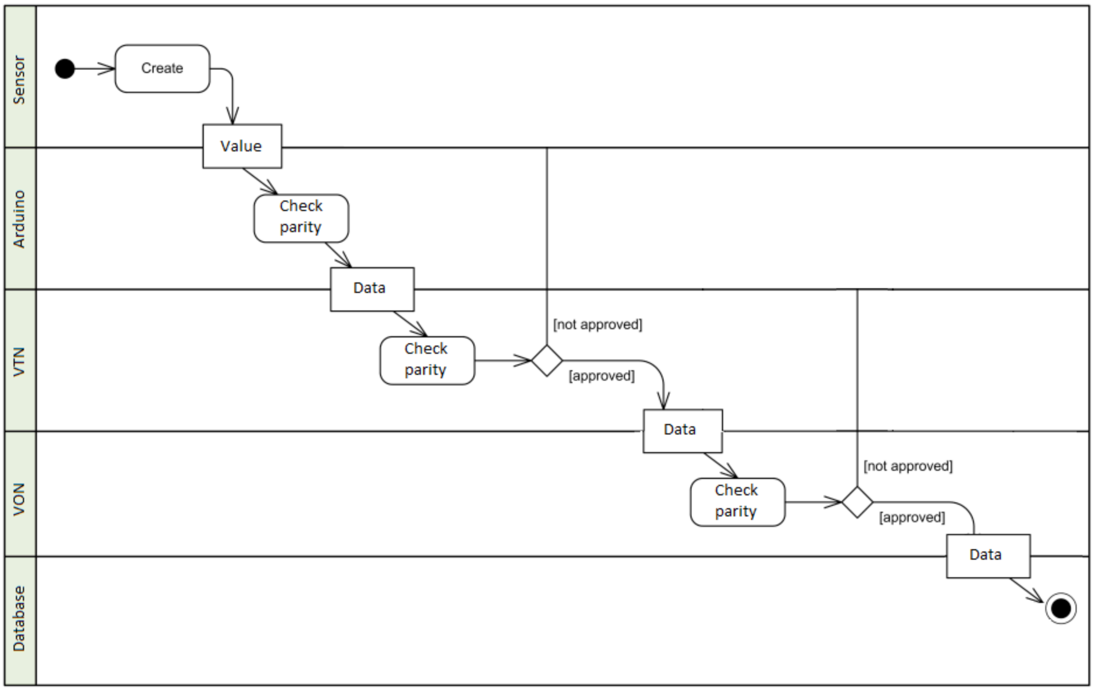


Figure 4.7 The basic work flow and work load of elements of this system

4.4.4 Class Diagram

Class diagram models the static structure of a system. It shows the classes, its instances, functions, and relationships between the objects in a system. In Figure 4.3 class diagram of this project can be examined. All of the connections between classes are done through using foreign keys.

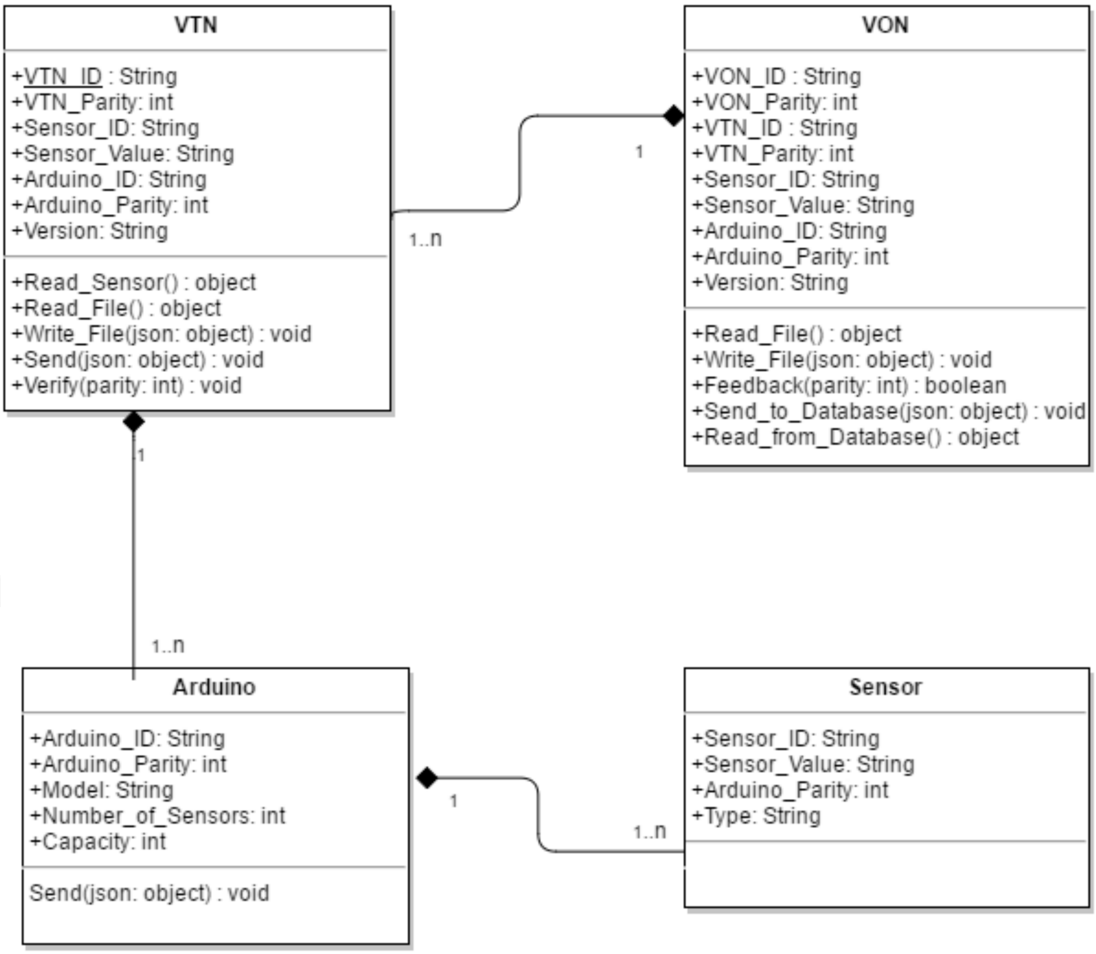


Figure 4.8 The basic class diagram of the elements of this system

4.4.5 Component Diagram

Component diagram shows the relationships and interactions between components of a system. The component diagram of this project is shown in Figure 4.9 and the related information and explanation is given below.

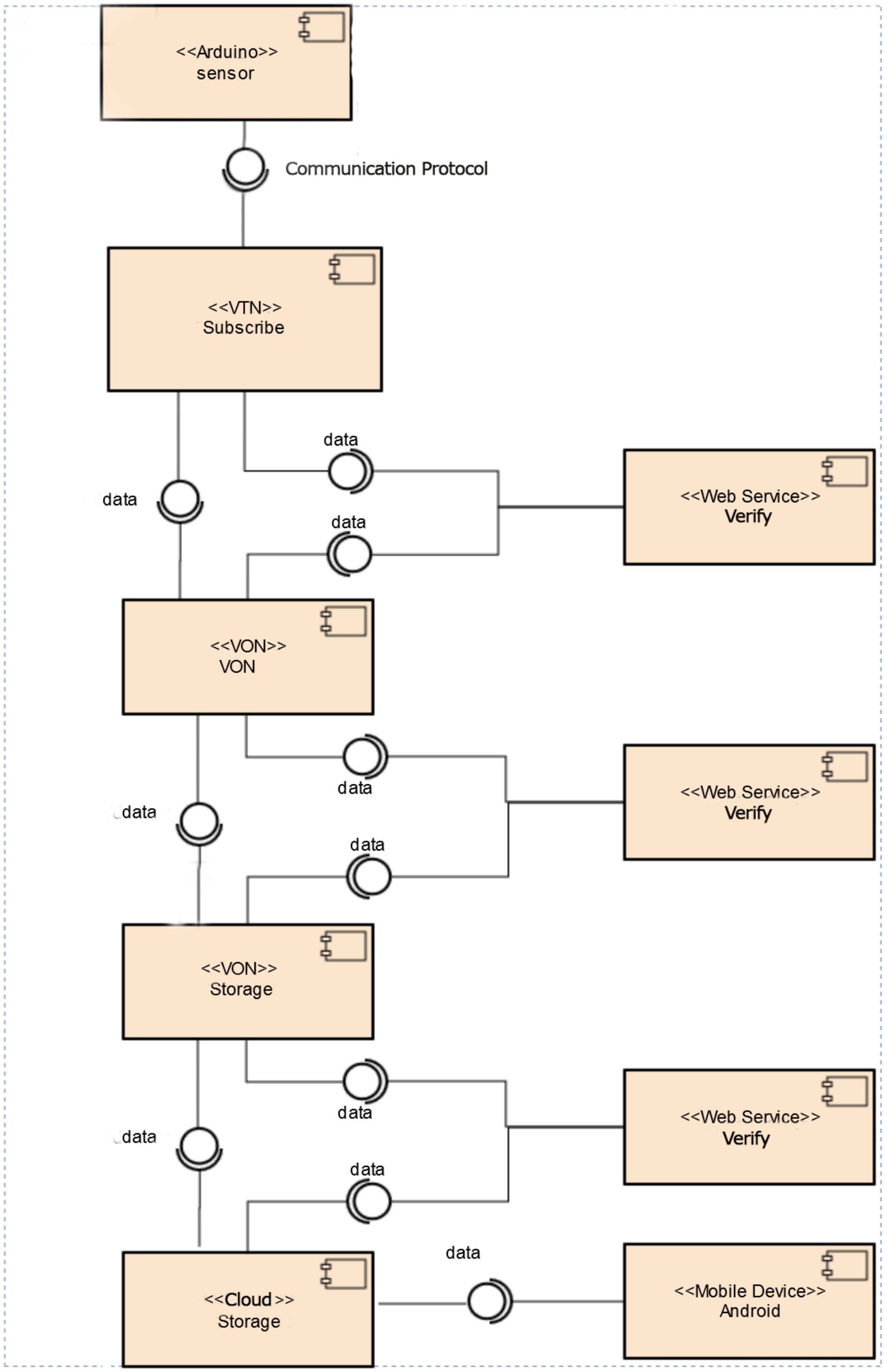


Figure 4.9 The component diagram of this project

At first, data is retrieved from sensors. This data is then sent to the VTN by Board (in this case, Arduino) via communication protocol. Communication protocol provides a secure communication between components. VTN sends the received data to Web Service for verification. Web Service controls the parity and if there are no detected problems, it passes data to VON and informs Board (i.e. Arduino) about the successful transfer.

