



YAŞAR UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MASTER THESIS

**DESIGN AND IMPLEMENTATION OF A
CUSTOMIZED MICRO SERVER PLATFORM FOR
INDOOR MONITORING AND CONTROL**

ZİRVE BARAN TOTUK

THESIS ADVISOR: ASSOC. PROF. DR. TUNCAY ERCAN

COMPUTER ENGINEERING

PRESENTATION DATE: 08/08/2019

BORNOVA / İZMİR
AUGUST 2019



We certify that, as the jury, we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

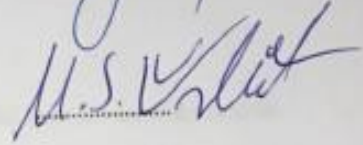

Jury Members:

Assoc. Prof. Tuncay ERCAN
Yaşar University

Prof. Dr. Mehmet S. ÜNLÖTÜRK
Yaşar University

Assoc. Prof. Mehmet Hilal ÖZCANHAN
Dokuz Eylül University

Signature:



Prof. Dr. Cüneyt GÜZELİŞ
Director of the Graduate School



ABSTRACT

DESIGN AND IMPLEMENTATION OF A CUSTOMIZED MICRO SERVER PLATFORM FOR INDOOR MONITORING AND CONTROL

Totuk, Zirve Baran

Msc., Computer Engineering

Advisor: Assoc. Prof. Dr. Tuncay Ercan

August 2019

The Industrial Internet of Things (IIoT) concept, emerging from the use of the Internet of Things (IoT) applications in industry, provides end-users with important functions such as productivity, flexibility, traceability, and security in their workplaces. Indoor monitoring and control systems, which are the leading applications of the IIoT, play a major role in providing these elements. Over the years, these systems have evolved from wired alarm systems to data collection terminals, wireless sensor networks, and then to intelligent gateways with cloud support. However, the number of points a single end-user wants to control can reach up to tens of thousands, as well as high service quality and commercial expectations. At this point, the existing cloud computing solutions do not provide the required service quality, while the new edge and fog computing solutions are far beyond the acceptable limits in terms of cost.

In order to meet all these expectations, a modular, single-board computer-based micro-server platform with a compact software structure, offering optimum performance and cost, is designed and developed in accordance with the fog computing concept.

When the results obtained from the test processes implemented under real scenarios are examined, it is seen that the hardware used and the software infrastructure, which are all open-source, are sufficient and usable for indoor monitoring and control applications and they are more efficient in comparison to the systems that are currently used.

Key Words: Internet of things, industrial internet of things, IoT, IIoT, fog computing, micro server, indoor monitoring and control systems

ÖZ

KAPALI ORTAMLARDA İZLEME VE DENETLEME İÇİN ÖZELLEŞTİRİLMİŞ MIKRO SUNUCU PLATFORMUNUN TASARIMI VE UYGULANMASI

Totuk, Zirve Baran

Yüksek Lisans Tezi, Bilgisayar Mühendisliği

Danışman: Doç. Dr. Tuncay Ercan

Ağustos 2019

Nesnelerin interneti uygulamalarının (IoT) sanayide kullanılmasıyla ortaya çıkan endüstriyel nesnelerin interneti (IIoT) konsepti son kullanıcılarına çalışma alanlarında ihtiyaç duyduğu verimlilik, esneklik, takip edilebilirlik ve güvenlik gibi önemli fonksiyonları sağlamaktadır. Endüstriyel nesnelerin interneti uygulamalarının başında gelen kapalı ortam izleme ve kontrol sistemleri bu unsurların sağlanmasında büyük rol oynamaktadır. Bu sistemler yıllar içerisinde kablolu alarm sistemlerinden, veri toplama terminallerine, kablosuz sensör ağlarına ardından bulut bilişim destekli akıllı ağ geçitlerine evrilmiştir. Ancak günümüzde tek bir son kullanıcının kontrol etmek istediği nokta sayısı on binleri bulabildiği gibi, yüksek servis kalitesi ve ticari beklentilerin karşılanması da zorunludur. Bu noktada günümüzde var olan bulut bilişim çözümleri gerekli servis kalitesini sağlayamazken sunulan yeni kenar ve sis bilişim çözümleri ise maliyet açısından kabul edilebilir sınırların çok üzerinde kalmaktadır.

Tüm bu beklentileri karşılamak adına bu çalışmada sis bilişim konseptine uygun, kompakt yazılım yapısına sahip, optimum performans ve maliyet sunabilen, çalışma alanına göre özelleştirilebilir, modüler, tek kartlı bilgisayar tabanlı bir mikro sunucu platformu tasarlanmış ve geliştirilmiştir.

Gerçek senaryolar altında uygulanan test süreçlerinden elde edilen sonuçlar incelendiğinde geliştirilen donanımların ve tamamı açık kaynak kodlu olan yazılım altyapısının kapalı ortam izleme ve kontrol uygulamaları için yeterli ve kullanılabilir

olduđu, ayrıca maliyet olarak hali hazırda kullanılan sistemlere göre daha verimli olduđu görölmüştür.

Anahtar Kelimeler: Nesnelerin internet, endüstriyel nesnelerin interneti, IoT, IIoT, sis bilişim, mikro sunucu, kapalı ortam izleme ve kontrol sistemleri



ACKNOWLEDGMENTS

First of all, I would like to thank my supervisor Tuncay Ercan for his guidance and patience during this study.

I would like to express my enduring love to my parents, who are always supportive, loving and caring to me in every possible way in my life.

I would like to thank Zeynep İnci Koçer for her supports and motivation during this study.

I would like to thank Mustafa Seçmen, Ceyhan Türkmen, Çağla Sarvan and Mehmet Barbaros Küçük for their supports and helps during this study.

Zirve Baran Totuk
İzmir, 2019



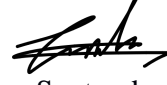
TEXT OF OATH

I declare and honestly confirm that my study, titled “DESIGN AND IMPLEMENTATION OF A CUSTOMIZED MICRO SERVER PLATFORM FOR INDOOR MONITORING AND CONTROL” and presented as a Master’s Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions. I declare, to the best of my knowledge and belief, that all content and ideas drawn directly or indirectly from external sources are indicated in the text and listed in the list of references.

Full Name

Signature

.....Zirve Baran TOTUK.....



September 6, 2019



TABLE OF CONTENTS

ABSTRACT	v
ÖZ	vii
ACKNOWLEDGMENTS	ix
TEXT OF OATH	xi
TABLE OF CONTENTS	xiii
LIST OF FIGURES	xv
LIST OF TABLES	xvii
SYMBOLS AND ABBREVIATIONS	xix
CHAPTER 1 INTRODUCTION	1
1.1. MOTIVATION AND LITERATURE SEARCH.....	1
1.2. AIM OF THE THESIS.....	3
1.3. THESIS OVERVIEW AND OUTLINE OF THE THESIS.....	3
CHAPTER 2 INDOOR MONITORING AND CONTROL SYSTEMS	9
2.1. TYPES OF COMPUTING OVERVIEW.....	9
2.1.1. CLOUD COMPUTING.....	9
2.1.2. EDGE COMPUTING.....	11
2.1.3. FOG COMPUTING.....	11
2.2. HARDWARE OVERVIEW.....	13
2.2.1. ALARM SYSTEMS.....	13
2.2.2. DATA LOGGERS.....	14
2.2.3. SMART GATEWAYS.....	15
2.2.4. MICRO SERVERS.....	16
2.3. SOFTWARE OVERVIEW.....	17
2.3.1. SYSTEM SOFTWARE.....	17
2.3.2. MONITORING SOFTWARE.....	17
CHAPTER 3 MICRO SERVER PLATFORM HARDWARE DESIGN	18

3.1. ARCHITECTURE.....	18
3.2. RASPBERRY PI.....	19
3.1. MOTHERBOARD.....	20
3.2. WATCHDOG MODULE.....	21
3.3. WIRED COMMUNICATION MODULE.....	22
3.4. WIRELESS COMMUNICATION MODULE.....	23
3.5. MAIN NETWORK INTERFACE MODULE.....	24
3.6. ETHERNET SWITCH MODULE.....	25
3.7. RELAY MODULE.....	25
3.8. INPUT/OUTPUT MODULE.....	25
3.9. STORAGE MODULE.....	26
CHAPTER 4 MICRO SERVER PLATFORM SOFTWARE DESIGN.....	27
4.1. OPERATING SYSTEM AND DOCKER.....	27
4.2. INPUT/OUTPUT SERVICE.....	28
4.3. NETWORK ARCHITECTURE.....	30
4.4. DATABASE SERVICE AND SYSTEM METRICS AGENT SERVICE.....	31
4.5. NETWORK ATTACHED STORAGE SERVICE.....	32
4.6. DASHBOARD SERVICE.....	32
4.7. TEST PROCEDURES AND RESULTS.....	34
CHAPTER 5 CONCLUSIONS AND FUTURE RESEARCH.....	39
REFERENCES.....	41
appendix 1 – dashboard Listener Python Script.....	43
appendix 2 – Energy Analyzer Parser Python Script.....	47
appendix 3 – Wireless Network Module Listener Python Script.....	49

LIST OF FIGURES

Figure 2.1 Structure of Cloud Computing (from Gubbi, J.).....	10
Figure 2.2 Structure of Edge Computing (from Varghese, B).....	11
Figure 2.3 Structure of Fog Computing (from Verma, 2018).....	12
Figure 2.4 Structure of Generic Alarm System (from Networktechinc.com, 2019).....	14
Figure 2.5 Reference Design of Data Logger Device	15
Figure 2.6 Structure of Smart Gateway (from Kang, 2017)	16
Figure 3.1 The Hardware Architecture of Micro Server Platform.....	18
Figure 3.2 Pinout of Raspberry Pi (from Raspberrypi.org, 2019)	20
Figure 3.3 Circuit Design of Motherboard.....	21
Figure 3.4 Motherboard	21
Figure 3.5 Hardware Structure of the Watchdog Module.....	22
Figure 3.6 Hardware Structure of the Wired Communication Module.....	22
Figure 3.7 Circuit Design of the Wired Communication Module	23
Figure 3.8 Hardware Structure of Wireless Communication Module	23
Figure 3.9 Circuit Design of Wireless Communication Module	24
Figure 3.10 Hardware Structure of Main Network Module.....	24
Figure 3.11 Connection Diagram of Ethernet Switch Module	25
Figure 3.12 Circuit Design of Input/Output Module	26
Figure 4.1 Software Architecture of the Micro Server Platform	28
Figure 4.2 Output Flow.....	29
Figure 4.3 Input Flow	29
Figure 4.4 Network Architecture of Micro Server Platform.....	30
Figure 4.5 Remote Access and Network Features	31
Figure 4.6 InfluxDB Admin Panel of Micro Server Platform	32
Figure 4.7 System Metrics Dashboard.....	33
Figure 4.8 Input/Output Module and Relay Module Dashboard	33

Figure 4.9 Indoor Measurements Dashboard.....	34
Figure 4.10 Outdoor (Parking Area / LoRa) Measurements Dashboard	34
Figure 4.11 Prototype Front Side	35
Figure 4.12 Prototype Back Side	35
Figure 4.13 Scenario Details	36



LIST OF TABLES

Table 2.1 Advantages and Disadvantages of Cloud Computing	10
Table 2.2 Comparison of the Computing Types	13
Table 3.1 Specifications of Raspberry Pi (from Raspberrypi.org, 2019).....	19
Table 4.1 Comparison of DietPi and Raspbian (from Dietpi.com, 2019).....	27
Table 4.2 Summary of Test Results	37
Table 4.3 Comparison with Other Devices	37
Table 4.4 Comparison with Other Devices	38



SYMBOLS AND ABBREVIATIONS

ABBREVIATIONS:

IIoT	Industrial Internet of Things
IoT	Internet of Things
ERP	Enterprise Resource Planning
ATM	Automated Teller Machine
QoS	Quality of Service
ADC	Analog to Digital Converter
VPN	Virtual Private Network
IDC	Insulation Displacement Connector
I ² C	Inter-Integrated Circuit
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
SSH	Secure Shell
CSV	Comma Separated Values
JSON	JavaScript Object Notation
SNMP	Simple Network Management Protocol
RTSP	Real Time Streaming Protocol





CHAPTER 1

INTRODUCTION

1.1. Motivation and Literature Search

The Industrial Internet of Things (IIoT) is a concept that is composed of a combination of information technologies and operational technologies. At this point, the industrial internet of things provides many benefits such as data collection, analysis, estimation, visualization, autonomous control or integration into enterprise resource planning (SAP etc.) systems to end-users (Mbohwa et. al., 2017). In addition, this concept, which covers many industrial applications, includes environmental monitoring and control applications.

