FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

DEGREE PROGRAMME IN WIRELESS COMMUNICATIONS ENGINEERING

MASTER'S THESIS

Lightweight Edge-Based Networking Architecture for Low-Power IoT Devices

Author	Archana Rajakaruna
Supervisor	Mika Ylianttila
Second Examiner	Madhusanka Liyanage
Technical Advisor	Pawani Porambage

May 2019

Rajakaruna A. (2019) Lightweight Edge-Based Networking Architecture for Low-Power IoT Devices. University of Oulu, Faculty of Information Technology and Electrical Engineering, Degree Programme in Wireless Communications Engineering. Master's Thesis, 64 p.

ABSTRACT

The involvement of low power Internet of Things (IoT) devices in the Wireless Sensor Networks (WSN) allow enhanced autonomous monitoring capability in many application areas. Recently, the principles of edge computing paradigm have been used to cater on-site processing and managing actions in WSNs. However, WSNs deployed in remote sites require human involvement in data collection process since internet accessibility is still limited to population dense areas. Nowadays, researchers propose UAVs for monitoring applications where human involvement is required frequently. In this thesis work, we introduce an edge-based architecture which create end-to-end secure communication between IoT sensors in a remote WSN and central cloud via UAV, which assist the data collection, processing and managing procedures of the remote WSN. Since power is a limited resource, we propose Bluetooth Low Energy (BLE) as the communication media between UAV and sensors in the WSN, where BLE is considered as an ultra-low power radio access technology. To examine the performance of the system model, we have presented a simulation analysis considering three sensor nodes array types that can realize in the practical environment. The impact of BLE data rate, impact of speed of the UAV, impact of distance between adjacent sensors and impact of data generation rate of the sensor node have been analysed to examine the performance of system. Moreover, to observe the practical functionality of the proposed architecture, prototype implementation is presented using commercially available off-the-shelf devices. The prototype of the system is implemented assuming ideal environment.

Key words: Internet of Things (IoT), edge-based architecture, Wireless Sensor Networks (WSN), Unmanned Aerial Vehicles (UAVs), Bluetooth Low Energy (BLE)

TABLE OF CONTENTS

FOREWORD

The Master's thesis titled "Lightweight Edge-Based Networking Architecture for Low-Power IoT Devices" is composed at the Centre for Wireless Communications (CWC), University of Oulu as per the requirement of the Master's Degree Programme in Wireless Communications Engineering. The main intension of this thesis work is to present a remote monitoring architecture that creates secure end-to-end communication between diversified WSN and central cloud via UAV. Therefore, we introduce an edge-computing based remote site monitoring architecture, which consist of an UAV to coordinate data retrieving, processing and management in assorted WSN. Performance of the proposed system model is analysed by implementing a simulation in MATLAB. Moreover, a prototype implementation is performed using commercially available devices to observe the practical feasibility of the proposed remote monitoring architecture.

This research work is assisted by The Business Finland in Towards Digital Paradise, The Academy of Finland in SECURE Connect, 6Genesis Flagship (grant no. 318927), 5GEAR project, and EU in RESPONSE 5G (Grant No: 789658) project.

Here I sincerely convey my appreciation to my supervisor Prof. Mika Ylianttila for his enormous support and advise during the thesis work. Also, I am thankful to Dr. Madhusanka Liyanage, for providing the financial support as the project manager, for the guidance and providing me the feedbacks of the research work as second examiner of this thesis work. I extend my thank to Dr. Pawani Porambage, the technical supervisor of my thesis work, for giving me the continuous feedbacks about the research work and directing to achieve the thesis objectives. I would like to thank Ahsan Manzoor, the doctoral student of the CWC for the support and ideas given to implement the prototype of the proposed system model of this thesis. I convey my appreciation to all my teammates in NS research team for providing corporative and friendly atmosphere. I am grateful to CWC for allowing me the opportunity to contribute this research work.

Here I would like to thank my dearest husband Yushan, for his enormous support, guidance and encouragements which helped me to fulfil my goals. Last but not least, I want to thank my parents and two sisters for encouraging me to succeed my life objectives.

Oulu, 31 May 2019

Archana Rajakaruna

LIST OF ABBREVIATIONS AND SYMBOLS

ADC	Analog to Digital Converter
ATT	Attribute Protocol
BLE	Bluetooth Low Energy
CPU	Central Processing Unit
CSI	Camera Serial Interface
DSI	Display Serial Interface
E2E	End-to-End
EEPROM	Electrically Erasable Programmable Read-Only Memory
ETSI	European Telecommunications Standards Institute
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GPIO	General Purpose Input Output
GPRS	General Packet Radio Service
GPS	Global Positioning System
GPU	Graphic Processing Unit
GW	Gateway
HAP	high altitude platforms
HAVC	Heating, Ventilation, and Air Conditioning
HCI	Host Controller Interface
HDMI	High Definition Multimedia Input
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrated development environment
IoT	Internet of Things
ISM	Industrial Scientific Medical
LAN	Local Area Network
LAP	low altitude platform
LLCAP	Logical Link Control and Adaptation Protocol
MAC	Media Access Control
ME	Mobile Edge
MEC	Multi-access Edge Computing
MEO	Mobile Edge Orchestrator
MEPM	Mobile Edge Platform Manager
NFC	Near Field Communication
NS	Networks and Systems
PC	Personal Computer
QoS	Quality of Service
RAM	Random Access Memory
RFID	Radio Frequency Identifier
RSSI	Received Signal Strength Indication
RTC	Real Time Clock
RTLS	Real-Time Location System
SM	Security Manager
SRAM	Static Random-Access Memory
SUN	Smart Utility Networks
TVWS	TV White Spaces
UAV	Unmanned Aerial Vehicles

UE	User Equipment
ULP	Ultra Low Power
USB	Universal Serial Bus
UWB	Ultrawideband
WAN	Wide Area Network
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network

T_W	waiting time of the drone near a one sensor node
T_{DL}	data download time from drone to sensor node
T_P	data processing time inside the drone
T_F	total flying time of the drone
T_C	battery charging time of the drone
R_{BLE}	data transfer rate between drone and sensor node
R_S	data generation rate of sensor node
S	speed of the drone
n	number of sensor nodes in the array
n_1	number of sensor nodes in linear sensor nodes array
n_2	number of sensor nodes in circular sensor nodes array
n_3	number of sensor nodes in square sensor nodes array
D	distance between two sensor nodes

B/min	bytes per minutes
m/s	meters per second

1 INTRODUCTION

The Internet of Things (IoT) said otherwise the Internet of Everything is a modern advance technology paradigm which is capable of consolidating scattered global network of devices and machines with each other [1]. IoT platform requires different protocols to possess the internet connectivity and inter-device communication. Then it allows different services to the user; such as real time data accessibility to the devices, remote device management and processing. Also, it introduces different web applications which are globally accessible, scalable and cater interfaces to external applications. [2]

Recently, Unmanned Aerial Vehicles (UAVs) are used in diverse autonomous monitoring applications in order to relay the user services. Hence, the monitoring systems may eliminate or limit the human labour and reduce the infrastructure cost [3].

1.1 Background and Motivation

With the advancement of modern technology, the IoT paradigm has influenced in many ways to enhance the productivity of digital device networks. IoT empowers communication between digital devices via internet and provides secure platform to preserve the data for further use. Nowadays, low power IoT devices are the most popular devices in these smart device networks since power is becoming a scarce resource. Therefore, most of the wireless sensor networks are implemented using low power IoT devices. As a fundamental example of such sensor networks incorporates HAVC (Heating, Ventilation, and Air Conditioning), thermostats monitoring and controlling systems that entitle smart homes [4]. Moreover, IoT has become the major concept behind the assorted industrial domains and environmental aspects to enhance the quality of our life. IoT applications include healthcare, industrial automation, transportation, agriculture, water resource management and emergency response to manmade and natural disasters where direct human interaction is difficult [4].

In IoT platform, multiple devices retrieve data according to their duties and transmit the data to the cloud servers, in order to build more comprehensive picture of the whole ecosystem [4]. Nevertheless, these networks need seamless connectivity between central cloud and IoT devices, which is difficult to cater every time [5]. Therefore, edge-computing paradigm has been introduced to assist the communication gap between remote IoT devices and central cloud servers [6], [7]. Edge computing paradigm caters the computation tasks at the edge of the network, on upload data stream on behalf of IoT services and on download data stream on behalf of cloud services [6].

Nowadays most of the low power IoT devices use short range radio access techniques such as Bluetooth Low Energy (BLE), Near-Field Communication (NFC), Halow, Ultrawideband (UWB), Smart Utility Networks (SUNs) and ZigBee [8]. BLE is a novel wireless communication protocol that provides short range communication capability between digital devices with ultra-low power and low-cost features [9]. BLE allows an adaptable topology that can be adjusted to accommodate in many applications. The IoT sensor nodes that use standalone BLE to connect with external devices require devoted nodes that act as local gateways (GWs) to cater back-end connectivity with the central cloud. In our previous work, we presented a mobile-based relay assistance architecture which creates end-to-end (E2E) communication between low power IoT devices and central cloud servers with the absence of devoted local gateway [10], [11].

Recently, Unmanned Aerial Vehicle (UAV) have become a hot topic for the autonomous monitoring applications due to their high mobility and low cost [12]. Although, there is still no

sufficient work to show the collaboration between UAV systems and low power IoT devices and their cooperative applicability with practical implications in sensor networks.

1.2 Research Problem and Objectives

Since the collaboration of UAV systems and low power IoT devices and their behaviour in a wireless sensor network has not been addressed comprehensively, the deployment of such collaborative networks is challenging. Therefore, to address the problem of having lack of knowledge of the collaborative operation of UAV systems and IoT devices, this domain has been thoroughly investigated and novel mechanisms have to be implemented. This problem has been addressed in this thesis and the thesis proposes a novel system architecture to a remote sensor network, which is unable to get the human involvement frequently. Apart from that, having an internet connectivity to such remote sensor networks is also not guaranteed.

By introducing UAVs or drones to the monitoring system, we eliminate the human interaction for the data collection procedure. Therefore, a drone can be sent to the remote location to collect the data extracted by the low power IoT sensors. Furthermore, using these drones, it is possible to instruct the actuators located in the remote area to perform specific actions based and to perform software updates in sensors. By processing the retrieved data from the sensors, drone is capable of activating these actions. The user can programme the drone according to the requirements. Therefore, user is capable of controlling drone, programming drone and decision making of the monitoring system at any time.

Therefore, the objective of this research is to create a secure communication system using UAVs and low power IoT devices which is capable of monitoring a remote wireless sensor network, perform a set of actions based on the monitored data. Moreover, the system is capable of collecting the data of the remote Wireless Sensor Network (WSN) and storing the data in a cloud environment for future reference.

1.3 Selected Scope

In order to integrate end-to-end secure connectivity between UAVs and low power IoT sensor networks in practical scenarios, we consider a smart agricultural use case, which has heterogeneous wireless sensors deployed in a remote field. Introducing IoT associated with cloud computing to monitor such sites will yield more benefits. However, practically it is difficult to provide the internet connectivity to each remote field, therefore an alternate solution to solve the problem has to be considered. Introduction of unlicensed band short-range radio access technologies can merge the communication gap between cloud and remotely deployed sensors. The devices, which facilitate these technologies, are low power consuming and perform low data rates. Remote agricultural fields do not need frequent data collection, video streaming and other high data rate requirements. Therefore, to monitor a remote agricultural field equipped with a wireless sensor network, the use of short-range radio access technologies which utilize low power, provides a viable implementation.

We present an edge-computing based remote site monitoring architecture, which consist of an UAV to coordinate data retrieving, processing and management in assorted Wireless Sensor Network (WSN). According to the proposed system, it caters both remote and insight processing capabilities to the remote WSNs. The proposed system model can be used to monitor any remote location which unable to have internet access. Therefore, by introducing the architecture, we replace the mobile phone, which is considered as the communication relay in [10], using a drone to perform as an edge processor.

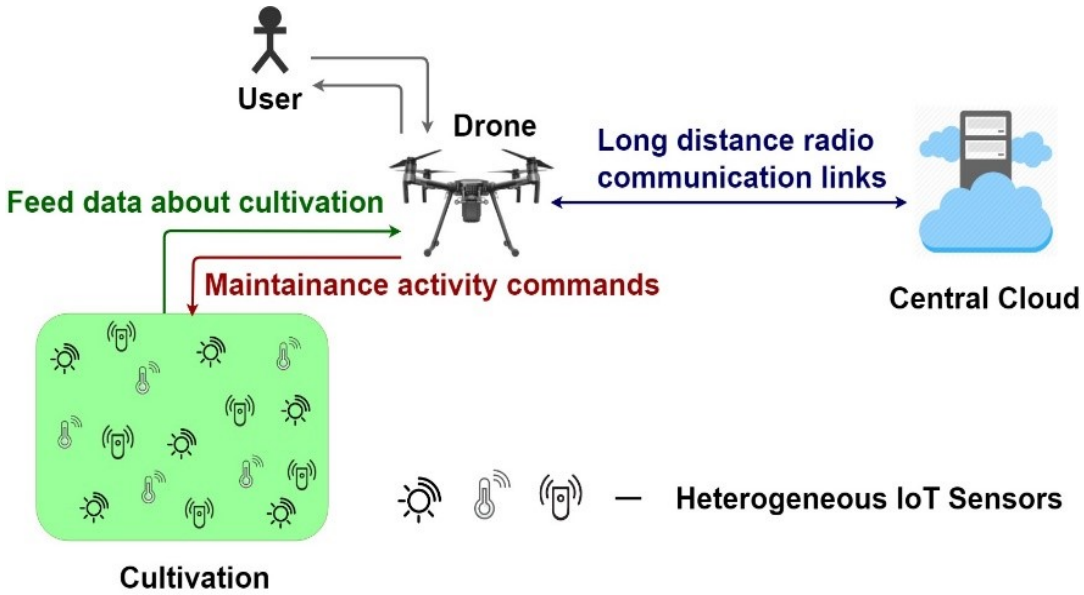

Figure 1. Potential use case: Edge based architecture for a smart agriculture application.

For the smart agriculture, remote monitoring architecture depicts in Figure 1, we implement a system model using an off the shelf drone and low power sensors. We implement the process for a remote agricultural site, which is inaccessible to internet. We used BLE as the communication protocol between drone and sensor nodes because it does not need the internet connectivity and it consumes low power. The development of commercial wireless sensors steers to wireless technologies combined with low power consumption and resource saving simple protocols. BLE is a latest wireless standard, which fulfil these basic requirements [13]. When drone returns to the central cloud server location or home location, it transfers the data to the cloud server over HTTPS [14] protocol.

To examine the viability of the proposed system model we carry out simulations for three main practical sensor nodes arrays such as linear, circular and squared assuming the sensors of the agricultural fields are distributed with equal mutual distances.

The same structure of the system model can be used for disaster detection, wastewater management, environmental monitoring, anomaly detection in sensor networks, mobile crowd sensing applications, etc. In future, this system can be extended for dew computing based applications.

1.4 Methodology

To realize the design of the remote smart agricultural monitoring system, we have implemented the prototype using commercially available BLE sensors and a drone. We used Waspmote sensors as the sensor nodes of the WSN, which is coupled with its BLE communication modules. The Phantom 3 SE drone coupled with a Raspberry Pi has been used for the data collection procedure since Phantom 3 SE drone is capable of carrying the intended weight.

The prototype implementation was carried out considering two physical locations (i.e. home location and remote field). In order to establish communication between drone and sensor nodes at the remote location we used BLE protocol while using the HTTPS protocol for the communication between drone and cloud server at the home location. BLE communication between drone and sensor nodes has been carried out under two main scenarios. First protocol is used to upload the data from sensor node to the drone and second protocol is used to upload

the data from the drone to the sensor node. The cloud server is implemented on the Google Firebase where Raspberry Pi first authenticates itself using its private key. After authentication, Raspberry Pi starts uploading the sensed data to cloud and stores in the database.

To analyse the feasibility of the implemented system model we have conducted a MATLAB simulation. For the simulation, we contemplated linear, circular and square orientations of sensor nodes arrays. The simulation results provide the graphical relationship between the number of sensor nodes that can be covered during the drone's data collection procedure within one flying cycle.

1.5 Contribution of the Thesis

The thesis work presents a remote monitoring architecture that creates secure end-to-end communication between diversified WSN and central cloud via UAV, which assist the data retrieving, data processing and management tasks in a remote site. To evaluate the performance of the proposed system model we have implemented a simulation considering three sensor nodes array types. Moreover, to observe the practical operability of the proposed architecture, we have implemented a prototype using commercially available equipment. Comprising the research work of this thesis, paper titled "Enabling End-to-End Secure Connectivity for Low-Power IoT Devices with UAVs" [15] has been published in the 2nd Workshop on Intelligent Computing and Caching at the Network Edge, IEEE WCNC 2019 conference.

1.6 Organization of the Thesis

The remainder of the thesis is organized as follows:

Chapter 2 describes the literature review of the research work. The literature review contains seven sub sections and its first subsection describes the BLE architecture and its GATT profile hierarchy since we used BLE as the communication media between drone and sensor node. Since our proposed architecture is benefited from different computing paradigms we have described few computing paradigms such as edge computing, ETSI MEC and dew computing in the next three subsections of the literature review. Then this chapter explains the contribution of WSNs and UAVs on the IoT platform. Finally, it provides viable use cases for this proposed system model.

Chapter 3 explains the proposed architecture in the thesis work. This chapter consists with two main subsections that describe the system model of the proposed architecture and communication protocols used in the data transferring process from WSN to central cloud. The system model describes three links that can be used in the remote monitoring application with proposed architecture. Then this chapter explains three communication protocols used in the data transfer process such as communication protocol used for the data upload process, communication protocol used for the data download process and communication protocol used in the data transfer process from drone to central cloud server at the home location.

Chapter 4 elaborates the experimental setup of the prototype implementation and simulation work carried out to examine the practical viability of the proposed system model. The first subsection of this chapter describes the simulation work of the thesis. Therefore, it provides information about variables, sensor nodes array types, mathematical expressions and parameters that have been used to get the simulation results. Then this chapter explains the prototype implementation methodology.

Chapter 5 analyses the simulation and experimental results of the thesis work. We considered four key parameters to examine the performance of the proposed architecture throughout the

simulation work. Therefore, we have addressed the impact of each parameter under the simulation results analysis subsection. Then we explain the experimental data rates that have extracted from the prototype implementation.

Chapter 6 demonstrates an overall discussion about the thesis work and it critically analyses the outcomes of the thesis work. Also, it provides a comparison between the thesis work and other similar domain research work. Then it addresses the successful objectives of the thesis. The last subsection of this chapter describes about the extended work that can be carried out in the future.

Chapter 7 summaries the thesis work. Therefore, it provides a summary about proposed architecture, system model, key objectives, prototype implementation, simulation results and feasible future research work.

2 LITERATURE REVIEW

2.1 Bluetooth Low Energy (BLE)

Bluetooth Low Energy (BLE) is the next generation of Bluetooth technology as well as the lowest feasible power wireless technology that has already been designed [16]. In addition, BLE has been introduced as a novel enhancement of Bluetooth 4.0 specification. As the name "Bluetooth Low Energy" implies, the aim of this technology is at ultra-low power as well as low-cost devices. Moreover, a similar technology that has been used in the past was known as Wibree and Ultra Low Power (ULP) [17].

The purpose of introducing the low power concept to the Classic Bluetooth is to enhance the performance of low bandwidth applications. Therefore, with BLE, devices consume very low power and because of that, those devices can operate for a longer time with smaller batteries or coin cells without replacing or recharging. Hence, this is best suited for the applications where long-life batteries are important and it is difficult to recharge frequently [17].

Depending on the functionality of BLE technology, devices can be categorized into two main types: [17].

1) Single mode BLE devices: These devices are compatible only with BLE (i.e. not support Classic Bluetooth). This is due to the fact that, mostly these devices are presumed to have very low power consumption and last for the longer period on coin cell batteries. Key fobs, thermometers, heart rate monitors, fitness and sports equipment and some sensors are few examples for single mode BLE devices [16].
2) Dual mode BLE devices: These devices are compatible with both Classic Bluetooth and BLE; also, they are capable of communicating with both Bluetooth only devices and single mode devices simultaneously. Moreover, these devices supposed to have bigger batteries or most of them are to recharge frequently although they do not have stringent needs on power consumption as single mode BLE devices [16]. Usually, smartphones, PCs and tablets can be known as dual mode devices.

Figure 2 illustrates the compatibility between Bluetooth devices, single mode BLE devices and dual mode BLE devices.