If there’s a problem with the data, it requests the data to be send once more from Board (i.e. Arduino). Also, if there’s a problem with data receival, communication protocol will be able to save the values and later, it will be able to send them for the next time. Same operating procedure is repeated for VON and other devices. If the verification is successful, VON will store the received data. When the number of received and saved data reached a certain amount (decided by the system operator), VON will send the collected data to a Server. Verification process will also be repeated for this step in execution.

Meanwhile, Mobile monitoring device can request the previously collected data from a server of the system. Mobile device is used only for monitoring operations, it does not have authority to update or delete collected data by the system.

4.4.6 Architectural View

Architectural views are representations of the general architecture of the system. The views are meant for describe the system from the stakeholder’s (or possibly customer’s) point of view, meaning its goal is to show and represent the system to ab audience who does not have technical knowledge related to the field in question. It is especially useful if the system in question is extremely complex to be able to shown to a non-technical crowd.

In this project, there are three layers: presentation tier, middle tier, and data tier. Presentation tier is the one that a user (or a system maintainer) interacts with the system. In this project, a user (or a system maintainer) interacts via mobile application for monitoring purposes only. Middle tier acts as a bridge between data tier and presentation layer of the system. Data tier is where a server of the system stores all the data, created by the system. In Figure 4.10, relationships between aforementioned tiers of this system is shown.

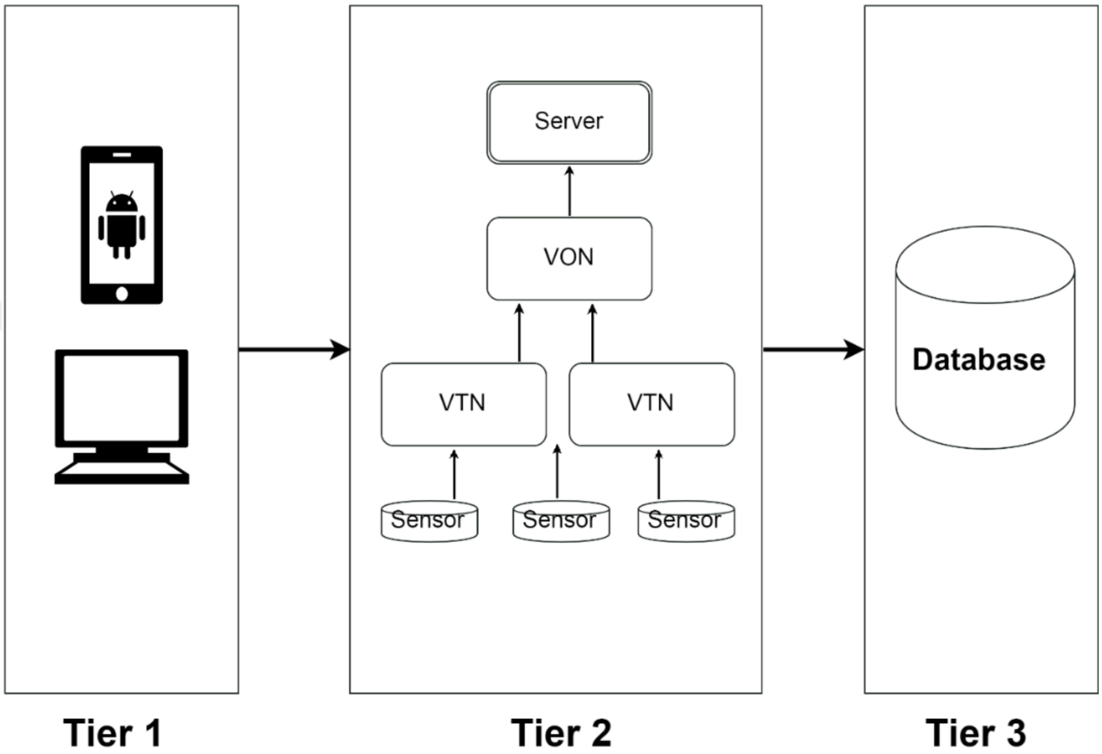


Figure 4.10 The architectural view of this project

4.4.7 Deployment Diagram

A deployment diagram is used to visualize the system components as they would appear after a real world implementation and installation. They are used to describe what would the final product would be like after finished.

Deployment diagrams use a group of UML diagram element shapes. The three-dimensional boxes, known as nodes, represent the basic software or hardware elements, or undefined nodes, in the system. Lines drawn from node to node indicate relationships, and the smaller shapes contained within the boxes (e.g. nodes) represent the software artifacts that are deployed or installed inside that node.

Deployment diagrams allow developers or customers to see which software elements are deployed on which hardware elements, visualize the runtime processing of hardware elements in relation with each other and the system's real world topology of hardware and software. The basic deployment diagram of this system cab be seen in Figure 4.11.

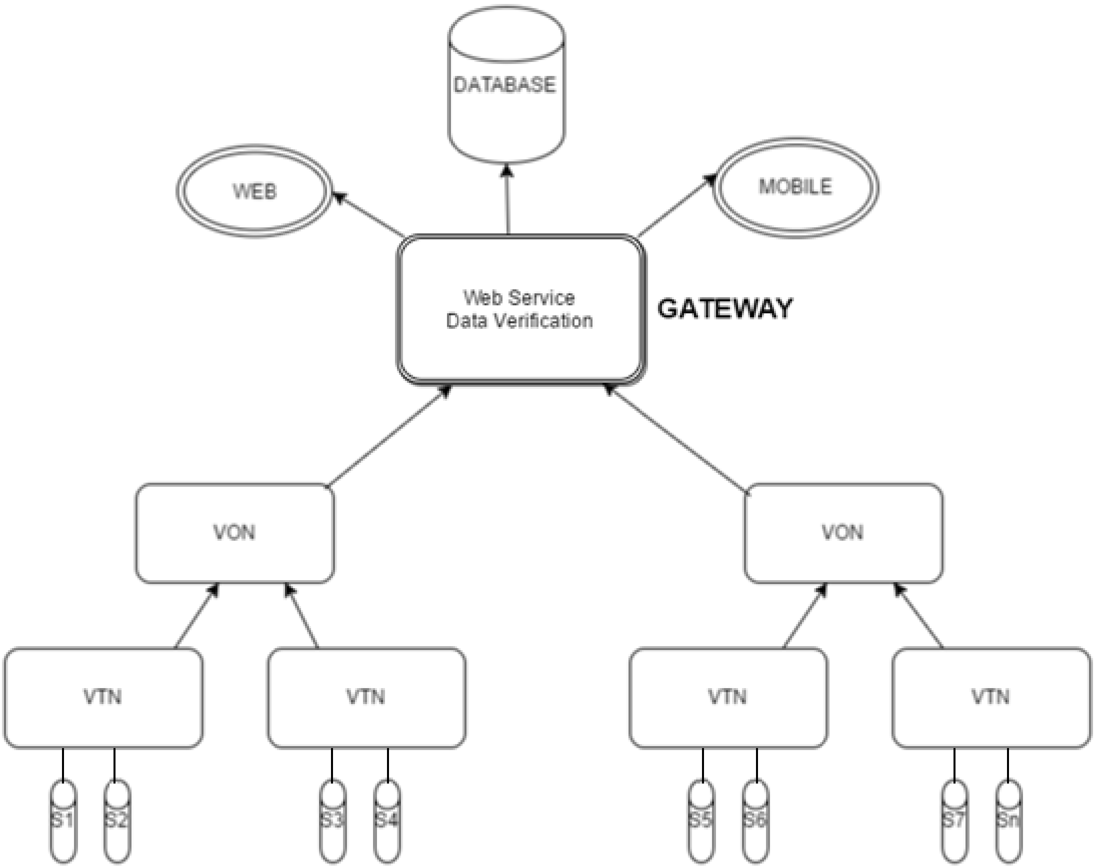


Figure 4.11 The basic deployment diagram of this project

With proof of concept implementations to test implementability of a simple version of the system, overall system design, modeling and representation, standard designed message format for communication within the system and UML design diagrams to represent the system from different perspectives, structures and operations are completed, real life testing, theoretical system simulations and comprehensive prototype implementations will be given and explained in the next chapter.

CHAPTER FIVE
TESTING AND EXPERIMENTATION

5.1 Implemented Message Format and Structure

As it was given and explained in previous chapter, a message format has been designed for this IoT system. To implement this IoT system, message format will also have to be developed more to work on this system. This extended and used message format is given in Figure 5.1 below.