Indoor monitoring and control applications are generally used for measure and control environmental, security or operational parameters such as temperature, humidity, energy consumption, motion detection and remote air conditioner control in areas such as base station, ATM and data centers where the number of branches reaches up to tens of thousands. The increase in the number of branches reveals one of the hidden problems of these applications. The high number of monitoring and control points has led to the solutions being low cost, simple alarm or contact systems until recently. In recent years, when the Internet of Things trend has increased, these simple systems have evolved into cloud-supported hardware and software. However, at this point, data losses, delayed or non-performing control operations, access problems in areas with a weak internet connection and high cloud service costs have occurred.

This poor performance of cloud computing in indoor monitoring and control applications has been demonstrated in many applications under the title of industrial internet of things and it has led to the emergence of new generation solutions such as edge or fog computing. These architectures started to be presented as specialized, local server systems operating near the application area.

The choice of high-performance edge or fog computing software and hardware in the IIoT with a high number of peripheral units and complexity has led to the acceptance

of technological innovation and commercial success under the leadership of companies such as Dell and Cisco (Pahl et al., 2016). However, the products that these companies put on the market, features such as high processing power and their ability to withstand the harsh working conditions (temperature, humidity) caused the supply, installation and integration costs to rise. In addition, all of these high-cost features has become unnecessary for indoor monitoring and control end-users.

Due to the high hardware costs, clouds created by single card computers which are developing rapidly (Raspberry Pi, Beaglebone, etc.) have become the solution proposition. These equipment are inexpensive, consume low energy and have sufficient resources to support indoor monitoring and control applications (Perera et al., 2017). In addition, the latest advances in lightweight operating system virtualization technologies such as Docker and Unikernels allow this hardware to use and replicate on-demand services on the edge of the network, that is, support edge/fog computing (Lertsinsrubtavee et al., 2017).

In recent studies, it is seen that edge and fog computing solutions developed with lightweight single card computers can replace cloud solutions.

Pahl and friends (2016) have argued with a similar view that the existing cloud computing systems are transformed into edge or fog information systems, but the costs of products offered by familiar companies in this field are unacceptable. As a solution, he proposed a single-card computer-based (Raspberry Pi), virtualized lightweight monitoring system.

Lertsinsrubtavee et al. (2017) suggested that end-users need both qualities of service (QoS) requirements and lower costs for edge computing solutions. In order to meet these expectations, they proposed a platform called Picasso where it was made lightweight by the virtualization of low-cost hardware. In addition, they aimed to determine the effect of the number of users on hardware resources and obtained results on the number of users and response times on the web server (called as Nginx).

Morabito et al. (2017) argued that edge and fog computing solutions will replace cloud computing solutions, and that services should be modular and virtualized to provide an expandable and flexible structure in edge computing solutions. Accordingly, they tested their virtualized, lightweight gateway (called LegIoT) in a real sensor network

and demonstrated the suitability of the results in edge computing-based IoT applications.

1.2. Aim of the Thesis

In this thesis, the design and development of a customized, modular and price/performance-optimized platform capable of performing indoor monitoring and control operations, operating in the fog layer are being discussed. Within the scope of the thesis, a new generation system will be introduced as an alternative to conventional data collection devices, smart gateways and servers related with indoor monitoring and control applications and thus efficiency about data transmission quality, data security, ease of integration and cost will be achieved.

1.3. Thesis Overview and Outline of the Thesis

Chapter 1 includes literature studies related to this thesis. The thesis was shaped according to these researches. The chapter also conveyed the aim of creating this thesis.

In Chapter 2 gives brief information about indoor monitoring and control systems. It describes types of computing, related hardware and software technologies and working areas.

Chapter 3 includes hardware design and implementation of micro server platform. This chapter includes the whole system architectures and details about hardware that are used or developed. In addition, the results of tests that are implemented have been given.

Chapter 4 includes software design and implementation of micro server platform. This chapter includes whole system architectures, flow charts and details about software that are used or developed. In addition, test scenarios that are realized on the whole system and results have been given.

In Chapter 5, the contributions of the thesis have been discussed. According to the results, future works and expectations have been describing.

CHAPTER 2

INDOOR MONITORING AND CONTROL SYSTEMS

Indoor monitoring and control systems, which are the sub-branches of the Industrial Internet of Things (IIoT), are systems that provide efficiency, traceability, manageability and security to end users with functions such as data collection, data processing, data storage, autonomous decision making and forecasting. In this section, the basic components of indoor monitoring and control systems are discussed under an overview of computing, hardware and software.

2.1. Types of Computing Overview

2.1.1. Cloud Computing

Cloud computing can be defined as systems that provide users with a compute resource at a remote point through an internet connection. User can store, process or monitor their data in the cloud. Cloud computing typically uses transfer protocols for data transfer, such as Hyper-Text Transfer Protocol (HTTP), Hyper-Text Transfer Protocol Secure (HTTPS), Message Queuing Telemetry Transport (MQTT), or The Constrained Application Protocol (CoAP).

For indoor monitoring and control applications, cloud computing can provide users with the same functions as storage, processing and monitoring of data collected from indoor areas. Structure of cloud computing is given in Figure 2.1.

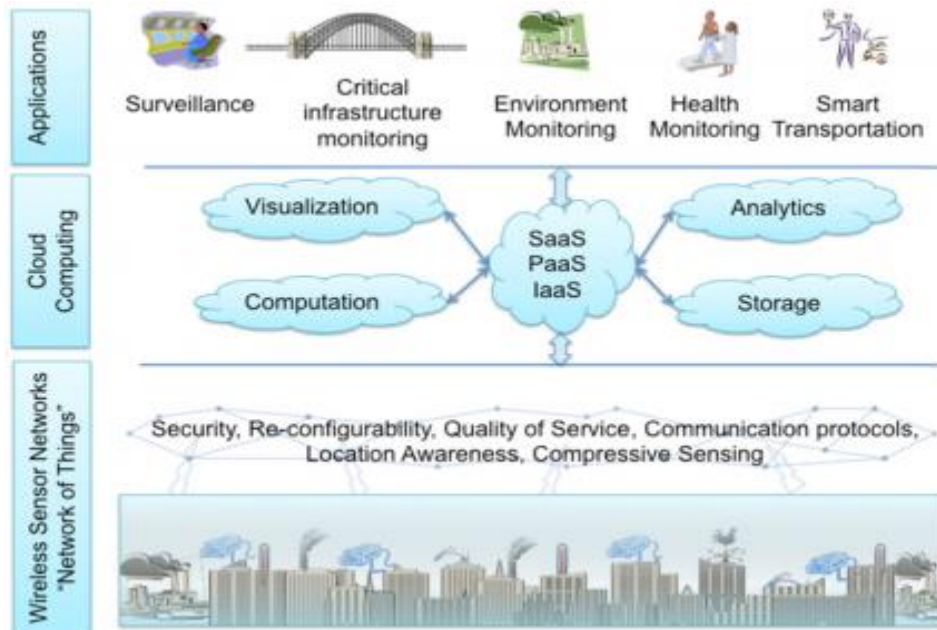


Figure 2.1 Structure of Cloud Computing (from Gubbi, J.)

At the end of the 90s, cloud computing systems, which are the continuous supporters of the Internet of Things (IoT) concept, started to lose their effectiveness due to the increase in the number of devices connected to the internet, data sizes and user needs. Data loss and delays in cloud computing based complex applications have led to edge computing and fog computing approaches, especially for IoT and IIoT applications (AI et. al., 2018). These approaches have revealed all the hidden disadvantages of cloud computing. Details of cloud computing advantages and disadvantages are given in Table 2.1.

Table 2.1 Advantages and Disadvantages of Cloud Computing

Advantages	Disadvantages
Fast start-up	Lack of control
Flexibility	Security
Unlimited data store	Latency
Easy back-up and recovery	Requires internet connection
Low cost	Data lose
	Technical Issues
	Software licensing and pricing

2.1.2. Edge Computing

The predictions that the number of devices connected to the Internet will continue to increase rapidly in the next few years and that central architectures such as cloud computing in Internet of Things (IoT) applications will be insufficient in many aspects such as management, security and communication quality have been supported by many companies such as Cisco and other authorities. In order to minimize these problems, it has been suggested that computing operations should be carried out by peripheral units connected to the internet or by computing elements located nearby. Thus, in edge computing, large data generated by different IoT devices can be processed at the edge of the network instead of transferring them to the central cloud infrastructure due to bandwidth and energy consumption concerns. Edge computing can provide faster response and higher quality services for more particularly critical operations than cloud computing. It also has a more convenient architecture to integrate with IoT to provide efficient and secure services to many end-users (AI et. al., 2018). Structure of edge computing is given in Figure 2.2.

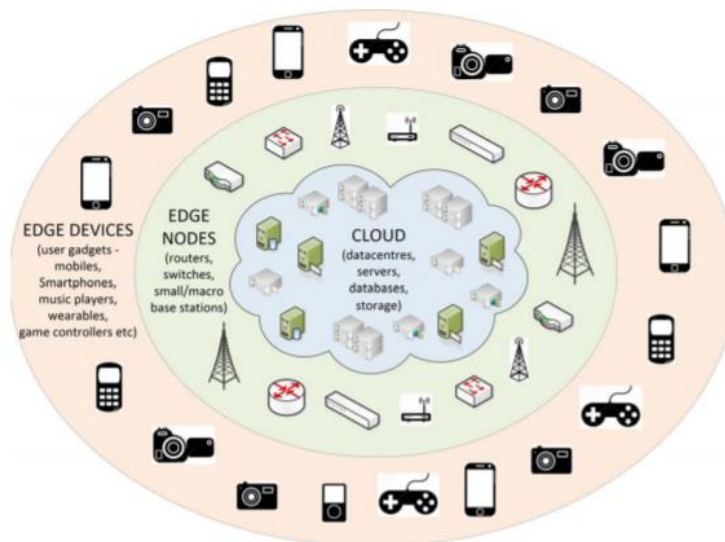


Figure 2.2 Structure of Edge Computing (from *Varghese, B*)

2.1.3. Fog Computing

Fog is an architecture that distributes computation, communication, control and storage closer to the end-users along the cloud-to-things continuum (Chiang et.al.,

2016). Sometimes the term “fog” is used interchangeably with the term “edge” although fog is broader than the typical notion of edge. The relevance of fog/edge is rooted in both the inadequacy of the traditional cloud and the emergence of new opportunities for the IoT, embedded artificial intelligence. In edge computing, data is processed without being transferred anywhere on the device or sensor. In contrast to edge computing, with fog computing, the data is processed in the IoT device within a fog node or local area network, and the meaningful data obtained can be stored in that device or converted into operations with low latency and bandwidth usage according to specified rules (Cisco.com, 2019). Fog computing structure is given in Figure 3. In addition, the three types of computing are compared in Table 2.2.