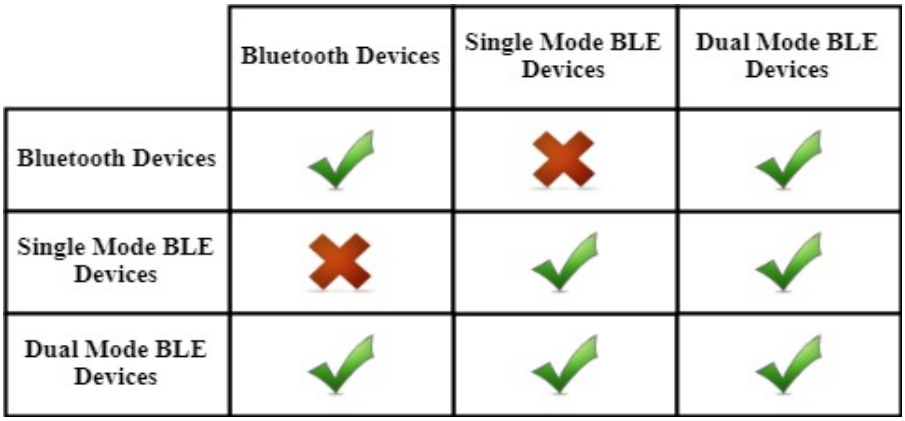

Figure 2. Compatibility between Bluetooth, single mode BLE and dual mode BLE devices [16].

Ultra-low power communication methods are obligatory facts for recent Internet of Things applications. Most of the ultra-low power IoT devices have been developed using short range

radio access techniques including Bluetooth Low Energy (BLE), Halow, ZigBee, Ultrawideband (UWB) and Smart Utility Networks (SUNs) [18]. Hence, BLE provides low power and low-cost communication platform for those IoT applications [19]. Moreover, BLE consumes intensely low energy and includes an attractive ratio of energy per bit transmitted, when compared with one of the most popular short-range radio access technologies ZigBee [19]. The connection time of BLE communication is very low, therefore, it provides seamless communication between user and devices without any impact on the user experience [16].

2.1.1 Bluetooth Low Energy (BLE) Architecture

The Bluetooth Low Energy architecture is fundamentally uncomplicated. BLE architecture consists with three basic components named applications, host and controller. Usually, the controller is a hardware device that transmits and receives radio signals and recognize the information packet structure of the signals. The host is a software component that caters services related to enabling connectivity between two devices. Also, the host provides multiple services to the connected devices using the same radio signals. The application uses the software stack, and therefore the controller enables a use case. The controller consists with Physical layer, direct test mode and the link layer of the BLE architecture. Moreover, it includes the host controller interface that enables communication between the controller and the host. The BLE architecture is depicted in Figure 3.

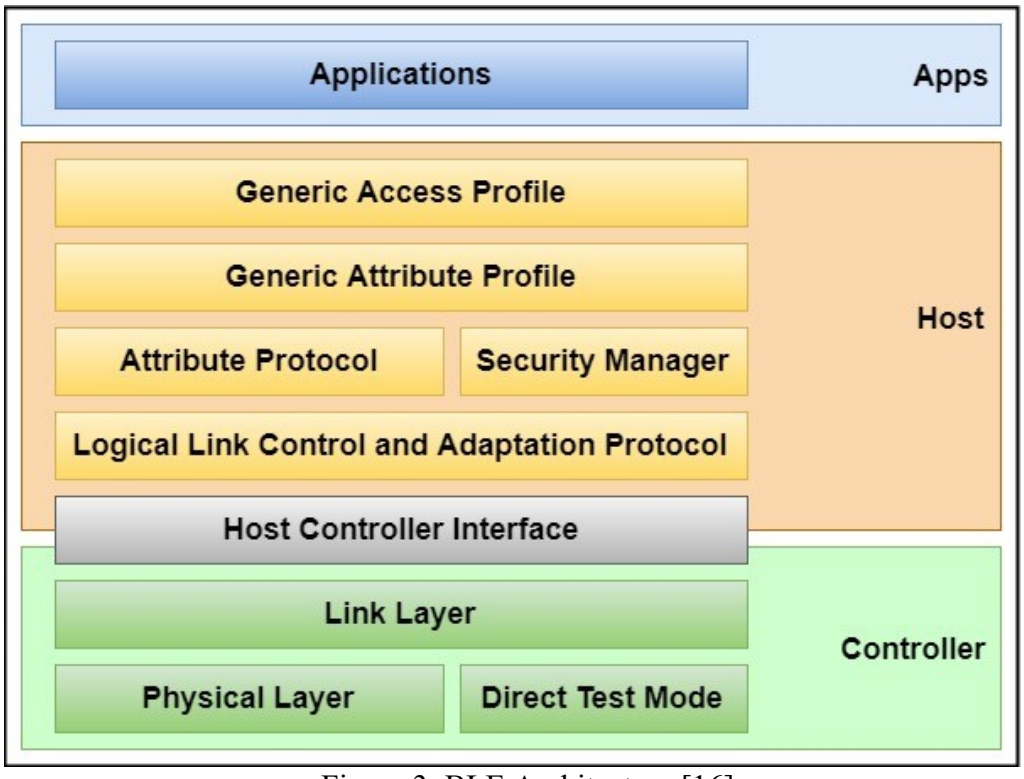

Figure 3. BLE Architecture [16].

Controller: This caters signal transmission and reception of the data packets using analog and digital parts of the radio frequency component. This is a hardware component of the architecture. The controller communicates with external devices over an antenna. The Host Controller Interface (HCI) provides the communication interface between the controller and the host [16]. The components of the controller are described below.

- **Physical Layer:** BLE operates in the 2.4 GHz Industrial Scientific Medical (ISM) band, which is considered as the license-free frequency band. This layer consists with 40 radio frequency channels with 2 MHz channel spacing [20].

- **Direct Test Mode:** This allows a tester to command a controller's physical layer to transmit or receive a sequence of test packets [16].

- **Link Layer:** This is the most complicated component of the BLE architecture. The link layer provides services relate to scanning, advertising, creating and maintaining of the connection. The basic link layer operations are as follow [20]:
  - Advertising
  - Scanning
  - Connection establishment

- **Host/ Controller Interface:** Provides communication interface between the host and the controller [16].

Host: This can be introduced as the middle block of the BLE architecture. The Host can communicate with the BLE module via Host Controller Interface (HCI). Components of the Host Layer and their contributions can be described as follows.

- **Logical Link Control and Adaptation Protocol (LLCAP):** LLCAP performs as a protocol multiplexer and controls reassembly and segmentation of the data packets. In addition, this protocol produces logical channels by multiplexing channels over single or multiple logical links. The LLCAP used in BLE is a simplified and improved protocol, which has been developed using classic Bluetooth LLCAP [20].

- **Attribute Protocol (ATT):** ATT provides services related to data transmission between BLE devices. Therefore, this relies on a BLE or classic Bluetooth connection and allows read, write, notify and indicate attribute values through the connection [20].

- **Generic Attribute Profile (GATT):** The GATT organizes the individual attributes into a particular group of logical services (i.e. heart rate sensor is controlled by the Heart Rate Service). Moreover, GATT provides information about security levels and access profiles of the attributes [19].

- **Generic Access Profile (GAP):** This layer is dedicated for advertising, device discovering, open and manage connections and broadcasting data [20].

- **Security Manager (SM):** This provides services related to encrypt and decrypt data, bonding devices and enables device privacy [20].

Application: This layer contains the application logics, user interface and the overall application architecture. BLE networks developers require the knowledge on this layer in order the implement BLE oriented applications

2.1.2 BLE GATT Profile Hierarchy

The Generic Attribute Profile (GATT) describes the structure of services and characteristics based on the attributes in BLE architecture. In addition, it defines operations related to read, write, notify and indicate characteristics and technics that are used to configure the broadcast of attributes [16]. The position of GATT profile in BLE architecture is depicted in Figure 4. The ATT describes a flat set of attributes and techniques to access those attributes, but it does not describe the hierarchy of those attributes. Therefore, the GATT profile authorizes the server to define a hierarchy and then attributes are grouped into primary and secondary services. These services consist with different characteristics and those characteristics include their own set of permissions, which has been defined by GATT profile or an upper layer entity. Both the BLE and classic Bluetooth architectures comprise GATT and ATT profiles, because these profiles provide the preliminary capabilities to identify the services of a remote device. The GATT profile is a mandatory component of BLE architecture, but it is not a mandatory one in classic Bluetooth [16].

The GATT profile defines the client and server performances. Therefore, the GATT server collects the data has been transported over the ATT and acknowledges the requests from GATT client. In addition, it sends responses to those requests by indicating and notify the GATT client. GATT also defines the format of data contained on the GATT server [20]. The GATT profile data structure is depicted in Figure 4.

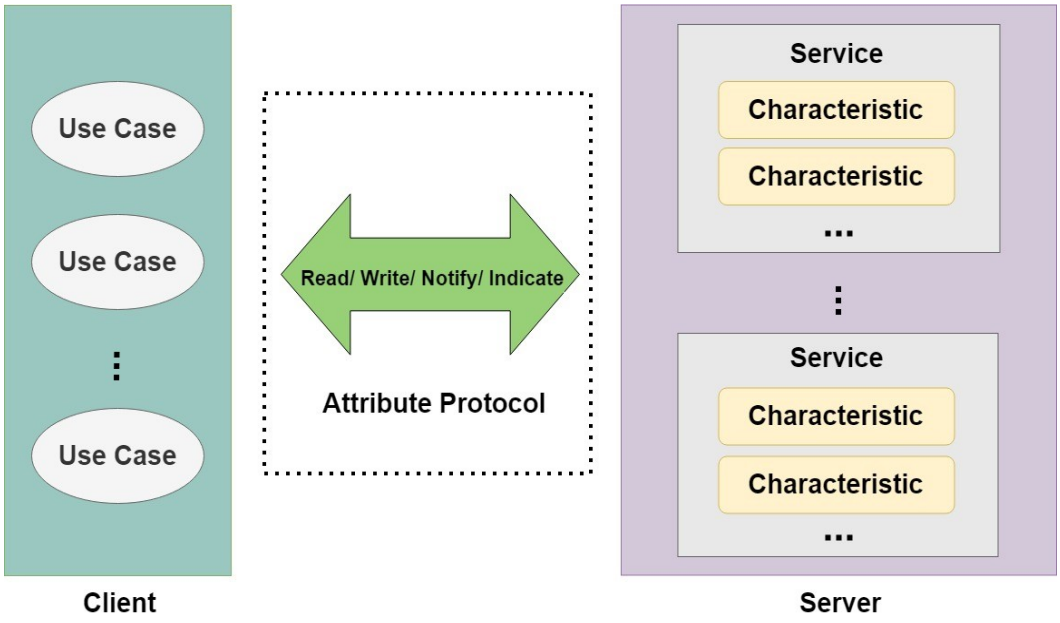

Figure 4. GATT data structure [21].

2.2 Edge Computing

The enabling technologies, which allows the computations to be performed at the edge of the network, are referred to edge computing. For the downstream data, edge computing is done on behalf of cloud services and for the upstream data; it is done on behalf of IoT services. The term “edge” can be defined as any computing procedure and network resources in between cloud data centre and data sources. The concept behind the edge-computing paradigm is to perform computing at the proximity of the data source [6]. Moreover, this paradigm is

interchangeable with fog computing [22]. Although most of the infrastructure applications focus of fog computing while edge computing focuses on IoT applications [6].

The bidirectional flow of computing used in edge computing have been depicted in Figure 5. In edge computing, the devices in the network are considered as data producers and consumers. Therefore, at the edge of the network IoT devices are capable to requesting services and data from the cloud and they execute the computing tasks as well. The edge is capable of computing offload, data processing, data storage, data caching, distribute request and service delivery from the user. Moreover, the edge itself has to be well organized to reach the requirements efficiently in service such as security, privacy protection and reliability [6].

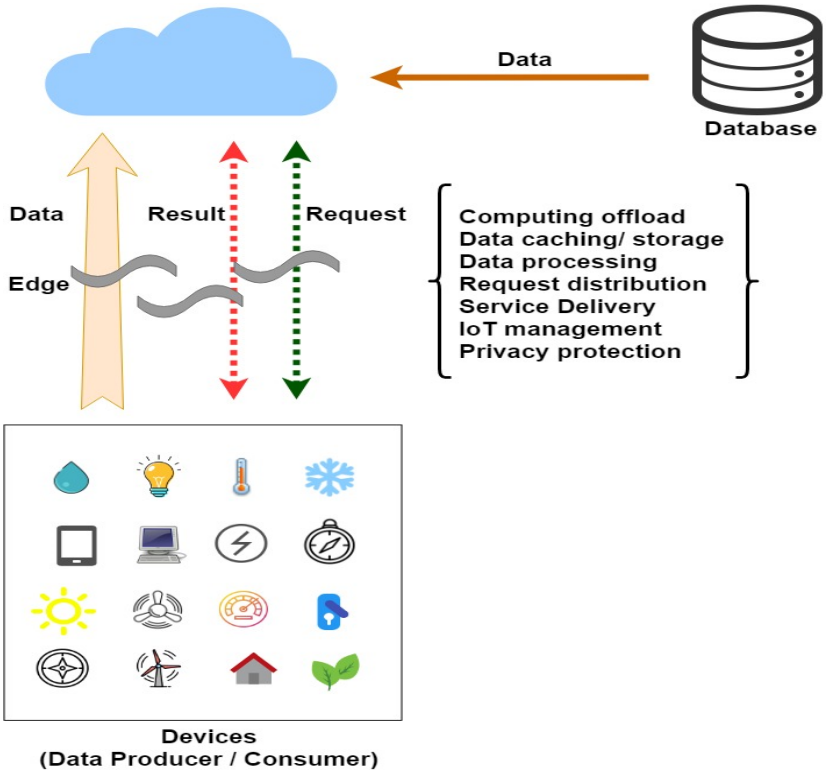

Figure 5. Edge computing paradigm [6]

Many benefits can be yielded from edge computing. As examples from the researches, when cloudlets are used to offload computing tasks for wearable cognitive-assistance systems, the response times can be improved by 80 to 200ms, [23] and power usage can be reduced by 30 to 40 percent. Response times and power usage can be reduced by 95 percent, by using CloneCloud technology for tested applications [24].

Edge computing has to deal with some of the issues and the practitioners need a good understanding of them for a better implementation. Those issues are system reliability, energy constraints, and inadequate wireless communications. Even with these issues, system developers can still implement the edge computing to provide fundamental functions [25].

Security and privacy are also considered as additional concerns. Having the data closer to the device provides better security. On the other hand, supporting for privacy and security can be an additional challenge due to topologies, inexpensive mobile devices and sensor unreliability [25].

Generally, the structure of the edge computing networks has been categorized into three main aspects [26] as,

- Front- end
- Near-end
- Far-end

A general architecture of edge computing networks is depicted in Figure 6. The differences and contributions among these sectors are described below in detail.

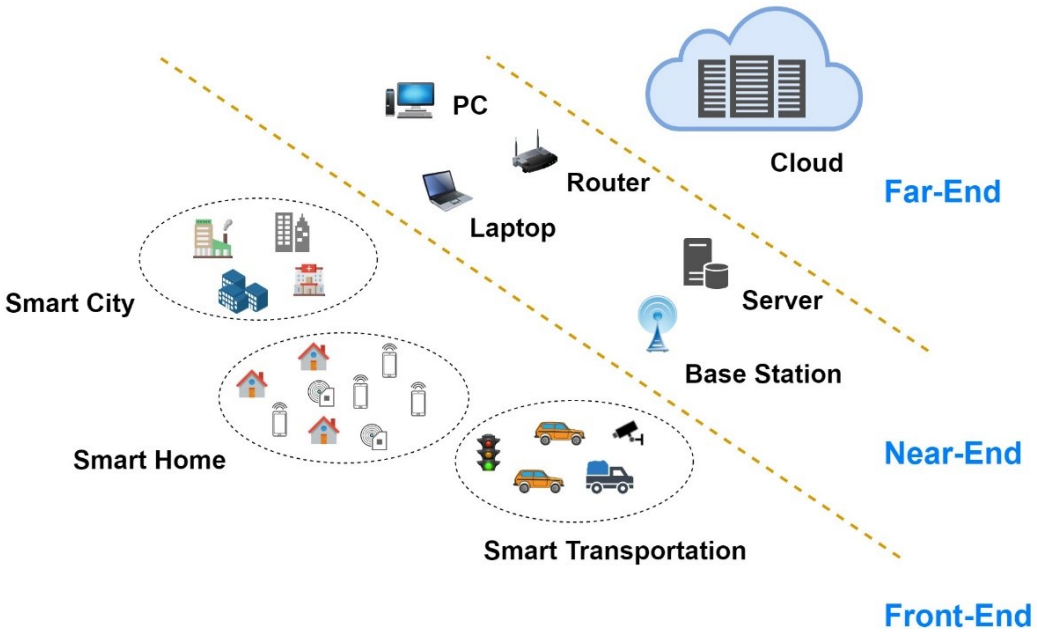

Figure 6. General architecture of edge computing networks [26].

1) **Front-End:** The front-end of the edge computing structure consists with various end devices (e.g., actuators, sensors). Therefore, the environment of the front-end region provides better cooperation and responsiveness to the end users. With the computing capacity of nearby end devices, edge computing is capable to catering real-time services to some applications. However, with the finite capacity of the end devices, not all the requirements at the front-end devices can be satisfied. Hence, the end devices have to send the resource requirements to the servers [26].
2) **Near-End:** Most of the traffic flows in the networks are supported by the deployed gateways in the near-end region. Edge or cloudlet servers provide various resource needs, such as data caching, computation offloading and real-time data processing. Moreover, in the edge computing paradigm, near-end environment caters most of the data computation and storage services. Hence, the users at the end of the network can achieve better performance on data storage and data computing with a small latency increment [26].
3) **Far-End:** The transmission latency has been increased due to the long distance between the cloud server and end devices. Nevertheless, the far end cloud servers can cater more data storage and computing powers. The cloud servers provide machine learning, massive parallel data processing, big data mining and management, etc [26].

2.3 ETSI Multi-access Edge Computing

Multi-access Edge Computing (MEC) can be considered as a key enabler for 5G networks which is an emerging technology nowadays. MEC is also compatible with present 4G networks and it will address many key applications of the 5G. One key aspect which motivates MEC is Internet of Things (IoT). ETSI defines the functional elements of MEC architecture. Key reference points and an overview of interactions of such elements of MEC are seen in [27].

The rapid increase of popularity of IoT devices contributes to creating a significant increase in demand for data processing and communication on the backbone network and the central cloud infrastructure. This will result in congestion in backbone network. MEC overcomes this issue by moving the computations towards the edge of the network.

2.3.1 ETSI Multi-access Edge Computing Architecture

A detailed comprehension on MEC can be attained from the reference architecture depicted in Figure 7, which defines the functional structure in a descriptive manner and their relations to each other. The main components of the MEC architecture and their contributions are described in Table 1.

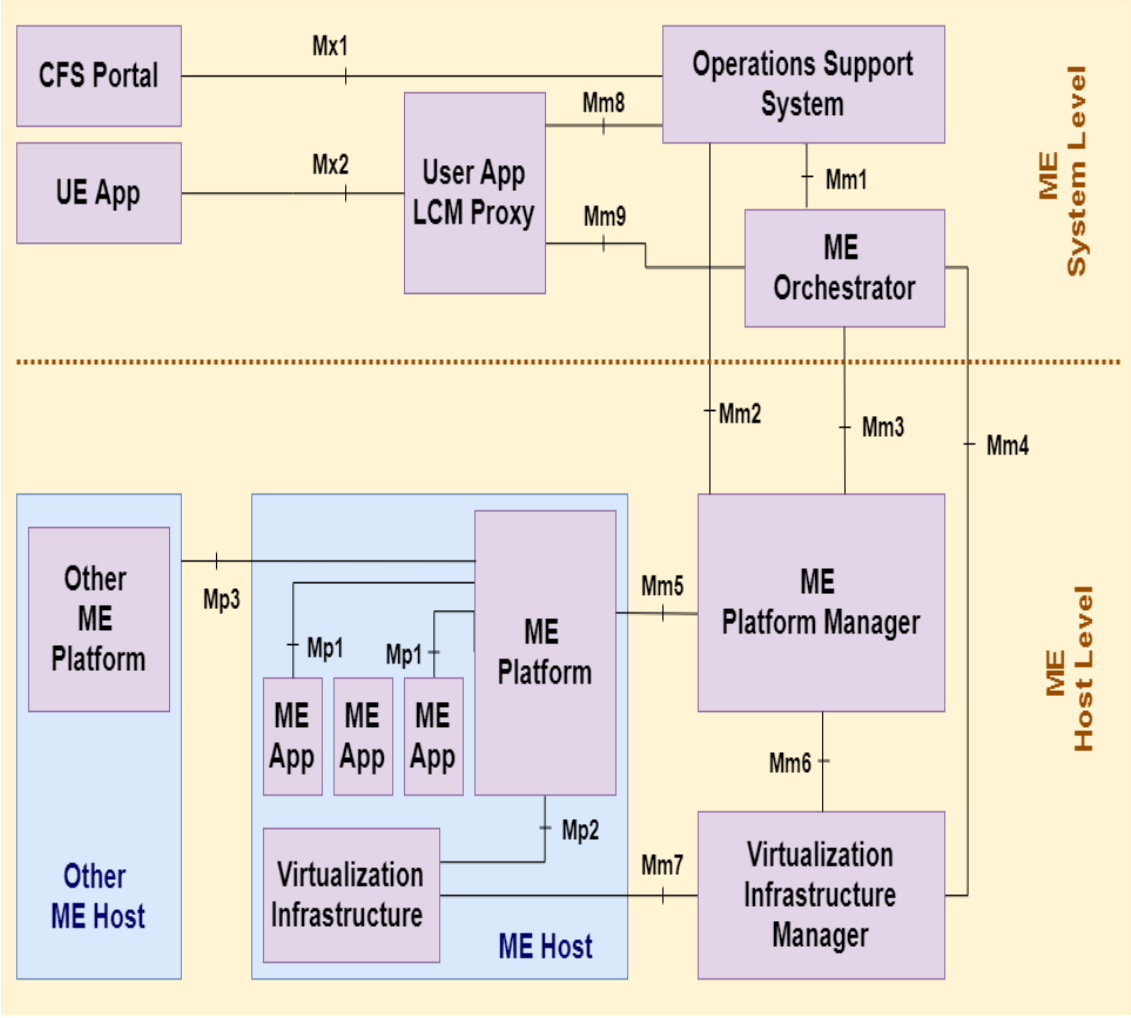

Figure 7. The MEC reference architecture [27].

