PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

Ministry of Higher Education and Scientific Research

University Mohammed Khider BISKRA

Faculty of the Exact Sciences, Natural and Life Sciences

Department of Computer Science

Order number: xx /M2/2020

Report

Presented to obtain the diploma of academic master in Computer Science

Path: Network and Information and communications technology

(***RTIC***).

An Opportunistic publish/subscribe communications with UAV-based Broker, in smart city applications.

By :

Nouioua Djamel Eddine

Defended on **27/09/2020**, in front of the jury composed of:

President	**xx**	**Xx**
Supervisor	Dr. Sahraoui Somia	University of Biskra
Members	**xx**	**Xx**

Academic year: 2019/2020.

Dedication

To my father, ***Said Nouioua,***

To my mother, ***Malika Ourari,***

To my brothers, ***Taha, Islam, Ayoub*** *and* ***Abdellah,***

And who are always here for me.

Thanks

First, I thank **ALLAH** for giving me the courage, the strength and the patience to complete this work.

I would like to warmly thank my research director, **Dr. Sahraoui Somia** for her advices, encouragements and patience the face of my doubts, my mistakes and my many reversals in this work. I thank all those who have accompanied me and helped me to achieve this long-term work.

Thanks, finally, to my parent, my brothers, my friends and my colleagues for their support, which enabled me to carry out this research to its end.

Table of contents

Chapter 2:
Overview on Opportunistic Computing and Publish /Subscriber Communications.

Chapter 3:
Overview on our Solution: Publish/Subscribe Communications with Opportunistic-UAV-based broker in smart city context.

Chapter 4:

Evaluation of the solution and results.

List of Figures

List of Figures

List of Tables

List of Tables

ملخص:

يشهد العلم اليوم حرب تكنولوجية كبيرة في مختلف المجالات (برمجيات، الكترونيات، شبكات... الخ)، حيث اصبحت تغطي بصفة رئيسة جُل خدمات الإنسان. من بين اهم التكنولوجيات الصاعدة هي انترنت الاشياء هذه التكنلوجيا تتركز بشكل كلي على شبكة المستشعرات الذكية والتي لها القدرة على التقاط وجمع البيانات بشتى انواعها واصدار قرارات بالإضافة إلى تفاعلها المتكامل بين بعضها البعض. مما جعل العلماء والباحثين يرون ان انترنت الاشياء هي اصدار جديد للإنترنت وأنها ايضا منبع حقيقي لتطبيقات مذهلة بإمكانها احداث التغيير في هذه الاطروحة سنقوم باستغلال تكنلوجيا انترنت الاشياء في تطبيقات المدن الذكية حيث هناك ملايين الاجهزة المختلفة المتصلة ببعضها، وكم كبير من البيانات تحتاج الى التسيير والمعالجة.

اهم شيء يميز هذه التكنلوجيا هو انواع الاتصال التي تستعمله، هناك عدت انواع اصدرها الباحثون بإمكانها توصيل الاجهزة الذكية المختلفة ببعضها البعض. سنركز في عملنا هذا على أشهر الانواع وهو اتصال الناشر المشترك الذي يعتمد على 3 عناصر اساسية الناشر المشترك والوسيط. هذا الاخير هو اهم عنصر تبنى عليه الشبكة. لأنه يربط بين الناشر والمشترك .سوف نستغل هذا الاتصال لمحاكاة تطبيق للمدينة الذكية حيث سنقوم بإدراج طائرات بدون طيار (UAVs) وهي أجهزة ذكية مهمة في تطبيقات المدن الذكية. ستمكننا هذه الطائرات من إنشاء اتصال مباشر بين الأجهزة الذكية عن طريق نقلها إلى المكان المطلوب. أظهرت عمليات المحاكاة التي أجريت على جهاز محاكاة ذكي مخصص للمدينة (Cupcarbon) كفاءة الحل الذي نقدمه.

Abstract:

Today, science is witnessing a great technological war in various fields (software, electronics, networks, etc.), as it has mainly covered most human services. Among the most important emerging technologies is the Internet of Things (IoT). This technology is a big network of interconnected smart sensors, which have the ability to capture and collect data of all kinds and make decisions in addition to their integrated interaction between each other. This made scientists and researchers see the Internet of Things as a new version of the Internet. In this memory, we will focus on Internet of things technology in smart city applications. We make use of the popular communication model (publish/subscribe), which relies on three basic entities: publisher, subscriber and broker, the latter is the most important component of the network; because it connects the publishers with subscribers. We will take advantage of this connection to simulate a smart city application where we include drones (UAVs) which are important smart devices in smart city applications. These aircrafts will enable us to create direct communication between smart devices by moving them to the required place. The simulations conducted on a smart city-dedicated simulator (Cupcarbon) have shown the efficiency of our solution.

General Introduction:

The internet of things or IoT is a big smart network of sensors that implements various technologies such as (ZigBee, RFID, Wi-Fi ..etc). Thus, IoT uses several communication models like publish /subscribe, request/response, push/pull, etc. These models bring large range of use cases and applications in several domains. In this work we focus on the smart city application, where large amount of devices that generate big data that need efficient communication, storage and analysis strategies.

The publish subscribe communication is the most popular communication model in IoT. It is based on three important entities: the publisher, the subscriber and the broker. This communication used a lightweight message exchange between the subscribers and publishers using the event bus or broker. Typically, the broker becomes in between publishers and subscribers. Several protocols that are based on such a communication model (like MQTT, MQTT-SN, DDS, etc.) may use different types of brokers. The MQTT-SN is the most appropriate due to its suitability to IoT constraints.

The UAVs (Unmanned aerial vehicles) are smart flying devices that make part of IoT's world. In this work, we have worked on a solution that uses UAV-based brokers to carry opportunistic publish/subscribe communications, in the context of Smart city applications with the aim to achieve this goal at low cost, and low energy consumption. To assess this new approach, we used the Cupcarbon simulator that gives the possibility to simulate IoT networks in a smart city context.

This memory is organize in four chapters:

Chapter 1: Generalities on the Internet of Things.

Chapter 2: Overview on Opportunistic Computing and Publish /Subscribe Communications.

Chapter 3: Overview on our Solution: Publish/Subscribe Communications with Opportunistic-UAV-based broker in smart city context.

Chapter 4: Evaluation of the solution and results.

CHAPTER 1:

Generalities on the Internet of Things

1 Introduction :

The internet is constantly evolving and provides many services to improve the quality of life. Nevertheless, to provide the best progress of services, we must interconnect all things surrounding us (things and peoples …etc.) through a global internet so called the Internet of Things (IoT)This new scenario allowed "smart" electronic devices (such as televisions, refrigerators, doors, , homes ,drones ....etc.) to connect to the global network to send and receive information. Likewise, people can access these devices (objects or things) from anywhere [1].

Internet of Things (IoT) technology is the extension of the Internet network to smart things / objects and places in the physical world. Alternatively, it can be said for these things as a listener and collector of information in a real environment and in different places.

The IoT is a center of analyzes from which we can make decisions for all these things by the data obtained across a specific network, This data will be stored in cloud or broker ... etc. up to a specific time. With this technology, we will have a great improvement in internet services and with high quality.

2 Definition of Internet of Things:

The internet of things is a scenario in witch object ,animal or people are provide with an unique identifiers and the ability to automatically transfer data over a network without human-to-human communication or human to computer interaction. The IoT is the network of physical object that contain embedded technology to communicate and sense of interact with their internal state or the external environment. **[See Figure 1.1]**

Figure 1. 1: Internet of Things (IoT).

3 The History of IoT:

The history of IOT have been visions of smart, communicating objects even before the global computer network was launched forty-five years ago. Here are a few milestones in the evolution of the mashing of the physical with the digital. [2]

- **1926**

Nicola Tesla

When wireless is perfectly applied the whole earth will be converted into a huge brain , which in fact it is , all things being particles of a real and the instruments through which we shall be able to do this will be amazing simple compared with our present telephone . A man will be able to carry one in his vest pocket.

❖ **1998**

We then present inTouch, which applies Synchronized Distributed physical object to create a "tangible telephone "for long distance haptic communication. Scott brave and Professor Hiroshi Ishii.

InTouch (MIT project)

❖ **1998**

Mark Weiser, who developed a water fountain that was amazing and delightful to every one who saw it, touched the real IoT. It rose and fell respectively according to the pricing trends and the volume of stock on the NYSE.

Stock Fountain

❖ **1999**

I could be wrong, but I'm fairly sure the phrase "Internet of Things" started life as the title of a presentation I made at Procter & Gamble (p&g) in 1999. Linking the new idea of RFID in P&G's supply chain to the then-red-hot topic of the internet was more than just a good way to get executive attention.

Kevin Ashton

❖ **1999**

In the next century, planet earth will use the internet as a scaffold to support and transmit its sensations. This skin is already being stitched together. It consist of millions of embedded electronic measuring devices: pollution, detector, cameras, microphone…act. These will probe and monitor cities and endangered species, the atmosphere, our ships, highways and fleets of trucks, our conversations, our bodies-even our dreams.

Business Week

4 The Fundamentals characteristics of the IoT:

The fundamental characteristics of the IoT are as follows [3]:

Interconnectivity: In IoT technology, anything can be interconnected with the global information and communication infrastructure.

Things-related services: The IoT is capable of providing thing related services within the constraints of things, such as privacy protection and semantic consistency between physical things and their associated virtual things. In order to provide thing-related services within the constraints of things, both the technologies in physical world and information world will change.

Heterogeneity: The devices in the IoT are heterogeneous as based on different hardware platforms and networks. They can interact with other devices or service platforms through different networks.

Dynamic changes: The state of devices change dynamically, for example, sleeping and waking up, connected and/or disconnected as well as the context of devices including location and speed. Moreover, the number of devices can change dynamically.

Enormous scale: The number of devices that need to be managed and to communicate with each other will be at least an order of magnitude larger than the devices connected to the current Internet. Even more critical will be the management of the data generated and their interpretation for application purposes. This relates to semantics of data, as well as efficient data handling.

Connectivity: Connectivity enables network accessibility and compatibility. Accessibility is getting on a network while compatibility provides the common ability to consume and produce data**.**

Mobility: the object in internet of things like Mobile phones drones and vehicles «nowadays come equipped with advanced sensing and communication capabilities. These sensors can capture a wide range of information, as sensing data are collected from mobile devices, they can be transferred to a centralized server for storage and analysis.

5 Architecture of IoT:

There is no single consensus on architecture for IoT, which is agreed universally. Different architectures have been proposed by different researcher's .Three- and Five-Layer Architectures [4]; the most basic architecture is a three-layer architecture as shown **[Figure 1.2].**

- ***The perception layer*** is the physical layer, which has sensors for sensing and gathering information about the environment. It senses some physical parameters or identifies other smart objects in the environment.
- ***The network layer*** is responsible for connecting to other smart things, network devices, and servers. Its features are also used for transmitting and processing sensor data.
- ***The application layer*** is responsible for delivering application specific services to the user. It defines various applications in which the Internet of Things can be deployed, for example, smart homes, smart cities, and smart health.

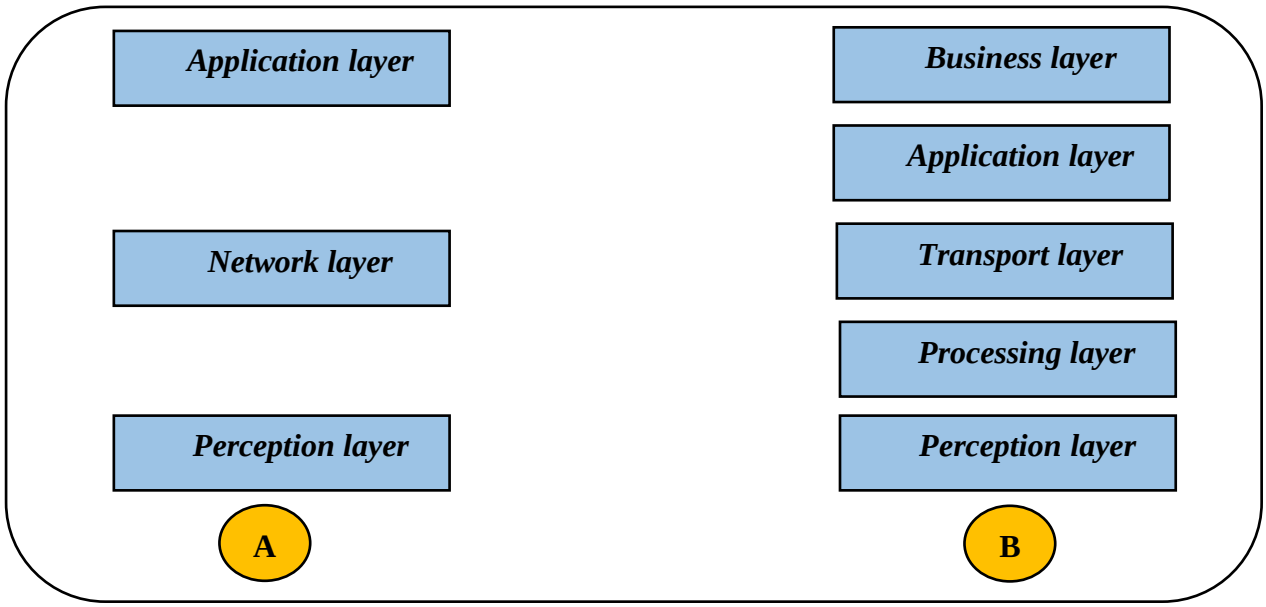

Figure 1. 2: Architecture of IoT (A: three layers) (B: five layers).

The three-layer architecture defines the main idea of the Internet of Things, but it is not sufficient for research on IoT because research often focuses on finer aspects of the Internet of Things. That is why, we have many more layered architectures proposed. One is the five-layer architecture **[see Figure 1.2],** which additionally includes the processing and business layers. The five layers are perception, transport, processing, application, and business layers the role of the perception and application layers is the same as the architecture with three layers. We outline the function of the remaining three layers.

- ***The transport layer*** transfers the sensor data from the perception layer to the processing layer and vice versa through networks such as wireless, 3G, LAN, Bluetooth, RFID, and NFC.
- ***The processing layer*** is also known as the middleware layer. It stores, analyzes, and processes huge amounts of data that comes from the transport layer. It can manage and provide a diverse set of services to the lower layers. It employs many technologies such as databases, cloud computing, and big data processing modules.
- ***The business layer*** manages the whole IoT system, including applications, business and profit models, and users' privacy.

Another architecture proposed by Ning and Wang [6]; is inspired by the layers of processing in the human brain. It is inspired by the intelligence, feel, remember, make decisions. It is constituted of three parts. First is the human brain, which is analogous to the processing and data management unit or the data center. Second is the spinal cord, which is analogous to the distributed network of data processing nodes and smart gateways. Third is the network of nerves, which corresponds to the networking components and sensors.

6 Typology physical of objects (Things):

- **Electronics:** like vehicle connected in 4G to optimize performance.
- **Electrical:** Everything related to home automation, remote ignition...etc.
- **Non-electric:** clothes, animals…
- **Environmental sensors:** drones**,** connected buildings, as depicted in **[Figure 1.3]**.

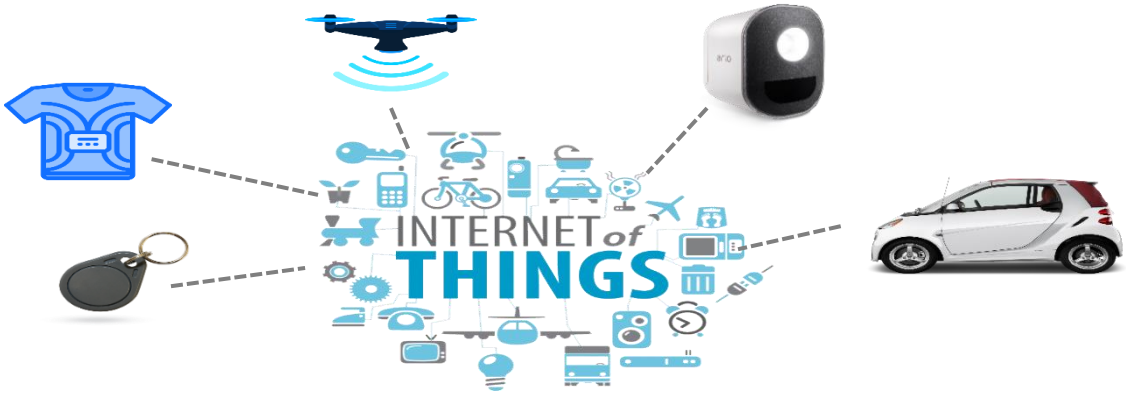

Figure 1. 3: Smart Objects in IoT.

7 Life cycle of object (Things) in Internet of Things:

A generic life cycle for IoT device describes the stages that an IoT device goes through from (re)construction to decommissioning [7]. **[See Figure 1.4]**

- The (re)construction stage pertains to the construction of a device hardware and its initial software and to the reconstruction of device firmware and the initial software.
- Then an IoT device goes to the production stage, while the reconstruction activity starts its iteration independently, parallel to the rest of the device life cycle.
- One iteration of reconstruction results in a software component or firmware update, which can then interrupt the device life cycle to go from operation to the update stage.
- The production stage pertains to the large-scale production of the device.
- The installation and commissioning stage prepares the device for operation and secure communication within the network.

[Figure 1.4] shows the activities involved in installation and commissioning.

- Configuring the location and group information for the device during commissioning allows the realization of functionalities that require this information.
- Configuring operational parameters during commissioning enables the device to operate as desired.
- Configuring keys and certificates for bootstrapping can be done during commissioning if they are not yet deployed during production.
- Bootstrapping in IoT relates to the process of acquiring configurations and secret keys for authentication and secure communication with other devices in the network.

❖ The operation stage supports the execution of IoT services and applications by running relevant system components. The performance of the system depends on how well these system components perform their functions.

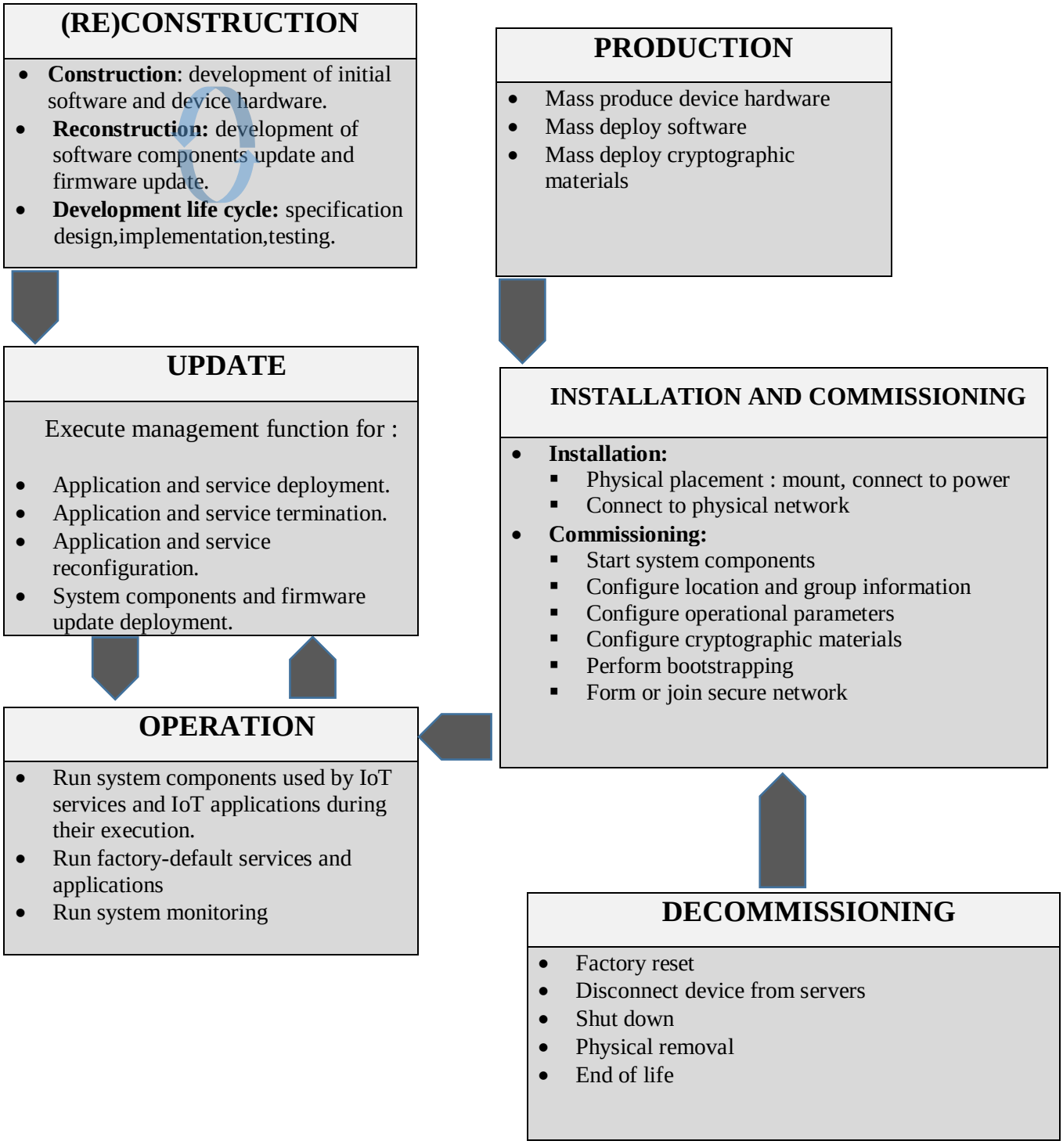

Figure 1. 4: Generic life cycle for IoT devices [7].

8 IoT Technologies

In This section, we will present a short overview of the major alternative Internet protocols specially dedicated for use by IoT systems. This overview concerns the most popular IoT radio technologies broken down by radio-frequency range achieved by each of the solutions: short-range IoT radio solutions, medium range solutions, and long-range Wide Area Networks solutions.

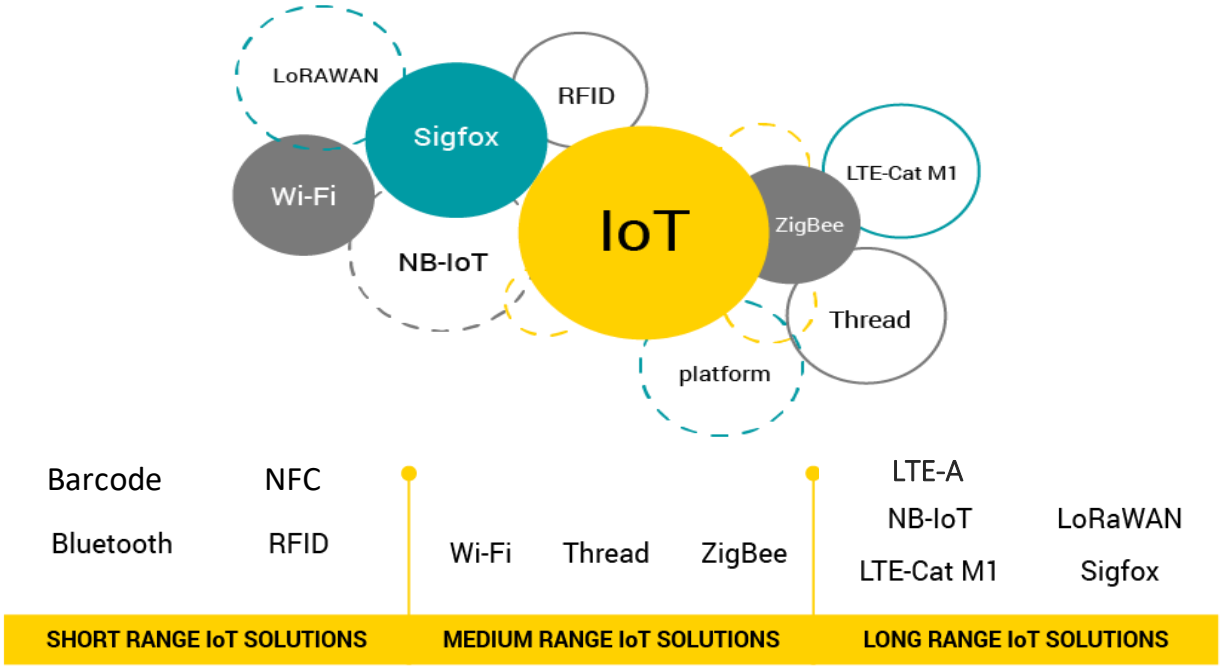

Figure 1. 5: Internet of Things Technologies [8].