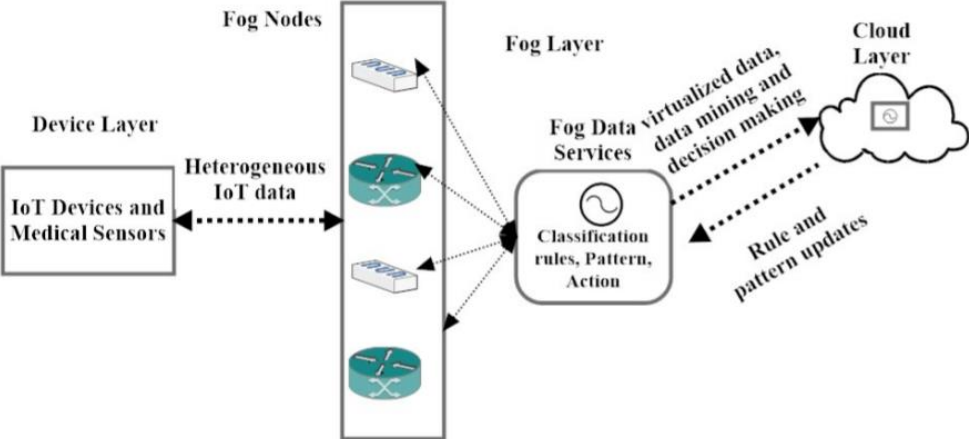


Figure 2.3 Structure of Fog Computing (from Verma, 2018)

Table 2.2 Comparison of the Computing Types

	Cloud Computing	Fog Computing	Edge Computing
Architecture	<ul style="list-style-type: none"> ✦ Central processing based model ✦ Fulfills the need for large amounts of data to be accessed more quickly, this demand is ever-growing due to cloud agility ✦ Accessed through internet 	<ul style="list-style-type: none"> ✦ Coined by CISCO ✦ Extending cloud to the edge of the network ✦ Decentralized computing ✦ Any device with computing, storage, and network connectivity can be a fog node, can be put on railway track or oil rig. ✦ Fog computing shoves intelligence down to the local area network level of network architecture, processing data in a fog node or IoT gateway 	<ul style="list-style-type: none"> ✦ Fog computing usually work with cloud and Edge can work without cloud or fog. ✦ Edge is limited to smaller number of peripheral layers ✦ Edge computing pushes the intelligence, processing power and communication of an edge gateway or appliance directly into devices like programmable automation controllers (PACs)
Pros	<ul style="list-style-type: none"> ✦ Easy to scale ✦ Low cost storage ✦ Based on Internet driven global network on robust TCP/IP protocol 	<ul style="list-style-type: none"> ✦ Real time data analysis ✦ Take quick actions ✦ Sensitive data remains inside the network ✦ Cost saving on storage and network ✦ More scalable than edge computing ✦ Operations can be managed by IT/OT team 	<ul style="list-style-type: none"> ✦ Edge computing simplifies internal communication by means of physically wiring physical assets to intelligent PAC to collect, analysis and process data. ✦ PACs then use edge computing capabilities to determine what data should be stored locally or sent to the cloud for further analysis
Cons	<ul style="list-style-type: none"> ✦ Latency/Response time ✦ Bandwidth cost ✦ Security ✦ Power consumption ✦ No offline-mode ✦ Sending raw data over internet to the cloud could have privacy, security and legal issues 	<ul style="list-style-type: none"> ✦ Fog computing relies on many links to move data from physical asset chain to digital layer and this is a potential point of failure. 	<ul style="list-style-type: none"> ✦ Less scalable than fog computing ✦ Interconnected through proprietary networks with custom security and little interoperability. ✦ No cloud-aware ✦ Cannot do resource pooling ✦ Operations cannot be extended to IT/OT team
Misc.		<ul style="list-style-type: none"> ✦ Less sensitive and non-real-time data is sent to the cloud for further processing ✦ Fog node can be deployed in private, community, public or hybrid mode 	<ul style="list-style-type: none"> ✦ PACs (programmable automation controllers) then use edge computing capabilities to determine what data should be stored locally or sent to the cloud for further analysis ✦ intelligence is literally pushed to the network edge, where our physical assets are first connected together and where IoT data originates ✦ The current Edge Computing domain is a sub-set of Fog Computing domain.

2.2. Hardware Overview

Indoor monitoring and control systems have developed in parallel with the development of technology over the years. These systems, which have been implemented with simple alarm systems, are now being implemented with customized smart gateways or micro servers. The equipment that has been used or is still being used up to now is examined under four sub-headings in this section.

2.2.1. Alarm Systems

Alarm systems are usually microcontroller-based equipment that performs parameters such as measuring environmental parameters and providing security in closed areas. A generic alarm system can detect emergencies, turn off and on existing devices, and send relevant data to a dashboard if there is an internet connection. They are simple to install and easy to use, and cost-effective compared to other alternatives. A generic alarm system structure is given in Figure 2.4.

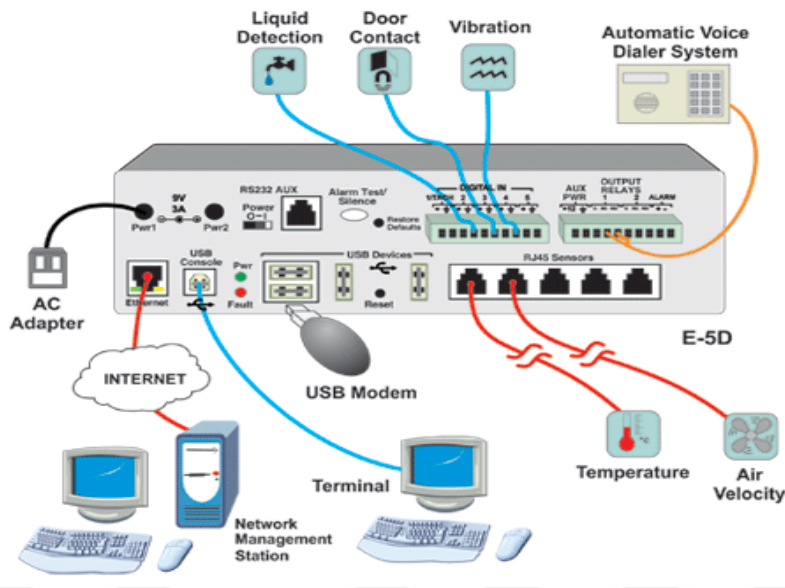


Figure 2.4 Structure of Generic Alarm System (from Networktechinc.com, 2019)

However, these systems are insufficient to meet the needs of users such as real-time monitoring/control, data storage, data analysis, remote management and updating. In addition, these devices are generally used in small areas such as base stations, data centers. For some big facilities, the need to increase the number of sensors and peripherals, record the measured values and display these results on a local network has led to the development of Data Logger hardware.

2.2.2. Data Loggers

Data logger hardware can manage data storage, processing and simple monitoring operations, especially in industrial areas. These devices are more heterogeneous in terms of communication protocols compared to customized alarm systems for indoor monitoring and control, but they have nearly the same hardware architecture. Generally, data loggers have analog to digital converters (ADC), flash memory and various wired or wireless communication modules. With this structure, the data logger can read digital or analog sensor values and store it. In addition, these devices communicates end devices such as a computer, smart phones via communication modules. The reference design of data logger device is given in the Figure 2.5.

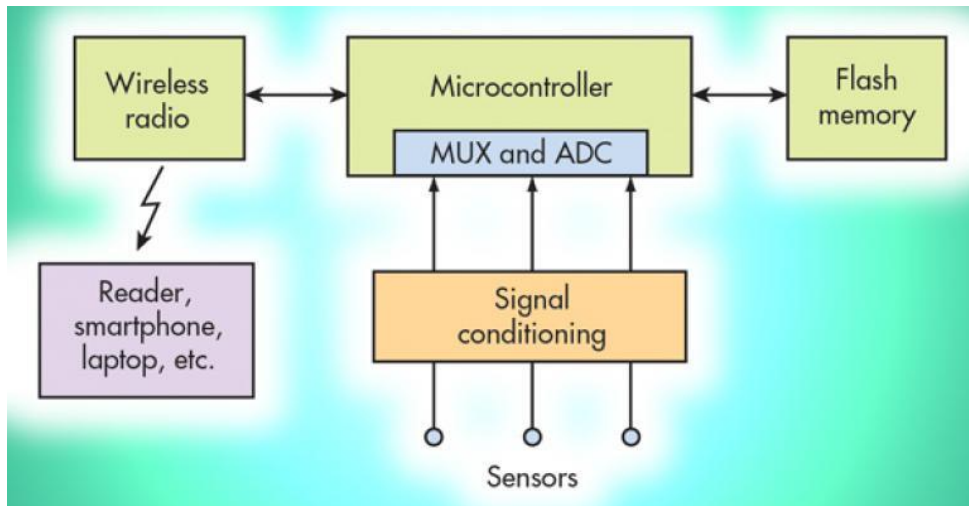


Figure 2.5 Reference Design of Data Logger Device

2.2.3. Smart Gateways

With the rapid rise of the Internet of Things trend, smart gateways have been developed as hardware with the operating system, remote access and heterogeneous network structures. (Ni et. al., 2013) These devices act as a bridge during the transmission of data from sensor networks or peripherals to the Internet. Thus, the data is transmitted to remote services. New generation gateways can perform data analysis or filter on small data packets. This reduces bandwidth usage and the need for the cloud. The smart gateway structure is given in Figure 2.6.

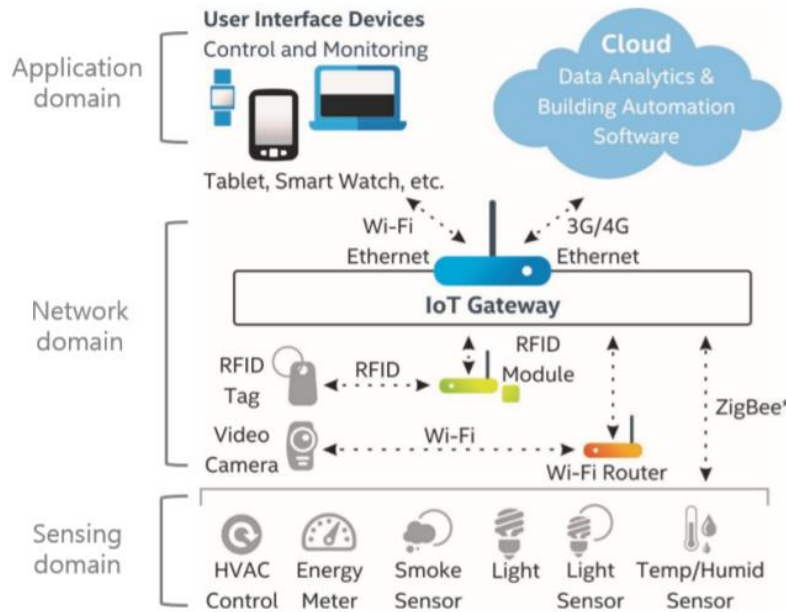


Figure 2.6 Structure of Smart Gateway (from Kang, 2017)

2.2.4. Micro Servers

In order to provide the necessary computing power in large IoT and IIoT applications and to create a cloud independent system, micro servers can perform operations such as data storage, processing or monitoring, as well as operations such as data analysis for large data sets. In addition, these systems reduce the monthly or annual software costs by enabling end-user software to run on-premise.

When evaluated within the scope of indoor monitoring and control applications, the fact that this hardware is a general-purpose server with reduced computing power leads to the use of too many software and hardware add-ons in related applications. This significantly increases system costs.

This equipment, which can best meet the expectations of the users, will significantly improve the efficiency of edge and fog computing if it is customized according to their usage areas and presented as ready to use platforms.

2.3. Software Overview

2.3.1. System Software

The system software used for indoor monitoring and monitoring systems does not vary much. If the system is based on microcontroller, usually embedded software developed with C / C ++ software languages is used.

For computer-based systems such as smart gateways or micro servers, Linux-based operating systems are often used as system software. In rare cases, Windows or manufacturer-specific operating systems can also be used. In these systems, software flexibility is provided in order to perform the necessary operations. The user may develop applications in any software language or use licensed applications on the system.

2.3.2. Monitoring Software

One of the most important parts of indoor monitoring and control systems is dashboard software. This software allows the user to display data, errors or alarms. In addition, remote control operations are also performed through these dashboards.

Nowadays, many providers provide monthly/annual paid dashboard service. This software provides services over the cloud by integrating an appropriate communication protocol into an existing system.