Table 1. Description about MEC Architecture Components.

Architecture Component	**Description**
Mobile Edge Host (ME Host)	Provides storage, compute, and network resources upon which the MEC platform runs [28].
Mobile Edge Platform (ME Platform)	Consists of the basic operations needed for mobile edge applications to provide and access services on the mobile edge [28].
Mobile Edge Applications (ME App)	Software applications running on the virtualization platform of the mobile edge [28].
Mobile Edge Platform Manager (MEPM)	Manages the lifecycle of the ME Apps as well as the mobile edge platform [28].
Mobile Edge Orchestrator (MEO)	The central management entity in the mobile edge system, it oversees and manages all key functionalities of the entire edge system [28].

2.3.2 MEC and IoT

MEC will bring significant benefits to the mobile operators, third party service providers and also to the end customers. It will help in achieving high performance in terms of reduced latency and increased throughput. For the IoT systems which will be there with 5G, it will be easier in terms of technical aspects and will be more economical also because of the lesser traffic in the backbone network [29].

Security, Safety and Data Analytics

Security and safety have become two of the most important verticals for IoT because IoT connects the number of devices and these devices will transfer data between each other. MEC able to cater services for the analytics applications near the source. This allows to enhance performance, reliability and considerable savings by local data processing. With these concepts, security and safety aspects are also changed. Keeping the data close to the source means that the data is more secure than those are in the cloud. Because of this, the data is less vulnerable to attacks and making it safer for communications. MEC caters an automated, scalable, adaptable and profitable setting for network security. Analytics operation provides adaptable extensions to mediators via plug is integration [27].

Vehicle-to-Infrastructure Communication

Connected cars are an interesting use case which will frequently be seen in the future. This is because of the stringent latency requirements. Because of the low latency requirements, there is not much time to send the data along the backbone network and they need to be processed near the source of the data. MEC enables the concept of moving the processing towards the

edge of the network. It is predicted that the first commercially available self-driving car will be in the market in 2020 [27].

Computation Offload into the Edge Cloud

When the terminal devices have less computational power, computation offloading capability provided by MEC can be a very important feature. With this, the terminal devices can save power and by doing that they can have an increased battery life. Such offload may happen statically or dynamically. In addition, in this case, applications benefit from the low delay provided by MEC.

IoT can be considered as a real-world application of MEC. This is because MEC tries to bring the computation closer to the edge of the network and IoT enables that concept. Various applications depicted in Figure 8 show the value proposition of MEC. End user experience while a customer is using IoT service can be measured and this decides the utility factor. Utility factor provides the evidence of the value proposition of MEC [27].

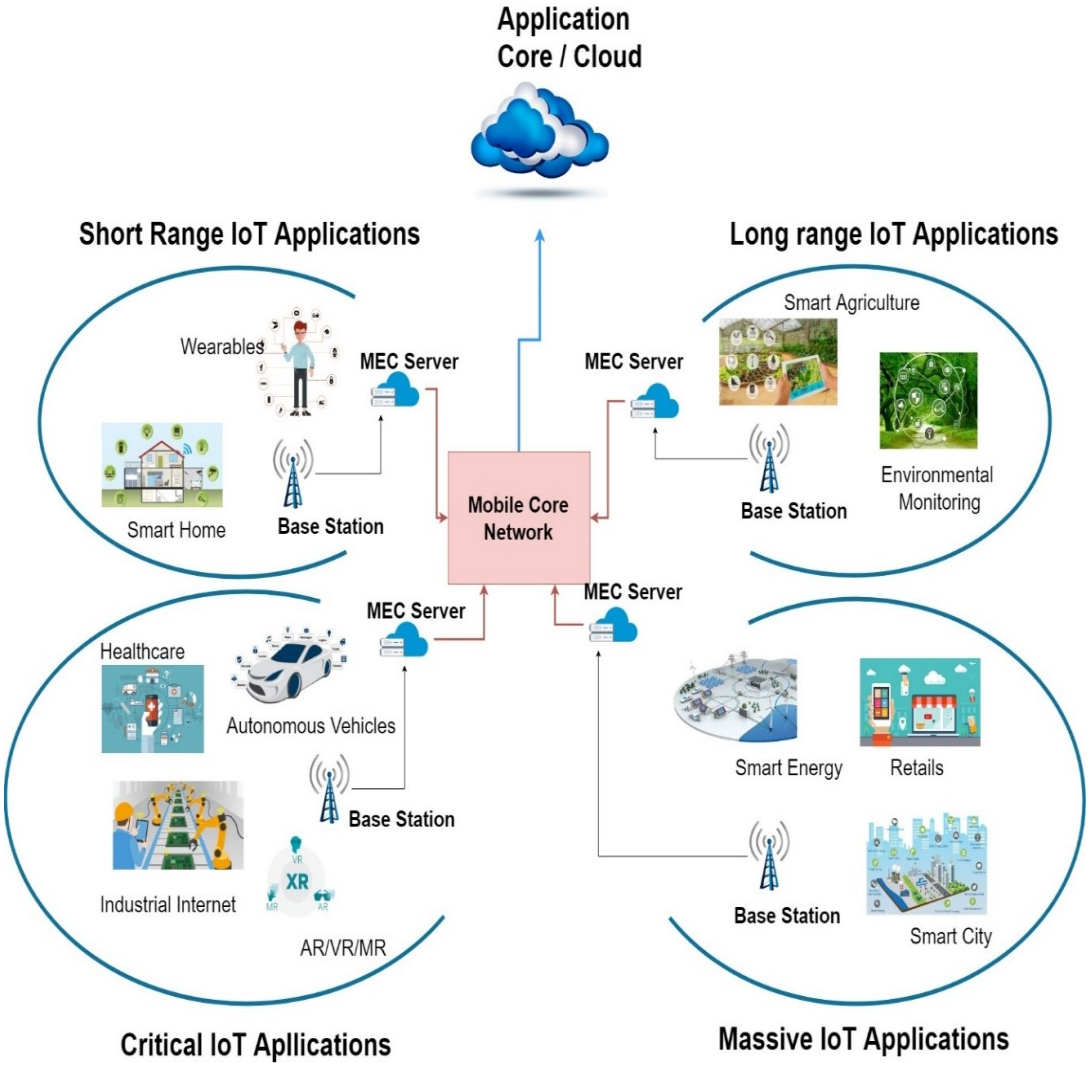

Figure 8. IoT and MEC application scenarios [7].

2.4 Dew computing

Dew computing is a novel computing paradigm, which merges the cloud computing with the capabilities of local computing. In cloud computing, users are able to store their data in the cloud server and simply they can use their computers to access the cloud. The main advancement of the cloud computing is data mobility; user can access his/her data from anywhere using internet. The main disadvantage of this concept is that the user highly depends on the internet connectivity. Hence, the Cloud-Dew Architecture [30] proposed as a feasible solution to the offline data accessibility problem [31]. Figure 9 illustrates comparison between cloud computing architecture and dew computing architecture.

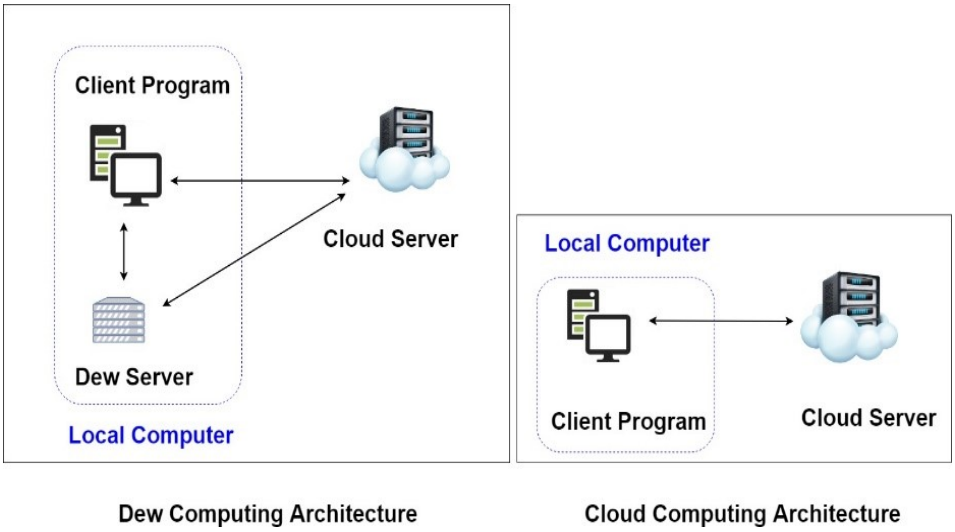

Figure 9. Dew Computing vs Cloud Computing [30].

Compared with the cloud server architecture, the novelty of the dew computing architecture is that the dew server which is a web server that occupies on user's local computer [30]. The dew server and its database have two main functions,

- Caters services similar to the cloud server [30].
- Synchronizes the dew server database with the cloud server database [30].

Characteristics of the dew server can be briefed as follows,

- Lightweight and normally it serves only one client [30].
- Size of the dew server is very small in comparison with the size of the cloud server, usually it stores users own data [30].
- Dew server can be vanished easily, due to different factors like hardware destruction and viruses [30].
- It is easy to recreate a vanished dew server since the cloud server has the copy of dew server data [30].
- Accessible both online or offline [30].

Currently, dew computing is not limited to research area. It has now been expanded to application area as well. Nowadays, many researchers try to implement dew computing in Internet of Things (IoT) streaming, medical care, indoor navigation, and cyber physical systems [31].

2.5 Wireless Sensor Networks (WSN)

Wireless Sensor Network (WSN) is a set of sensor elements which are deployed in a particular environment to capture certain data related to the behaviours of the deployed environment. Typically, WSN consists of many types of sensors such as temperature sensors, humidity sensors, infrared sensors, motion sensors etc. These different types of sensors are capable of monitoring a significantly larger area and different physical parameters such as [32]:

- Evaporation,
- Soil makeup,
- Temperature,
- Vehicular motion,
- Pressure,
- Lightning,
- Noise fluctuation,
- Existence of a given object,
- Level of Mechanical force on certain objects, and
- Characteristics of certain object's velocity.

Usually, WSN are equipped with the large number of sensors. One of the main features is that those sensors are low power sensors because they continuously monitor, and power is a critical factor when it comes into day to day operation. The communication of such sensors can be direct, which means that the sensors can talk to each other. On the other hand, they can talk with a centralized server also. To get an accurate measurement it is always better to increase the number of sensors. Figure 10 depicts the components of a sensor node. Processing unit, transmission unit and power unit are the main units of a sensing unit [33].

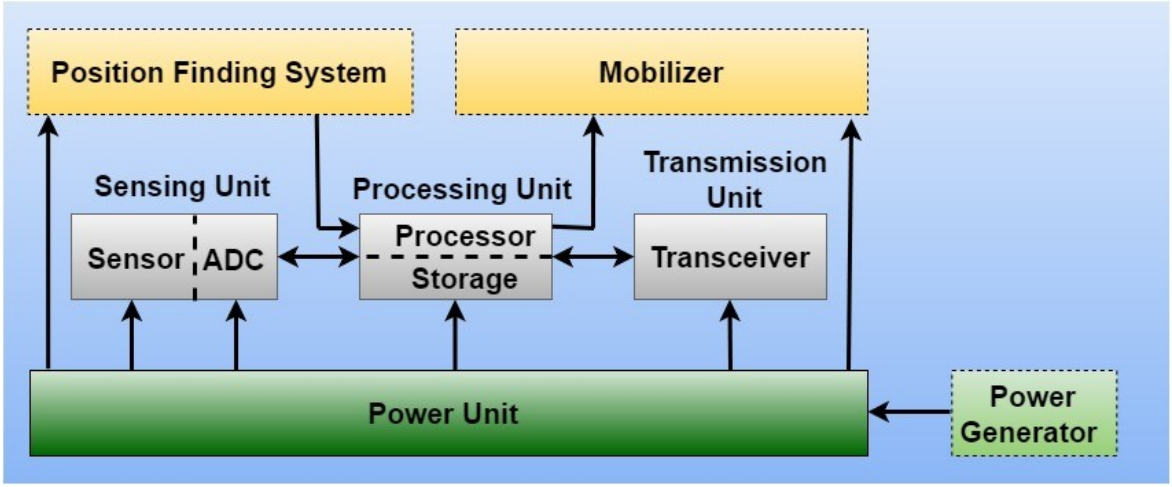

Figure 10. Components of a sensor node [33].

Figure 11 illustrates the communication architecture of a WSN. Typically, sensor nodes are deployed in a sensor field and these sensor nodes communicate with each other in order to produce the high-quality accurate information regarding the behaviour of their physical environment. Each sensor node in a WSN takes its decisions based on its mission, knowledge of computing, energy resources and communication. Moreover, these sensor nodes are capable of data collecting and routing to other sensor nodes or back to a gateway. The gateway can be

a fixed or mobile node that allows connection between sensor network and communication infrastructure or the internet where a user can have access to the collected data [33].

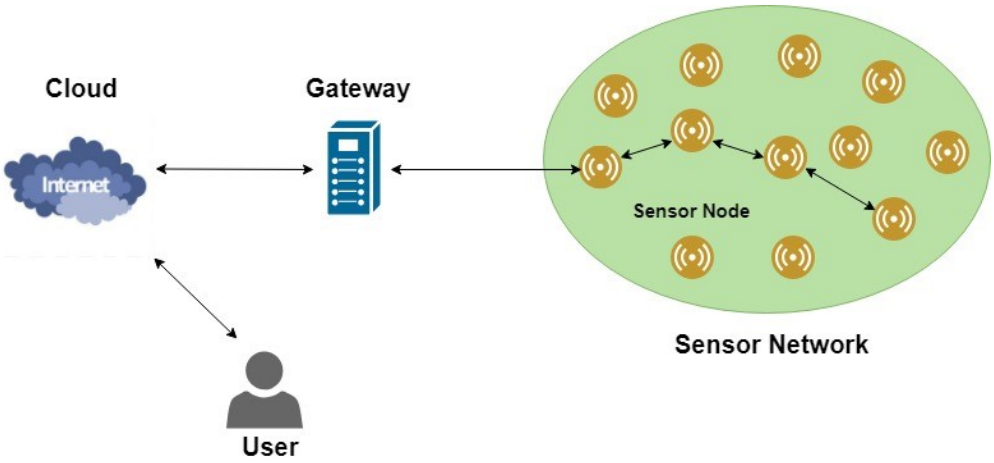

Figure 11. Communication architecture of a WSN [33].

Compared with the traditional networks, WSNs contains its own its own design and resource constraints. Moreover, these resource constraints consist of limited energy, low bandwidth, short-range communication and limited data storage and processing in each node [34]. Broad classification of various issues in a WSN are depicted in Figure 12.

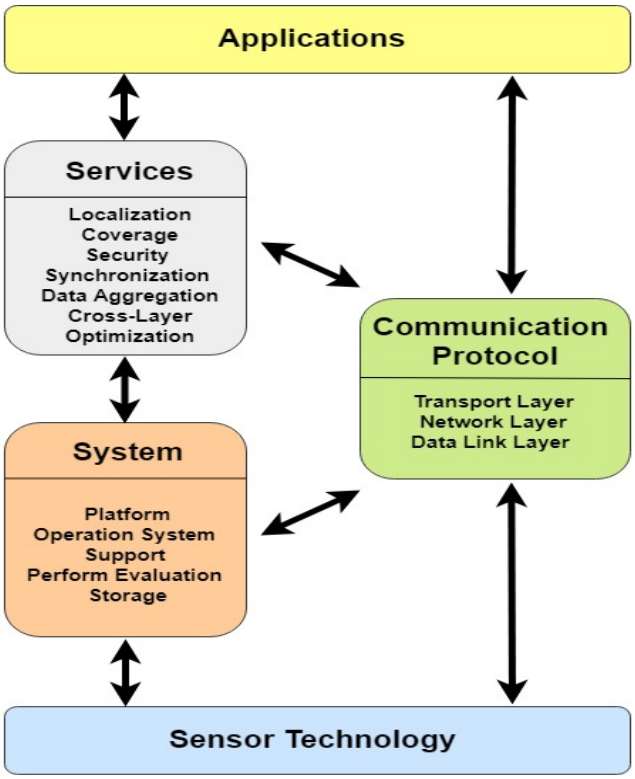

Figure 12. Broad classification of various issues in a WSN [34].

There are two main categories of WSN applications. They are monitoring and tracking. Health and wellness monitoring, indoor/outdoor environmental monitoring are some of the key

applications belong to the monitoring domain. Other applications include power monitoring, inventory location monitoring and structural monitoring. On the other hand, tracking applications are used for tracking purposes. Examples are object tracking, vehicles and humans tracking. Figure 13 illustrates few of many applications that can be deployed and tested, especially in real environment [34].

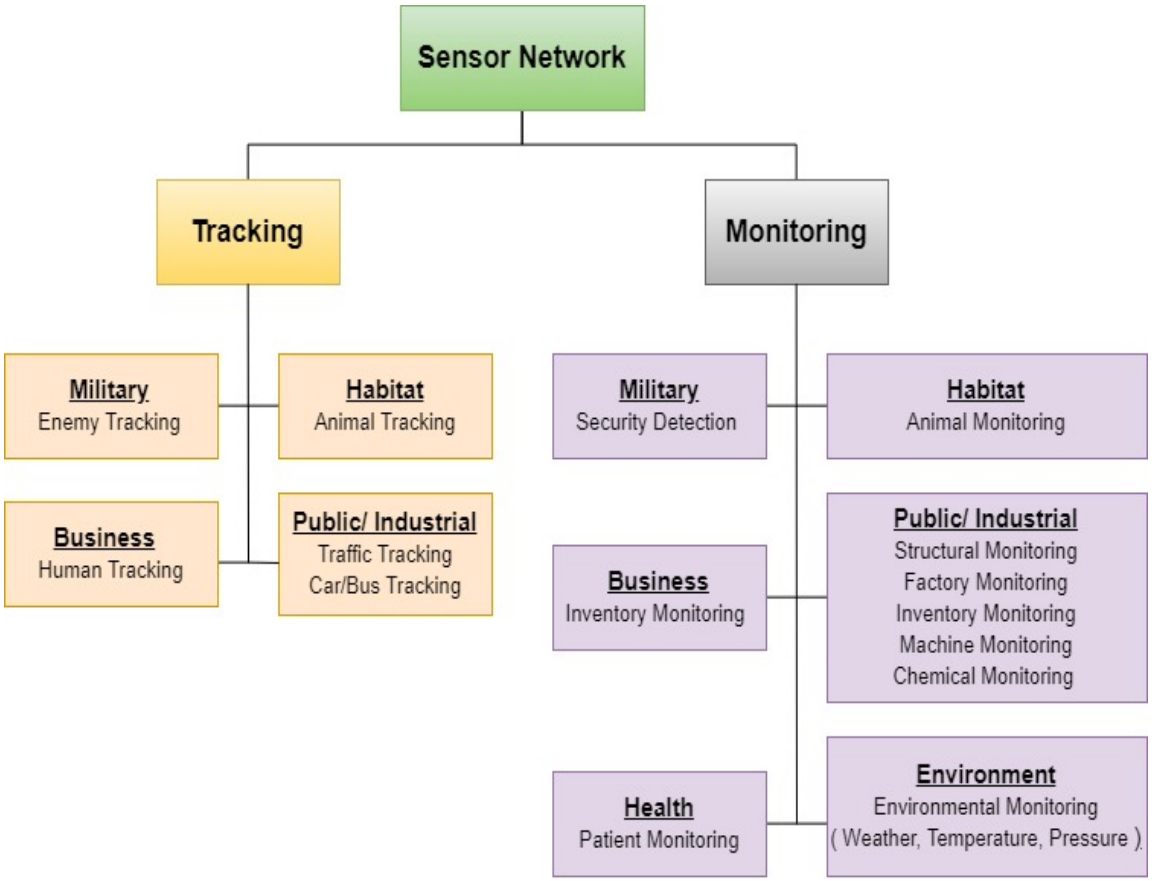

Figure 13. Overview of the sensor applications [34].