8.1 Short-range IoT technologies :

- **Bluetooth:**

Bluetooth is a short-range wireless communication technology that permits devices, such as computers, mobile phones, and peripherals to transmit data or voice wirelessly over a short distance. The aim of Bluetooth is to substitute the cables that normally connect devices, while still keeping the communications between them secure. The "Bluetooth" name is taken from a Danish king named Harald Bluetooth; a set of Bluetooth device sharing a common channel for communication is called Piconet. This Piconet is capable of 2 - 8 devices at a time for data sharing, and that data may be pictures, video, text, and sound. The Bluetooth Special Interest Group comprises more than 1000 companies with HP, Aruba, Intel, Cisco, Ericson, IBM,

Motorola and Toshiba. Bluetooth v3.0 + HS. Bluetooth high-speed technology devices can deliver up to 24 Mbps of data, which is faster than the 802.11b Wi-Fi standard. [9]

- **Radio Frequency Identification (RFID):**

The RFID is a form of wireless communication that incorporates the use of electromagnetic or electrostatic coupling in the radio frequency portion of the electromagnetic spectrum to uniquely identify an object, person. A tag can be read from up to many feet away and does not require to be within direct line-of-sight of the reader to be tracked. RFID technology plays an essential role in IoT find a solution to the identification dispute of objects around us in a cost effective manner. The technology is classified into three categories based on the technique of power supply provision in Tags Active RFID, Semi Passive RFID and Passive RFID. The primary components of RFID are tagged, antenna, access controller, reader, software and server. It is more authentic, efficient, secured, cheap and accurate. [9]

- **Near filed Communication (NFC):**

Near Field Communication (NFC) is a set of short-range wireless technology at 13.56 MHz, typically be in need of a distance of 4 cm. Near-field communication is a wireless technology that permits users to make payments by placing a compatible device like a smartphone or payment card within a few centimeters of another compatible device like a terminal, tablet or another smartphone. NFC tags are passive devices. They store data that can be repossessed by active NFC devices. Near-field communication transmits data via electromagnetic radio fields and is the technology behind payment services like Apple Pay and Google Wallet. This device must include NFC chips for an NFC transaction to take place. It also works in a dirty environment, does not need line of sight, easy and unpretentious connection method. [9]

- **Barcode:**

A barcode, be made up of bars and spaces, is a machine-readable representation of numerals and characters. Today, packages of products sold at supermarkets, convenience stores and other stores are all over the place. These are barcodes. There are 3 types of barcodes of 2 dimensional, Numeric and Alpha Numeric. The barcodes are designed to be machine-readable and that can be read with an optical barcode scanner, they can also be read using a camera.

8.2 Medium range IoT technologies:

- **ZigBee:**

The ZigBee is a standards-based wireless technology developed to enable low-cost, low power wireless machine-to-machine and internet of things networks. ZigBee operates on the IEEE 802.15.4 specification and is used to create networks that need a low data transfer rate, secure networking and energy efficiency. It is employed in a number of applications such as heating and cooling control, building automation systems, heating and cooling control and in medical devices. ZigBee has range of around 100 meters and a bandwidth of 250 kbps and the topologies that its works are star, cluster tree and mesh. ZigBee is designed to be easier and less costly than the other person is networked technologies like as Bluetooth. [9]

- **Wireless Fidelity (Wi-Fi):**

Wi-Fi is a type of wireless network technology used for connecting to the Internet. The frequencies Wi-Fi works at are 2.4Ghz or 5GHz, make sure no interference with TV antenna, broadcast radio, cellphones and two-way radios are encountered during transmission. Wi-Fi is mainly just radio waves broadcast from a Wi-Fi router, a device detecting and deciphering the waves, and then sending back data to the router. It works very likewise for an FM & AM radio, but it is a two-way communication channel. Wi-Fi works over extended distances than Bluetooth or infrared and is also a low power unobtrusive technology, making it appropriate for portable devices such as laptops and palmtops. This technology include any type of WLAN product support any of the IEEE 802.11 together with dual-band, 802.11a, 802.11b, 802.11g and 802.11n. [9]

8.3 Long Range IoT technologies:

- **NB-IoT:**

A product of existing 3GPP technologies, Narrowband IoT is a brand-new radio technology standard that ensures extremely low power consumption (10 years of battery power operation) and provides connectivity with signal strength approx. 23 dB lower than in the case of 2G. What is more, it uses existing network infrastructure, which ensures not only global coverage in LTE networks, but also guaranteed signal quality. In many cases, this fact allows for implementing NB-IoT instead of solutions that required the construction of local networks, such as LoRa or Sigfox. [8]

- **LTE-Cat M1:**

LTE Cat M1 is a low-power wide-area (LPWA) connectivity standard that connects IoT and M2M devices with medium data rate requirements. It supports longer battery lifecycles and offers enhanced in-building range as compared to cellular technologies such as 2G, 3G, or LTE Cat 1.Being compatible with the existing LTE network, CATM1 doesn't require the carriers to build new infrastructure to implement it. As compared to NB-IoT, LTE Cat M1 proves to be perfect for mobile use cases, as its handling of hand-over between cell sites is significantly better and is very similar to high speed LTE.

- **LoRaWAN:**

LoRaWAN is a low-power Long Range Wide-Area Networking protocol optimized for low-power consumption and supporting large networks with millions of devices. Aiming at wide-area network (WAN) applications, LoRaWAN is designed to furnish low-power WANs with features required to support low-cost, mobile and secure bi-directional communication within IoT, M2M, smart city, and industrial applications.[8] **[See Figure 1.6]**

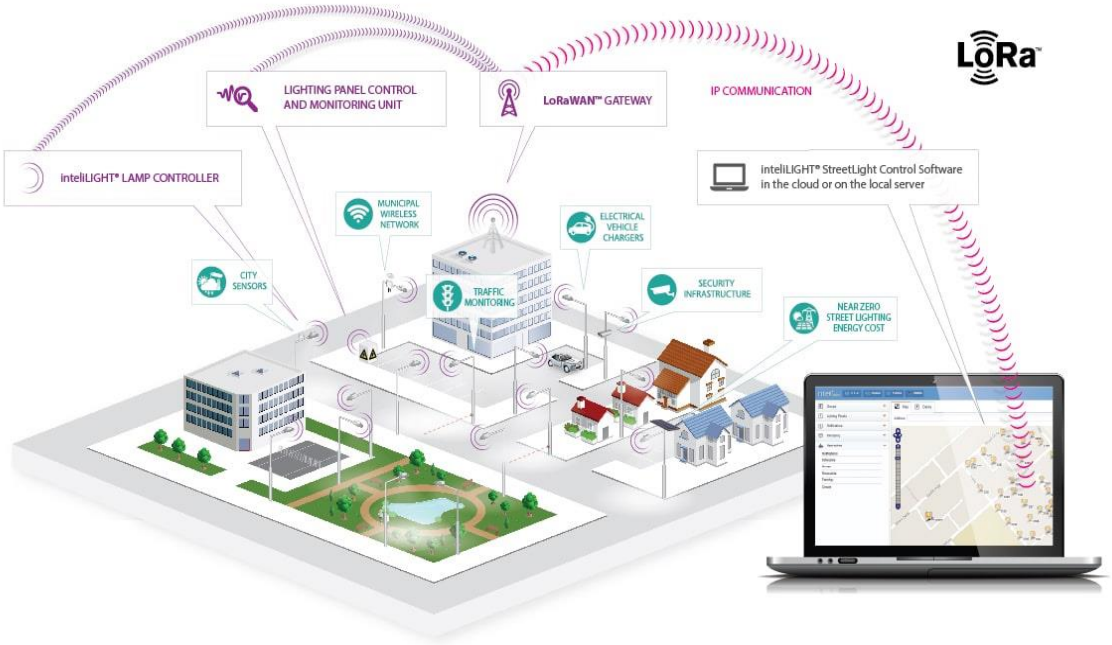

Figure 1. 6: LoRaWAN Technology [9].

- **Sigfox:**

The concept behind Sigfox is to provide an effective connectivity solution for low-power M2M applications requiring low levels of data transfer for which the WiFi range is too short, and cellular range is too expensive and too power-hungry. Sigfox employs UNB, a technology that enables it to handle low data-transfer speeds of 10 to 1,000 bits per second. Consuming up to 100 times less energy compared to cellular communication solutions, it delivers a typical stand-by time of 20 years for a 2.5Ah battery. Offering a robust, energy-efficient and scalable network able to support communication between thousands of thousands of battery-operated devices across areas of several square kilometres, Sigfox proves suitable for various M2M applications, including smart street lighting, intelligent meters, patient monitors, security devices, and environmental sensors. Sigfox is currently employed in a growing number of IoT technology solutions.[8] **[See Figure 1.7]**

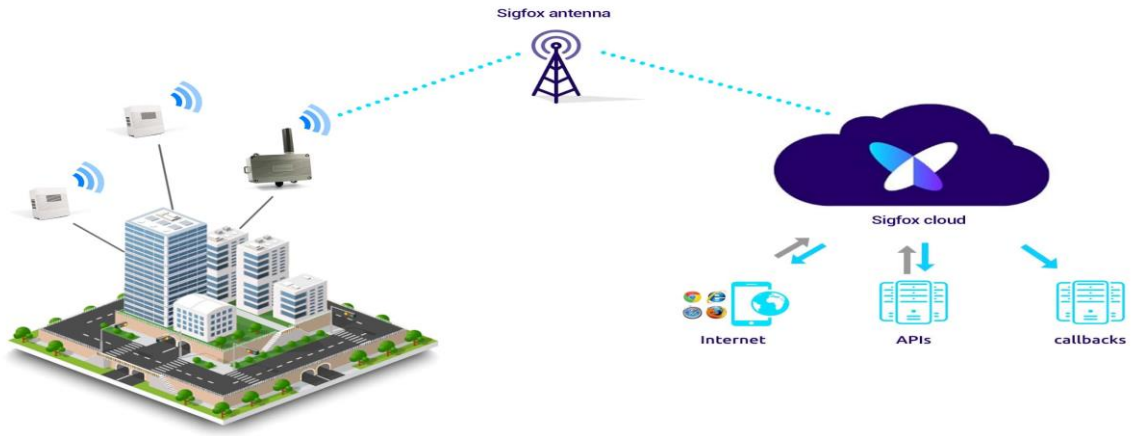

Figure 1. 7: Sigfox Technology.[8]

- **LTE-A:**

LTE-A stands for LTE-advanced incorporated a number of new techniques that enabled the system to confer very much higher data rates, and also much excellent performance, especially at cell edges and other areas where performance would not normally have been so healthy . LTE-A, delivers an essential upgrade to LTE technology by elongate, not only its coverage, but also deficiency its latency and raising its throughput. It gives IoT a terrific power via expanding its range, with its most valued applications being vehicle, UAV, and identical

communication. The OFDM and MIMO are two of the fundamental technologies that will be enablers. LTE D2D is a provision that has been requested by a number of users, in special the emergency services. It enables fast, intense access via direct communication. Accordingly, LTE-advanced has provided refinement to both users and operators, as well as those endue additional services. [9]

9 Enabling technologies for IoT:

9.1 Unmanned Aerial vehicles (UAV):

Unmanned aerial vehicles (UAVs) have enormous potential in enabling new applications in various areas, ranging from military, security, medicine, and surveillance to traffic-monitoring applications. Lately, there has been heavy investment in the development of UAVs and multi-UAVs systems that can collaborate and complete missions more efficiently and economically. Emerging technologies such as 4G/5G networks have significant potential on UAVs equipped with cameras, sensors, and GPS receivers in delivering Internet of Things (IoT) services from great heights, creating an airborne domain of the IoT. However, there are many issues to be resolved before the effective use of UAVs can be made, including security, privacy, and management. [10]

9.2 Wireless Sensor Network:

9.2.1 WSN definition:

Typically, a WSN can be defined as a network of nodes that work in a cooperative way to sense and control the environment surrounding them. These nodes are linked via wireless media. Nodes use this connection to communicate among each other. The architecture of a typical WSN consists of following 3 components: sensor nodes, gateway and observer (user). Sensor nodes and gateways constitute the sensor field. Gateways and observers are interconnected via special networks or more commonly via internet **[see Figure. 1.8].** Depends on the fact that Sensing + CPU + Radio = Lots of Potential. Sensing Unit is necessary to monitor surrounding environment and its conditions such as humidity, pressure and vibration. After completing monitoring and sensing processes, necessary computations are accomplished in CPU. Lastly, Radio Unit transfers computed environmental data through the wireless communication channels among the nodes. Finally, these data are sent towards the Gateway.[11]

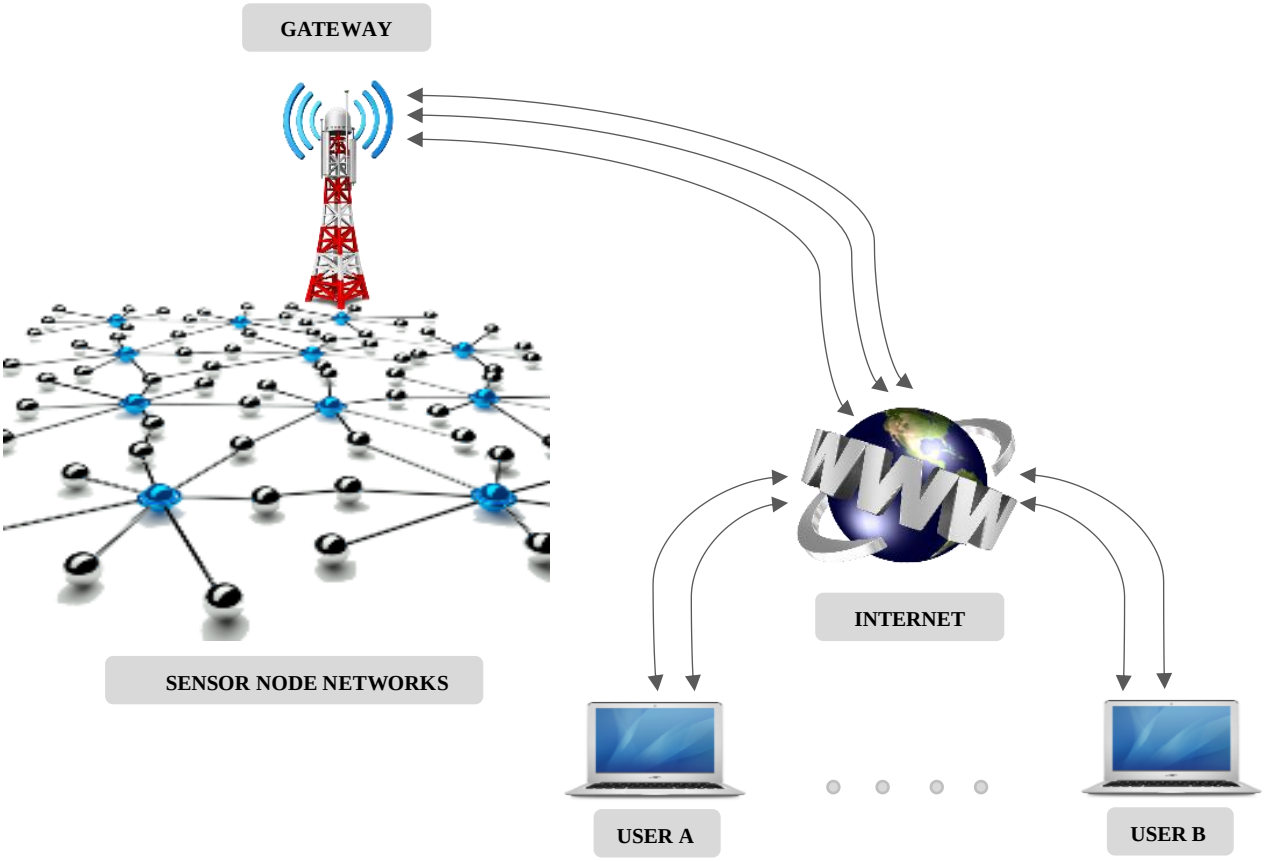

Figure 1. 8: Wireless Sensor Network (WSN). [11]

9.2.2 WSN Composition:

A WSN is composed of several numbers of sensors and a gateway to provide connection to the Internet. [11]

- **Sensor Node**

  Sensor node is one of the main components of any WSN . A sensor node is a low powered small device. Although it has limited energy resources, it has concurrent processing feature and also it has a low cost.

- **Power Source**

  Power Source is placed to the sensor node's base. It supplies energy for various units of sensor nodes like sensing units (sensors), CPU and radio. In order to continue to perform

sensing, computing and communicating tasks; energy is needed. Therefore, ambient energy harvesting techniques (from external resources) are used to power small sensor nodes. Power resources can be watch batteries, solar cells or smart systems. For any sensor nodes the energy resources are limited and energy is crucial to perform all tasks. Therefore, nodes spend as much as 99% of their time in sleep to conserve energy. They only wake up to record data, to send data and to receive data.

❖ **Microcontroller**

Typically, the CPU (also called the electronic brain) of a sensor is composed of a microprocessor and a flash memory. In most of sensor nodes, it includes connectors to add external processing units and sensors to the main unit easily. Making decision and dealing with collected data can be listed as examples for the crucial functions of the CPU. The CPU stores data in flash memory until enough data has been collected. Once enough data is collected by the system, then microprocessor unit of the CPU puts the data in envelopes because envelopes provide great efficiency in data transmission. Then, these envelopes are sent to the radio for broadcast. Meanwhile the brain communicates also with other nodes in much the same way it deals with data to maintain the most effective network structure.

❖ **Sensor Transducer**

The most crucial part of a WSN is the sensors. Sensors convert environmental variables like light, smoke, heat, and sound etc. into electrical signals. In the past two decades, there has been rapid development in multiple sensing technologies.

These advancements have made sensors widely in use in daily life notably in sensor nodes. A typical node consists of three types of sensors, which are temperature, vibration and moisture. But some nodes can have extra features such as taking photographs of surroundings, sensing motion, sensing pressure, sensing smoke, sensing light, etc.

❖ **Transceiver**

It is responsible for the wireless communications of a sensor node. Transceiver has mainly four operational states which are Receive, Transmit, Idle and Sleep. As a wireless media, Radio Frequency (RF), Infrared and Laser can be chosen in transceiver. Among these wireless communication technologies, RF is widely preferred for WSNs. Typical operation range of RF (for the operation frequencies of WSNs) is 10s of meters indoors and 100s of meters outdoors.

❖ **Operating System**

Operating System Tiny OS, Contiki, SOS, MANTIS, BTunt are the examples of operating systems that are used for WNSs. Among these systems, Tiny OS is the one that is open source and energy efficient. Instead of multithreading, Tiny OS uses event driven programming methodology.

10 Internet of Things Communication Models:

We explain some major IoT communication models [12]:

10.1 Publish-Subscribe Model:

Publish-Subscribe is a communication model that involves publishers, brokers and consumers. Publishers are the source of data. Publishers send the data to the topics which are managed by the broker. Publishers are not aware of the consumers. Consumers subscribe to the topics, which are managed by the broker. When the broker receive data for a topic from the publisher, it sends the data to all the subscribed consumers.

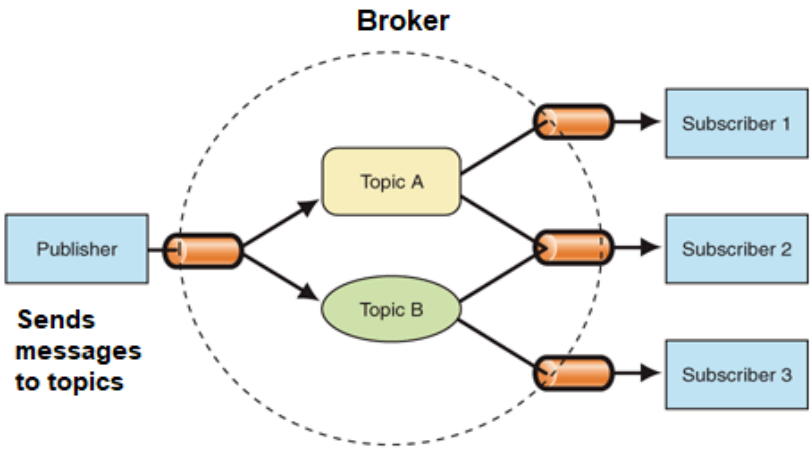

Figure 1. 9: Publish-Subscribe Model.

10.2 Request-Response Model:

Request-response model is communication model in which the client sends requests to the server and the server responds to the requests. When the server receives a request, it decides how to respond, fetches the data, retrieves resource representation, prepares the response, and then sends the response to the client. Request-response is a stateless communication model and each request-response pair is independent of others.

HTTP works as a request-response protocol between a client and server. A web browser may be the client, and an application on a computer that hosts a web site may be the server.

Example: A client (browser) submits an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

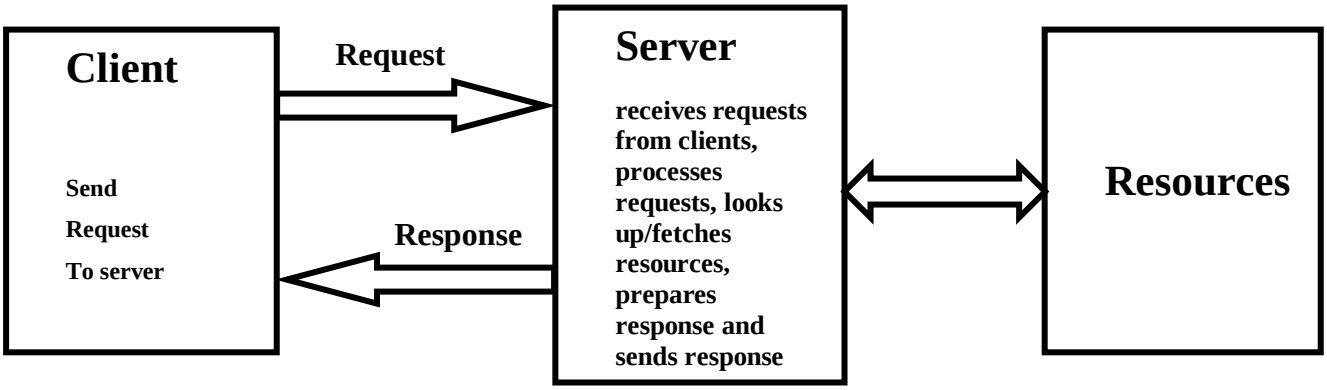

Figure 1. 10: Request-Response Model.

10.3 Push-Pull Model :

Push-Pull is a communication model in which the data producers push the data to queues and the consumers pull the data from the Queues. Producers do not need to be aware of the consumers. Queues help in decoupling the messaging between the Producers and Consumers. Queues also act as a buffer which helps in situations when there is a mismatch between the rate at which the producers push data and the rate at which the consumer pull data.

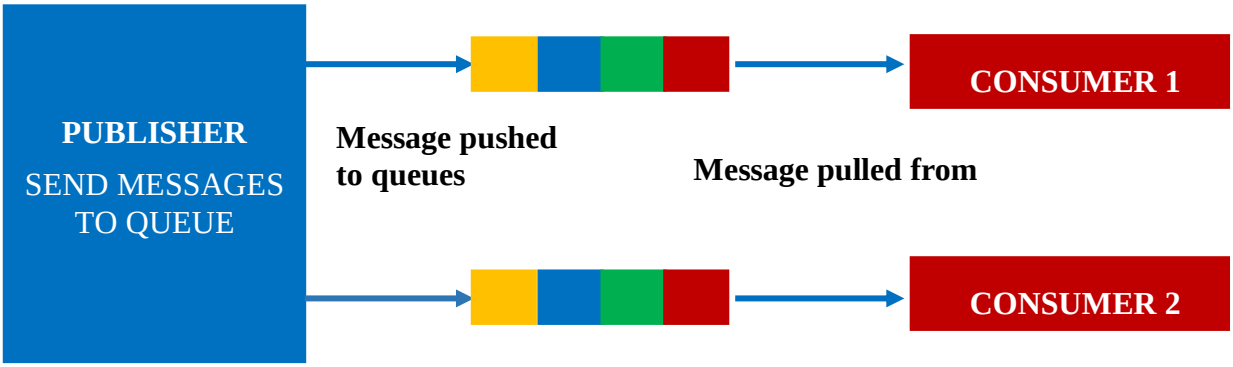

Figure 1. 11: Push-Pull Model.

