UNIVERSITY OF NINEVAH
COLLEGE OF ELECTRONICS ENGINEERING
COMPUTER AND INFORMATION ENG. DEPARTMENT

Investigation of Communication Networks and Infrastructure in the Internet of Everything

By

RAWIA TALAL AZEZ

M.Sc. Thesis

In

Computer and Information Engineering

Supervised by

Dr.Abdulbary Raouf Suleiman

Assistant Prof.

2021 A.D. **1442 A.H.**

Investigation of Communication Networks and Infrastructure in The Internet of Everything

A Thesis Submitted

By

Rawia Tala Azez

To

The Council of the College of Electronic Engineering

University of Ninevah

As a Partial Fulfillment of the Requirements

For the Degree of Master of Science

In

Computer and Information Engineering

Supervised by

Dr. Abdulbary Raouf Suleiman

Assistant Prof.

2021 A.D. **1442 A.H.**

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

سورة العلق

الأية (5)

Supervisor's Certification

I certify that the dissertation entitled (**Investigation of Communication Networks and Infrastructure in The Internet of Everything**) was prepared by **Rawia Talal Azez** under my supervision at the Department of Computer and Information Engineering, University of Ninevah, as a partial requirement for the Master of Science Degree in Computer and Information Engineering.

Signature:

Name: Dr.Abdulbarry Raouf Suleiman

Department of Computer and Information Engineering

Date: / /2021

Linguistic Advisor Certification

I certify that the linguistic evaluation of this thesis entitled **" Investigation of Communication Networks and Infrastructure In The Internet Of Everything"** was carried out by me and it is accepted linguistically and in expression.

Signature:

Name:

Date: / /2021

Post-Graduate Committee Certification

According to the recommendations presented by the supervisor of this dissertation and the linguistic reviewer, I nominate this dissertation to be forwarded to discussion.

Signature:

Name: Assistant Prof. Maan A. S. Al-Adwany

Department Head Certification

I certify that this dissertation was carried out in the Department of Computer and Information Engineering. I nominate it to be forwarded to discussion.

Signature:

Name: Assistant Prof. Maan A. S. Al-Adwany

Committee Certification

We the examining committee, certify that we have read this dissertation entitled (Investigation of Communication Networks and Infrastructure In The Internet Of Everything) and have examined the postgraduate student (Rawia Talal Azez) in its contents and that in our opinion; it meets the standards of a dissertation for the degree of Master of Science in Computer and Information Engineering.

Signature:

Name:

Head of committee

Date: / /2021

Signature:

Name:

Member

Date: / /2021

Signature:

Name:

Member

Date: / /2021

Signature:

Name: Dr. Abd Albarry Aulayman

Member and Supervisor

Date: / /2021

The college council, in its ………… meeting on / /2021, has decided to award the degree of Master of Science in Computer and Information Engineering to the candidate.

Signature:
Name:
Dean of the College
Date: / /2021
Signature:

Name:
Council registrar
Date: / /2021
Signature:

ACKNOWLEDGEMENTS

"Praise be to ALLAH, Lord of the whole creation"

I would like to express my sincere gratitude and thanks to my supervisor, ***Dr. Abdulbarry Raouf Suleiman*** for his continuous guidance, helpful suggestions and constant encouragement throughout this work.

Thanks are due to the Dean of the Electronics Engineering College for his valuable assistance. My appreciation is extended to the Head and all members of Computer and Information Engineering Department for their support and assistance.

I dedicate this thesis to the soul of my dear father and brother. Also i would like to extend my sincere appreciation to my mother , my husband, my father and mother in law, my lovely son, and my daughter for their encouragements, support and patience throughout the duration of my graduate study.

Researcher

Rawia Talal Azez

2020

Abstract

A competent and structured approach to improve the health and well-being of humanity is provided by the Internet of everything technology. One of the practical ways to provide better life quality to people is based on IoE. Development of applications like monitoring human health, intelligent management, and decision making in industry field, smart buildings, road traffic management, and so many other applications contribute to crystalize the concept of Smart City. This concept integrates all smart applications and intelligently managing their data to make better and faster decisions in order to make the dream of people true of luxury and comfortable life. Initiating such a smart city is the obsession of the world nowadays.

In this work, we investigated the infrastructure of the internet of everything concerning the communication technologies, devices, and protocols. The IoE architecture standard is still under development but most of the researchers adopt the three-layer architecture. To implement the IoE infrastructure a smart city prototype is designed including three platforms which are Healthcare, Environment, and Security.

To create such a prototype, we developed the codes of all devices and servers in addition we built our algorithms for intelligent and smart responses to the systems events. To extend the existing functionality of the servers and achieve the interactions between servers and cloud the Application Programming Interface (API) has been used. Python 3.8 programming language is adopted in the software work of our smart city porotype.

For enhancing the performance of our system the edge computing concept is adopted by extending cloud computing service to the edge of the

network so Fog Nodes (FN) have been added. The real-time data is transmitted by the IoT devices to the remote Fog computing node for real-time, visualization, processing, and analysis making real-time decisions for each platform in our smart city prototype. This step reduced the system latency and made systems reliable even if the connection with the cloud has some discontinuities.

The data generated from the smart city is fuzzy and has many issues like redundancy and duplicity so we use a data aggregation technique to enhance the performance of our proposed smart city prototype. The aggregated data generated from the smart city is uploaded to the Microsoft Azure cloud storage preparing this data to be ready for analyzing and data mining.

Cisco packet tracer 7.2 is used as an effective tool for designing and implementing the proposed smart city prototype with an easy to use GUI for all platforms.

TABLE OF CONTENTS

LIST OF FIGURES

List of Tables

List of Abbreviations

Abbreviation	Name
IoE	Internet of Everything
QoS	Quality of Service
IoV	Internet of Vehicles
IoT	Internet of Everything
SoS	System of Systems
P2P	People-to-People
M2M	Machine-to-Machine
M2P	Machine-to-People
PaaS	Process as a Service
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
MQTT	Message Queuing Telemetry Transport
XML	eXtensible Markup Language
RFID	Radio Frequency Identification
GSM	Global System for Mobile
LTE	Long-Term Evolution
LoRaWAN	Low Power Wide Area Network
6LowPan	Low-power Wireless Personal Area Networks
TCP	Transmission Control Protocol

FN	Fog nods
DM	Data Mining
IT	Information Technology
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
XaaS	Anything as a Service
NIST	National Institute of Standards and Technology
ML	Machine Learning
AI	Artificial Intelligence
CSV	Comma-Separated Values
SoS	System of Systems
API	Application Programming Interface
GUI	Graphical User Interface
IP	Internet Protocol
ISP	Internet Service Provider
LCD	Liquid Crystal Display
GHz	GigaHertz
CHCS	Cluster Healthcare Server
UDP	User Datagram Protocol
DHCP	Dynamic Host Configuration Protocol
PDU	Protocol Data Unit

SpO2	Peripheral oxygen saturation
4G	fourth Generation
CSCS	Cluster Security Control Server
IR	Infrared Radiation
GPS	Global Positioning System
SEM	Smart Environment Monitoring
CO	Carbon monoxide
CO2	Carbon Dioxide Detector
AC	Air Conditioner
CECS	Cluster Environment control server
ATM	Atmosphere

Chapter One

Introduction to the Internet of Everything

This chapter includes the definition, importance, and applications of the internet of everything (IoE), the related work, the statement of the problem, thesis objectives and structure, and finally, its contribution.

1.1 IoE Definitions and Applications

The 'Internet of Everything (IoE)' is considered the last phase of internet evolution, this phase links people, processes, data, and things translating information into activities that build new skills, richer interactions, and unparalleled opportunities [1]. It describes a world where trillions of intelligent devices have sensors to verify measure and estimate their positions, all connected over public or private networks that use specific protocols[2].

There is a wide range of applications that take the advantage of the IoE concept including helping in achieving public policy goals, smart building, smart grid, smart education, economic development, transportation, and social services as shown in figure (1.1). Flexibility and high scalability are the prosperities of the IoE Identity Platform to accommodate billions of IoT devices of all types. This is done by using secure protocols as core identity mechanisms, so all devices authenticate as they come online, prove their integrity, and securely communicate with other devices, services, and users[3].

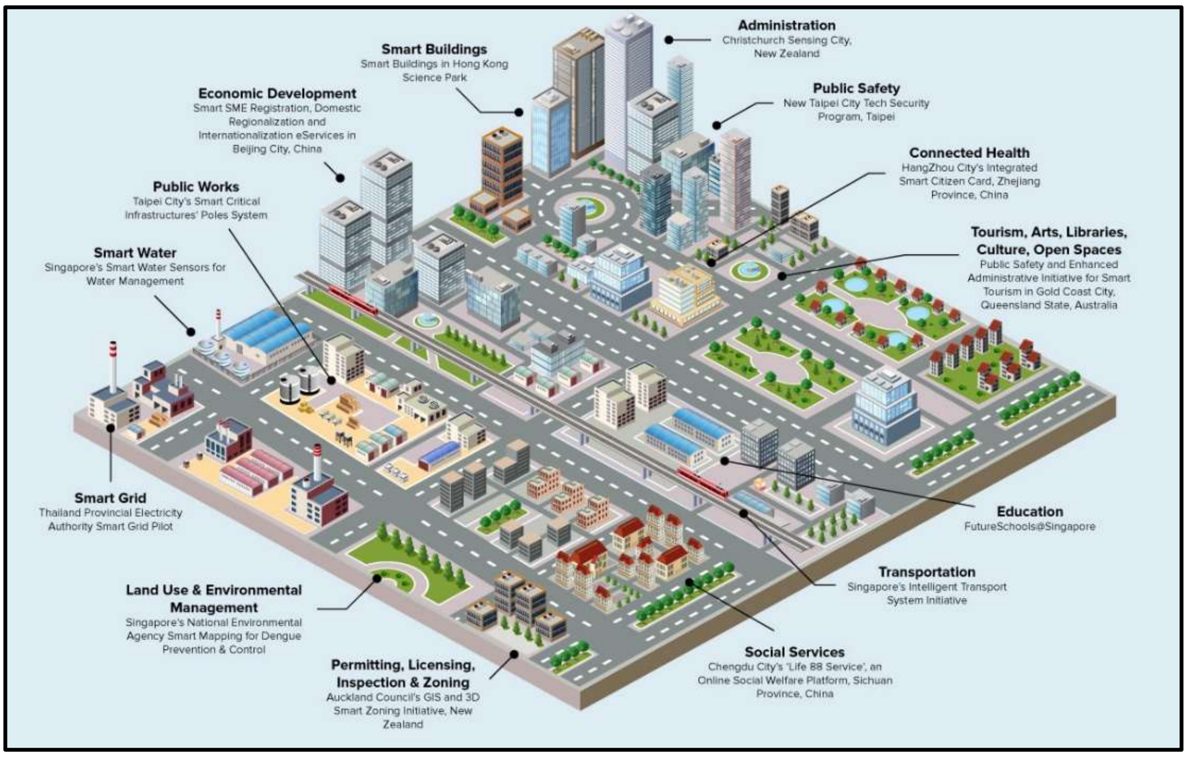

Smart Buildings
Smart Buildings in Hong Kong Science Park
Administration
Christchurch Sensing City, New Zealand
Economic Development
Smart SME Registration, Domestic Regionalization and Internationalization eServices in Beijing City, China
Public Safety
New Taipei City Tech Security Program, Taipei
Connected Health
HangZhou City's Integrated Smart Citizen Card, Zhejiang Province, China
Public Works
Taipei City's Smart Critical Infrastructures' Poles System
Smart Water
Singapore's Smart Water Sensors for Water Management
Tourism, Arts, Libraries, Culture, Open Spaces
Public Safety and Enhanced Administrative Initiative for Smart Tourism in Gold Coast City, Queensland State, Australia
Smart Grid
Thailand Provincial Electricity Authority Smart Grid Pilot
Education
FutureSchools@Singapore
Transportation
Singapore's Intelligent Transport System Initiative
Land Use & Environmental Management
Singapore's National Environmental Agency Smart Mapping for Dengue Prevention & Control
Social Services
Chengdu City's 'Life 88 Service', an Online Social Welfare Platform, Sichuan Province, China
Permitting, Licensing, Inspection & Zoning
Auckland Council's GIS and 3D Smart Zoning Initiative, New Zealand

Figure 1.1 Smart City Applications

Several governments starting from 2009 actively pursuing a smart city strategy like Amsterdam, Barcelona, Copenhagen, Dubai, London, and others. Smart cities are the integration of many IoE systems that uses information technologies to make more efficient use of physical infrastructures like roads, built environment, and other physical assets through artificial intelligence and data analytics in order to support a strong and healthy economic, social, cultural development. Also, take advantage of learn, adapt, and innovate and thereby respond more effectively and promptly to changing circumstances by improving the intelligence of the city[4]. Smart cities evolve towards a strong integration of all dimensions of human intelligence, collective intelligence, and also artificial intelligence within the city.

The intelligence of cities "resides in the increasingly effective combination of digital telecommunication networks (the nerves), ubiquitously embedded intelligence (the brains), sensors and tags (the sensory organs), and software (the knowledge and cognitive competence)[5]. Figure(1.2) show an example of smart city services.

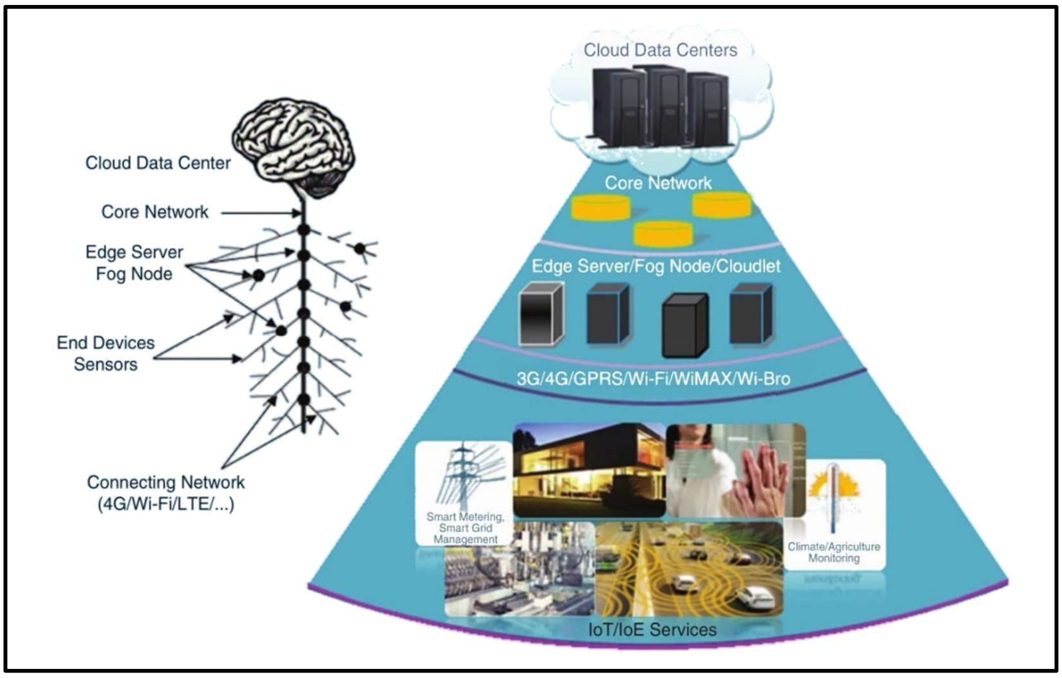


Figure 1.2 Analogy Between Nervous System and IoE

The architecture of IoE is being developed and enhanced to provide the user with the best services and efficient performance [3]. Remote cloud servers are used for storing and processing big data that have been collected from a large number of sensor nodes. Although of huge benefits of cloud computing such as the capability of large data storage volume and smart data analysis, still, many challenges exist in these systems concerning latency sensitivity issues, location awareness, and transmission of large data[6]. The traditional way of data processing centralization in the Cloud cannot satisfies most of the current IoE requirements like high availability and ubiquity with regards to storage and computing capabilities. Especially, with the growing number of smart objects and the amount of data produced effects on generating high network traffic which increases the latency, that degrading the Quality of Service (QoS) for several IoE applications like the Internet of Vehicles (IoV), industry, and healthcare.

The service latency is very critical and the lag may cause critical issues. To overcome the previously mentioned challenges, the paradigm of Fog computing has been proposed to create a distributed computing infrastructure closer to the network edge which performs easier tasks that require a quick response. This reduces the data burden on the network and enhances flexibility by allowing smart devices to operate when network connections to the cloud are lost. The security is also enhanced by keeping sensitive data beyond the edge where it is needed instead of being transported [7]. On the other hand, the rate of generated data is usually huge for processing and storing, therefore, applying aggregation to the data from various sites is an efficient technique. The fundamental purpose of the data aggregation strategy is to aggregate and collect the data packets in an effective manner to improve the network lifetime, energy consumption, data accuracy, and traffic bottleneck [8].

1.2 Literature review

In 2014, S.Abdelwahab et al. [9] made a survey on Cloud-assisted remote sensing for IoE by describing its benefits and capabilities then presenting its multilayer architecture. Finally, discussing the major design requirements and challenges.

In 2015, R.Balfour [10] Presented some technological capabilities that are critical to achieving the IoE, particularly for emergency managers including security, a global M2M standard, M2M applications, and Data Privacy and trust.

In 2016, P. Pena and et al. [11] propose a smart system where security and cognition are reactive modules. using a big-data centric modular architecture

In 2016, B.Ahlgren and et al. [12] developed a Green IoT system that includes heterogeneous sensors to collect data and incorporates with cloud computing technologies to get more interactive and responsive administration of a city in Uppsala, Sweden.

In 2017, M.Naas and J.Boukhobza [7] presented a data placement strategy for Fog infrastructures called iFogStor which takes the benefits of the heterogeneity and location of Fog nodes to minimize the overall latency of storing and retrieving data in the Fog.

In 2017, S.Clement et al. [13] proposed a reference architecture for the smart city based on Service Oriented Architecture (SOA) concepts; integrating IoT, Cloud, and Edge technologies with existing city infrastructure.

In 2018, C.Badii et al. [14] introduced a platform where sophisticated IoT applications for controlling city dashboards as well as IoT mobile applications. Particular attention is paid to the tools and solutions for monitoring communication performance and to perform the estimation of scalability of the IoT smart city infrastructure.

In 2018, M.Yang et al. [15] proposed a multifunctional data aggregation method with differential privacy. Machine learning is the base of proposed methods and can support a wide range of statistical aggregation functions.

In 2019, S.Muralidharan [16] build a container-based system in Seoul port of cloud-based monitoring system for IoT applications with low latency, reliable and secure communication with a strong focus on horizontal interoperability among various IoT applications.

In 2019, R.Mahmud and R.Buyya [17] discussed some examples of Fog environment scenarios and explained how to implement custom application placement in iFogSim to simulate Fog environment along with an IoT-enabled smart healthcare case study.

In 2019, P.Naranjo et al. [5] presented a smart city network architecture called Fog Computing Architecture Network (FOCAN). A multi-tier structure in which the applications are running on things with taking into consideration the latency and power through the smart city environment.

In 2019, M.Masoud et al. [18] attempted to demo the process of concluding significative data from sensors in smart devices, especially, smartphones. Also, different useful machine learning applications based on smartphones' sensors data are given.

In 2020, S.Ullo and G.Sinha [19], the authors have studied how the advancements in sensor technology, IoT, and machine learning methods make environment monitoring a really smart monitoring system.

1.3 Statement of the Problem

The technology of the Internet of Things and the Internet of everything are ones of the modern technologies that take a wide area in literature but appointing an IoE standard infrastructure is still in the research phase. Consequently, it is found that there is a need to investigate this issue in this work.

Implementing an IoE infrastructure by connecting a large number of smart devices produces a huge amount of raw data that is difficult to be managed directly by the cloud. This difficulty is an issue that needs a solution.

1.4 Thesis Objectives

- ▶ To Investigate a standard infrastructure of the internet of everything concerning the networks, devices, and protocols.
- ▶ Creating and implementing a smart city prototype including three platforms using the most powerful simulator.
- ▶ To develop the required programming and algorithms for networking and devices.
- ▶ To use the edge computing concept to extend cloud computing service to the edge devices.
- ▶ To develop a data aggregation algorithm to enhance the performance of our proposed smart city prototype
- ▶ Uploading the aggregated data generated from the smart city to cloud storage and prepare this data to be ready for analysis and data mining.
- ▶ To address the most effective challenge by trying to give reasons and solutions.

1.5 Contribution

The process of connecting billions of things to the internet and provide an interactive environment between them and people, also managing and processing the data produced is considered a very interesting subject called IoE. Throughout this work its dived deeply into the infrastructure and studied the protocols, the data flow, and the common architectures of IoE. Then implement this architecture by building a Smart City prototype consists of three systems: Healthcare monitoring, Environment Monitoring, and Security Control.

The accomplishment of such a prototype required developing the smart devices algorithms and networking devices programming to get the better performance of smart city platforms and make them flexible and extendable. The most critical issue in smart systems is the latency and real-time response to achieve this we used the edge computing concept to extend cloud computing service to the network edge and that improves the latency and real-time response.

The IoE generates a large amount of data each second collected by sensors and smart devices as well as the network data, all this raw data needed organizing and filtering before it is uploaded to the cloud, this process called data aggregation. An algorithm to aggregate data generated by our proposed smart city prototype was developed . The aggregated data is then uploaded to the Microsoft Azure HUB in order to store it in the Custom storage service.

1.6 Thesis structure

Chapter two introduces the theoretical background of the Internet of Everything: concepts, Architecture, services, advantages, and applications.

Chapter three of this thesis contains an introduction to our simulated smart city prototype then discusses in detail the IoT layer contents and the communication process between them. The implementation of the relation with the Edge networking sublayer is also described in this chapter. Finally explaining the algorithms of the cluster server for each of our platforms.

Chapter four explains the implementation of the top-level sublayer of the networking layer and introduces the proposed algorithm of the Centralized server. Furthermore, discuss the contribution of the aggregation process in reducing the amount of data uploaded to the cloud with results. Finally, this chapter clarifies the process of submission to the Microsoft Azure cloud and how to manage this data and rout it between cloud services.

Chapter five includes the conclusions of this work in addition to the all obstacles of this scientific research and future works.

Chapter Two

Background of IoE Infrastructure

In this chapter, we will explain the theoretical background of the internet of everything (IoE)

2.1 Introduction

The internet has developed in many distinct phases. Each phase has a more profound effect on business and society than the previous one. Nowadays, we are in the world digitizing phase which connects not only the smart devices but also connects everything in the world as people, data, and process to originate the key concept of the internet of everything (IoE)[1]. The huge growth in smart devices industry like smartphones, smart sensors, smart medical devices, and so on, helps the "Internet of Everything" (IoE) concept to explore and expand. It describes a world where trillions of smart devices have sensors for verifying measures and estimating their position and then turning information into actions. That enhances our daily lives by creating new capabilities, worthy experiences, and unprecedented opportunities [20].

A continuous increase in the urban population fatigues the limited resources of local servers so the use of cloud computing service becomes necessary to deal with the large amounts of data and to analyze it [21]. This Cloud-centric approach satisfies the requirements of many IoE applications like computing and storage capabilities, ubiquity, and high availability. but for applications that latency service is very critical and the least delay may cause critical issues such as the Internet of Vehicles (IoV), electronic healthcare, and smart industry the paradigm of Fog computing has been proposed that provides a hierarchical and geographically distributed architecture to store and process data in network equipment located between smart objects and Cloud data centers[7].

The connection between the internet of everything four pillars thing, people, data, and process needs a reference architecture and while the IoE is still in its infancy, its future structure and ingredients must be further studied and standardized and require the support of many applications, some of which presently exist and others that will be developed in the future [3].

Smart cities are the new goal for the integrated System of Systems (SoS). The computational infrastructure for a smart city will be based on a combination of distributed computing paradigms, enabling the use of low-power IoT devices, Cloud computing virtualization, and the use of localized processing with Edge computing[13].

To establish a smart IoE system many technical requirements should be taken into consideration such as Standardizations to find out what technology is required to allow these systems to communicate with other systems, also check the scalability and Security to verify the best services and applications need to be installed to simplify the management of these updated systems. Programming and Data processing also takes big considerations by the smart system developer to make sure that the system has the capability of communicating with devices and can forwarding data to the Cloud for processing or fog computing when data needs to be processed closer to the source.

2.2 Integration of internet of everything:

The IoE integrates four pillars that lead to making networked connections more relevant and valuable more than ever before: people, process, data, and things. The information from these connections leads to decisions and actions that create new capabilities, richer experiences, and unprecedented economic opportunity for individuals, businesses, and countries.

The interactions between the elements in the four pillars create a wealth of new information. The pillars interact in a way that establishes three main connections in the IoE environment: people communicate with people (P2P), machines communicate with people (M2P), and machines communicate with machines (M2M)[1][22]. Figure (2.1) shows the IoE four pillars

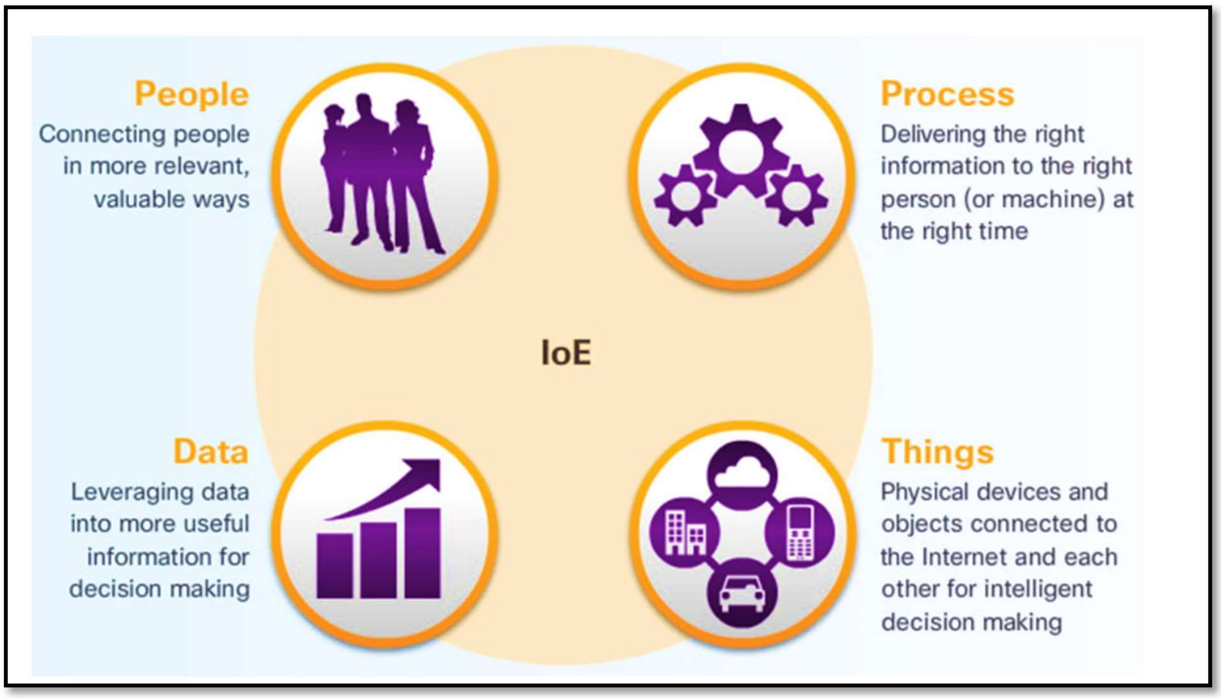

People
Connecting people in more relevant, valuable ways
Process
Delivering the right information to the right person (or machine) at the right time
IoE
Data
Leveraging data into more useful information for decision making
Things
Physical devices and objects connected to the Internet and each other for intelligent decision making

Figure 2.1 IoE Four Pillars

2.2.1 Things pillar

This pillar includes all types of objects. Cisco predicts that 99% of physical objects one day be connected to the internet. These devices have embedded technologies to communicate with the external environment and internal servers. The intelligent devices have network capabilities so it can communicate over a secure, reliable, and on available network platform [1].

On the IoT embedded devices, the data obtained may be completely or partially processed. The type of on-board processing procedure depends on the IoT application. However, data compression methods, signal processing, feature extraction, and classification are primarily included in most smart devices. The low-power IoT devices have constraints in capabilities of computation and memory limitations, so the need to optimizing the on-board processing has appeared. So, to make the on-board processing efficient and affordable both software and hardware optimizations are needed. Three main functions in the proposed techniques of an IoT device must be considered which are acquisition, computation, and communication. Figure (2.2) shows the general structure for IoT embedded devices.

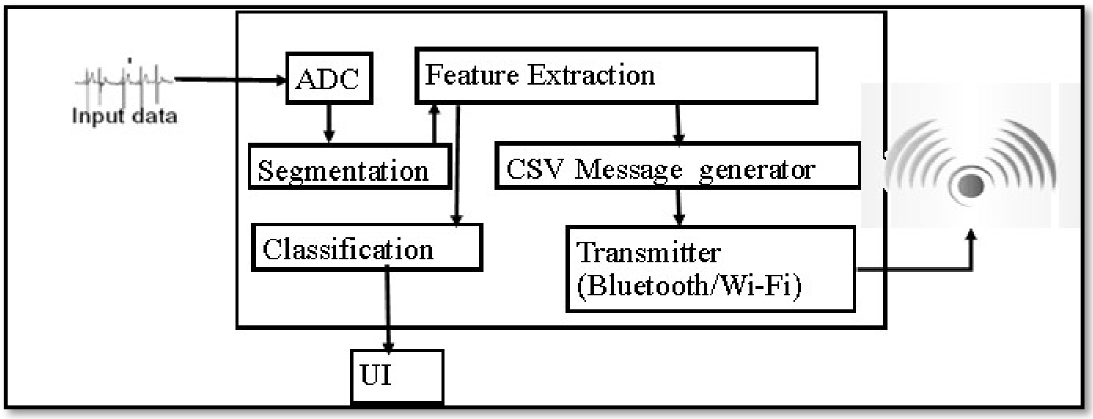


Figure 2.2 IoT Embedded Device

2.2.2 Data pillar

For all computing systems, data is the core component because the main goal of them is to process and transmit data. a concerning issue about data is the volume since a huge number of data is produced by humans every day. These data are stored in local servers, centralized storage, or distributed storages where the last two provide the ability of remote access from multiple devices. In smart systems, the data passes through four non-overlapping phases: Data Collection, Data Pre-Processing, Data Mining, and Data Post-processing. These four phases can be organized by data management centers[11][23]. Figure (2.3) contains a brief explanation of each data phase.