2.6 Unmanned Aerial Vehicles (UAV)

Recently, the interest in Unmanned Aerial Vehicles (UAVs) has been rapidly increased due to their lack of modern features. With the evolution of high-power density batteries, low power and long-range micro radio devices, low cost airframes and structures and powerful motors and microprocessors, small UAVs or drones' haves become suited in many civilian circumstances. Therefore, UAVs have been used for remote sensing, traffic monitoring, mapping, search etc. [35]. Most of the small UAVs are lightweight, easy to assemble and operated. Therefore, the UAVs provide potential for collecting remote data faster and at low cost than the piloted Ariel vehicles [36].

The small UAVs caters a latest controllable platform for data acquisition in remote areas. Therefore, the mobility of UAVs permits remote data acquisition in the environments where inaccessible humans such as forest fire, volcanos, transportation disasters, etc.

Choosing the appropriate type of UAV is a decision that is depending on many factors such as the desired Quality of Service (QoS), federal regulations, environmental factors and application objectives. To use a UAV properly for a given application, the properties of UAV

such as flying altitude, flying time of the UAV should be considered. Generally, UAVs can be categorized into two types, considering their flying altitude. They are low altitude platform (LAPs) and high-altitude platforms (HAPs). LAPs able to fly within the range of tens of meters to few kilometers. LAPs are flexible and take quick movements. The average altitudes of the HAPs are above 17km and usually quasi stationary [37].

2.6.1 UAVs in IoT Platform

Nowadays, UAVs are used to carry out many tasks. Amazon parcel delivery and SharperShape power line monitory are two such examples. The capabilities of today's UAVs can be used in offering various value-added services. Especially when they are equipped with remotely controllable IoT devices, there are many uses of UAVs in IoT world. In this way, UAVs will create an innovative UAV-based IoT platform operationally in the sky. Having this will help to reduce any capital and operational expenditure in case of establishing a new communication system. Using these deployments, UAVs equipped with remotely controllable IoT devices can collect data from IoT sensors whenever and wherever needed [38].

Based on the nature of the collected data and the data capacity the amount of energy to process such data is decided. If the energy requirement is high, the data processing can be offloaded to cloud servers on the ground and if the energy requirement is low, the data can be processed on board and UAVs and actions based on the processed data can be taken. To build such an ecosystem, it is mandatory to have a platform orchestrator, which has the knowledge of diverse contextual details about UAVs, such as their flying paths, their IoT devices, and their battery life [38].

In general, these devices do not have the capability to transmit over long distances because of the constraint in the battery life. In these cases, UAVs able to approach IoT devices, extract data and transfer those data to the external devices that are inaccessible to transmitters. Hence for these applications, UAVs acts as mobile aggregators or base stations of the IoT network. Still there are some deployment challenges such as optimal positioning, power efficient operations of UAVs and mobility [39].

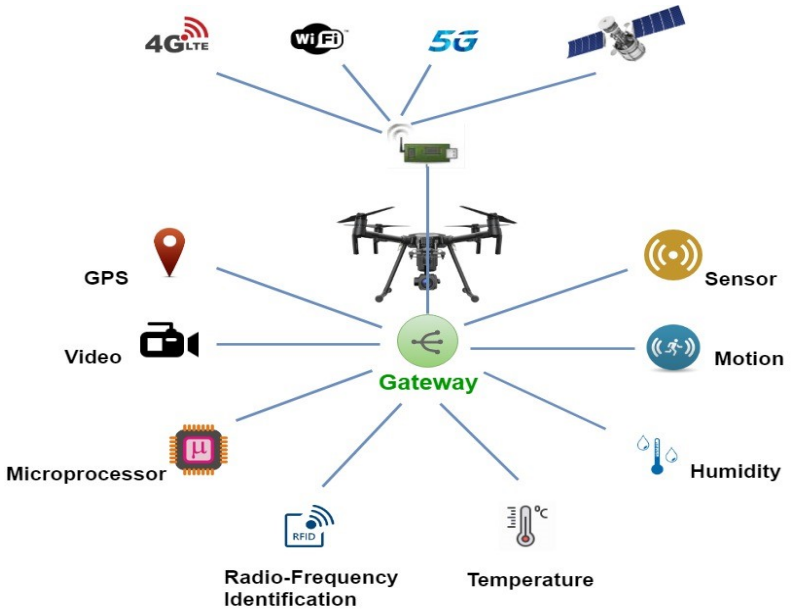

Figure 14. UAVs equipped with diverse IoT devices.

2.7 Use Cases

Use case 1: Instead of applying chemicals to the crops in a traditional way, set of UAVs can be used to spray chemicals into the crops. In this case, UAVs need feedback from the ground and it is provided by WSNs deployed. Monitoring of frosts in vineyards can also be considered as another application. In all these use cases, it is fundamental that there will be a moving node accompanied by a UAV ensures the communication between the base station and the field equipment [40].

Use case 3: Forest fire is a catastrophe and once occurred it is nearly impossible for humans to reach to an area in close proximity. Monitoring the conditions of a forest fire can also be done using wireless sensor networks. By using the wireless sensor network and UAVs, values related to temperature and humidity can be read and the fire fighters can act accordingly based on the inputs received to them [41], [42].

Use Case 4: Monitoring the conditions of a river can be considered as another use case. With the collected data using UAVs, conditions of river can be modelled, and this information can later be used to quick incident responses [43].

Use case 5: To reduce the frequency of accidents happening in mining industry, UAVs and wireless sensor networks can be used. This will help to minimize the losses to human lives and also provide a way of detecting disaster situations in advance and enable the workers to take appropriate actions before a disaster actually happens. As examples, the falling of roofs, falling of walls, sudden flooding can be monitored [44], [45].

Use Case 6: A bridge has a lifetime and during that lifetime, it undergoes to various kinds of damages. Hence, maintaining bridges is an important thing to ensure the safety of people who are using the bridge. Monitoring will help to significantly improve the lifetime of a bridge [46].

Use Case 7: During the rehabilitation period of a person who is subjected to an accident, he has to perform a series of exercises and activities which will enable him to recover to his initial conditions. To gain a maximum outcome of a rehabilitation period, the patient has to be continuously monitored to avoid any improper exercises and make sure everything is done in a correct way. A wireless sensor network can be used to monitor such patients. In such cases, a small wireless device can be attached to the patient's body, which allows the doctors to examine the accuracy of the performed exercises. These devices are capable of providing accurate information without much involvement of the patient [47]. Using wireless sensor networks will help practical and innovative applications to be used in improving the quality of life [48].

3 PROPOSED ARCHITECTURE

Recently, Unmanned Aerial Vehicles (UAV) or drones have been used for various autonomous monitoring applications. Therefore, by introducing UAVs into WSNs can improve the remote surveillance of the systems. With the advancement of UAV technology, monitoring systems have improved their remote sensing platform with mobility, productiveness, cost effectiveness, etc. Nevertheless, the contribution of the UAVs to the WSNs can depend on various parameters such as time, position, speed of the UAV communication media and path.

The next generation of WSNs will be more useful when the sensed data is preserved in a cloud server for their further use. Hence, the traditional computing techniques and approaches will be replaced or discover a place in data manipulation prior to the data being moved into the cloud server [49].

In this thesis work, we propose a novel edge-based architecture which entails UAV data retrieving, data processing, and data management in a heterogeneous Wireless Sensor Network (WSN) located in a remote area [15]. In order to explain the concept and necessity of integrating end to end secure communication between UAV and low power IoT devices in a real-world practical scenario, we have considered a smart agriculture use case which has been located in a remote area with diversified sensors. In addition, we have assumed that this agriculture site has to be monitor frequently and difficult to reach by humans frequently.

In this chapter, we describe the system model of the proposed architecture and communication protocols that has been used in the data collection procedures in the remote field.

3.1 System Model

In Figure 15, we depict the system model that we have considered for UAV assisted edge computing-based architecture.

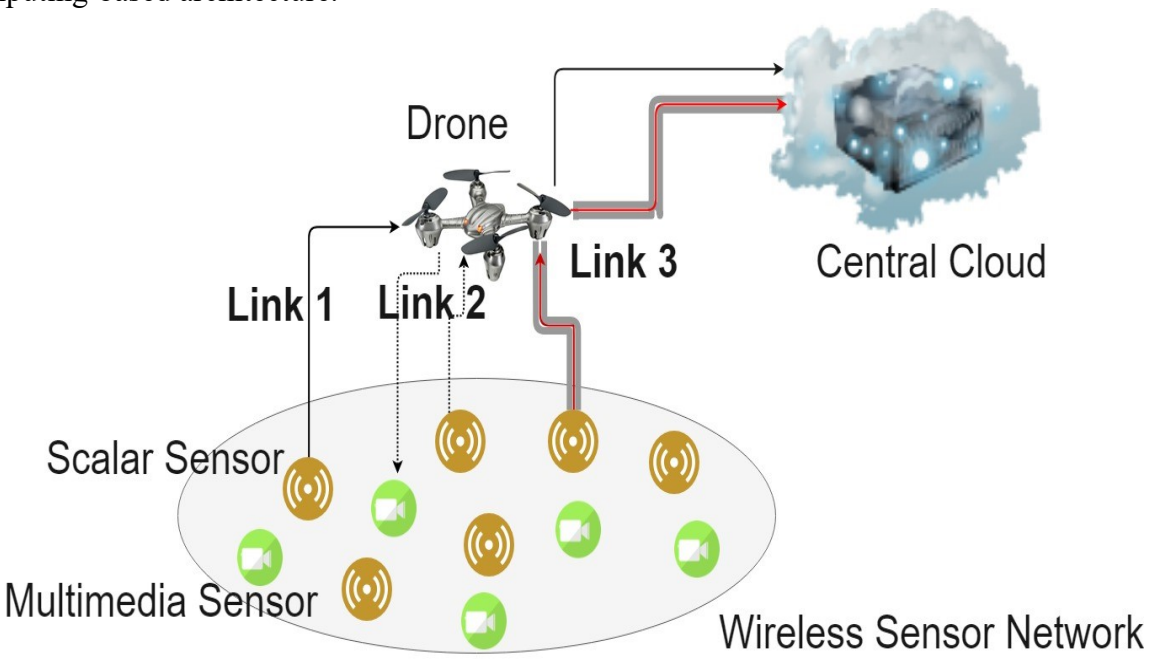

Figure 15. Proposed System Model [15].

In the proposed system model, we have considered heterogeneous IoT devices that contain assorted sensor nodes with different capabilities whereas the UAV performs as the edge server of the system. Moreover, the drone can be considered as a lightweight server, which is capable of serving one sensor at a particular time. The drone can cater end-to-end connectivity from sensor to cloud when there is Internet connectivity or allow offline accessibility otherwise. Hence, we contemplate that the drone will contribute to the system model by in sight data processing and take the control functions of the IoT sensor network.

The WSN of the proposed architecture, can be consist with sensor type nodes that monitor and collects data belongs to various environmental conditions or the actuator type devices that perform given actions based on the received commands. The sensor type nodes or scalar sensors can be used to monitor humidity, temperature, light, soil pH and many other environmental conditions of the remote agricultural field. The actuator type devices can be used to collect multimedia data according to user requirement.

In order to explain the data transferring procedure from sensor network to cloud server via drone, we have introduced three types of links that can be exists in the proposed setting. These three links are depicted in Figure 15.

- **Link 1:** Scalar sensor collects data in the remote field and stores them inside its internal memory. When the drone comes to the remote location, sensor starts transferring saved and real time data to the drone. Then drone save those data into a database and send them to the central cloud when it returns to the home location. Moreover, drone can access and process the retrieved data [15]. This link is depicted in solid line in Figure 15.

- **Link 2:** Scalar sensors collects data in the remote field and save them into its internal memory. When the drone comes, scalar sensor starts sending real time data and saved data to the drone for further processing. Then the drone processes those data and controls actuator nodes based on processed outcomes. Hence, no data is transferred to the central cloud. Moreover, the drone will provide offline data accessibility to the actuator nodes, even though without real-time access to the central cloud [15]. Link 2 is depicted in dotted line in Figure 15.

- **Link 3:** Scalar sensor senses data from its environment and save them in to its internal memory. When the drone comes, the scalar sensor will send saved and real-time data to the drone in an encrypted format. Therefore, the drone is acting as a relay and it cannot decrypt those data. Then the data offloading from the drone to cloud can be performed once the drone returns to the home location [15]. This link has been introduced to address the use cases which require data privacy. Link 3 is depicted in red line in Figure 15.

These proposed links can be programmed according to user requirements or according to the nature of the use case. Also, the user can control the drone, send control commands, or program the flight route whenever needed. Assuming the internet inaccessibility at the remote agriculture field, we have proposed BLE communication to implement the connectivity between drone and sensor nodes. The proposed system model supports edge computing architecture to provide both insight and remote processing capabilities to the remotely located IoT sensor networks.

3.2 Communication Protocol

In the proposed architecture, we have introduced three main communication protocols. Two protocols have used for communication between drone and sensor node at the remote location and one protocol has used for communication between drone and central cloud at the home location. To establish the connectivity between the drone and sensor node at the remote location we considered BLE as the communication media between those two components. Internet accessibility will provide the connectivity between drone and central cloud at the home location. Therefore, the following three communication protocols have used in the proposed architecture.

To communication between drone and sensor node:

- Communication Protocol used for data upload process
- Communication Protocol used for data download process

To communication between drone and central cloud:

- Communication Protocol used for data transfer process to the central cloud

3.2.1 Data upload process

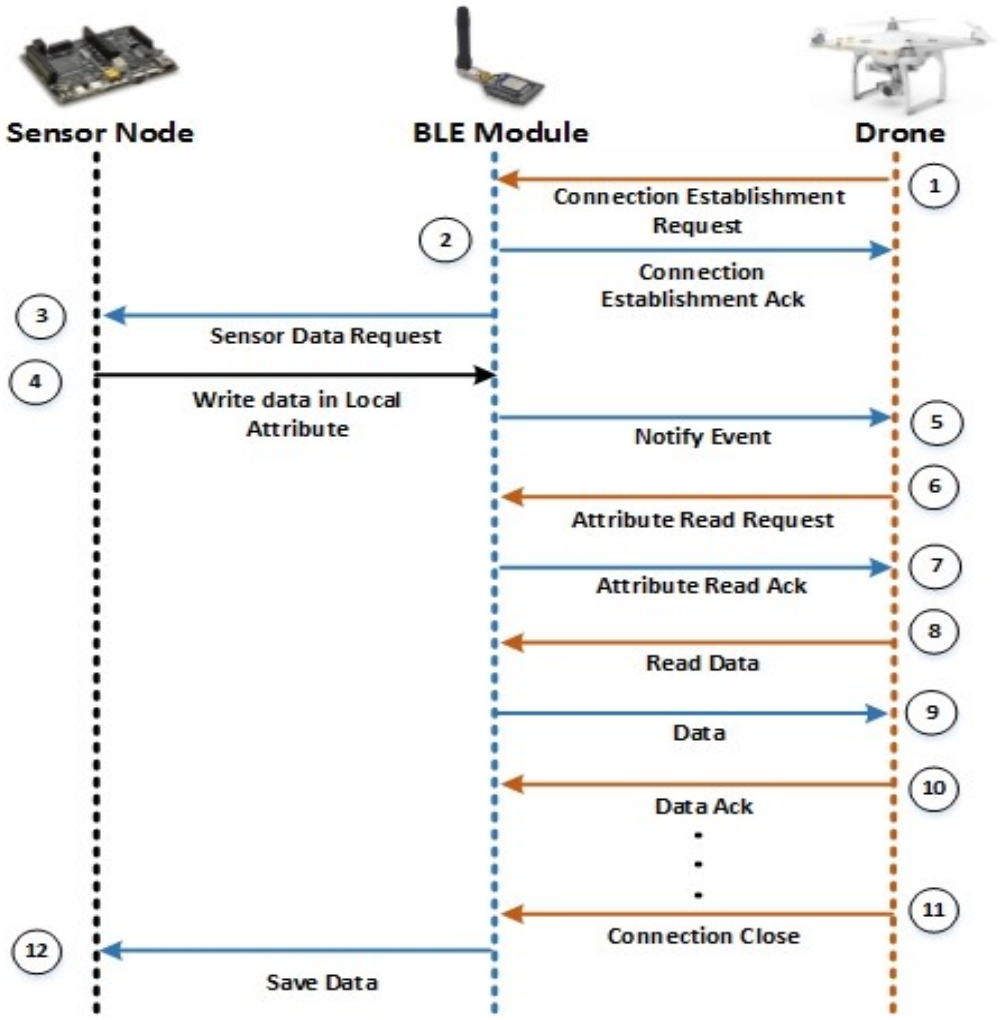

Figure 16. Communication Protocol for data upload process [15].

Here we have considered that every sensor node is equipped with BLE module which enables BLE communication between sensor node and drone during data transferring process. The message sequence between the sensor node, BLE module and the drone for the data upload process from sensor node to drone is illustrated in Figure 16.

When the drone comes to the range of sensor node, the drone initiates the connection establishment by sending connection establishment request to the BLE module of the sensor node. When the connection request received, BLE module acknowledges the request and connects with the drone. Then BLE module sends sensor data request to the sensor node by requesting both saved and real time sensed data. After that, sensor node starts writing those data on a pre-defined local attribute of the BLE module. In the next step, BLE module notifies the drone that it has data to be transferred to the drone. When the drone received that notification, it sends attribute read request to the BLE module. Once the attribute read request is acknowledged by the BLE module, the drone starts reading real-time and saved data available in the local attribute of the BLE module. To ensure a reliable communication between BLE module and drone, drone sends data received acknowledgement to the BLE module every time it reads data from the BLE module. BLE module does not allow next data reading until it receives the data acknowledgement from the drone. This data upload process continues until drone collects sufficient information from the sensor node. Then, drone terminates the connection between drone and BLE module and returns to the home location or to the next sensor node in the remote WSN. Once the drone has terminated the connection, BLE module convey a save data command to the sensor node.

Afterwards sensor node starts saving sense data into its internal memory until the drone comes back in the next flying cycle.

3.2.2 Data download process

This protocol has been introduced to use whenever the drone needs to send commands or sensor update data to the sensor node. The message sequence between drone, BLE module and sensor node during the data download procedure is depicted in Figure 17.

Similar to the data upload communication protocol, drone initiates the connectivity with the sensor node by sending a connection establishment request to BLE module of the sensor node. Then drone receives the acknowledgement request from BLE module and establish the connection to transfer the data from drone to sensor node. When the connection process succeeded, drone sends back an attribute write request to the BLE module. After that BLE module acknowledges the attribute write request from the drone and start retrieving data from drone. At the same time, BLE module conveys a command to the sensor node to save those data to its internal memory. After saving the retrieved data from the drone, sensor node sends back an acknowledgment to BLE module. Drone also receives an acknowledgement from BLE module, so that it knows the data has been successfully transferred. This process continues until drone sends the required information to the sensor node. In this scenario we have considered sensor node is capable of processing the retrieved data from drone. After successful completion of data download process, drone terminates the connection between drone and the BLE module and fly back to the home location or to the neighbouring sensor node of the WSN.