10.4 Exclusive Pair Model

Exclusive Pair is a bidirectional, fully duplex communication model that uses a persistent connection between the client and server. Connection is setup it remains open until the client sends a request to close the connection. Client and server can send messages to each other after connection setup. Exclusive pair is state full communication model and the server is aware of all the open connections.

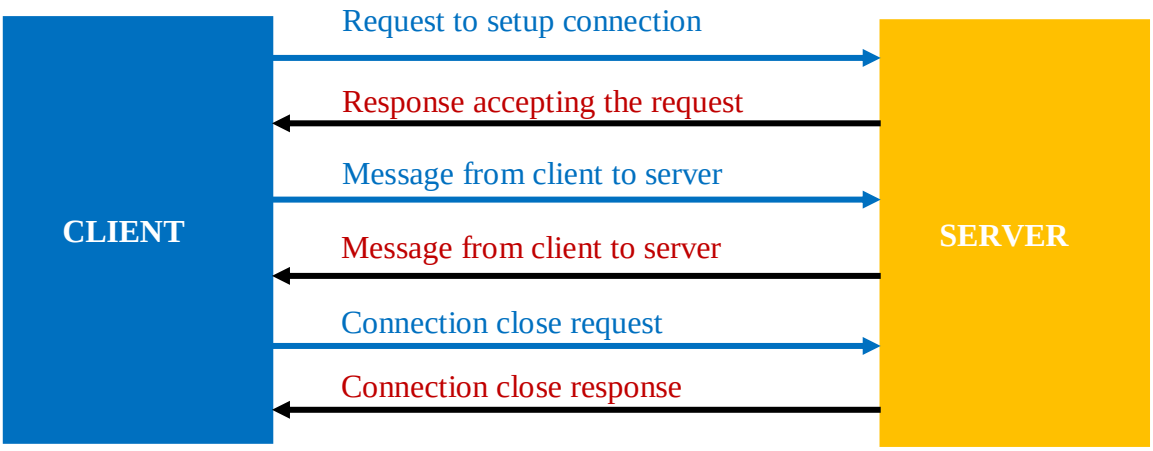

Figure 1. 12: Exclusive Pair Model.

11 Internet of Things Applications:

Potential applications of the IoT are numerous and diverse, permeating into practically all areas of every-day life of individuals, enterprises, and society as a whole. The IoT application covers “smart” environments/spaces in domains such as Transportation, Building, City, Lifestyle, Retail, Agriculture, Factory, Supply chain, Emergency, Healthcare, User interaction, Culture and tourism, Environment and Energy **[see Figure 1.13]**. Below are some of the IOT applications. [13]

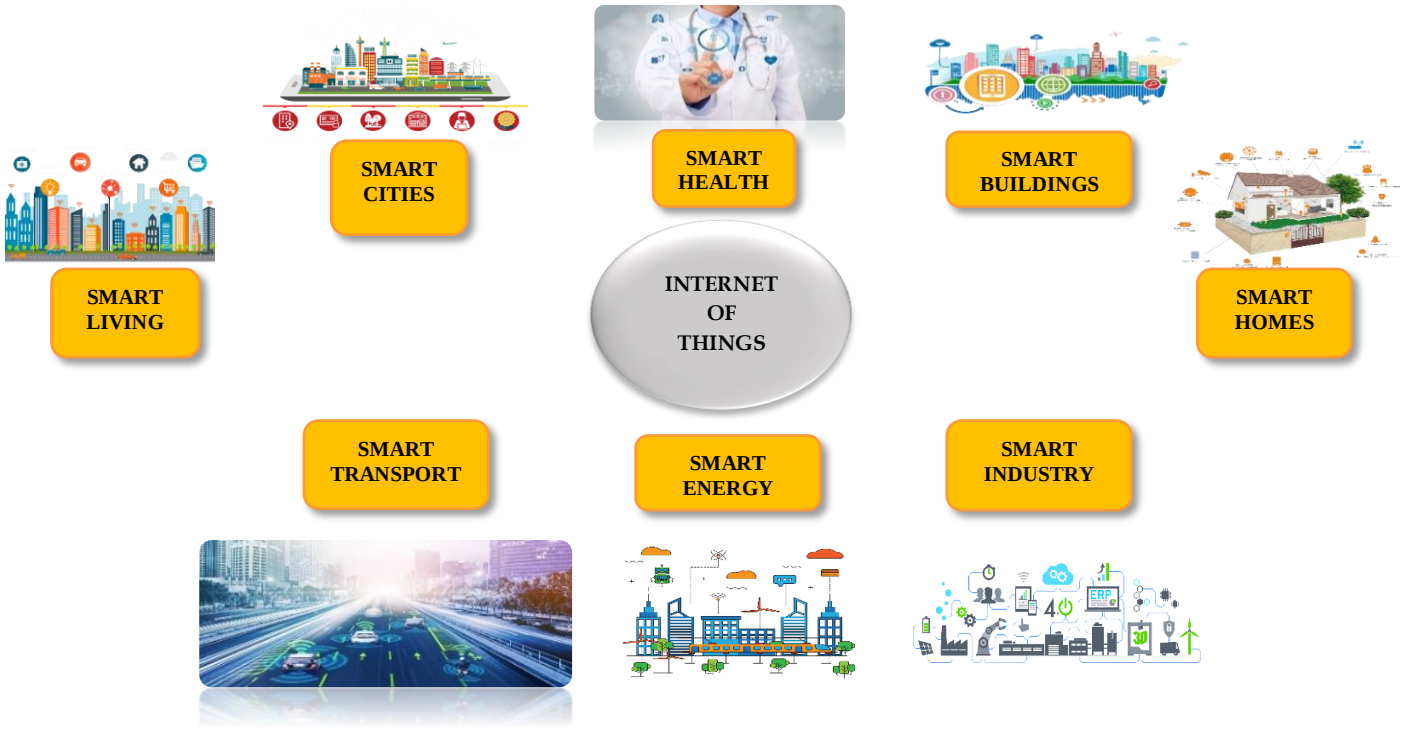

Figure 1. 13: IoT Applications.

A. IOsL (Internet of smart living):

- ***Remote Control Appliances****:* Switching on and off remotely appliances to avoid accidents and save energy.
- ***Weather:*** Displays outdoor weather conditions such as humidity, temperature, pressure, wind speed and rain levels with ability to transmit data over long distances.
- ***Smart Home Appliances:*** Refrigerators with LCD screen telling what's inside, food that's about to expire, ingredients you need to buy and with all the information available on a Smartphone app. Washing machines allowing you to monitor the laundry remotely, and. Kitchen ranges with interface to a Smartphone app allowing remotely adjustable temperature control and monitoring the oven's self-cleaning feature.

- ***Safety Monitoring:*** cameras, and home alarm systems making people feel safe in their daily life at home.
- ***Intrusion Detection Systems:*** Detection of window and door openings and violations to prevent intruders.
- ***Energy and Water Use:*** Energy and water supply consumption monitoring to obtain advice on how to save cost and resources.

B. IOsC (Internet of smart cities):

- ***Structural Health:*** Monitoring of vibrations and material conditions in buildings, bridges and historical monuments, Lightning: intelligent and weather adaptive lighting in street lights, Safety: Digital video monitoring, fire control management, public announcement systems.
- ***Transportation:*** Smart Roads and Intelligent High-ways with warning messages and diversions according to climate conditions and unexpected events like accidents or traffic jams.
- ***Smart Parking:*** Real-time monitoring of parking spaces availability in the city making residents able to identify and reserve the closest available spaces,
- ***Waste Management:*** Detection of rubbish levels in containers to optimize the trash collection routes. Garbage cans and recycle bins with RFID tags allow the sanitation staff to see when garbage has been put out.

C. IOsE (Internet of smart environment):

- ***Air Pollution monitoring:*** Control of CO2 emissions of factories, pollution emitted by cars and toxic gases generated in farms.
- ***Forest Fire Detection:*** Monitoring of combustion gases and pre-emptive fire conditions to define alert zones.
- ***Weather monitoring:*** weather conditions monitoring such as humidity, temperature, pressure, wind speed and rain, Earthquake Early Detection.
- ***Water Quality:*** Study of water suitability in rivers and the sea for eligibility in drinkable use.
- ***River Floods:*** Monitoring of water level variations in rivers, dams and reservoirs during rainy days.
- ***Protecting wildlife:*** Tracking collars utilizing GPS/GSM modules to locate and track wild animals and communicate their coordinates via SMS.

D. IOsI (Internet of smart industry):

- ***Explosive and Hazardous Gases:*** Detection of gas levels and leakages in industrial environments, surroundings of chemical factories and inside mines, Monitoring of toxic gas and oxygen levels inside chemical plants to ensure workers and goods safety, Monitoring of water, oil and gas levels in storage tanks and Cisterns,
- ***Maintenance and repair:*** Early predictions on equipment malfunctions and service maintenance can be automatically scheduled ahead of an actual part failure by installing sensors inside equipment to monitor and send reports.

E. IOsH (Internet of smart health):

- ***Patients Surveillance:*** Monitoring of conditions of patients inside hospitals and in old people's home.
- ***Medical Fridges:*** Control of conditions inside freezers storing vaccines, medicines and organic elements, Fall Detection: Assistance for elderly or disabled people living independent, Dental: Bluetooth connected toothbrush with Smartphone app analyzes the brushing uses and gives information on the brushing habits on the Smartphone for private information or for showing statistics to the dentist.
- ***Physical Activity Monitoring:*** Wireless sensors placed across the mattress sensing small motions, like breathing and heart rate and large motions caused by tossing and turning during sleep, providing data available through an app on the Smartphone.

F. IOsE (internet of smart energy):

- ***Smart Grid:*** Energy consumption monitoring and management,
- ***Wind Turbines/ Powerhouse:*** Monitoring and analyzing the flow of energy from wind turbines & powerhouse, and two-way communication with consumers' smart meters to analyze consumption patterns,
- ***Power Supply Controllers:*** Controller for AC-DC power supplies that determines required energy, and improve energy efficiency with less energy waste for power supplies related to computers, telecommunications, and consumer electronics applications,
- ***Photovoltaic Installations:*** Monitoring and optimization of performance in solar energy plants.

12 IoT Security :

Within any IoT architecture, threats are everywhere and vulnerabilities ca be exploited for a number of malicious purposes. [14]

- **Network:** By taking control of a gateway or router, an attacker can steal data that is being communicated between the IoT devices and the backend data management system. Or broadcast fake content to the devices or the backend infrastructure.
- **Application:** Finally, attackers can attempt to take control of the IoT application giving them control of both the IoT devices and backend system.
- **Devices:** at the sensor an attacker can attempt to take control of the device, or insert unauthorized devices in the IoT architecture (a man-in-the-middle attack).the gateways and routers that IoT devices communicate within an IoT architecture are also potentially vulnerable to an attack.
- **Chip:** Attackers can attempt to take control of device by targeting its microprocessor or integrated circuit(IC). Security control. Particularly in ICs, have Historically been very weak.

  Cryptography is the foundation of IoT security and is implemented by using hardware and software technologies .Cryptography is the science of encrypting and decrypting data communication in order to project information .the three main function of cryptography are: **Confidentiality, integrity** and **authentication**.

13 Conclusion

Internet of things is the concept in which the virtual word of information and services connected with the real word across a specific model of communication. Each thing in IoT can be provide a service using a real time Applications. Therefore, we will need a real-time communication model with highlight messages and infrastructure more organised.

In the next chapter, we will talk about a publish/subscriber communication where have a sensor make our life become better and comfortable.

CHAPTER 1:

Overview on Opportunistic Computing and Publish /Subscriber Communications

1 Introduction:

In This chapter, we will present the opportunistic computing/communications, as well as the publish /subscribe communication model in internet of things context.

The communication between things is the most important stage in the Internet of things, or can be said that is the basis for the interaction of things between each other.

This Communication branched in several models for example Publish-Subscribe, Request-Response, Push-Pull, Exclusive Pair (we identified them in chapter 1), Each model has its own preferences that are limited by lowest energy consummation with lowest cost and good quality of service and special harmony with environment and sensors. .etc.

Publish/Subscribe (Pub/Sub) systems have been gaining popularity for every sort of applications; it is a communication mechanism or paradigm for distributed systems. Each node that is on a publish/subscribe network communicates with each other by publishing data and receiving (subscribing) data anonymously; in this section we will focus on publisher/ subscriber communication and its principal protocols like AMQP(Advanced Message Queueing Protocol), MQTT (Message Queueing Telemetry Transport), MQTT_SN (Message Queueing Telemetry Transport for Sensor Network), DDS (Data Distribution Service), XMPP (Extensible Messaging and Presence Protocol). .etc. when used in specific IoT applications with an opportunistic idea for more improvement in several applications.

2 Opportunistic computing:

Opportunistic computing utilize the shared resources, content, services, applications, and computing resources, by the devices connected in an opportunistic mobile social network, to provide a platform for the execution of distributed computing tasks. [15]

In current time and with available means we can making a communication and transmit more when and where the channel is good.

Exploits fading to achieve higher long-term throughput, but no guarantee that the "channel is always there".

3 Publish subscriber communication model:

A different way for devices to communicate on a network is called publish-subscribe, or pub-sub. In a pub-sub architecture, a central source called a broker (also sometimes called a server) receives and distributes all data. Pub-sub clients can publish data to the broker or subscribe to get data from it or both.

Clients that publish data send it only when the data changes (report by exception, or RBE). Clients that subscribe to data automatically receive it from the broker/server, but again, only when it changes. The broker does not store data; it simply moves it from publishers to subscribers. When data comes in from a publisher, the broker promptly sends it off to any client subscribed to that data. [16] **[See figure 2.1]**

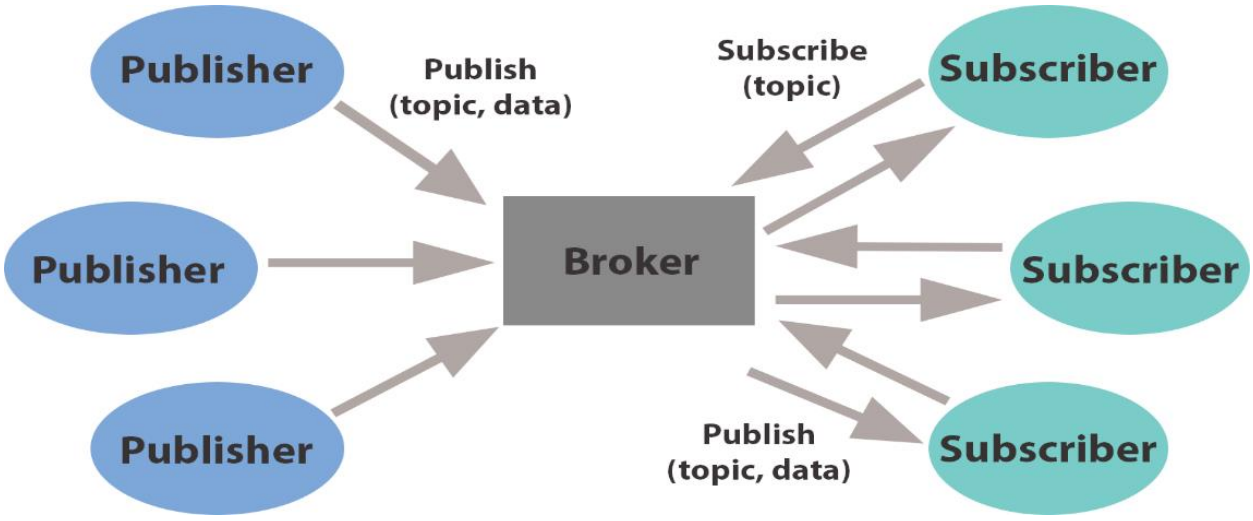

Figure 2. 1: Publish/subscribe communication model. [17]

4 Publish/Subscriber Architecture (PSA):

Publish Subscribe or (pub/sub) model has three main entities. **Publisher**, **Broker**, **Subscriber**. **[See figure 2.1]**

Senders of messages called publishers: it does not program the messages to be sent directly to specific receivers, because it has no knowledge of who the receivers are; so just categorize the messages in specific topics and sent them to the broker.

Recipients of interested messages called subscribers: these entities listen out for messages reading the topic or categories that are interested in without any knowledge of who the publishers are. Finally the broker or event bus: this entity transfer the messages from the publishers to Subscribers. Each subscriber only receives a subset of the messages that have been sent by the publisher. They only receive the message topics or categories they have subscribed to.

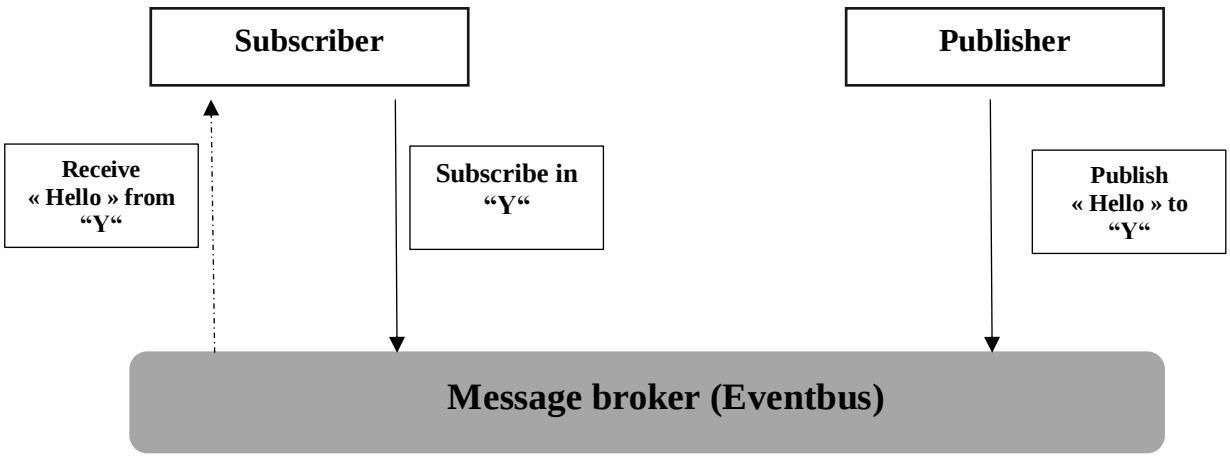

Figure 2. 2: Publish/subscribe communication model. [19]

The most important aspect of pub/sub is the decoupling of the publisher of the message from the recipient (subscriber) [20].This decoupling has several dimensions:

- **Space decoupling:** Publisher and subscriber do not need to know each other (for example, no exchange of IP address and port).
- **Time decoupling:** Publisher and subscriber do not need to run at the same time.
- **Synchronization decoupling:** Operations on both components do not need to be interrupted during publishing or receiving.

The broker plays important role in the pub/sub process. But how does the broker manage to filter all the messages so that each subscriber receives only messages of interest? As you will see, the broker has several filtering options. [20]

4.1 Topic-based filtering:

This filtering is based on the subject or topic that is part of each message. The receiving client subscribes to the broker for topics of interest. From that point on, the broker ensures that the receiving client gets all message published to the subscribed topics. In general, topics are strings with a hierarchical structure that allow filtering based on a limited number of expressions.

4.2 Content-based filtering:

In content-based filtering, can publish the content and subscriber follows the content as per need. In this system, publisher publishes the content on the message system. After this, message broker broke the message to know message content. Then send this message to the particular subscribers who want to know about this content. Message brokers use filtering pattern that applied on consumers subscription to elect events by using a subscription language.

4.3 Type-based filtering:

When object-oriented languages are used, filtering based on the type/class of a message (event) is a common practice. For example, a subscriber can listen to all messages and Acknowledges, which are of type Exception or any sub-type.

5 Publish/Subscribe Benefits:

The main benefits of Publish/Subscribe are mention below [18]:

- Implement the publisher and subscriber parties independently from each other.
- Publishers and Subscribers don't require to know each other.
- One Subscriber could receive from many different Publishers.
- One Publisher could send data to many different Subscribers.
- A major advantage is in simplicity and flexibility of decentralisation implementation as this enables to the system to support a large number of client and huge amount of data transfers, hugely scalable.

6 Publish/Subscribe Drawbacks:

The main Drawbacks of Publish/Subscribe are mention below [18]:

- The potential loss of messages due to events needed to be pruned.
- There in no guarantees in delivery of messages as the message, that stored for longer period might be discarded. Less control over ordering of messages.
- PSA is designed to delivery message for a specified time, but then stopped attempting delivery.
- Increased latency.

7 Comparison between the IoT Application Protocols that using Publish/Subscribe communication:

Some IoT standards are proposed to help developers and service providers in the application layer. In addition, various groups created standards to support IoT protocols, e.g., World Wide Web Consortium (W3C), Internet Engineering Task Force (IETF), EPC global, Institute of Electrical and Electronics Engineers (IEEE) and the European Telecommunication Standards Institute (ETSI). Table in **[Table 2.1]** shows the most important protocols and their differences defined by these groups. [21]

Table 2. 1: Comparison between the IoT Application Protocols that using Publish Subscribe communication.

Application Protocols	Restful	Transport	Publish/Subscribe	Request/Response	Security	QoS	Header Size (Byte)
COAP	V	UDP	V	V	DTLS	V	4
MQTT	-	TCP	V	-	SSL	V	2
MQTT-SN	-	UDP / Others (e.g. ZigBee)	V	-	SSL	V	2
XMPP	-	TCP	V	V	SSL	-	-
AMQP	-	TCP	V	-	SSL	V	8
DDS	-	TCP/UDP	V	-	SSL/DTLS	V	-

8 The IoT Application Protocols for Publish Subscribe communications:

There are many protocols in IoT applications used Publish/Subscribe communication model, we mention some important protocols:

8.1 DDS (Data Distribution Service):

DDS is an IoT standard for real-time, scalable and high performance machine-to-machine communication. It was developed by the Object Management Group (OMG). We can deploy DDS both in low-footprint devices and in the cloud [22]. The DDS standard has two main layers:

- Data-Centric Publish-Subscribe (DCPS), which delivers the information to subscribers
- Data-Local Reconstruction Layer (DLRL), which provides an interface to DCPS functionalities.

8.2 AMQP (Advanced Message Queuing Protocol)

AMQP is an application layer protocol for message-oriented middleware environments. It is approved as an international standard. The processing chain of the protocol includes three components that follow certain rules. Exchang_ets messages and puts them in the queues Message queue_stores messages until they can be safely processed by the client app Binding_states the relationship between the first and the second component. [22]

8.3 Message Queuing Telemetry Transport (MQTT):

8.3.1 What is an Mqtt protocol?

MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol in application layer. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. Developed by Dr. Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom (now Eurotech), in 1999, MQTT's main functional principle is the existence of topics in an intermediate entity named broker. Basically, when a client subscribes or publishes on a certain topic, that topic is registered on the broker. Multiple subscriptions can be made by the same client to different topics, as well as different clients can

subscribe the same topic. This works the same way to publishers. So the clients subscribing the topic X will receive every message published under that topic. Concluding, the broker and MQTT act as a simple, common interface for everything to connect to, as said in the website of one of the available brokers.

8.3.2 MQTT architecture :

The MQTT consists of three components, subscribers, publishers, and brokers. Figure 2.4 illustrates the architecture of MQTT. To communicate, the device will register as a subscriber to a specific topic of its interest in the broker. When the publishers publish to the topic, the broker delivers the information to one or more subscribers Many applications use MQTT e.g., health care, monitoring, energy meter, Facebook notification. Smart city, the MQTT represents one of the appropriate messaging protocol for IoT and M2M communications because provides routing for small, cheap, limited power and memory devices that belong to vulnerable and low bandwidth networks.