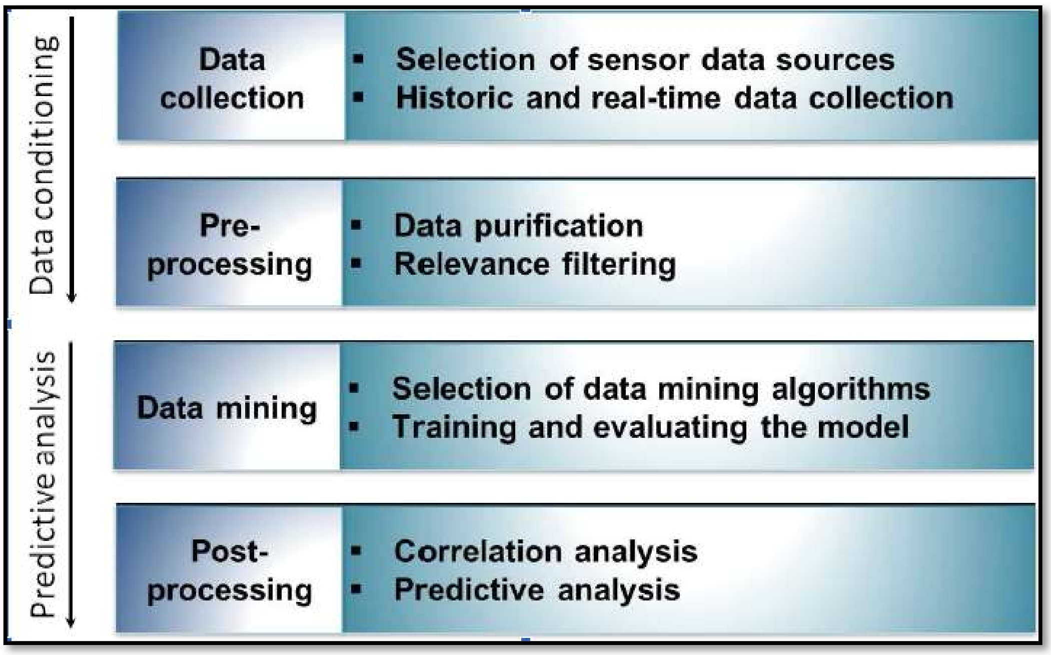


Figure 2.3 IoE Data processing Phases

2.2.2.1 Data collection phase

Smart systems data is collected from sensors, smart devices, social media, and other sources.

2.2.2.2 Data Pre-processing phase

In this phase, the system utilizes computational technologies like edge computing services. In edge computing, data processing is distributed on edge nodes which have less computation power than what is available to cloud servers. So, the computation tasks need to be assigned to several edge nodes to meet the same demands by offloading the computing and storage to the edge of the networks. Edge nodes also communicate with the aggregator nodes to avoid sending redundant data in addition to store collected data in cloud storage[24].

2.2.2.3 Data Mining phase

There is a need to store the processed data in a database. Varied types of data like metadata, historical data, and real-time data are stored in a distributed database. This PaaS service is used to process these types of data because it provides tools to access the data when a user requires it.

2.2.2.4 Data post-processing phase:

The applications of system monitoring have a notification service to the user of an event. Also, the applications of database management provide the user with accessing and requesting for specific data from the system. In both systems, the users' requests will be processed through the system and queried through the database. If requested data is not available in the cloud, an alert will be sent to data sources to fetch data needed by users.

2.2.3 People pillar

People are interacting as data producers and consumers to improve their well-being and satisfy human needs. The data generated from connecting people-to-people (P2P), machine-to-people (M2P), or machine-to-machine (M2M), can be used to enhance the value for people.

2.2.4 Process as a pillar

The Processes play an important role in how the other pillars work with each other to evaluate the value of the connected world of IoE. By merging machine-to-machine (M2M), machine-to-people (M2P), and people-to-people (P2P) ties, the IoE puts them all together, as seen in figure (2.4).

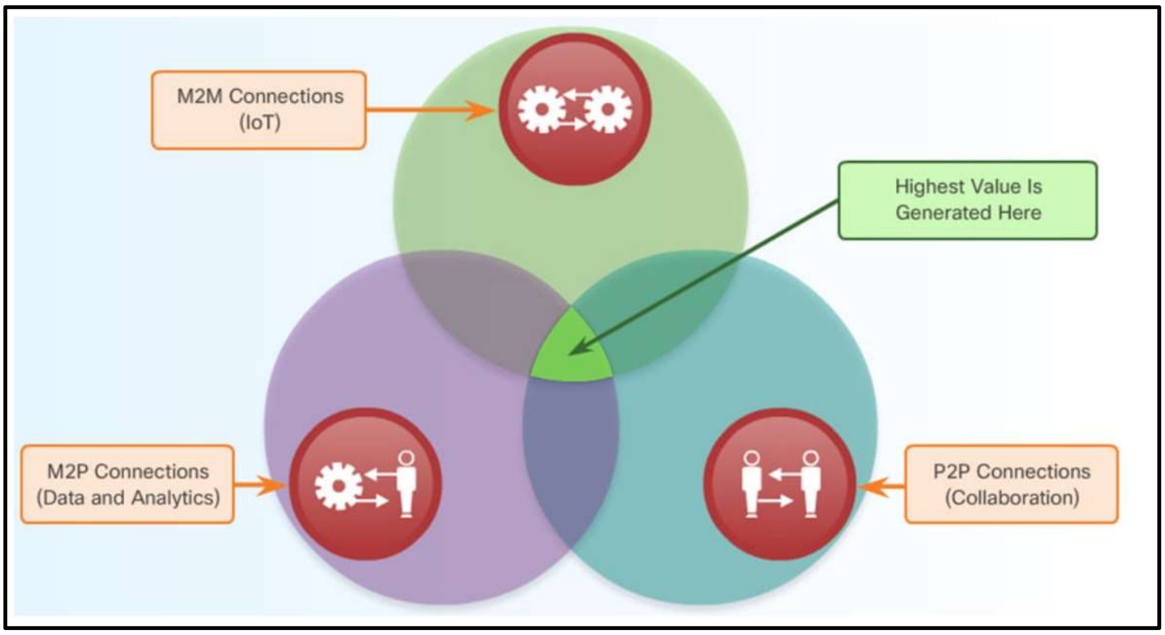


Figure 2.4 IoE Pillars Connections

(M2M) connections happen when data is transferred by one device or 'thing' to another via a network. Sensors, robots, computers, and mobile devices are considered as devices. These M2M connections are often referred to as the Internet of Things.

(M2P) connections occur when information is transferred between a machine and a person. Whether a person gets information from a database or conducts a complex analysis, this is an M2P connection. These M2P connections facilitate the movement, manipulation, and reporting of data from machines to help people make informed judgments. The actions that people take based on their informed judgments complete an IoE feedback loop.

(P2P) connections occur when information is transferred from one person to another. Increasingly, P2P connections happen through video, mobile devices, and social networks. These P2P connections are often called Collaboration.

Implementing an IoE solution using M2M, M2P and P2P connections provides organizations and individuals with actionable insights and easy automation.

2.3 Internet of everything Architecture and protocols

The standard approach to understanding an IoE framework is to picture every device being attached to the global network, enabling everyone to have access to every device. For simpler and faster communication between these objects, and for the potentiality to manage them, some models and criteria have been presented that have the capability to receive and present a special service automatically.

The Fog computing cloud base architecture consists of three main layers, namely, the IoT, the communication, and the cloud computing layers[1][7] [25]. Each one of the three fog-computing main layers has its specific functionality. The IoT layer consists of the sensors and smart devices that are responsible for collecting persistent data from objects. The communication layer is composed of two sub-layers, the first one is the Edge computing sub-layer includes the communication devices like switches and routers also include the fog nodes that collect the data from

IoT and control actuators with a real-time response. The second is the aggregation sub-layer which collects the data from the fog nodes and aggregates it then reorganizes it in JSON format to make it suitable to store in the cloud warehouse. Finally, the third layer of the Fog computing architecture is the cloud computing layer where the data warehouses exist, and where the service of the data mining and smart systems analysis and monitoring is applied on the big data. Figure (2.5) shows this structure.

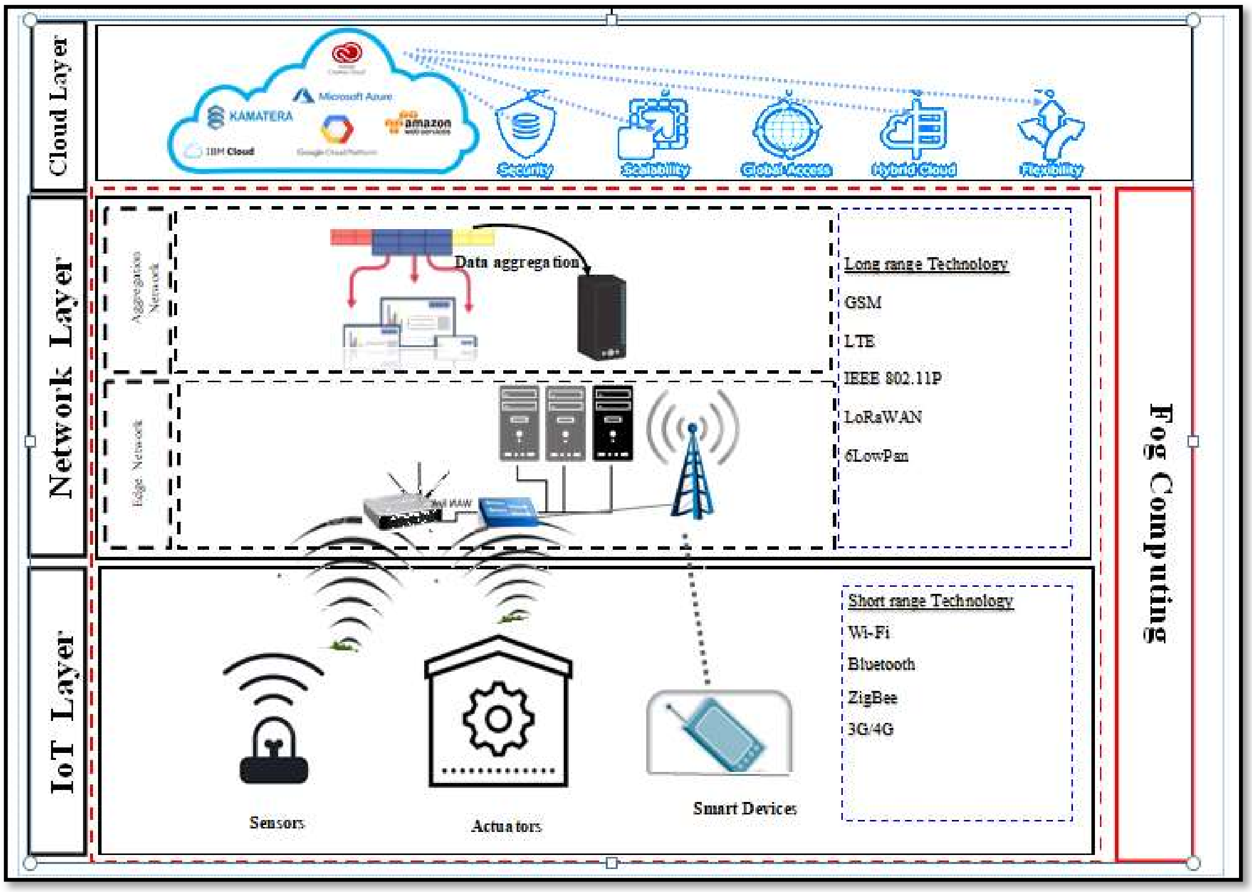


Figure 2.5 Fog Computing IoE Architecture

After verifying the structure, it is time to choose what protocol to be used for going forward in creating our custom IoT platform. While there are two major protocols HTTP and MQTT, both are very different in operation and have their pros and cons. Table (2.1) explains the two protocols differences [26].

Table 2.1 comparison between IoT protocols

	MQTT	HTTP
Design orientation	Data-Centric	Document centric
Complexity	Publish/Subscribe	Request/Respond
Message Size	Simple	More Complex
Service Level	Small, with a compact binary header just two bytes in size	Larger, partly because status detail is text-based
Extra Libraries	Libraries for C (30 KB) and Java (100 KB)	Depends on application (JSON,). But typically not small
Data Distribution	Support 1 to zero, 1 to 1, and 1 to n	1 to 1 only

Fog computing architecture encapsulates all physical objects, machines, and anything that is equipped with computing, storage, networking, sensing, and/or actuating resources, and that can be connected to and be part of the Internet [17]. Major functions of fog computing are to provide[27]:

1) Heterogeneous networking and communication infrastructures

to connect billions of things.

2) Unique identification of all things

3) Data aggregation points to serve as sensing clusters.

Programming is at the core of all computing technologies in the IoE. While sensors measure a physical property and forward that information across the network and actuators perform actions based on a received signal, then a controller must be programmed with a set of instructions to receive that data and decide if it should be processed and relay that data to another device. There are many different computer languages used to program these controllers, for example, C++, Java, Python, and others.

2.3.1 Thing layer

It is the bottommost layer that encompasses smart mobile or fixed end-users' objects such as sensors, RFID, actuators, smart devices, and other Components. These objects communicate with each other using the networking layer services as well as to connect with IoE services implemented in both network and Cloud layers.

The IoT architecture must assure its operations, which bridges the gap between the virtual and the physical worlds. Infrastructure devices in IoT are primarily responsible for transferring data between the controllers and other smart devices, and they provide a variety of services including:

- Wireless and wired connectivity to connect the individual end devices to the network, and can connect multiple individual networks to form an internetwork
- Quality of service queuing
- High availability
- Secure transfer.

2.3.2 Networking layer

The communication between smart devices, sensors, and actuators is managed by this layer also it is an embodiment of the connectivity with the cloud. There are several wireless network protocols available today. The characteristics of these protocols vary greatly. Figure (2.6) shows a visual representation of a few common wireless protocols and where these protocols fit in the classification spectrum.

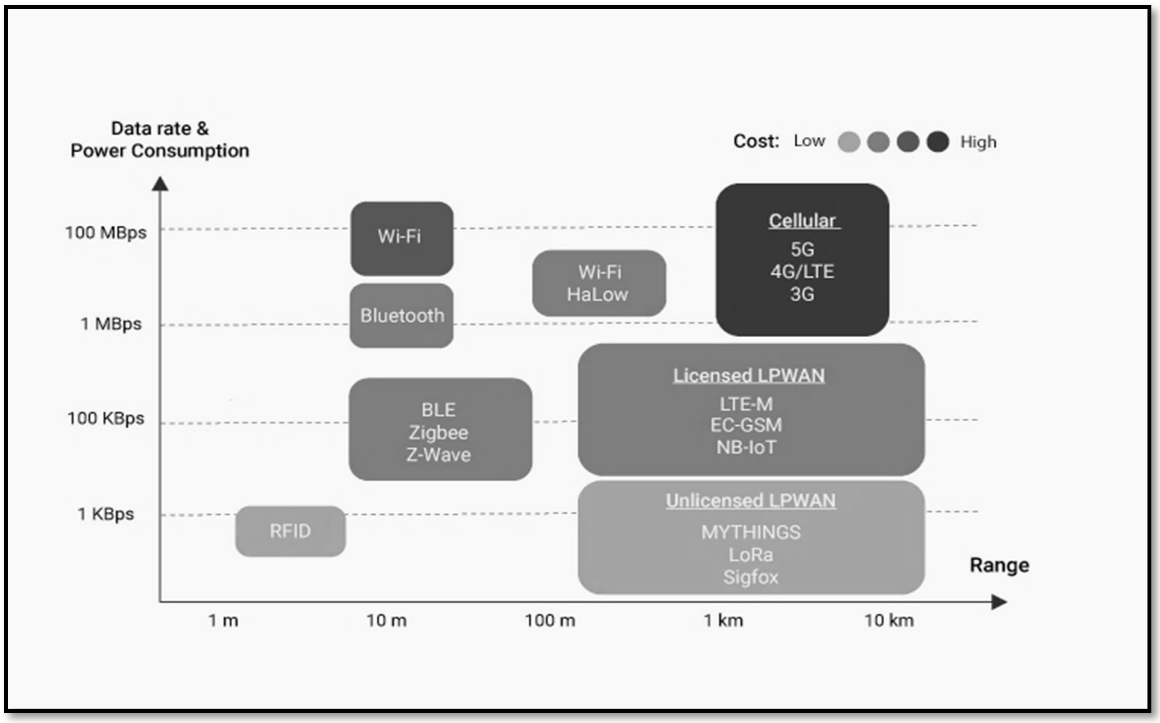


Figure 2.6 Common wireless communication technologies

The long-range communication technologies are used in this layer like GSM, LTE, IEEE 802.11P, LoRaWAN, 6LowPan. Furthermore, which has extra features such as smart analyzing and data aggregation. This layer consists of two sub-layers to be explained below.

2.3.2.1 Edge Networking sublayer

This sublayer consists of communication devices like routers, switches, wireless routers, and gateways. It supports both wired and wireless technology using the TCP protocol to associate the connection.

The infrastructure devices or intermediary devices managing the flowing of data through the network. They use the destination address in tandem with network interconnection information to decide how messages should be sent through the network. Figure (2.7) shows the infrastructure devices.

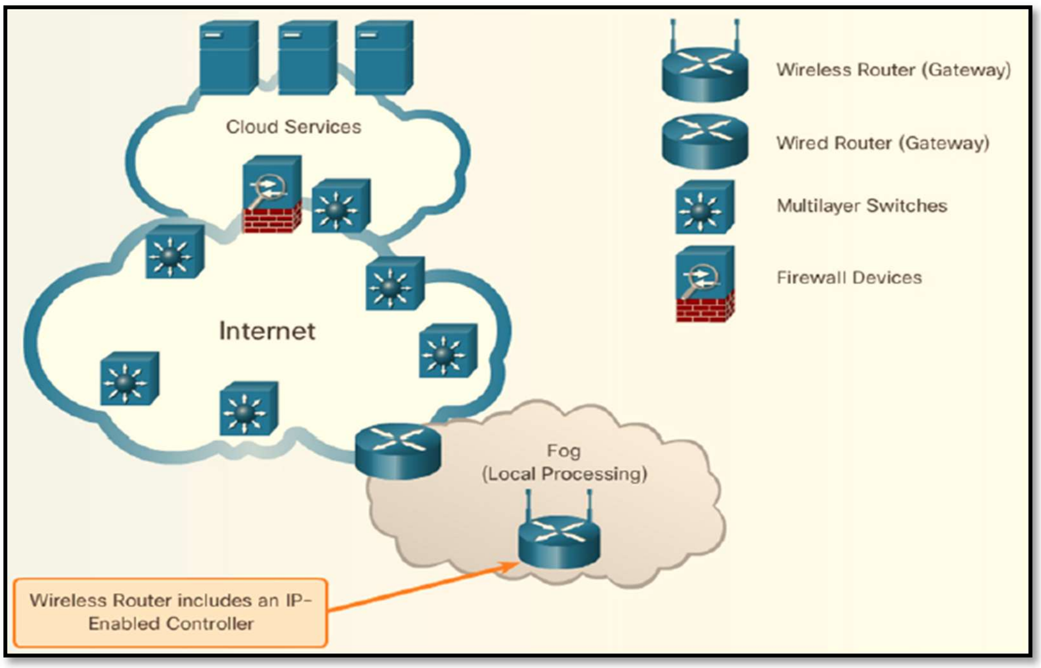


Figure 2.7 Infrastructure and intermediary devices

The basic trait of an IoE system is the ability to achieve an instant understanding of its environment, which strongly depends on the efficiency of data processing. In many practical IoE systems, the nature of the monitoring or control tasks set by involving machine learning in the forms of a state classifier or object detector. This processing can be performed either with a single computing node having spot access to all the sensors data, such as cloud computing, or the processing workload has to be reorganized so that it could be adaptively divided among the nodes in the network which is called Fog nods (FN). FN collects information from sensors and consolidates the information received and forwards the real-time smart decisions to change the status of the actuators using the TCP/IP protocol suite. Small servers or microcontrollers programmed with python are used to achieve this goal. Figure (2.8) shows how FN interacts with sensors and actuators[28].

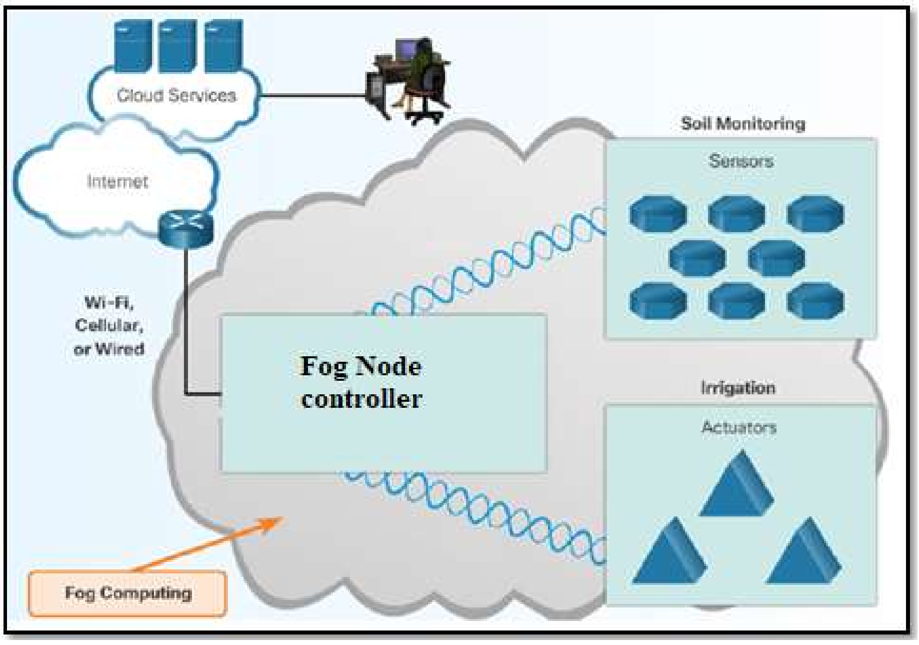


Figure 2.8 Fog Nodes interactions with sensors and actuators

Using edge distributed processing represented by Fog Nodes allows us to achieve the following guiding objectives[29]:

- Minimize the amount of traffic to avoid congesting shared wireless resources and violating requirements on processing latency.
- Minimize the induced processing load on the sensors and intermediate processing nodes to avoid depletion of their computational resources.
- Minimize the loss of information in the intermediate stages of processing resulting in traffic reduction, to avoid deterioration of the system's decision-making quality

2.3.2.2 Aggregation Networking Sublayer

Data aggregation is one of the influential techniques in the elimination of data redundancy and improvement of energy efficiency and the efficient data aggregation technique can reduce network traffic [29]. Data aggregation is done by the integration of incoming packets from multiple sources without processing them and sending packets. Computational Intelligence was used for pre-processing and filtering data in order to conserve bandwidth and processing power. Only relevant information is then transmitted to the cloud and stored, analyzed, and visualized there[23]. To evaluate the performance of the network it is required that environmental parameters are calculated such as data accuracy, delay, and bandwidth[29].

The important state-of-the-art data count aggregation mechanisms, as well as their differences, benefits, and weaknesses are divided into three categories, including tree-based, cluster-based, and centralized aggregation mechanisms for IoE[8].

- **Tree-based mechanism:** sensor nodes are organized into a tree where the aggregation is performed in intermediate nodes along with the network and aggregated data is transmitted to the root node.
- **Cluster-based mechanism:** sensors can send data to a nearby cluster or aggregator which aggregate the data from all sensors and transmit the short digest to the sink
- **Centralized base mechanism**: all sensors transmit their data packets via the shortest possible path to the central node and this node performs aggregation of data comes from all node and the result of this aggregation process will be a single message.

The format of data generated in real life is not suitable for data mining (DM) algorithms, also the quality of it is low because different sensors capture data in different formats. Therefore, using data cleaning is a critical step towards obtaining positive results. Filtering is another important preprocessing step to get good performance of IoE applications. Data deduplication, outlier, entity resolution, and feature selection are some important preprocessing steps. Feature selection will mean selecting the observation points that are used as input for the DM algorithms[23].

2.3.3 Cloud layer

Cloud computing provides shared computing resources and data on demand, for a distributed environment. Cloud computing build on multiple data centers that include multiple domains and geographical areas [9]. Nowadays, organizations required a dynamic Information Technology (IT) infrastructure because of that they shift to cloud computing due to its features as scalability, accessibility, and pay-per-use features. The most common services provided by the cloud are known as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), all of which are heading towards Anything as a Service (XaaS). However, data generated from billions of sensors referred to as big data, cannot be transferred and processed in the cloud directly it needs preprocessing management[27]. Cloud computing uses a shared pool of computing resources (e.g., networks, servers, storage, applications, and services) to provide on-demand network access. Using virtualization in data center environments, Cloud computing can be rapidly scaled with minimal management and effort. The National Institute of Standards and Technology (NIST) has defined four types of cloud deployment models: private, public, community, hybrid. The IoE requires a variety of cloud models[1][30]. Table (1) shows a brief detail about these services.

Table 2.2 Cloud Computing Services

Module	Definition	Managed by	Security level
Public	Public clouds are easily available from Google, Amazon, Microsoft, etc. Public cloud provides infrastructure and services to the public or any organization. Resources are shared by hundreds or thousands of people	Cloud service providers	Low
Private	works for a defined organization or business, e.g. cloud for a specific organization.	Many organizations or cloud service providers.	High
Community	the services and infrastructure are provided to organizations with similar interests.	Many organizations or cloud service providers.	High
Hybrid	Is a mixture of a private and public cloud. Though the clouds are mixed up, still each has its identity and therefore aiding multiple deployments.	Public and organization.	Medium

There are common cloud services for IoE [31] like storage, Big Query, machine learning (ML) techniques, and others. ML techniques which are a branch of Artificial Intelligence (AI) are well suited for handling data stream fuzziness and can be adapted quickly when the environment changes.

In spite of cloud computing's growing influence, there are some concerns about it remain as issues. Some common challenges are data protection, data recovery and availability, management capabilities, regulatory and compliance restrictions, and time latency.

2.4 Data Exchanging Formats

Whether building a thin client (web application) or a thick client (client-server application), at some point probably there is a need for making requests to a web server and need good data format for responses. As of today, three major data formats are being used to transmit data from a web server to a client: CSV, XML, and JSON. [32].Table 2.3 describes the features of the three data Xchanging formats.

Table 2.3 Data Exchanging formats

Data format	CSV	XML	JSON
Definition	stands for "comma-separated values". As the name implies, this data format is basically a list of elements separated by commas	stands for "extensible markup language". It was created to better represent data formats with a hierarchical structure.	stands for (Javascript Object Notation). It was created as an alternative to XML. it represents hierarchical data with the use of commas.
Format	Eric,Andrea,Kusco	<person> <name><br /> Eric<br /> </name><br /> <age><br /> 26<br /> </age> </person>	{"name":"Eric","age":"26"}
Drawbacks	The parser is required to convert the CSV data into a native data structure so that incur the overhead of maintaining an even more complex parser	Is about three times as large as CSV. This is because each data element has an associated open and close parameter tag.	Since JSON is relatively newer than XML, fewer APIs exist to automatically convert JSON to native data structures
Advantage	It is the most compact of all three formats so it can help reduce bandwidth	• Fully supports hierarchical data structures. • Human-readable. • First standard hierarchical data format.	• Supports hierarchical data while being smaller in size than XML. • Very useful for web applications.

2.5 Applications of the Internet of Everything

All domains of Smart Cities and digital cities need IoE/IoT applications that manipulate information in a collaborative environment and store it on the Internet cloud. Smart Cities and Smart establishments promise to enhance living conditions, safety, optimize traffic, and economic, social, and cultural development. A Smart City contains a large number of things, a set of integrated services, and a bunch of applications[23]. Many smart city domains like healthcare, e-learning, industrial, and so on are supported by IoE. Figure (2.9) shows domains of IoE applications.