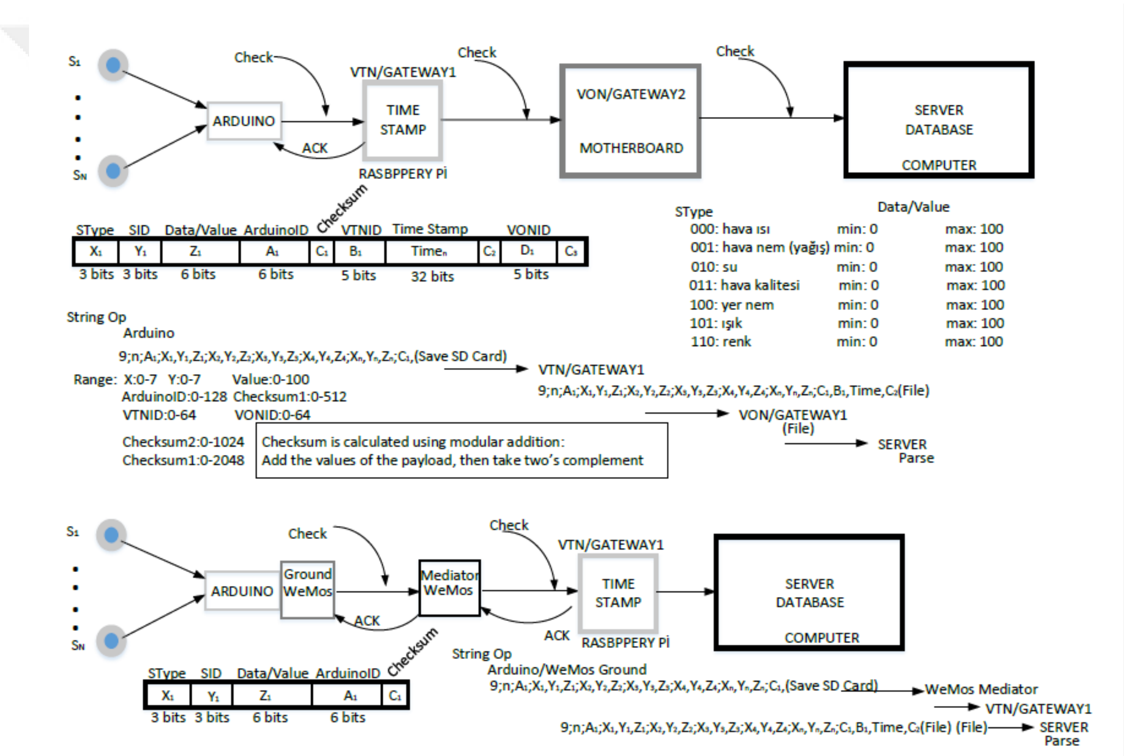


Figure 5.1 The implemented and improved message format for IoT system

This new format contains some new fields that were not present on the previous chapter. The first one is "Time Stamp" field that holds a 32 bit value to represent the message creation time in UNIX Time Format (Thompson & Ritchie, 1971). This enables us to precisely learn and store the read sensor value's original detection time, instead of message arrival time. The second one is "Sensor Type" field that contains 3 bit value to identify the kind of sensor that is being referred (heat, humidity, light, etc.). The reason this field is used instead of "Sensor ID" field, is to allow developers to use more than one of same type of sensor for error correction and to eliminate further limitations that could arise in the future implementations.

Table 5.1 Comparison of designed and implemented message format

<table>
<tr><th rowspan="2">PACKET SECTION NAME AND ORDER</th><th colspan="4">DESIGNED MESSAGE FORMAT (CHAPTER 4)</th><th colspan="4">IMPLEMENTED MESSAGE FORMAT (CHAPTER 5)</th></tr>
<tr><th>PACKET BIT ORDER</th><th>PACKET BIT SIZE</th><th>MAXIMUM PER PARENT</th><th>MAXIMUM PER SYSTEM</th><th>PACKET BIT ORDER</th><th>PACKET BIT SIZE</th><th>MAXIMUM PER PARENT</th><th>MAXIMUM PER SYSTEM</th></tr>
<tr><td>VON PARITY</td><td>0</td><td>1</td><td rowspan="2">2^{7} (128)</td><td rowspan="2">2^{7} (128)</td><td>0</td><td>1</td><td rowspan="2">2^{5} (32)</td><td rowspan="2">2^{5} (32)</td></tr>
<tr><td>VON ID</td><td>1…7</td><td>7</td><td>1…5</td><td>5</td></tr>
<tr><td>VTN PARITY</td><td>8</td><td>1</td><td rowspan="2">2^{7} (128)</td><td rowspan="2">2^{14} (16 384)</td><td>6</td><td>1</td><td rowspan="2">2^{5} (32)</td><td rowspan="2">2^{10} (1024)</td></tr>
<tr><td>VTN ID</td><td>9…15</td><td>7</td><td>39…43</td><td>5</td></tr>
<tr><td>BOARD PARITY</td><td>16</td><td>1</td><td rowspan="2">2^{11} (2 048)</td><td rowspan="2">2^{25} (33 554 432)</td><td>44</td><td>1</td><td rowspan="2">2^{6} (64)</td><td rowspan="2">2^{16} (65 536)</td></tr>
<tr><td>BOARD ID</td><td>17…27</td><td>11</td><td>45…50</td><td>6</td></tr>
<tr><td>SENSOR ID</td><td>28…31</td><td>4</td><td rowspan="2">2^{4} (16)</td><td rowspan="2">2^{29} (536 870 912)</td><td>57…59</td><td>3</td><td rowspan="2">2^{3} (8)</td><td rowspan="2">2^{19} (524 288)</td></tr>
<tr><td>SENSOR DATA</td><td>32…39</td><td>8</td><td>51…56</td><td>6</td></tr>
<tr><td>SENSOR TYPE</td><td>N/A</td><td>N/A</td><td>N/A</td><td>N/A</td><td>60…62</td><td>3</td><td>N/A</td><td>N/A</td></tr>
<tr><td>TIME STAMP</td><td>N/A</td><td>N/A</td><td>N/A</td><td>N/A</td><td>7…38</td><td>32</td><td>N/A</td><td>N/A</td></tr>
</table>

As you can see from Table 5.1, the ordering of message format sections is different, however, this is a trivial difference and as requirements and goals change, it can be redesigned. As it is a small scale implementation, overall device capacity of implemented message format is lower however, as it was stated before, this can be changed in the future. This also can be seen by looking at the message format sizes. Designed message format contain 40 bits of data while Implemented message format contains 62 bits of data. However, if we remove the time stamp size from second message format, it is only 30 bits of data, less than the first message format and explaining lower carrying capacity of the system.

5.2 IoT System and Gateway Implementation and Operations

Similar to the proof of concept prototype in the previous section, we implement MQTT communication protocol with WEMOS, a Wi-Fi communication board that contains a ESP8266 microchip for Ethernet and TCP/IP communication (ESP8266, 2014). As it was done in the previous section, MQTT Dashboard is used and its working mechanism has been explained in the previous chapters. The one of the main reason for using WEMOS D1 Mini and ESP8266 microchip is because it is considered to be reliable and cheap compared to alternatives in the market.

In this implementation, humidity and temperature sensors are used and connected to WEMOS Arduino board. Arduino IDE has been used to write the required source code to retrieve sensor data. The retrieved data from sensors has been published on MQTT Dashboard by using a mobile phone as a Wi-Fi Hotspot gateway ("Hotspot," n.d.). Related source code is given at Figure 5.2 and related MQTT Dashboard information is given at Figure 5.3.

```
const char* ssid = "AndroidAP";
const char* password = "12345678";
const char* mqtt_server = "broker.hivemq.com";
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char  msg[100];
```

Figure 5.2 Arduino IDE source code for Hotspot and MQTT Dashboard connection

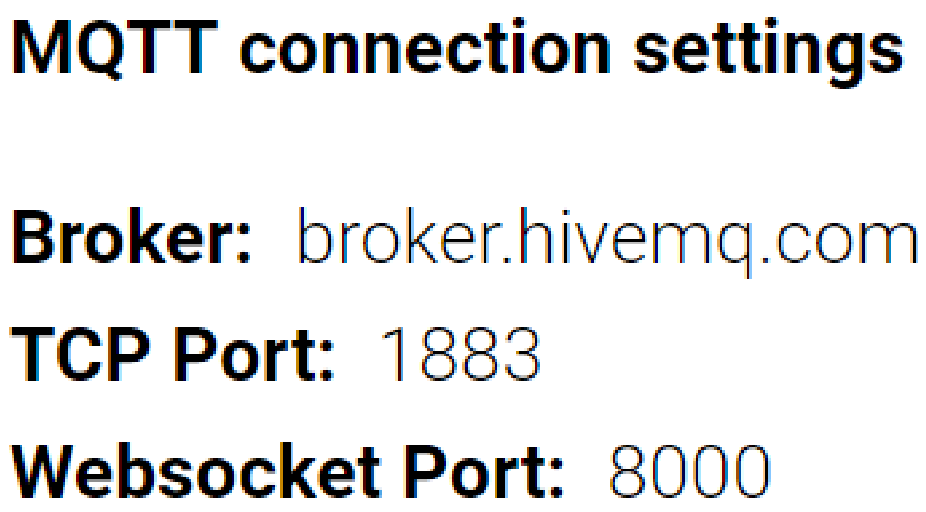


Figure 5.3 MQTT Dashboard connection configuration

The connection between WEMOS and MQTT Dashboard (MQTT broker in this implementation) must be continuous and if a disconnection occurs, WEMOS must try to reconnect until a connection is established again. The source code of this process and related operations is shown in Figure 5.4 below.

```
void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    String clientId = "ESP8266Client-";
    clientId += String(sensor(0xffff), HEX);
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      client.publish("iotDeu", "HelloWorld");}
    else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

Figure 5.4 Arduino IDE source code for reconnection to MQTT Dashboard. Upon a successful connection, WEMOS publishes “HelloWorld” string to “iotDeu” topic

Meanwhile, VTN is also connected to MQTT Dashboard and subscribed to the same topic that WEMOS is publishing data to. VTN will receive any published data by WEMOS and it will create a text file after five messages has been received. All text files contain only five messages from WEMOS and name of text files with be time stamped with that file’s creation time. After such a text file is created and five messages has been written on it, it will be send to VON. If it is successful, the original file will be deleted and if not, it will continue to try to send the file to VON until successful.

If there is a connection problem or lost between VTN and MQTT Dashboard, previously received five messages and the created file will be stored until connection is established again. After that, previously explained standard operation will resume.

Related and partial Python programming language code of these operations are given in Figure 5.5 and Figure 5.6 below.