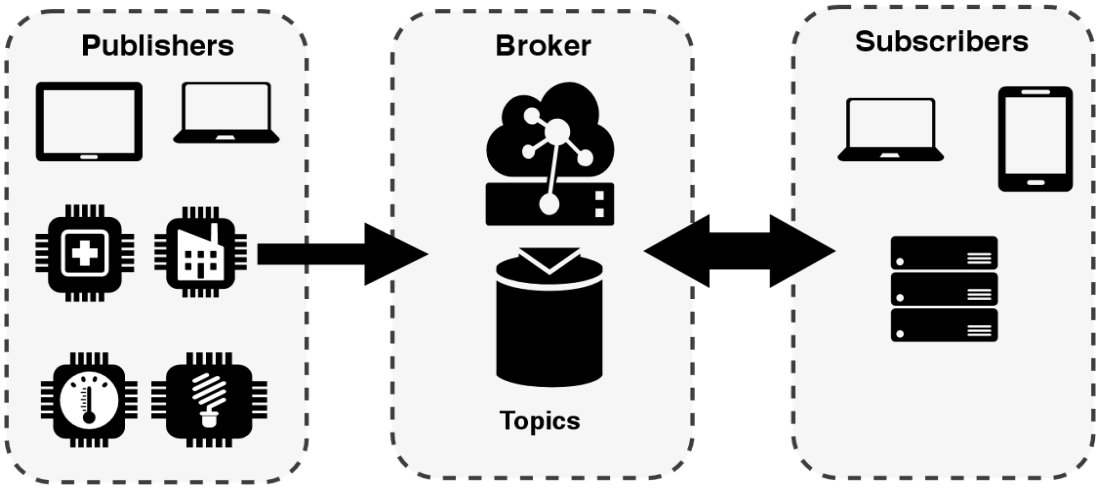

Figure 2. 3: MQTT architecture. [23]

8.3.3 Mqtt Quality of Services (QoS):

There are different configuration of quality of services (QoS) in MQTT this protocol provides another layer of reliability using three levels of QoS named as QoS level 0, QoS level 1 and QoS level 2.

8.3.3.1 QoS 0 (At most once, fire and forget):

QoS 0 the message is sent using the best effort on the TCP/IP network, the answer is not expected and the message is not sent again. The message can arrive at the server or not. The sender sends a PUBLISH packet as presented in **[Figure 2.4]**. [23]

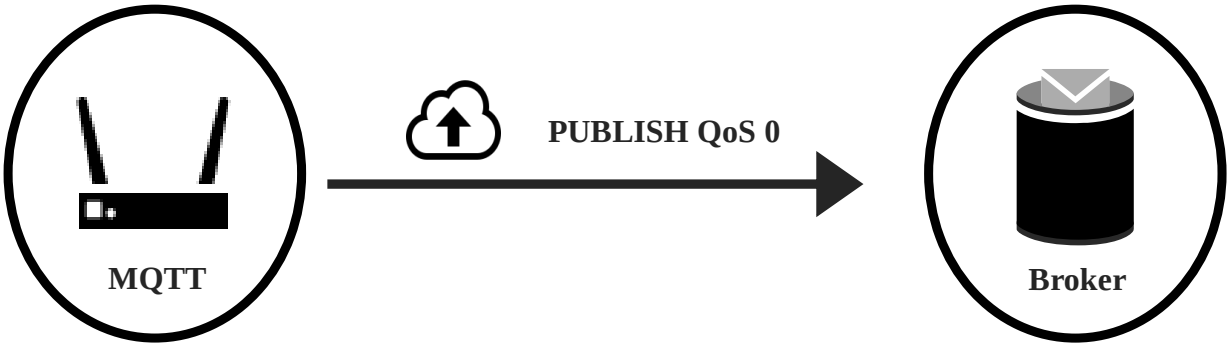

Figure 2. 4: QoS level 0. [23]

8.3.3.2 QoS 1(At least once):

QoS 1 it is the default mode of message transfer The message arrives at least once to the receiver, to ensure delivery at least once. If a failure is identified, there may be a failure on the communication link, or the message is not received after a specified period of time, the message will be resent by the sender. It allows the recipient to receive the message multiple times. After the message is processed, it is deleted from the receiver. The sender using QoS 1 sends a PUBLISH packet containing the Packet Identifier. The Publish packet retains a state of unacknowledged until the sender, receives the PUBACK packet from the broker/receiver. **[Figure 2.5]** shows message exchange between a publisher and broker using QoS 1 technique. After deleting the message the receiver sends the acknowledgment to the sender. The same occurs with the sender after receiving the acknowledgment from the receiver. Both sender and receiver delete the message after the communication. [23]

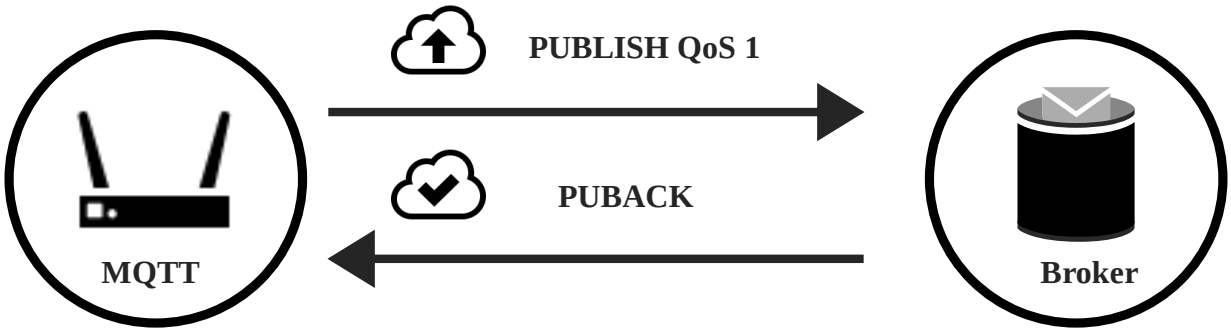

Figure 2. 5: QoS level 1. [23]

8.3.3.3 QoS 2 (Exactly once):

Qos 2 Duplicate messages and losses are not acceptable. It increases the network traffic, but it is acceptable because the QoS 2 is used to send relevant messages, it is the highest Quality of Service This technique guarantees that a message is received once, when the receiver confirms the message has arrived. The QoS 2 PUBLISH packet involves two-step process. The PUBLISH packet is treated as unacknowledged until the sender receives the corresponding PUBREC packet from the receiver. The sender sends a PUBREL packet and waits from the receiver, the corresponding PUBCOMP packet. The PUBREL packet is treated as unacknowledged until the PUBCOMP packet reception. **[Figure 2.7]** represents messages exchange between a publisher and broker using QoS 2 technique.

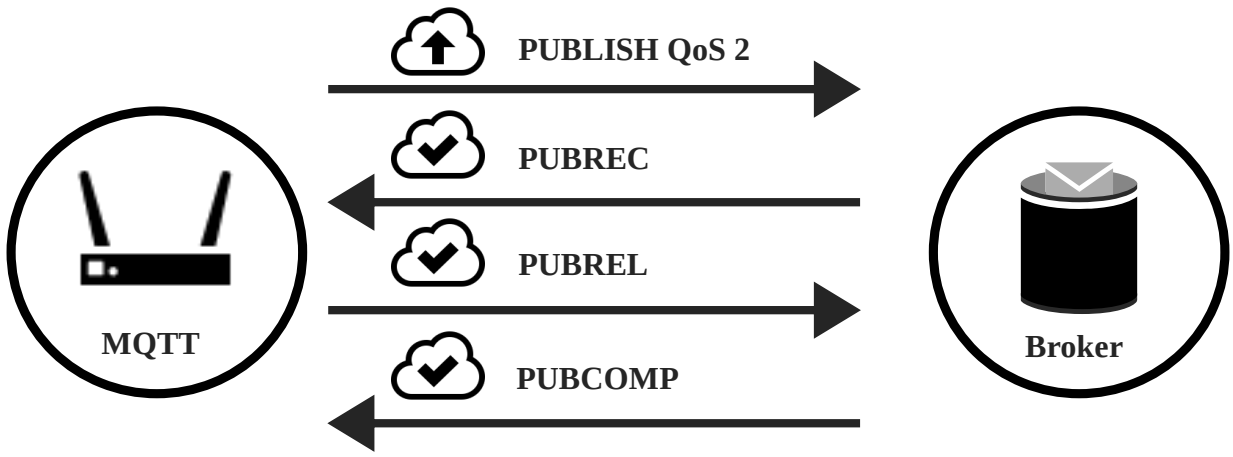

Figure 2. 6: QoS level 2. [23]

8.4 Message Queuing Telemetry Transport for Sensor Network (MQTT_SN):

8.4.1 What is an MQTT-SN Protocol:

The Message Queuing Telemetry Transport for Sensor Network (MQTT-SN) protocol was developed specially for Wireless Sensor Networks (WSNs), normally made up of low cost and easy developed environments. Typically, a WSN has a large number of sensors and actuators, from different device types, that present a limited amount of storage and processing capabilities. The devices are developed to detect and notify events through wireless links, used habitually in monitoring environment, traffic and building management, battlefield surveillance, and home automation. In my project we will focus at this protocol with some modifications in packets management and its structure.

8.4.2 MQTT-SN architecture :

The architecture of the MQTT-SN is illustrated in Figure 6. The classic architecture is composed of the MQTT-SN client (publisher), the MQTT-SN Gateway (GW), and the MQTTSN forwarder. The MQTT-SN protocol transfer messages between an MQTT-SN client and a broker using an MQTT-SN GW as the middleware. The function of an MQTT-SN GW is to translate messages from MQTT to MQTT-SN or vice-versa, if it is a stand-alone topology.

The MQTT-SN Forwarder is configured when the MQTT-SN GW is not present in the same network. [24]

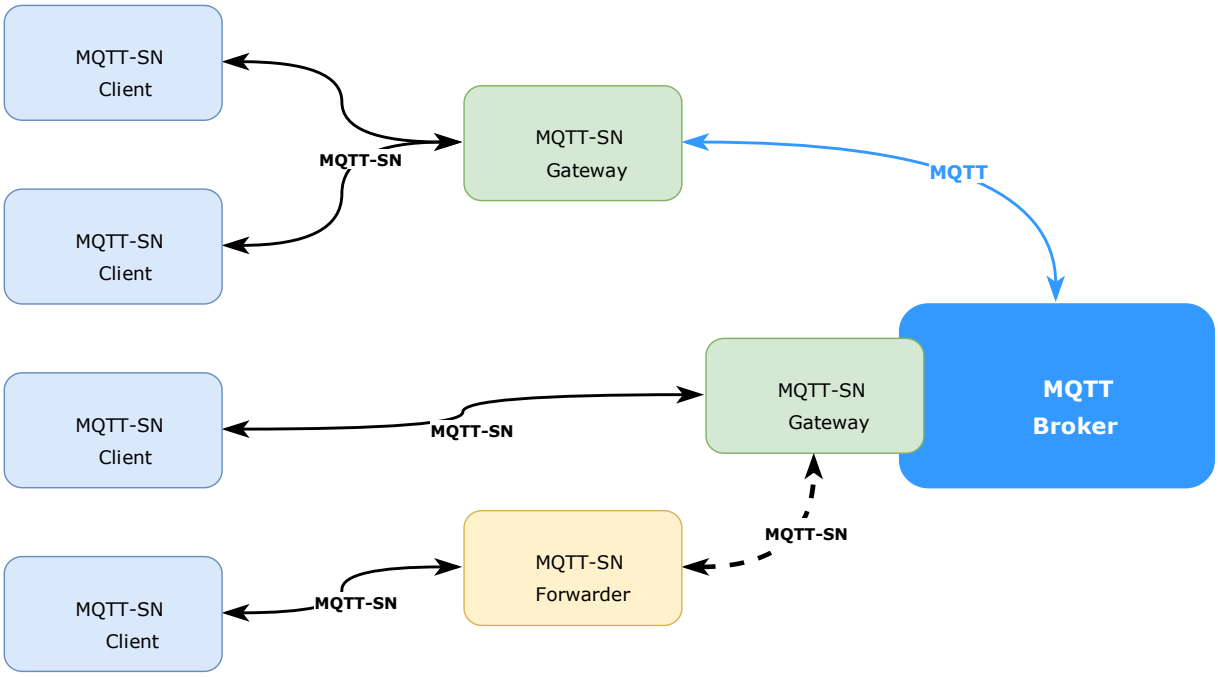

Figure 2. 7: MQTT-SN architecture. [24]

To access a MQTT-SN GW that is not attached in the same network topology is used a MQTT-SN forwarder. The forwarder encapsulates the MQTT-SN frames when receives them from the gateway and sends to the clients. There are two types of GWs to translate frames between MQTT and MQTT-SN, and vice versa. They are named Transparent and Aggregating GWs. The Transparent and Aggregating Gateways are shows in **[Figure 2.8]**.

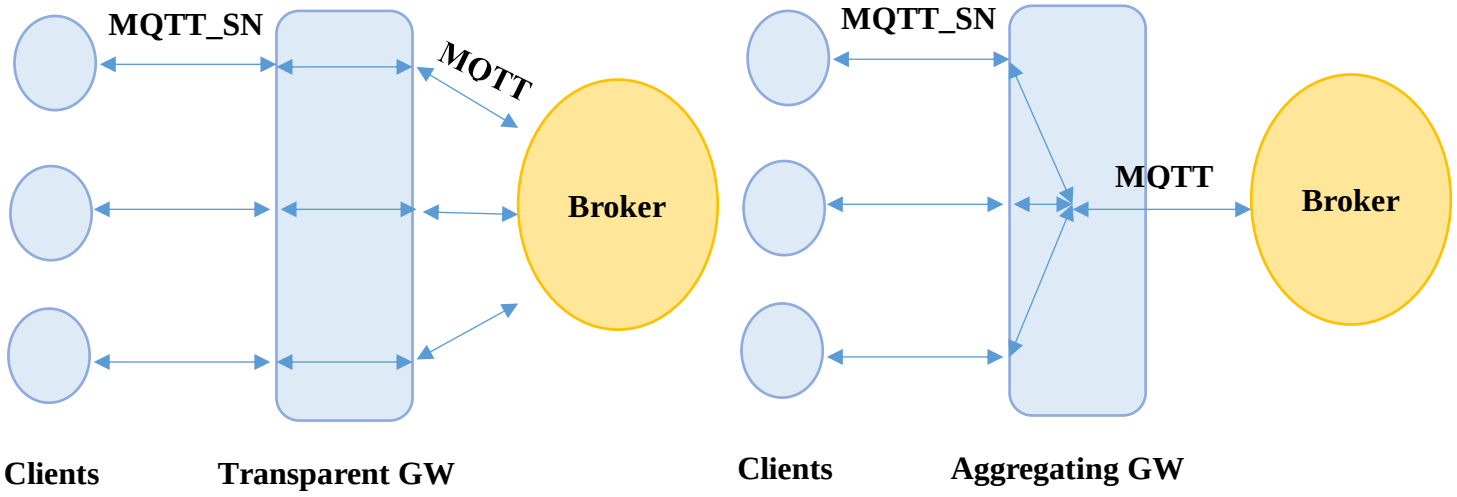

Figure 2. 8: The Transparent and Aggregating. [24]

8.4.3 MQTT_SN Message Formats :

- **General Message Format :**

The format general are shown in **[Table 2.2].**

Table 2. 2: General Message Format .[24]

Message Header	Message Variable Part
(2 or 4 octets)	(n octets)

A MQTT-SN message consists of two parts: a 2- or 4-octet long header and an optional variable part. The header is always present and contains the same fields.

- **Message Header :**

The format of the message header is illustrated in in **[Table 2.3].**

Table 2. 3: Message Header.[24]

Length	Message Type
(1 or 3 octets)	(1 octet)

- **Message Type:**

The Message Type field is 1-octet long and specifies the message type. It shall be set to one of the values shows in in **[Table 2.4].**

Table 2. 4: Value of The Message Type Field. [24]

Message Type Field Value	Message Type	Message Type Field Value	Message Type
0x00	**ADVERTISE**	0x01	**SEARCHGW**
0x02	**GWINFO**	0x03	**Reserved**
0x04	**CONNECT**	0x05	**CONNACK**
0x06	**WILLTOPICREQ**	0x07	**WILLTOPIC**
0x08	**WILLMSGREQ**	0x09	**WILLMSG**
0x0A	**REGISTER**	0x0B	**REGACK**
0x0C	**PUBLISH**	0x0D	**PUBACK**
0x0E	**PUBCOMP**	0x0F	**PUBREC**
0x10	**PUBREL**	0x11	**Reserved**
0x12	**SUBSCRIBE**	0x13	**SUBACK**
0x14	**UNSUBSCRIBE**	0x15	**UNSUBACK**
0x16	**PINGREQ**	0x17	**PINGRESP**
0x18	**DISCONNECT**	0x19	**reserved**
0x1A	**WILLTOPICUPD**	0x1B	**WILLTOPICRESP**
0x1C	**WILLMSGUPD**	0x1D	**WILLMSGRESP**
0x1E-0xFD	**Reserved**	0xFE	**Encapsulated message**
0xFF	**Reserved**		

❖ **Message Variable Part:**

The content of the message variable part depends on the type of the message. The following fields are defined for the message variable part. [24]

ClientId: As with MQTT, the ClientId field has a variable length and contains a 1-23 character long string that uniquely identifies the client to the server.

Data: The Data field corresponds to payload of an MQTT PUBLISH message. It has a variable length and contains the application data that is being published

Duration: The Duration field is 2-octet long and specifies the duration of a time period in seconds. The maximum value that can be encoded is approximately 18 hours.

Flag:

Table 2. 5: Flags Field. [24]

DUP	QoS	Retain	Will	CleanSession	TopicIdType
(bit 7)	(6,5)	(4)	(3)	(2)	(1,0)

The Flags field is 1-octet and contains the following **flag [see Table 2.5]**:

- **DUP**: same meaning as with MQTT, i.e. set to "0" if message is sent for the first time; set to "1" if retransmitted (only relevant within PUBLISH messages);
- **QoS**: meaning as with MQTT for QoS level 0, 1, and 2; set to "0b00" for QoS level 0, "0b01" for QoS level 1, "0b10" for QoS level 2, and "0b11" for new QoS level -1 (only relevant within PUBLISH messages sent by a client);
- **Retain**: same meaning as with MQTT (only relevant within PUBLISH messages).
- **Will**: if set, indicates that client is asking for Will topic and Will message prompting (only relevant within CONNECT message);
- **CleanSession**: same meaning as with MQTT, however extended for Will topic and Will message (only relevant within CONNECT message);
- **TopicIdType**: indicates whether the field TopicId or TopicName included in this message contains a normal topic id (set to "0b00"), a pre-defined topic id (set to "0b01"), or a short topic name (set to "0b10"). The value "0b11" is reserved.**[Table 2.4]**
- **GwAdd:** The GwAdd field has a variable length and contains the address of a GW. Its depends on the network over which MQTT-SN operates and is indicated in the first octet of this field. For example, in a ZigBee network the network address is 2-octet long.
- **GwId**: The GwId field is 1-octet long and uniquely identifies a gateway.
- **MsgId**: The MessageId field is 2-octet long and corresponds to the MQTT 'Message ID' parameter. It allows the sender to match a message with its corresponding acknowledgment.
- **ProtocolId** : The ProtocolId is 1-octet long. It is only present in a CONNECT message and corresponds to the MQTT 'protocol name' and 'protocol version'. It is coded 0x01. All other values are reserved.
- **Radius**: The Radius field is 1-octet long and indicates the value of the broadcast radius. The value 0x00 means "broadcast to all nodes in the network".
- **ReturnCode**: The value and meaning of the 1-octet long ReturnCode field is shown in **[Table 2.6]**.

Table 2. 6: Return Code Values. [24]

ReturnCode Value	Meaning
0x00	Accepted
0x01	Rejected: congestion
0x02	Rejected: invalid topic ID
0x03	Rejected: not supported
0x04 - 0Xff	Reserved

- **TopicId:** The TopicId field is 2-octet long and contains the value of the topic id. The values "0x0000" and "0xFFFF" are reserved and therefore should not be used.
- **TopicName:** The TopicName field has a variable length and contains an UTF8-encoded string that specifies the topic name.
- **WillMsg:** The WillMsg field has a variable length and contains the Will message.
- **WillTopic:** The WillTopic field has a variable length and contains the Will topic name.

❖ **The main Message exchange in MQTT-SN:**

➢ **SEARCHGW :**

Table 2. 7: SEARSHGW Message. [24]

Length	MsgType	Flags	TopicId	MsgId	ReturnCode
(octet 0)	(1)	(2)	(3,4)	(5,6)	(7)

The SEARCHGW message is broadcasted by a client when it searches for a GW. The broadcast radius of the SEARCHGW is limited and depends on the density of the clients deployment. The format of a SEARCHGW message is illustrated in **[Table 2.7].**

- Length and Message Type: **[See Table 2.4].**
- Radius: the broadcast radius of this message.

- **GWINFO:**

Table 2. 8: GWINFO Message. [24]

Length	MsgType	GwId	GwAdd*
(octet 0)	(1)	(2)	(3:n)

The GWINFO message is sent as response to a SEARCHGW message using the broadcast service of the underlying layer, with the radius as indicated in the SEARCHGW message. If sent by a GW, it contains only the id of the sending GW; otherwise, if sent by a client, it also includes the address of the GW, **[See Table 2.8]:**

- Length and MessageType: **[See Table 2.4].**
- GwId: the id of a GW.
- GwAdd: address of the indicated GW; optional, only included if message is sent by a client.

- **CONNECT :**

Table 2. 9: CONNECT Message. [24]

Length	MsgType	Flags	ProtocolId	Duration	ClientId
(octet 0)	(1)	(2)	(3)	(4,5)	(6:n)

The CONNECT message is sent by a client to setup a connection. Its format is shown in **[Table 2.9]**:

- Length and MsgType
- Flags:
  - DUP, QoS, Retain, TopicIdType: not used.
  - Will: if set, indicates that client is requesting for Will topic and Will message prompting;
  - CleanSession: same meaning as with MQTT, however extended for Will topic and Will message.
- ProtocolId: corresponds to the “Protocol Name” and “Protocol Version” of the MQTT CONNECT message.

- Duration: same as with MQTT, contains the value of the Keep Alive timer.
- ClientId: same as with MQTT, contains the client id which is a 1-23 character long string, which uniquely identifies the client to the server.

➢ **CONNACK :**

Table 2. 10: CONNACK Message. [24]

Length	MsgType	ReturnCode
(octet 0)	(1)	(2)

The CONNACK message is sent by the server in response to a connection request from a client. Its format is shown in **[Table 2.10]:**

- Length and MsgType: **[See Table 2.4]:**.
- ReturnCode: encoded according to **[Table 2.6]:**.

➢ **PUBLISH :**

Table 2. 11: PUBLISH Message. [24]

Length	MsgType	Flags	TopicId	MsgId	Data
(octet 0)	(1)	(2)	(3-4)	(5-6)	(7:n)

This message is used by both clients and gateways to publish data for a certain topic. Its format is illustrated in **[Table 2.11]:**

- Length and MsgType: **[See Table 2. 4]**.
- Flags:
  - DUP: same as MQTT, indicates whether message is sent for the first time or not.
  - QoS: same as MQTT, contains the QoS level for this PUBLISH message.
  - Retain: same as MQTT, contains the Retain flag.
  - Will: not used.

- CleanSession: not used
- TopicIdType: indicates the type of the topic id contained in the *TopicId* field.

- TopicId: contains the topic id value or the short topic name for which the data is published.
- MsgId: same meaning as the MQTT "Message ID"; only relevant in case of QoS levels 1 and 2, otherwise coded 0x0000.
- Data: the published data.