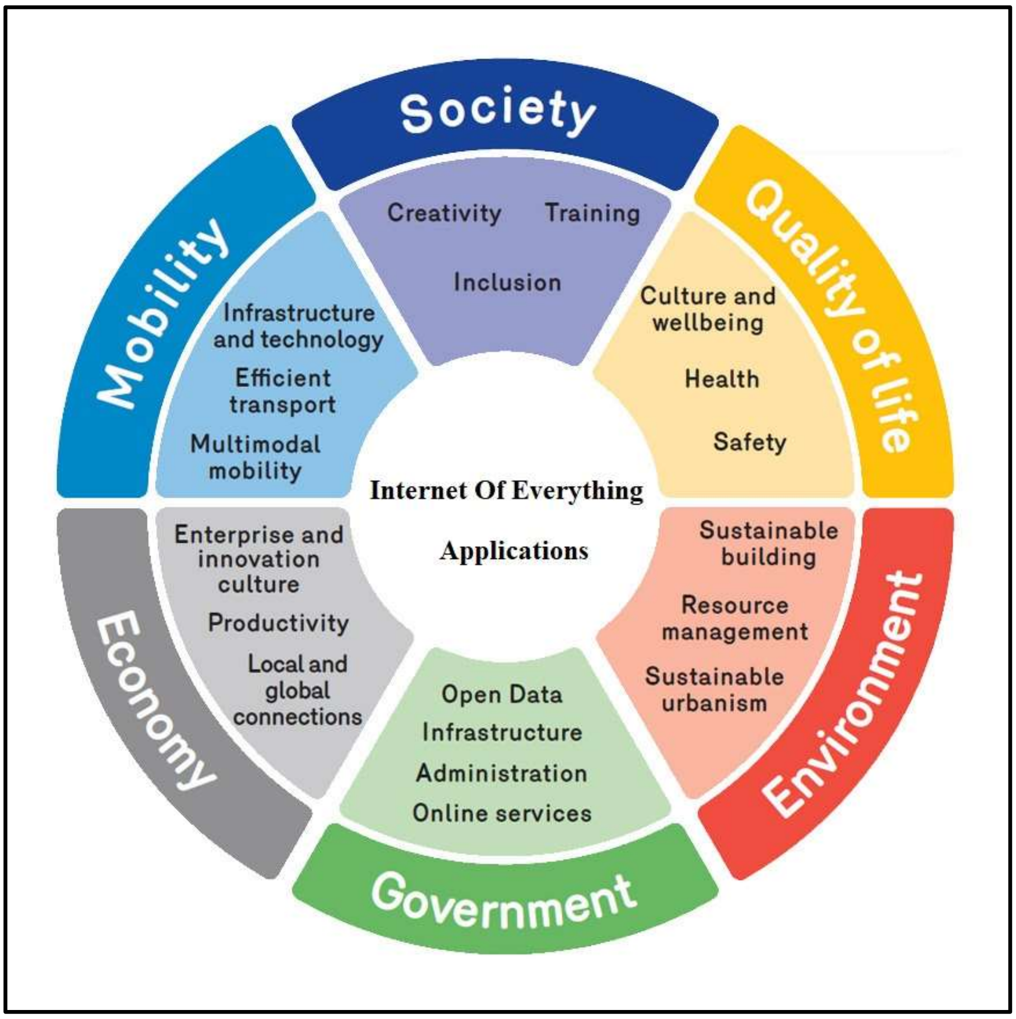


Figure 2.9 Domains of IoE Applications

The industrial section in many sides makes benefit from IoE capabilities, for example. The IoE also supports human security applications like using movement detectors and camera data to detect possible intruders. Furthermore, the Healthcare applications supported by IoE are considered as an important sector so eHealth applications can measure blood pressure, glucose levels, and other patient vitals. A patient has a new fitness tracker that can add fitness data to the eHealth solution[14].

Chapter Three

Programming and Implementation of Smart City Prototype Using Fog Computing Architecture

3.1 The Proposed Smart City Prototype

Smart cities are the new goal for the integrated System of Systems (SoS) and it defines the stringent connectedness between the real and virtual worlds to improve the life quality of citizens and the sustainability of cities. There are three important essential services that role direct contact with people's daily life like Health care services, security control services, and environment sense services. These sectors need to be accurate and have an immediate response to accomplish better health and secure society.

To be able to design a smart system many points should be considered:

- What services should the system provide?
- How can these services improve this sector?
- What are the physical devices needed to make the topology and are they available in the market?
- Cost of the smart system.
- The network connection and programming.
- Make suitable APIs to run the system smoothly.

In our smart city prototype Instead of designing a specific smart establishment, we decided to design smart systems that are flexible to be installed in any establishment and make it smart. Therefore three clusters (platforms) are designed to enhance healthcare, security, and the environment sectors. They are flexible and expandable to be installed in any organization like homes, offices, factories, markets, hospitals, and many others.

Figure (3.1) shows a general block diagram for our proposed Smart City IoT-based prototype. This figure clarifies the expandability of the prototype since each cluster can handle many copies of the same system that has been installed in any location in the Smart City. The flexibility is achieved through the ability to install the proposed system prototype in any location in the city.

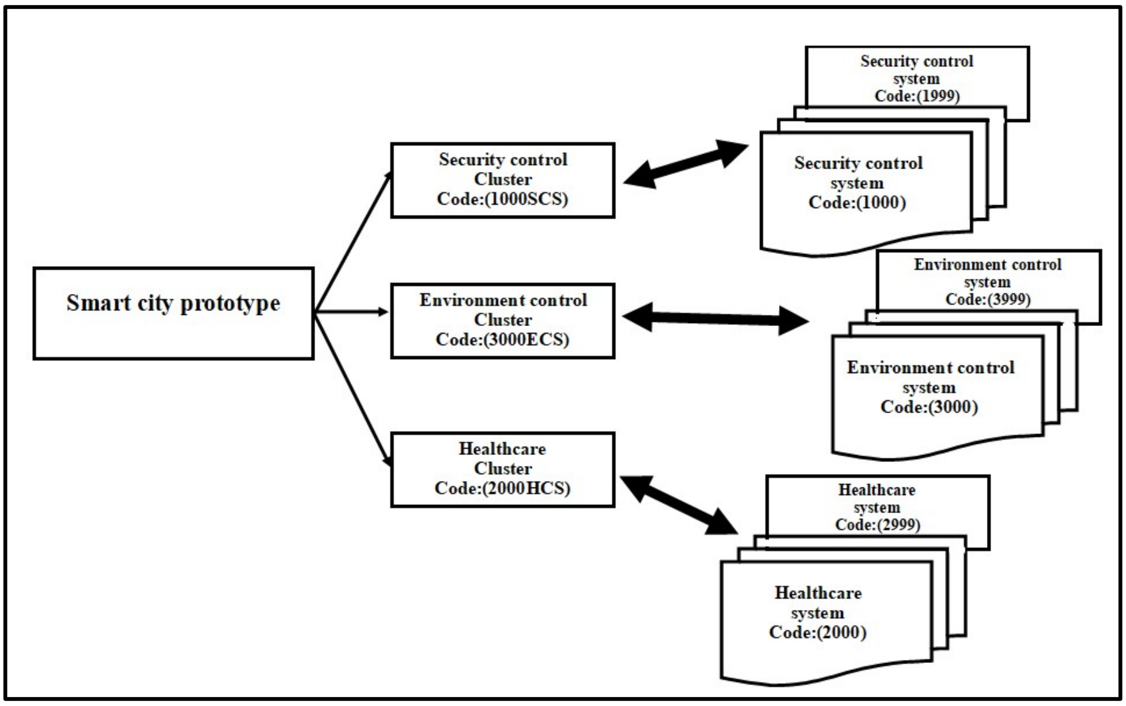

Smart city prototype
Security control Cluster Code:(1000SCS)
Environment control Cluster Code:(3000ECS)
Healthcare Cluster Code:(2000HCS)
Security control system Code:(1999)
Security control system Code:(1000)
Environment control system Code:(3999)
Environment control system Code:(3000)
Healthcare system Code:(2999)
Healthcare system Code:(2000)

Figure 3.1 General block diagram for our proposed Smart City IoT based prototype

Each Cluster is identified by unique code (shown in figure 3.1) and consists of many subsystems lied under the same cluster (such as security control). Table 3.1, gives unique codes for the clusters. All systems upload the data collected from their sensors and smart devices to the cluster server which in turn sends the received data to the centralized server as will be explained later.

Table 3.1 The unique code of each Cluster and the maximum number of systems

Clusters	Identification Code	Range of systems
Environment Sense Cluster	3000ESS	*1000*
Health Care Cluster	2000HCS	*1000*
Security Control Cluster	1000SCS	*1000*

As you can be seen in Figure (3.2) Cluster servers use Fog computing technologies to process massive data and information locally and achieve two main jobs on data, first, it sends back real-time response to control the actuators and smart devices in the system, and second, it reformats and arranges the collected data from all subsystems in this cluster and send it to the Centralized server as a stream of messages. The role of the Centralized server will be explained in chapter four.

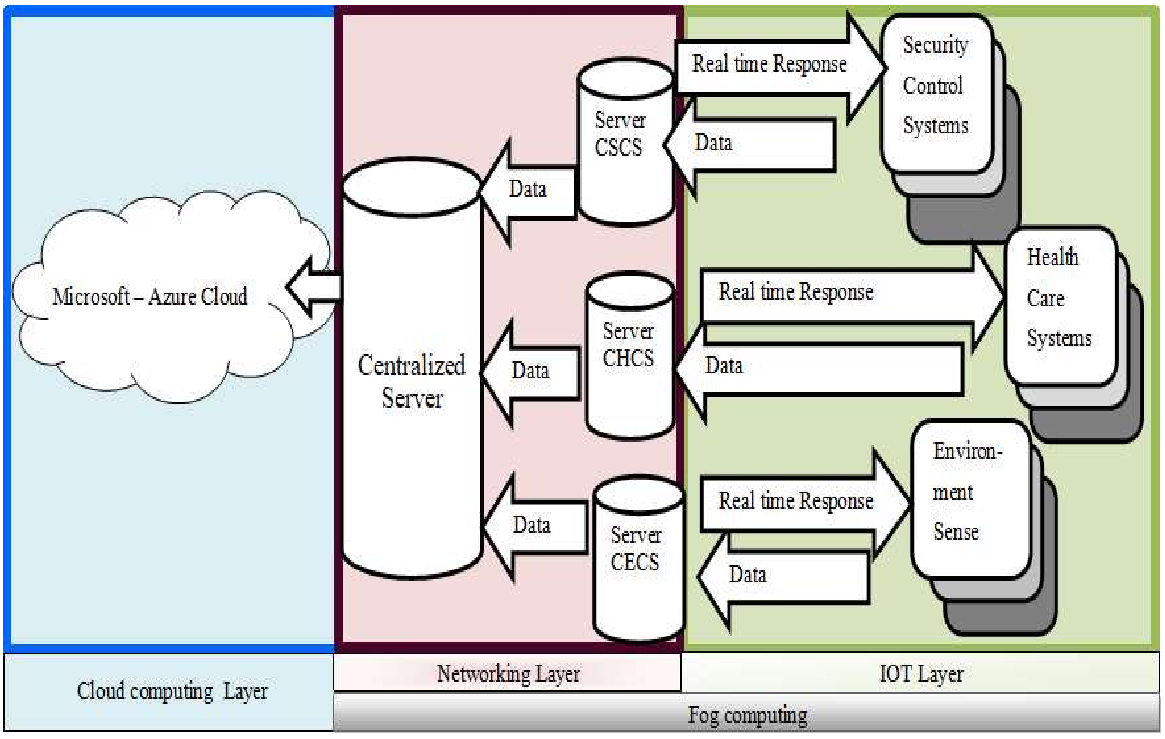


Figure 3.2 Smart city Data Flow Structure

3.2 IoE Simulators

One of the most powerful and widespread evaluation methodologies is network simulation in the area of computer networks[31]. There are many simulators to represent the wired and wireless networks for the IoT. We experiment with four simulators Omnetpp 5.5, Cupcarbon, iFogSim, and packet tracer. The packet tracer is found to be the best simulating tool for our proposed system because it includes many qualitative criteria mentioned in table 3.2.

Table 3.2 Nine Qualitative Criteria of Packet Tracer

Criteria	Packet Tracer Characteristic
Nature of the software	Simulator
Type of the software	Discrete event
License	Proprietary, but an End User License Agreement (EULA) exists
User Interface	GUI: Yes, a built-in GUI interface is supported, with a possibility to trace and store all events. Different languages are supported for the GUI including English, Russian, German, Portuguese, Spanish, and French. Supported programming language: Non, it is private property, but scripting is allowed using the Cisco IOS Syntax.
Platform	Linux, Android 4.1+, iOS 8+, and Microsoft Windows.
Heterogeneity	It is supported, different types of real routers, such as Cisco 1941, Cisco 2901, Cisco 2911, and others are supported, as well as different types of real switches like: Cisco Catalyst 2950, Cisco Catalyst 2960, Cisco Catalyst 3560-24PS are supported. In addition to that, Linksys WRT300N wireless router, Cisco 2504 wireless controller, and Cisco Aironet 3700 access point are supported. Cisco ASA 5505 firewall is supported as well. Variety of IoT devices are supported.
Modeling	It is not supported.
Level of details	Packet level.

Criteria	Packet Tracer Characteristic
Supported technology and protocols	Application Layer: Protocols: DHCP, DHCPv6, FTP, HTTP, HTTPS, RADIUS, POP3, SMTP,SNMP, SSH, Telnet, TACACS. Technology: Access Lists, DNS, IoT, IoT TCP, SYSLOG. Transport Layer: Protocols: SCCP, TCP, UDP. Network Layer: Protocols: ARP, CAPWAP, HSRP, HSRPv6, ICMP, ICMPv6, IP, IPv6, NDP.Technology: IPsec, Cisco NetFlow. Link Layer: Protocols: Bluetooth, CDP, CTP, H.323, LACP, LLDP, PAgP, STP, USB, VTP. Routing Protocols: BGP, EIGRP, EIGRPv6, OSPF, OSPFv6, RIP, RIPng,

3.3 GUI -based Implementation of The Proposed Smart Platforms.

A prototype of a smart city is designed using the Cisco Packet tracer 7.2 simulator taking in our considerations that this prototype is suitable to be applied in the reallocation. This is because the connected devices and links are used to simulate Cisco's real devices like routers, switches, and wireless gateways. Secondly, the sensors and actuators representing devices that are available in markets. Finally, the programming language chosen to program servers is quite common and strong language and compatible with most devices and servers.

Figure (3.3) shows the GUI of the proposed smart city prototype we can see there are three systems connected to the cluster servers all systems are connected to the internet wirelessly and obtain their IPs from the Internet Service Provider (ISP). The devices are communicating with Fog Nodes(cluster servers) sending their status. the street region contains smart streetlight which senses the Motion and Light and on and off if there is amotion at Night. Also, the police and fire engine cars have connected to Fog Nods to receive the notification if there are problems.

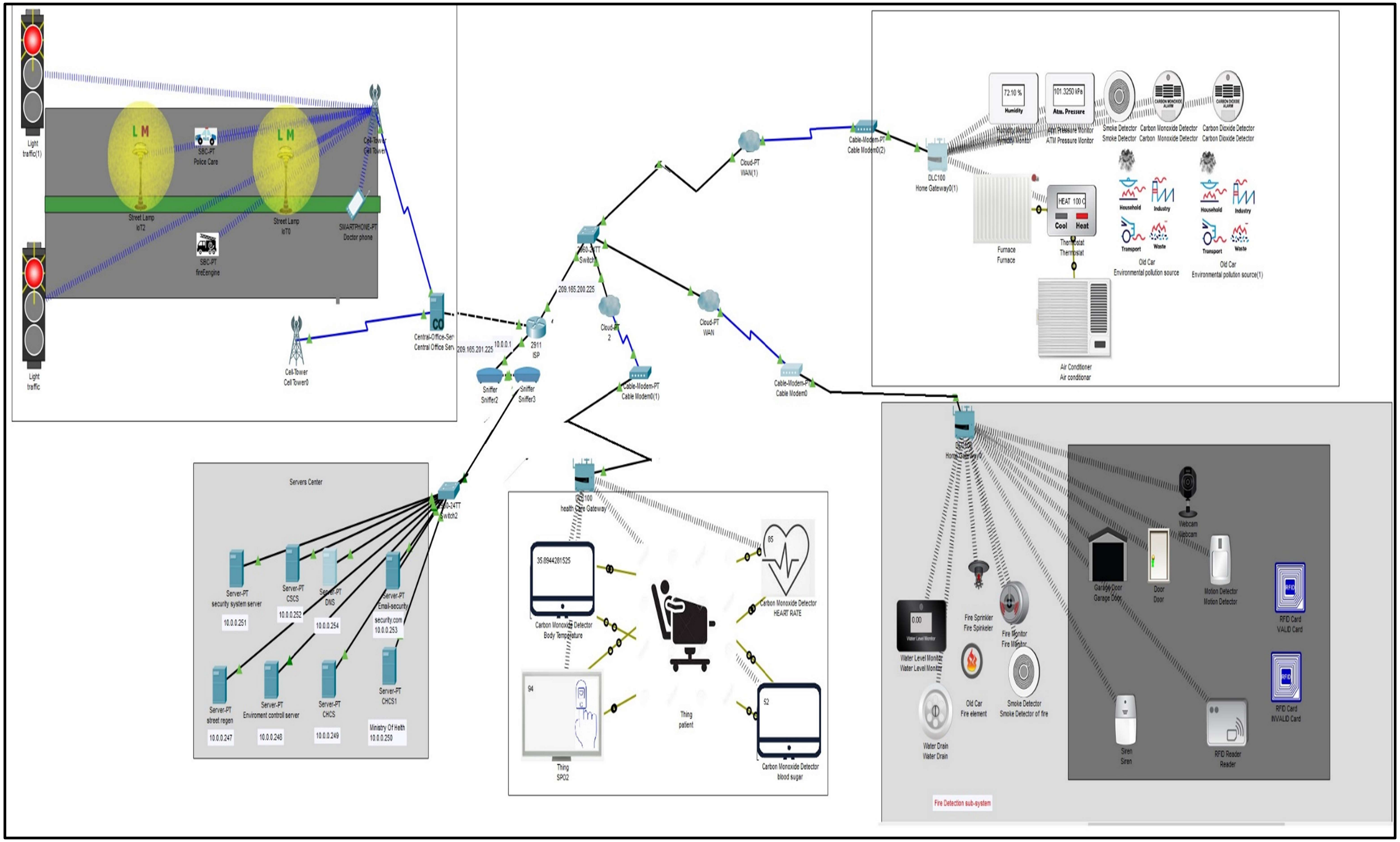


Figure 3.3 the GUI of designed Smart City Prototype

Each system consists of sensors and actuators that have been programmed with python to be able to simulate the real device. General algorithms for programing both the sensors and actuators are shown below.

	Sensors General Algorithm Algorithm 3-1
1	Set all the initial values
2	Sets up the IOE API # for the remote monitor from the IoE server.
3	Sense the Change either in Environmental or in Physical #according to sensor function
4	Send report to IOE server
5	Delay for 1 second
6	Loop to step (3)

	Actuators General Algorithm Algorithm 3-2
1	Set all the initial state
2	Sets up the IOE API # for remote monitor and control from the IoE server.
3	Check for Input # input can be direct as (digital or analog) or from a remote server
4	Process the Received input and change state according to it
5	Send the state to the IOE server
6	Delay for 1 second
7	Loop to step 3

Each server in our smart city is also programmed with a python programming language to meet its dedicated goal. The server's algorithms will be explained for each system separately later.

Simulation implementation of the Smart systems in our proposed smart city is given in the following sections.

3.4 Health care system

Healthcare systems have a wide variety around the world, many companies have developed different kinds of healthcare systems according to their requirements and resources. The smart healthcare system developed by us is an essential health vital indicator measurement of patients and it allows the doctors and nurses to follow-up their patient's health condition remotely to provide better healthcare for society.

Our proposed system consists of four devices that are connected to the patient to sense its health condition. Figure (3.4) shows the Health care system topology.

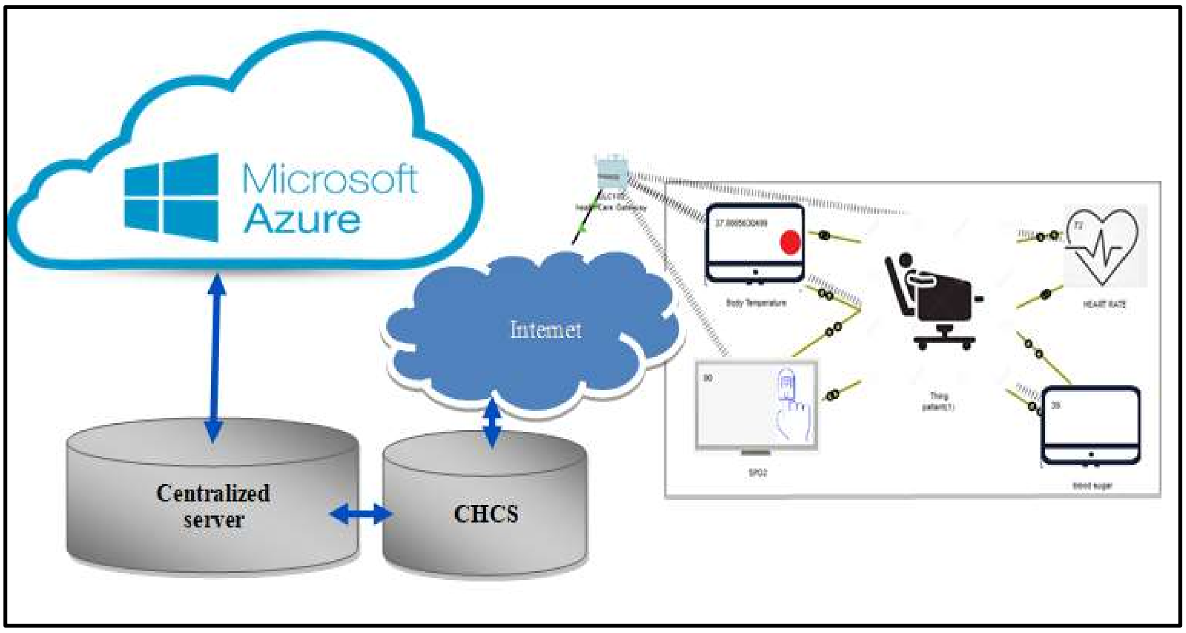


Figure 3.4 Health care system topology

The Devices Functionality is explained below:

3.4.1 Heart rate recorder

Patient heart rate is one of the most important vital indicators of its health so the medical staff in charge of its condition must receive the updated patient's heart rate constantly.

The device is connected to the patient's body sensing the heart rate and displaying it on its LCD screen. Figure 3.5 shows the outer physical shape of the heart rate recorder.

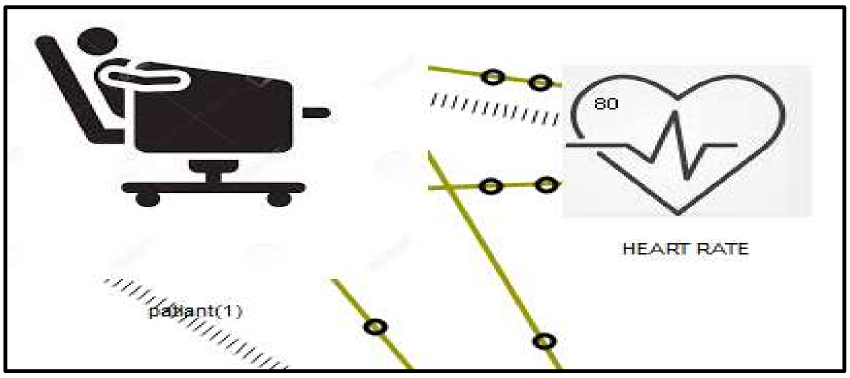
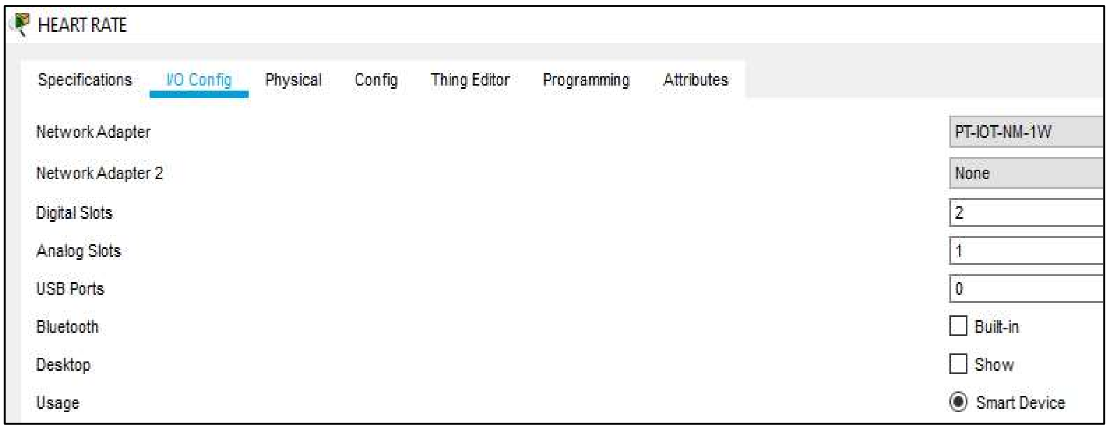


Figure 3.5 Heart rate recorder

The device is provided with a wireless interface 2.4GHz and has two digital slots and one analog slot as mentioned in figure (3.6).



Network Adapter	PT-IOT-NM-1W
Network Adapter 2	None
Digital Slots	2
Analog Slots	1
USB Ports	0
Bluetooth	☐ Built-in
Desktop	☐ Show
Usage	● Smart Device

Figure 3.6 Heart Rate Input/output configuration

The analog input reads the heart rate from the patient and the digital input slot (0) reads the patient Identification code, the device analyzes the analog data if it is less than 60 heartbeats/second and more than 90 heartbeats/sec, it set the alarm state to (1). Also, it displays the heart rate on the device's LCD and changes the color to red if there is abnormal status as figure (3.7) shows.

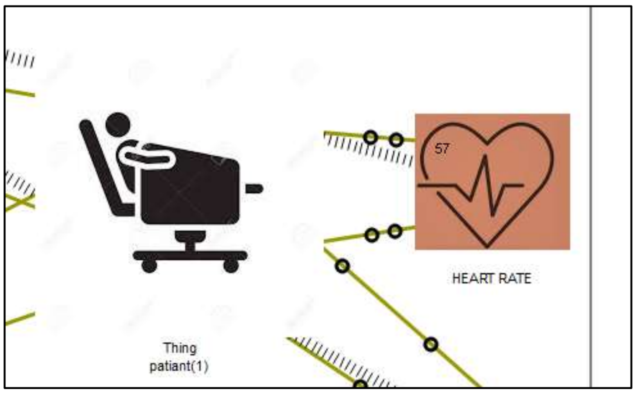


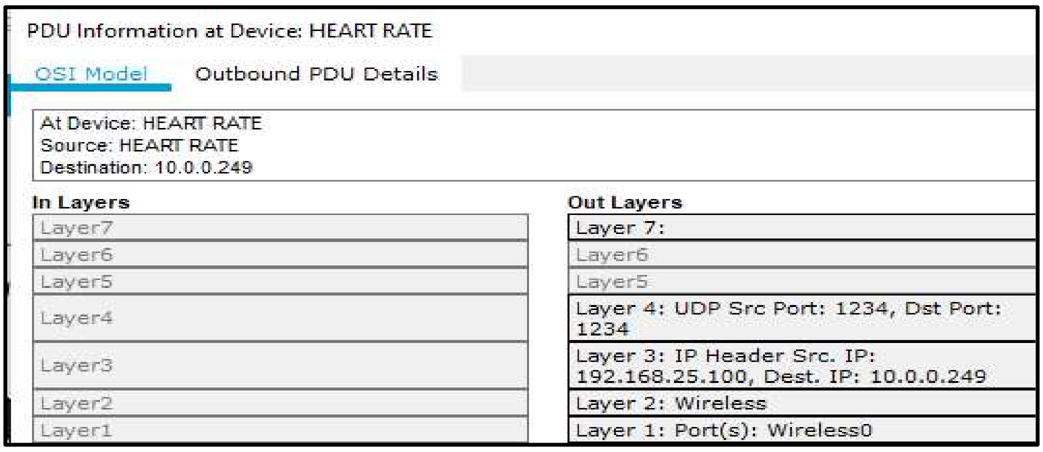
Figure 3.7 Heart Rate alarm

The device then reformats the Data as Colom separated value format (CSV) to prepare it to send to the Cluster Healthcare Server (CHCS). The message format below consists of six fields, the first four used to identify the heart rate reader device to the CHCS, and the last two fields are the patient ID and heart rate value:

Heart rate Device to CHCS message format:

1	2	3	4	5	6
Cluster Code (2000HCS)	**Local IP Address**	**Heart Rate**	**Device Serial**	**Patient ID**	**Heart Rate value**

The Heart rate device uses the wireless interface to send the produced message to the CHCS. Figure (3.8) shows the message header format.


PDU Information at Device: HEART RATE

OSI Model | Outbound PDU Details

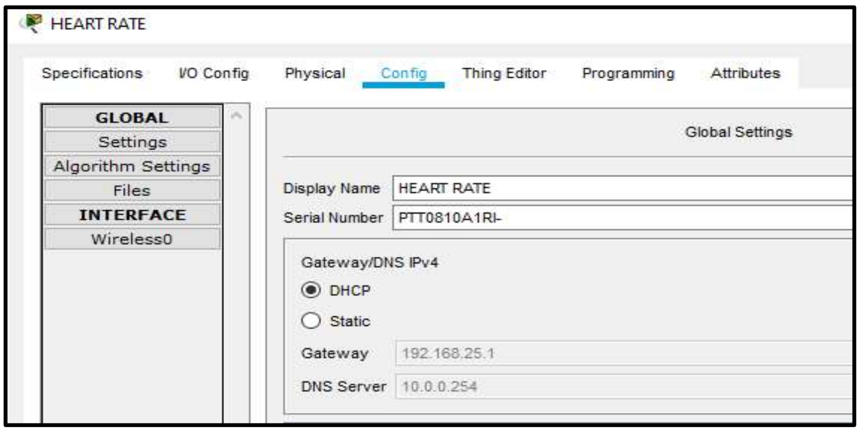
At Device: HEART RATE
Source: HEART RATE
Destination: 10.0.0.249

In Layers	Out Layers
Layer7	Layer 7:
Layer6	Layer6
Layer5	Layer5
Layer4	Layer 4: UDP Src Port: 1234, Dst Port: 1234
Layer3	Layer 3: IP Header Src. IP: 192.168.25.100, Dest. IP: 10.0.0.249
Layer2	Layer 2: Wireless
Layer1	Layer 1: Port(s): Wireless0

Figure 3.8 PDU Information of Heart rate Device Message

The messages are sent over the UDP protocol every 6 seconds so we predict that the device may send about 600 messages to the server each hour. This large number of messages are analyzed and filtered in aggregation sublayer as will be seen in Chapter(4).

The Health care getaway wireless device provides the IP address to all devices connected to it using the DHCP protocol. Figure (3.9) shows the network configuration of the Heart rate reader device.