```
subscribe.py ×   VON.py ×

    temp_list.append(von)
    global count
    count +=1
    print "count:", str(count)
if count %5==0:

    aa = open('/home/deneme/Belgeler/output.txt', "a")
    with open('/home/deneme/Belgeler/backup.txt', "r+") as fo:
        for line in fo:
            aa.write(line)
    fo.close()
    aa.close()
    Post()
```

Figure 5.5 Python code for writing received five messages and writing it to a text file. On contrary to previous statement, text file names are fixed and not time stamped because they are taken before this developmental change

```
def Post():
    serviceurl = 'http://localhost/webservice.php'
    url = serviceurl
    print 'Retrieving', url
    uh = urllib.urlopen(url)
    data = uh.read()
    try:
        js = json.loads(str(data))
    except:
        js = None

        json.dumps(js, indent=1)
```

Figure 5.6 Python code for sending received messages to VON. The reason “localhost” was used as delivery address is because, during the time of taking this screenshot, VON was located in the same computer as VTN

After successful transmission, VON will send to VTN device a “YES” string via “webservice1.php” as an acknowledgement of successful transfer of the text file from VTN to VON. The screenshots of this process are shown in Figure 5.7 and Figure 5.8 below.

Figure 5.7 Console output for sending received messages to VON. Retrieved messages are send to VON via “webservice.php” and the reply “YES” is received via “webservice1.php” localhost services

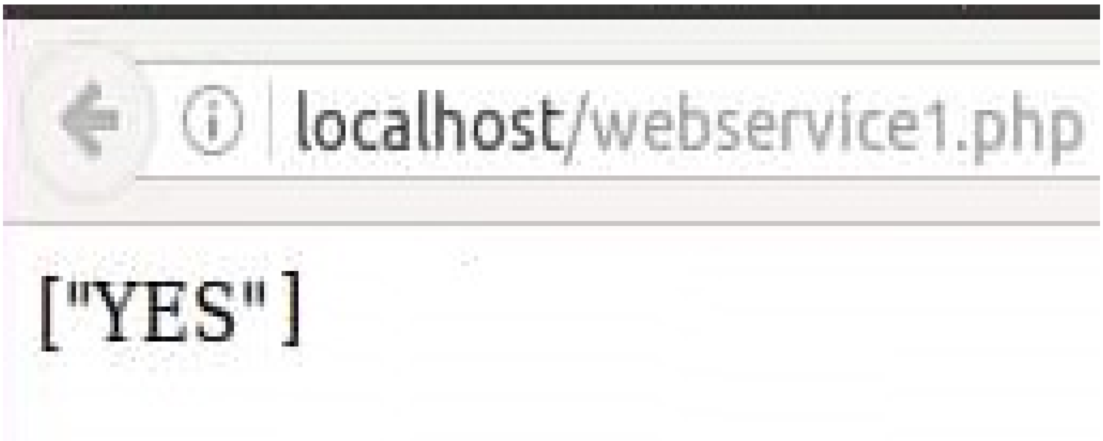


Figure 5.8 The screenshot of “webservice1.php” where the result “YES” from VON is displayed in “localhost” and on a web browser

After VON receives the messages from VTN, it appends “VON_ID” and “VON_Parity” fields to the beginning of received messages. After this operation, modified messages are written to a text file in the same manner as VTN. Unlike VTN, VON waits for 15 messages to be received in total before trying to transmit them to a SERVER. These numbers are arbitrary and can be changed in the future to increase performance or etc.

As it was the case with VTN, if the connection is lost during a transmission, latest received data is stored and VON tries to connect with VTN, until it is established again. After successful reconnection, the process continues as before. After 15 records has been written to the text file, the messages in this file are send to SERVER which contains the database for message insertion and storage. The same connection lost operations and procedures that had given before also applies to VON and SERVER connection. Database connection and data insertion code that is executed on SERVER computer is given in Figure 5.9 below.

```python
def Database():
    db = MySQLdb.connect(host='127.0.0.1', user='root', passwd='', db='iot', port=80)
    cur = db.cursor()
    with open('/home/deneme/Belgeler/database.txt', "r") as ins:
        array = []
        for line in ins:
            array.append(line)
    for item in array:
        arr = item.split((' '))
        sql = "INSERT INTO `iot_project`(`ID`, `Version`, `VON_Parity`, `VON_ID`, " \
              "`VTN_Parity`, `VTN_ID`, `Arduino_Parity`, `Arduino_ID`, `Sensor_ID`, " \
              "`Value`) VALUES (null,'%s', '%s', '%s', '%s','%s', '%s', '%s', '%s', '%s')"\
              % (arr[0], arr[1], arr[2], arr[3], arr[4], arr[5], arr[6], arr[7], arr[8])
        cursor = db.cursor()
        try:
            cursor.execute(sql)
            print "success"
            db.commit()
        except:
            db.rollback()
            print "failed"
```

Figure 5.9 Python code for connecting to and inserting records into database that is working on a SERVER computer in this implemented IoT system.

In addition to above connections and operations, it is also important to establish communication between VTN, VON and MQTT Dashboard, just like it was established for Arduino, as it was shown before. Unlike Arduino, this was done with a Python programming code, instead of C programming code and because Python will be used for both VTN, VON and even for SERVER computers, they will be very similar in structure and operation. An example of such Python code for MQTT Dashboard connection is given in Figure 5.10 below.

```python
msg=list(stringOne)
broker_address="broker.hivemq.com"
client1 = mqtt.Client("P1")    #create new instance
client1.on_connect= on_connect        #attach function to callback
client1.on_message=on_message        #attach function to callback
time.sleep(1)
client1.connect(broker_address)      #connect to broker
client1.loop_start()    #start the loop
client1.subscribe("IoTproject")
client1.publish("IoTproject",str(msg))
time.sleep(1)
client1.disconnect()
client1.loop_stop()
```

Figure 5.10 Python code for connecting to MQTT Dashboard by VTN, VON and SERVER computer to communicate with each other

Couple of problems had been encountered while using Raspberry Pi 3 for VTN or VON purposes. First of these was the lack of internal space for OS, required programs and related data. This problem has been with solved by using a micro SD (“Micro SD,” n.d.) memory card as an alternative hard disk for aforementioned requirements.

Second problem was related to successfully establishing connection with eduroam (eduroam, 2002) Internet and networking service that is used in many universities around the world. Eduroam configuration did not allow MQTT and other protocol connections between the devices and Internet services (e.g. MQTT Dashboard) that was used in this system. The only solution to this problem was using a non eduroam Internet service as it was not possible to change eduroam security configuration for this project.

Third problem was related to the requirement of input and output resources and general usability of Raspberry Pi 3 during system operations, such as a monitor, a keyboard and a mouse. In order to implement a remote management and reachability solution, PuTTY (PuTTY, 1999), a remote connection and terminal emulator tool for many different use case scenarios, has been used. An example screen of PuTTY program is given in Figure 5.11 below.

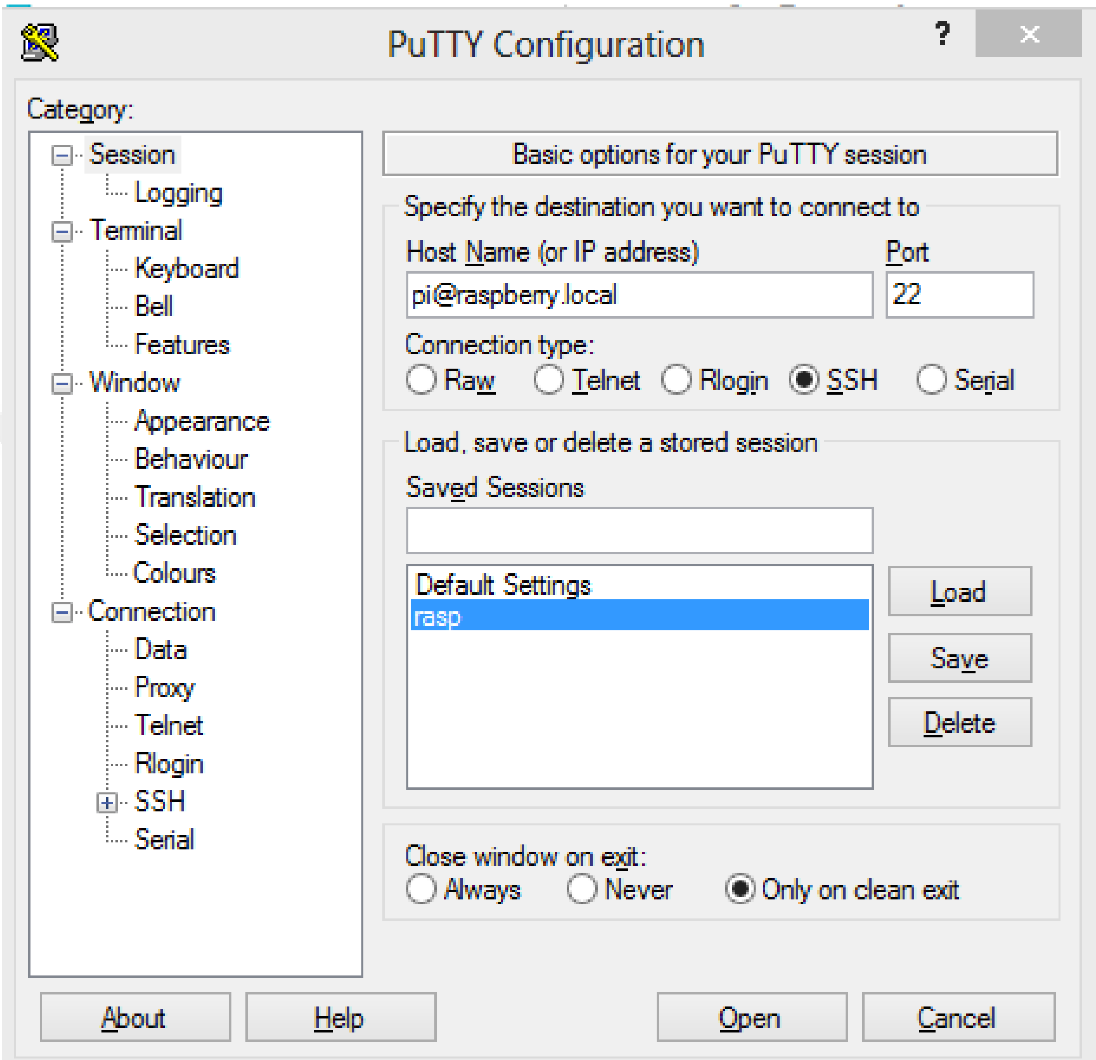


Figure 5.11 Screenshot of PuTTY computer program that is used for remote connection and management in this project via using Secure Shell (SHH) functionality

After this connection is established, Atom text editor (Atom, 2014) has been used write the required programming codes of this project and executing the previously written codes on the devices of this system via PuTTY program connection. Configuration code for this operation is given in Figure 5.12 below.