➢ **PUBACK:**

Table 2. 12: PUBACK Message. [24]

Length	MsgType	TopicId	MsgId	ReturnCode
(octet 0)	(1)	(2,3)	(4,5)	(6)

The PUBACK message is sent by a gateway or a client as an acknowledgment to the receipt and processing of a PUBLISH message in case of QoS levels 1 or 2. It can also be sent as response to a PUBLISH message in case of an error; the error reason is then indicated in the *ReturnCode* field. Its format is illustrated in **[Table 2.12]:**

- Length and MsgType: **[See Table 2.6]**.
- TopicId: same value the one contained in the corresponding PUBLISH message.
- MsgId: same value as the one contained in the corresponding PUBLISH message.
- ReturnCode: "accepted", or rejection reason.

➢ **SUBSCRIBE :**

Table 2. 13: SUBSCRIBE Message.[24]

Length	MsgType	Flags	MsgId	Topic_Name or TopicId

(octet 0)	(1)	(2)	(3-4)	(5:n) or (5-6)

The SUBSCRIBE message is used by a client to subscribe to a certain topic name. Its format is illustrated in **[Table 2.13]:**

- Length and MsgType: **[See Table 2.4]:**.
- Flags:
  - DUP: same as MQTT, indicates whether message is sent for first time or not.
  - QoS: same as MQTT, contains the requested QoS level for this topic.
  - Retain: not used
  - Will: not used
  - CleanSession: not used
  - TopicIdType: indicates the type of information included at the end of the message, namely "0b00" topic name, "0b01" pre-defined topic id, "0b10" short topic name, and "0b11" reserved.
- MsgId: should be coded such that it can be used to identify the corresponding SUBACK message.
- TopicName or TopicId: contains topic name, topic id, or short topic name as indicated in the TopicIdType field.

➢ **SUBACK :**

Table 2. 14: SUBACK Message.[24]

Length	MsgType	Flags	TopicId	MsgId	ReturnCode
(octet 0)	(1)	(2)	(3,4)	(5,6)	(7)

The SUBACK message is sent by a gateway to a client as an acknowledgment to the receipt and processing of a SUBSCRIBE message. Its format is illustrated in **[Table 2.14]**:

- Length and MsgType: **[Table 2.4]**.
- Flags:

- DUP: not used.
- QoS: same as MQTT, contains the granted QoS level.
- Retain: not used.
- Will: not used.
- CleanSession: not used.
- TopicIdType: not used.

- TopicId: in case of “accepted” the value that will be used as topic id by the gateway when sending PUBLISH messages to the client (not relevant in case of subscriptions to a short topic name or to a topic name which contains wildcard characters).
- MsgId: same value as the one contained in the corresponding SUBSCRIBE message.
- ReturnCode: “accepted”, or rejection reason.

8.4.4 Features of the MQTT-SN:

- To work with short message length and limited transmission bandwidth in a wireless network, the "topic name" in PUBLISH message is replaced for a two-byte “topic-id”. The clients register their topic name in the server/gateway and obtain the corresponding topic id;
- “Pre-defined” topic ids and “short” topic names are introduced. The short topics presents a length of two octets, so they are shorter enough for being carried simultaneously with data in PUBLISH messages;
- The discovery procedure is used to assist clients that do not know the server/gateway’s address to discover the network address;
- A new offline keep-alive procedure is defined to support sleeping clients. The battery-operated devices can go to a sleeping state; all messages designed to them are buffered at the server/gateway and delivered later when they wake up.

8.5 The differences between the MQTT and MQTT-SN protocols:

Table 2. 15: MQTT VS MQTT-SN. [24]

Characteristic	MQTT	MQTT-SN
Transport	TCP	UDP / Other(ZigBee)
Replace topic for an Id	-	V
Pre-defined topics Id	-	V
Discovery procedures	-	V
Support for sleeping clients	-	V

8.6 The publically accessible MQTT brokers:

The publically accessible MQTT brokers shown in **[Table 2.16].**

Table 2. 16: The publically accessible MQTT brokers.

Type of MQTT brokers	
Mosquitto	
Address	www.cloudmqtt.com (Note: actual host varies, see dashboard)
Port	The standard port is 1883.
Info	designed by Andy Stanford-Clark (IBM) Requirs signup/username and password provides lightweight methods of carrying out messaging MQTT websocket support for web browsers is provided by the JavaScript MQTT client. The mosquitto broker supports MQTT v5 in addition to MQTT v3.11.
VerneMQ	
Address	mqtt.teserakt.io
Port	1883 (TCP)
Info	free, open, not for production (several performance limits, etc.), operated by Teserakt
HiveMQ	
Address	broker.hivemq.com
Port	1883, 8000 (WebSockets)
Info	Free to write an MQTT client that connects with this broker. offer an open source edition of HiveMQ for developers requiring a basic MQTT broker
RSMB(Really Small Message Broker) for MQTT-SN	
Address	http://git.eclipse.org/c/mosquitto/org.eclipse.mosquitto.rsmb.git/
Port	1883, 1884
Info	RSMB is a server implementation of the MQTT and MQTT-SN protocols. Released onto IBM alphaWorks in 2008. RSMB has been tested on Linux, Mac OS and Windows.

9 Conclusion:

The Publish/Subscribe (pub/sub) paradigm has been used as a powerful communication protocol, to develop a wide range of distributed applications. In this paradigm, communicating entities are loosely coupled in terms of time, space, and synchronization; thus, the paradigm can support useful and flexible characteristics, such as anonymity, many-to-many, and asynchronous communications, that are critical for distributed systems.

In next chapter, we present in detail our solution that is based on an opportunistic pub/sub scheme with UAV-based broker, to support dynamic topology for more flexibility and effectiveness, especially in smart city applications.

CHAPTER 3:

Overview on our Solution:

Publish/Subscribe Communications with Opportunistic- UAV-based broker in smart city context.

1 Introduction :

A smart city applications is a big project in IoT technology where is there a large number of IoT devices will be communicate to each other, a big data will be transmitted for storage and analysis on top of that ; all them working automatically ,in this case, some problems will arise such as a high latency , loss messages and delay ..etc. All these metrics can effect the project outcome.

In our project, some factors need to be studied, taking into consideration the quality of services that we get, consummation of energy , scalability and latency .

Publish/Subscribe communication model is a solution to give good results, and they allows us to fix the most of these problems.

2 Opportunistic publish/subscribe communication in edge level:

A pub/sub communication is a lightweight interaction, often used in the wide applications that needs independently communication with short data exchange. The main entity in this model is the broker typically, it is placed in central location like a cloud or fog (cloud based pub/sub or fog based pub/sub) **[Figure 3.1].**

Broker faces some problems as it works on smart city applications .it needs a real time data received in critical cases for example when messages are lose or late. To solve those problems we design an opportunistic pub/sub communication makes broker closer to edge devices (edge based pub/sub **[Figure 3.1]**), to implement this new idea we will use a new IoT devices under development, which is UAVs (unmanned Aerial Vehicles).

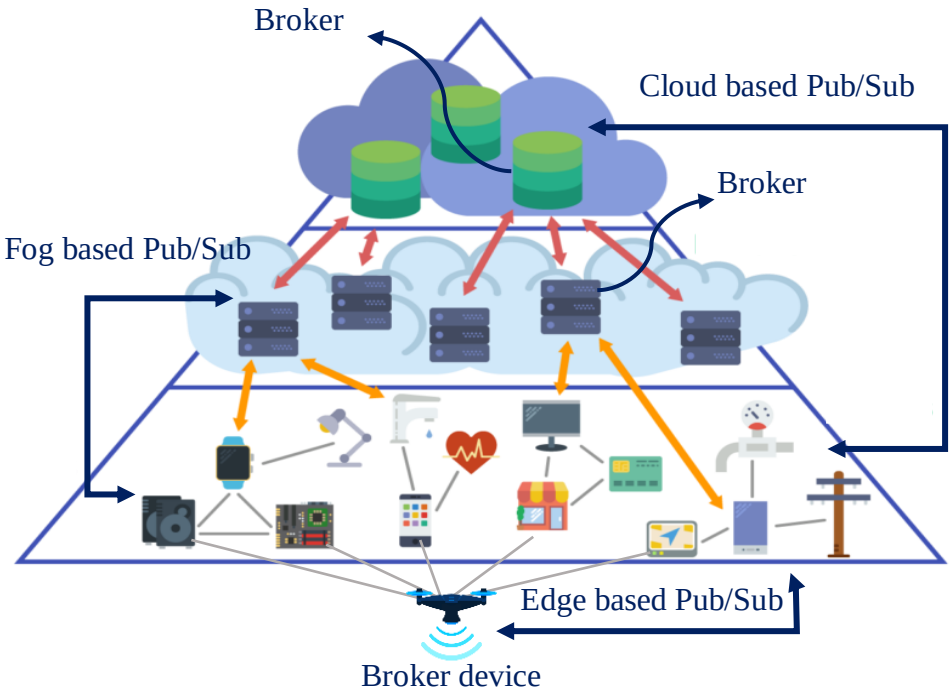

Figure 3. 1: publish/subscribe communication based three layer.

3 The most popular type of UAVs:

Recently, unmanned aerial vehicles (UAVs), or drones, have attracted a lot of attention. Worldwide deployment of these UAVs is expected. Thanks to the high mobility of drones, they can be used to provide many applications, such as service delivery, pollution mitigation, farming, and in the rescue operations.[25]

There are several types of UAVs; we mention the most popular one.

Multi-rotor: Multirotors are UAVs that use more than two rotors with fixed-pitch spinning blades that generate lift. Multirotor aircraft is the simpler rotor mechanics required for flight control.

Fixed-wing: A Fixed-wing UAV is an aircraft that operates without a human pilot on board. Fixed wing UAVs are controlled either remotely by a human operator or autonomously via on board computers**.**

Single rotor: A single-rotor helicopter allows longer blades to enable slower spinning and therefore less energy expended. However, single-rotor helis have significantly more vibration than multi-rotor UAVs. They also pose more danger due to their large blades.

Fixed-wing hybrid: Hybrid VTOL Fixed-Wing UAVs combine the benefits of multi-rotor platforms with fixed-wing drones and transition between the two modes during flight.

In our project, we need drones that operate in wide area that is why we chose the Hexacopter to work on.

The Hexacopter drone is a flying device from Multi-rotor UAV type family; hobbyist's model making and surveillance and aerial photography companies use it. Vertical take-off aircraft with electric motors, it incorporates mechatronic technologies allowing for scheduled flights. Its use for surveillance and aerial photography activities is possible due to its ability to carry a large load.

4 UAV and Broker:

Pub/sub communication use several type of protocols based on broker; among these protocols, there are mqtt and mqtt-sn who work with lightweight brokers like mosquitto for mqtt and rsmb for mqtt-sn. Both can installed in simple devices such as an UAV.

We chose to work by mqtt-sn broker because in smart city context we have a wireless sensor network environment that is why we are going to make some changes in this communication model based on mqtt-sn protocol to work in a mobility state.

5 The main component for create an UAV Broker:

The main component needs for create Uav or drone broker showing in **[Table 3.1].**

Table 3. 1: The main components for making an UAV broker.

Device Name	Information	Weight	Price	Image
Hexacopter frame	Motor to motor length: 800mm	**3000g**	**191.10 $**	

Six Motor Hobbywing X8 FOC With propeller	recommended lipo battery : 12S throttle signal frequency :50 – 500 mhz PWM input signal level: 3.3v/5v propeller 3cm ccw carbon	**180g*6**	**150$*6**	
Six ESC Hobbywing Platinum Pro V3 100A Opto HV	ESC **works under** 5S-12S LiPo Weight **104g** **Price**	**104g*6**	**81.59$*6**	
ArduPilot Mega 2.8 APM Flight Control Board with Protective Case for Multicopter Rc Airplane	an open source, unmanned vehicle Autopilot Software Suite capable of controlling autonomous: • Multirotor drones • Fixed-wing and VTOL aircraft • Helicopters • Ground rovers • Boats • Submarines • Antenna trackers	**37g**	**2.57 $**	
lipo batterie ZDF	Capacity 10000 mah Continuous discharge rate 20c Burst rate 40c Voltage per pack 44.4 v Max voltage per cell 4.2v Cells s12	**2460 g**	**232.74$**	

Raspberry Pi 4 modèle B	ABS housing 3.5 inch fan or touch screen Alimentation 5V 3A type-c heat sink for Raspberry Pi 4 4B carte SD 64 go RAM 8gb	**120g**	**125 $**	
ZigBee Wireless Sniffer Bare Board Packet Protocol Analysis Module CC2531 CC2540 USB	Interface Dongle Capture Packet Module Energy consumption : <20mA (receive); <25mA (transmit)	**15g**	**7.02 $**	
VGE Battery	Pack for Raspberry Pi 4 B, Capacity 4000mAh, Adhesive (Pi 4 (USB-C))	**40.8g**	**23.95 $**	
BerryGPS-IMUv3 -GPS and 10DOF for The Raspberry Pi	GPS and 10DOF for The Raspberry Pi - Accelerometer, Gyroscope, Magnetometer and Barometric/Altitude Sensor	**35.45g**	**53.50$**	
By adding auxiliary tools, the price and weight may reach as follows		**8.5kg**	**2307,95$**	

We can test UAV performance using xcopter calculator website that provide in this link https://www.ecalc.ch/. **[See Figure 3.2].**

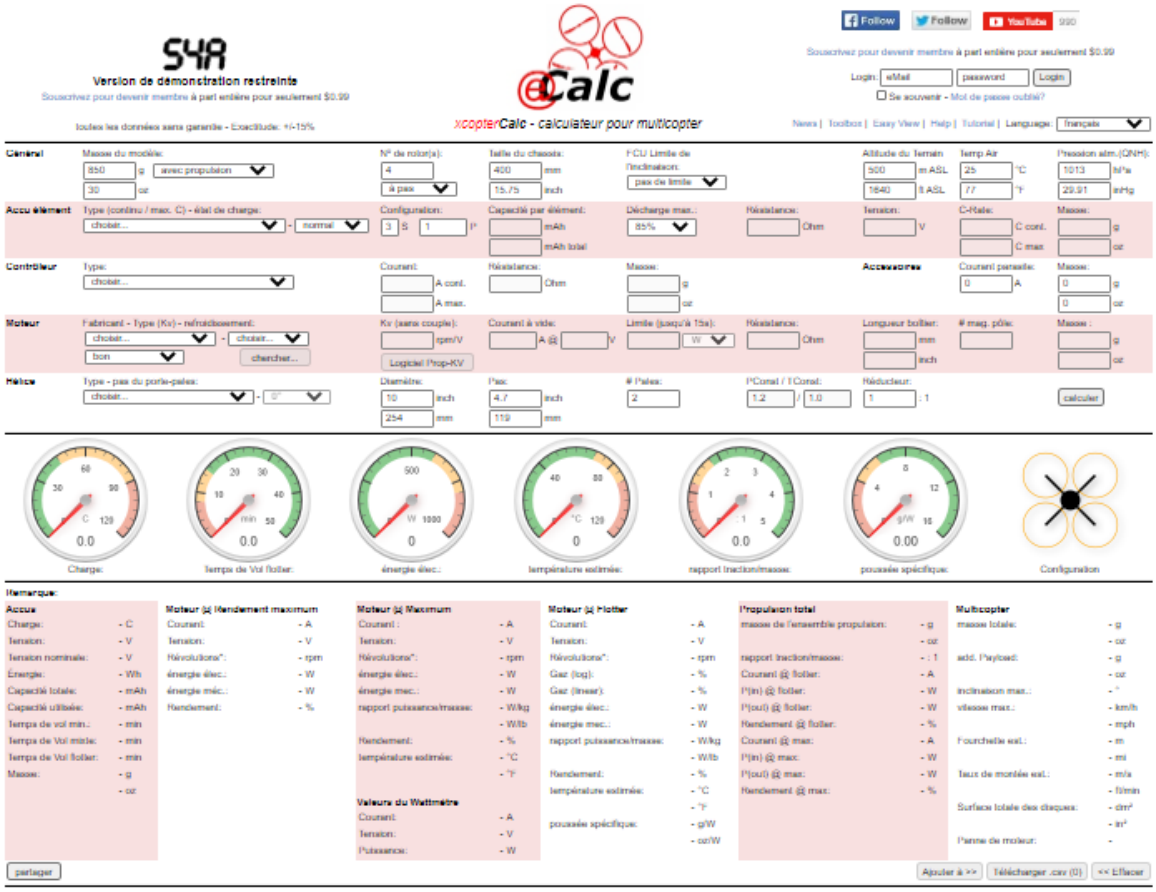

Figure 3. 2: xcopter calculator Website.

6 The hardware architecture of the UAV Broker :

The hardware architecture of the UAV Broker showing in **[Figure 3.3].**

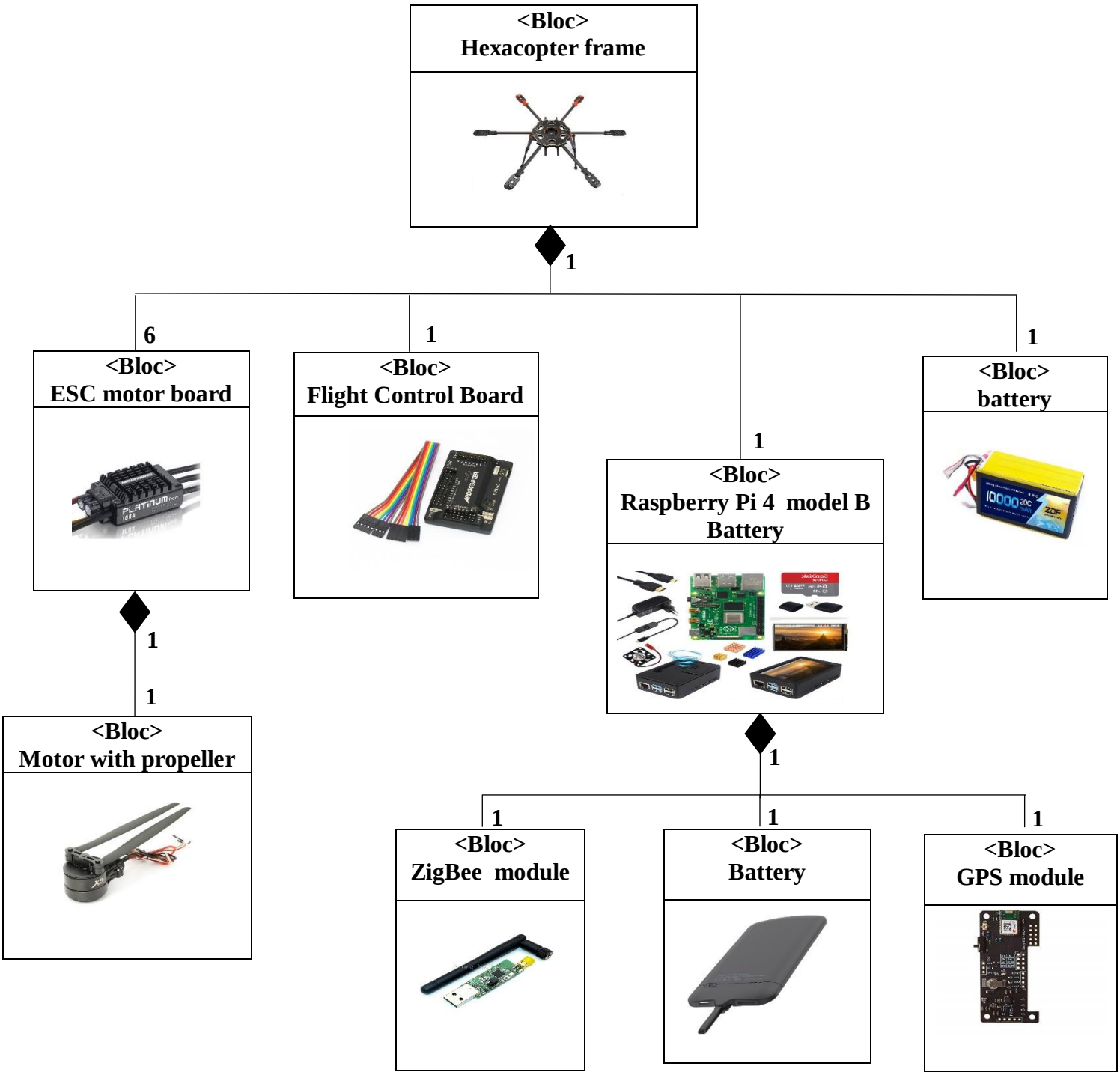

Figure 3. 3: The hardware architecture of the UAV Broker.

7 Our conception:

We will make some changes to the co-publisher connection structure, so thatwe have five elements instead of three **[Figure 3.5]**, two of them are mobile devices and the other three are fixed. We explain each one of them as follows.

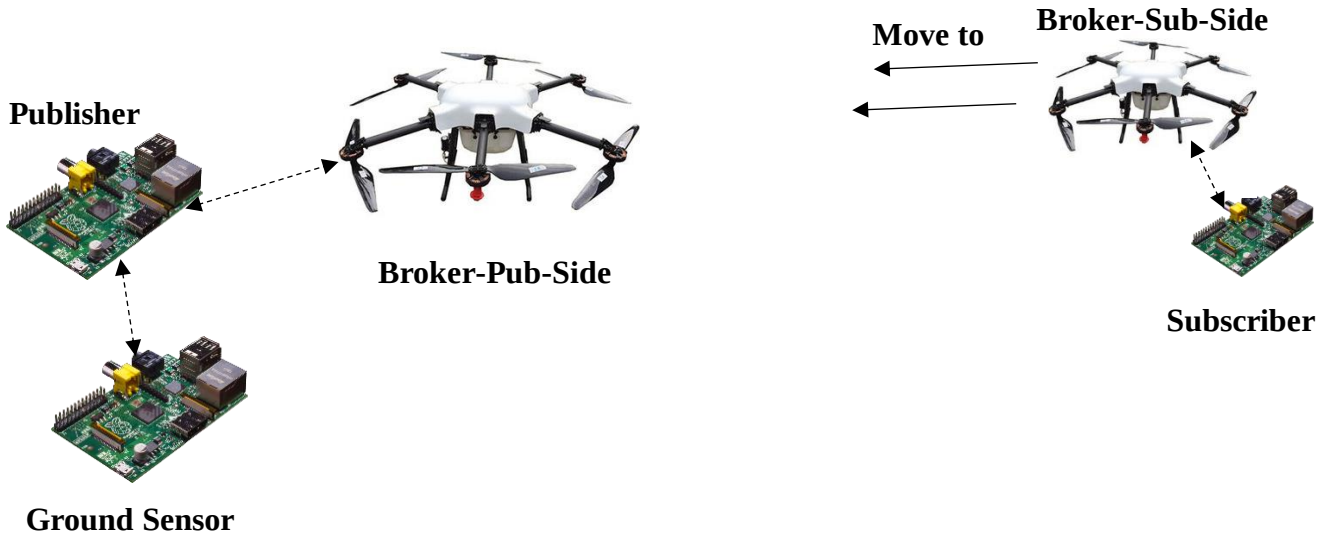

Figure 3. 4: publish /subscribe Architecture based UAV Brokers.

7.1 Ground Sensor:

It is a smart device placed at a fixed location to sense specific events such as temperature, shock, or something like that. This device includes its information, geographic position, sending time and the topic of interest, and then it sends this message to publisher.

7.2 Publisher:

It is a smart device placed near to ground sensors, this device receives all incoming events, and then it tries to search the Broker_pub_side to publish the data collected.

7.3 Broker_pub_side:

It is a drone with high performance that works to cover a specific area for monitoring any topics that come from publishers, this broker support different per-define topics and move autonomously using GPS, this broker saves data in queue of its topics, and waiting for the

Broker_sub_side. This broker supports a specific number of clients that connect in same time, and can monitor these drones by service provider.

7.4 Broker_sub_side:

It is a drone device which does not differ from the Broker_pub_side in terms of manufacturing, but it is deferent in method of work, as it is sent by the subscriber to specific area carrying with him the type of topic to will be collected (support one type of topic).
This broker search Broker_pub_side and subscribe in using its topic then will get the date and move to the next Broker_pub_side to get more, if it finish; it go back and wait for another order from subscriber.