In a system created with the micro server or dedicated server, all elements such as dashboard, database or webserver can be run on-premise without the need for the cloud. At this point, the cloud will be needed for remote access to the system or to use services such as a virtual private network (VPN).

CHAPTER 3

MICRO SERVER PLATFORM HARDWARE DESIGN

3.1. Architecture

In order to have a modular and extensible structure in hardware, an architecture where all peripheral units will be connected to the motherboard by insulation displacement connector (IDC) connections is preferred. First, a circuit board with Raspberry Pi single-board computer connection, power input and IDC connectors has been designed and manufactured. Then, hardware changes were made on the peripheral units to be used and the data flow of the peripheral units was provided via USB and the power distribution was provided from the external power supply. The hardware architecture of micro server platform is given in Figure 3.1.

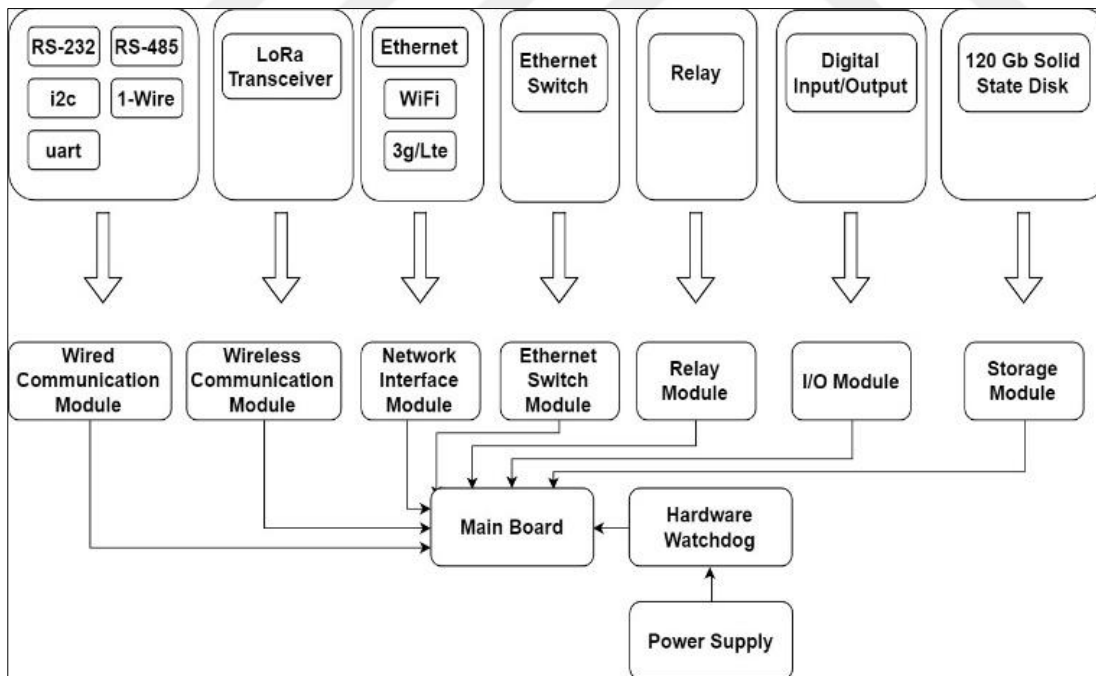


Figure 3.1 The Hardware Architecture of Micro Server Platform

3.2. Raspberry Pi

Raspberry Pi is a low-cost, credit-card-sized computer and supports many Linux-based operating systems. There are 40 input/output pins for developers. Most of these pins are used as digital input/output and some of them are reserved for wired communication with Inter-Integrated Circuit (I²C), Serial Peripheral Interface (SPI), and Universal Asynchronous Receiver Transmitter (UART) protocols (Raspberrypi.org, 2019). Built-in Wi-Fi, Bluetooth and Ethernet networking capabilities are also available on this single board computer.

In this study, this single board computer was used as the main control unit. The detailed specifications of the card and the pinout scheme are shown in Table 3.1 and Figure 5.

Table 3.1 Specifications of Raspberry Pi (from *Raspberrypi.org*, 2019)

SoC	Broadcom BCM2837 quad-core A53 1.4 GHz
Ram	1 GB LPDDR2 SDRAM
Networking	Gigabit Ethernet, Wi-Fi, BT 4.2
Storage	Micro SD
Input/Output	40 Pin
Ports	HDMI, Audio, 4xUSB, Camera and Display Interface

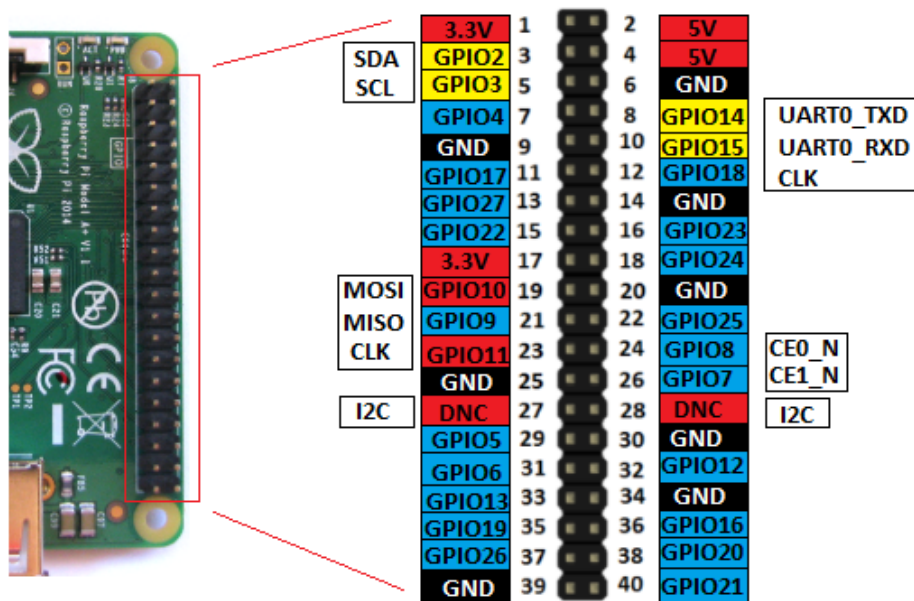


Figure 3.2 Pinout of Raspberry Pi (from *Raspberrypi.org*, 2019)

3.1. Motherboard

The task of the designed and produced motherboard is to distribute power to the modules on the platform and to transfer data between Raspberry Pi and the modules.

The electronic card shown in Figure 3.3 has 2x20 header suitable for Raspberry Pi pins, 8 IDC sockets for module connections and 4 screw terminals for power input and output.

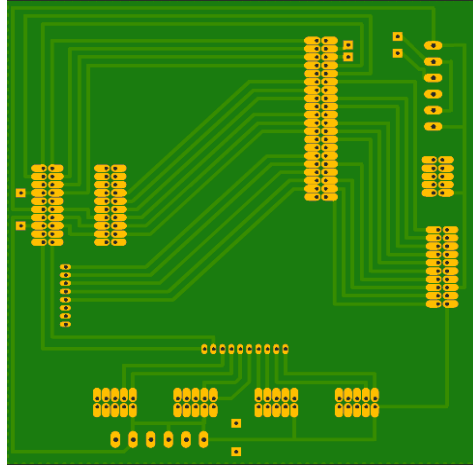


Figure 3.3 Circuit Design of Motherboard

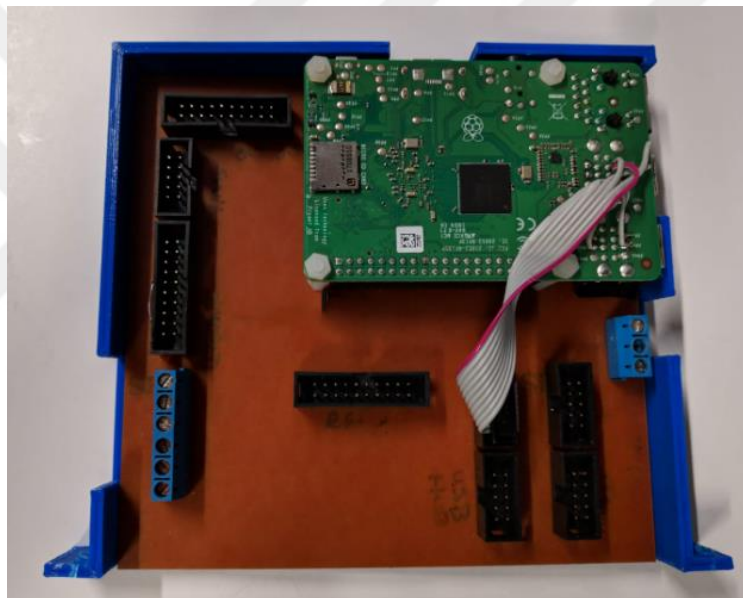


Figure 3.4 Motherboard

3.2. Watchdog Module

The external hardware watchdog module performs an autonomous safe restart operation by detecting the unexpected stop of the platform operation. This developed external hardware is based on STM32F103 microcontroller. The watchdog, which has serial communication with the Raspberry Pi on the platform, checks whether the Raspberry Pi is operational every two seconds. If the data flow stops, it restarts the system. This can be done as many times as the number of repetitions the user determines. The hardware structure of the watchdog module is given in Figure 3.5.

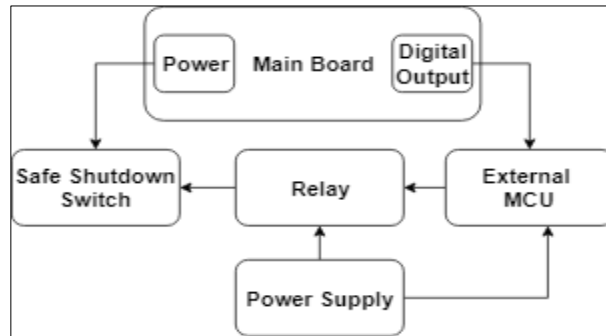


Figure 3.5 Hardware Structure of the Watchdog Module

3.3. Wired Communication Module

The wired communication module allows the platform to support wired communication protocols such as RS-232, RS-485, i2c, 1-wire and UART. All data flow is carried out via USB with the converters in the module and isolated ports. The hardware structure of the module is given in Figure 3.6 and the circuit design of the module is given in 3.7.