HEART RATE

Specifications | I/O Config | Physical | Config | Thing Editor | Programming | Attributes

GLOBAL
Settings
Algorithm Settings
Files
INTERFACE
Wireless0

Global Settings

Display Name: HEART RATE
Serial Number: PTT0810A1RI-

Gateway/DNS IPv4
[✓ DHCP] Static
Gateway: 192.168.25.1
DNS Server: 10.0.0.254

Figure 3.9 Network configuration of Heart rate reader

3.4.2 Body temperature meter

Measuring body temperature is the basic step in any patient checkup so a prototype device is developed that senses the patient’s body temperature and sends it wirelessly to the server. The device is connected physically to the patient's body as shown in figure (3.10).

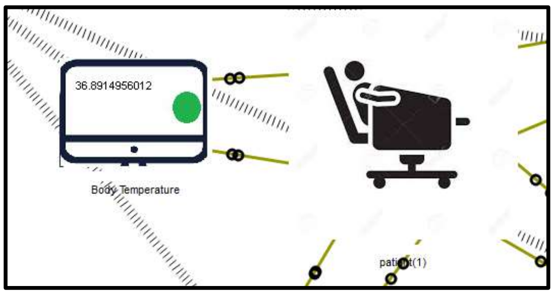
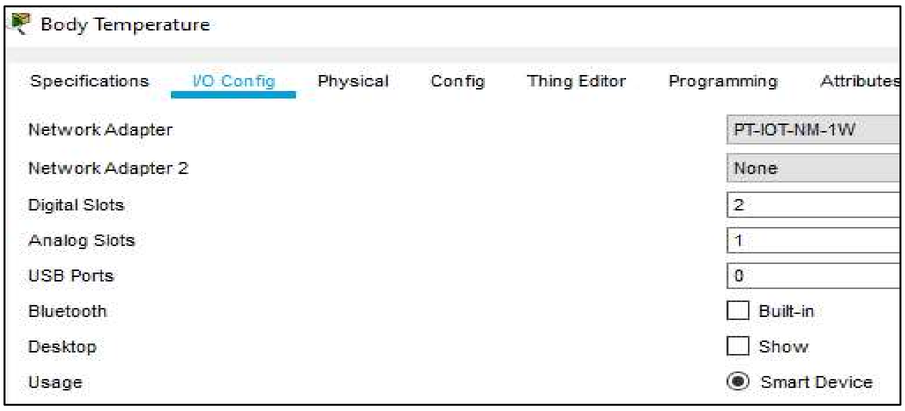


Figure 3.10 Body temperature Meter

The device is provided with a wireless interface 2.4GHz and has two digital slots and one analog slot as explained in figure (3.11).

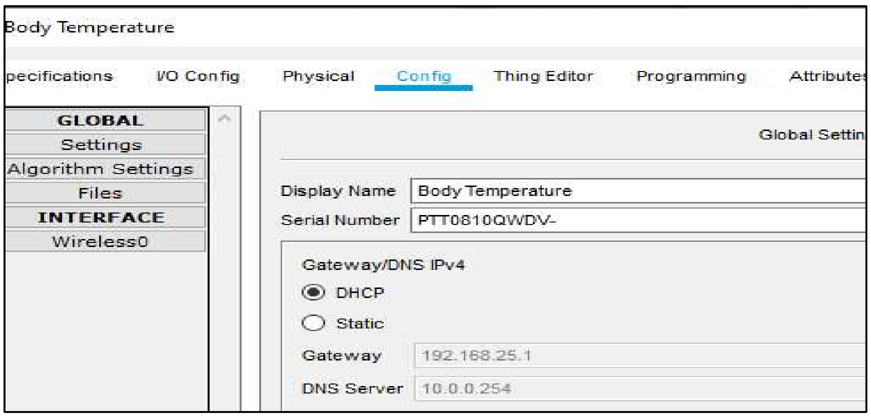

Body Temperature

Specifications | I/O Config | Physical | Config | Thing Editor | Programming | Attributes

Network Adapter	PT-IOT-NM-1W
Network Adapter 2	None
Digital Slots	2
Analog Slots	1
USB Ports	0
Bluetooth	☐ Built-in
Desktop	☐ Show
Usage	● Smart Device

Figure 3.11 Body Temp. Meter Input/output configuration

The wireless interface is connected to the Health care gateway which provides the IP address to all devices connected to it using DHCP protocol. Figure (3.12) shows the network configuration of the Body Temperature Meter device.


Body Temperature

pecifications | I/O Config | Physical | Config | Thing Editor | Programming | Attribute

GLOBAL
Settings
Algorithm Settings
Files
INTERFACE
Wireless0

Global Settin

Display Name	Body Temperature
Serial Number	PTT0810QWDV-

Gateway/DNS IPv4
● DHCP
○ Static

Gateway	192.168.25.1
DNS Server	10.0.0.254

Figure 3.12 Network configuration of Body Temp. Meter

The digital inputs functionalities are to read the patient Identification code with slot (0) and display the received data on the device Monitor with slot (1), while the analog input reads the temperature of the patient. The device analyzes the analog data if it is between 97°F (36.1°C) to 99°F (37.2°C), it is considered that this is a normal situation and set the alarm state to (0), otherwise, it sets the alarm state to (1). Figure (3.13) shows the alarm state of high body temperature.

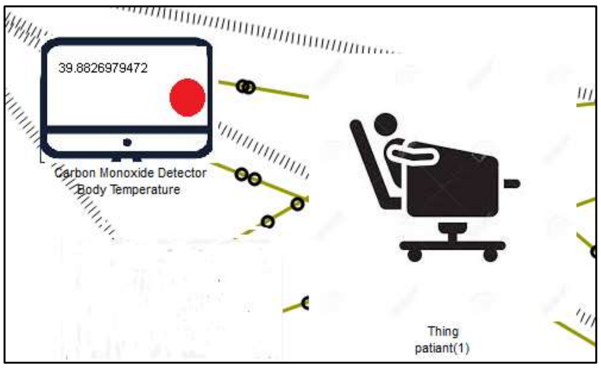


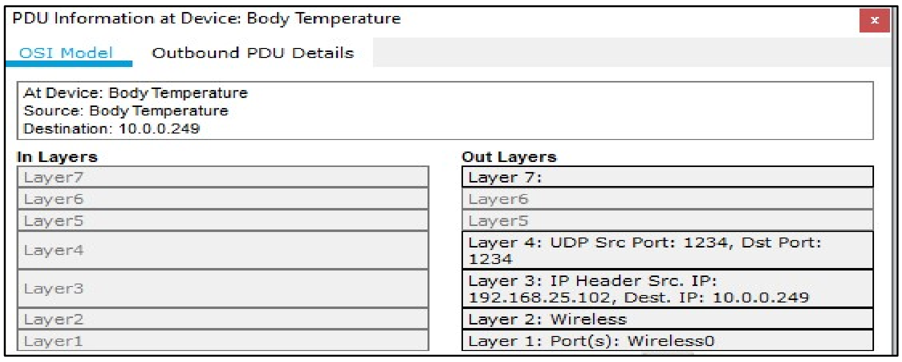
Figure 3.13 High body temperature

The collected data are organized and formatted as (CSV) data and then sent as a message to the cluster Healthcare Server (CHCS). the message format is almost similar to all devices connected to the server just different in the device name field (5) and record field (6) as can be seen below.

Body temperature meter to CHCS message format:

1	2	3	4	5	6
Cluster Code (2000HCS)	**Local IP Address**	**Body temperature**	**Device Serial**	**Patient ID**	**Body temperature value**

This message is sent over UDP protocol to the IoE API in the server and Fig. (3.14) shows the message PDU information.



At Device: Body Temperature
Source: Body Temperature
Destination: 10.0.0.249

In Layers	Out Layers
Layer7	Layer 7:
Layer6	Layer6
Layer5	Layer5
Layer4	Layer 4: UDP Src Port: 1234, Dst Port: 1234
Layer3	Layer 3: IP Header Src. IP: 192.168.25.102, Dest. IP: 10.0.0.249
Layer2	Layer 2: Wireless
Layer1	Layer 1: Port(s): Wireless0

Figure 3.14 Message PDU information

The body temperature meter device sends periodic messages to the CHCS every 6 seconds with a message packet size (64) byte, which means that in an hour, the server receives about (600) messages to be about (38.5) kilobyte of data, the CHCS should analyze and respond to it in the real-time as will be explained.

3.4.3 Peripheral oxygen saturation meter (SpO2):

Pulse oximetry is a method used to approximate the percentage of oxygen bound to hemoglobin in the blood. The SPO2 is a small device that clips to the body (typically a finger) and transfers its readings to a reading meter using wired or wireless technology. The device uses light-emitting diodes in conjunction with a light-sensitive sensor to measure the absorption of red and infrared light in the extremity. The difference in absorption between oxygenated and deoxygenated hemoglobin makes the calculation possible.

Oxygen saturation values for healthy individuals at sea level are usually between 99% and 96% and should be above 94% [33] as shown in table 3.3.

Table 3.3 Effects of decreased oxygen saturation

SaO_2	Effect
95% and above	No evidence of impairment
80% and less	Impaired mental function on average
75% and less	Loss of consciousness on average

A simulated smart device is designed according to the above considerations as shown in Figure (3.15)

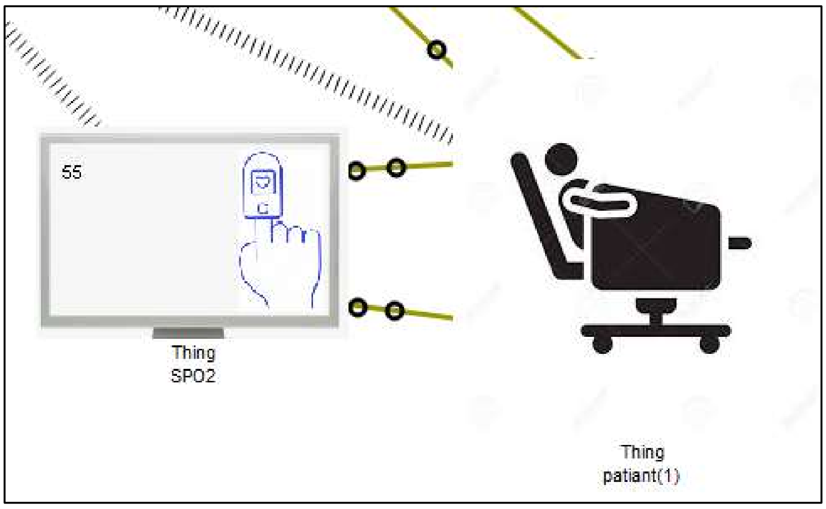
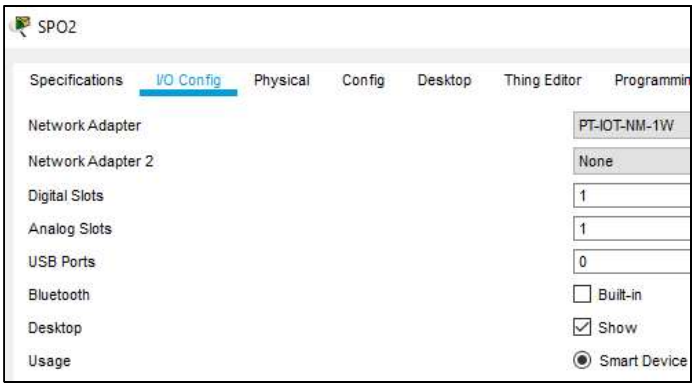


Figure 3.15 Peripheral oxygen saturation meter (SpO2)

The device has one digital slot to read the Patient's ID and an analog input to read the Spo2 value as shown in figure (3.16). these values are analyzed and then sent as an accumulated message to the CHCS wirelessly in the same procedure as mentioned in section 3.4.2.



Network Adapter	PT-IOT-NM-1W
Network Adapter 2	None
Digital Slots	1
Analog Slots	1
USB Ports	0
Bluetooth	☐ Built-in
Desktop	☑ Show
Usage	● Smart Device

Figure 3.16 SPO2 input-output configurations

3.4.4 Blood Sugar meter:

Diabetes is described as "a metabolic problem in which the body is unable to regulate the levels of glucose in the blood". The cause of this disorder is the lack of insulin, which is the hormone responsible for regulating the levels of glucose. the glucose in blood concentration in a safe range between 54 mg/dL and 120 mg/dL. [34].

Our proposed simulated device lied under the above limits, it tests the blood glucose if it is under the lower limit or above the upper limit it alarms the server that there is a critical health situation for this patient. Figure (3.17) shows the blood sugar meter device.

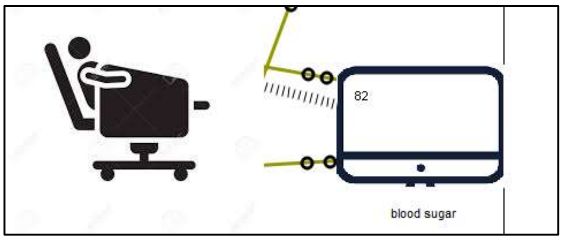
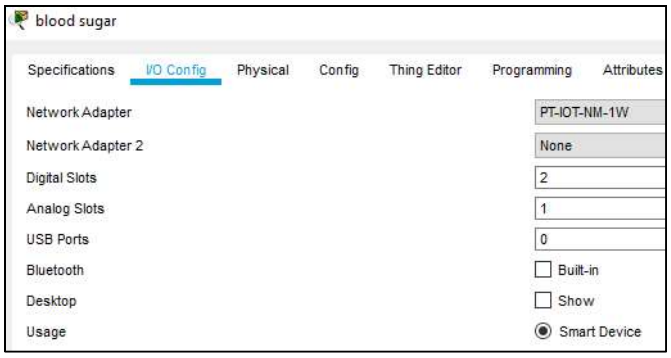


Figure 3.17 Blood sugar meter

As we can notice from figure (3.17) the device displays the patient glucose percentage on its Lcd with alarming red led if there is an emergency.

The Input-output configuration consists of 2-digital slots first one connected to the patient's bed to get its ID, the other slot is to display the glucose value on device Lcd.

Also, the device has an analog input to read the analog input coming from its sensor pasted on the patient body to detect glucose level. Finally, the device is provided with a wireless adapter of 2.4GHz to communicate smartly with the Cluster Healthcare server (CHCS). Figure (3.18) shows the Blood sugar meter input-output configurations.



Network Adapter	PT-IOT-NM-1W
Network Adapter 2	None
Digital Slots	2
Analog Slots	1
USB Ports	0
Bluetooth	☐ Built-in
Desktop	☐ Show
Usage	● Smart Device

Figure 3.18 Blood sugar meter input-output configurations

The device accumulates the incoming data and, then the system information sending it as a message to the CHCS as explained in section (3.2.1). This message is sent every 6 seconds using the UDP socket connection with the Custer server.

3.4.5 Cluster Healthcare server (CHCS)

This cluster server uses the edge Fog Computing concept to collect data from the medical devices explained previously analyzes the data and replies to the system with a real-time response. The data analysis is done according to the following equations

$$\text{Halarm} = \begin{cases} +1 & \text{H} > 90 \\ -1 & \text{H} < 60 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Galarm} = \begin{cases} +1 & \text{G} > 120 \\ -1 & \text{G} < 54 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{SPO2alarm} = \begin{cases} +1 & \text{SPO2} < 80 \\ -1 & \text{SPO2} < 75 \\ 0 & \text{SPO2} \geq 95 \end{cases}$$

$$\text{Talarm} = \begin{cases} -1 & \text{T} < 36.1 \\ +1 & \text{T} > 37.2 \\ 0 & \text{otherwise} \end{cases}$$

While:

H: Heart rate value

Halarm: is a heart rate alarm

T: Body Temperature value

Talarm: Body temperature alarm

Spo2: spo2 value

SPO2alarm: SPO2 alarm

G: Blood sugar value

Galarm: Blood Sugar Alarm

It is work according to the algorithm (3.5) below:

	Clustered Health care control server Algorithm 3-3
1	Setup the TCP connection
2	Setup the UDP connection
3	Initiate IOE connection with smart devices
4	Initiate Email configuration: Email Address: sever@health.com Incoming/outgoing mail Server: www.health.com Password: ********

5	Receive inputs from devices
	H= heart rate
	G=blood sugar
	Spo2= SPO2
	T=body Temperature

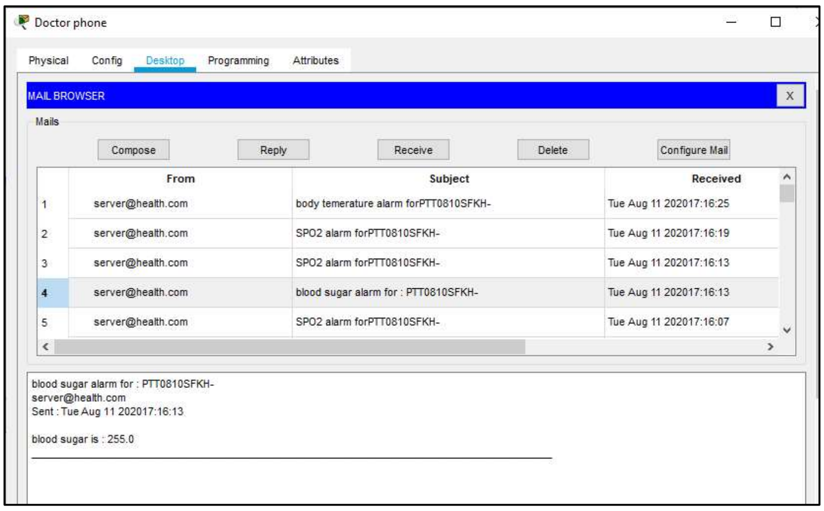
6	Analyze the inputs of each device
7	If (90<H) OR(H<60)
8	Halarm = 1
9	Send Email (**Destination:** doctor@health.com **subject:** Patient (Patient ID) Heart rate alarm **Body:** Heart rate is: H
10	End if

11	If (120< G)OR(G <56)
12	Galarm=1
13	Send Email (**Destination:** doctor@health.com **subject:** Patient (Patient ID) Blood Sugar alarm **Body:** Blood Sugar is: G
14	End if
15	If (94> SPO2)
16	SPO2alarm=1
17	Send Email(**Destination:** doctor@health.com **subject:** Patient (Patient ID) SPO2 alarm **Body:** SPO2 is: SPO2
18	End if
19	If (36.1> T) OR(T>37.2)
20	Talarm=1
21	Send Email (**Destination:** doctor@health.com **subject:** Patient (Patient ID) Body Temperature alarm **Body:** Body Temperature is: SPO2
22	End if
23	Send report to Centralize server as UDP packets
24	Loop to step 5

As we can see from the algorithm the centralizes healthcare control server has two main functionalities

Firstly, it analyzed the inputs by checking their upper and lower limits if there is any abnormal situation, the server sends an alarming email to the doctor’s smartphone responsible for this case.

The email body consists of the abnormal value and the patient ID as shown in figure (3.19).



	From	Subject	Received
1	server@health.com	body temerature alarm forPTT0810SFKH-	Tue Aug 11 202017:16:25
2	server@health.com	SPO2 alarm forPTT0810SFKH-	Tue Aug 11 202017:16:19
3	server@health.com	SPO2 alarm forPTT0810SFKH-	Tue Aug 11 202017:16:13
4	server@health.com	blood sugar alarm for : PTT0810SFKH-	Tue Aug 11 202017:16:13
5	server@health.com	SPO2 alarm forPTT0810SFKH-	Tue Aug 11 202017:16:07

```
blood sugar alarm for : PTT0810SFKH-
server@health.com
Sent : Tue Aug 11 202017:16:13

blood sugar is : 255.0
```

Figure 3.19 Email service in Doctor Phone

Doctor phone is receiving the email when it is connected to the internet from any location as an example in the street as shown in figure (3.20).

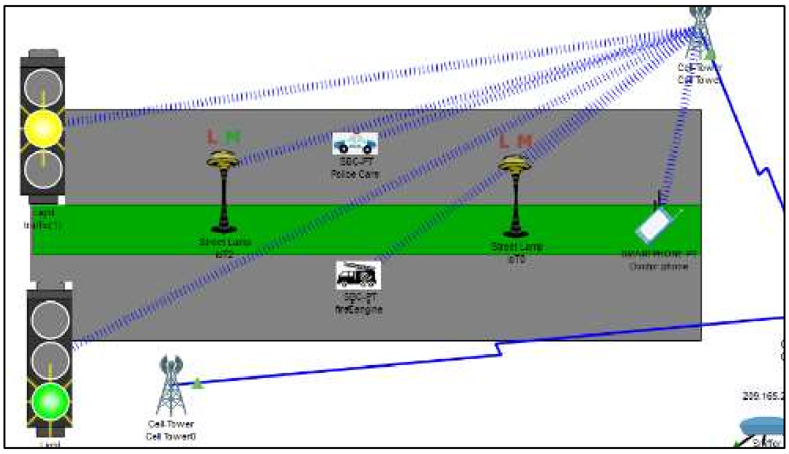


Figure 3.20 doctor phone connected through 4G to the internet

The second functionality of the CHCS is to export the data to the Centralized Fog Server the data is organized in CSV format and send over the UDP Socket connection between CHCS and the Centralized server.

The message format sends as below:

Gateway	Gateway port	Cluster Code	Device local IP Address	Patient ID	Device Name	Device Serial	data
1	2	3	4	5	6	7	8

This message will be uploaded to Microsoft Azure Cloud as will be explained in Chapter 4

3.5 Security Control System

Ordinary security systems are good but still, have some issues with the real-time response in the emergency stations, unlike the smart security systems which inform all the commands to controlling centers within seconds to halt or minimize the loss.

The main contribution of this system is to design a Security Control System able to be installed in any location to provide instant notification to concerned departments to prevent or to minimize the disaster. This is achieved by remotely control and monitoring the sensors and security devices during normal conditions to guarantee full control over the Security Control System at any time and from any place. The system also can remotely program and configure all these devices. Finally, the Security Control System collects and analyzes data of all sensors and security devices to make smart decisions during abnormal situations or an emergency.

The smart security system consists of two subsystems: the Fire detection and the building security subsystems so we will explain each subsystem separately.

3.5.1 Fire detection subsystem

In this system, the Network topology consists of a fire monitor sensor, smoke detector sensor, water level monitor, sprinkler, and water drain. All these devices are provided with a wireless adapter interface of 2.4GHz band which is suitable for connection to wireless networks or home gateways.

The devices are wirelessly connected to the home gateway and obtained their IP addresses by DHCP protocol; the gateway connects the system to the internet. as shown in figure (3.21).

All devices are registered to the Cluster Security Control Server (CSCS).

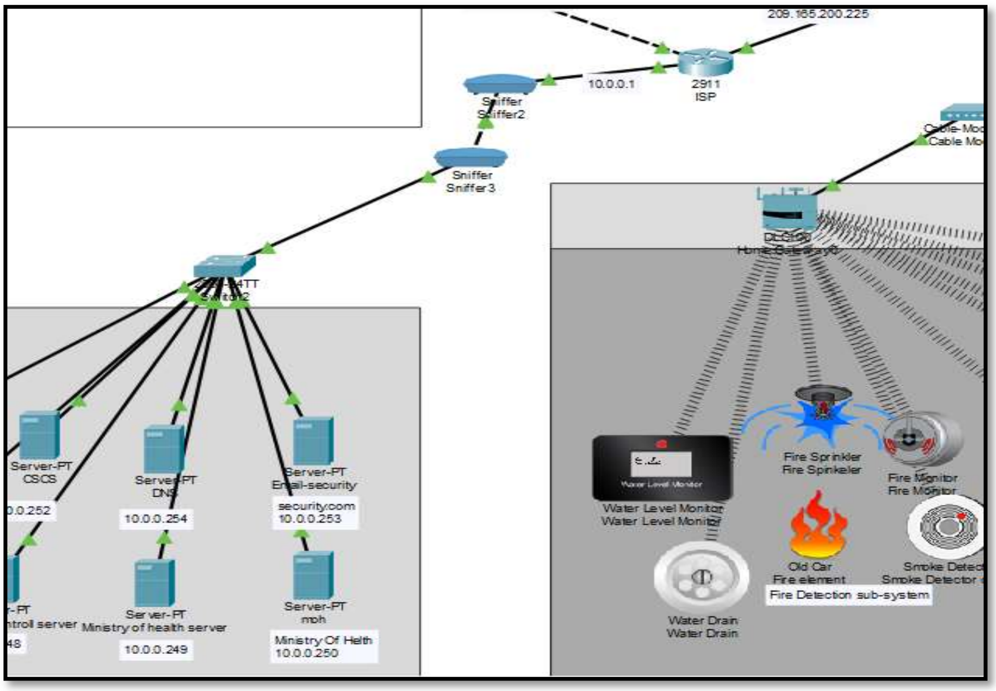
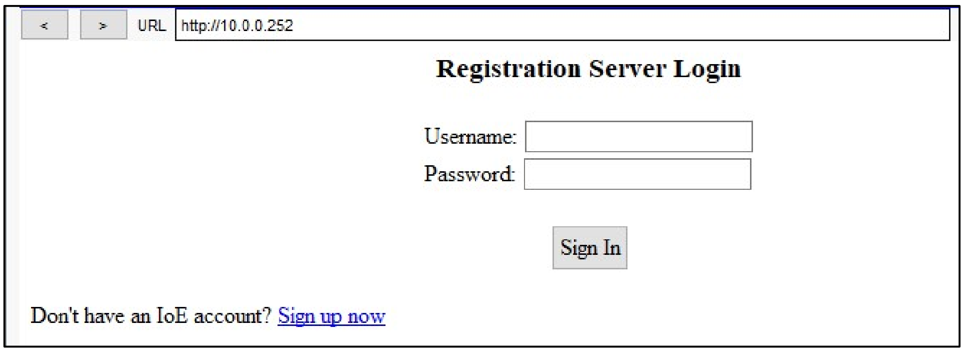


Figure 3.21 Network topology of Fire Detection subsystem

The CSCS collects data from sensors analyzing it according to algorithm 3-3 which describes smart decisions based on multi-inputs from more than one sensor which implements the M2M connection and then responses to the system in the real-time. The CSCS is provided by the user interface web page as seen in figure (3.22) allowing administrators and decision-makers to login to the system and program the settings remotely.



http://10.0.0.252

Registration Server Login

Username:
Password:

Sign In

Don't have an IoE account? Sign up now

Figure 3.22 web interface for registration server login

	Fire Detection Server Algorithm Algorithm 3-4
1	Setup the TCP connection
2	Setup the UDP connection
3	Initiate IOE connection with smart devices

4	Receive inputs
	S=smoke Detector Level
	IR=Fire Monitor Level
	W=Water Monitor Level

5	If (760< IR <1100) AND (S>40)
6	$\mathcal{F}alrm$= 1
7	Else
8	$\mathcal{F}alarm = 0$
9	End if
10	If $\mathcal{F}alarm = 1$
11	Send ON to sprinkler
12	Setup email (server@security.com)
13	Send Email(Destination: fireEngine@security.com, subject: Alarm, Body: an unusual situation in system1001, please check the location urgently address is building number/street name/town
14	Else if $\mathcal{F}alarm = 0$
15	Send OFF to sprinkler
16	End if
17	If W>20
18	Send ON to Water Drain
19	else if w < 2
20	Send OFF to Water Drain
21	End if
22	Send report to Centralize server using UDP packets
23	Loop to step 4

The following equations represent algorithms 3-3 :

$$\boldsymbol{\mathcal{F}}\mathbf{alarm} = \begin{cases} \mathbf{1} & \mathbf{(760 < IR > 1100)and(S > 40)} \\ \mathbf{0} & \mathbf{else} \end{cases} \quad \ldots\ldots\ldots\ldots \quad \textit{Equation3.1}$$

$$\mathbf{Sp} = \begin{cases} \mathbf{1} & \boldsymbol{\mathcal{F}}\mathbf{alarm = 1} \\ \mathbf{0} & \boldsymbol{\mathcal{F}}\mathbf{alarm = 0} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\text{Equation 3.2}$$

$$\mathbf{Wd} = \begin{cases} \mathbf{1} & \mathbf{w > 20} \\ \mathbf{0} & \mathbf{w < 2} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\text{Equation 3.3}$$

While:

$\boldsymbol{\mathcal{F}alarm}$ is flame detection.

IR is the input of the Fire Monitor.

S is the smoke Detector Value.

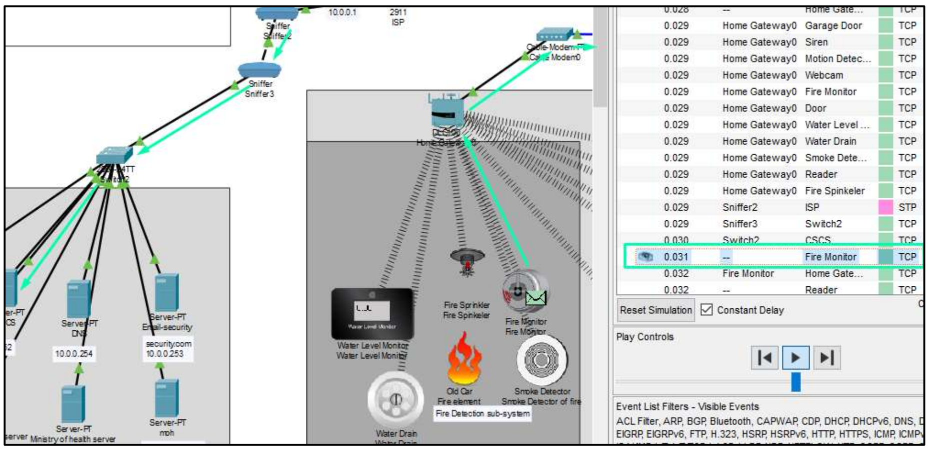
Sp is the status of the sprinkler.

Wd is a water drain status.

S is the smoke Detector Value.

w is water Level.

The communication between devices and CSCS start with sending a TCP connection establishment message to the cluster security control server (CSCS) as shown in figure (3.23). The green line represents the packet path from the device to CSCS. The simulation panel at the right shows the time, destination and, source devices, and protocol.