```json
{
  "uploadOnSave": true,
  "useAtomicWrites": false,
  "deleteLocal": false,
  "hostname": "raspberrypi.local",
  "port": "22",
  "target": "~/project/",
  "ignore": [
    ".remote-sync.json",
    ".git/**"
  ],
  "username": "pi",
  "password": "raspberry",
  "watch": [],
  "transport": "scp"
}
```

Figure 5.12 Atom text editor configuration for connecting Raspberry Pi 3 devices via PuTTY program's SHH connection

The next operation is to implement a SERVER machine in Windows Server 2012 OS with MQTT broker software Mosquitto, to prevent the dependency on MQTT Dashboard web service, for security and reliability improvements of the system. This SERVER device will also be connected by Android application to monitor and review the overall system remotely. In this scenario, Raspberry Pi 3 was the publisher of data while the Windows Server 2012 machine was both the MQTT broker service of the system and the receiver of previously published data. Coding wise, there was no need for significant changes for both devices, just the MQTT broker connection configuration needed to be changed. The Python code for this operation is given Figure 5.13 below and the console output of the execution of this code is given in Figure 5.14 below.

C:\Users\Administrator\Desktop\sub.py - Sublime Text (UNREGISTERED)

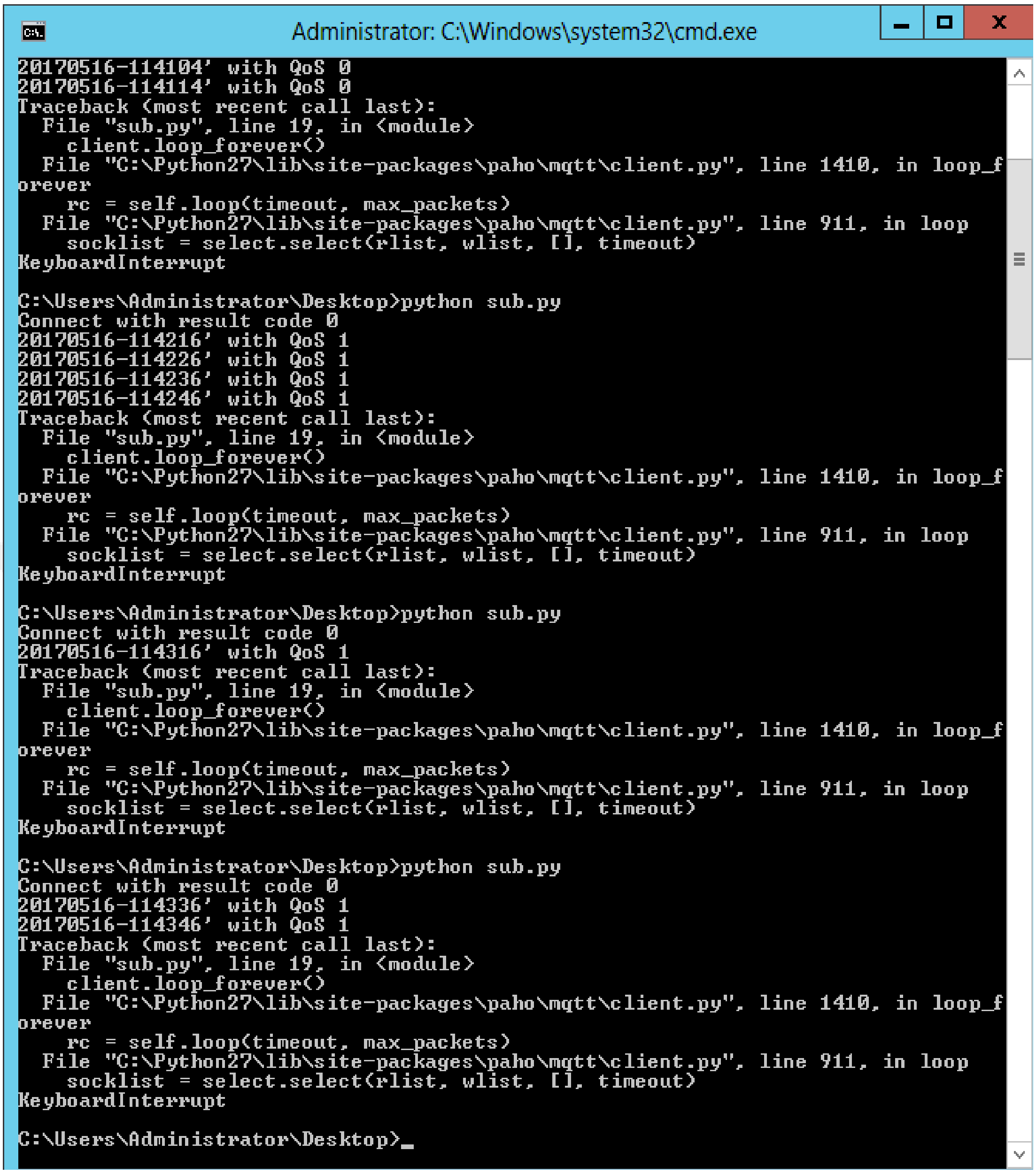
```python
import paho.mqtt.client as mqtt
import time

def on_connect(client, userdata, rc):
    print("Connect with result code " + str(rc))
    client.subscribe("IOTDEU", qos=1)

def on_message(client, userdata, msg):
    timestr = time.strftime("%Y%m%d-%H%M%S")
    print(timestr + "' with QoS " + str(msg.qos))
    f = open('C:/Users/Administrator/Desktop/MQTT/'+timestr+'.txt', 'w')
    f.write(msg.payload)
    f.close()

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect("broker.hivemq.com", 1883, 60)
client.loop_forever()
```

Figure 5.13 Python code of subscribe and data receival operation executed in SERVER machine of the system. Note that connection configuration is for MQTT Dashboard, not local Mosquitto instance that is running on the same machine



```
Administrator: C:\Windows\system32\cmd.exe
20170516-114104' with QoS 0
20170516-114114' with QoS 0
Traceback (most recent call last):
  File "sub.py", line 19, in <module>
    client.loop_forever()
  File "C:\Python27\lib\site-packages\paho\mqtt\client.py", line 1410, in loop_f
orever
    rc = self.loop(timeout, max_packets)
  File "C:\Python27\lib\site-packages\paho\mqtt\client.py", line 911, in loop
    socklist = select.select(rlist, wlist, [], timeout)
KeyboardInterrupt

C:\Users\Administrator\Desktop>python sub.py
Connect with result code 0
20170516-114216' with QoS 1
20170516-114226' with QoS 1
20170516-114236' with QoS 1
20170516-114246' with QoS 1
Traceback (most recent call last):
  File "sub.py", line 19, in <module>
    client.loop_forever()
  File "C:\Python27\lib\site-packages\paho\mqtt\client.py", line 1410, in loop_f
orever
    rc = self.loop(timeout, max_packets)
  File "C:\Python27\lib\site-packages\paho\mqtt\client.py", line 911, in loop
    socklist = select.select(rlist, wlist, [], timeout)
KeyboardInterrupt

C:\Users\Administrator\Desktop>python sub.py
Connect with result code 0
20170516-114316' with QoS 1
Traceback (most recent call last):
  File "sub.py", line 19, in <module>
    client.loop_forever()
  File "C:\Python27\lib\site-packages\paho\mqtt\client.py", line 1410, in loop_f
orever
    rc = self.loop(timeout, max_packets)
  File "C:\Python27\lib\site-packages\paho\mqtt\client.py", line 911, in loop
    socklist = select.select(rlist, wlist, [], timeout)
KeyboardInterrupt

C:\Users\Administrator\Desktop>python sub.py
Connect with result code 0
20170516-114336' with QoS 1
20170516-114346' with QoS 1
Traceback (most recent call last):
  File "sub.py", line 19, in <module>
    client.loop_forever()
  File "C:\Python27\lib\site-packages\paho\mqtt\client.py", line 1410, in loop_f
orever
    rc = self.loop(timeout, max_packets)
  File "C:\Python27\lib\site-packages\paho\mqtt\client.py", line 911, in loop
    socklist = select.select(rlist, wlist, [], timeout)
KeyboardInterrupt