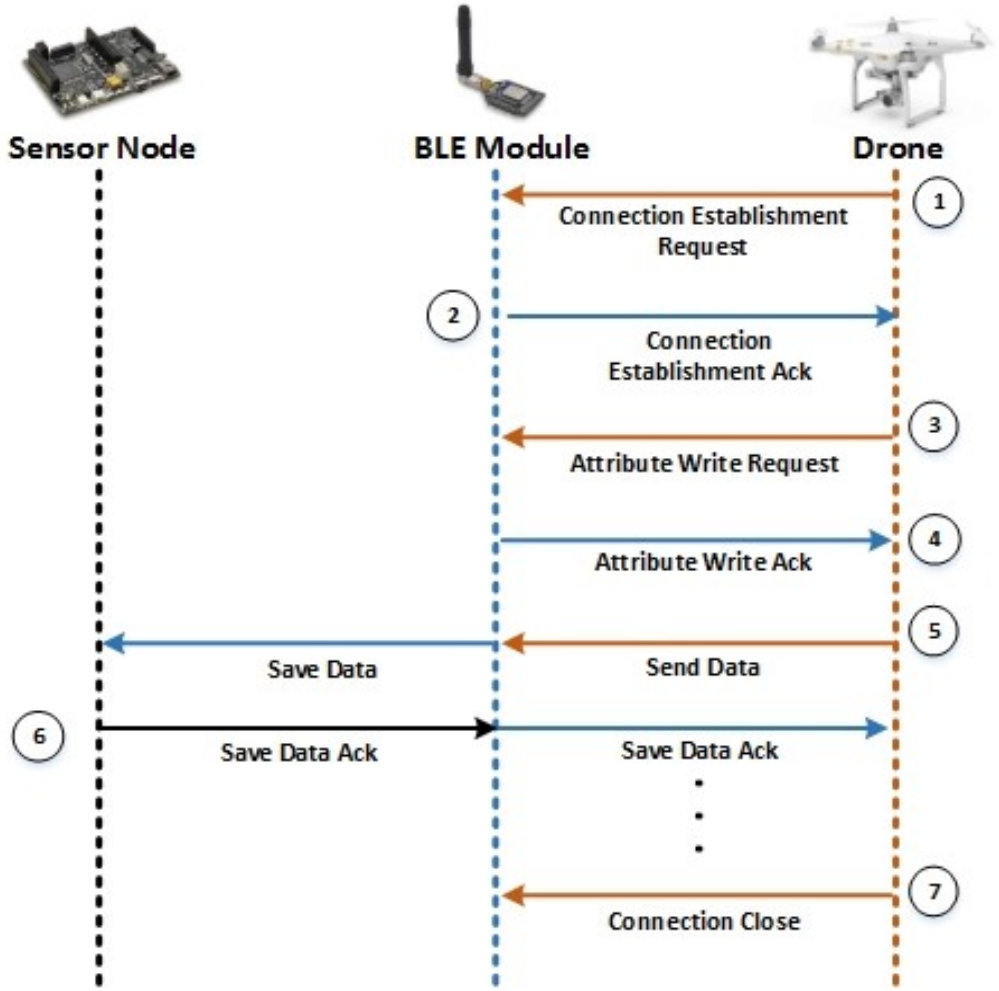

Figure 17. Communication protocol for data download process [15].

3.2.3 Data transfer process to the cloud server

This communication protocol has been introduced for the data transfer process from drone to central cloud server at the home location. Hence, the relevant message sequence between drone and central cloud server has been illustrated in Figure 18.

When the drone arrives to the home location, it connects with internet and send an authentication request to authenticate itself to the cloud server. After receiving the authentication request acknowledgment from the cloud server, drone sends a connection request to the central cloud server. Then central cloud server sends a connection approved message to the drone and establish the connection. Next, the drone starts data transfer process to the central cloud server and cloud server will save transferred data into a database for further use. When all collected data is transferred to the central cloud server, drone convey a cease connection request to the cloud server to terminate the connection. If the data has been successfully received to the cloud server, it will acknowledge the connection request of the drone and terminate the connection. Then drone will clear its internal database and return to the changing station or begin the next flying cycle.

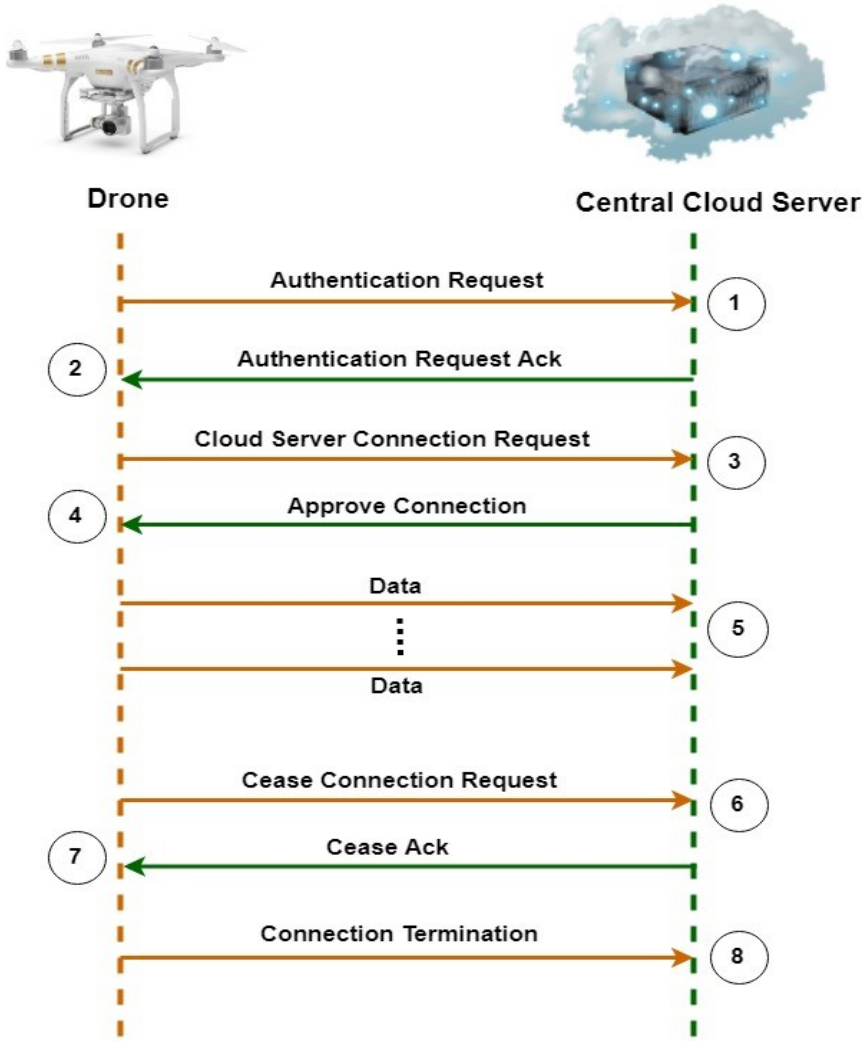

Figure 18. Communication protocol for data transfer process to the central cloud.

4 EXPERIMENTS

4.1 Simulation

To examine the performance of the proposed architecture, we implemented a simulation using MATLAB simulator and obtained the results graphically. In this section we describe the mathematical background and the practical parameters that have been used for the simulation. The final outcome of this simulation represents the relationship between total flying time of the drone and the numbers of sensor nodes that can be covered by the drone during data collection procedure.

Considering practical implementation of a WSN, we examined three sensor nodes array types that has depicted in Figure 19. Therefore, we considered linearly deployed sensor nodes array, circularly deployed sensor nodes array and a squared sensor nodes array to observe the relationship between the total flying time and the numbers of sensor nodes that can be cover by the drone within its one flying cycle during data collection procedure. Here we have assumed that the drone starts its flying cycle from the first sensor node of the array [15]. The flying cycle of each array has been depicted in blue line in the Figure 19.

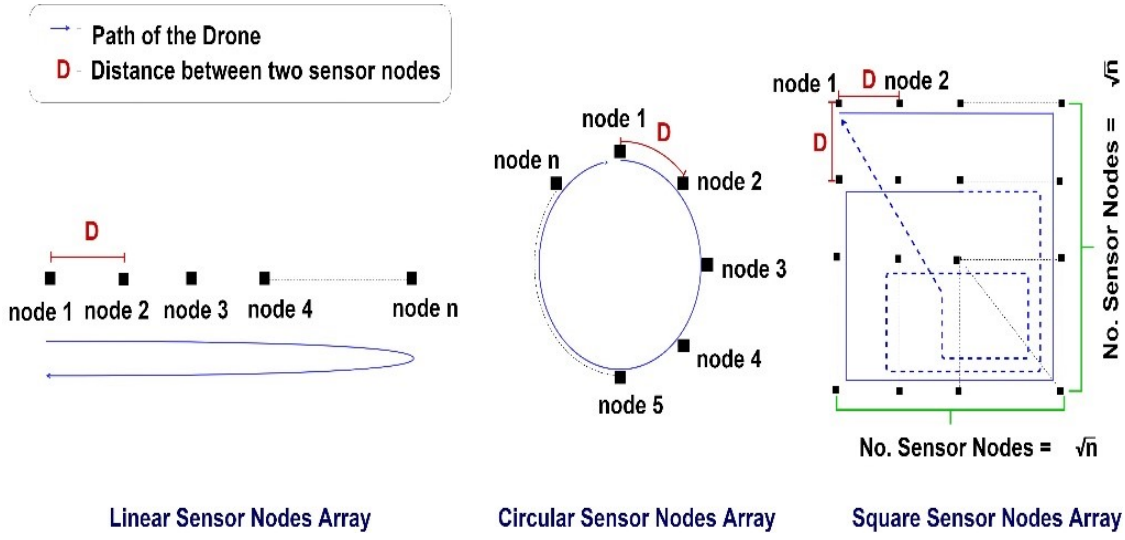

Figure 19. Orientation of Sensor Arrays.

Table 2. Number of Sensor Nodes and Drone Flying Path Length in Sensor Nodes Arrays

Sensor nodes array type	**Number of sensor nodes in the array**	**Length of the drone flying path**
Linear sensor nodes array	n	$2D(n-1)$
Circular sensor nodes array	n	$D \times n$
Square sensor node array	n	$D(n-1)+\left(\sqrt{\frac{(\sqrt{n}-1)^2}{2}}\right)D$ **Note:** $\sqrt{n}$ considered as an odd number

As depicted in Table 2 and Figure 19, each sensor nodes array type contains *n* number of sensor nodes. The length of the flying path of the drone has been changed due to the orientation of sensor nodes array. The area of the squared senor nodes array has been considered as follows:

$$D\sqrt{n} \times D\sqrt{n} = D^2 n$$

To simplify the simulation procedure, we have only considered square nodes arrays that constructed with number of sensor nodes per side equals to $\sqrt{n}$ where $\sqrt{n}$ be an odd number.

Using the following waiting time equation (Equation 1) we have derived an equation for total flying time of the drone during one flying cycle that can be achieved with full charged battery. Here T_W represents the waiting time of the drone near a one sensor node.

$$T_W = T_{data\ upload} + T_P + T_{DL} \quad (1)$$

Where, T_P is the data processing time inside the drone, T_{DL} is the data download time from drone to sensor node. The following data upload time equation have been used to simplify the Equation 1.

$$T_{data\ upload} = \frac{R_S(T_C + T_F)}{R_{BLE}}$$

T_F is the total flying time of the drone, assuming that the drone starts flying from the first sensor node of the array. T_C is the battery charging time of the drone. R_{BLE} is the data transfer rate between drone and sensor node, R_S is the data generation rate of sensor node. Therefore, the simplified equation for drone's waiting time near a sensor node are shown below (Equation 2). Here all the time parameter values are given in seconds.

$$T_W = \frac{R_S(T_C+T_F)}{R_{BLE}} + \frac{T_P}{60} + \frac{T_{DL}}{60} \quad (2)$$

The equation for total flying time (T_F) of the drone during one flying cycle can be derived as follows,

$$T_F = \begin{pmatrix} Total\ waiting\ time\ at\ the \\ sensor\ nodes\ in\ the\ array \end{pmatrix} + \begin{pmatrix} Time\ taken\ to\ travel \\ length\ of\ the\ flying\ path \end{pmatrix}$$

Therefore,

$$T_F = (n \times T_W) + \frac{Length\ of\ the\ drone\ flying\ path}{S} \quad (3)$$

Where, S is the speed of the drone, T_W is the waiting time of the drone near a one sensor node and n is the number of sensor nodes in the array. Hence by substituting mathematical expressions to length of the drone flying path in the Table 2 to the Equation 3, we have derived equations for the numbers of sensor nodes that can be cover by the drone within its one flying cycle during data collection procedure. Therefore, the following Equation 4, 5, and 6 illustrates

numbers of sensor nodes that can be cover by the drone within its one flying cycle during data collection procedure for above three types of sensor nodes arrays.

For linear sensor nodes array:

$$n_1 = \frac{T_F + \frac{D}{30 \times S}}{T_W + \frac{D}{30 \times S}} \tag{4}$$

For circular sensor nodes array:

$$n_2 = \frac{T_F}{T_W + \frac{D}{30S}} \tag{5}$$

For squared sensor nodes array:

$$n_3 = \left\{ \frac{\frac{-D}{\sqrt{2}} + \sqrt{\frac{D^2}{2} + \left[\; 4 \times (60 \times ST_W + D) \left(D \times \left(1 + \frac{1}{\sqrt{2}} \right) + 60 \times ST_F \right) \; \right]}}{2 \times (60ST_W + D)} \right\}^2 \tag{6}$$

Where, n_1, n_2, n_3 are the number of sensor nodes in linear sensor nodes array, circular sensor nodes array and square sensor nodes array respectively that can be covered during data collection procedure. D is the distance between two sensor nodes.

Then using equations 4, 5 and 6 we implemented the simulation setup in MATLAB simulator to observe the relationship between total flying time of the drone and the numbers of sensor nodes that can be covered by the drone during data collection process.

Table 3 summarizes the parameters we used in the experiments in order to obtain the simulation results. Moreover, we have evaluated how significantly each parameter contributes to decide the number of sensor nodes serviced by the drone [15]. Values of the total flying time (T_F) and battery charging time of the drone (T_C) belong to the actual parameter values of the drone which we used for the prototype implementation.

Table 3. General simulation parameters.

Parameter	Value
Parameter Value Total Flying Time (T_F)	25 minutes [50]
Battery Charging Time of The Drone (T_C)	90 minutes [50]
Data Processing Time of the Drone (T_P)	1 second
Data Download Time from Drone to Node (T_{DL})	10 seconds
Data Generation Rate of the Sensor Node (R_S)	20 B/min
BLE Data Rate (R_{BLE}) between Drone and Node	10.5×104B/min [51]
Speed of the Drone (S)	12 m/s [50]
Distance between two Sensor Nodes (D)	100m

4.2 Prototype Implementation

4.2.1 Equipment Used

For the prototype implementation, we mainly focused on the implementation of the BLE communication between drone and sensor node at the remote field. Therefore, we used four main equipment to setup the remote location prototype such as DJI Phantom 3 SE drone, Raspberry Pi, Waspmote and a BLE module to implement the BLE communication capability in Waspmote.

4.2.1.1 DJI Phantom 3 SE Drone

Figure 20. DJI Phantom 3 SE

The specifications of the DJI Phantom 3 SE drone are depicted in Figure 21

Takeoff Weight	1236 g
Diagonal Distance (propellers excluded)	350 mm
Max Ascent Speed	5 m/s
Max Descent Speed	3 m/s
Hover Accuracy Range	Vertical: ±0.1 m (with Vision Positioning) ±0.5 m (with GPS Positioning) Horizontal: ±0.3 m (with Vision Positioning) ±1.5 m (with GPS Positioning)
Max Speed	16 m/s
Max Service Ceiling Above Sea Level	6000 m
Max Hovering Time	Approx.25 minutes
Operating Temperature Range	32° to 104°F (0° to 40°C)
Satellite Positioning Systems	GPS/GLONASS
Controllable Range	Pitch: -90° to +30°
Stabilization	3-axis (pitch, roll, yaw)

Figure 21. Specifications of DJI Phantom 3 SE [50].

In the prototype implementation, we used a Raspberry Pi coupled with DJI Phantom 3 SE drone. The reason for choosing the Phantom 3 SE was its capability to carry 1236g during its flying cycle.

4.2.1.2 Raspberry Pi

Raspberry Pi is a hackable, inexpensive, small, high performable and education-oriented computer board launched in 2012 [52]. The Raspberry Pi Foundation in the United Kingdom introduced this small computer board to promote teaching of fundamentals in digital making and computing in schools. Raspberry Pi operates similar to a standard PC, requiring a power supply, display unit and a keyboard. It uses an operating system same as other computers. Raspberry Pi uses a Linux operating system named Raspbian as its operating system, since its low cost, open source, and making the Raspberry Pi more hackable. However, the Raspberry Pi is compatible with few non-Linux OS as well [52].

For the prototype implementation in thesis work, we used Raspberry Pi 3 Model B to couple with drone. Hence it proves the data processing and saving capability of the drone, since DJI Phantom 3 SE unable cater data computing abilities. The components of the Raspberry Pi 3 Model B are depicted in Figure 22.

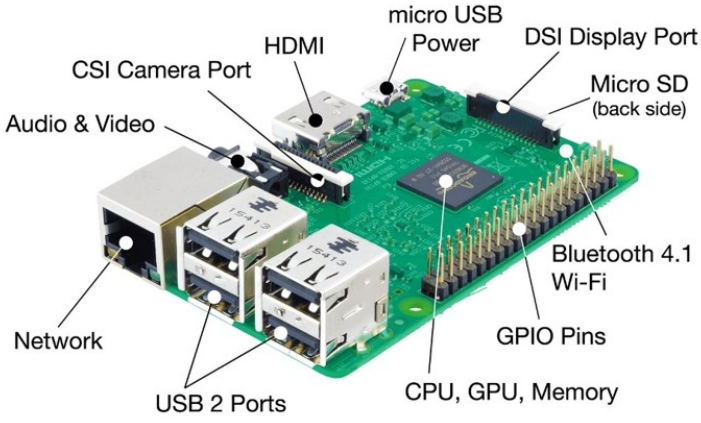

Figure 22. Hardware components of the Raspberry Pi 3 Model B [53].

The Raspberry Pi 3 Model B is the first model of the third generation series of Raspberry Pi. It has been introduced in February 2016 as a replacement to the Raspberry Pi 2 Model B [53]. The specifications of Raspberry Pi are depicted in Figure 23.

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

Figure 23. Specifications of Raspberry Pi model B [53].

4.2.1.3 Waspmote

Waspmote is an open source wireless sensor platform which can be used to implement low power consuming autonomous sensor applications. The unique hardware architecture of the Waspmote has been designed to operate with ultra-low power consumption. Connecting the dedicated module, Waspmote is capable to provide communication interfaces including short-range and long-range radio access technologies. This is battery powered device with 1 to 5 years lifetime depending on application used. Waspmote was officially launched in 2009 by Libelium [54]. Hardware components of the Waspmote are depicted in Figure 24.

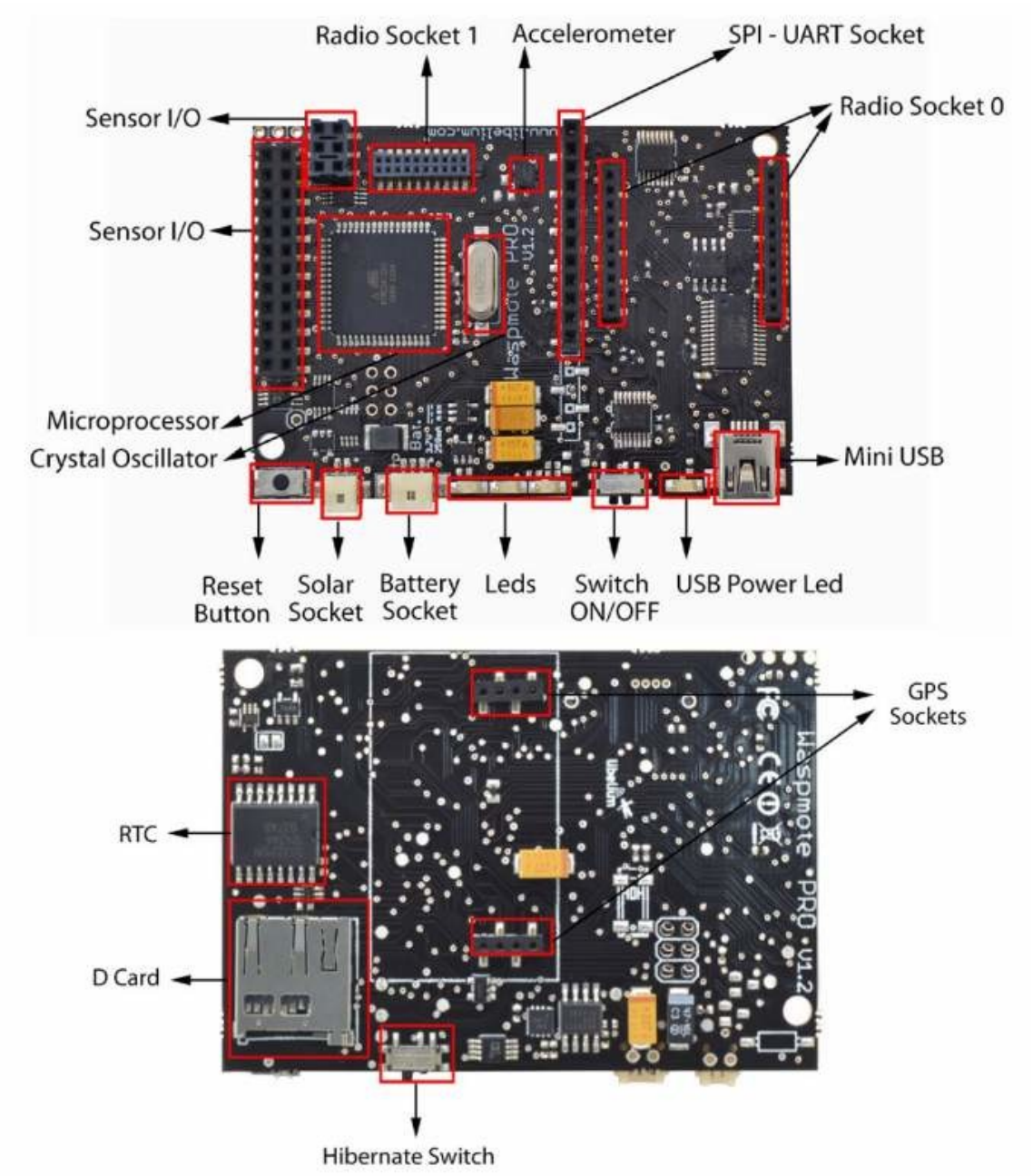

Figure 24. Hardware components of the Waspmote [55].