Time(sec)	Last Device	At Device		Type
0.028	--	Home Gate...	🟩	TCP
0.029	Home Gateway0	Garage Door	🟩	TCP
0.029	Home Gateway0	Siren	🟩	TCP
0.029	Home Gateway0	Motion Detec...	🟩	TCP
0.029	Home Gateway0	Webcam	🟩	TCP
0.029	Home Gateway0	Fire Monitor	🟩	TCP
0.029	Home Gateway0	Door	🟩	TCP
0.029	Home Gateway0	Water Level ...	🟩	TCP
0.029	Home Gateway0	Water Drain	🟩	TCP
0.029	Home Gateway0	Smoke Dete...	🟩	TCP
0.029	Home Gateway0	Reader	🟩	TCP
0.029	Home Gateway0	Fire Spinkeler	🟩	TCP
0.029	Sniffer2	ISP	🟪	STP
0.029	Sniffer3	Switch2	🟩	TCP
0.030	Switch2	CSCS	🟩	TCP
0.031	--	Fire Monitor	🟩	TCP
0.032	Fire Monitor	Home Gate...	🟩	TCP
0.032	--	Reader	🟩	TCP

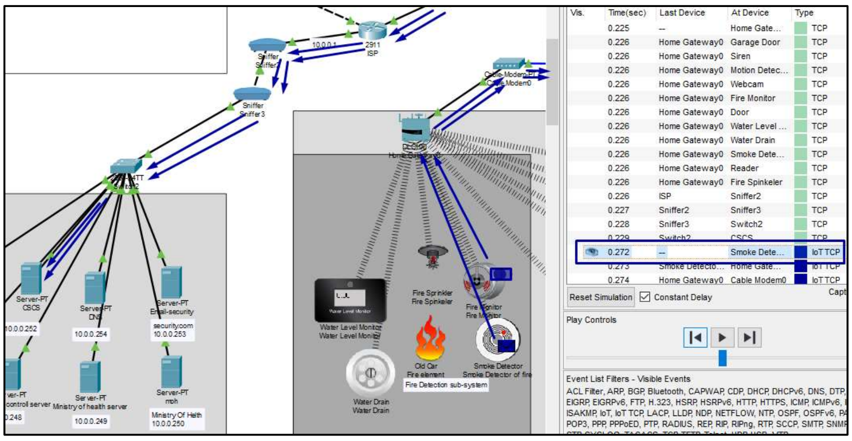
Reset Simulation ☑ Constant Delay

Play Controls

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS,
EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv

Figure 3.23 TCP Connection Establishment between devices CSCS

After the TCP establishment is completed the fire Monitor checking for flames by testing the IR wavelength in the environment if its value between 760 nm and 1100 nm it recognizes that there is a fire, it sends the IR value to the CSCS over TCP. The smoke detector also sends the smoke level value to the server if its value is more than 40. Figure 3.24 shows this process.



Vis.	Time(sec)	Last Device	At Device		Type
	0.225	--	Home Gate...	🟩	TCP
	0.226	Home Gateway0	Garage Door	🟩	TCP
	0.226	Home Gateway0	Siren	🟩	TCP
	0.226	Home Gateway0	Motion Detec...	🟩	TCP
	0.226	Home Gateway0	Webcam	🟩	TCP
	0.226	Home Gateway0	Fire Monitor	🟩	TCP
	0.226	Home Gateway0	Door	🟩	TCP
	0.226	Home Gateway0	Water Level ...	🟩	TCP
	0.226	Home Gateway0	Water Drain	🟩	TCP
	0.226	Home Gateway0	Smoke Dete...	🟩	TCP
	0.226	Home Gateway0	Reader	🟩	TCP
	0.226	Home Gateway0	Fire Spinkeler	🟩	TCP
	0.226	ISP	Sniffer2	🟩	TCP
	0.227	Sniffer2	Sniffer3	🟩	TCP
	0.228	Sniffer3	Switch2	🟩	TCP
	0.229	Switch2	CSCS	🟩	TCP
	0.272	--	Smoke Dete...	🟦	IoT TCP
	0.273	Smoke Detecto...	Home Gate...	🟦	IoT TCP
	0.274	Home Gateway0	Cable Modem0	🟦	IoT TCP

Reset Simulation ☑ Constant Delay

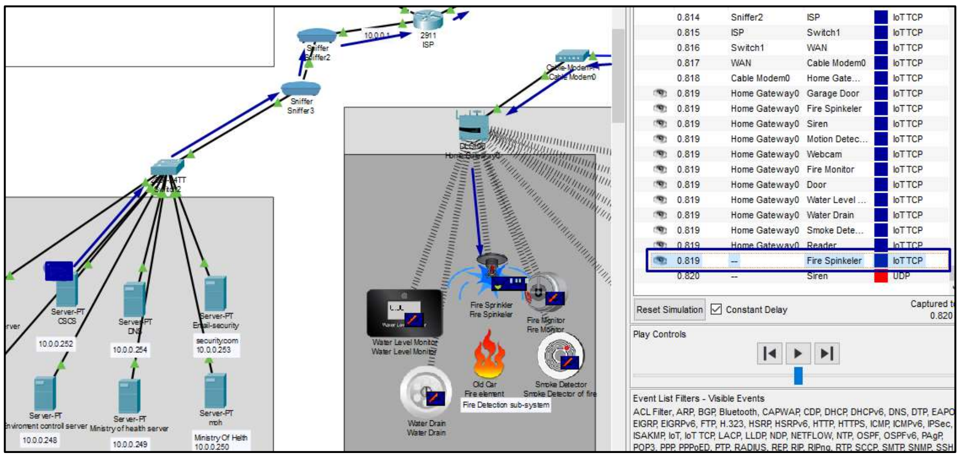
Play Controls

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP,
EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6,
ISAKMP, IoT, IoT TCP, LACP, LLDP, NDP, NETFLOW, NTP, OSPF, OSPFv6,
POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP

Figure 3.24 Fire Monitor and Smoke Detector Send their levels to CSCS

When CSCS checks the received data of Fire monitor input (IR) and if the range from 760nm to 1100nm and if the percentage of smoke in the system environment is more than 40%, the CSCS considers that there is an emergency, so an immediate response is going to be taken as below:

First, turn the sprinkler on by sending an IoE-server change status message over TCP see figure (3.25).



```
0.814  Sniffer2          ISP              IoT TCP
0.815  ISP               Switch1          IoT TCP
0.816  Switch1           WAN              IoT TCP
0.817  WAN               Cable Modem0     IoT TCP
0.818  Cable Modem0      Home Gate...     IoT TCP
0.819  Home Gateway0     Garage Door      IoT TCP
0.819  Home Gateway0     Fire Spinkeler   IoT TCP
0.819  Home Gateway0     Siren            IoT TCP
0.819  Home Gateway0     Motion Detec...  IoT TCP
0.819  Home Gateway0     Webcam           IoT TCP
0.819  Home Gateway0     Fire Monitor     IoT TCP
0.819  Home Gateway0     Door             IoT TCP
0.819  Home Gateway0     Water Level ...  IoT TCP
0.819  Home Gateway0     Water Drain      IoT TCP
0.819  Home Gateway0     Smoke Dete...    IoT TCP
0.819  Home Gateway0     Reader           IoT TCP
0.819  --                Fire Spinkeler   IoT TCP
0.820  --                Siren            UDP
```

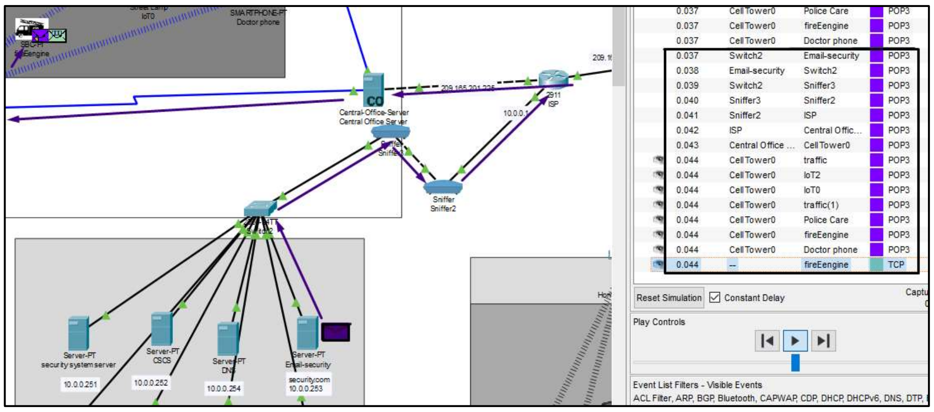
Reset Simulation ☑ Constant Delay Captured t 0.820

Play Controls

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPO
EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec,
ISAKMP, IoT, IoT TCP, LACP, LLDP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP,
POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH

Figure 3.25 The Fire Sprinkler state is ON

Second, sends an email Message to the Fire-engine in order to provide backup to the local system as shown in figure (3.26).



```
0.037  CellTower0        Police Care      POP3
0.037  CellTower0        fireEengine      POP3
0.037  CellTower0        Doctor phone     POP3
0.037  Switch2           Email-security   POP3
0.038  Email-security    Switch2          POP3
0.039  Switch2           Sniffer3         POP3
0.040  Sniffer3          Sniffer2         POP3
0.041  Sniffer2          ISP              POP3
0.042  ISP               Central Offic... POP3
0.043  Central Office ...CellTower0       POP3
0.044  CellTower0        traffic          POP3
0.044  CellTower0        IoT2             POP3
0.044  CellTower0        IoT0             POP3
0.044  CellTower0        traffic(1)       POP3
0.044  CellTower0        Police Care      POP3
0.044  CellTower0        fireEengine      POP3
0.044  CellTower0        Doctor phone     POP3
0.044  --                fireEengine      TCP
```

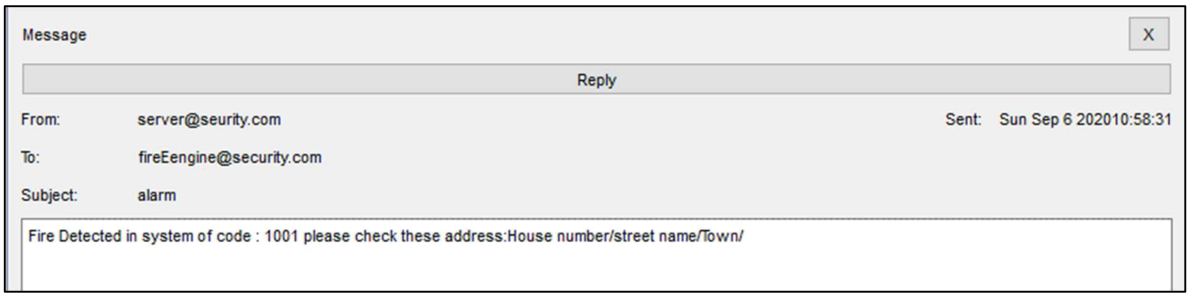
Reset Simulation ☑ Constant Delay Captu

Play Controls

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP,

Figure 3.26 Sending Email Message using POP3 protocol to the Fire Engine

The Fire-engines Receive an email from CSCS including the system code and location so it is directed to this location then updates its GPS location to the server every 5 seconds. Figure (3.27) shows the email format.


Message

Reply

From: server@seurity.com Sent: Sun Sep 6 202010:58:31

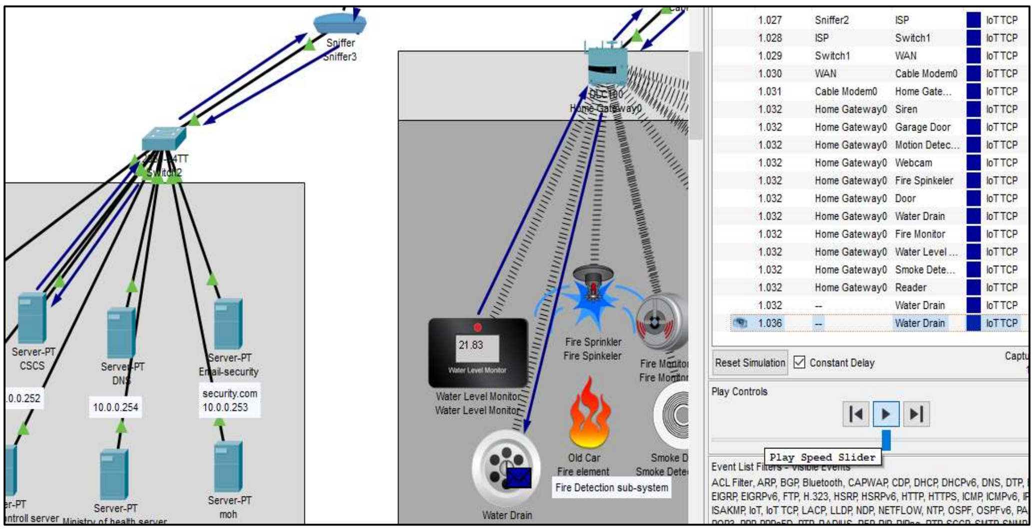
To: fireEengine@security.com

Subject: alarm

Fire Detected in system of code : 1001 please check these address:House number/street name/Town/

Figure 3.27 Email format from Server to Fire engine

At the same time while sprinkler and fire-engine push a large amount of water to cool down the flames, the water level increases without control that may make problems in location so the CSCS checks the water level if it is more than 20 cm it puts on the water drainer until it is less than 2 cm. Figure (3.28) shows this mechanism.



1.027	Sniffer2	ISP	🟦	IoT TCP
1.028	ISP	Switch1	🟦	IoT TCP
1.029	Switch1	WAN	🟦	IoT TCP
1.030	WAN	Cable Modem0	🟦	IoT TCP
1.031	Cable Modem0	Home Gate...	🟦	IoT TCP
1.032	Home Gateway0	Siren	🟦	IoT TCP
1.032	Home Gateway0	Garage Door	🟦	IoT TCP
1.032	Home Gateway0	Motion Detec...	🟦	IoT TCP
1.032	Home Gateway0	Webcam	🟦	IoT TCP
1.032	Home Gateway0	Fire Spinkeler	🟦	IoT TCP
1.032	Home Gateway0	Door	🟦	IoT TCP
1.032	Home Gateway0	Water Drain	🟦	IoT TCP
1.032	Home Gateway0	Fire Monitor	🟦	IoT TCP
1.032	Home Gateway0	Water Level ...	🟦	IoT TCP
1.032	Home Gateway0	Smoke Dete...	🟦	IoT TCP
1.032	Home Gateway0	Reader	🟦	IoT TCP
1.032	--	Water Drain	🟦	IoT TCP
1.036	--	Water Drain	🟦	IoT TCP

Event List Filters - Visible Events

ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, ISAKMP, IoT, IoT TCP, LACP, LLDP, NDP, NETFLOW, NTP, OSPF, OSPFv6, PA

Figure 3.28 Draining the high-level water mechanism

As mentioned in paragraph 3.1 that the speed of system response is very important, so we measure our system response to change of Fire element and record the time value as shown in table (3.4).

Table 3.4 Response time of system components to the Fire element state changes.

Fire Element Status	Change Fire Element Status time(sec.)	Fire Monitor response time(sec.)	Smoke Detector response time(sec.)	Fire Sprinkler change status time(sec.)	Fire Detection system Response time VS change in status(sec.)
off	0.09	1.382	0.868	1.4	1.31
on	1.413	2.489	1.875	2.506	1.093
off	3.642	4.691	3.89	4.709	1.067
on	4.714	5.797	5	5.815	1.101
Off	0.004	0.881	0.115	0.899	0.895
on	0.906	1.970	1.13	1.987	1.081

The whole system response time vs the change in the environment is shown in figure (3.29).

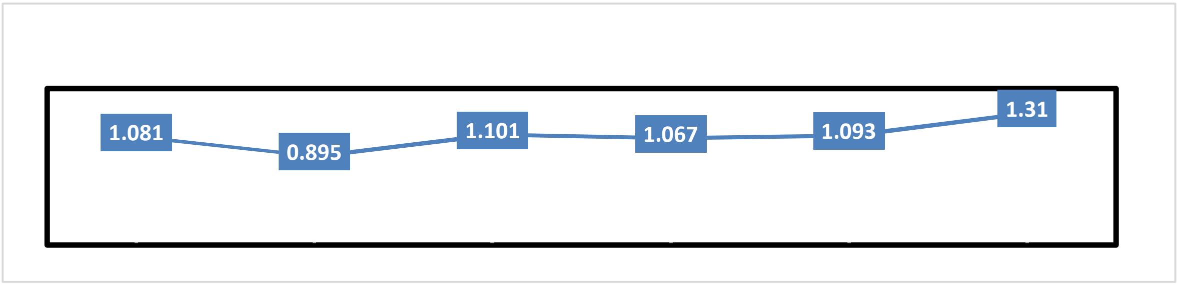


Figure 3.29 Response Time per each status change in the system environment

To calculate the average response time, we use the following equation is used:

$$TR=\frac{\sum_{N-1}^{0} TRn}{N} \quad \text{................. Equation 3.4}$$

While:

TR: Time Response.

TRn: Time response per change.

N: Number of changes in the system environment.

Fire detection TR is: 1.091166667 second

3.5.2 Security Subsystem

The need for life security systems is increased with increment in live complexity and requirements. The IoT makes connecting smart devices easier as ever that gives us the ability to design smart Monitoring and alarming systems that have the flexibility to be installed in any location and send data further forwarded to fog cloud centralized server which analyzes a large amount of data and take real-time reactions.

Our proposed system consists of Motion Detector, RAFID reader, RAFID cards, Webcam, Door, Garage Door, police car, and Siren and all of these devices are provided with a wireless interface of 2.4GHz band which is suitable to connecting them to wireless networks through the home gateway.

The home gateway provides IP addresses to the devices by using DHCP protocol and connect devices to the cluster security control server (CSCS) through the internet. Figure (3.30) shows the topology of the system.

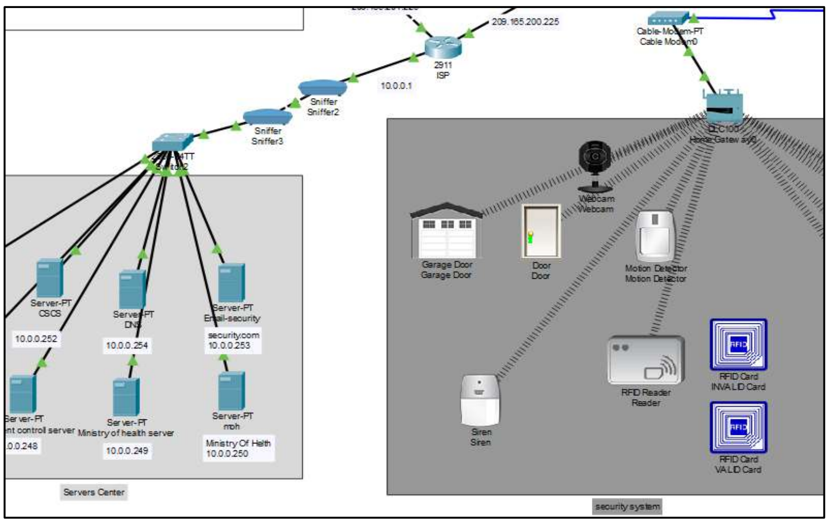
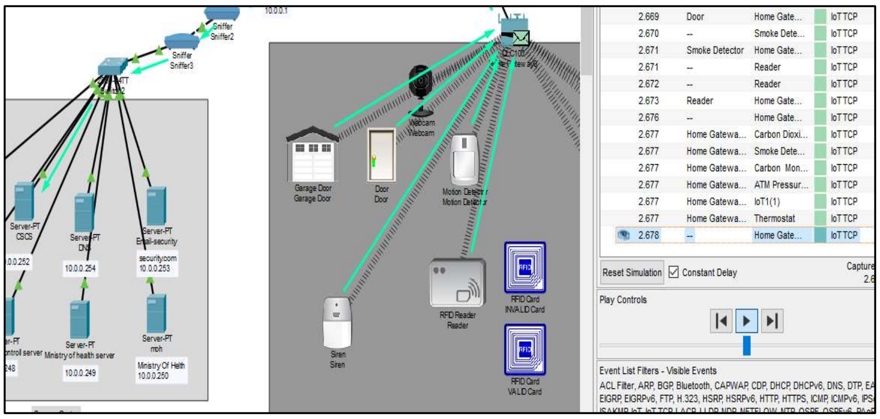


Figure 3.30 Topology of Smart Security system

In this platform, the communication between devices and server is done by sending TCP connection establishments between devices and CSCS. Figure (3.31) shows the TCP connection establishment.



2.669	Door	Home Gate...	🟩	IoT TCP
2.670	--	Smoke Dete...	🟩	IoT TCP
2.671	Smoke Detector	Home Gate...	🟩	IoT TCP
2.671	--	Reader	🟩	IoT TCP
2.672	--	Reader	🟩	IoT TCP
2.673	Reader	Home Gate...	🟩	IoT TCP
2.676	--	Home Gate...	🟩	IoT TCP
2.677	Home Gatewa...	Carbon Dioxi...	🟩	IoT TCP
2.677	Home Gatewa...	Smoke Dete...	🟩	IoT TCP
2.677	Home Gatewa...	Carbon Mon...	🟩	IoT TCP
2.677	Home Gatewa...	ATM Pressur...	🟩	IoT TCP
2.677	Home Gatewa...	IoT1(1)	🟩	IoT TCP
2.677	Home Gatewa...	Thermostat	🟩	IoT TCP
2.678	--	Home Gate...	🟩	IoT TCP

Reset Simulation ☑ Constant Delay Capture 2.6

Play Controls

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EA
EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPS

Figure 3.31 TCP Connection Establishment of Security System

The motion detector sends any discovered motion to CSCS, the RAFID reader also sends its status if there is a RAFID card or Not, and if the card is valid or not, these data are sent on TCP protocol as IoE client's data to the IoE sever figure (3.32).

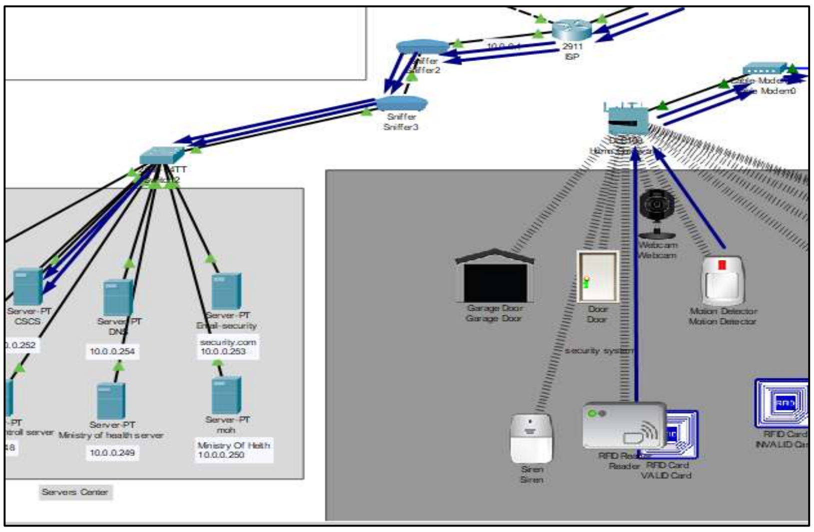


Figure 3.32 Motion detector and RAFID reader sending status to the server

The CSCS checks the incoming data from the Motion detector and RAFID reader according to algorithm 3-4 which mimic the following equations:

$$\text{Ꝛcard} = \begin{cases} \text{NV} & \text{ID} = \text{NULL} \\ \text{NV} & 1000 > \text{ID} < 2000 \\ \text{V} & 1000 < \text{ID} > 2000 \end{cases} \quad \text{……………… Equation 3.5}$$

$$\text{Salarm} = \begin{cases} 1 & \text{M} = 1 \text{ AND Ꝛcard} = \text{NV} \\ 0 & \text{M} = 1 \text{ AND Ꝛcard} = \text{V} \\ 0 & \text{M} = 0 \end{cases} \quad \text{…………… } \textit{Equation 3.6}$$

While:

Ꝛcard: RAFID card Status

ID: Identification number of RAFID card.

NV: Not Valid

V: Valid

M: Motion Detection

Salarm: Security alarm.

Fire Detection Server Algorithm	Algorithm 3-5
1	Setup the TCP connection
2	Setup the UDP connection
3	Initiate IOE connection with smart devices
4	Receive inputs M=Motion Detector ID=RAFID reader
5	If (1000> ID <2000) OR (ID =Null)
6	$Ꝛcard$= NV
7	Else IF (1000< ID > 2000)
8	$Ꝛcard$ = V
9	End if
10	If (($Ꝛcard = NV$) AND (M=1))

11	Salarm = 1
12	Else if (($Rcard = V$) AND (M=1))
13	Salarm = 0
14	Else if (M=0)
15	Salarm = 0
16	End if
17	If (Salarm=1)
18	Send IOE server Message to ON the Siren
20	Send IOE server Message to ON the Webcam
21	Send IOE server Message to LOCK the Door
22	Send IOE server Message to LOCK the Garage Door
23	Send Email(**Destination**: police@security.com, **subject**: Alarm, **Body**: unusual situation in system1001 , please check the location urgently address is building number/street name/town
24	Else if ((Salarm = 0) AND(M=1))
25	Send IOE server Message to OFF the Siren
26	Send IOE server Message to OFF the Webcam
27	Send IOE server Message to UNLOCK the Door
28	Send IOE server Message to OPEN the Garage Door
29	else if ((Salarm = 0) AND(M=0))
30	Send IOE server Message to OFF the Siren
31	Send IOE server Message to OFF the Webcam
32	Send IOE server Message to LOCK the Door
33	Send IOE server Message to LOCK the Garage Door
34	End if
35	Send report to Centralize server as UDP packets
36	Loop to step 4

When the Motion detector finds out a motion and the RAFID Reader read a Not Valid card the CSCS considers that there is an emergency so it switches the Webcam ON to see the Enterer and close the door and the garage door to prevent any stranger from entering. The CSCS also puts the Siren ON to alarm the security guards. Figure (3.33) shows this situation.

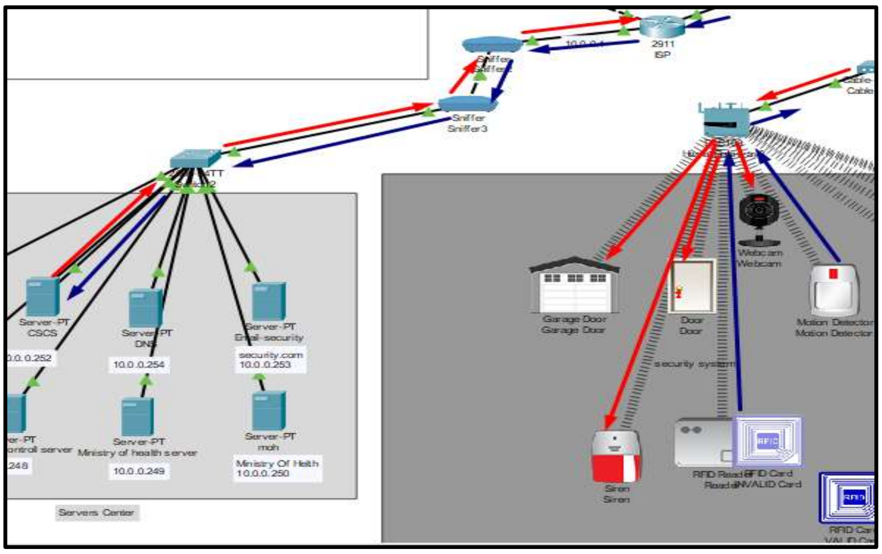
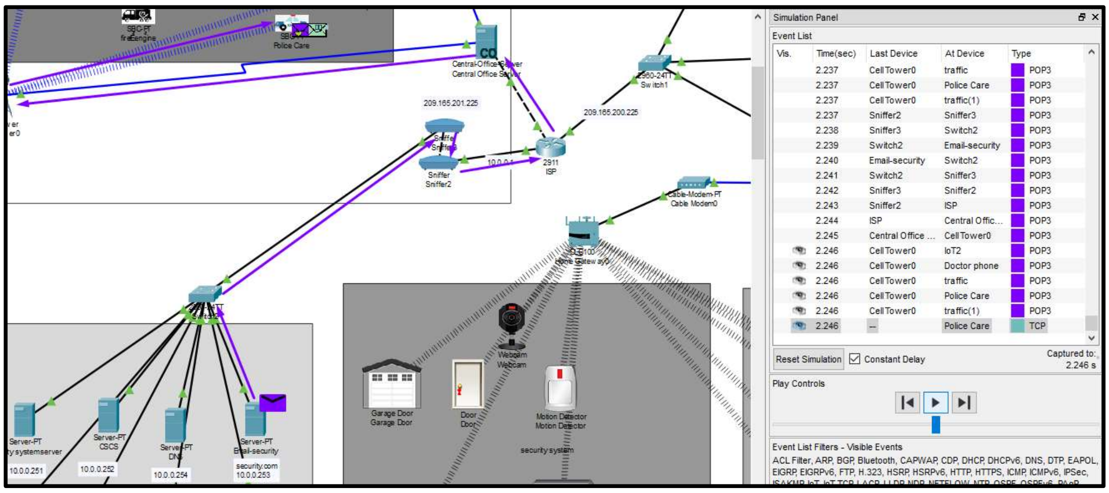


Figure 3.33 CSCS Response to motion detection and invalid card

and lastly, an email sent to the Police cars that there are some problems in this location, so they go directly to the location and manages it. Figure (3.34) shows the email sending mechanism



Vis.	Time(sec)	Last Device	At Device	Type
	2.237	CellTower0	traffic	POP3
	2.237	CellTower0	Police Care	POP3
	2.237	CellTower0	traffic(1)	POP3
	2.237	Sniffer2	Sniffer3	POP3
	2.238	Sniffer3	Switch2	POP3
	2.239	Switch2	Email-security	POP3
	2.240	Email-security	Switch2	POP3
	2.241	Switch2	Sniffer3	POP3
	2.242	Sniffer3	Sniffer2	POP3
	2.243	Sniffer2	ISP	POP3
	2.244	ISP	Central Offic...	POP3
	2.245	Central Office ...	CellTower0	POP3
	2.246	CellTower0	IoT2	POP3
	2.246	CellTower0	Doctor phone	POP3
	2.246	CellTower0	traffic	POP3
	2.246	CellTower0	Police Care	POP3
	2.246	CellTower0	traffic(1)	POP3
	2.246	--	Police Care	TCP

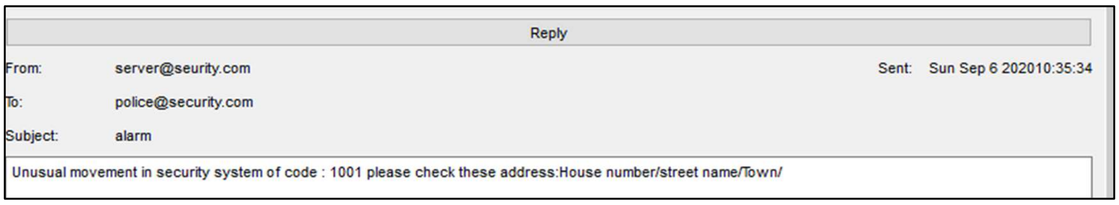
Reset Simulation ☑ Constant Delay Captured to: 2.246 s

Play Controls

Event List Filters - Visible Events
ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec,

Figure 3.34 Email server sending the email to police car over POP3 protocol

The alarming Email format received by the Police care is shown in figure (3.35) .