7.5 Subscriber:

It is a smart device that is often located in big centers such as hospitals, security centers, study centers, etc. The subscriber sends the broker-sub-side to collect data as he studies and processes it and makes decisions.

8 MQTT-SN modification:

In our conception, we must change something in Mqtt-sn protocol to become more compatible with the novel communication requirements. We mention the main appended modifications below:

- The broker of Mqtt-sn have pre-defined Topics.
- The broker is devised into tow independent mobile brokers, installed on UAVs.
- The subscriber controls broker-sub-side using another type of messages **[Figure 3.5]**.
- A new entity that added for captured the events also use another type of messages [**Figure 3.7].**
- The port that using represented in 1883, 1884, 1885, 1886.
- In this version modified, we will use QoS (Quality of Service) level 1.
- The topic format is like: sys/topic name/topic content.
- Topic content: contains the sensor id of ground sensor, the key, the time it captures this event. , is the position of this event.

8.1 The Message header:

Just we add just 2 parameters into Message header **[See Table 3.2]**.

Table 3. 2: Mqtt-sn Message header.

Port	Sensor ID	Length	MsgType

8.2 The new messages type are proposed:

Table 3. 3: The new Mqtt-sn Message type.

Message Type Field Value	Message Type
0x1f	**ACTION**
0x21	**ACTIONACK**
0x22	**MISSION**
0x23	**MISSIONACK**
0x24	**END_MISSION**
0x25	**END_MISSIONACK**

8.3 Messages format:

- **ACTION:**

  [Table 3.4] shows the format of ACTION packet.

Table 3. 4: ACTION Message type format.

PORT	ID	Length	ACTION	Topic name	Key	Time	Longitude	Latitude

Topic name: is the type of event that captured them.

Key: is a unique id to identify this event.

Time: is the time it captures this event.

Longitude and Latitude: is the position of this event.

- **ACTIONACK:**

  [Table 3.5] shows the format of ACTIONACK packet.

Table 3. 5: ACTIONACK Message type format.

PORT	ID	Length	ACTIONACK	Key

Key: is the key of event, it is resent to validate the request message (ACTION).

- **MISSION:**

[Table 3.6] shows the format of MISSION packet.

Table 3. 6: MISSION Message type format.

PORT	ID	Length	MISSION	Mission ID	Next Route

Mission ID: is an id to check if there is an UAV in mission.

Next Route: the destination of the road he will take.

- **MISSIONACK**

[Table 3.7] shows the format of MISSIONACK packet.

Table 3. 7: MISSIONACK Message type format.

PORT	ID	Length	MISSION

Topic content: [Table 3.8] shows the Topic content.

Table 3. 8: Topic content List.

sensor id	key	time of event	Longitude	latitude

Sensor id: is the id of the sensor that captured an event.

Key: is a unique id to identify this event.

Time of event: is the time it captures this event.

Longitude and Latitude: is the position of this event.

- **END_MISSION:**

[Table 3.9] shows the format of END_MISSION packet.

Table 3. 9: END_MISSION Message type format.

PORT	ID	Length	END_MISSION	Mission ID	Topic name	Topic content

Mission ID: is an id to check if there is a UAV in mission.

Next Route: the destination of the road he will take.

- **END_MISSIONACK:**

[Table 3.10] shows the format of END_MISSIONACK packet.

Table 3. 10: END_MISSIONACK Message type format.

PORT	ID	Length	END_MISSIONACK

- **SERCHGW and GWINFO and CONNECT and CONACK and PUBLISH and PUBACK and SUBSCRIBE:**

All packets they have not changed **[Table 2.7], [Table 2.8], [Table 2.9],**

[Table 2. 10], [Table 2.11], [Table 2.12], [Table 2.13]; Respectively**.**

- **SUBACK :**

[Table 3. 11] shows the format of SUBACK packet.

Table 3. 11: SUBACK Message type format.

PORT	SensorID	Length	MsgType	Flags	Topic name	MsgId	Next route	Last content	Topic content
Return Code									

Last content: this variable field set to 1 if the last topic content and 0 if not.

9 The UML diagram of our application:

In this item, we will explain the sequence diagram of our application this diagram should be devised into 4-sub diagram to make it easier to understand. All sensors work in loop mode and start in the same time.

9.1 Ground Sensor and Publisher sequence diagram:

[Figure 3.5] shows the interaction between Ground Sensor and Publisher, under this figure we will explain the detail.

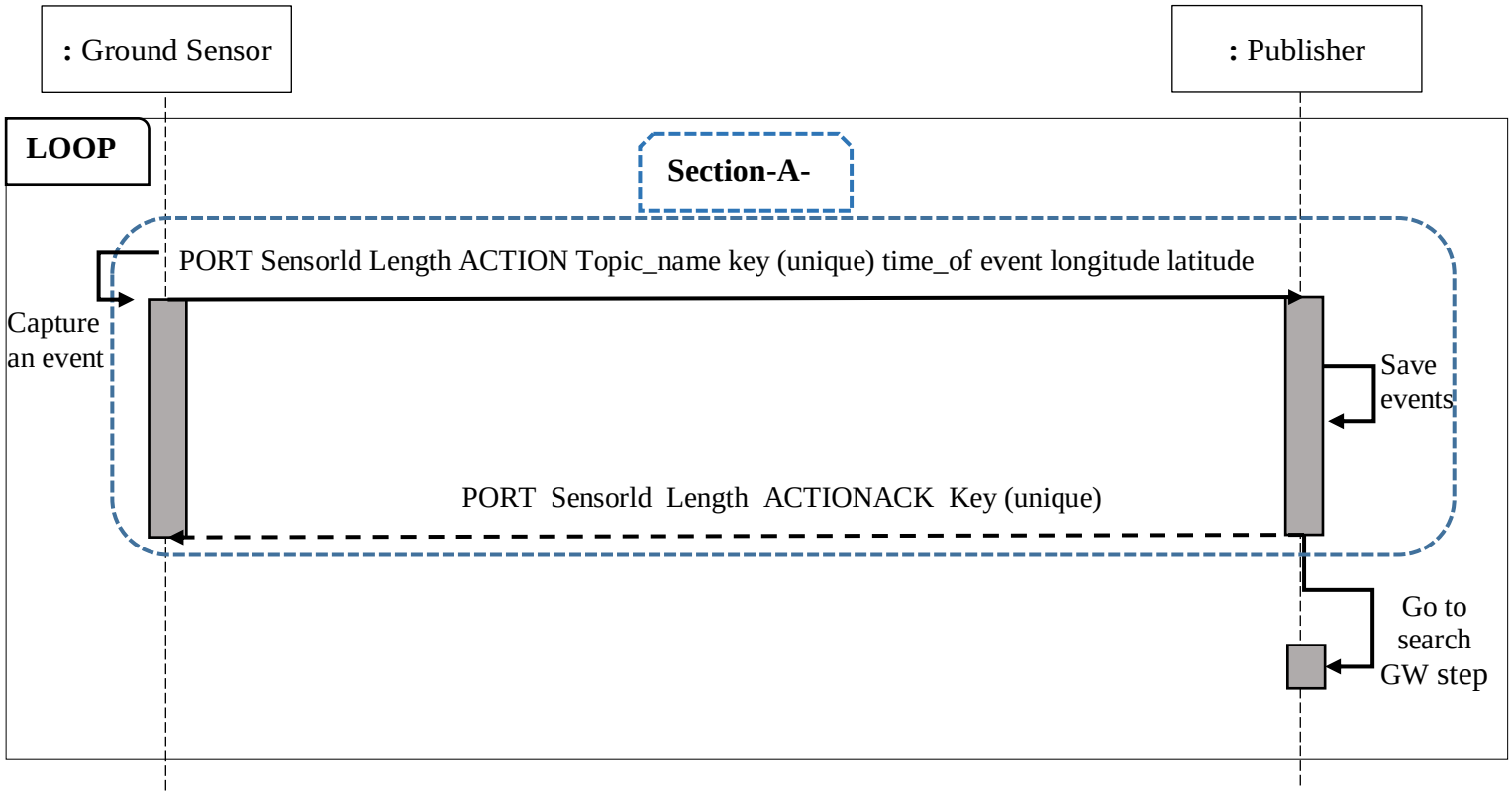

Figure 3. 5: Ground Sensor and Publisher sequence diagram.

- The Ground Sensor wait for an event in sleep mode.
- Then publisher wait 500 ms listening to two ports. A first port 1883 to receive events from the ground sensor that are represented in ACTION message type, and a second port 1884 to publish topics content in the broker pub side.

- Ground sensor sends an ACTION message type to publisher if it captures an event then it waits 1sec for its response presented in ACTIONACK message type, ground sensor try to send requests, if it does not receive any response in 60 sec the ground sensor will change the mode to sleep and wait for another event.
- If publisher receives an ACTION message type (event), it will save the event in its queue then it sends the response presented in ACTIONACK message type.
- Then, publisher will search a gateway (Broker_Sub_Side) by using a SEARSHGW message type to find it and publish the events collected.

9.2 Publisher and Broker_Pub_Side sequence diagram:

[Figure 3.6] shows the interaction between Publisher and Broker_Pub_Side**,** under this figure we will explain the detail.

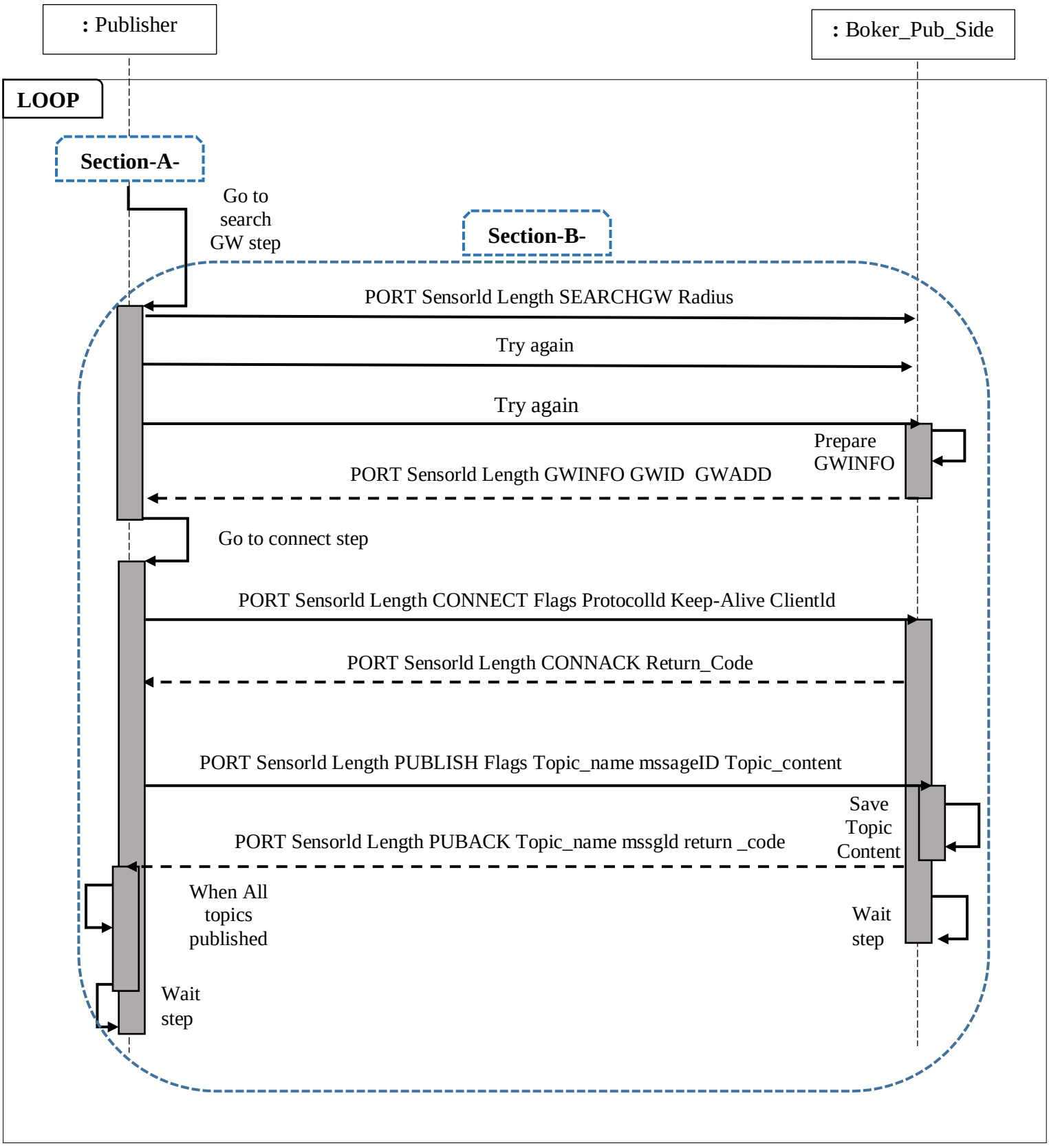

Figure 3. 6: Publisher and Broker_Sub_Side sequence diagram.

- The broker_pub_side wait 500 ms listening to two ports. A first port 1884 to receive the publish messages (or PUBLISH message type) from the publisher and a second port 1885 to communicate with broker_sub_side.
- When publisher captures an event **section-A- [figure 3.5]** it will search a gateway in the port 1884 using SEARSHGW message type, this message stay sending for 60 sec in broadcast mode if there are no response.
- When broker_pub_side receives SEARSHGW message type, it will sends its information to publisher using GWINFO message type.
- The publisher receives this information and sends 4 CONNECT message type to open a connection with broker_pub_side.
- The broker_pub_side receives the CONNECT message type and open a connection with publisher in estimate time, then broker_pub_side sends response represented in CONNACK message type to confirm the connection.
- Publisher receives CONNACK message type then it starts to publish all the list of event in the topic that it wants in broker_pub_side using PUBLISH message type. Any event published the publisher are marked, when it receives the response (using pub_flag variable to facilitate this operation).
- The broker_pub_side receives these events and saves them in the right topic, then it sends the response using PUBACK message type then it closes the connection.

- When publisher receives PUBACK message type it will find in its queue, if there are unpublished event.

- If all of them are published, it will go to repeat the same steps.
- Both of sensors use the port number 1884 for communicate to each other.

9.3 Broker_Pub_Side and Broker_Sub_Side sequence diagram:

[Figure 3.7] shows the interaction between Broker_Pub_Side and Broker_Sub_Side,

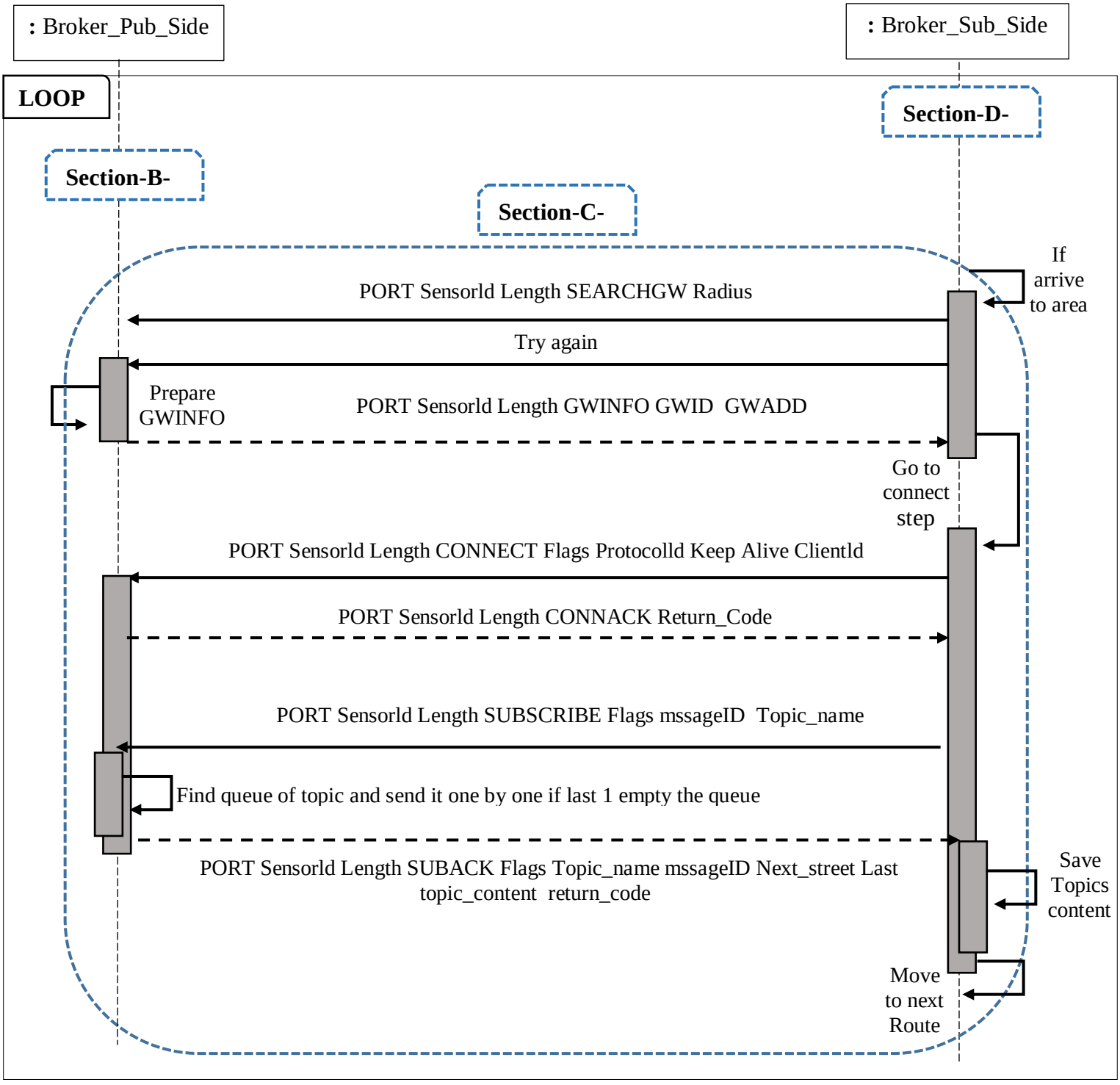

Figure 3. 7: Broker_Pub_Side and Broker_Sub_Side sequence diagram.

- The Broker_Sub_Side wait 500ms listening to two ports, A first port 1885 to communicate with Broker_Pub_Side and a second port 1886 to communicate with Subscriber.
- The Broker_Sub_Side is sent by Subscriber to broker-pub-side **Section-D-[figure 3.8]** to collect the data.
- When Broker_Sub_Side arrives, it searches for Broker_Pub_Side, if it find it, it will be connected with it to see a specific topics.
- To get the topics content needed, it will send a SUBSCRIBE message type contain the name of the wanted topic.
- The Broker_Pub_Side receives a SUBSCRIBE message type and gets its topic name, then it will find it in its queue. The queue of this Broker_Pub_Side is represented in the following: The first level have a list, which contains per-defined topic name and a variable to determine that there is a new in the queue, and variable in which there is a length of queue. Each topic has its own per-defined queue.
- The Broker_Pub_Side is looking in queue of the topic that the broker_sub_side want, if it exists it will send it the queue and the next street; using SUBACK message type. When it finishes, it empties the queue.
- The Broker_Sub_Side receives a SUBACK message type one by one and it saves them in its queue, then it moves to the next street when it finishes (using a variable last send it with SUBACK).
- When Broker_Sub_Side finishes the mission, it will return to starting location.

9.4 Subscriber and Broker_Sub_Side sequence diagram:

[Figure 3.8] shows the interaction between Subscriber and Broker_Sub_Side.

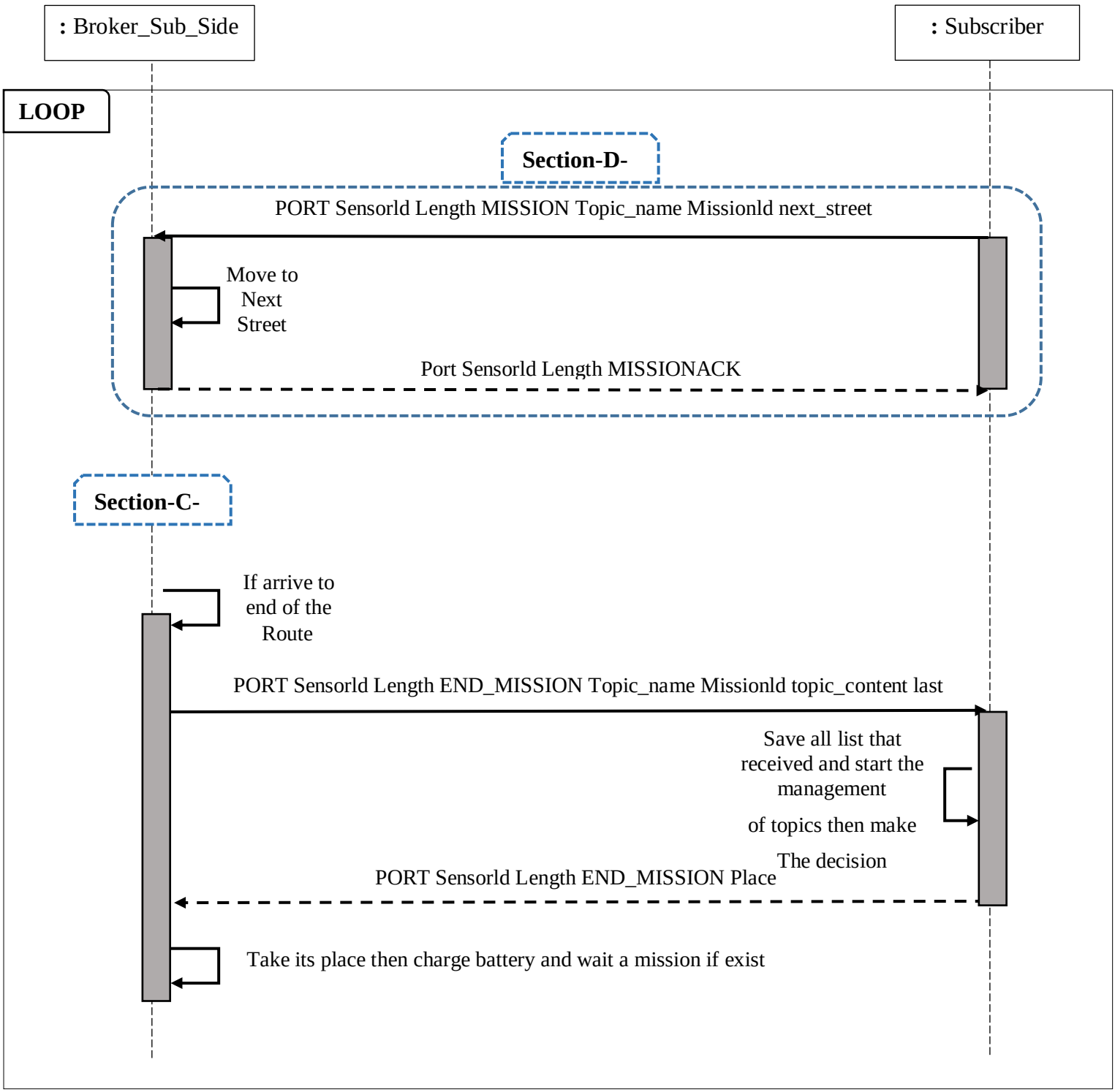

Figure 3. 8: Subscriber and Broker_Sub_Side sequence diagram.

- The Subscriber wait for 500 ms to receive any message coming, and in specific time, it sends Broker_Sub_Side to any street to collect the data.
- The Subscriber triggers the broker by sending a MISSION message type contain the topic name, the street wanted, and a unique missionID variable for more security.
- The Broker_Sub_Side receives this information in MISSION message type then it moves to the street where it is meant to be sent to; using GPS **section-C-[figure 3.7].**
- When it arrives, it will send all collected data and the missionID in END_MISSION message type, then it empties the queue, takes its place, and waits for another mission when it receives the END_MISSION.
- (The Broker_Sub_Side) can also do some operations like charging battery or technically verified. .etc.

The Subscriber blocks any Broker_Sub_Side who has different missionID.