Waspmote is designed as a modular architecture. Hence, the concept is to accommodate only the needed modules in the motherboard of the device. The modules can be changed or developed according to user application requirements.

Modules available for integration in Waspmote are depicted in Figure 25 and technical specifications are depicted in Figure 26. These information are extracted from the Waspmote technical guide [55].

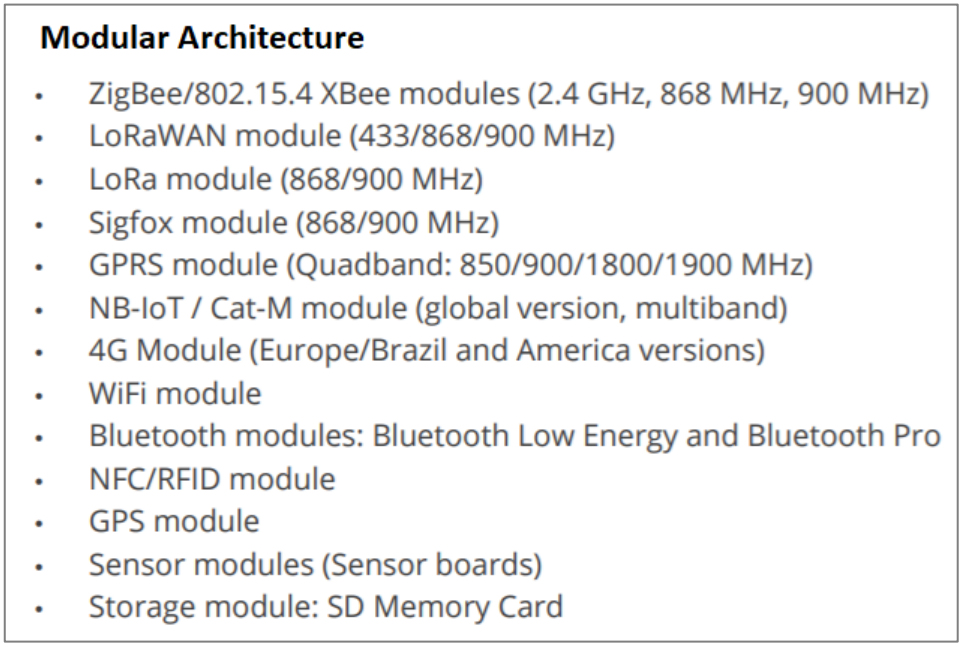

Figure 25. Modules of Waspmote [55].

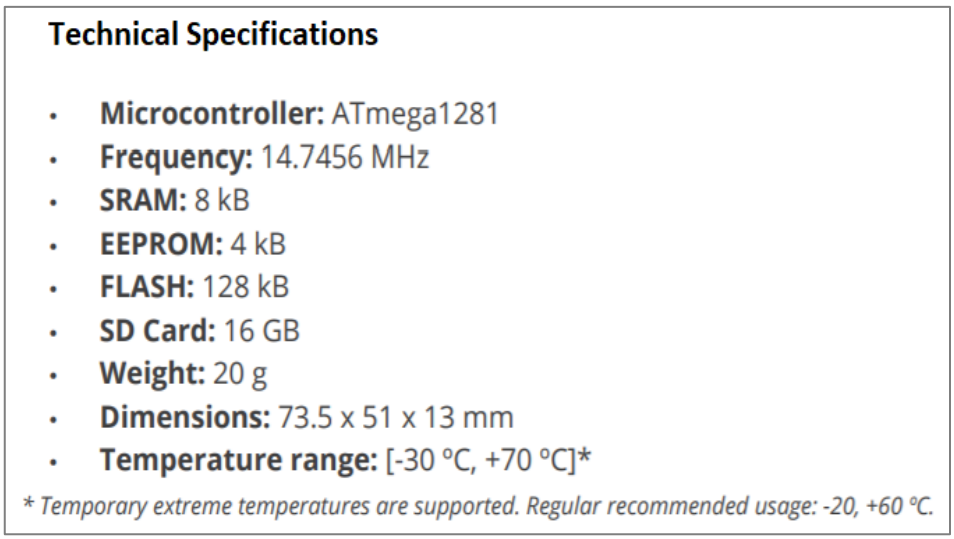

Figure 26. Specifications of the Waspmote [55].

4.2.1.4 BLE Module

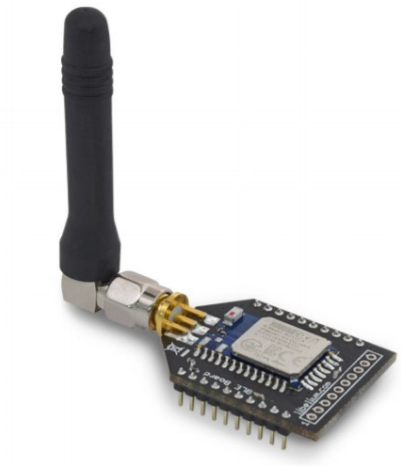

Figure 27. BLE Module of the Waspmote [56].

Figure 27 illustrates the BLE module of the Waspmote. The BLE module is managed by UART and it has to be connected on SOCKET0 or SOCKET1 of the Waspmote in order to establish the BLE communication capability. The main technical specifications of the module are illustrated in Figure 28.

Technical specifications:

- Protocol: Bluetooth v.4.0 / Bluetooth Smart
- Chipset: BLE112
- RX Sensitivity: -103 dBm
- TX Power: [-23 dBm, +3 dBm]
- Antenna: 2 dBi/5 dBi antenna options
- Security: AES-128
- Range: 100 meters (at maximum TX power)
- Consumption: sleep (0.4 uA) / RX (8 mA) / TX (36 mA)
- Send broadcast advertisements (iBeacons)
- Connect to other BLE devices as Master / Slave
- Connect with smartphones and tablets
- Set automatic cycles sleep / transmission
- Calculate distance using RSSI values
- Perfect for indoor location networks (RTLS)
- Scan devices with maximum inquiry time
- Scan devices with maximum number of nodes
- Scan devices looking for a certain user by MAC address

Figure 28. BLE module specifications [55].

The BLE module communicates with Waspmote using a binary protocol via a serial interface. When the BLE module receives a command from Waspmote, the module sends back an acknowledgement to the Waspmote. Events can be produced using these commands. The developer can use Waspmote IDE tool to implement functions to manage these commands, acknowledgments and events in order handle the module easily [56].

4.2.2 Communication between Drone and Sensor Node

To setup the prototype implementation, we used a DJI phantom 3 SE drone, five Waspmotes and five BLE modules, a Raspberry Pi 3 Model B and an external power source to supply power to the Raspberry Pi as core instruments. In the prototype implementation we considered two main scenarios described in the proposed architecture. Therefore, the first task was to implement communication between drone and sensor node at the remote location. Then the second task was to implement the communication between drone and the central cloud server at the home location. Figure 29 illustrates the prototype of the proposed architecture.

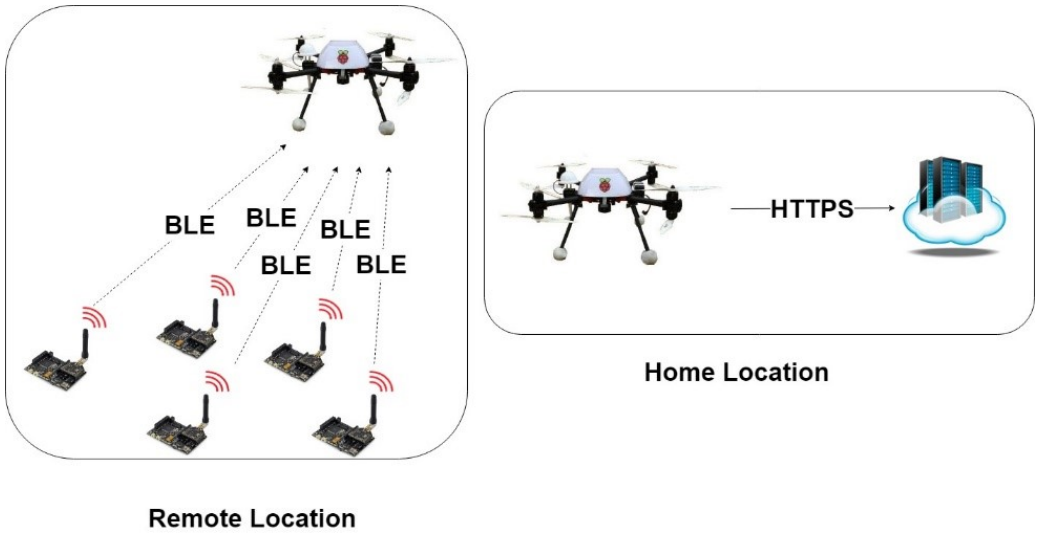

Figure 29. Prototype [15].

In order to implement the communication between drone and sensor node, initially we implemented the sensor network using five Waspmotes. Then we connected BLE modules to the SOCKET0 of the Waspmote. These BLE modules are manufactured by the Libelium to allow the BLE communication capability of the Waspmotes. For the prototype implementation we used in-built accelerometer sensor of the Waspmote to generate the sense data, because accelerometer is the only in-built sensor of the Waspmote. Therefore, this accelerometer data is transferred to the Raspberry Pi over BLE.

The DJI phantom 3 SE drone is capable of carrying 1236g weight while hovering. Therefore, to implement the data collection capability of the drone we coupled a Raspberry Pi with the drone. Generally, Raspberry Pi 3 Model B is powered by a +5.1V micro USB supply and recommended input current is 2.5A [53]. Therefore, to provide these power requirements to the Raspberry Pi, we coupled an external power source (i.e. power bank) to the Raspberry Pi.

To establish the connection between Raspberry Pi and Waspmote's BLE module, the Waspmote considered as master of the network and Raspberry Pi as slave. Hence, Waspmote advertises its availability through advertising attribute to make a connection with a slave device. Then, when the drone comes it acts as a slave of the network and sends connection request to the BLE module of the Waspmote and connects with the Waspmote. Moreover, we programmed Raspberry Pi to scan all the MAC addresses of neighbouring devices when it comes to the range of sensor network. When the corresponding Waspmote's MAC adress is visible, Raspberry Pi establish the connection with that Waspmote in order to start the data

transferring process. To implement the data transferring process from, we used different user attributes called handlers available in BLE user services profile of the Waspmote.

The prototype implementation of the data uploading from Waspmote to Raspberry Pi has been carried out considering two scenarios. Therefore, the first scenario is introduced for the real time data transfer to the drone and second is introduced for the non-real-time data or saved data transferring to the drone. The data downloading process from Raspberry Pi to Waspmote has been implemented to observe the practical downloading data rates of the waspmotes. The implementation of different data transferring processes is described below.

Real time data transferring process from Waspmote to Raspberry Pi:

The real time scenario has implemented to collect the data when the drone is within the range of sensor network. This data transferring process is operated simultaneously with non-real time data transferring process. To implement this scenario in the prototype, we create a customized local attribute in Waspmote using a user attribute from BLE user services profile. Therefore, we used handler 0x0038 to get the real time sensed data from the accelerometer and transfer them to the Raspberry Pi. Handler 0x0038 is capable of creating a reliable BLE communication between Raspberry Pi and Waspmote, because it caters the connected device with read/write and indication operations. When the Raspberry Pi received this real time sensed data from the Waspmote, it saves these data to a different directory of its SQLite [57] database.

Real time data transferring process from Waspmote to Raspberry Pi:

Non-real time scenario has implemented to maintain the continuity of data sensing while drone was away from the sensor network. Hence to implement this scenario in the porotype implementation, we used Waspmote's internal memory to save the sensed data when the drone is away. Waspmote uses micro SD memory card to save its sensed data. Therefore, when the Raspberry Pi terminated the connection, Waspmote starts saving accelerometer sensed data to a text file and saved them in the SD memory. Then when the drone comes these text files has been transferred to the Raspberry Pi over handler 0x0030. Handler 0x0030 also capable of creating a reliable BLE communication between Raspberry Pi and Waspmote, since it caters with read/write and indication operations. Similar to the real time scenario, when the Raspberry Pi received this non-real time sensed data from the Waspmote, it saves these data to a different directory of its SQLite database.

Data downloading process from Raspberry Pi to Waspmote:

This has been implemented to address the data transferring from drone to sensor node when the drone needs to convey commands to the sensor node during data collection procedure. Therefore, same as in real time and non-real-time data transferring process, we used a handler from the BLE user services profile that is 0x0020. Handler 0x0020 allows Raspberry Pi to read/write and indication operations. Since we use different handlers to data uploading, this process can be operated at the same time when drone extracts data from the Waspmote.

When the raspberry Pi finishes its data retrieving, it terminates the connection with Waspmote. Then drone comes to the docking station, Raspberry Pi connects 802.11 WLAN router and transfers those saved data from the SQLite database to central cloud server database over HTTPS protocol. Here we used Google Firebase [58] to save the extracted data from the Raspberry Pi.

The prototype setup with equipment is illustrated in Figure 30. In this setup we powered the Raspberry Pi with Samsung [59] external power source. Waspmotes can be powered by rechargeable Lithium ion batteries.

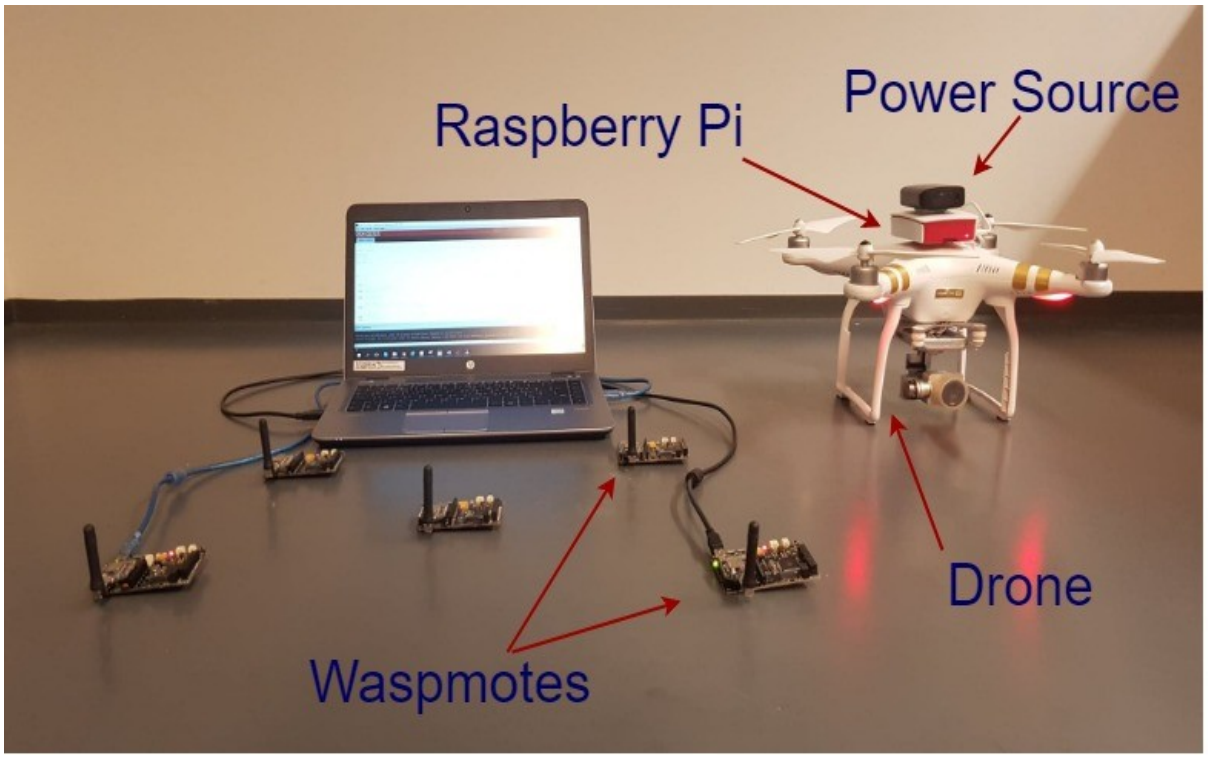

Figure 30. Prototype Implementation [15].

4.2.3 Cloud Integration

The final step of the prototype implementation was to implement the cloud storage server at the home location. As the drone arrives to the home location after extracting data from the sensors in WSN, Raspberry Pi 3 Model B coupled with the drone connects to the 802.11 WLAN router with internet access to upload the saved data.

The central cloud server is implemented on the Google Firebase where Raspberry Pi first authenticates itself using its private key. After authentication, Raspberry Pi initiate the data uploading process to the central cloud server and cloud server stores those data into a database. Data has been stored as JSON [60] format and those data is synchronized with each client at the same time.

Firebase utilizes HTTPS connectivity through Transport Layer Security, therefore it allows secure communication between central cloud server and Raspberry Pi as well as it caters database with real-time network security. The database provides a flexible, expression-based rules language, including information relates to required data structures and data accessible timelines. It is capable of defining data privacy, authorization and data access procedures. This procedure ensures the data privacy of the use case.

After successful transferring of the data, Raspberry Pi deletes the stored data from the local database [15].

5 RESULTS ANALYSIS

5.1 Simulation Results Analysis

In this section, we analyse the results of the implemented MATLAB simulation. Therefore, the analysis categorised into four sections in order to observe the impact of different parameters on simulation outcome. The simulation has carried out to examine the relationship between total flying time of the drone and the numbers of sensor nodes that can be covered by the drone during data collection procedure. Considering the drone, we used for the prototype implementation, here we examined the variation of number of sensor nodes during 25 minutes flying cycle. The other general parameters that used to get the simulation results are depicted in Table 3, Chapter 4. The following key parameters have changed to analyse the simulation results:

- Impact of BLE data rate (R_{BLE})
- Impact of sensor data rate (R_S)
- Impact of drone's speed (S)
- Impact of distance between sensors (D)

5.1.1 Impact of BLE data rate

Since we used BLE communication to implement the connectivity between drone and sensor node, the impact of BLE data rate has to be examine. The simulation graphs correspond to different sensor nodes array types are shown in the Figure 31. When we consider the practical scenario of data collection procedure, there is a significant impact from BLE data rate to the waiting time of the drone near a sensor node. If the BLE data rate between drone and sensor node is low, drone has to wait longer time at a particular sensor node. Therefore, the overall number of sensor nodes that can be covered during one flying cycle has to be decrease. Hence, the higher BLE data rates between drone and sensor node will lead to collect data from more sensor nodes during one flying cycle.

In order to observe the impact of BLE data rate in total number of sensor nodes that can be covered by the drone during one flying cycle, we changed BLE data rate (R_{BLE}) as 2 kbps, 4 kbps, 8 kbps, 12 kbps, 16 kbps while keeping the other parameters fixed. The simulation has been carried out for each sensor node array type that has considered for the implementation of simulation work. According to the plots illustrated in Figure 31, with higher BLE data rates drone can collect data from more sensor nodes than with lower BLE data rates. However, when BLE data rates are higher than 8kbps, the curves of Figure 31 tend to coincide. This means that the total number of sensor nodes that can be covered is approximately similar when the data rate is higher than 8 kbps. Moreover, when BLE data rates get higher values, the curves become linear. These observations are valid for all three sensor nodes arrays.