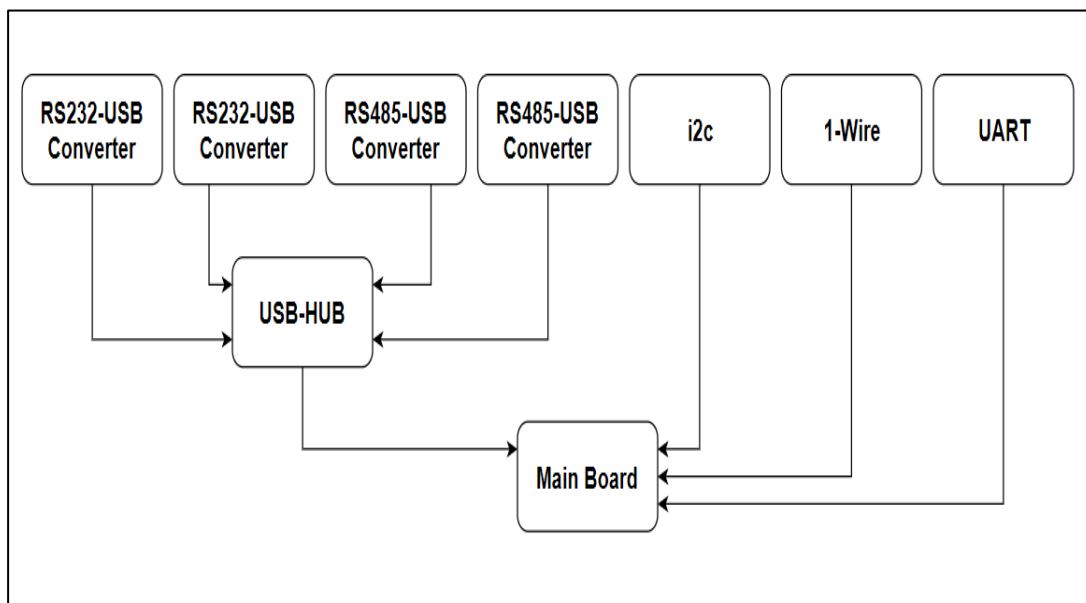


Figure 3.6 Hardware Structure of the Wired Communication Module

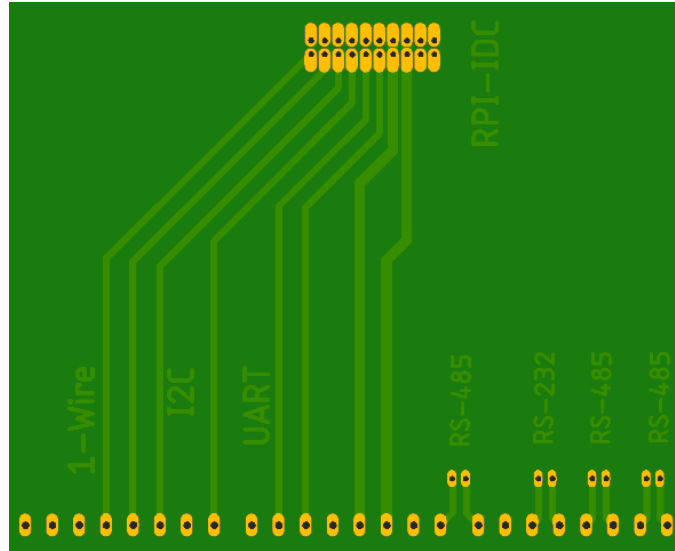


Figure 3.7 Circuit Design of the Wired Communication Module

3.4. Wireless Communication Module

The wireless communication module was designed to support the platform's highly preferred wireless communication protocols, such as LoRa, Xbee, or Z-Wave. The STM32F103 microcontroller in the module has been installed with an external hardware structure and the data collected by the selected communication protocols have been transferred to Raspberry Pi via USB port and made meaningful with the necessary software. The hardware structure of the wireless communication module is given in Figure 3.8 also circuit design is given in Figure 3.9.

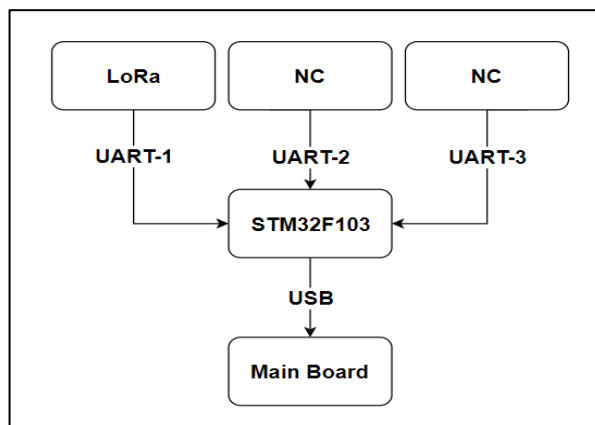


Figure 3.8 Hardware Structure of Wireless Communication Module

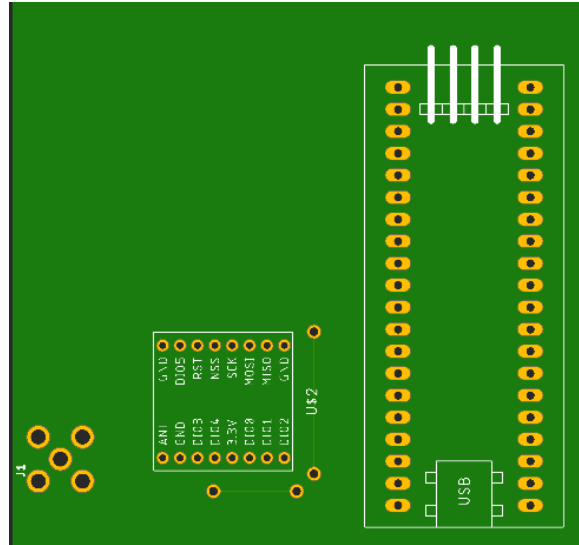


Figure 3.9 Circuit Design of Wireless Communication Module

3.5. Main Network Interface Module

The network interface module provides Ethernet, Wi-Fi and 3G / LTE interfaces required for an internet connection to the platform. These three connection types are made into a single module by selecting the corresponding USB converters. In addition, it is completely isolated from other network hardware and services on the platform (Ethernet switch, Wi-Fi hotspot etc.). The hardware structure of the main network interface module is given in Figure 3.10.

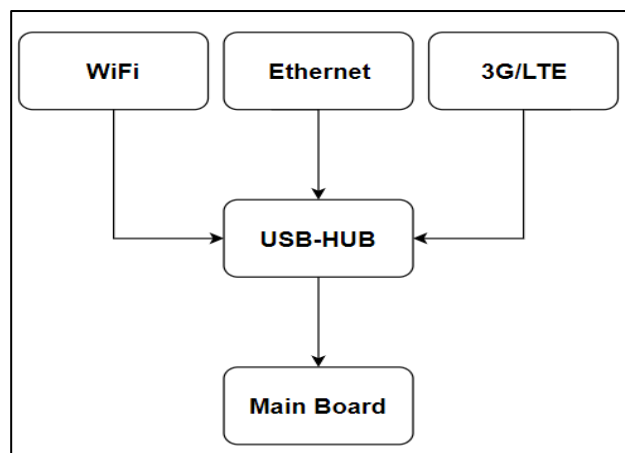


Figure 3.10 Hardware Structure of Main Network Module

3.6. Ethernet Switch Module

The network switch module provides an Ethernet switch with 8 ports to the developed platform. In this way, the platform can support communication protocols over Ethernet (SNMP, Modbus TCP / IP, RTSP, etc.). In addition, the platform can be used as a network unit that is isolated from the main network connections and can be controlled by firewall rules. Connection diagram of the module is given in Figure 3.11.

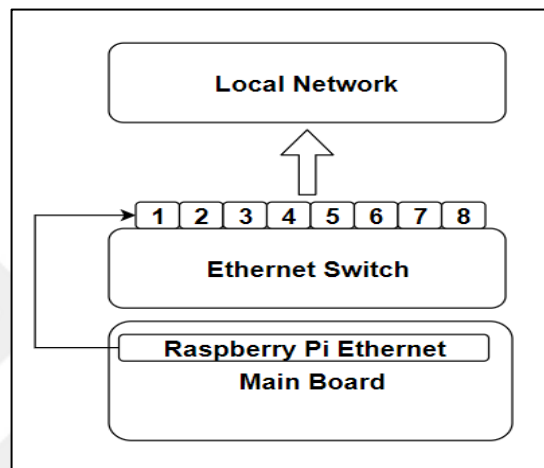


Figure 3.11 Connection Diagram of Ethernet Switch Module

3.7. Relay Module

The relay module contains six relays capable of switching up to 10 Amps in the 5-220 Volts range. As with similar modules, a card design has been realized for this module. Relay input and output ports were simplified on the produced electronic board and the connections required for Raspberry Pi were provided with IDC type connectors. This module allows remote device control operations to be performed in the working environment.

3.8. Input/Output Module

The Input / Output module has an 8-channel digital input. It is used for detecting data from sensors (flooding, fire, movement, magnetic contact etc.) which are used especially for emergency and safety. The module is responsible for transmitting low

and high information from digital sensors to the motherboard. The electronic board produced for this module has the necessary Raspberry Pi connections and 3.3-5-12 Volts outputs for various sensors. The circuit design of the module is given in Figure 3.12.

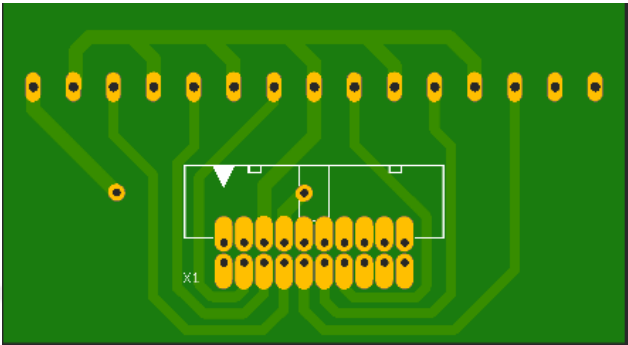


Figure 3.12 Circuit Design of Input/Output Module

3.9. Storage Module

The Raspberry Pi on the platform supports SD cards for the operating system and data storage in normal use. However, considering that high data storage size may be necessary for the developed platform, the SD card has been replaced with a 120-gigabyte solid-state disk connected via USB with the necessary software changes. In this way, the operating system can be selected to meet the needs of the end-user or the disk size can be increased to terabytes.

CHAPTER 4

MICRO SERVER PLATFORM SOFTWARE DESIGN

4.1. Operating System and Docker

A Debian / Linux based operating system called DietPi was chosen as the platform operating system. This operating system is an open-source distribution that has been lightweighted to operating systems such as Debian or Raspbian. DietPi's high performance is due to its small size, memory optimizations and removal of 3rd party software from the operating system which is not required by developers (Dietpi.com, 2019). Further details about DietPi and Raspbian operating systems are given in Table 4.1 with comparisons.

Table 4.1 Comparison of DietPi and Raspbian (from Dietpi.com, 2019)

General Stats	Diet Pi	Raspbian Lite
Total Image Size	589 MB	1424 MB
Minimum SD Card Size	4 GB	2 GB
Memory Usage (RAM)	23 MB	27 MB
Total Processes	11	18
Total Installed Packages	265	406
Bootup Time (Seconds)	14.2 Seconds	15.5 Seconds
Root Filesystem Usage	533 MB	931 MB
Core System Optimizations	Diet Pi	Raspbian Lite
Ramlog	Ram-Dietpi-Ramlog	Disk
Swapfile Swapiness	1	-
NTP Usage (System Time)	Yes + Other Choices	No
Auto HDD Spindown	10 Min	-
System Response Time	25 Ms	100 Ms
Temp Mounted to Ram	Ram	Disk
Scheduler	Noop	Deadline

In addition, virtualization was realized with the Docker architecture for the database, data agent and data display services that will work on the developed platform. In this way, these services, which have the most important role of the whole system, can easily be updated, cloned and managed, and this is one of the biggest factors in the platform showing a micro server characteristics. All services and architectures designed for the platform are given in Figure 4.1.

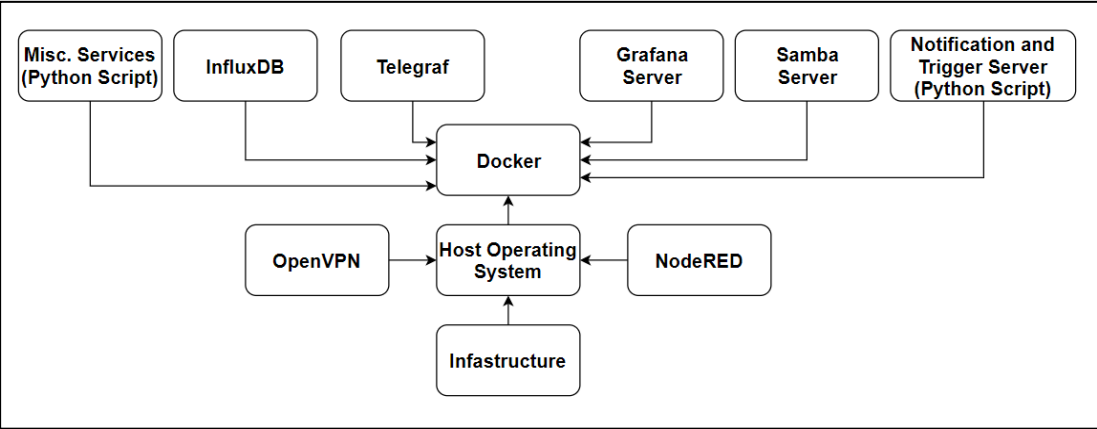


Figure 4.1 Software Architecture of the Micro Server Platform

4.2. Input/Output Service

NodeRed which is JavaScript-based, third-party, open-source development platform was used to allow the user to see the status of the sensors connected to the input / output module or to control the devices connected to the relay module (Nodered.org, 2019). In addition to ease of use, this development platform has recently started to be used frequently in industrial automation applications. The integration of this simplified platform allows end-users to create remote scenarios, decision-making mechanisms and make quick updates. All NodeRed flows that perform input/output operations are given in Figure 4.2 and Figure 4.3.