C:\Users\Administrator\Desktop>_
```

Figure 5.14 Console output on SERVER machine after executing subscription Python code for MQTT protocol. Note that the received error of “in loop_forever” is actually correct because this program is designed to work in an infinite loop until termination by the user, a use case that Python interpreter assumes a programming error

After this operation, the implementation of this system is complete and we have successfully operated it for the aforementioned operations. A simple representation of the operations and the implemented structure of this system is given in Figure 5.15 below.

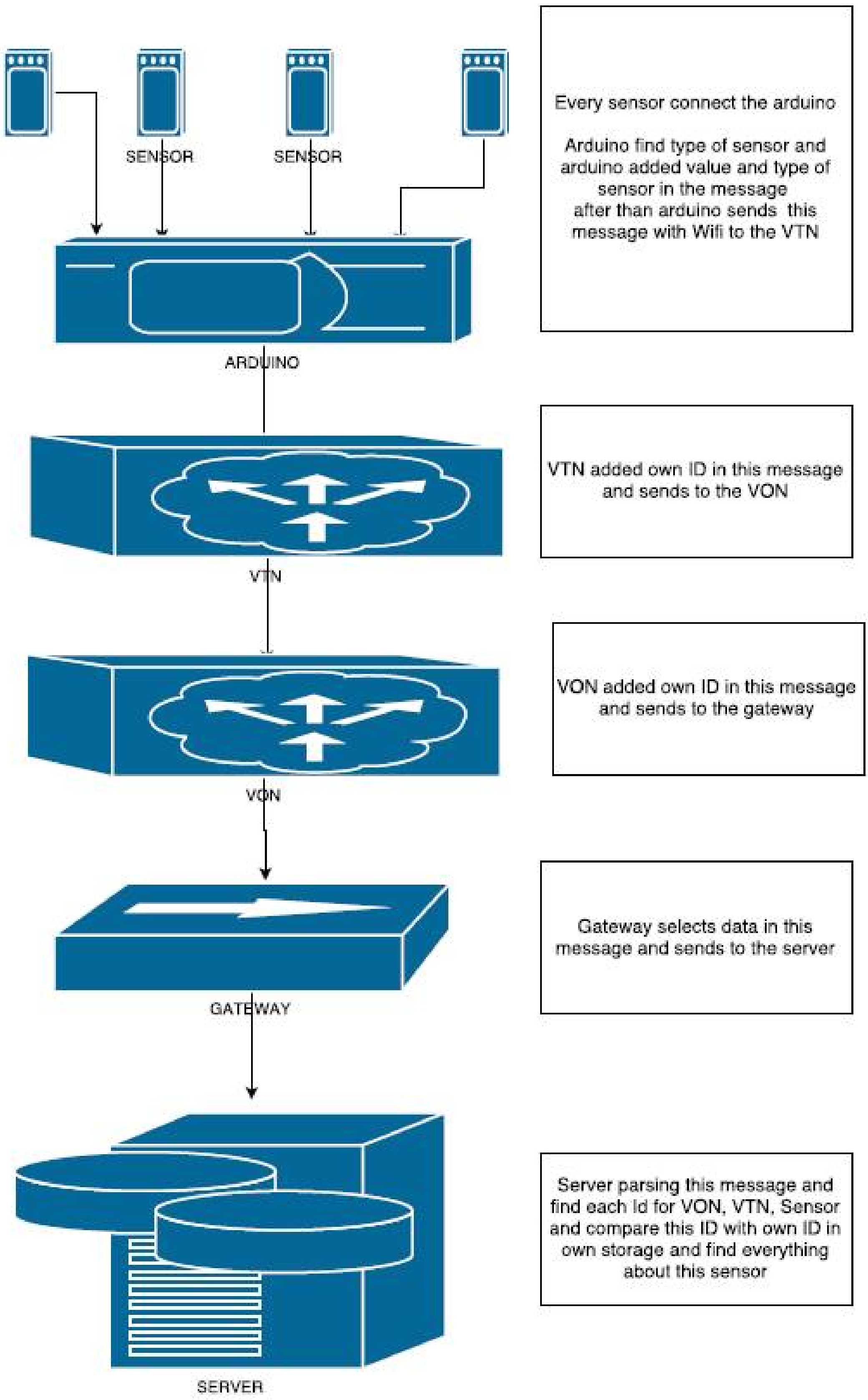


Figure 5.15 The implemented structure and operations of this IoT network and gateway system

5.3 IoT System Simulation in ns-3 Software

The simulation of this system is done in ns-3 (ns-3, 2008), an open source network simulator that can be used to simulate a large collection of devices (e.g. sensors, gateways and servers) to see what would happen if it were to be implemented on a very large scale.

In the created and used model, all devices communicate over TCP/IP protocol and there are three layers, Servers at the top (representing SERVER computers of the network), Gateways at the middle (representing VTN and VON devices of the network) and Sensors at the bottom (representing Arduino devices of the network). A simple representation of this network structure is given in Figure 5.16 below.

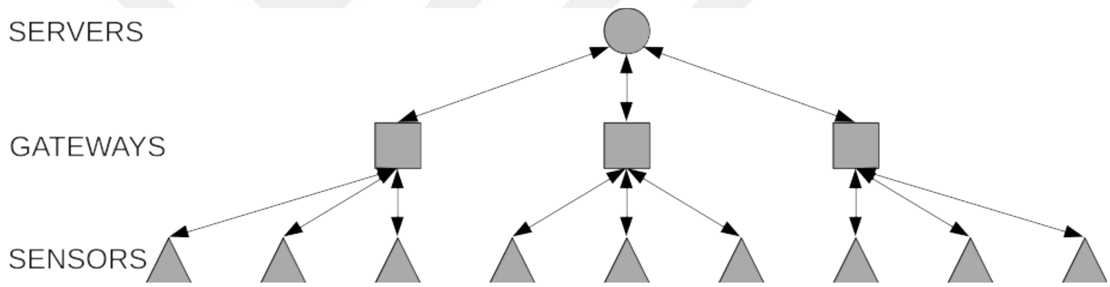


Figure 5.16 Basic representation of simulated network in ns-3 where circles represent the Servers, rectangles represent the Gateways and triangles represent the Sensors

ns-3 is very complex software and for that reason you need to have experience beforehand with it and C++ programming language, the main programming language used in ns-3. Python programming language is also being used, however, there are still incomplete or unusable functionalities due to the differences between the languages and development focus ("ns-3 Python Manual," n.d.). Another reason for using C++ instead of Python is the more availability of tutorials for ns-3 in C++ language, compared to Python.

In Figure 5.17 below, ns-3 source code can be seen inside NetBeans IDE (NetBeans, 2003), which is used to make coding C++ easier than a standard text editor.

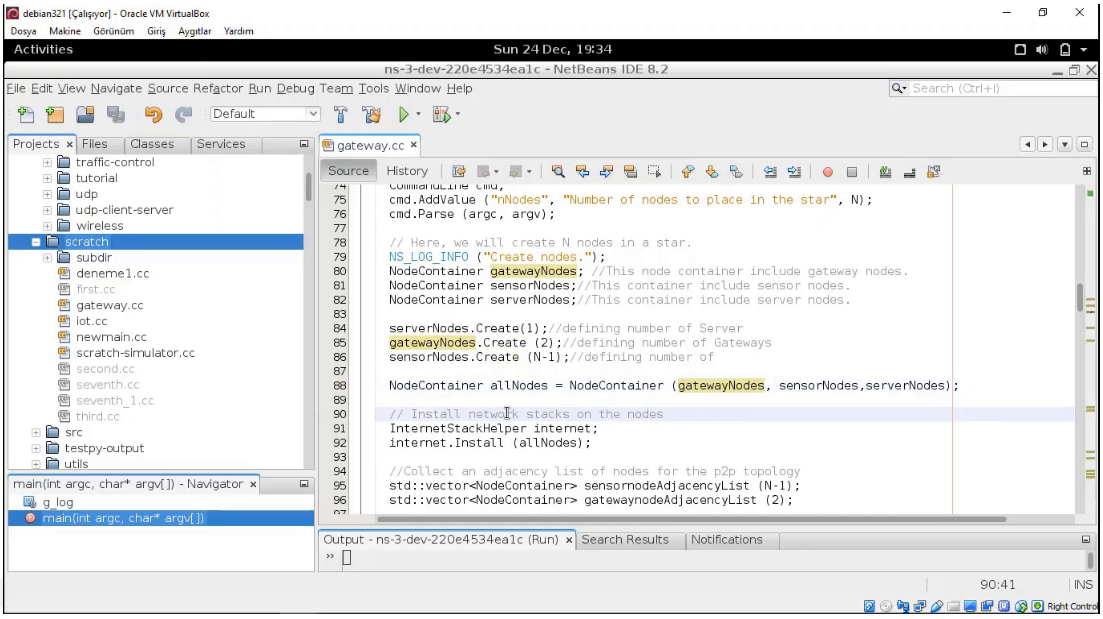


Figure 5.17 C++ source code for ns-3 to initialize network elements according to the given numbers. Even though Server and Gateway numbers are fixed (number of Servers being 1 and Gateways 2), they can be changed to examine different scenarios, whereas in contrast, the number of Sensors are requested from the user at the start of the program

The different steps of resulting animation of the simulation after the execution, created by using NetAnim (NetAnim, n.d.) component of ns-3 software, is given in Figures 5.18, 5.19 and 5.20 below.

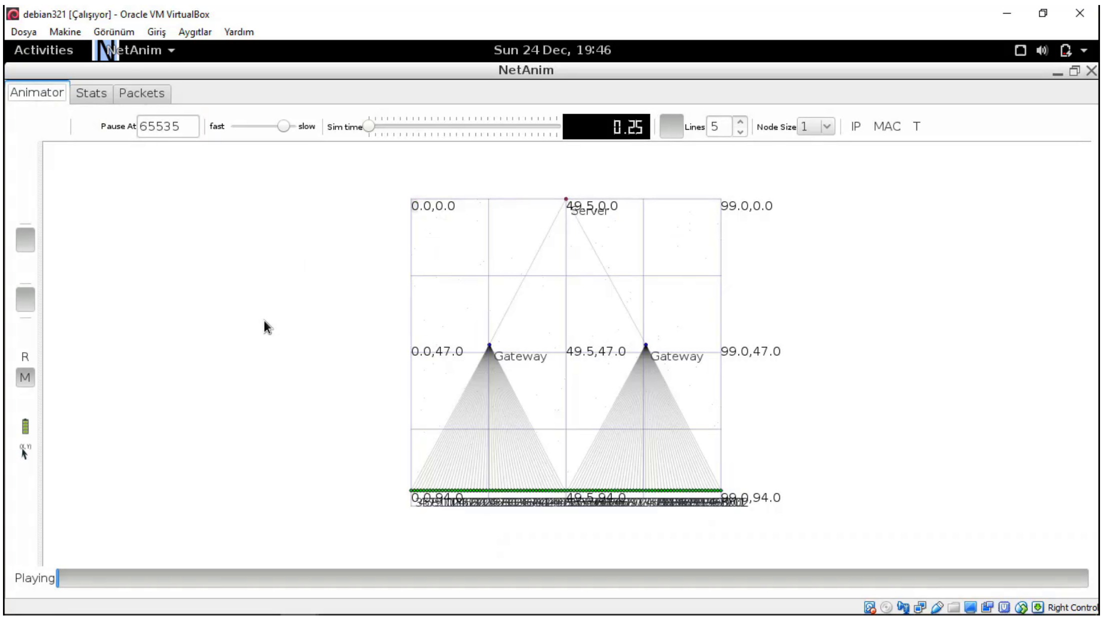


Figure 5.18 Simulation starting and initialization status where no messages have been send yet between the devices of the network

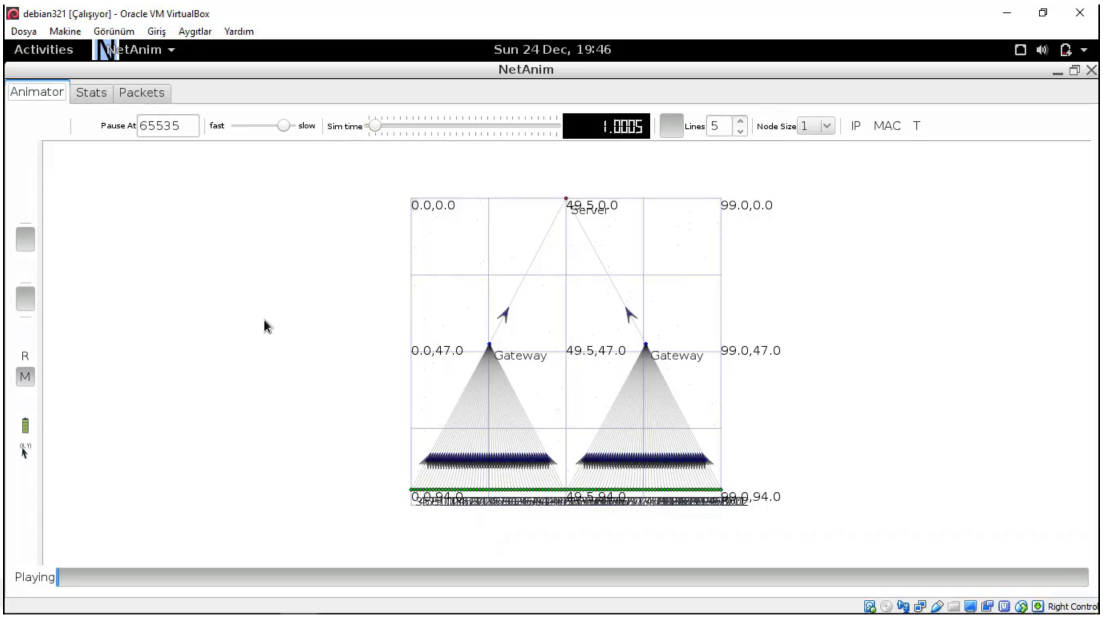


Figure 5.19 The first step of simulation where messages from the lower level elements of network (e.g. Sensors and Gateways) to the higher level elements of network (e.g. Gateways and Servers) are transmitted

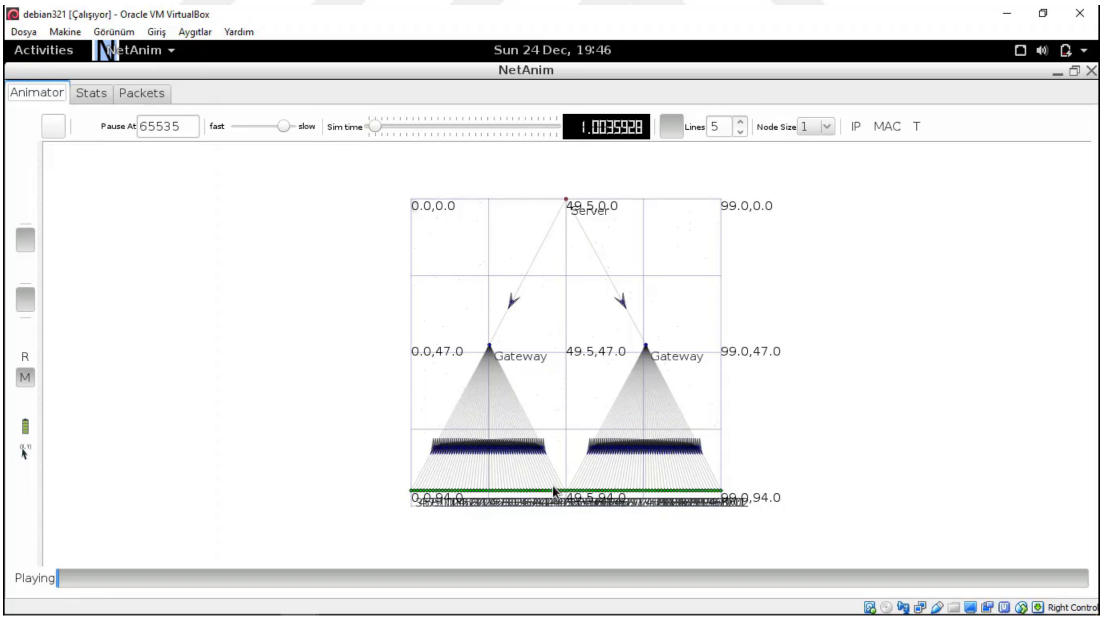


Figure 5.20 The second step of simulation where the replies from previously send messages are transmitted from the higher level elements of network (e.g. Servers and Gateways) to the lower level elements of network (e.g. Gateways and Sensors)

As it can be seen, we have done a simulation with 1 Server, 2 Gateways and 100 Sensors (50 per Gateway) and successfully send packages between the devices of the network, thus completing our objective.

CHAPTER SIX
RESULTS AND CONCLUSIONS

The examination of results and conclusions of this thesis will be done in the related sub sections below in detail.

6.1 Comparison of Goals and Results of the Thesis

The main goal of this thesis, as the title suggests, was to design and develop a IoT Gateway. For this purpose, scientific literature has been studied for implementations and comparisons. Current trends on IoT and related technology has also been examined and used in this thesis.

Compared to alternatives, this thesis is average in its success and does not contain solutions or implementations for most of the challenges that were presented in it.

Therefore, the results can be said to be satisfactory enough.

6.2 Critical Discussion Related to the Thesis

The concept of IoT Gateway is not standardized and is not universally recognized worldwide. The main future study could be about if there are alternative approaches or ideas related to IoT networks and its function.

Therefore, future study of alternative concepts is required.

6.3 The Role and Effect of the Used Technology in the Thesis

Because it is a new field, there are a lot of different technological applications related to IoT. In this thesis, group of the most common and popular implementations has been research and executed, even though alternative technologies and solutions are mentioned. However, all of these possible technologies and products need to be examined to have a clear vision of what the “IoT of the future” could be. Even so, a newly developed technology could also disrupt the market and change the way the IoT was heading into something that could not be imagined currently.