When we compare the graphs correspond to different sensor nodes arrays, circular and square arrays yield better performance than the linear array of sensor nodes. Therefore, the maximum number of sensor nodes that can be covered by a drone which has 25 minutes flying time are approximately 75 with 16kbps BLE data rate.

Thus, using this simulation results, we can conclude that the drone can collect data from more sensors, when the BLE data rate between drone and sensor node is high [15].

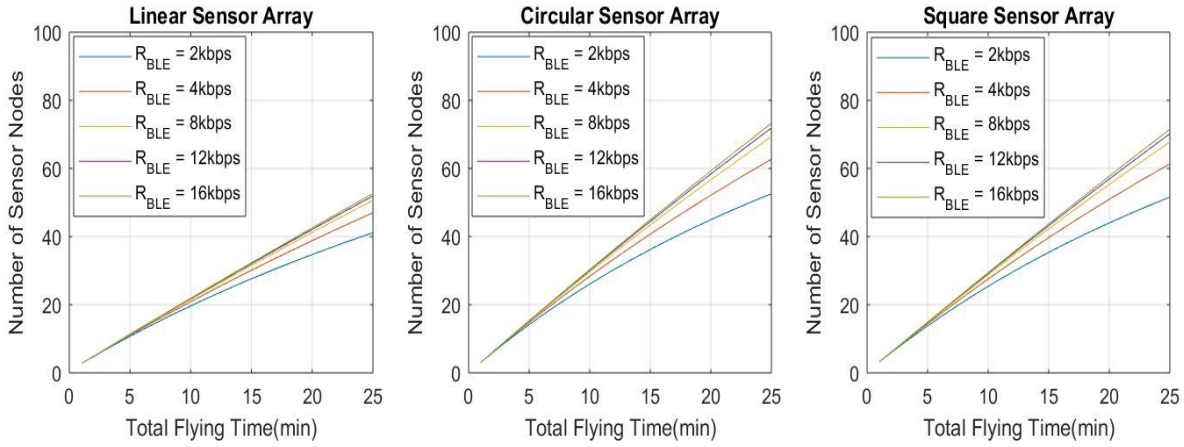

Figure 31. Impact of BLE data rate.

5.1.2 Impact of sensor data rate

The data generation rate of the sensor node has a substantial impact on the overall performance of the system. Because with lower sensor node data rate, sensor produces less amount of data. Therefore, when the drone comes to the sensor node, it can quickly collect those less amounts of data than collecting higher amount of data. Thus, the lower sensor node data rate drone capable to retrieve data from more sensor nodes than having higher sensor node data rates. The impact of sensor data rate provides contradictory observations compared to the observations we had in BLE data rate.

Figure 32 illustrates the simulation results for this experiment. Here we changed sensor nodes data rates (R_S) as 20 Bytes/min, 40 Bytes/min, 60 Bytes/min, 80 Bytes/min, and 100 Bytes/min while keeping other parameters fixed. Moreover, we assumed one in-built sensor in a sensor node generates data with the rate of 20 Bytes/min. The maximum number of sensor nodes that can be covered by a drone which has 25 minutes flying time are approximately 75 with sensor data rate of 20 Bytes/min.

Similar to the previous case, circular and square arrays provided better performance than linear sensor nodes array. Thus, using this simulation results, we can conclude that the drone can collect data from more sensors, when the sensor data rate is low [15].

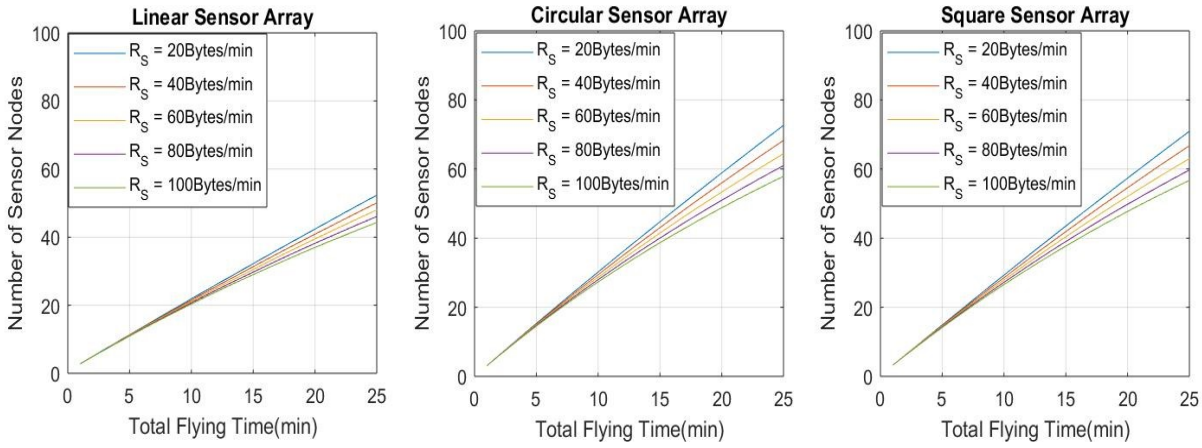

Figure 32. Impact of sensor data rate.

5.1.3 Impact of drone's speed

For the prototype implementation, we used DJI Phantom 3 SE drone which has hovering speed of 16m/s. Hence for the simulation experiments, we used 12 m/s as average speed considering the practical time taken to accelerate and decelerate. The simulation results correspond to impact of drone's speed (*S*) on proposed system is depicted in Figure 33. The overall performance of the proposed system can be increased by using a high-speed drone. This means when the drone has a higher speed, it can travel quickly from one sensor node to another.

To validate these phenomena, we carried out the simulation experiments from lower drone speeds to higher speeds. Therefore, we varied speed of the drone as we carried out our simulation experiments from lower drone speeds to higher speeds. According to Figure 33, higher speed drones speed is capable to collect data from more sensor nodes than drones with lower speeds. However, when drone's speed gets higher values, the gap between curves become decrease. With compared to other two scenarios, the plots of the simulation outputs have become linear.

In this case also, circular and square array provided better performance compared to linear array.

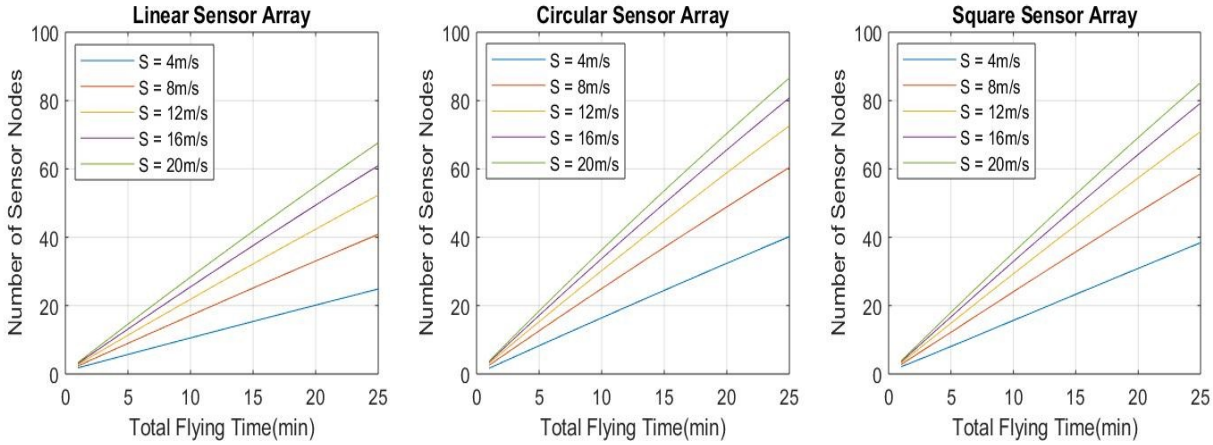

Figure 33. Impact of drone's speed.

5.1.4 Impact of distance between sensors

The distance between two sensor nodes has a significant impact to the proposed system architecture. In the practical scenario, if the sensor array consists with adjacent sensor nodes, the drone can reach to more sensor nodes. Therefore, to experiment the validity of this phenomena, we changed the distance between two sensor nodes (*D*) as 50 m, 100 m, 150 m, 200 m, 250 m in the simulation experiments. Similar to the previous three cases, we kept other parameters fixed as in Table 3, Chapter 4. The simulation results related to this scenario has depicted in Figure 34.

According to the simulation outcomes, when the distance between two sensors are low, the number of sensor nodes that can be covered during data collection procedure becomes increase. Moreover, when the distance between two sensor nodes gets higher values, the gap between curves become decrease the plots of the simulation outputs has become linear.

When we compare the graphs correspond to different sensor nodes arrays, circular and square arrays yield better performance than the linear array of sensor nodes.

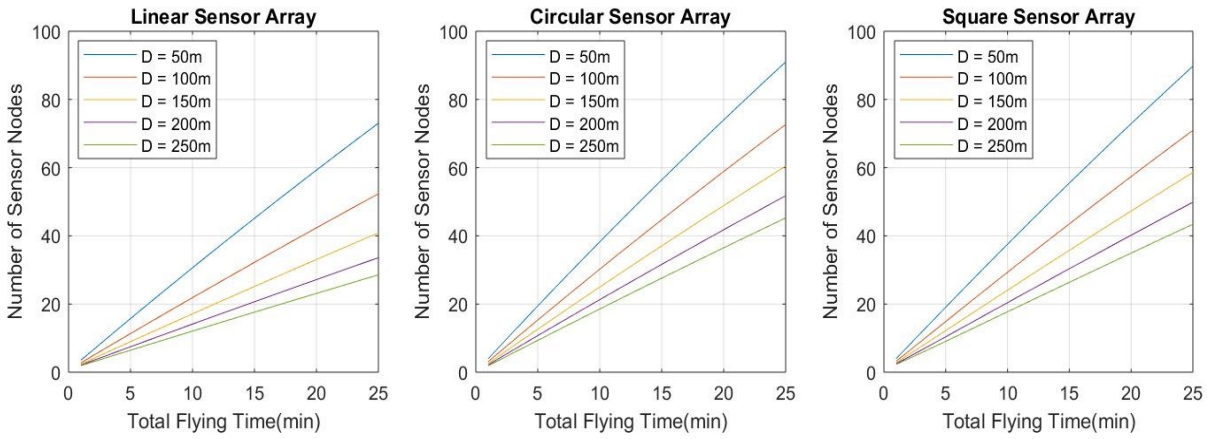

Figure 34. Impact of distance between sensors.

5.2 Experimental Results

In this section we describe the experimental results that we extracted from the prototype implementation. Using prototype implementation, we realized actual uploading and downloading data rates between Raspberry Pi and Waspmote. In order to realize the experimental parameters, we carried out 20 repetitive tests for each phenomenon. Also, to obtain the performance results, we fixed the drone's flying altitude as 10 m above the ground level.

In the data uploading process we programmed handler 0x0038 from BLE user services profile of the Waspmote to send the accelerometer sensor data from Waspmote to the Raspberry Pi. Maximum data size that can be transferred during a particular transmission attempt is 20 Bytes. Moreover, Waspmote sends those accelerometer sensor data in an encoded format which has to be decoded at Raspberry Pi to make it readable. After decoding these data at the Raspberry Pi, it extracts to 112 Bytes. Hence, we consider this 112 Bytes as maximum data that can be send within one particular transmission.

In the data download process (i. e. the transmission from Raspberry Pi to Waspmote) we programmed handler 0x0020. Here we considered 10 seconds time duration to send the data packets over BLE from Raspberry Pi to Waspmote. Because with the buffer size limitation in Waspmote, Raspberry Pi is only allowed to send maximum 74 data packets (each 20 Bytes) to the Waspmote and it takes 10 seconds to transfer these 74 data packets. This maximum data transmission causes to fill the buffer size at Waspmote and then it stops retrieving data from Raspberry Pi.

The following Table 4 summarizes the actual data rates obtained from the experiments.

Table 4. Experimental Data Rates

Data Rate Parameter	Value
Data transfer rate from Waspmote to Raspberry Pi (R_{UL})	672 Bytes/min
Data transfer rate from Raspberry Pi to Waspmote (R_{DL})	148 Bytes/s

According to the results depicted in Table 4, Raspberry Pi to Waspmote data download rate is lower than the Waspmote to Raspberry Pi data upload rate. The reason for lower download data rate was the software limitations in Waspmote.

6 DISCUSSION

The first section of this chapter critically examines and compares the thesis work with other works in the same research field. Therefore, we have considered a similar research works which are already exist in the research field of remote monitoring in WSN and compared them with this thesis work and analyse the novel contribution of thesis work. Next, we examine the final outcomes of this thesis work with its main objectives. Finally, this chapter focus on the future research directions and possible approaches that have appeared during the thesis work.

6.1 Comparison with Similar Work

In this thesis work, we have introduced a novel edge based, remote monitoring architecture for low power wireless sensor networks using UAVs. Recently, most of the wireless sensor networks have been steered towards low power IoT sensor networks, however to the best of our knowledge there is no sufficient work carried out to collaborate the UAVs with remote low power WSN monitoring. This thesis work has been carried out as an extended contribution to the work presented in [10]. The research work in [10], a mobile-based relay assistance architecture which creates an end-to-end communication between low power IoT devices and central cloud servers with the absence of devoted local gateway [10]. Therefore, from this thesis work we introduce UAV in order to cater the connectivity between remote sensor network and central cloud server. Therefore, by introducing UAVs to the proposed mobile relay architecture in [10] will enhance the mobility of the entire proposed system.

The possibility of crowd surveillance using UAVs have discussed in [37], the article presents UAV assists crowd surveillance system considering face recognition of the people. The proposed architecture in [37], has considered WiFi, 4G/5G technologies to communicate between drone and sensors. Since these technologies consume more power and still limited to urban areas, the use cases are limited. Therefore, in this thesis work, we proposed system model that uses low power concept by proposing BLE to the communication between drone and sensor node. Moreover, the remote location of the proposed architecture can locate anywhere without internet accessibility.

The thesis work has contemplated a smart agriculture use case to analyse the practical viability of the proposed architecture. The proposed IoT platform for data-driven agriculture as FarmBeats [61] has used TV White Spaces (TVWS) link to access the internet connection of farmer's house via IoT base stations on the farm site and then the IoT base station provides a Wi-Fi interface for connections from sensors in order to connect with drone. Therefore, the solution in [61] is high power consumable and only applicable to the areas with internet accessibility. In our thesis work we presented a solution for remote agriculture field monitoring applications where the remote location is unable to cater internet accessibility to the sensor nodes in the network. Therefore, the proposed system modal provides more reliable solution than the presented work in [61]. The edge-based architecture of the proposed system model in the thesis provides secure platform for data storage in the central cloud server than the work presented in [61].

Therefore, considering all these scenarios, we can conclude that this thesis work has provided significant contribution to the applications in remote monitoring applications in low power IoT sensor networks.

6.2 Evaluation on Meeting the Thesis Objectives

The objective of this thesis work is to create an edge based secure communication architecture using UAVs and low power IoT devices which is capable of monitoring a remote wireless sensor network, perform a set of actions based on the monitored data. Therefore, the scope of the thesis work is confined to defining a system model, protype implementation of the system architecture and simulation implementation to analyse the performance of the system.

In the prototype implementation, we considered two main operations described in the proposed architecture. Therefore, the first task was to implement communication between drone and sensor node at the remote location. Then the second task was to implement the communication between drone and the central cloud server at the home location. Since we proposed this system model to a remote site with internet inaccessibility, we introduced BLE communication to connect the sensor node with drone. Therefore, the sensor node is capable to transfer the data with low power consumption since BLE is well known low power radio access technology. To implement the data transferring operation at the remote location we used Raspberry Pi coupled with DJI Phantom 3 SE drone and a BLE module connected Waspmote sensor. We were able to implement a reliable BLE communication between Raspberry Pi coupled with drone and Waspmote sensor node. However due to hardware and software limitations of the Waspmote the experimental data rates were low. We may eliminate this problem if we use a sensor node that dedicated to BLE communication and consist with considerable internal memory. As the second task of the prototype implementation we implemented the communication between drone and central cloud. The data transfer rates between Raspberry Pi and cloud were not tested, since we are planning to implement end-to-end fully working prototype of the proposed architecture in the future.

In the simulation implementation, we considered three types of sensor nodes arrays that can be implement in practical environment setup. Hence, we considered linearly deployed sensor nodes array, circularly deployed sensor nodes array and a squared sensor nodes array to observe the relationship between the numbers of sensor nodes that can be cover by the drone within its one flying cycle during data collection procedure. the simulation results were same as we observed in the practical setup. Therefore, the number of sensor nodes that can be covered by the drone during one flying cycle (25 minutes) is directly proportional to the speed of the drone. The distance between two sensor nodes is the most dominant parameter that defines the overall performance of the system. Circular and square array type sensor node topologies always outperform the linear array topology.

6.3 Future Research Work

In the near future, the advancement of low power IoT technologies and their efficient sensing and monitoring applications may lead the processing at the edge of the network, instead of accessing the remote central cloud. For the prototype implementation, we used DJI Phantom 3 SE drone as the data collecting UAV in the proposed architecture. Since this drone unable to program to according to the prototype implementation requirements we had to couple a Raspberry Pi with the drone to do the data transfer operations. Therefore, in future we can introduce programable drones to the prototype implementation. The experimental results are taken considering ideal situation assuming wind speed zero. Therefore, the research can be extended to evaluate the performance of the system modal at the practical environment, considering wind speed impact on drone speed. Moreover, we can use drones with higher battery life to increase the hovering time.

For the implementation of the WSN we used Waspmote sensors, hence the BLE data rate was low. As enhancement to the prototype implementation we may use high rate BLE communication sensors to implement the low power WSN. Moreover, in the prototype implementation we used only one scalar sensor to get the experimental results. Therefore, we can examine the performance of the proposed system by transferring data from multiple scalar sensors or multimedia sensors in one sensor node to the drone.

We intend to extend the research by implementing a fully working prototype and measuring energy consumption in an actual WSN where the sensors are deployed in a higher density. As advancement to this research work, the proposed edge-based remote monitoring architecture can be extended to dew computing-based architecture since dew computing is an emerging topic in modern computing research areas.

7 SUMMARY

This thesis work introduces novel edge-based remote monitoring architecture for low power IoT sensors in a WSN with the help of UAVs. Therefore, this proposed architecture creates secure end-to-end communication between diversified WSN and central cloud via UAV, which assists the data retrieving, data processing and management tasks in a remote site. The main objective of this thesis work is to introduce an energy efficient, cost effective, labour saving and systematic architecture for remote monitoring in a heterogeneous WSN. By introducing UAVs or drones to the remote monitoring architecture we eliminate the human labour from data collection procedure. To describe the potential viability of the proposed architecture we contemplated a smart agriculture use case. The same architecture can be used in many other use cases such as disaster detection, wastewater management, environmental monitoring, anomaly detection in sensor networks, mobile crowd sensing applications, etc.

The system model operations can be categorised into two main segments; the data transferring process between drone and sensor node at the remote location and the data transferring process between drone and central cloud server at the home location. To establish the connection between drone and sensor node we proposed BLE communication, since BLE is a novel low power radio access technology. To set up the connectivity between drone and central cloud at the home location we used HTTPS protocol. The system model introduces three types of links that can be programmed according to user requirements or according to the nature of the use case. Using the first link, sensors can send data to the cloud server without and processing inside the drone. The second link can be used to the application where drone has to activate the actuators of the drone. Therefore, in the second link drone collects data, process those data inside the drone and take actions according to processing outcome. The third link ensures the data privacy of the data transmission from sensor network to central cloud. There are three main communication protocols have used in the proposed architecture including two BLE communication protocols and a cloud integration protocol. The two BLE protocols have introduced to implement the communication between drone and sensor node at the remote location. Therefore, these BLE protocols provide data downloading and uploading process between drone and sensor node. When the drone comes to the home location after extracting data from the sensor nodes, it uses HTTPS protocol to transfer the data to cloud server.

The experiment set up of the thesis work carried out under two tasks. Therefore, a prototype implementation and a simulation work has presented in the thesis. We used four main equipment to setup the remote location prototype such as DJI Phantom 3 SE drone, Raspberry Pi, Waspmote and a BLE module to implement the BLE communication capability in Waspmote. The Raspberry Pi coupled with the drone collects data from Waspmote sensor nodes and saves them into a SQLite database. The when the drone comes to the home location it connects 802.11 WLAN router with internet access to upload the saved data to the central cloud. The central cloud server is implemented on the Google Firebase. Authentication of the Raspberry Pi is required before uploading data to the central cloud.