10 Conclusion:

After having presented the model of our solution. The considered smart city application must be tested in scenario of prototype or simulation; using a network simulator in order to evaluate its performances according to well-determined metrics.

In the next chapter, we will use Cupcarbon; to simulate a scenario in order to evaluate this application and see how it operates and what will be the assessment results.

CHAPTER 4:

Evaluation of the solution and results.

1 Introduction:

In the previous chapter, we have thoroughly presented the essentials of our solution. In the present chapter, we are going to present the evaluation and the simulation results of our solution. A simulator is an efficient tool used in many domains; it provides us with an approximate view of the results that can be found in realty.

With extensive simulations, we can discover the errors, overcome them, and do tests in several scenarios. In addition, we report the application's percentage of realization without much effort and cost.

In our smart city application scenario, we used a dedicate network simulator to evaluate the solution according to many evaluation parameters, namely: energy consumption of broker nodes, latency of communicated messages from publisher to subscriber, packet delivery ratio.

2 CubCarbon Network Simulator Version 4.1:

CupCarbon is a Smart City and Internet of Things Wireless Sensor Network (SCI-WSN) simulator. Its objective is to design, visualize, debug and validate distributed algorithms for monitoring, environmental data collection, etc., and to create environmental scenarios such as fires, gas, mobiles, and generally within educational and scientific projects.

CupCarbon offers a simulation environments which enables the design of mobility scenarios and the generation of natural events such as fires and gas as well as the simulation of mobiles such as vehicles and flying objects, for example UAVs.

As shown by **[Figure 4.1],** the CupCarbon Graphical User Interface (GUI) is composed of the following five main parts:

1. The map (in the center).
2. The menu bar (on the top).
3. The Toolbar (below the menu).
4. The parameter menu (on the left).
5. The state bar (at the bottom).
6. The console.

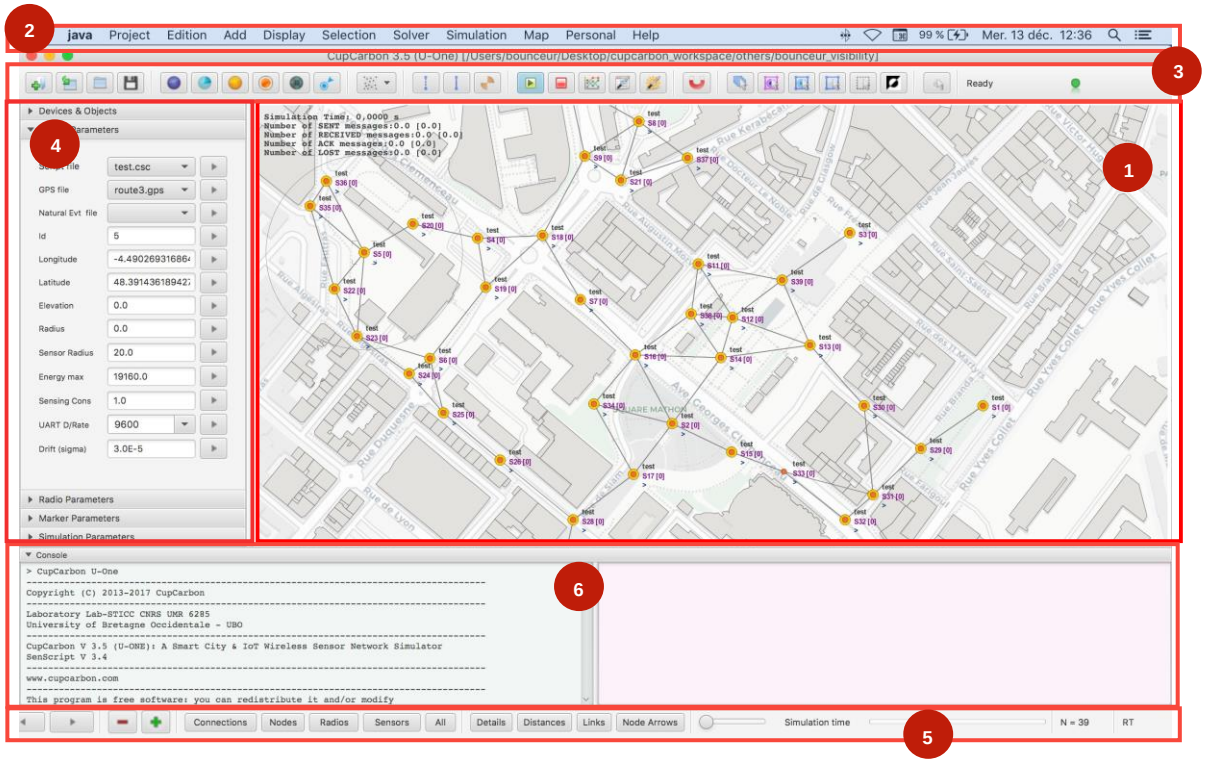

Figure 4. 1: user interface of CupCarbon Network Simulator.

The most important features of cupcarbon simulation are represented as follows:

1. It's easy to use JavaFX interface using the OpenStreetMap (OSM) framework to deploy sensors directly on the map.
2. It includes a script called SenScript, which allows to program and to configure each sensor node individually.
3. The map can be changed according to the preference of the user or the way the information must be presented.
4. It is also possible to generate codes for hardware platforms such as Arduino/XBee from this script, (This part is not fully implemented in CupCarbon)
5. CupCarbon is based on the application layer of the nodes; it offers the possibility to simulate algorithms and scenarios in several steps.
6. The current version of CupCarbon allows to configure the nodes dynamically in order to be able to split nodes into separate networks or to join different networks.
7. The energy consumption can be calculated and displayed as a function of the simulated time. This allows to realistic implementation of a network before its real deployment.
8. The propagation visibility and the interference models are integrated and includes the ZigBee, LoRa and WiFi protocols.
9. Very easy script language based on the SenScript, which includes intuitive commands, Possibility to take into account the topology of a city (buildings) as well as the radio propagation and visibility in this environment,
10. P ossibility to consider the clock drift.

The participants of this project CupCarbon are:

- **Prof. Mohammad Hammoudeh.**
- **Dr. Olivier Marc.**
- **Prof. Laurent Clavier.**
- **Dr. Pierre Combeau.**
- **Prof. Ahcène Bounceur.**

This project that is a part of a bigger research project called PERSEPTEUR and supported by the French research agency ANR (Agence Nationale de la Recherche) under the reference ANR-14-CE24-0017-01. We can see more information about cupcarbon in CupCarbon® User Guide that provided on this link: https://cupcarbon.com/cupcarbon_ug.html.

3 SenScript Language:

SenScript is the script used to program sensor nodes of the CupCarbon simulator. It is a script where variables are not declared and without types, but they can be initialized (set command). For string variables, it is not necessary to use the quotes. A variable is used by its name and its value is determined by $. It is possible to use the instruction function to add complex and additional functions programmed in Java (in a Source code mode only). add complex and additional functions programmed in Java (in a Source code mode only).

4 Why we are choose Cupcarbon network simulator for simulate our application?

We choose cupcarbon network simulator because in our application we need to simulate:

- a smart City(with 3d building map)
- The mobiles nodes.
- The ZigBee Radio of communication.
- Energy Consummation
- GPS

All these elements are supported by Cupcarbon simulator.

5 Simulation parameters:

For simulate our application, we need to adjust some setting in the simulator **[See Table below)** and test it in appropriate scenario.

Table 4. 1: Simulation parameters.

	Parameters	**Values**
Simulation parameters	Time of simulation	10800sec (3 hour)
	Speed of simulation	80ms
	Arrow speed	100ms
	Type of Map	Google Map, light Map Dark Map

	Area	1.5 km^2
Sensor nodes	Type of nodes	Mobile , fix
	Network interface	ZigBee 802.11.4
	GPS	PR.gps,ST1.gps,ST2.gps ,ST3.gps
	Communication range	30m and 75m
	Capacity of battery	20000 Joule
	Elevation	20m for mobile nodes
Network information	Number minimal of mobile nodes	2
	Number minimal of simple nodes	3
	Number of the route	4

6 Our Scenario:

For test our application we suggest to coverage an area such as district “les Halles” in Paris city we needs to connection this area with security center that away from it with **1.3 Km** of distance , so **2.6 Km** for surrounding (distance to go and return) **[see figure 4.2]** to monitor any accident that occurs.

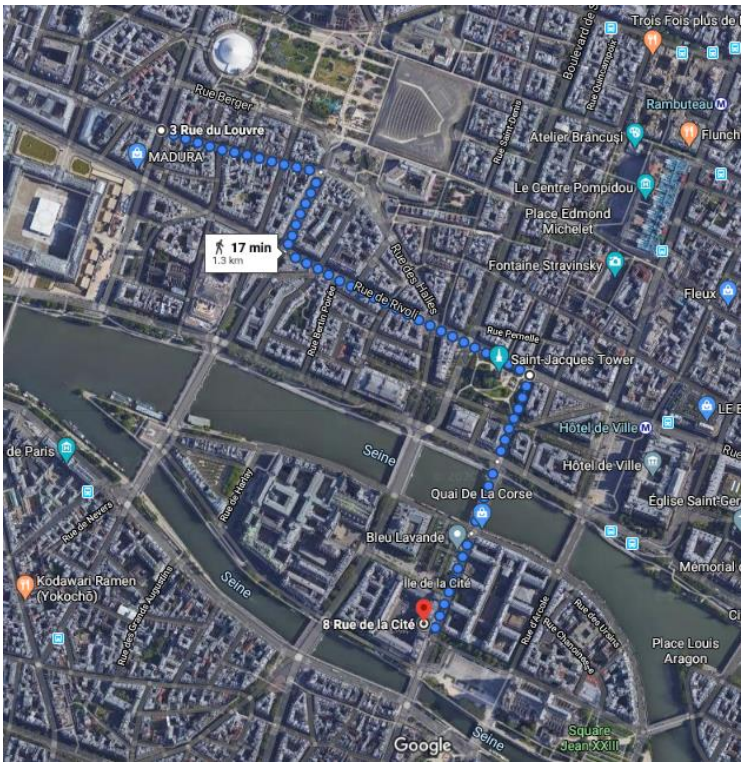

Figure 4. 2: the length of principal route linking between the security center and the covered area.

Then, we suggest to coverage 3 streets in that area.

The First Street **[Figure 4.3]**: have a length of **60 m**, so **120 m** for surrounding (distance to go and return).

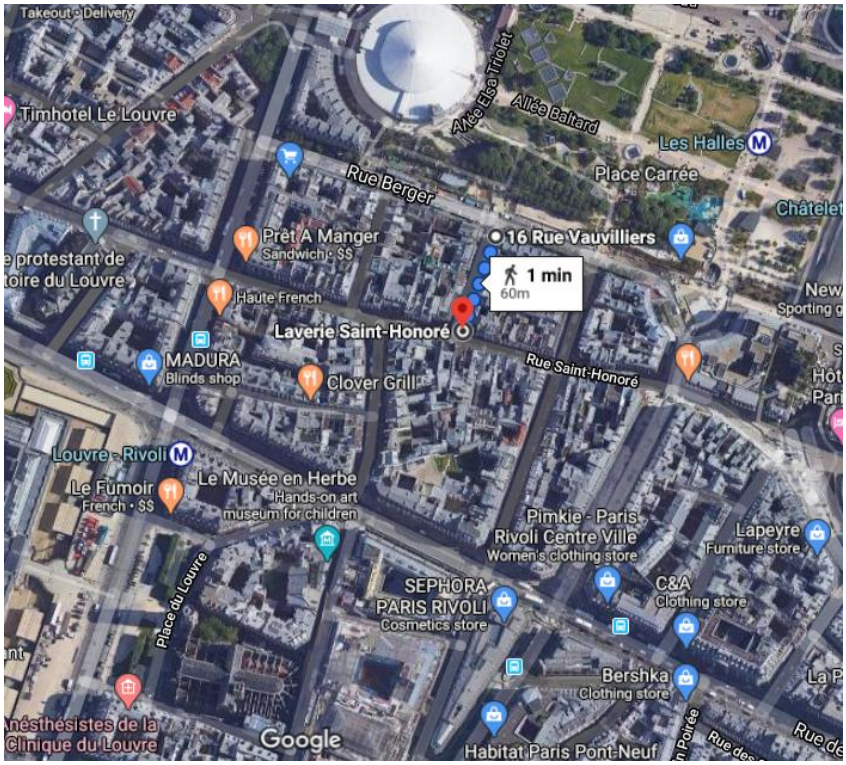

Figure 4. 3: The length of First Street.

The Second Street **[Figure 4.5]**: have a length of **84 m**, so **184 m** for surrounding (distance to go and return).

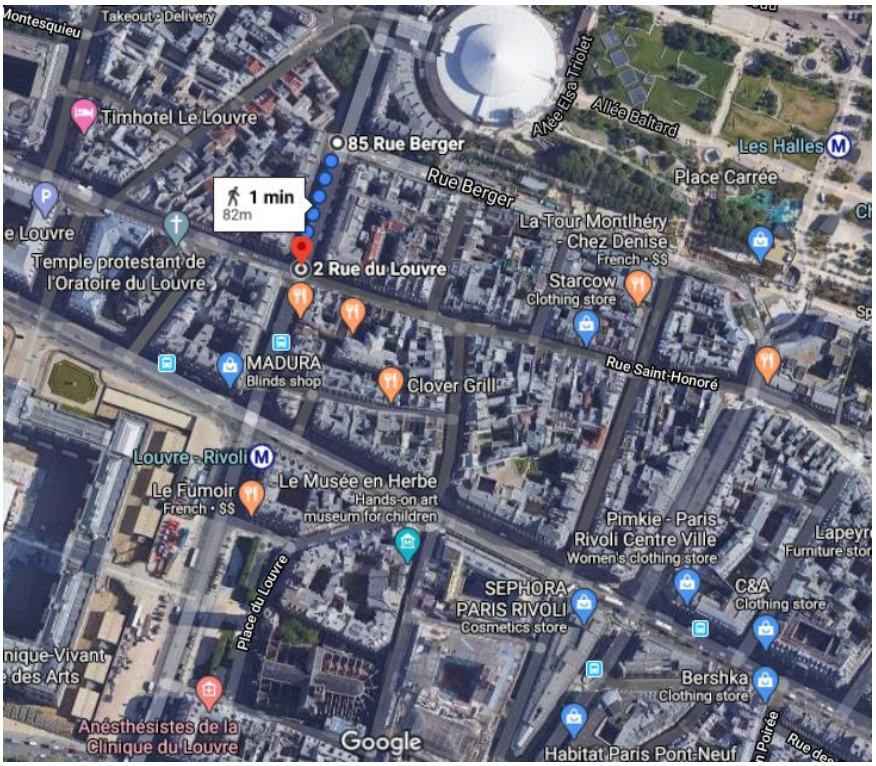

Figure 4. 4: The length of Second Street.

The Third Street **[Figure 4.5]**: have a length of **110 m**, so **220 m** for surrounding (distance to go and return).

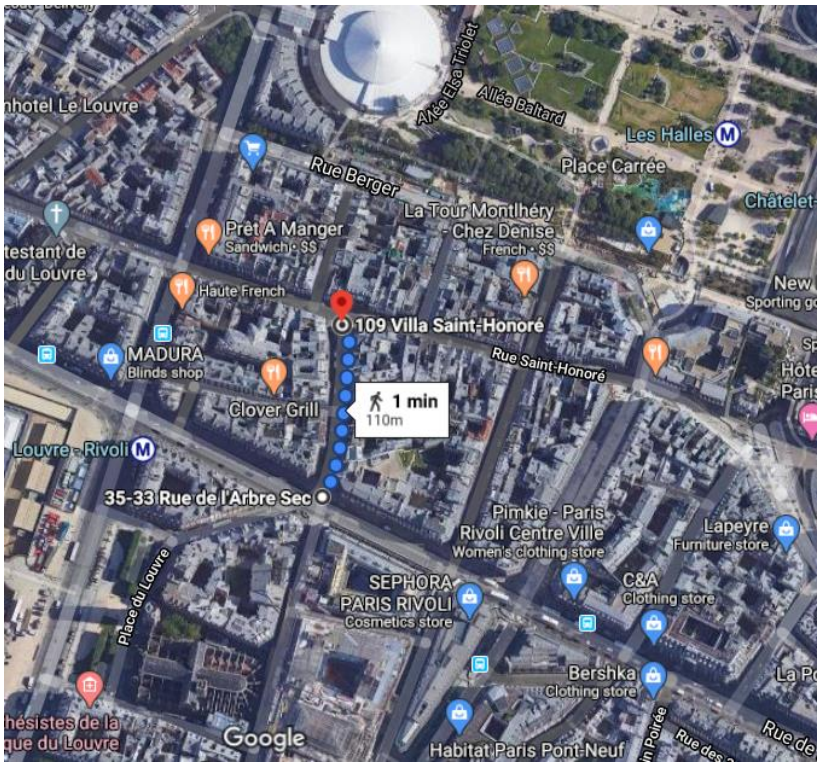

Figure 4. 5: The length of Third Street.

7 The Configuration of network:

In this scenario, we will install 3 ground sensors (see the code source in the end of this chapter) in each street to capture any events that farther occur. In this scenario, we create the events according to the following table.

Figure 4. 6 : The Events that will be captured for each ground sensor.

The Ground Sensors	Type (or topic)	Event time (s)
Ground Sensor 1	ACCIDENT	300
		600
		900
Ground Sensor 2	ACCIDENT	1500
		2000
		2200
Ground Sensor 3	ILLNESS	3000
		3500
		4000

Indeed, the ground sensor needs Publishers and Brokers publisher side (see the code source in the end of this chapter), so we install:

- Three publisher near to ground sensor for each street.
- Three brokers publisher side also for each street, its move in the surround of its street with speed of **30 km/h**, and Elevation of **20m**.

The **[Figure 4.7]** shows how install these nodes in our scenario.

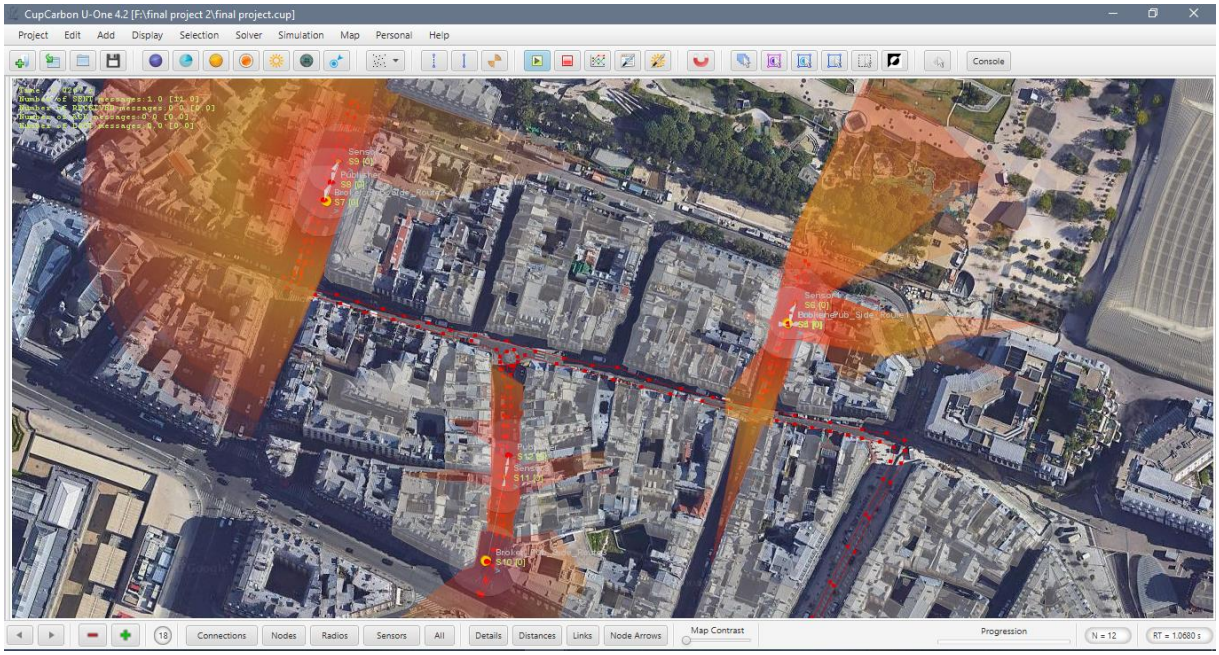

Figure 4. 7: The deployment of the nodes.

In addition, this scenario needs Subscriber and Brokers Subscriber Side (see the code source in the end of this chapter), so we install:

- One Subscriber placed in security center to get the information about accidents topic in the covered area.
- One Broker Subscriber Side placed next to subscriber, and waiting any order for move to the wanted area to collect events.

The **[Figure 4.9]** shows how install these nodes in our scenario.

Note: we added a Helper sensor just for understand more this scenario.

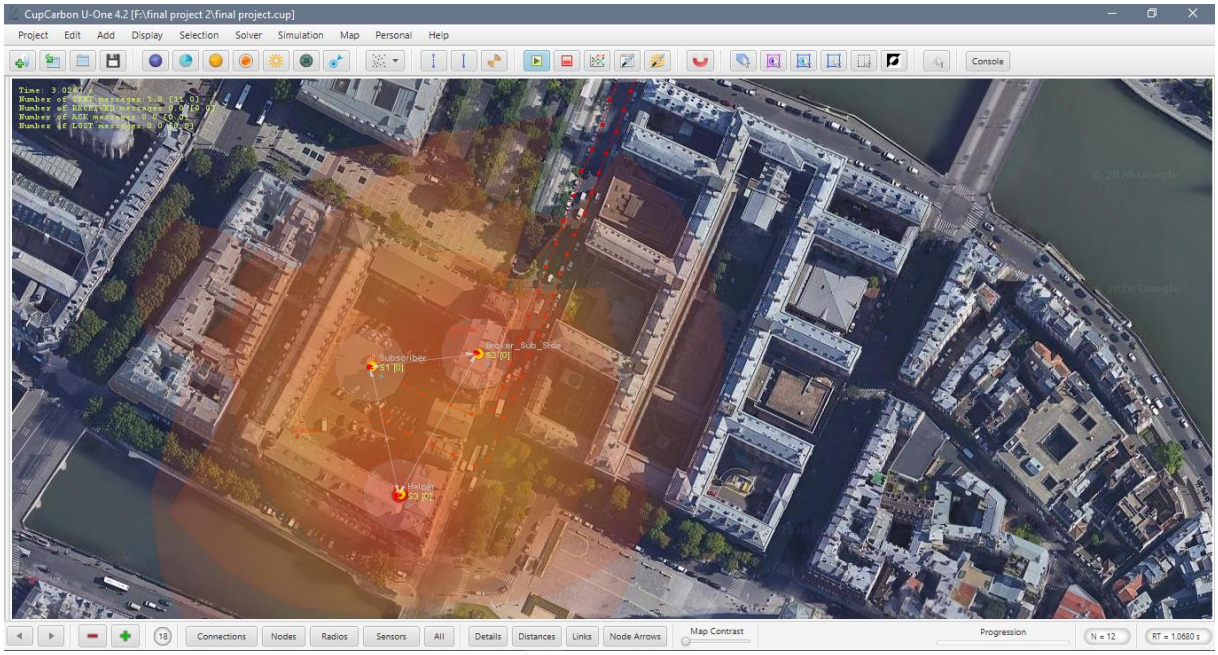

Figure 4. 8: The deployment of Subscriber and Broker_Subscriber_Side nodes.

[Figure 4.9] shows a total vision for our scenario when installed all routes and nodes.