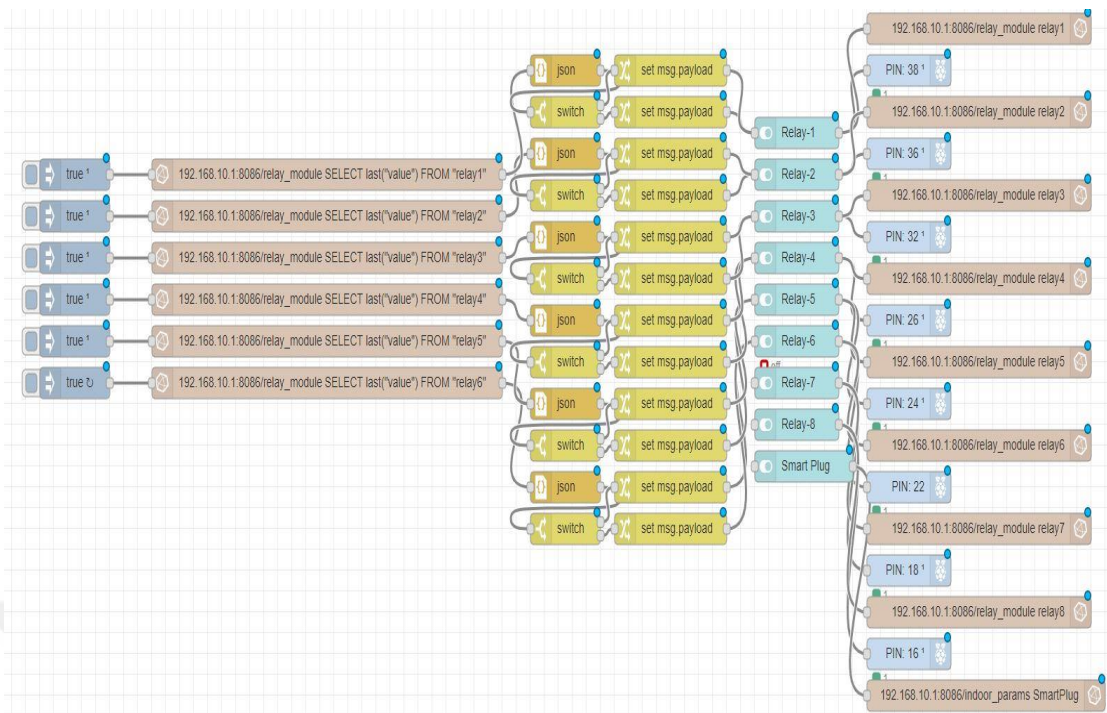


Figure 4.2 Output Flow

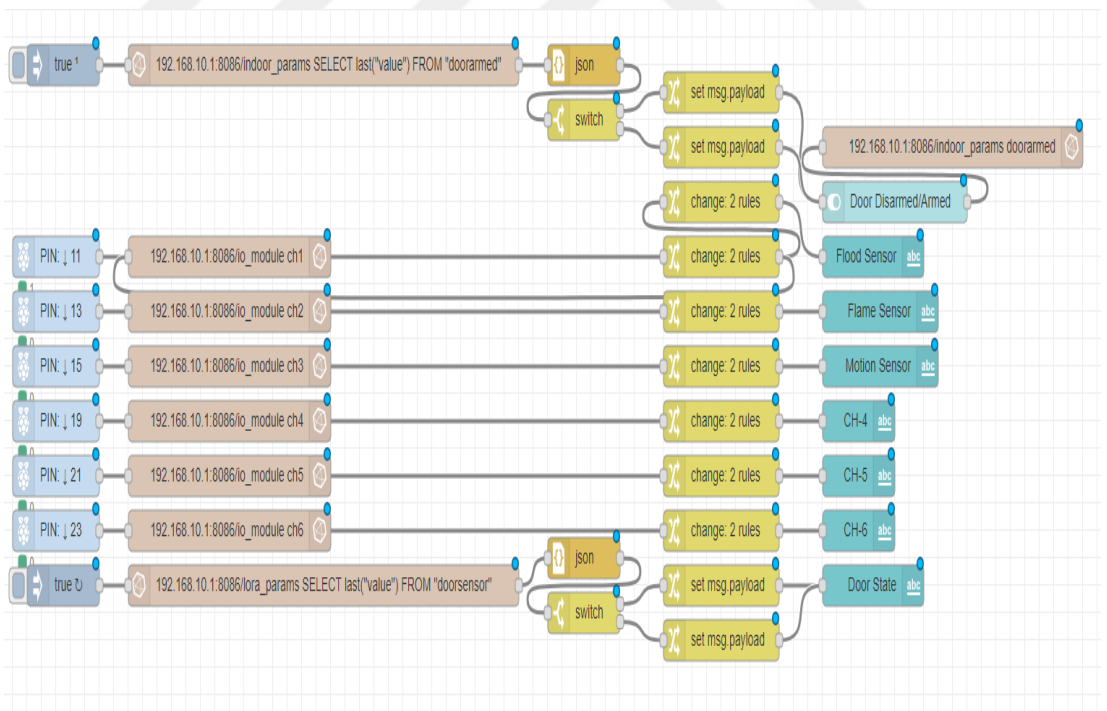


Figure 4.3 Input Flow

4.3. Network Architecture

There are Ethernet, Wi-Fi and 3G / LTE primary interfaces on the platform to provide internet or local network connection. There is also a Wi-Fi access point and Ethernet switch isolated from these interfaces. This access point enables devices such as Wi-Fi supported sensors and cameras to be connected to the platform, while Ethernet switch allows wired devices using Ethernet connections such as TCP/ IP, MODBUS or SNMP to be connected to the platform. In addition, the platform can be used as a network device in the area where it works, providing a local connection or internet connection to different devices when necessary. These connections are also configurable with firewall rules that will be defined on the platform. The summary of the network architecture of the platform and the IP configurations are given in Figure 4.4.

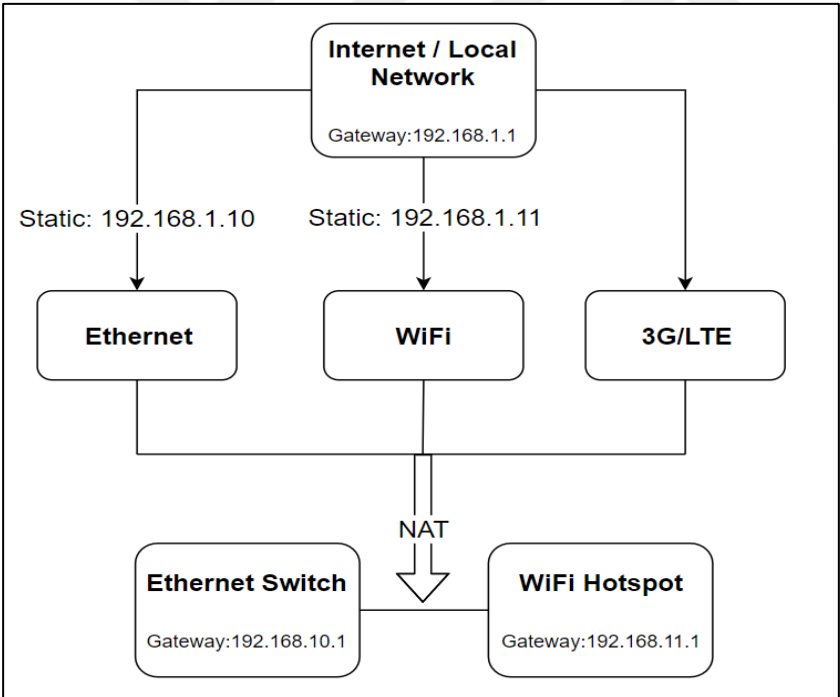


Figure 4.4 Network Architecture of Micro Server Platform

In addition to the conventional port forwarding or local network connection options for remote direct access to the platform (Secure Shell, VNC Remote desktop), an open-source virtual private server called OpenVPN was installed on a remote machine and the platform was defined as a virtual private network client. In this way, even if the

system does not work on the local network, all connections can be made securely over a private network. The system's detailed network connection architecture and all features offered for remote access are given in Figure 4.5.

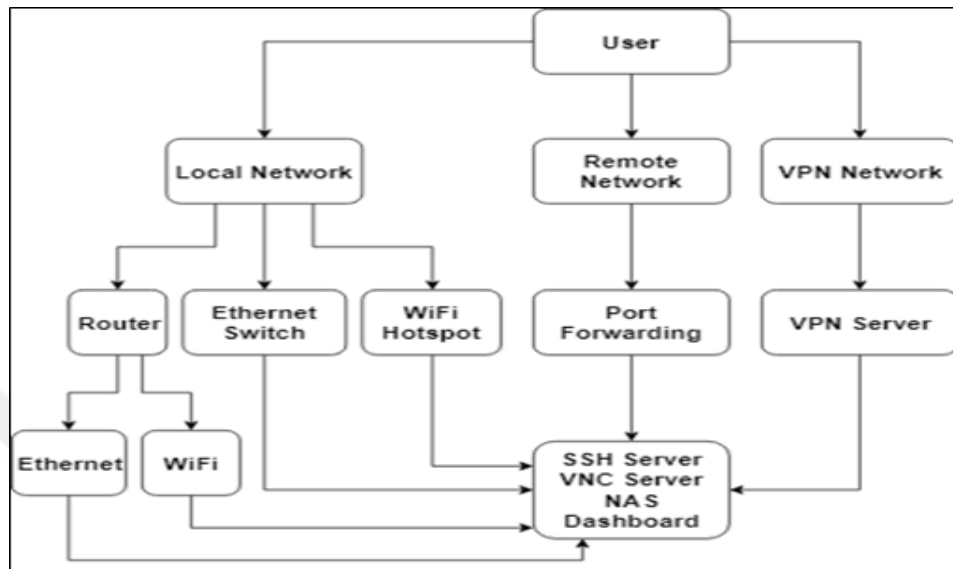


Figure 4.5 Remote Access and Network Features

4.4. Database Service and System Metrics Agent Service

InfluxDB time-series database was integrated into the developed platform. This database was optimized for DietPi and has fast query-answer capability (Docs.influxdata.com, 2019). InfluxDB has been used to record all platform-related actions, such as storing data from peripheral units, input / output operations, alarms and system metrics. In addition, InfluxDB supports export to third-party software (Time Series Admin, etc.) in file formats such as comma-separated values (CSV) or JavaScript object notation (JSON), which are commonly used for data analysis. The figure shows admin panel of the micro server platform.

The open-source data agent service called Telegraf is integrated into the platform for storing and presenting critical system data such as uptime, processor usage or network traffic to the end-user. In this way, the user will be able to monitor the system and detect any possible anomalies quickly.