Reply

From: server@seurity.com Sent: Sun Sep 6 202010:35:34

To: police@security.com

Subject: alarm

Unusual movement in security system of code : 1001 please check these address:House number/street name/Town/

Figure 3.35 Alarming Email Format on Security Emergency

If the status of received inputs in CSCS is Motion Detected True and a valid RAFID card placed in the RAFID reader, it sends an order to open the doors and allow entry for valid persons. The orders are sent using the IoE server to client service over TCP protocol. This procedure is shown in figure (3.36).

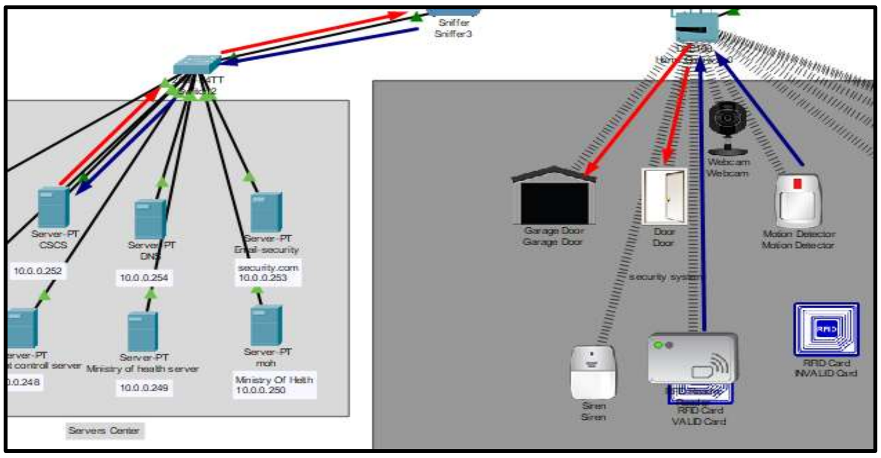
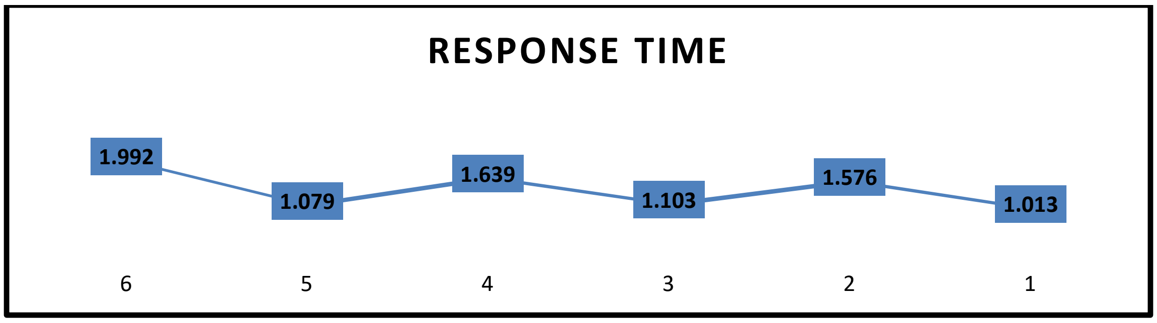


Figure 3.36 CSCS Controlling the Doors to allowing Valid Persons

The whole system response time vs change in the environment is shown in figure (3.37).



3.37 system Response time vs status changes.

To calculate the average response time, The following equation is used:

$$TR=\frac{\sum_{N-1}^{0} \mathbf{TRn}}{\mathbf{N}} \qquad \text{.................. Equation 3.7}$$

While:

TR: Time Response.

TRn: Time response per change.

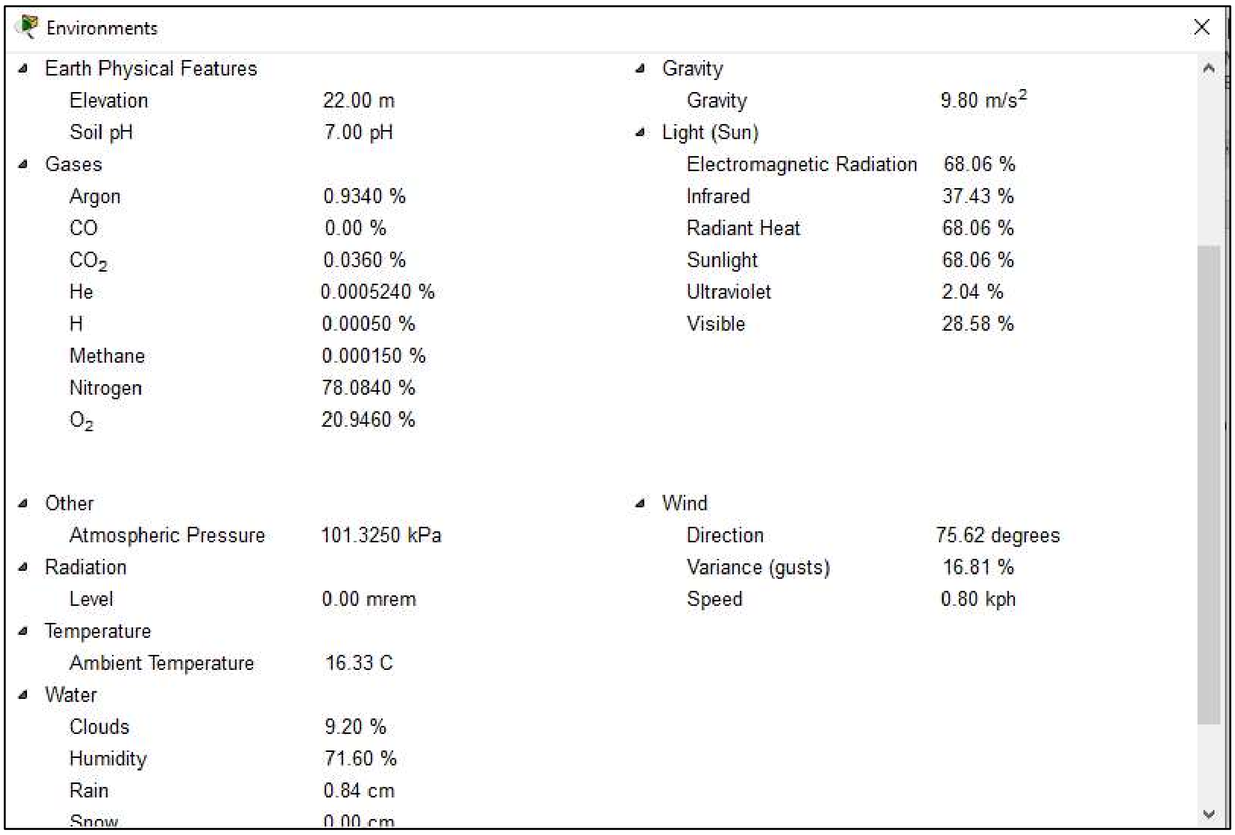
N: Number of changes in the system environment.

Fire detection TR is: 1.400333 second

3.6 Environment control system:

Air quality and environmental pollution are the main factors that pose real environmental challenges. Appropriate supervision is required so that the world can achieve sustainable growth through the preservation of a healthy society. Recently, environmental monitoring, with the developments of the Internet of Things (IoT) and the advancement of digital sensors, has become a smart environment monitoring (SEM) system.[19].

The packet tracer simulator environment allows us to control many environment elements like gases, light, wind, radiation, temperature, and water as shown in figure (3.38). This feature in packet tracer allows us to have an environment close to real.



Environments

Earth Physical Features		Gravity	
Elevation	22.00 m	Gravity	9.80 m/s^2
Soil pH	7.00 pH	Light (Sun)	
Gases		Electromagnetic Radiation	68.06 %
Argon	0.9340 %	Infrared	37.43 %
CO	0.00 %	Radiant Heat	68.06 %
CO_2	0.0360 %	Sunlight	68.06 %
He	0.0005240 %	Ultraviolet	2.04 %
H	0.00050 %	Visible	28.58 %
Methane	0.000150 %		
Nitrogen	78.0840 %		
O_2	20.9460 %		
Other		Wind	
Atmospheric Pressure	101.3250 kPa	Direction	75.62 degrees
Radiation		Variance (gusts)	16.81 %
Level	0.00 mrem	Speed	0.80 kph
Temperature			
Ambient Temperature	16.33 C		
Water			
Clouds	9.20 %		
Humidity	71.60 %		
Rain	0.84 cm		
Snow	0.00 cm		

3.38 Packet tracer environment elements

Smart cities are planned using wireless networks that assist the monitoring of effective wastes management, temperature control, vehicle marking, and pollution control. Therefore, modern methods of environmental monitoring are known as SEM systems, due to the use of IoT, and wireless modern sensors which operate on AI base monitoring and controlling methods. Figure (3.39) shows the smart environment monitoring system topology.

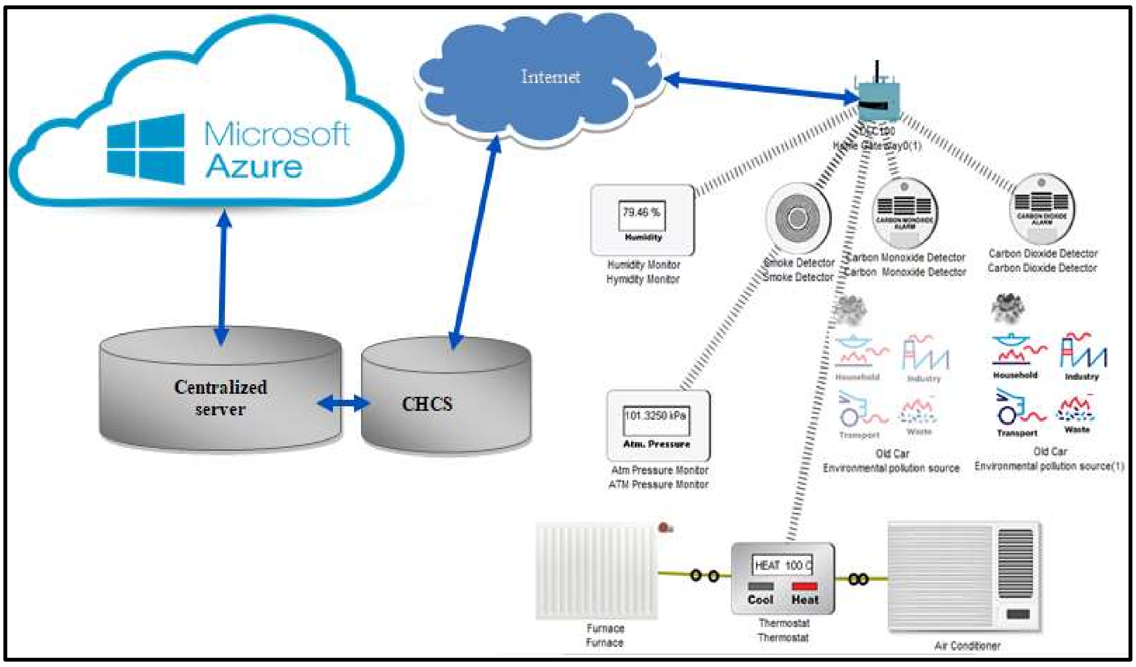


Figure 3.39 Smart Environment system topology.

The proposed system consists of the smart devices explained below.

3.6.1 Humidity Monitor:

Keeping the relative humidity of the environment at an ideal level between 30% and 50% is important for both the health and the condition of buildings[35]. the humidity monitor device is used for reading environment values to determine the humidity. Figure 3.40 shows the device and how it is connected wirelessly to the Home gateway.

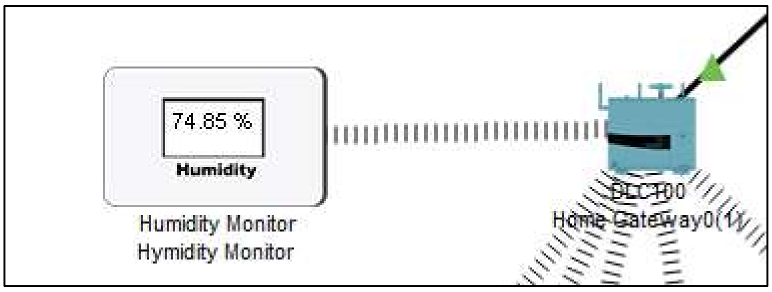


Figure 3.40 Humidity Monitor

3.6.2 Humidifier:

Humidifiers are devices that emit water. Using a humidifier when relative humidity is too low may help alleviate many of the health symptoms associated with dry air[35].

3.6.3 Dehumidifier:

Dehumidifiers work by reducing humidity from buildings, making it a less hospitable environment for allergens like dust, mold, and mildew.

3.6.4 Carbon Monoxide Detector:

Carbon monoxide (CO) is a colorless, non-irritant, odorless, and tasteless toxic gas. It is produced by the incomplete burning of carbonaceous fuels such as wood, petrol, coal, natural gas, and kerosene[36].

If the detector detects a Carbon Monoxide level more than 20% Alarm will go on. The device updates the Registration server with the alarm status and the level of CO.

3.6.5 Carbon Dioxide Detector:

CO2 is a gas produced by exhaling and from the combustion of carbonaceous fuels. The CO2 detector checks the percentage of the gas in the environment if it is more than 60%, alarm goes on.

The device is connected wirelessly to the registration server and update its levels and status continuously.

3.6.6 Thermostat:

This device is responsible for keeping the temperature levels comfortable for humans by controlling the conditioner and heater devices. Figure (3.41) shows the Thermostat.

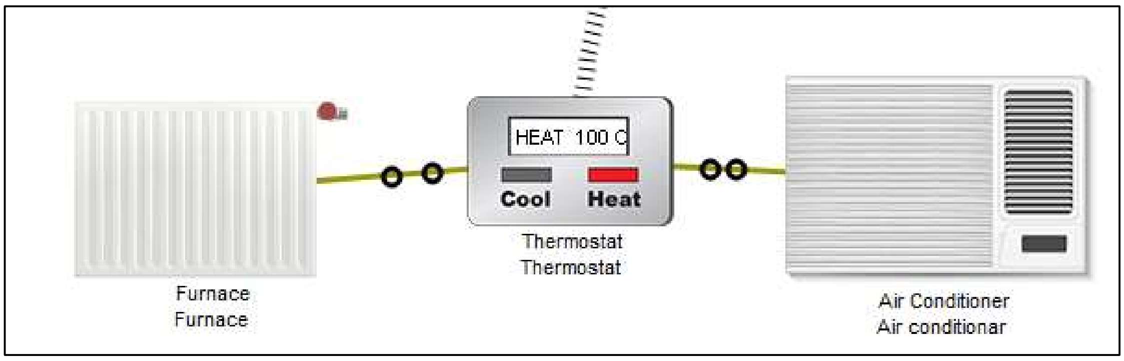
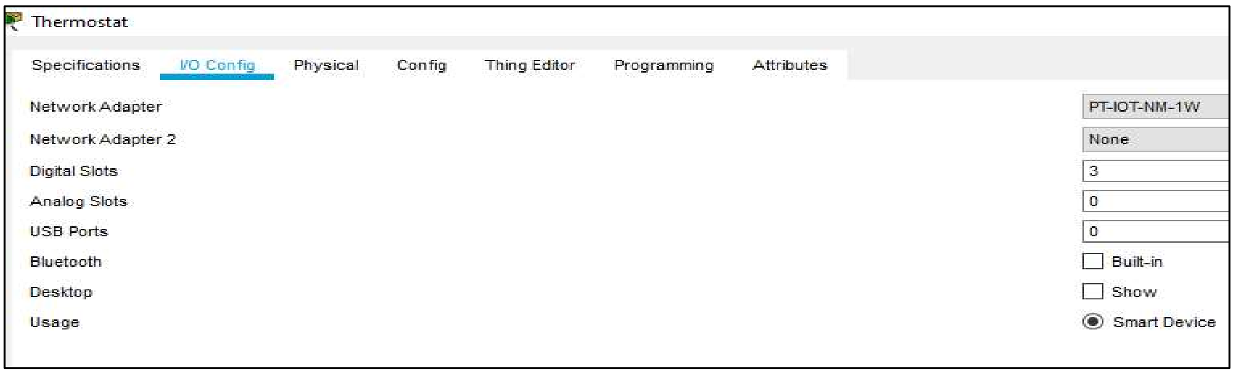


Figure 3.41 Thermostat

As it can be seen in figure 3.42, the thermostat has three digital outputs (two of them used to control heat and cool) and the other used to display the temperature value on the screen it also has a wireless interface connecting it to the gateway.



Network Adapter	PT-IOT-NM-1W
Network Adapter 2	None
Digital Slots	3
Analog Slots	0
USB Ports	0
Bluetooth	☐ Built-in
Desktop	☐ Show
Usage	● Smart Device

Figure 3.42 input/output configuration of the Thermostat

3.6.7 Furnace

It increases the temperature of typical office space at 10C per hour. Also, Reduces the humidity by 2% per hour. It is connected to the D1 slot of the Thermostat to take the control orders ON/OFF.

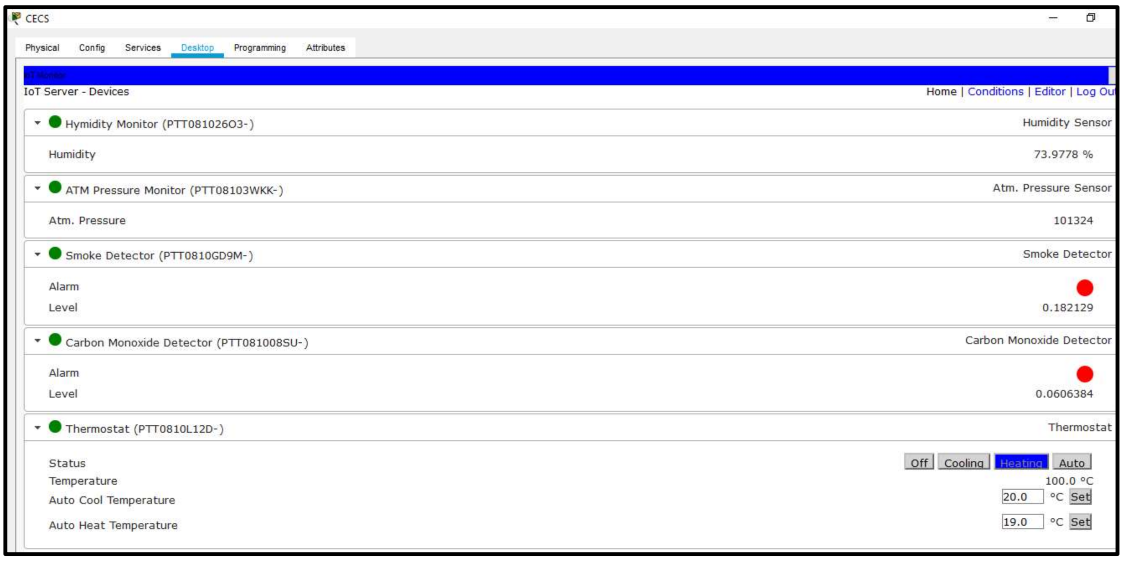
3.6.8 Air Conditioner (AC)

Cools the temperature of typical office space at -10C per hour. Also, Reduces the humidity by 2% per hour.

It is controlled by the thermostat via Connecting it to the D2 slot of the Thermostate also can be controlled by the registration server by connecting to it wirelessly.

3.6.9 Cluster Environment control server (CECS)

Monitoring all the devices of the environment control system is done by the CECS as shown in figure(3.43). The GUI in this figure shows the details of the status of the humidity, ATM pressure, smoke, carbon, and temperature. these environmental details can also be acceded by any smart device like a laptop or smartphone connected to the internet and have an authentication. As well as the CESC transmits the collected data to the Centralized server after adding cluster code to each message.



IoT Server - Devices

● Hymidity Monitor (PTT0810260З-)	Humidity Sensor
Humidity	73.9778 %
● ATM Pressure Monitor (PTT08103WKK-)	Atm. Pressure Sensor
Atm. Pressure	101324
● Smoke Detector (PTT0810GD9M-)	Smoke Detector
Alarm	●
Level	0.182129
● Carbon Monoxide Detector (PTT081008SU-)	Carbon Monoxide Detector
Alarm	●
Level	0.0606384
● Thermostat (PTT0810L12D-)	Thermostat
Status	Off Cooling [✓ Heating] Auto
Temperature	100.0 °C
Auto Cool Temperature	20.0 °C
Auto Heat Temperature	19.0 °C

Figure 3.43 GUI interface of monitoring and controlling environment in CECS

Chapter Four

Implementation of Data Aggregation and Cloud Computing with Results

4.1 Introduction

Data aggregation is responsible for increasing the network lifetime and reducing energy consumption. Nevertheless, each node has the capacity to keep data received from the next nodes or data generated by itself for some time, aggregate them, and send out the aggregated result [8]. So, redundancy is removed from raw data and the communication cost is decreased [37]. The Centralized server in this proposed Smart City platform which lays in the Networking layer of IoE architecture is responsible for the data aggregation process. It is programmed using python 3.8 to perform data preprocessing functions as figure (4.1) shows the blue rectangle represents the centralized server functions and each one will be discussed in detail.

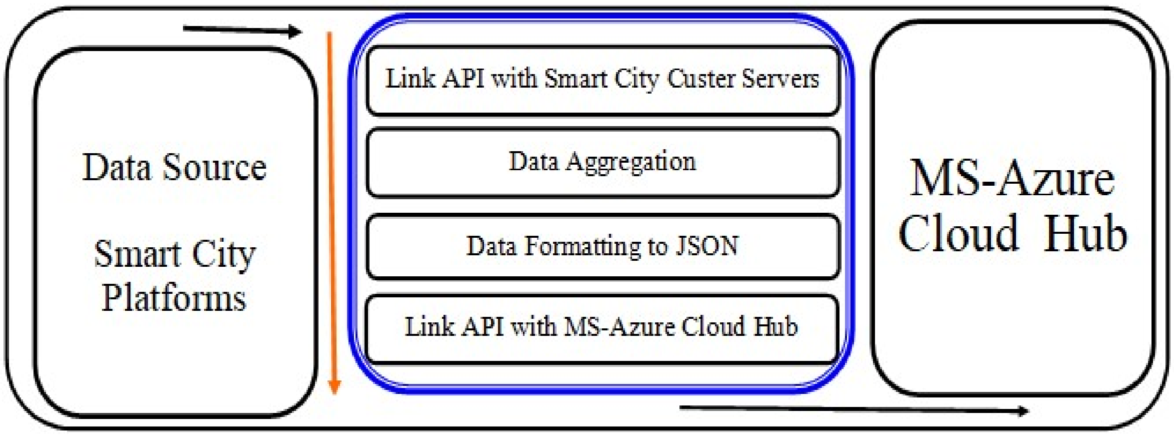
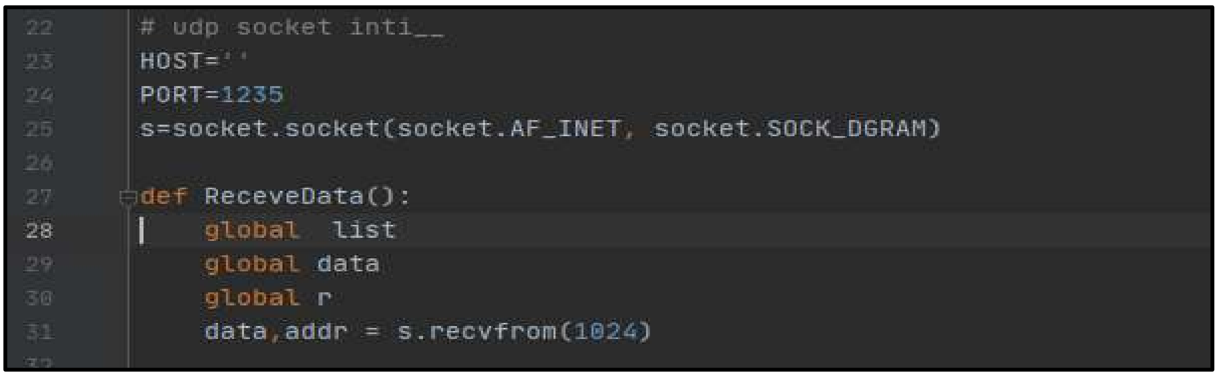


Figure 4.1 Centralized Server Functions

4.2 Link API with Smart City Cluster Servers

The socket is the software abstraction used to represent the "terminals" of a connection between two machines. For a given connection, there is a socket on each machine, working as a virtual hypothetical "cable" running between the two machines with each end of the "cable" plugged into a socket. Minute

We used the UDP-Socket connection to make the link between Cluster servers in the Packet tracer and the Centralized server in Local Host using Python 3.8 programming. Figure (4.2) shows a piece of code.



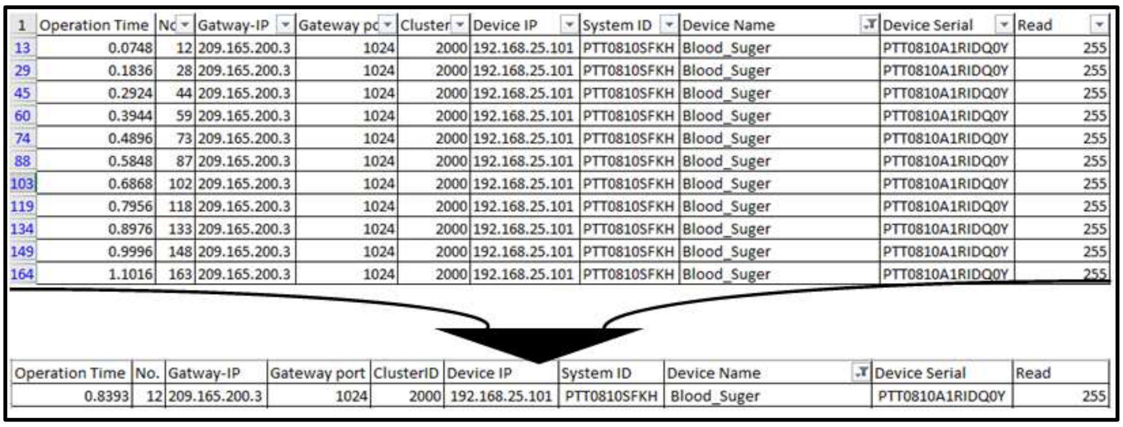
```
# udp socket inti__
HOST=''
PORT=1235
s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

def ReceveData():
    global  list
    global data
    global r
    data,addr = s.recvfrom(1024)
```

Figure 4.2 UDP Socket API piece of code

4.3 Data Aggregation

Data Aggregation as an efficient method helps to improve the efficiency and accuracy of the information, which is achieved via a whole network. Specific redundancy exists in the data gathered from nodes, so this process is required to decrease the unnecessary information. Also, it decreases the traffic load and saves the energy of the nodes. As it can be seen in Figure (4.3) in about one minute the server received 11 messages from the Blood _Sugar Meter for the Same Patient. The Aggregation reduced all these massages to one message which will be sent to the cloud.