Therefore, available technology and related products greatly influence the direction of research and development and even though they can all be taken into account, new developments could shatter the previously foreseen predictions.

6.4 Continued Development and Future Work

As stated before, this thesis barely scratched the surface of IoT methodologies, technologies and concepts. The main focus of the future work should be in the direction of named and explained challenges in this thesis. However, this is not a restriction in itself because new ideas, solutions and even problems may arose in time to be tackled by researchers and engineers.

Therefore, future work related to IoT should focus on the challenges and if possible find new challenges to face and research in the future.

REFERENCES

A Light, R. (2017). Mosquitto: server and client implementation of the MQTT protocol. *The Journal of Open Source Software*, *2*(13), 265.

Abbasi, M. A., Memon, Z. A., Memon, J., Syed, T. Q., & Alshboul, R. (2017). Addressing the future data management challenges in IoT: a proposed framework. *International Journal of Advanced Computer Science and Applications*, *8*(5), 197–207.

Arduino Software IDE. (2005). *Arduino Company*. Retrieved September 11, 2019, from https://www.arduino.cc/en/Main/Software

Ashton, K. (2009). That "Internet of Things" Thing. *RFID Journal*, *22*(7), 97–114.

Atom. (2014). *GitHub*. Retrieved October 7, 2019, from https://atom.io/

Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, *54*(15), 2787–2805.

Axelson, J. (2015). *USB Complete: The Developer's Guide, Fifth Edition* 5th ed. Chicago, IL: *Lakeview Research LLC*.

Banks, A., & Gupta, R. (2014). MQTT Version 3.1. 1. *OASIS Standard*.

Boyd, B. (2014). *MQTT - A practical protocol for the Internet of Things*. Retrieved February 12, 2019, from https://www.slideshare.net/BryanBoyd/mqtt-austin-api

Buyya, R., & Dastjerdi, A. V. (2016). *Internet of Things: Principles and Paradigms*. Amsterdam Boston Heidelberg: Elsevier.

Chase, J. (2013). The Evolution of the Internet of Things. *Texas Instruments*, *1*, 1–6.

Chen, Y., Shu, J., Zhang, S., Liu, L., & Sun, L. (2009). Data Fusion in Wireless Sensor Networks. In *2009 Second International Symposium on Electronic Commerce and Security*, 2, 504–509.

Comparison of MQTT implementations. (2019). In *Wikipedia*,. Retrieved September 4, 2019, from https://en.wikipedia.org/w/index.php?title=Comparison_of_MQTT_implementations&oldid=913230850

Cope, S. (2017). *Introduction to MQTT-SN (MQTT for Sensor Networks)*. Retrieved September 10, 2019, from http://www.steves-internet-guide.com/mqtt-sn/

D1 mini. (2017). *WEMOS Electronics*. Retrieved May 3, 2018, from https://wiki.wemos.cc/products:d1:d1_mini

D1 Mini Image. (n.d.). Retrieved May 3, 2018, from https://wiki.wemos.cc/_media/products:d1:d1_mini_v3.0.0_1_16x9.jpg

Desktop PC Image. (n.d.). Retrieved May 3, 2018, from https://pixabay.com/en/computer-desktop-workstation-office-158675/

Díaz, M., Martín, C., & Rubio, B. (2016). State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. *Journal of Network and Computer Applications*, *67*, 99–117.

Eclipse Mosquitto. (2009). *Eclipse Foundation*. Retrieved September 4, 2019, from https://mosquitto.org/

Eduroam – World Wide Education Roaming for Research & Education. (2002). Retrieved October 6, 2019, from https://www.eduroam.org/

Emara, K. A., Abdeen, M., & Hashem, M. (2009). A gateway-based framework for transparent interconnection between WSN and IP network. In *IEEE EUROCON 2009*, 1775–1780. St. Petersburg, Russia: *IEEE*.

ESP8266. (2014). *Espressif Systems*. Retrieved September 30, 2019, from https://www.espressif.com/en/products/hardware/esp8266ex/overview

Evans, D. (2012). The internet of everything: How more relevant and valuable connections will change the world. *Cisco Internet Business Solutions Group (IBSG)*, 1–9.

Glória, A., Cercas, F., & Souto, N. (2017). Design and implementation of an IoT gateway to create smart environments. *Procedia Computer Science*, *109*, 568–575.

Grønbæk, I. (2008). Architecture for the Internet of Things (IoT): API and Interconnect. In *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*, 802–807.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, *29*(7), 1645–1660.

Guoqiang, S., Yanming, C., Chao, Z., & Yanxu, Z. (2013). Design and Implementation of a Smart IoT Gateway. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, 720–723.

HiveMQ Logo. (n.d.). Retrieved May 3, 2018, from https://www.hivemq.com/wp-content/uploads/logo.png

Hossain, M. M., Fotouhi, M., & Hasan, R. (2015). Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things. In *2015 IEEE World Congress on Services*, 21–28.

Hunkeler, U., Truong, H. L., & Stanford-Clark, A. (2008). MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks. In *3rd International Conference on Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008*, 791–798.

Karhula, P. (2016). *Internet of Things : a gateway centric solution for providing IoT connectivity*. Retrieved from https://jyx.jyu.fi/handle/123456789/50610

Khalil, E. A., & Özdemir, S. (2018). Overview of internet of things: Concept, characteristics, challenges and opportunities. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, *24*(2), 311–326.

Kim, S., Choi, H., & Rhee, W. (2015). IoT home gateway for auto-configuration and management of MQTT devices. In *2015 IEEE Conference on Wireless Sensors (ICWiSe)*, 12–17.

Kumar, R. (2017). *MQTT SN benefits over MQTT - Bevywise Networks*. Retrieved September 10, 2019, from https://www.bevywise.com/blog/benefits-of-mqtt-sn-over-mqtt/