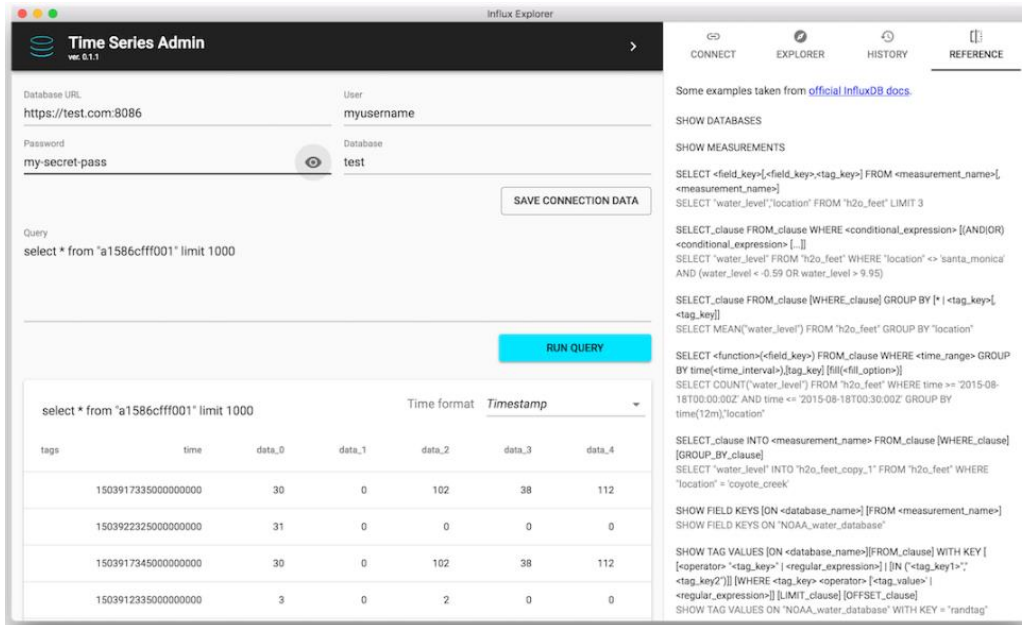


Figure 4.6 InfluxDB Admin Panel of Micro Server Platform

4.5. Network Attached Storage Service

An open-source network-attached storage server named Samba has been integrated to enable the system to share files to a local or remote point without the need for a cloud service. This integration supports both the system acting as a micro-server and running on the fog layer because users will be able to quickly transfer end-to-end data by accessing designated paths directly on the platform's storage without any cloud service, remote access or dashboard connection.

4.6. Dashboard Service

An open-source dashboard software called Grafana was used to visualize all monitoring and control operations on the system. This software is preferred because of its features such as Influxdb and Telegraf compatibility, data analysis, alarm and notification scenarios (Grafana Labs Blog., 2019). The dashboard screens of the developed platform are also given in Figure 4.7- 4.10.



Figure 4.7 System Metrics Dashboard

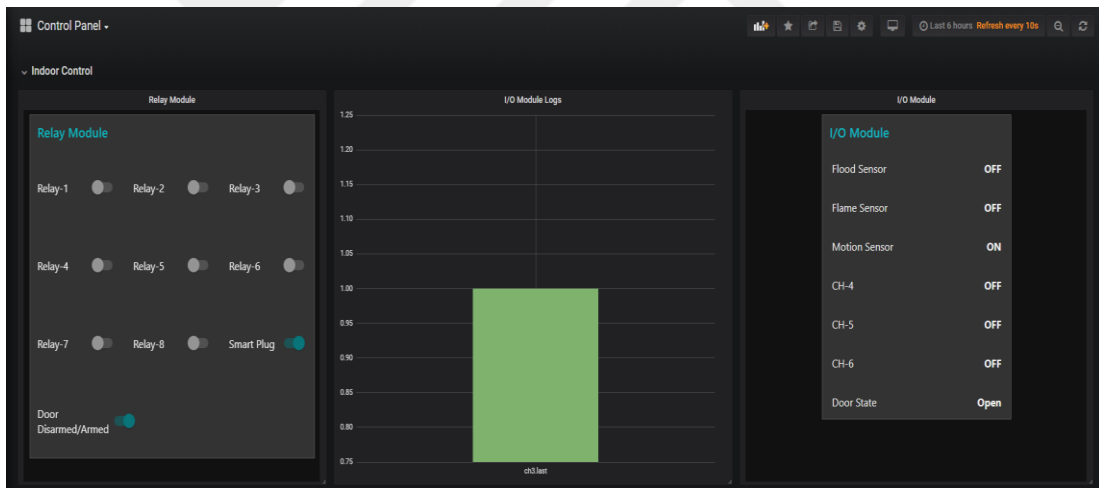


Figure 4.8 Input/Output Module and Relay Module Dashboard

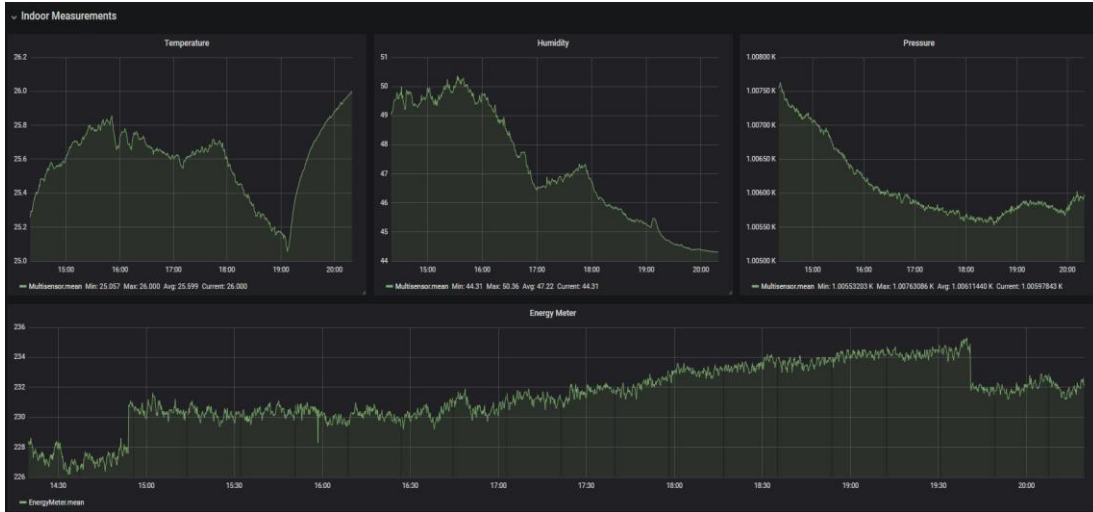


Figure 4.9 Indoor Measurements Dashboard

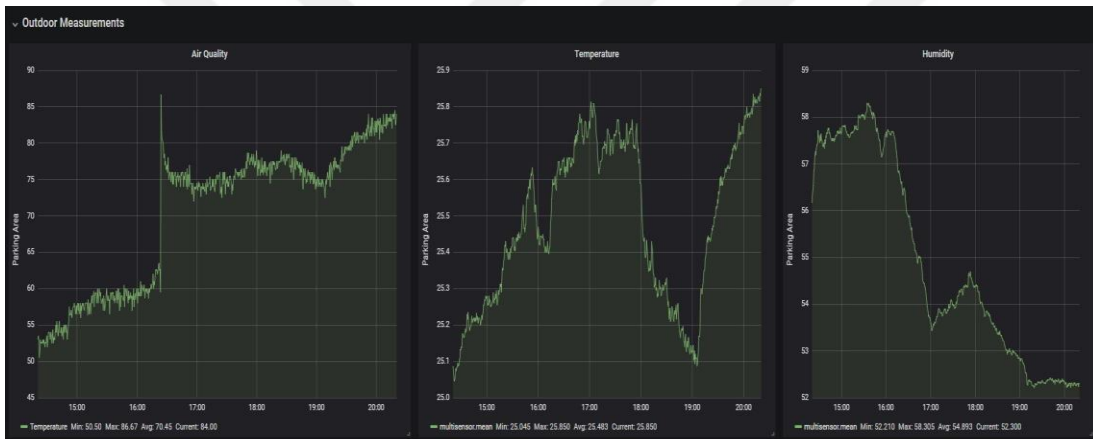


Figure 4.10 Outdoor (Parking Area / LoRa) Measurements Dashboard

4.7. Test Procedures and Results

After all the hardware and system software work packages were completed, 3D printed cases were produced for all modules and the prototype system was combined in 2U-rack case. The first prototype is given in Figure 4.11 and Figure 4.12.



Figure 4.11 Prototype Front Side



Figure 4.12 Prototype Back Side

After casing, a real indoor monitoring and control scenario was created to test the developed system and the system was operated for three weeks. Various peripheral units which related indoor monitoring and control were used during the test process. These peripherals include sensors, detectors, IP cameras and relay outputs. The measurement of temperature, humidity and pressure from the relevant sensors was carried out every 10 seconds. On the other hand, the detector has been continuously listened with Python listener services to obtain true / false information. In addition to this, real time stream was performed without recording via IP camera, and relay outputs and control operations (On / Off) were performed. All data and actions, except video data, are stored in the database by InfluxDB clients added to related services.

With the integration of InfluxDB into Grafana, all data is displayed on the dashboard with data-time graphs that can be filtered. In addition, user actions were taken from the dashboard using the Nodered widgets in the Grafana Dashboard for control operations and recorded in the database and the relevant operation was performed. Details such as peripheral units used in the scenario, collected or modified data types are given in Figure 4.13.

Device	Data Type	Operation	Interval	Database Record
Energy Analyzer	Active Energy	Read	10 seconds	+
Temperature Sensor	Temperature (C)	Read	10 seconds	+
Humidity Sensor	Humidity (%)	Read	10 seconds	+
Pressure Sensor	Pressure (hPa)	Read	10 seconds	+
Air Quality Sensor	Air Quality	Read	10 seconds	+
Fire Sensor	True/False	Read	Continuous	+
Flood Sensor	True/False	Read	Continuous	+
Motion Sensor	True/False	Read	Continuous	+
Magnetic Door Contact	True/False	Read	Continuous	+
IP Camera	Video (h264)	Read	Continuous	-
Data Agent	System Metrics	Read	10 seconds	+
Relay Module	True/False	Write	-	+
Smart Plug	True/False	Write	-	+

Figure 4.13 Scenario Details

In order to make sense of the data coming from the sensors and peripheral units, parser and listener scripts were developed on the platform with Python software language and these scripts were converted into system service. According to the data collected for 15 days, it was seen that the developed system worked without error under the scenario that can be considered as almost complex. System performance is summarized in Table 4.2.

Table 4.2 Summary of Test Results

Mean CPU Usage	38%
Mean CPU Temperature	62 C
Mean Memory Usage	45%
Total Database Number	5
Total Measurements	~2.8 Million
Total Database Size	172 Mb
Data Loss	Not Detected
Latency for Get Data from Digital Sensors	Maximum 1 second (10 seconds interval)
Latency for Get Data from Detectors	<35 milliseconds
Latency for Post Data to Peripherals	<100 milliseconds

In addition, when the platform developed was compared with three different indoor monitoring and control devices currently on sale, it was successful in almost all comparison parameters. These results are summarized in Table 4.3 and Table 4.4.

Table 4.3 Comparison with Other Devices

Feature	Standart Alarm System	Smart Gateway	Edge Computing Server	Micro Server Platform
CPU	Microcontroller	Quad-core Cortex A53 @ 1.2 GHz	Intel CPU E3825 1.33GHz	Quad-core Cortex A53 @ 1.4 GHz
Memory	Flash etc.	1 Gb	2 Gb	1 Gb
Storage	Eeprom etc.	64Gb (SD)	120gb (SSD)	120gb (SSD)
Operating System	-	Debian/Linux	Debian/Linux	Debian/Linux
Ethernet Interface	1x	1x	2x	1x Main Network 8x Switch
Serial Interface	-	-	-	1x
Rs-232 Interface	-	2x	1x	2x
Rs-485 Interface	-	2x	3x	2x
Usb Interface	-	2x	2x	4x

Table 4.4 Comparison with Other Devices

Feature	Standart Alarm System	Smart Gateway	Edge Computing Server	Micro Server Platform
Digital Input	+	+	+	+
Digital Output	+	+	+	+
Relay	2x	-	-	8x
Bluetooth	-	-	-	+
WiFi	-	-	1x	2x (Main / Hotspot)
LoRa	-	-	-	1x
LTE/3G	-	-	+	+
Software	Embedded Software	Operating System	Operating System	Operating System Docker Nodered Database System Metric Agent Data Visulation
Price	~250\$	~400\$	~2000\$	~125\$ (Cost)

CHAPTER 5

CONCLUSIONS AND FUTURE RESEARCH

Indoor monitoring and control systems are a concept that will increase the importance for end-users. Because these systems provide efficiency, security, manageability and savings with the data sets collected from the closed areas to the users. However, it is an important criterion to provide these benefits without interruption. At this point, many factors such as dependence on an internet connection, decision mechanisms, use of bandwidth, query-response times and remote updating become important.