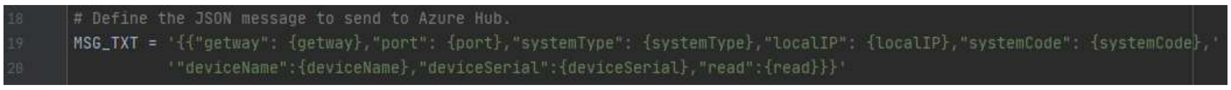
1	Operation Time	No	Gatway-IP	Gateway port	Cluster	Device IP	System ID	Device Name	Device Serial	Read
13	0.0748	12	209.165.200.3	1024	2000	192.168.25.101	PTT0810SFKH	Blood_Suger	PTT0810A1RIDQ0Y	255
29	0.1836	28	209.165.200.3	1024	2000	192.168.25.101	PTT0810SFKH	Blood_Suger	PTT0810A1RIDQ0Y	255
45	0.2924	44	209.165.200.3	1024	2000	192.168.25.101	PTT0810SFKH	Blood_Suger	PTT0810A1RIDQ0Y	255
60	0.3944	59	209.165.200.3	1024	2000	192.168.25.101	PTT0810SFKH	Blood_Suger	PTT0810A1RIDQ0Y	255
74	0.4896	73	209.165.200.3	1024	2000	192.168.25.101	PTT0810SFKH	Blood_Suger	PTT0810A1RIDQ0Y	255
88	0.5848	87	209.165.200.3	1024	2000	192.168.25.101	PTT0810SFKH	Blood_Suger	PTT0810A1RIDQ0Y	255
103	0.6868	102	209.165.200.3	1024	2000	192.168.25.101	PTT0810SFKH	Blood_Suger	PTT0810A1RIDQ0Y	255
119	0.7956	118	209.165.200.3	1024	2000	192.168.25.101	PTT0810SFKH	Blood_Suger	PTT0810A1RIDQ0Y	255
134	0.8976	133	209.165.200.3	1024	2000	192.168.25.101	PTT0810SFKH	Blood_Suger	PTT0810A1RIDQ0Y	255
149	0.9996	148	209.165.200.3	1024	2000	192.168.25.101	PTT0810SFKH	Blood_Suger	PTT0810A1RIDQ0Y	255
164	1.1016	163	209.165.200.3	1024	2000	192.168.25.101	PTT0810SFKH	Blood_Suger	PTT0810A1RIDQ0Y	255

Operation Time	No.	Gatway-IP	Gateway port	ClusterID	Device IP	System ID	Device Name	Device Serial	Read
0.8393	12	209.165.200.3	1024	2000	192.168.25.101	PTT0810SFKH	Blood_Suger	PTT0810A1RIDQ0Y	255

Figure 4.3 Effect of Data Aggregation in removing data redundancy

4.4 Data Formatting to JSON

The data received from the smart city is in CSV format as mentioned in Chapter three. After aggregating this data, the result should convert to the JSON format in order to be understandable in the cloud. Figure (4.4) shows the message format.



```
# Define the JSON message to send to Azure Hub.
MSG_TXT = '{{"getway": {getway},"port": {port},"systemType": {systemType},"localIP": {localIP},"systemCode": {systemCode},'
          '"deviceName":{deviceName},"deviceSerial":{deviceSerial},"read":{read}}}'
```

Figure 4.4 Message format

4.5 Link API with Azure Microsoft HUB

In order to upload the messages to the cloud, Microsoft provides a simple example with Copyright (c) Microsoft contain the uploading instructions. The useful part of this code that is compatible with our program is used. Below is the main instruction of this API.

```
from azure.iot.device import IoTHubDeviceClient, Message # Import
Azure library
CONNECTION_STRING = "HostName=smartcitydata.azure-
devices.net;DeviceId=MyPythonDevice;SharedAccessKey=aT6fbILt//Y/WklB3
ULudJa2RuMYdGC3yoBN3KGlrh4="  # device identification string
client = iothub_client_init()  # Identifies Azur-Hub Client
client.send_message(message)   # Send Client Message to Hub
```

4.6 Centralized Server Functionalities

The flow of data processing in this server is considered as a sequence of three ordered steps, each possibly involving a series of activities on these constituents, as follows:

Preparation of the raw data A data aggregation process starts with preparing the raw data required for the aggregation. This step may involve the locating, extraction, transportation, and normalization of raw data, if necessary.

Aggregation of the raw data An aggregate function is applied by the aggregator that transforms the raw data into the aggregated data.

Post-handling of the aggregated data may be further handled by the aggregator, for instance, saved into persistent storage or provided for other processes. Figure (4.5) shows the flow chart of the centralized server algorithm.

The flow chart is based on the algorithm of aggregating data by removing the duplicity from streaming IoT messages in real-time. The concept of this algorithm depends on making a comparison between the Number Of Messages(N) which is called Window, that are stored in temporary dynamic cash. When the server starts receiving data it sends the N number of messages directly to the cloud and store a copy in the cash. After that, each new message, is compared with all N messages in cash and if matching is found the cash pops the old value and inserts the new message at the end of the cash. otherwise, the algorithm sends the message to the cloud and then updates the cash by removing the last reading of the same device and insert the new reading at the end of the cash.

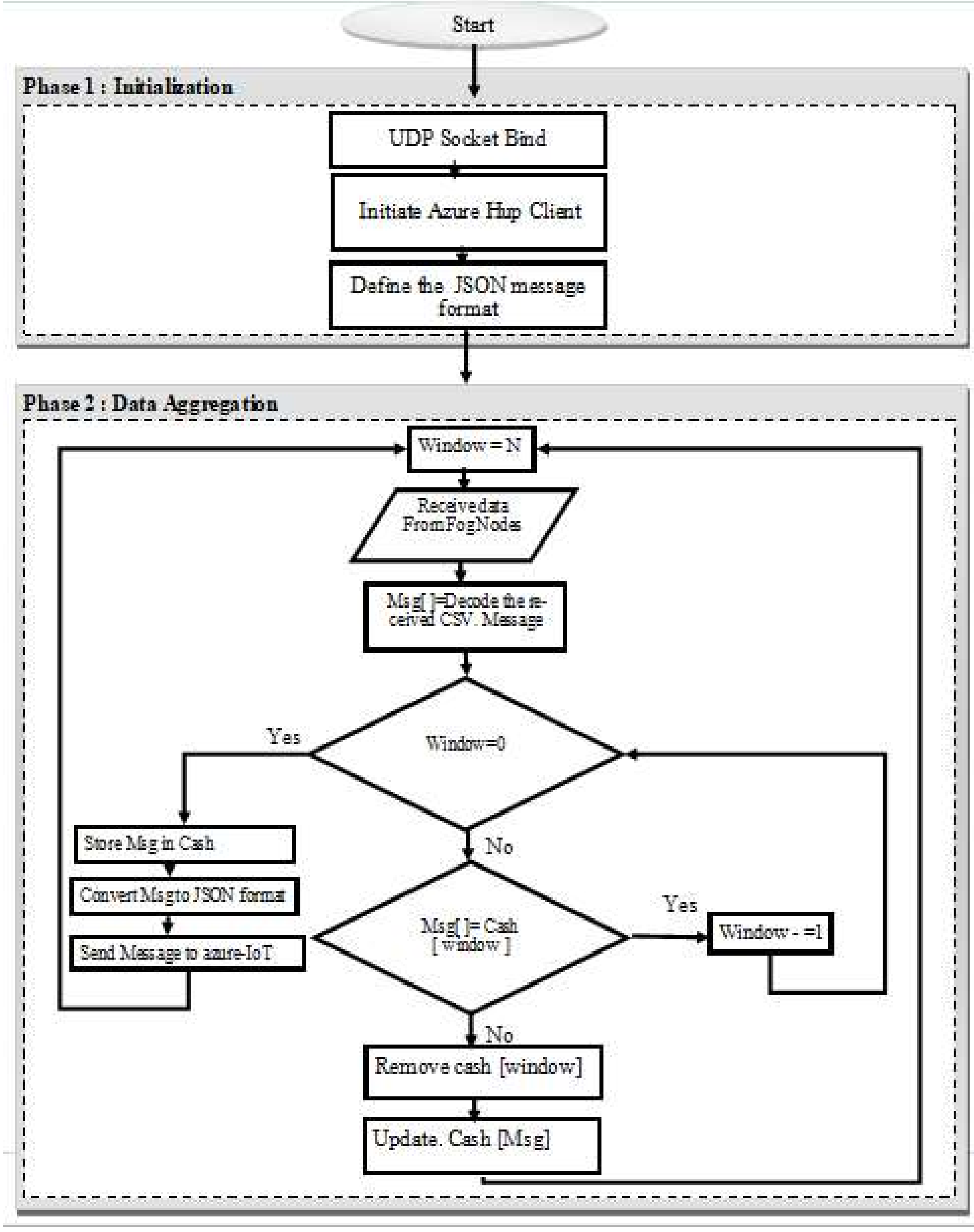


Figure 4.5 Centralized server Flow chart

4.7 Results of Aggregation

Our aggregation algorithm is tested to find out the best window size for the proposed system to gain the best data quality and size.

Table 4.1 Aggregation Percentage for All Platforms

Windows Size	Before Aggregation	After Aggregation	Aggregation Percentage
15	4,314	447	89.64%
20	4,524	541	88.04%
25	4,409	393	91.09%
30	3,636	350	90.37%

As can be noticed from Table 4.1 The best aggregation percentage was gained at a Window size of 25 messages regarding data integrity. To test the algorithm of the different types of devices connected to the network two devices are taken. The first one has almost stable inputs most of the time (ATM pressure meter) from the environment cluster, and the second device has varied inputs (Blood-sugar meter) from the health care cluster. Tables (4.2) and (4.3) contain the aggregation percentages for each device.

Table 4.2 ATM Pressure Aggregation (almost stable Inputs)

Windows Size	Before Aggregation	After Aggregation	Aggregation Percentage
15	280	1	99.64%
20	270	2	99.26%
25	275	2	99.27%
30	239	2	99.16%

Table 4.3 Blood sugar Meter Aggregation (vary Reads)

Windows Size	Before Aggregation	After Aggregation	Aggregation Percentage
15	298	20	93.29%
20	285	19	93.33%
25	285	17	94.04%
30	212	21	90.09%

As can be noticed from the values in the table (4.2) the aggregation percentage is high, and the best percentage was gained in the window of size 15. In the case of varied inputs in the table (4.3), the best aggregation percentage is obtained at window size 25. Meanwhile, most of our devices provide varied inputs, so a window size of 25 was dependent.

Figure (4.6) shows the number of messages per minute for each of the designed prototypes before and after aggregation. It can be noticed that there are considerable reductions in the number of messages sent by the security cluster and this is due to the fact that events for security are mostly static .

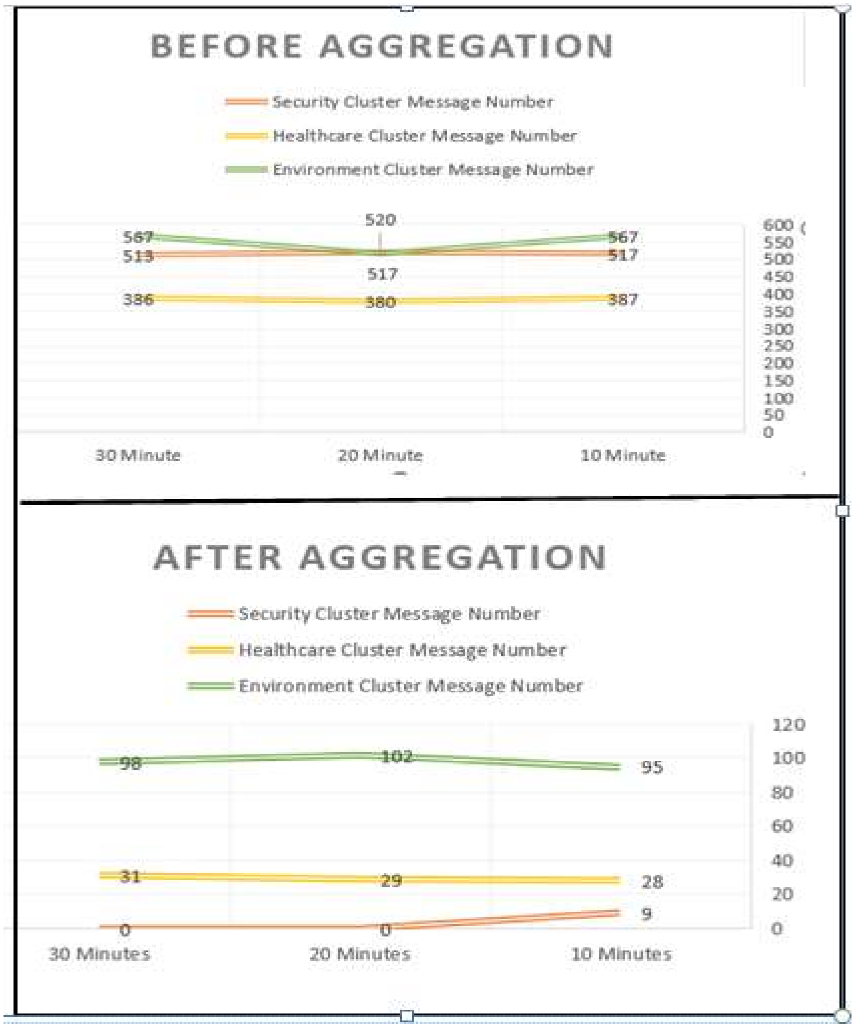


Figure 4.6 Effect of Aggregation on the Number of Messages

The results of aggregation for the prototype are viewed in figure (4.7). The proposed algorithm helps in reducing the number of messages by removing redundancy and duplicity with saving the data integrity.

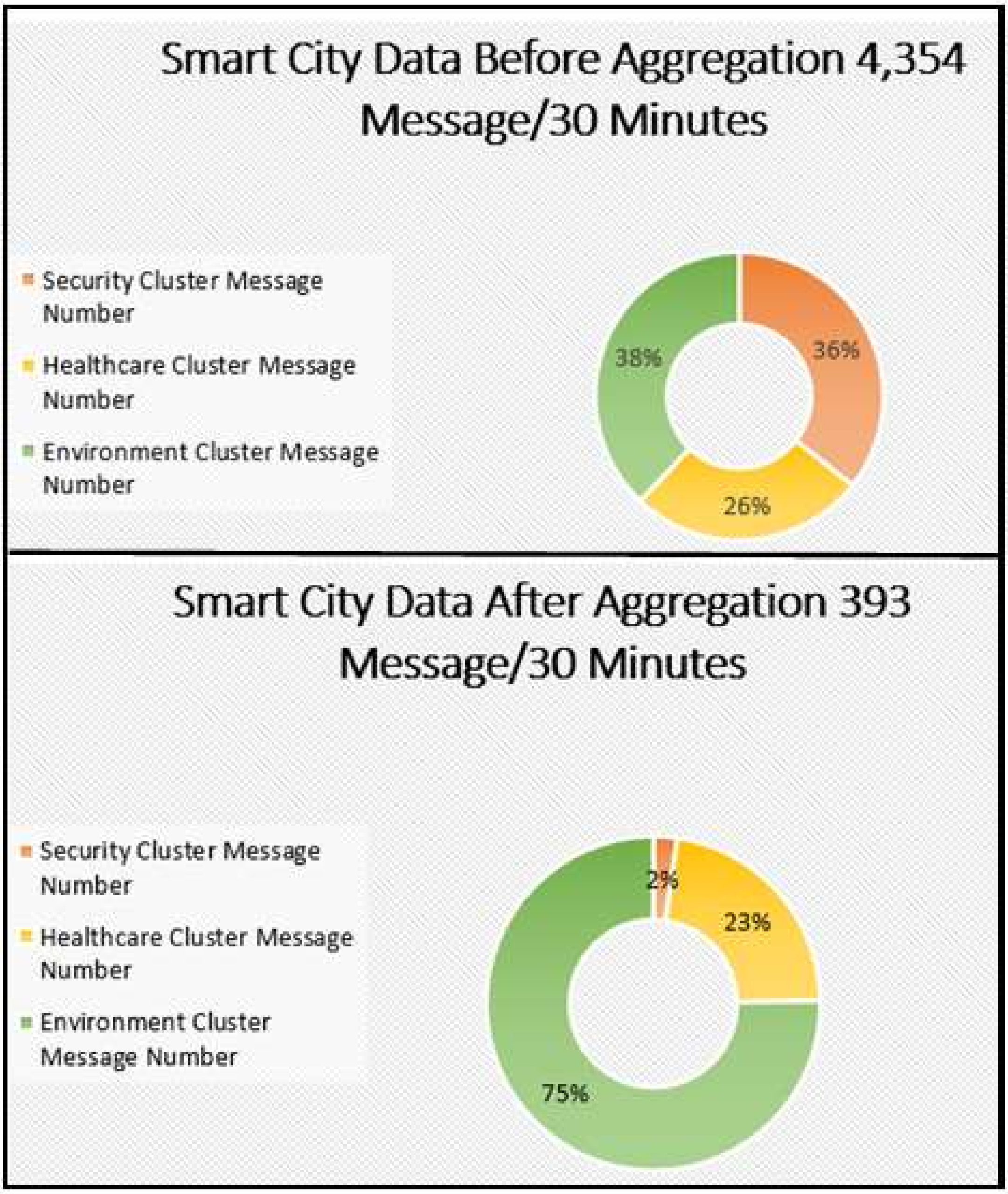
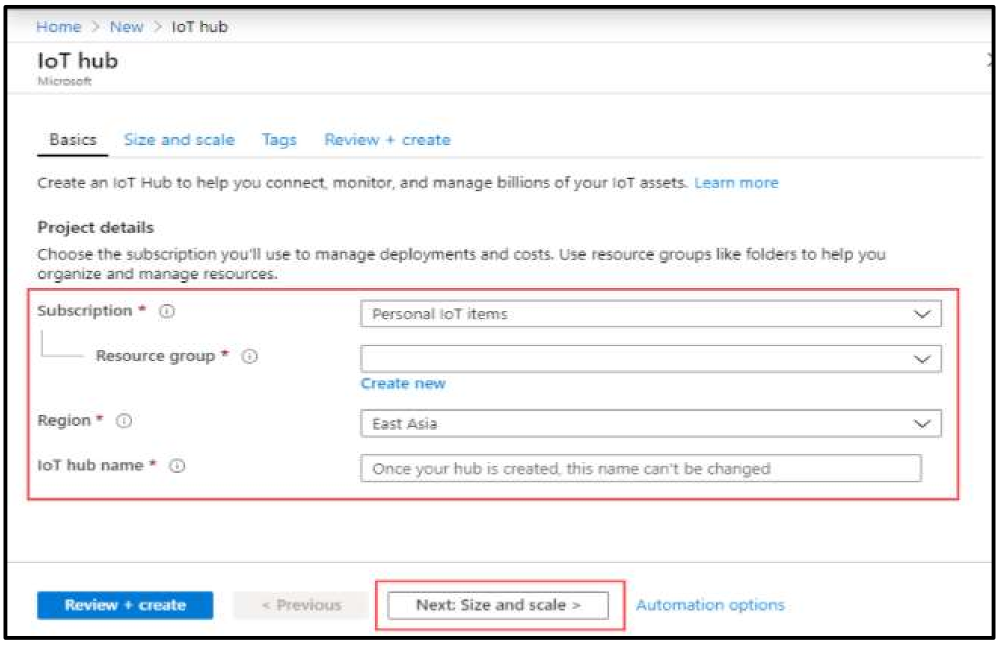


Figure 4.7 Statistical comparison of Number of Messages Before and After Aggregation

4.8 Microsoft Azur Cloud HUB

IoT Hub is an Azure service that enables the client to ingest high volumes of telemetry from IoT devices into the cloud for storage or processing. To be connected to the cloud hub, some prerequisites must be fulfilled like creating an Azure account with an active subscription and Python 3.7+ project, also checking if Port 8883 is open in the client firewall because the device uses MQTT protocol, which communicates over port 8883.

To create an IoT hub using the Azure portal from the Azure homepage, the client must select the (Create a resource) button, fill in all information required as shown in Figure (4.8).


Home > New > IoT hub

IoT hub
Microsoft

Basics | Size and scale | Tags | Review + create

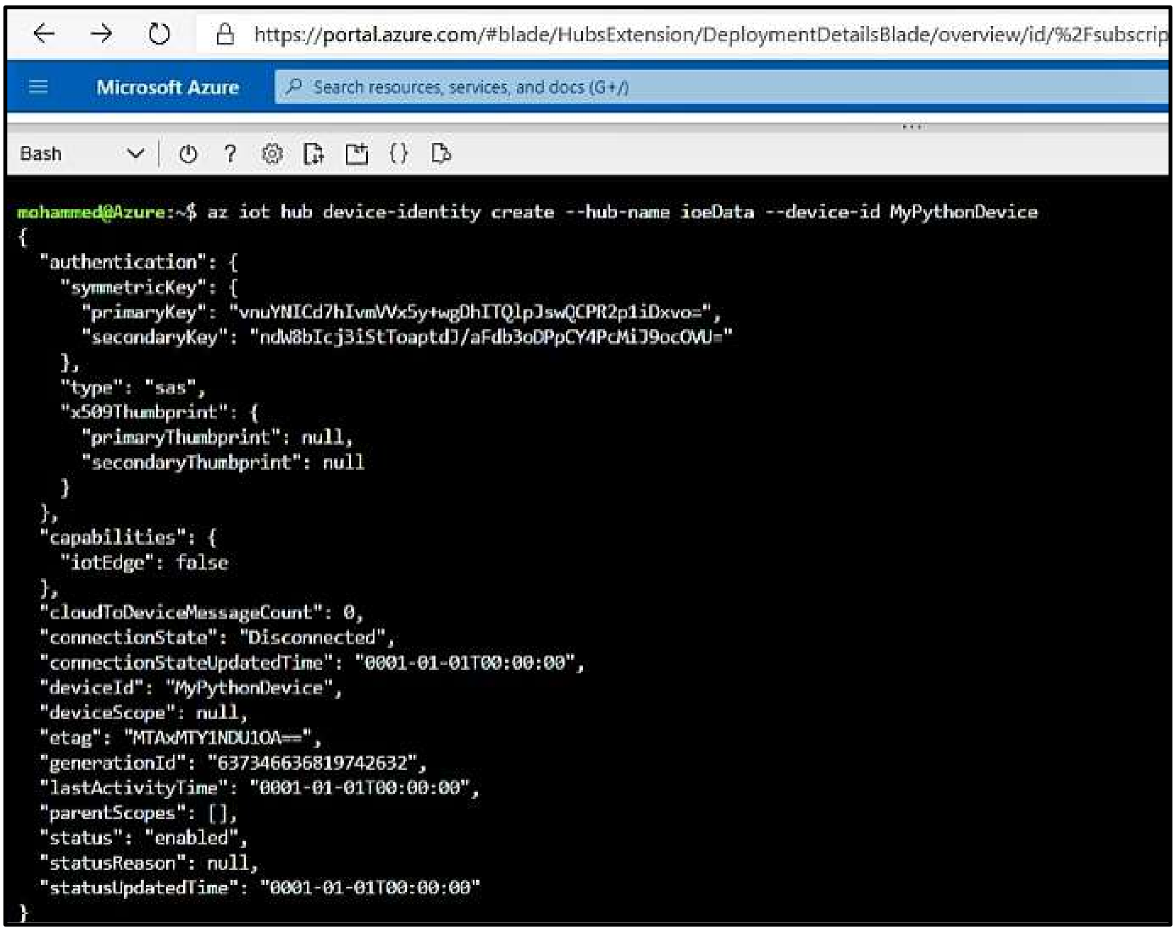
Create an IoT Hub to help you connect, monitor, and manage billions of your IoT assets. Learn more

Project details
Choose the subscription you'll use to manage deployments and costs. Use resource groups like folders to help you organize and manage resources.

Subscription *	Personal IoT items
Resource group *	
	Create new
Region *	East Asia
IoT hub name *	Once your hub is created, this name can't be changed

Figure 4.8 IoT hub creation window

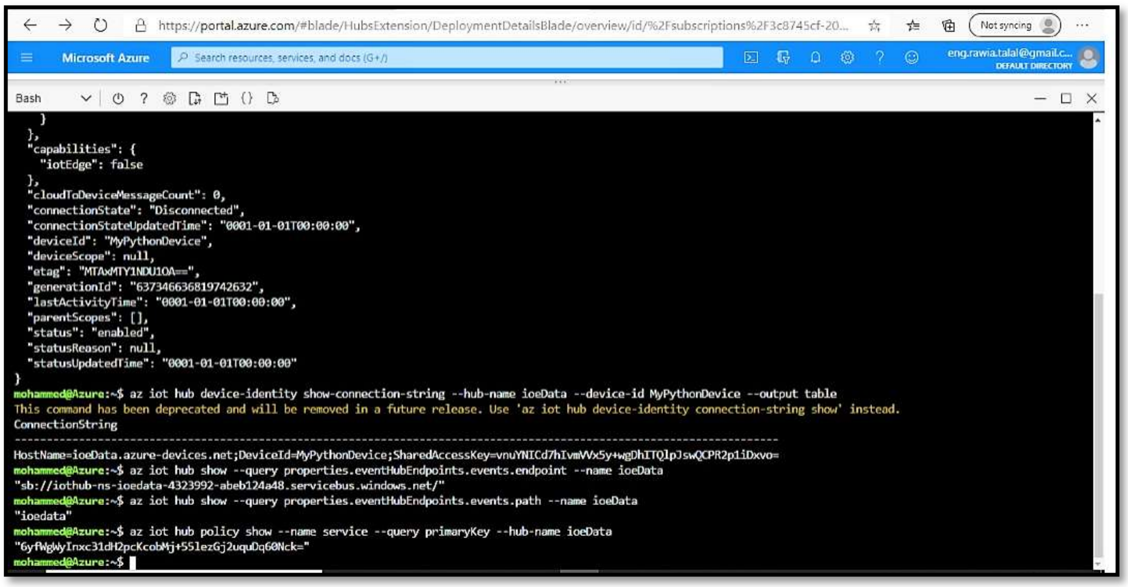
After completing this step we submitted our device as a client in this Hub by using the cloud shell command line. Figure (4.9) shows the commands of creating a client to the hub.


https://portal.azure.com/#blade/HubsExtension/DeploymentDetailsBlade/overview/id/%2Fsubscrip
Microsoft Azure
Search resources, services, and docs (G+/)
Bash

```
mohammed@Azure:~$ az iot hub device-identity create --hub-name ioeData --device-id MyPythonDevice
{
  "authentication": {
    "symmetricKey": {
      "primaryKey": "vnuYNICd7hIvmVVx5y+wgDhITQlpJswQCPR2p1iDxvo=",
      "secondaryKey": "ndW8bIcj3iStToaptdJ/aFdb3oDPpCY4PcMiJ9ocOVU="
    },
    "type": "sas",
    "x509Thumbprint": {
      "primaryThumbprint": null,
      "secondaryThumbprint": null
    }
  },
  "capabilities": {
    "iotEdge": false
  },
  "cloudToDeviceMessageCount": 0,
  "connectionState": "Disconnected",
  "connectionStateUpdatedTime": "0001-01-01T00:00:00",
  "deviceId": "MyPythonDevice",
  "deviceScope": null,
  "etag": "MTAxMTY1NDU1OA==",
  "generationId": "637346636819742632",
  "lastActivityTime": "0001-01-01T00:00:00",
  "parentScopes": [],
  "status": "enabled",
  "statusReason": null,
  "statusUpdatedTime": "0001-01-01T00:00:00"
}
```

Figure 4.9 commands of creating a client to the hub in the cloud shell

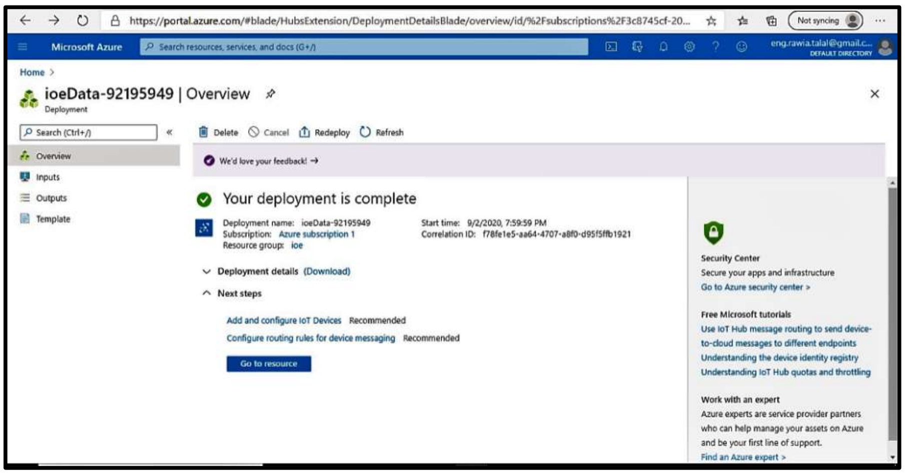
After that, the device information in the hub such as the (connection string) was retrieved by us. Figure (4.10) shows the information of the centralized server device. This information is important for identifying the local device to the IoT Hub.


https://portal.azure.com/#blade/HubsExtension/DeploymentDetailsBlade/overview/id/%2Fsubscriptions%2F3c8745cf-20...
Not syncing
Microsoft Azure
Search resources, services, and docs (G+/)
eng.rawia.talal@gmail.c...
DEFAULT DIRECTORY
Bash

```
    }
  },
  "capabilities": {
    "iotEdge": false
  },
  "cloudToDeviceMessageCount": 0,
  "connectionState": "Disconnected",
  "connectionStateUpdatedTime": "0001-01-01T00:00:00",
  "deviceId": "MyPythonDevice",
  "deviceScope": null,
  "etag": "MTAxMTY1NDU1OA==",
  "generationId": "637346636819742632",
  "lastActivityTime": "0001-01-01T00:00:00",
  "parentScopes": [],
  "status": "enabled",
  "statusReason": null,
  "statusUpdatedTime": "0001-01-01T00:00:00"
}
mohammed@Azure:~$ az iot hub device-identity show-connection-string --hub-name ioeData --device-id MyPythonDevice --output table
This command has been deprecated and will be removed in a future release. Use 'az iot hub device-identity connection-string show' instead.
ConnectionString
------------------------------------------------------------------------------------------------------------------------
HostName=ioeData.azure-devices.net;DeviceId=MyPythonDevice;SharedAccessKey=vnuYNICd7hIvmVVx5y+wgDhITQlpJswQCPR2p1iDxvo=
mohammed@Azure:~$ az iot hub show --query properties.eventHubEndpoints.events.endpoint --name ioeData
"sb://iothub-ns-ioedata-4323992-abeb124a48.servicebus.windows.net/"
mohammed@Azure:~$ az iot hub show --query properties.eventHubEndpoints.events.path --name ioeData
"ioedata"
mohammed@Azure:~$ az iot hub policy show --name service --query primaryKey --hub-name ioeData
"6yfWgWyInxc31dH2pcKcobMj+55lezGj2uquDq60Nck="
mohammed@Azure:~$
```

Figure 4.10 The information of the client device in the Azure IoT hub

Finally, The cloud subscription for the smart city data got an IoT hub containing one client as shown in Figure (4.11).


https://portal.azure.com/#blade/HubsExtension/DeploymentDetailsBlade/overview/id/%2Fsubscriptions%2F3c8745cf-20...
Not syncing
Microsoft Azure
Search resources, services, and docs (G+/)
eng.rawia.talal@gmail.c...
DEFAULT DIRECTORY
Home >
Search (Ctrl+/)
Overview
Inputs
Outputs
Template
Delete
Cancel
Redeploy
Refresh
We'd love your feedback! →
Go to resource

ioeData-92195949 | Overview

Deployment

Your deployment is complete

Deployment name: ioeData-92195949
Subscription: Azure subscription 1
Resource group: ioe
Start time: 9/2/2020, 7:59:59 PM
Correlation ID: f78fe1e5-aa64-4707-a8f0-d95f5ffb1921

Deployment details (Download)

Next steps

Add and configure IoT Devices Recommended
Configure routing rules for device messaging Recommended

Security Center
Secure your apps and infrastructure
Go to Azure security center >

Free Microsoft tutorials
Use IoT Hub message routing to send device-to-cloud messages to different endpoints
Understanding the device identity registry
Understanding IoT Hub quotas and throttling

Work with an expert
Azure experts are service provider partners who can help manage your assets on Azure and be your first line of support.
Find an Azure expert >

Figure 4.11 deployment of the centralized server in the Azur IoT Hub

Now, the hub starts to receive the messages sent by the Smart city Centralized server as shown in figure (4.12).