Lueth, K. L. (2018). *State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating*. Retrieved February 8, 2019, from https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/

Ma, H.-D. (2011). Internet of Things: Objectives and Scientific Challenges. *Journal of Computer Science and Technology*, *26*(6), 919–924.

Medaglia, C. M., & Serbanati, A. (2010). An Overview of Privacy and Security Issues in the Internet of Things. In D. Giusto, A. Iera, G. Morabito, & L. Atzori (Eds.), *The Internet of Things*, 389–395. Springer New York.

Micro SD. (n.d.). Retrieved October 6, 2019, from https://www.sdcard.org/developers/overview/index.html

Min, D., Xiao, Z., Sheng, B., Quanyong, H., & Xuwei, P. (2014). Design and implementation of heterogeneous IOT gateway based on dynamic priority scheduling algorithm. *Transactions of the Institute of Measurement and Control*, *36*(7), 924–931.

Min, D., Xiao, Z., Sheng, B., & Shiya, G. (2012). Design and implementation of the multi-channel RS485 IOT gateway. In *2012 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, 366–370.

Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, *10*(7), 1497–1516.

Miraz, M. H., Ali, M., Excell, P. S., & Picking, R. (2015). A review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT). In *2015 Internet Technologies and Applications (ITA)*, 219–224.

MQTT. (1999). Retrieved September 18, 2019, from http://mqtt.org/

MQTT Dashboard. (n.d.). *HiveMQ*. Retrieved May 3, 2018, from http://www.mqtt-dashboard.com/

NetAnim. (n.d.). (Version 3.108). *ns-3 Project*. Retrieved October 7, 2019, from https://www.nsnam.org/wiki/NetAnim_3.108

NetBeans. (2003). *Apache Software Foundation*. Retrieved October 7, 2019, from https://netbeans.org/

Nordrum, A. (2016). *The Internet of Fewer Things*. Retrieved September 11, 2018, from https://spectrum.ieee.org/telecom/internet/the-internet-of-fewer-things

Ns-3. (2008). *ns-3 Project*. Retrieved May 4, 2018, from https://www.nsnam.org/

Ns3 Manual. (2019). ns-3 Project. Retrieved September 4, 2019, from https://www.nsnam.org/docs/release/3.30/manual/ns-3-manual.pdf

Ns-3 Python Manual. (n.d.). Retrieved October 7, 2019, from https://www.nsnam.org/docs/manual/html/python.html

Piper, A. (2013). *MQTT for Sensor Networks - MQTT-SN | MQTT*. Retrieved October 11, 2018, from http://mqtt.org/2013/12/mqtt-for-sensor-networks-mqtt-sn

PuTTY. (1999). Retrieved October 6, 2019, from https://www.chiark.greenend.org.uk/~sgtatham/putty/

PyCharm: Python IDE for Professional Developers. (2010). *JetBrains*. Retrieved May 4, 2018, from https://www.jetbrains.com/pycharm/

Python Programming Language. (1991). *Python Software Foundation*. Retrieved September 10, 2019, from https://www.python.org/

Raspberry Pi 3 Image. (n.d.). Retrieved May 3, 2018, from https://www.raspberrypi.org/app/uploads/2017/05/Raspberry-Pi-3-Ports-1-1833x1080.jpg

Raspberry Pi 3 Model B. (2016). *Raspberry Pi Foundation*. Retrieved September 11, 2019, from https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Raspbian. (2013). Retrieved May 4, 2018, from https://www.raspberrypi.org/downloads/raspbian/

Reyna, A., Martín, C., Chen, J., Soler, E., & Díaz, M. (2018). On blockchain and its integration with IoT. Challenges and opportunities. *Future Generation Computer Systems*, *88*, 173–190.

Riedel, T., Fantana, N., Genaid, A., Yordanov, D., Schmidtke, H. R., & Beigl, M. (2010). Using web service gateways and code generation for sustainable IoT system development. In *2010 Internet of Things (IOT)*, 1–8.

Singh, D., Tripathi, G., & Jara, A. J. (2014). A survey of Internet-of-Things: Future vision, architecture, challenges and services. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, 287–292.

Stanford-Clark, A., & Truong, H. L. (2013). MQTT For Sensor Networks (MQTT-SN) Protocol Specification. *International Business Machines (IBM) Corporation Version*, *1*, 28.

Stout, W. M. S., & Urias, V. E. (2016). Challenges to securing the Internet of Things. In *2016 IEEE International Carnahan Conference on Security Technology (ICCST)*, 1–8.

Thakare, S., Patil, A., & Siddiqui, A. (2016). The internet of things – Emerging technologies, challenges and applications. *International Journal of Computer Applications*, *149*(10), 21–25.

Thompson, K., & Ritchie, D. M. (1971). Unix Manual, first edition. Bell Labs. Retrieved September 29, 2019, from https://www.bell-labs.com/usr/dmr/www/1stEdman.html

UML Tutorial. (n.d.). Retrieved September 24, 2019, from https://sparxsystems.com/resources/tutorials/uml/part1.html

Vermesan, O., & Friess, P. (Eds.). (2013). *Internet of things: converging technologies for smart environments and integrated ecosystems*. Aalborg: River Publishers.

Vermesan, O., Friess, P., Guillemin, P., Gusmeroli, S., Sundmaeker, H., Bassi, A., … Doody, P. (2009). Internet of things strategic research roadmap. *Internet of Things-Global Technological and Societal Trends*, *1*, 9–52.

What is Hotspot? (n.d.). Retrieved September 30, 2019, from https://www.intel.com/content/www/us/en/tech-tips-and-tricks/what-is-a-hotspot.html

Zeinab, K. A. M., & Elmustafa, S. A. A. (2017). Internet of things applications, challenges and related future technologies. *World Scientific News*, *67*(2), 126–148.

Zhu, Q., Wang, R., Chen, Q., Liu, Y., & Qin, W. (2010). IOT Gateway: Bridging Wireless Sensor Networks into Internet of Things. In *Proceedings of the 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, 347–352. IEEE Computer Society.