Another important factor related to these systems is cost. Today, IT companies offer solutions with software and hardware add-on packages that are priced in addition to basic hardware prices instead of developing ready-to-use platform products due to commercial concerns. In addition, data analysis services such as anomaly detection and forecasting are sold with monthly or annual charges. All these costs are frighteningly high for end users who want to provide multi-point management, such as telecommunications companies or banks.

In this thesis, which chooses these two main problems as a goal, a micro server platform that is customized for indoor monitoring and control, based on fog computing and single board computer is designed and developed.

Thanks to fog computing, data storage, processing and monitoring operations on the local network without the need for internet connection have been realized as distributed. In addition, system performance and manageability have been increased with the lightweight operating system and containerization. In order to reduce hardware costs, the platform was developed based on single board computer and all hardware was designed as modular. Modular hardware will allow the platform to be edited according to the needs of the user and to reduce costs.

In order to reduce software costs, all the software needed for indoor monitoring and control such as on-premise database, dashboard, system metric agent have been developed or open source applications have been used.

After the completion of the whole system, a 15-day test was carried out with a real indoor monitoring and control scenario. Approximately 2.8 million data from 11 peripherals, including various sensors, detectors, analyzers and cameras, were recorded in the platform database with sender ID and time information. When this dataset (nearly 172 megabytes) was checked, no data loss or abnormal delay was detected.

In the test performed on the dashboard of the platform for remote control operations, it was found that the relay module reacts with a delay of up to 35 milliseconds. This delay includes the animation of opening and closing the button widget on the dashboard and saving this operation to the database.

In addition, the application developed in the Python software language has performed demo actions by checking the collected data at the specified interval (5 seconds). These actions are to control the devices connected to the relay module or to an external smart socket (Vera Smart Plug), and to give an alarm warning on the dashboard when measurements exceeding the specified limit values are detected.

According to the test results of the micro server platform which is customized for the indoor monitoring and control which is the output of the thesis, it is seen that it is sufficient to operate in this field in hardware and software. Cost studies after the completion of the tests showed that the amount spent on for the platform was approximately \$ 125. Considering the alternatives, it was concluded that the cost level meets the expectations.

In this study, pre-made equipment such as USB hub, converters or relay card is used to save time. In the future, if these equipments are manufactured in an integrated design, both performance and cost-efficiency can be achieved. In addition, the hardware and software of this developed platform can be considered as milestone, and the optimized applications such as machine learning-based data analysis, forecasting or cooperative data processing will be improved and the productivity of IIoT applications can be increased significantly.

REFERENCES

- Ai, Y., Peng, M. and Zhang, K. (2018). Edge computing technologies for the Internet of Things: a primer. *Digital Communications and Networks*, 4(2), pp.77-86.
- Chiang, M. and Zhang, T. (2016). Fog and IoT: An Overview of Research Opportunities. *IEEE Internet of Things Journal*, 3(6), pp.854-864.
- Cisco.com. (2019). [online] Available at: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf [Accessed 22 Feb. 2019].
- Dietpi.com. (2019). DietPi - Lightweight justice for your SBC. [online] Available at: <https://dietpi.com/> [Accessed 05 Jul. 2019].
- Docs.influxdata.com. (2019). *InfluxDB 1.7 documentation | InfluxData Documentation*. [online] Available at: <https://docs.influxdata.com/influxdb/v1.7/> [Accessed 13 Jul. 2019].
- Grafana Labs Blog. (2019). Grafana documentation. [online] Available at: <https://grafana.com/docs/> [Accessed 15 Jul. 2019].
- Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), pp.1645-1660.
- Kang, B., Kim, D. and Choo, H. (2017). Internet of Everything: A Large-Scale Autonomic IoT Gateway. *IEEE Transactions on Multi-Scale Computing Systems*, 3(3), pp.206-214.
- Lertsinsrubtavee, A., Ali, A., Molina-Jimenez, C., Sathiaseelan, A. and Crowcroft, J. (2017). PiCasso: A lightweight edge computing platform. *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*.
- Mbohwa, C. and Kumar Sahu, A. (2018). Performance Assessment of Companies Under IIoT Architectures: Application of Grey Relational Analysis Technique. *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*.
- Networktechinc.com. (2019). Environment Monitoring System Server Room Temperature IP Sensor Alert. [online] Available at: <http://www.networktechinc.com/environment-monitor-5d.html#tab-7> [Accessed 13 Feb. 2019].
- Ni, Y., Miao, F., Liu, J. and Chai, J. (2013). Implementation of Wireless Gateway for Smart Home. *Communications and Network*, 05(01), pp.16-20.
- Nodered.org. (2019). Running on Raspberry Pi : Node-RED. [online] Available at: <https://nodered.org/docs/getting-started/raspberrypi> [Accessed 11 Jul. 2019].

- Pahl, C., Helmer, S., Miori, L., Sanin, J. and Lee, B. (2016). A Container-Based Edge Cloud PaaS Architecture Based on Raspberry Pi Clusters. *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*.
- Perera, C., Qin, Y., Estrella, J., Reiff-Marganiec, S. and Vasilakos, A. (2017). Fog Computing for Sustainable Smart Cities. *ACM Computing Surveys*, 50(3), pp.1-43.
- Raspberrypi.org. (2019). [online] Available at: https://www.raspberrypi.org/magpi-issues/Beginners_Guide_v1.pdf [Accessed 02 Jun. 2019].
- Varghese, B., Wang, N., Barbhuiya, S., Kilpatrick, P. and Nikolopoulos, D. (2016). Challenges and Opportunities in Edge Computing. *2016 IEEE International Conference on Smart Cloud (SmartCloud)*.
- Verma, P. and Sood, S. (2018). Fog Assisted-IoT Enabled Patient Health Monitoring in Smart Homes. *IEEE Internet of Things Journal*, 5(3), pp.1789-1796.

APPENDIX 1 – Dashboard Listener Python Script

```
from influxdb import InfluxDBClient

import time

import pandas as pd

import os

import serial

import time

import sys

import threading

def call_smart_plug_off():
    os.system("python /root/services/smartPlug.py 0")
def call_smart_plug_on():
    os.system("python /root/services/smartPlug.py 1")
def call_siren_1s():
    os.system("python /root/services/siren1s.py")
client = InfluxDBClient(host='192.168.10.1', port=8086)
door_armed = False
door_temp_state = False
temp_state = 0
while True:
    ##SMART PLUG OPERATION
    client.switch_database('indoor_params')
    result = client.query('SELECT last("value") FROM
    "SmartPlug" ')
    plug_state =
    list(result.get_points(measurement='SmartPlug'))
    last_plug_state = int((plug_state[0]["last"]))
    time.sleep(0.2)
    if temp_state != last_plug_state:
        if last_plug_state == 0:
            t =
            threading.Thread(target=call_smart_plug_off).start()
            temp_state = 0
```

```

        elif last_plug_state == 1:
            t =
threading.Thread(target=call_smart_plug_on).start()
            temp_state = 1

##FLOOD ALARM

    client.switch_database('io_module')
    result = client.query('SELECT last("value") FROM "ch1"')
    flood = list(result.get_points(measurement='ch1'))
    last_flood = int((flood[0]["last"]))
    time.sleep(0.2)

##FLAME ALARM

    client.switch_database('io_module')
    result = client.query('SELECT last("value") FROM "ch2"')
    flame = list(result.get_points(measurement='ch2'))
    last_flame = int((flame[0]["last"]))
    time.sleep(0.2)

##MOTION ALARM

    client.switch_database('io_module')
    result = client.query('SELECT last("value") FROM "ch3"')
    motion = list(result.get_points(measurement='ch3'))
    last_motion = int((motion[0]["last"]))
    time.sleep(0.2)

##DOOR ALARM

    client.switch_database('indoor_params')
    result = client.query('SELECT last("value") FROM
"doorarmed"')
    armed = list(result.get_points(measurement='doorarmed'))
    last_armed = ((armed[0]["last"]))
    time.sleep(0.2)

    client.switch_database('lora_params')
    result = client.query('SELECT last("value") FROM

```

```

"doorsensor"' )

    door = list(result.get_points(measurement='doorsensor'))
    last_door_state = ((door[0]["last"]))
    time.sleep(0.2)

    if (door_temp_state != last_door_state)and(last_armed !=
False):

        if last_door_state == 1:
            door_temp_state = 1
        elif last_door_state == 0:
            t =
threading.Thread(target=call_siren_1s).start()
            door_temp_state = 0

##INDOOR TEMPERATURE ALARM
    client.switch_database('indoor_params')
    result = client.query('SELECT last("temp") FROM
"Multisensor"' )
    indoor_temp =
list(result.get_points(measurement='Multisensor'))
    last_indoor_temp = float((indoor_temp[0]["last"]))
    time.sleep(0.2)

##ENERGY METER VOLTAGE ALARM
    client.switch_database('indoor_params')
    result = client.query('SELECT last("voltL1") FROM
"EnergyMeter"' )
    voltage = list(result.get_points(measurement='EnergyMeter'))
    last_voltage = (float((voltage[0]["last"]))/10)
    time.sleep(0.2)

```



APPENDIX 2 – Energy Analyzer Parser Python Script

```
import time

import minimalmodbus

from influxdb import InfluxDBClient

import serial.tools.list_ports

client = InfluxDBClient('192.168.10.1', 8086, 'root', 'root',
                        'indoor_params')

comlist = serial.tools.list_ports.comports()

comport = " "

def modbus_request(device_addr):

    instrument = minimalmodbus.Instrument(comport,
    device_addr) # port name, slave address (in decimal)

    instrument.serial.baudrate = 9600

    instrument.serial.timeout = 3

    instrument.mode = minimalmodbus.MODE_RTU

    #instrument.debug = True

    parameters = {'voltL1':0}

    parameters['voltL1'] = instrument.read_long(0) #
    Registernumber, number of decimals

    return parameters

if __name__ == "__main__":

    i = 0

    for element in comlist:

        comport = (str(comlist[i])[0:12])

        try:

            (modbus_request(1))

        except:

            i+=1

    while True:

        parameters=modbus_request(1)

        json_body = [

            {

                "measurement": "EnergyMeter",
```

```
    "tags": {
      "valueOf": "parameters",
    },
    "fields": {
      "voltL1": parameters["voltL1"]
    }
  }
]

client.write_points(json_body)
time.sleep(10)
```



APPENDIX 3 – Wireless Network Module Listener Python Script

```
import serial
import RPi.GPIO as GPIO
import time
import sys
from influxdb import InfluxDBClient

if __name__ == '__main__':
    ser = serial.Serial('/dev/ttyACM0',115200)

    json_body = [{"measurement": " ", "tags": {"parameter": " "},
    }, {"fields": {"value": " "}}]

    client = InfluxDBClient('192.168.10.1', 8086, 'root',
    'root', 'lora_params')

    while True:

        data= (ser.readline())
        print(data)
        pre = data[0:3]

        if ( pre == "Tem"):

            try:

                data = float(data[3:8])

                json_body = [{"measurement":
"multisensor", "tags": {"parameter": "temperature", }, "fields":
{"value": data}}]

                client.write_points(json_body)

            except:

                pass

        elif pre == "Hum":

            try:

                data = float(data[3:8])

                json_body = [{"measurement":
"multisensor", "tags": {"parameter": "humidity", }, "fields": {"value":
data}}]

                client.write_points(json_body)
```

```

        except:
            pass
    elif pre == "Air":
        try:
            data = float(data [3:len(data)])
            json_body = [{"measurement":
"multisensor","tags": {"parameter": "airquality",},"fields":
{"value": data}}]
            client.write_points(json_body)
        except:
            pass
    elif pre == "Dor":
        try:
            data = int(data [3:len(data)])
            json_body = [{"measurement":
"doorsensor","tags": {"parameter": "doorstate",},"fields": {"value":
data}}]
            client.write_points(json_body)
        except:
            pass

```