The simulation implemented to examine the relationship between total flying time of the drone and the numbers of sensor nodes that can be covered by the drone during data collection procedure. Three sensor nodes arrays types are considered in the simulation; linearly deployed sensor nodes array, circularly deployed sensor nodes array and a squared sensor nodes array. The key parameters changed to analyse the simulation results are BLE data rate, sensor data rate, drone's speed and distance between two sensor nodes. According to the simulation results, the number of sensor nodes that can be covered by the drone during one flying cycle is directly proportional to the speed of the drone, the distance between two sensor nodes is the most

dominant parameter that defines the overall performance of the system, Circular and square array type sensor node topologies always outperform the linear array topology.

As future work, we hope to extend this thesis work by implementing fully working prototype which ensures the end-to-end connectivity between WSN and central cloud server and measuring the overall energy consumption in an actual WSN where the sensors are deployed in a higher density. Also, we plan to enhance the performance of the prototype implementation by using programmable equipment with latest technology. Since current experimental results are taken considering ideal environment. Therefore, in the future, we plan to investigate how the drone speed and the wind speed will impact on the proposed system model. Since this proposed architecture based on edge-computing paradigm, the enhanced research on this proposed architecture may lead to merge dew computing paradigm to this thesis work.

8 REFERENCES

[1] Lee I, Lee K. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. Business Horizons. 2015 Jul 1;58(4):431-40.

[2] Doukas C, Maglogiannis I. Bringing IoT and cloud computing towards pervasive healthcare. In2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing 2012 Jul 4 (pp. 922-926). IEEE.

[3] Jeong S, Simeone O, Kang J. Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning. IEEE Transactions on Vehicular Technology. 2018 Mar;67(3):2049-63.

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347–2376, 2015.

[5] O. Bello and S. Zeadally, "Intelligent Device-to-Device Communication in the Internet of Things," IEEE Systems Journal, vol. 10, no. 3, pp.1172–1182, 2016.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and Challenges," IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, 2016.

[7] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on Multi-Access Edge Computing for Internet of Things Realization," IEEE Communications Surveys Tutorials, 2018.

[8] N. Xia, H.-H. Chen, and C.-S. Yang, "Radio Resource Management in Machine-to-Machine Communications-A Survey," IEEE Communications Surveys & Tutorials, 2017.

[9] Cho K, Park W, Hong M, Park G, Cho W, Seo J, Han K. Analysis of latency performance of Bluetooth low energy (BLE) networks. Sensors. 2015;15(1):59-78.

[10] P. Porambage, A. Manzoor, M. Liyanage, A. Gurtov, and M. Ylianttila, "Managing Mobile Relays for Secure E2E Connectivity of Low-Power IoT Devices," in IEEE Consumer Communications & Networking Conference, 2019.

[11] A. Manzoor, P. Porambage, M. Liyanage, M. Ylianttila, and A. Gurtov, "Mobile Relay Architecture for Low-Power IoT Devices," in IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM), 2018, pp. 14–16.

[12] Zeng Y, Zhang R, Lim TJ. Wireless communications with unmanned aerial vehicles: Opportunities and challenges. IEEE Communications Magazine. 2016 May;54(5):36-42.

[13] Mackensen E, Lai M, Wendt TM. Bluetooth Low Energy (BLE) based wireless sensors. InSENSORS, 2012 IEEE 2012 Oct 28 (pp. 1-4). IEEE.

[14] “How HTTPS Works” [Online]. Availabe: https://strongarm.io/blog/how-https-works/

[15] A. Rajakaruna, A. Manzoor, P. Porambage, M. Liyanage, M. Ylianttila and A. Gurtov,” Enabling End-to-End Secure Connectivity for Low-Power IoT Devices with UAVs”, to be appeared in 2nd Workshop on Intelligent Computing and Caching at the Network Edge IEEE Wireless Communications and Networking Conference (WCNC), Marrakech, Morocco, April 2019.

[16] Heydon R. Bluetooth low energy: the developer's handbook. Upper Saddle River, NJ: Prentice Hall; 2013.

[17] Gupta NK. Inside Bluetooth low energy. Artech house; 2016 Jun 30.

[18] Xia N, Chen HH, Yang CS. Radio resource management in machine-to-machine communications—A survey. IEEE Communications Surveys & Tutorials. 2018 Jan 1;20(1):791-828.

[19] Siekkinen M, Hiienkari M, Nurminen JK, Nieminen J. How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4. In2012 IEEE wireless communications and networking conference workshops (WCNCW) 2012 Apr 1 (pp. 232-237). IEEE.

[20] “Designing for Bluetooth Low Energy Applications.” [Online]. Available: https://www.silabs.com/documents/referenced/white-papers/designing-for-bluetooth-low-energy-applications.pdf

[21] Amaro, J & Patrão, Sérgio. (2015). A Framework for test and verification of inertial sensors and positioning algorithms.

[22] OpenFog Consortium Architecture Working Group. OpenFog Reference Architecture for Fog Computing, Feb. 2017.

[23] M. Satyanarayanan, "The Case for VM-Based Cloudlets in Mobile Computing", IEEE Pervasive Computing, vol. 8, no. 4, pp. 14-23, 2009.

[24] Chun BG, Ihm S, Maniatis P, Naik M, Patti A. Clonecloud: elastic execution between mobile device and cloud. InProceedings of the sixth conference on Computer systems 2011 Apr 10 (pp. 301-314). ACM.

[25] Shi, Weisong, and Schahram Dustdar. "The promise of edge computing." Computer 49.5 (2016): 78-81.

[26] Yu, W., Liang, F., He, X., Hatcher, W. G., Lu, C., Lin, J., & Yang, X. (2018). A survey on the edge computing for the Internet of Things. IEEE access, 6, 6900-6919.

[27] Sabella, Dario, Alessandro Vaillant, Pekka Kuure, Uwe Rauschenbach, and Fabio Giust. "Mobile-edge computing architecture: The role of MEC in the Internet of Things." IEEE Consumer Electronics Magazine 5, no. 4 (2016): 84-91.

[28] ETSI, MECISG. "Mobile edge computing (mec); framework and reference architecture." ETSI, DGS MEC 3 (2016).

[29] He, X., Jin, R., & Dai, H. (2018). Deep PDS-Learning for Privacy-Aware Offloading in MEC-Enabled IoT. IEEE Internet of Things Journal.

[30] Y. Wang, “Cloud-Dew Architecture,” International Journal of Cloud Computing, vol. 4, no. 3, pp. 199–210, 2015.

[31] Y. Wang, K. Skala, A. Rindos, M. Gusev, S. Yang, and Y. PAN, “Dew Computing and Transition of Internet Computing Paradigms,” ZTE COMMUNICATIONS, vol. 15, no. 4, 2017.

[32] Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E. Wireless sensor networks: a survey. Computer networks. 2002 Mar 15;38(4):393-422.

[33] Al-Karaki, Jamal N., and Ahmed E. Kamal. "Routing techniques in wireless sensor networks: a survey." IEEE wireless communications 11.6 (2004): 6-28.

[34] Yick, Jennifer, Biswanath Mukherjee, and Dipak Ghosal. "Wireless sensor network survey." Computer networks 52.12 (2008): 2292-2330.

[35] Everaerts, Jurgen. "The use of unmanned aerial vehicles (UAVs) for remote sensing and mapping." The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 37.2008 (2008): 1187-1192.

[36] Chao, HaiYang, YongCan Cao, and YangQuan Chen. "Autopilots for small unmanned aerial vehicles: a survey." International Journal of Control, Automation and Systems 8.1 (2010): 36-44.

[37] Mozaffari, Mohammad, et al. "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems." IEEE Communications Surveys & Tutorials (2019).

[38] Motlagh, Naser Hossein, Miloud Bagaa, and Tarik Taleb. "UAV-based IoT platform: A crowd surveillance use case." IEEE Communications Magazine 55.2 (2017): 128-134.

[39] Mozaffari, Mohammad, et al. "Mobile unmanned aerial vehicles (UAVs) for energy-efficient internet of things communications." IEEE Transactions on Wireless Communications 16.11 (2017): 7574-7589.

[40] Huang, Yanbo, et al. "Development of a spray system for an unmanned aerial vehicle platform." Applied Engineering in Agriculture 25.6 (2009): 803-809.

[41] Aslan, Yunus Emre, Ibrahim Korpeoglu, and Özgür Ulusoy. "A framework for use of wireless sensor networks in forest fire detection and monitoring." Computers, Environment and Urban Systems 36.6 (2012): 614-625.

[42] Chaczko, Zenon, and Fady Ahmad. "Wireless sensor network based system for fire endangered areas." Third International Conference on Information Technology and Applications (ICITA'05). Vol. 2. IEEE, 2005.

[43] Hughes D, Ueyama J, Mendiondo E, Matthys N, Horré W, Michiels S, Huygens C, Joosen W, Man KL, Guan SU. A middleware platform to support river monitoring using wireless sensor networks. Journal of the Brazilian Computer Society. 2011 Jun 1;17(2):85-102.

[44] Minhas UI, Naqvi IH, Qaisar S, Ali K, Shahid S, Aslam MA. A WSN for monitoring and event reporting in underground mine environments. IEEE Systems Journal. 2018 Mar;12(1):485-96.

[45] Minhas UI, Naqvi IH, Qaisar S, Ali K, Shahid S, Aslam MA. A WSN for monitoring and event reporting in underground mine environments. IEEE Systems Journal. 2017 Jan 20;12(1):485-96.

[46] Moud HI, Gheisari M. Coupling wireless sensor networks and unmanned aerial vehicles in bridge health monitoring systems. InProc. of the 33rd International Symposium on Automation and Robotics in Construction, Auburn, AL, USA 2016 Jul 18 (pp. 267-273).

[47] Hadjidj A, Souil M, Bouabdallah A, Challal Y, Owen H. Wireless sensor networks for rehabilitation applications: Challenges and opportunities. Journal of Network and Computer Applications. 2013 Jan 1;36(1):1-5.

[48] Ullah S, Higgins H, Braem B, Latre B, Blondia C, Moerman I, Saleem S, Rahman Z, Kwak KS. A comprehensive survey of wireless body area networks. Journal of medical systems. 2012 Jun 1;36(3):1065-94.

[49] Ahmed, Khandakar, and Mark A. Gregory. "Integrating Wireless Sensor Networks with Cloud Computing." In MSN, pp. 364-366. 2011.

[50] "DJI Specs." [Online]. Available: https://www.dji.com/phantom-3-se/info?lang=cn#specs

[51] "Practical BLE Throughput." [Online]. Available: https://rigado.zendesk. com/hc/en-us/articles/226824267-Practical-BLE-Throughput

[52] Maksimović, Mirjana, Vladimir Vujović, Nikola Davidović, Vladimir Milošević, and Branko Perišić. "Raspberry Pi as Internet of things hardware: performances and constraints." design issues 3, no. 8 (2014).

[53] "Raspberry Pi Power." [Online]. Available: https://www.raspberrypi. org/documentation/hardware/raspberrypi/power/README.md

[54] "Waspmote Inroduction." [Online]. Available: https://www.cooking-hacks.com/documentation/tutorials/Waspmote

[55] “Waspmote Technical Guide” [Online]. Available: http://www.libelium.com/development/Waspmote/documentation/?cat=general

[56] “Bluetooth Low Energy Networking Guide” [Online]. Available: http://www.libelium.com/development/Waspmote/documentation/bluetooth-low-energy-networking-guide/

[57] “SQLite Database .” [Online]. Available: https://www.sqlite.org/about.html

[58] “Google Firebase” [Online]. Available: https://firebase.google.com/

[59] “Samsung External Power Bank” [Online]. Availabe: https://www.samsung.com/us/mobile/mobile-accessories/phones/10000-mah-portable-battery-with-micro-usb-cable-silver-eb-p1100bsegus/

[60] “JSON” [Online]. Availabe: https://www.json.org/

[61] Vasisht D, Kapetanovic Z, Won J, Jin X, Chandra R, Sinha S, Kapoor A, Sudarshan M, Stratman S. Farmbeats: An iot platform for data-driven agriculture. In14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17) 2017 (pp. 515-529).

9 APPENDICES

Appendix 1 Code used in Waspmote: Transfer accelerometer data to Raspberry Pi

Appendix 2 Python code used in Raspberry Pi: Retrieve accelerometer data from Waspmote

Appendix 3 Code used in Raspberry Pi: Send data to Waspmote

Appendix 4 Code used in Waspmote: Retrieve data from Raspberry Pi

Appendix 5 MATLAB code of the simulation

Appendix 1 Code used in Waspmote: Transfer accelerometer data to Raspberry Pi

```
#include <WaspBLE.h>
uint8_t flag = 0;

uint8_t status;
int x_acc;
int y_acc;
int z_acc;
char test[32];

void setup()
{
  ACC.ON();
  // 0. Turn BLE module ON
  BLE.ON(SOCKET0);
  BLE.setTXPower(8);
  // 1. Make Waspmote visible to other BLE modules
  BLE.setDiscoverableMode(BLE_GAP_GENERAL_DISCOVERABLE);
  // 2. Make Waspmote connectable to any other BLE device
  BLE.setConnectableMode(BLE_GAP_UNDIRECTED_CONNECTABLE);
  USB.println(F("Waiting for incoming connections..."));
}

void loop()
{
  status = ACC.check();
  itoa(ACC.getX(), test,10);
  USB.print(test);
  USB.print(F("\n"));
  // 4.4.1 Write the local attribute (friendly name)
  flag = BLE.writeLocalAttribute(0x0038, BLE_INDICATE_ENABLED, test);

  if (flag == 0)
  {
    USB.println("Attribute changed");
  }
  else
  {
    USB.printf("Error writing. flag = %u\r\n", flag);
  }
}
```

Appendix 2 Python code used in Raspberry Pi: Retrieve accelerometer data from Waspmote

```
import pexpect
import time
import sys
import firebase_admin as fb
from firebase_admin import credentials,db
from datetime import datetime

cred = credentials.Certificate("/home/pi/ble-drone-firebase-adminsdk
    -rxato-2d3b8cae12.json")
fb.initialize_app(cred, {
    'databaseURL' : 'https://ble-drone.firebaseio.com'})
DEVICE = "00:07:80:05:90:CC " # address of your device
#"00:07:80:05:42:F3"
# Run gatttool interactively.
child = pexpect.spawn("gatttool -I")

root = db.reference()
# Connect to the device.
print("Connecting to:"),
print(DEVICE)
NOF_REMAINING_RETRY = 3

start = time.time()
print(start)

while True:
  try:
    child.sendline("connect {0}".format(DEVICE))
    child.expect("Connection successful", timeout=20)
  except pexpect.TIMEOUT:
    NOF_REMAINING_RETRY = NOF_REMAINING_RETRY-1
    if (NOF_REMAINING_RETRY>0):
      print "timeout, retry..."
      continue
    else:
      print "timeout, giving up."
      break
  else:
    print("Connected!")
    break
    # Presence Sensor

conend = time.time()
print(start)
contime = conend - start
elapsed = 0

while (elapsed < 120 - contime):

    hex =""
    child.sendline("char-read-hnd 0x0038") # put the handle you want to
        read
    child.expect("Characteristic value/descriptor: ", timeout=50)
    child.expect("\r\n", timeout=50)
    print("Presence Sensor:  "),
    #print(child.before),
    test = (child.before).split(" ")
    for i in test:
        hex = hex + i
    x = bytearray.fromhex(hex).decode()

    print(x)
    print(sys.getsizeof(x))

    timestamp = int(time.time())
    print (timestamp)
    now = time.time()
    time.sleep(1)
    elapsed = now - conend
    print(elapsed)
    time.sleep(1)
```

Appendix 3 Code used in Raspberry Pi: Send data to Waspmote

```
import pexpect
import time
import sys
import firebase_admin as fb
from firebase_admin import credentials,db
from datetime import datetime



cred = credentials.Certificate("/home/pi/ble-drone-firebase-adminsdk
    -rxato-2d3b8cae12.json")
fb.initialize_app(cred, {
    'databaseURL' : 'https://ble-drone.firebaseio.com'})
DEVICE = "00:07:80:05:90:CC " # address of your device
#"00:07:80:05:42:F3"
# Run gatttool interactively.
child = pexpect.spawn("gatttool -I")

root = db.reference()
# Connect to the device.
print("Connecting to:"),
print(DEVICE)
NOF_REMAINING_RETRY = 3


while True:
  try:
    child.sendline("connect {0}".format(DEVICE))
    child.expect("Connection successful", timeout=20)
  except pexpect.TIMEOUT:
    NOF_REMAINING_RETRY = NOF_REMAINING_RETRY-1
    if (NOF_REMAINING_RETRY>0):
      print "timeout, retry..."
      continue
    else:
      print "timeout, giving up."
      break
  else:
    print("Connected!")
    break
    # Presence Sensor

start = time.time()
print(start)
elapsed = 0

while (elapsed < 60):

    child.sendline("char-write-req 0x0020 abcd")
    child.sendline("char-write-req 0x0020 1234")
    timestamp = int(time.time())
    now = time.time()
    elapsed = now - start
    print(elapsed)
```

Appendix 4 Code used in Waspmote: Retrieve data from Raspberry Pi

```
#include <WaspBLE.h>
uint8_t flag = 0;
void setup()
{
  // 0. Turn BLE module ON
  BLE.ON(SOCKET0);
  // 1. Make Waspmote visible to other BLE modules
  BLE.setDiscoverableMode(BLE_GAP_GENERAL_DISCOVERABLE);
  // 2. Make Waspmote connectable to any other BLE device
  BLE.setConnectableMode(BLE_GAP_UNDIRECTED_CONNECTABLE);
  USB.println(F("Waiting for incoming connections..."));
}
void loop()
{
      // 4. Now try to read a remote attribute.
    USB.println(F("Reading attribute.. "));
    BLE.attributeRead(BLE.connection_handle,0x0020);
    // 4.1 Print attribute value. First byte of BLE.attributeValue is the length of the value.
    USB.print(F("Attribute Value: "));
    for(uint8_t i = 0; i < BLE.attributeValue[0]; i++)
    {
      USB.printHex(BLE.attributeValue[i+1]);
    }
    USB.println();
}
```

Appendix 5 MATLAB code of the simulation

```
Tf = 1:1:25;       % Total flying time(min)
M  = 6;          % Tc/Tf (90mins charge => 25mins fly)
x  = 20;          % Sensor data rate(bytes/min)
R = 105000;        % Data transfer data rate (bytes/min)=> 14kbps
Tdp = 1;             % Data processing time (seconds)
d = 100;             % Distance between 2 sensors (meters)
s = 12;              % Speed of the drone (meters/seconds)
Tdl = 10;            % Data download time (seconds)

n1 = zeros(1,(size(Tf,2)));
n2 = zeros(1,(size(Tf,2)));
n3 = zeros(1,(size(Tf,2)));
n4 = zeros(1,(size(Tf,2)));

for i = 1:size(Tf,2)

Tdu = (x*(1+M)*Tf(1,i))/R;                % data upload time for 1 sensor
%linear
num1 = Tf(1,i) + (2*d/(60*s));
den1 = Tdu + (Tdp/60) + (Tdl/60)+ (2*d/(60*s));
%circular
num2 = Tf(1,i);
den2 = Tdu + (Tdp/60) + (Tdl/60)+ (d/(60*s));

%square => odd
w = Tdu + (Tdp/60) + (Tdl/60);
m = 60*s*w;
c = (1 + (1/sqrt(2)))*d + 60*s*Tf(1,i);
a = m+d;
b = d/sqrt(2);
num3 = -b + sqrt(b^2 + 4*a*c);
den3 = 2*a;



n1(1,i) = num1/den1;% linear
n2(1,i) = num2/den2;%circular
n3(1,i) = (num3/den3)^2;%square => odd
end
subplot(1,3,1)
plot(Tf, n1)
title("Linear Sensor Array")
legend('T_c = 30mins','T_c = 1hour','T_c =1.5hours','T_c = 2.5hours')
xlabel("Total Flying Time(min)")
ylabel("Number of Sensor Nodes")
ylim([0 100])
hold on
grid on

subplot(1,3,2)
plot(Tf, n2)
title("Circular Sensor Array")
legend('T_c = 30mins','T_c = 1hour','T_c =1.5hours','T_c = 2.5hours')
xlabel("Total Flying Time(min)")
ylabel("Number of Sensor Nodes")
ylim([0 100])
hold on
grid on

subplot(1,3,3)
plot(Tf, n3)
title("Square Sensor Array")
legend('T_c = 30mins','T_c = 1hour','T_c =1.5hours','T_c = 2.5hours')
xlabel("Total Flying Time(min)")
ylabel("Number of Sensor Nodes")
ylim([0 100])
hold on
grid on
```