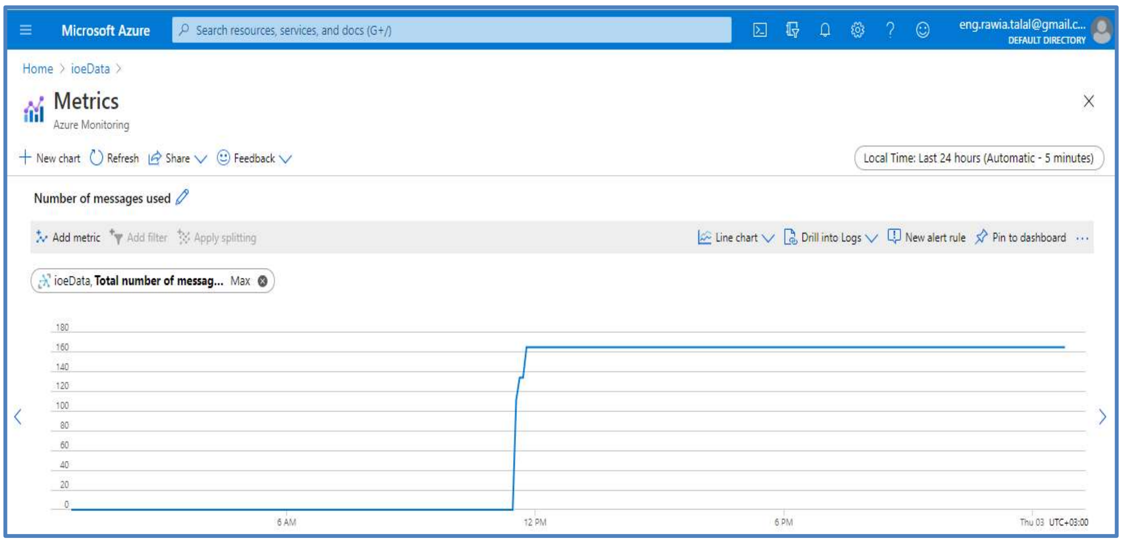


Figure 4.12 Smart city data at the Azure Hub

These messages can be routed to any cloud service in our case we rout it to the cloud storage in order to be ready for more analysis and mining.

Chapter Five

Conclusions and Challenges

5.1 Conclusion

After finishing this work and after overcoming challenges, the following conclusions can be deduced.

1. Today the world lives the four phases of the Internet Where this phase is called Internet of Things/Internet of Everything (IoT/IoE). The Smart City is one of the major fields still the research tackling it to complete the standards, develop the technologies, enhance the performance, and to extend the applications. This work comes to contribute to this direction.
2. The proposed smart city prototype which includes three service platforms is flexible and scalable. Flexible because services and functions within each platform can be easily monitored, controlled and managed. Scalable because any other platforms and any other services within the platform can be added without interrupting the whole system.
3. Enhancing the performance is achieved by programming an aggregation algorithm and appling it to be as the edge computing in the IoE infrastructure. This results in saving bandwidth and reducing latency in networking.
4. Machine to Machine (M2M) and Person to Machine (P2M) are two IoE pillars realized in this smart city prorotype.

5. The IoE architecture is applied with its three layers, IoT layer, network layer, and cloud layer, satisfying the software and hardware requirements for real-time operation.
6. fog computing provides notification service at the edge of the network which contributes to avoid critical situation.
7. Advantages like availability, low cost, and scalability are gained since fog computing stores the aggregated information in distributed storage.
8. The Packet Tracer 7.2 is used to design the three platforms because it is an efficient tool to simulate different network topology using wired and wireless connections and run the different protocols simultaneously. The high degree of accuracy in simulating Cisco's IOS by packet tracer enables efficient simulation for many information systems, like servers and terminals, in addition to simulating the concept of the Internet of Everything (IoE). Packet tracer support three programming languages JavaScript, python, and Blockly programming. All that is customized by a graphical user interface (GUI) and allows contribution for multi-users activities. All of this makes it a powerful tool to simulate complex and inter-protocols scenarios.

5.2 Challenges

1. Choice of the suitable simulator to perform efficiently the proposed smart city. Four simulators were tried: CupCarbon , Ifog Sim, Omnet-5, and Packet tracer. its found that the most suitable simulator for this work is packet tracer as mentioned in chapter three .

2. Exporting the real time data from packet tracer to local host was an issue because there are no references tackeld this process. So many methods were tried to achieve this step and it is found that the best solution to this issue is using TCP-Socket API.
3. The large number of messages produced by the smart city prototype results in overlapping between messages at the Tcp server, because TCP is a stream of bytes protocol . So, an attempts to fix this problem was made by implementing some forms of message framing, but it did not work. Finally, we turn to use UDP Socket because dose not cuse such problem.
4. The Google Cloud Platform (GCP) as a service provider was tried and after establishing an account in the cloud console and activated it by Turkish address because GCP service is not supported in Iraq, a problem was faced when loading a large number of telemetry messages, and to overcome this a MAC Client must be used. So the cloud services changed to the Azure Microsoft which supports windows and achieve the goal.

5.3 Future work

1. Deep analysis, Data Mining ,and Machine Learning can be applied to the smart city data platforms uploaded to cloud to provide statistical on the interacting performance among city platforms. This statistical supporting and enhancing decisions were made in the city control center.
2. As a result of this work, any platform of the smart city can be applied practically in Mosul city as a first step toward establishing smart city to enhance the quality of citizen life.

References

[1] Cisco, “Internet of Everything,” *cisco internet of everything course*. pp. 1–11, 2017, doi: 10.1007/978-3-319-55405-1_1.

[2] F. Garzia and L. Papi, “An Internet of Everything based integrated security system for smart archaeological areas,” *Proceedings - International Carnahan Conference on Security Technology*, vol. 0, 2016, doi: 10.1109/CCST.2016.7815684.

[3] Z. Nezami and K. Zamanifar, “Internet of Things/Internet of Everything: Structure and Ingredients,” *IEEE Potentials*, vol. 38, no. 2, pp. 12–17, 2019, doi: 10.1109/MPOT.2018.2855439.

[4] S. T. Gao, Xiao-zhi and M. C. T. K. K. Mishra, *Advances in Computational Intelligence and Communication Technology*. Springer Nature Singapore Pte Ltd. 2021, 2019.

[5] P. G. V. Naranjo, Z. Pooranian, M. Shojafar, M. Conti, and R. Buyya, “FOCAN: A Fog-supported smart city network architecture for management of applications in the Internet of Everything environments,” *Journal of Parallel and Distributed Computing*, vol. 132. pp. 274–283, 2019, doi: 10.1016/j.jpdc.2018.07.003.

[6] T. N. Gia, M. Jiang, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, “Fog computing in healthcare Internet of Things: A case study on ECG feature extraction,” *Proceedings - 15th IEEE International Conference on Computer and Information Technology, CIT 2015, 14th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2015, 13th IEEE International Conference on Dependable, Autonomic and Se*, pp.

356–363, 2015, doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.51.

[7] M. I. Naas, P. R. Parvedy, J. Boukhobza, and L. Lemarchand, "IFogStor: An IoT Data Placement Strategy for Fog Infrastructure," *Proceedings - 2017 IEEE 1st International Conference on Fog and Edge Computing, ICFEC 2017*, pp. 97–104, 2017, doi: 10.1109/ICFEC.2017.15.

[8] B. Pourghebleh and N. J. Navimipour, "Data aggregation mechanisms in the Internet of things: A systematic review of the literature and recommendations for future research," *Journal of Network and Computer Applications*, vol. 97, no. November, pp. 23–34, 2017, doi: 10.1016/j.jnca.2017.08.006.

[9] S. Abdelwahab, B. Hamdaoui, M. Guizani, and A. Rayes, "Enabling smart cloud services through remote sensing: An internet of everything enabler," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 276–288, 2014, doi: 10.1109/JIOT.2014.2325071.

[10] R. E. Balfour, "Building the 'Internet of Everything' (IoE) for first responders," *2015 IEEE Long Island Systems, Applications and Technology Conference, LISAT 2015*, 2015, doi: 10.1109/LISAT.2015.7160172.

[11] P. A. Pena, D. Sarkar, and P. Maheshwari, "A big-data centric framework for smart systems in the world of internet of everything," *Proceedings - 2015 International Conference on Computational Science and Computational Intelligence, CSCI 2015*, pp. 306–311, 2016, doi: 10.1109/CSCI.2015.62.

[12] B. Ahlgren, M. Hidell, and E. C. H. Ngai, "Internet of Things for Smart Cities: Interoperability and Open Data," *IEEE Internet*

Computing, vol. 20, no. 6, pp. 52–56, 2016, doi: 10.1109/MIC.2016.124.

[13] S. J. Clement, D. W. McKee, and J. Xu, “Service-Oriented Reference Architecture for Smart Cities,” *Proceedings - 11th IEEE International Symposium on Service-Oriented System Engineering, SOSE 2017*, pp. 81–85, 2017, doi: 10.1109/SOSE.2017.29.

[14] C. Badii *et al.*, “Snap4City: A scalable IOT/IOE platform for developing smart city applications,” *Proceedings - 2018 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovations, SmartWorld/UIC/ATC/ScalCom/CBDCo*, no. 688196, pp. 2109–2116, 2018, doi: 10.1109/SmartWorld.2018.00353.

[15] M. Yang, T. Zhu, B. Liu, Y. Xiang, and W. Zhou, “Machine Learning Differential Privacy With Multifunctional Aggregation in a Fog Computing Architecture,” *IEEE Access*, vol. 6, pp. 17119–17129, 2018, doi: 10.1109/ACCESS.2018.2817523.

[16] S. Muralidharan, “Monitoring and Managing IoT Applications in Smart Cities Using Kubernetes,” in *CLOUD COMPUTING 2019 : The Tenth International Conference on Cloud Computing, GRIDs, and Virtualization Monitoring*, no. Venice, Italy, pp. 1–6.

[17] R. Mahmud and R. Buyya, “Modeling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit,” *Fog and Edge Computing*, pp. 433–465, 2019, doi: 10.1002/9781119525080.ch17.

[18] M. Masoud, Y. Jaradat, A. Manasrah, and I. Jannoud, “Sensors of

smart devices in the internet of everything (IOE) era: Big opportunities and massive doubts," *Journal of Sensors*, vol. 2019, 2019, doi: 10.1155/2019/6514520.

[19] S. L. Ullo and G. R. Sinha, "Advances in smart environment monitoring systems using iot and sensors," *Sensors (Switzerland)*, vol. 20, no. 11, 2020, doi: 10.3390/s20113113.

[20] A. Raj and S. Prakash, "Internet of Everything: A survey based on Architecture, Issues and Challenges," *2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering, UPCON 2018*, pp. 1–6, 2018, doi: 10.1109/UPCON.2018.8596923.

[21] Z. Khan, A. Anjum, and S. L. Kiani, "Cloud based big data analytics for smart future cities," *Proceedings - 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, UCC 2013*, pp. 381–386, 2013, doi: 10.1109/UCC.2013.77.

[22] M. H. Miraz, M. Ali, P. S. Excell, and R. Picking, "A review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT)," *2015 Internet Technologies and Applications, ITA 2015 - Proceedings of the 6th International Conference*, pp. 219–224, 2015, doi: 10.1109/ITechA.2015.7317398.

[23] P. Wlodarczak, M. Ally, and J. Soar, "Data mining in IoT data analysis for a new paradigm on the internet," *Proceedings - 2017 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2017*, no. August, pp. 1100–1103, 2017, doi: 10.1145/3106426.3115866.

[24] W. Yu *et al.*, "A Survey on the Edge Computing for the Internet of

Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2017, doi: 10.1109/ACCESS.2017.2778504.

[25] Y. Mehmood, F. Ahmad, I. Yaqoob, A. Adnane, M. Imran, and S. Guizani, "Internet-of-Things-Based Smart Cities: Recent Advances and Challenges," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 16–24, 2017, doi: 10.1109/MCOM.2017.1600514.

[26] T. Kurian and G. C. CEO, "HTTP vs. MQTT: A tale of two IoT protocols," *INSIDE GOOGLE CLOUD*. https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols.

[27] R. K. Naha *et al.*, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47980–48009, 2018, doi: 10.1109/ACCESS.2018.2866491.

[28] I. Burago, D. Callegaro, M. Levorato, and S. Singh, "Intelligent data filtering in constrained IoT systems," *Conference Record of 51st Asilomar Conference on Signals, Systems and Computers, ACSSC 2017*, vol. 2017-Octob, pp. 928–935, 2017, doi: 10.1109/ACSSC.2017.8335485.

[29] S. Abbasian Dehkordi, K. Farajzadeh, J. Rezazadeh, R. Farahbakhsh, K. Sandrasegaran, and M. Abbasian Dehkordi, "A survey on data aggregation techniques in IoT sensor networks," *Wireless Networks*, vol. 26, no. 2, pp. 1243–1263, 2020, doi: 10.1007/s11276-019-02142-z.

[30] J. Gibson, R. Rondeau, D. Eveleigh, and Q. Tan, "Benefits and challenges of three cloud computing service models," *Proceedings of the 2012 4th International Conference on Computational Aspects of*

Social Networks, CASoN 2012, pp. 198–205, 2012, doi: 10.1109/CASoN.2012.6412402.

[31] A. Rashid and A. Chaturvedi, "Cloud Computing Characteristics and Services A Brief Review," *International Journal of Computer Sciences and Engineering*, vol. 7, no. 2, pp. 421–426, 2019, doi: 10.26438/ijcse/v7i2.421426.

[32] P. S, "Introduction - XML, JSON. In: Pro RESTful APIs. Apress, Berkeley," in *In: Pro RESTful APIs. Apress, Berkeley*, Apress, Berkeley, CA, 2017, p. 136.

[33] M. Clarke, "A reference architecture for telemonitoring," *Studies in Health Technology and Informatics*, vol. 103, pp. 381–384, 2004, doi: 10.3233/978-1-60750-946-2-381.

[34] C. Estela, "Blood Glucose Levels," *Blood Glucose Levels*, vol. 3, no. 2, 2019, doi: 10.5772/intechopen.73823.

[35] CHRISTINA VANVUREN, "What's an Ideal Level for Your Home?," *CONTRIBUTOR*, 2018. https://molekule.science/what-is-relative-humidity/.

[36] et al. Penney D, Benignus V, Kephalopoulos S, "Carbon monoxide. In: WHO Guidelines for Indoor Air Quality: Selected Pollutants.," *World Health Organization*, 2010. https://www.ncbi.nlm.nih.gov/books/NBK138710/.

[37] H. R. Dhasian and P. Balasubramanian, "Survey of data aggregation techniques using soft computing in wireless sensor networks," *IET Information Security*, vol. 7, no. 4, pp. 336–342, 2013, doi: 10.1049/iet-ifs.2012.0292.

Appendix (A)

Centralized Server Code

```
# Cloud Link API Copyright (c) Microsoft. All rights reserved.


import time
import socket
import sys


from azure.iot.device import IoTHubDeviceClient, Message # Using the Azure
CLI

# The device connection string to authenticate the device with your IoT hub.

# az iot hub device-identity
CONNECTION_STRING = "HostName=ioeData.azure-
devices.net;DeviceId=MyPythonDevice;SharedAccessKey=vnuYNICd7hIvmVVx5y+wgDhIT
QlpJswQCPR2p1iDxvo=" # az iot hub device-identity

# Define the JSON message to send to IoT Hub.
MSG_TXT = '{{"getway": {getway},"port": {port},"systemType":
{systemType},"localIP": {localIP},"systemCode":
{systemCode},"deviceName":{deviceName},"deviceSerial":{deviceSerial},"read":{
read}}}'


# tcp socket inti__
HOST=''
PORT=1235
s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
r=0
list=[]
data=[]
print('socket greated')
s.bind((HOST,PORT))
print("socket bind compleate")


def ReceveData():


    global  list
    global data
    global r
    data,addr = s.recvfrom(1024)

    temp = str(data)
    beforefilter=open("beforefilter.txt","a+")
    beforefilter.write(temp + "\n")
    beforefilter.close
    data=temp.split(',')
    if r<20:
            list.append(data)
            print("msg no is : "+ str(r) + "message is:" + str (data) +
str(sys.getsizeof(data)))
```

```
            r+=1
            return data


    if r>=20:
       state=False
       for i in range(len(list)):
            if list[i][4] == data[4] and list[i][6] == data[6] and list[i][7]
== data[7]:
                 print("find element: " + str(data))
                 state=True
                 return None
                 break
       if state is False:
          for i in range(len(list)):
             if list[i][4] == data[4] and list[i][6]==data[6]:
                list.remove(list[i])
                list.append(data)
                print("removed item position = " + str(i) + "and replaced
with: " + str (data))
                state=False
                return data
                break

def iothub_client_init():
    # Create an IoT Hub client
    client =
IoTHubDeviceClient.create_from_connection_string(CONNECTION_STRING)
    return client
def iothub_client_telemetry_sample_run():

    try:
        client = iothub_client_init()
        print ( "IoT Hub device sending periodic messages, press Ctrl-C to
exit" )
        while True:
                 mylist=[]
                 mylist=ReceveData()
                 print("inside first while true")
                 if  mylist != None:
                          print ("filterd Message is: "+ str(mylist))
                          afterfilter=open("afterfilter.txt","a+")
                          afterfilter.write(str(mylist ) + "\n" )
                          afterfilter.close
                          getway=mylist[0]
                          port=mylist[1]
                          systemType=mylist[2]
                          localIP=mylist[3]
                          systemCode=mylist[4]
                          deviceName=mylist[5]
                          deviceSerial=mylist[6]
                          read=mylist[7]

                          msg_txt_formatted =
MSG_TXT.format(getway=getway,port=port,systemType=systemType,localIP=localIP,
systemCode=systemCode,deviceName=deviceName,deviceSerial=deviceSerial,read=re
ad)
                          message = Message(msg_txt_formatted)
                          # Send the message.
                          print( "Sending message: {}".format(message) )
                          client.send_message(message)
                          print ( "Message successfully sent" )


    except KeyboardInterrupt:
        print ( "IoTHubClient sample stopped" )

if __name__ == '__main__':
    print ( "IoT Hub Quickstart #1 - Simulated device" )
    print ( "Press Ctrl-C to exit" )
    iothub_client_telemetry_sample_run()
```

Appendix (B)

Cluster Health care system server (CHCS) Code

```
from time import*
from physical import*
from udp import*
from realtcp import*
from email import*



def onUDPReceive(ip, port, data):
    temp =str( data)
    list=temp.split(',')
    device=list[0]
    read =float( list[2])
    patiant=list[1]
    d1 = "heart rate recorder"
    d2 = "blood sugar recorder"
    d3 = "SPO2"
    d4 = "body temerature"
    print(read)

    if device == d1:
        if read > 90:
            print ("alarm high beart rate")
            EmailClient.send("doctor@health.com", "heart alarm for : " +
patiant, "heart rate is : " +str( read) )

        elif read < 60:
            print ("alarm low heart rate")
            EmailClient.send("doctor@health.com", "heart alarm for : " +
patiant, "heart rate is : " +str( read) )
        else:
            print("normal heart rate")


    elif device == d2:
        if read > 180:
           print ("alarm high blood sugar")
           EmailClient.send("doctor@health.com", "blood sugar alarm for : " +
patiant, "blood sugar is : " +str( read) )

        elif read < 60:
            print ("alarm low blood suger")
            EmailClient.send("doctor@health.com", "blood sugar alarm for" +
patiant, "blood sugar is : " +str( read) )
        else:
            print("normal blood suger")



    elif device == d3:
        if read > 100:
            print ("alarm high SPO2")
            EmailClient.send("doctor@health.com", "SPO2 alarm for" + patiant,
"SPO2 is : " +str( read) )

        elif read < 74:
            print ("alarm low SPO2")
            EmailClient.send("doctor@health.com", "SPO2 alarm for" + patiant,
"SPO2 is : " +str( read) )
        else:
            print("normal SPO2")



    elif device == d4:
        if read > 37.5:
            print ("alarm high body temerature")
            EmailClient.send("doctor@health.com", "body temerature alarm for"
+ patiant, "body temerature is : " +str( read) )

        elif read < 36.5:
            print ("alarm low body temerature")
            EmailClient.send("doctor@health.com", "body temerature alarm for"
+ patiant, "body temerature is : " +str( read) )
        else:
            print("normal body temerature")

    else:
            print("not connected")



def main:()
        socket = UDPSocket()
        EmailClient.setup)
                "server@health.com,"
                ,"10.0.0.250"
                "server,"
                "123"
        (
        delay(7000)
        socket.onReceive(onUDPReceive)
        print(socket.begin(1234))



        count = 0


if __name__ == "__main:"__
        while True:
                main()
                delay(7000)
```

Appendix (C)

Heart rate meter smart device code

The following code is written with python 3.8.

```
from physical import*
from gpio import*
from environment import Environment
from ioeclient import IoEClient
from pyjs import*
from udp import*
from networking import*

ALARM_LEVEL_MIN = 60
ALARM_LEVEL_MAX = 90
socket=None

state = 0
level = 0
 #Top left position and the clip area size for the text.
textPos = JsObject({"x": 16, "y": 20})
textAreaSize = JsObject({"w": 230, "h": 52 })

def setup:()
    global state,socket
    IoEClient.setup})
"        type": "Heart Rate,"
"        states}] :"
"            name": "Alarm,"
"            type": "bool,"
"            controllable": False
,{
}
"            name": "Level,"
"            type": "number,"
"            controllable": False
[{
({


    state = restoreProperty("state", 0)
    setState(state)
    sendReport()

    socket=UDPSocket()
    socket.onReceive(onUDPReceive)
    print(socket.begin(1234))







def onUDPReceive(ip, port, data):
       print("received from"
                +ip + ":" + str(port) + ":" + data;(

def restoreProperty (propertyName, defaultValue):
    value = getDeviceProperty(getName(), propertyName)
    if  value != "" and value != None:
        if isinstance(defaultValue, (int, float)):
            value = int(value)

        setDeviceProperty(getName(), propertyName, value)
        return value

    return defaultValue

def loop:()
    detect()
    socket.send("10.0.0.249", 1234,"2000," + str(localIP()) +","+
str(customRead(0)) + ","+ "Heart Rate," + str(getSerialNumber()) +"," +
str(255 * analogRead(A0) / 1023))
    socket.send("10.0.0.250", 1234, "heart rate recorder," +
str(customRead(0)) + "," + str(255 * analogRead(A0) / 1023))
    setCustomText(textPos.x, textPos.y, textAreaSize.w, textAreaSize.h,
str(255 * analogRead(A0) / 1023))
    setDeviceProperty(getName(), "text",str(255 * analogRead(A0) / 1023))
    delay    (6000)



def detect:()
    value = 255 * analogRead(A0) / 1023
    if value >= 0:
        setLevel( 255 * analogRead(A0) / 1023)

def sendReport:()
    report = str(state) +","+str(level);    # comma seperated states
    IoEClient.reportStates(report)
    setDeviceProperty(getName(), "state", state)
    setDeviceProperty(getName(), "level", level)

def setState (newState):
    global state
    if  newState == 0:
        digitalWrite(1, LOW)
    else:
        digitalWrite(1, HIGH)

    state = newState

    sendReport()

def setLevel (newLevel):
    global level
    if level == newLevel:
        return

    level = newLevel
    if  level > ALARM_LEVEL_MAX:
        setState(1)
    elif level < ALARM_LEVEL_MIN:
       setState(1)
    else:
        setState(0)

    sendReport()

if __name__ == "__main:"__
    setup()
    while True:
        loop()
        sleep(0)
```

"الخلاصة"

من خلال تقنية إنترنت كل شيء (IoE) يتم توفير نهج مختص ومنظم لتحسين صحة ورفاهية البشرية. العديد من الطرق العملية لتوفير حياة ذات جودة أفضل للناس تعتمد على IoE، حيث يساهم تطوير تطبيقات مثل مراقبة صحة الإنسان، والإدارة الذكية، واتخاذ القرار في مجال الصناعة، والمباني الذكية، وإدارة حركة المرور على الطرق، والعديد من التطبيقات الأخرى في بلورة مفهوم المدينة الذكية. يدمج هذا المفهوم جميع التطبيقات الذكية ويدير بياناتها بذكاء لاتخاذ قرارات أفضل وأسرع من أجل جعل حلم الناس حقيقيًا عن الرفاهية والحياة المريحة. إن بدء مثل هذه المدينة الذكية هو هاجس العالم في الوقت الحاضر.

في هذا العمل، تم استقصاء البنية التحتية لإنترنت كل شيء فيما يتعلق بتقنيات الاتصال والأجهزة والبروتوكولات حيث ان معايير معمارية IoE لاتزال قيد التطوير ولكن معظم الباحثين يتبنون بنية ثلاثية الطبقات. لتنفيذ البنية التحتية لأنترنيت كل شيء، تم تصميم نموذج اولي لتمثيل واقعي للمدينة الذكية متضمناً ثلاث منصات هي الرعاية الصحية والبيئة والأمن.

لإنشاء مثل هذا النموذج، قمنا بتطوير برامج جميع الأجهزة والخوادم بالإضافة إلى أننا قمنا ببناء خوارزميات لتوفير الاستجابات الذكية لأحداث الأنظمة. من اجل توسيع الوظائف الحالية للخوادم وتحقيق التفاعلات بين الخوادم والسحابة، تم استخدام واجهة برمجة التطبيقات (API) . من اجل برمجة التطبيقات الخاصة بنمط المدينة الذكية التي صممت من قبلنا تم اعتماد لغة برمجة Python 3.8.

لتعزيز أداء نظامنا، تم العمل بمفهوم الحوسبة المتطورة من خلال توسيع خدمة الحوسبة السحابية إلى حافة الشبكة بحيث تمت إضافة نقاط الضباب (FN). يتم نقل البيانات من أجهزة إنترنت الأشياء إلى عقدة حوسبة الضباب البعيدة خلال الزمن الحقيقي من أجل المراقبة الفورية للنظام والمعالجة السريعة والتحليل الدقيق الذي يسهم في اتخاذ قرارات في الزمن الحقيقي للمنصات الثلاث في المدينة الذكية. أسهمت هذه الخطوة في التقليل من زمن استجابة النظام وجعلت الأنظمة موثوقة حتى إذا كان الاتصال بالسحابة يعاني من بعض التوقفات.

البيانات الناتجة من المدينة الذكية غير منظمة ولديها العديد من المشكلات مثل التكرار والازدواجية، لذلك تم استخدام تقنية تجميع البيانات (data aggregation). لتحسين أداء النموذج

المقترح للمدينة الذكية. يتم تحميل البيانات المجمعة الناتجة عن المدينة الذكية إلى التخزين السحابي Microsoft Azure cloud لإعداد هذه البيانات لتكون جاهزة للتحليل واستخراج المعلومات.

تم استخدام Cisco packet tracer 7.2 كأداة فعالة لتصميم وتنفيذ النموذج الأولي.

البحوث المنشورة

[1] Rawia Talal and Abdulbarry Raouf Suleiman, “IoT/IoE Smart Health Care System using Fog Computing Features", 2nd Annual International Conference on Information and Sciences AICIS 2020, Iraq, Fallujah for the period from (24-25) November, 2020. The conference will publish the papers in IEEE..

المقترح للمدينة الذكية.

إقرار المشرف

اشهد بان الرسالة الموسومة ب " **التحقيق في شبكات الاتصالات والبنية التحتية لإنترنت كل شيء** " تمت تحت اشرافي وهي جزء من متطلبات نيل شهادة الماجستير في هندسة الحاسوب والمعلوماتية.

التوقيع:
المشرف : د. عبد الباري رؤوف سليمان
التاريخ: / / 2020

إقرار المقيم اللغوي

اشهد باني قمت بمراجعة الرسالة الموسومة ب " **استقصاء شبكات الاتصالات والبنية التحتية لأنترنيت كل شيء** " من الناحية اللغوية وتصحيح ما ورد فيها من أخطاء لغوية وتعبيرية وبذلك أصبحت الرسالة مؤهلة للمناقشة بقدر تعلق الامر بسلامة الأسلوب وصحة التعبير.

التوقيع:
المقوم اللغوي:
التاريخ: / / 2020

إقرار رئيس لجنة الدراسات العليا

بناء على التوصيات المقدمة من قبل المشرف والمقوم اللغوي أرشح هذه الرسالة للمناقشة.

التوقيع:
الاسم: أ . م . معن أحمد شحاذة العدواني
التاريخ: / / 2020

إقرار رئيس القسم

بناء على التوصيات المقدمة من قبل المشرف والمقوم اللغوي ورئيس لجنة الدراسات العليا أرشح هذه الرسالة للمناقشة.

التوقيع:

الاسم: أ . م . معن أحمد شحاذة العدواني

التاريخ: / / 2020

استقصاء شبكات الاتصالات والبنية التحتية لأنترنيت كل شيء

رسالة تقدم بها

راوية طلال عزيز

إلى
مجلس كلية هندسة الالكترونيات
جامعة نينوى
كجزء من متطلبات نيل شهادة الماجستير
في
هندسة الحاسوب والمعلوماتية

بإشراف
د. عبد الباري رؤوف سليمان

أستاذ مساعد

1442 هـ   2020م

جامعة نينوى

كلية هندسة الالكترونيات

قسم هندسة الحاسوب والمعلوماتية

استقصاء شبكات الاتصالات والبنية التحتية لأنترنيت كل شيء

رسالة تقدم بها

راوية طلال عزيز

رسالة ماجستير علوم في هندسة الحاسوب والمعلوماتية

بإشراف

د. عبد الباري رؤوف سليمان

أستاذ مساعد

1442 هـ		2021 م