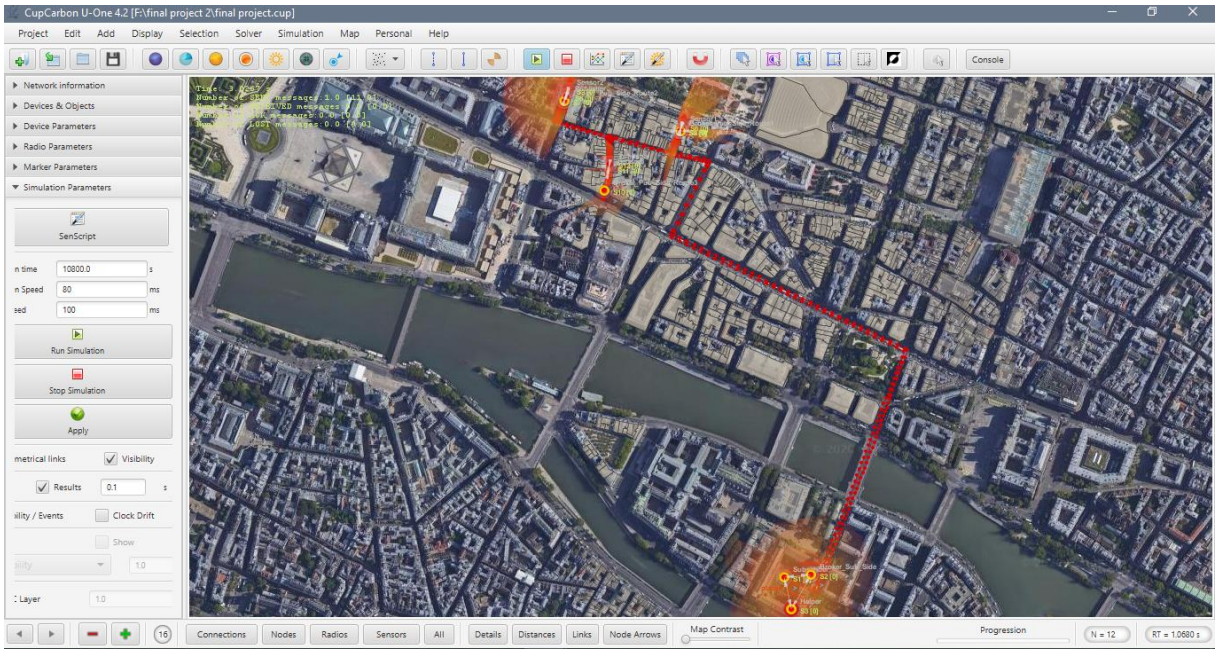

Figure 4. 9: Total vision.

8 The Results of our simulation:

After the configuration, we start the simulation, and we focus to three metrics to get its results, in which are enable us to evaluate our application.

8.1 Energy consummation of the nodes:

After start the simulation, the simulator capture the level of battery to all the sensors each 1ms, so the results shown in **[Figure 4.10]** represent the Energy consummation of all sensors in our scenario.

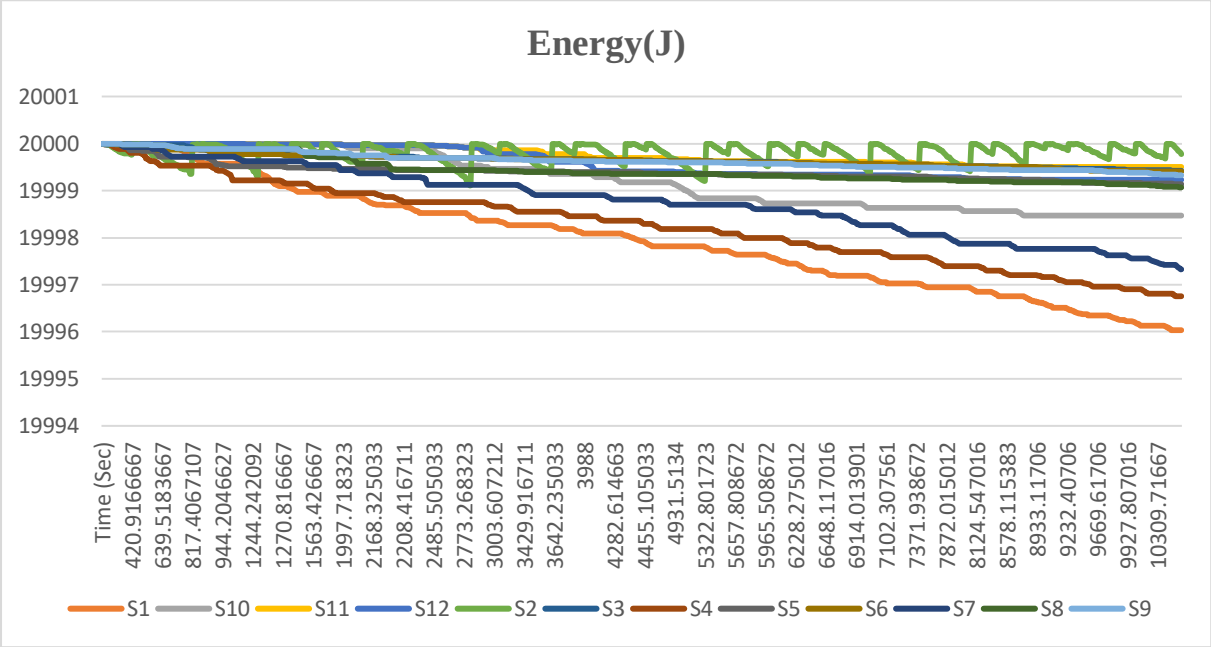

Figure 4. 10: The Energy consummation.

8.2 Latency of communicated messages from publisher to subscriber:

Latency is the time from sending an event to receiving a notification. To calculate it, we need to capture the time of sending the event and time of receiving the notification. The function below represents how we can calculate latency.

$$Latency(e) = Tr(e)\text{-}Ts(e).$$

e: the event sends.

Tr: time of receive event in second.

Ts: time of send event in second.

In our scenario, we will implement this function for 7 levels then we see the Latency results that showing in the **[Table 4.2]**.

Level 1: between Publisher 1 and Broker_Pub_Side 1.

Level 2: between Publisher 2 and Broker_Pub_Side 2.

Level 3: between Publisher 3 and Broker_Pub_Side 3.

Level 4: between Broker_Sub_Side and Broker_Pub_Side 1.

Level 5: between Broker_Sub_Side and Broker_Pub_Side 2.

Level 6: between Broker_Sub_Side and Broker_Pub_Side 3.

Level 7: between Subscriber and Broker_Sub_Side.

Table 4. 2: The Latency values.

Sensors	Time of send events (s)	Time of receive notification(s)	Latency (s)
s5:Publisher 1 s4: Broker pub Side 1	303.5527893	302.4427893	1.11
	603.7027893	602.5927893	1.11
	904.5761227	903.4661227	1.11
s8:Publisher 2 s7: Broker pub Side 2	1506.773879	1505.663879	1.11
	2005.513879	2004.403879	1.11
	2204.963879	2203.853879	1.11
s12:Publisher 3 s10: Broker pub Side 3	3003.556711	3002.446711	1.11
	3503.843879	3502.733879	1.11
	4012.843879	4011.733879	1.11
s5: Broker pub Side 1 s2: Broker Sub Side	544.1283227	543.8683227	0.26
	947.6846627	947.4746627	0.21
	947.7946627	947.6846627	0.11
s7: Broker pub Side 2 s2: Broker Sub Side	1951.668323	1951.458323	0.21
	2438.478323	2438.268323	0.21
	2438.588323	2438.478323	0.11
s10: Broker pub Side 3 s2: Broker Sub Side	544.1283227	543.8683227	0.26
	947.6846627	947.4746627	0.21
	947.7946627	947.6846627	0.11
s2: Broker Sub Side s1: Subscriber	817.4083227	816.7083227	0.7
	1245.342092	1244.542092	0.8
	2150.718323	2150.018323	0.7

The Latency changes from the Publisher to Subscriber showing in the curve in [**Figure 4.11].**

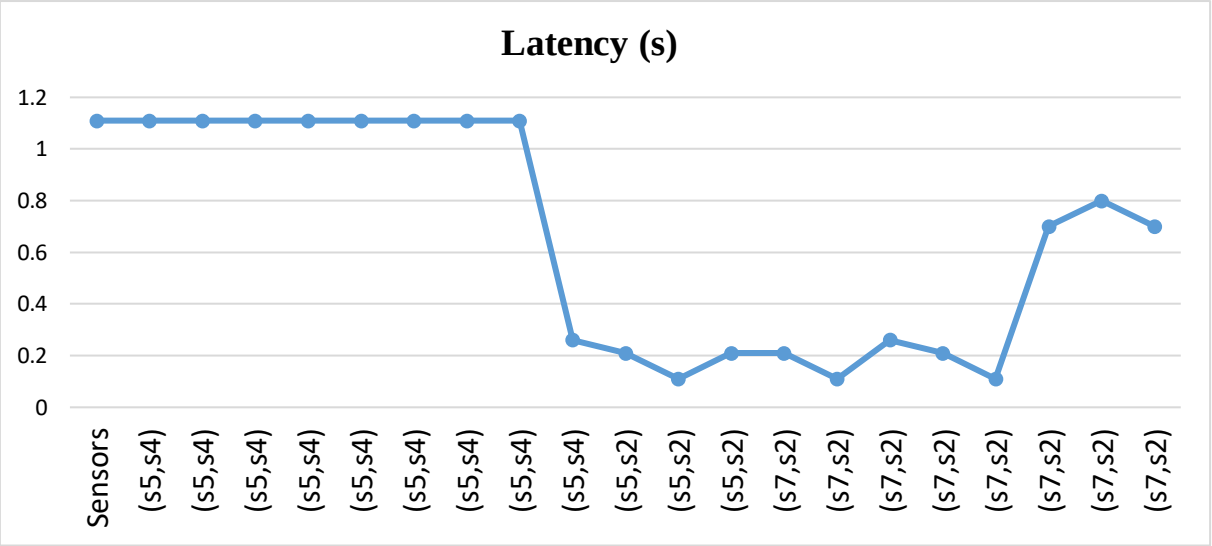

Figure 4. 11: The Latency changes from publisher to subscriber.

8.3 Packet Delivery Ratio:

The end-to-end delivery ratio is the time from publish the event to receive it by subscriber

[Table 4.3] represent end-to-end delivery in our scenario.

Table 4. 3: The End-To-End Delivery of three event.

Events	Tsub(e)	Tpub(e)	Delivery(s)
Event 1	661.258323	544.128323	117.13
Event 2	1243.09466	947.684663	295.41
Event 3	2149.31832	1951.66832	197.65
Average			203.3366

Therefore, the Delivery Ratio is the average showing in **[Table 4.3]** above and its calculated with this function:

$$Delivery\ Ratio = \sum_{e=1}^{n} (Tsub(e) - Tpub(e))/n$$

*n***:** number of event .

*e***:** the event will be transmitted.

*Tsub(e)***:** the time in second of arrive the event to subscriber.

*Tpub(e)***:** the time in second of publish the message.

9 The evaluation of results:

In our scenario, the results obtained are good for all metrics; in this section, we will evaluate them one by one.

9.1 Energy consumption :

The results that shows in the curve in **[Figure 4.10]** give a good results lock at **[Table 4.12],** with 4000 Mah (4000 Mah * 5 Volt = 20000 Joule) with same scenario this battery have life of **630 days**.

Table 4. 4: The evaluation of Energy consummation.

The nodes in our scenario	Max capacity of battery(J)	level of battery after end the simulation (J)	energy that consumed (J)	Percentage %
s1: Subscriber	20000	19996.0356	3.9644	0.019822
s4: Broker pub Side 1	20000	19996.761	3.239	0.016195
s7: Broker pub Side 2	20000	19997.3205	2.6795	0.0133975
s10: Broker pub Side 3	20000	19998.4691	1.5309	0.0076545
s8:Publisher	20000	19999.0747	0.9253	0.0046265
s2: Broker Sub Side	20000	19999.1103	0.8897	0.0044485
s5:Publisher	20000	19999.1417	0.8583	0.0042915
s12:Publisher	20000	19999.2359	0.7641	0.0038205
s6,s9,s11: The ground sensors	20000	19999.415	0.585	0.002925

9.2 Latency:

Our solution provides low Latency, a slight increase can be seen when the number of sensor increases in same street, but it is not a problem, we can fix that by using some techniques such as multichannel option, and dynamically quality of service.

9.3 End-To-End Delivery Ratio:

This metric give a good result in this scenario give 3.38 min for 1.3 km, we can be improved them using :

- A cooperative broker subscriber side (UAVs).
- Speed broker subscriber side (UAVs).
- Also intelligent autonomous UAVs.

10 The Source Codes:

This source codes writing by SenScript Language.

```
//**************************GROUND SENSOR*****************************
    atget id id  // the id of this sensor
    //--------------------------------type of control packet--------------------------
    //---- we propose a new packet in reserved range MQTT SN protocols------
    set ACTION 31 //0x1f  it used when have an action in specific time
    set ACTIONACK 32 //0x21  the response of action sent it by publisher
    //--------------- ------------------------------------------------------------------
    //------------------------------- This is a vector of events time ------------------
    //create the events in different times
    vec event 4
    vset 3000 event 0
    vset 3500 event 1
    vset 4000 event 2
    vset -1 event 3 //for stopping
    //-----------------------------------------------------------------------------------
    //----------------------------------Auxiliary variables ----------------------------
    set step 0 // to guide execution
    set count_event 0 // using for move in events list
    set key $id*100 // for checking the notification of packets
    set Topic ILLNESS  // the name of topic using by this sensor
    set timer 0 // using for sleeping sensor
    set flag_finish 0 // using for finishing the sending without notification
    set PORT 1883  // the port used between ground sensor  and publisher
    //-----------------------------------------------------------------------------------
    //----------------------------------Start the loop---------------------------------
    loop
    time t //The time of simulation
    int cloock $t //convert the time to integer value
    vget ev event $count_event //get events from the vector
    //-----------------------------------------------------------------------------------
    if($cloock == $ev)
      set timer $cloock //timer is the time of action
      inc key //is message id
      inc count_event
      set step 1  //led turn on with red color
      led 13 2
      printfile GroundSensor sent $Topic number $key
```

```
  delay 10
end
//-------------------------------------------------------------------------------------
//------------ send action to publisher with broadcast mode ----------------
if($step == 1)
  set flag_finish 1 // no finish
  set l 25  //is the length of  packet with Oct
  set step 2
  getpos2 lo la // position of event longitude and latitude
  data sp $PORT $id $l $ACTION $Topic $key $timer $lo $la
  send $sp
  printfile GroundSensor $id --action: $key -->Publisher in $t
  delay 100
end
//-------------------------------------------------------------------------------------
//------------------------- this step wait an ACTIONACK-------------------
if($step == 2)
wait 1000 //wait for any packet that coming
read rp  //read the packet
  if($rp != \)
  rdata $rp p rid len type
    if($p == $PORT)
      if($type == $ACTIONACK)
        rdata $rp p rid len type mssgId
        if($mssgId == $key)
        printfile GroundSensor $id <--actionack--Publisher in $t
        led 13 0
        set flag_finish 0
        set step 0
        set timer 0
        cprint  The event is sent with successful
        delay 10
        end
      end
    end
  else
  set step 1
  end
end
//-------------------------------------------------------------------------------------
//---------------------for try again the connection----------------------------
//in the case where it cannot connect, the publisher stay sending the packet for 60 sec
if($flag_finish == 1)
 minus taux $cloock $timer  //how long ago without notification (ack)
   if($taux > 60)
     set step 0
     set flag_finish 0
     set timer 0
   end
```

```
 delay 10
end
//-------------------------------------------------------------------------------
delay 1000
//-------------------------------------Next Loop ------------------------------
//*************************************************************************
//*****************************PUBLISHER**********************************
atget id id
//--------------------------------type of control packet---------------------------
//--- we propose a new packet in reserved range MQTT SN protocols----
set SEARCHGW 1 //0x01 using by publisher to find where is the broker_pub_side
set GWINFO 2 //0x02 is the reply of SEARCHGW sended by broker_pub_side
set CONNECT 4 //0x04 using for get connection
set CONNACK 5 //0x05 reply of CONNECT sended by broker_pub_side
set PUBLISH 12 //0X0B using to published the event to broker
set PUBACK 13 //0x0C reply of PUBLISH packet sended by broker
set ACTION 31 //0x1f
set ACTIONACK 32 //0x21
set THERAPY 39 //0x27
set THERAPYACK 40 //0x28
//-----------------------------------------------------------------------------------
//-------------------------vectors used by publisher-------------------------
// list of actions that are collected(publisher data base)
vec TopicDB 100
for i 0 100 1
vset \ TopicDB $i
end
//-----------------------------------------------------------------------------------
//---------------------------------Auxiliary variables ------------------------
set step 0
set PORT1 1883 // the port used between ground sensor and publisher
set PORT2 1884 // the port used between publisher and broker pub side
set topicDB_count 0 // index for topicDB
set pub_flag 0 //to test if not published set  to 0 else set to 1
set raduis 50 //this represent the distance to get connection
set SEARCHGW_luck 0 //try connection
set broker_pub_side \ //save GW information
set next 0 // for stop publishing
//return code value
set Accepted 0
set Rejected_congestion 1
set Rejected_invalid_topic 2
set Rejected_not_supported 3
//flags parameter
set DUP \ //set 0 if send first time else 1 using in publish packet
set QoS1 2817 // 0x0b01
set retain \
set will \
set CleanSession \
```

```
set TopicIdType 2817 //0x0b01
set index 0
set protocolId MQTT_SN // protocol version
set Duration 0 //keep alive
set ClientId 0
set CONNECT_luck 0 //test of connection
set CONNECT_timer 0 // if don't receive a CONNACK packet
//-------------------------------------------------------------------------------
//----------------------------------Start the Loop---------------------------------
loop
time t
int cloock $t
wait 500
read rp
if($rp != \)
 rdata $rp p rid len type
  if($p == $PORT1)
//-----------------------------Receive the event and save it-------------------
    if($type == $ACTION)
     rdata $rp p rid len type Topic mssgId t lo la
     data save_topics $mssgId $pub_flag $topicDB_count $rid $Topic $t $lo $la
     vset $save_topics TopicDB $topicDB_count
     inc topicDB_count
     led 13 2 //led turn on with red color
     //printfile Publisher<--action: $mssgId --GroundSensor in $t
     set step 1
     set l 8
     data sp $p $id $l $ACTIONACK $mssgId
     send $sp $rid
     //printfile Publisher--actionack: $mssgId -->GroundSensor in $t
     delay 100
    end
//-------------------------------------------------------------------------------
    if($type == $THERAPY)
     led 13 0
     set l 6
     data sp $p $id $l $THERAPYACK
     send $sp $rid
     delay 10
    end
  end

  if($p == $PORT2)
//------------Receive the GW for publish the data (events) ----------
     if($type == $GWINFO)
       rdata $rp p rid len type GWID GWADD
       set broker_pub_side $rp
       set SEARCHGW_luck 0
       set step 2
```

```
      delay 10
    end
//-------------------------------------------------------------------------------
//----------------------Receive connection confirmation-----------------
    if($type == $CONNACK)
      set CONNECT_luck 0
      set step 3
      delay 10
    end
//-------------------------------------------------------------------------------
//----------------------Receive publish confirmation----------------------
    if($type == $PUBACK)
      set step 0
      data $rp p rid len type Topic_name mssgId return_code
       for i 0 $topicDB_count 1
         vget data_topic TopicDB $i
         vdata content $data_topic
         vget n0 content 0
         vget n1 content 1
         vget n2 content 2
         vget n3 content 3
         vget n4 content 4
         vget n5 content 5
         vget n6 content 6
         vget n7 content 7
         if($n0 == $mssgId)
          printfile Publisher<--puback: $mssgId ---broker_pub_side in $t
                  //change state to published (Pub_flag set to 1)
          data alterTopic $n0 1 $n2 $n3 $n4 $n5 $n6 $n7
          vset $alterTopic TopicDB $i
          led 13 14
           if($next == 1)
            set step 3
           end

          set index $topicDB_count+1
           if($index == \)
            set step 0
           end
          set i $topicDB_count
          delay 10
         end
       end
      end
//-------------------------------------------------------------------------------
   end
end
//-------------------------send search gateway request----------------------
if($step == 1)
```

```
  set l 7 //length of packet
  data sp $PORT2 $id 7 $SEARCHGW $raduis
  inc SEARCHGW_luck
  //send the packet in broadcast mode
  set step 0
  send $sp
  delay 100
end
//-------------------------------------------------------------------------------
//--------------------send communication request to GW----------------
if($step == 2)
  vdata info $broker_pub_side
  vget gwid info 4 // get id of broker pub side
  vget gwadd info 5 // get port of broker pub side
//create a connect packet
  set step 0
  inc CONNECT_luck
  set CONNECT_timer $cloock
  set Duration $cloock+20
  set ClientId $id
  set l 23
  data Flags $DUP $QoS1 $retain $will $CleanSession $TopicIdType
  data sp $gwadd $id 16 $CONNECT $Flags $protocolId $Duration $ClientId
  send $sp $gwid
  delay 100
end
//-------------------------------------------------------------------------------
//------------Publish the events (topic content) on GW topic------------
if($step == 3)
  for i 0 $topicDB_count 1
    vget data_topic TopicDB $i
    vdata content $data_topic
    vget mssageID content 0
    vget pub_f content 1
    vget sensor_id content 3
    vget Tid content 4
    vget time_of_event content 5
    vget longitude content 6
    vget latitude content 7
      //test if exist topics for published
    if($pub_f == 0)
      data event_info $sensor_id $time_of_event $longitude $latitude
      data Flags $DUP $QoS1 $retain $will $CleanSession $TopicIdType
      set l 23
      data sp $PORT2 $id $l $PUBLISH $Flags $Tid $mssageID $event_info
      vdata broker_info $broker_pub_side
      vget bid broker_info 4
      set next 1 //exist topics for published
      send $sp $bid
```

```
        printfile DATA--> $event_info  in $t
        printfile Publisher--publish: $mssgId --->broker_pub_side in $t
        delay 100
        set i $topicDB_count
       // for stopped where found topic no published
      end
    end
   //from disconnect where all topics published
    if($next == 0)
     cprint finich
       set next 0
    set step 0
    end
    set step 0
  end
  //-------------------------------------------------------------------------------
  //---------------------------try for search GW-------------------------------
  if($SEARCHGW_luck > 0)
    set step 1
    delay 10
  end
  //-------------------------------------------------------------------------------
  //--------------allowed to send 4 packet to try the connection-----------
  if(($CONNECT_luck > 0)&&($CONNECT_luck < 4))
    minus taux $cloock $CONNECT_timer
      if($taux > 80)
       cprint time_out
       set step 2
      end
    delay 10
  end
  //-------------------------------------------------------------------------------
  if($cloock == 1100)
    set step -1
  end

  delay 1000
  //-------------------------------------Next loop ------------------------------
//*************************************************************************
//************************BROKER PUBLISHER SIDE **********************
  atget id id // the id of this sensor
  //------------------------------------ type of control packet ---------------------------------
  set SEARCHGW 1 //0x01
  set GWINFO 2 //0x02
  set CONNECT 4 //0x04
  set CONNACK 5 //0x05
  set PUBLISH 12 //0X0B using for publish the events in broker
  set PUBACK 13 //0x0C reply of PUBLISH packet sent it by broker
  set SUBSCRIBE 18 //0x12 using for get topic
```

```
set SUBACK 19 //0x13 reply of SUBSCRIBE send it by broker_sub_side
//-------------------------------------------------------------------------------------------------
//--------------------------------------Auxiliary variables ----------------------------------
set step 0
set PORT1 1884 // the port used between publisher and broker publisher side
set PORT2 1885 // the port used between broker publisher side and broker subscriber side
set Raduis 0 //
set access_CONNECT 0 //for optimist the code
set access_SEARCHGW 0 //for optimist the code
//return code value
set Accepted 0
set Rejected_congestion 1
set Rejected_invalid_topic 2
set Rejected_not_supported 3
set Nothing 4 //if there are no topic
set activate_keep_alive 0
set disponible 0 //using for checked return code and test availability of topics
set disponible2 0
set keep_alive_timer 0 //for count the time of connection
set st ST1 // its street
set protocolId MQTT_SN
set return_code 0
//level 0 list using for save the data (or events or the action that received) in its topic.
vec sys_topic 3 //in this case we propose 3 type of per-defined topics
// sys_topic  list --> |type of topic (ex:ACCIDENT)|there are new or no (0|1) |and length
of the //list|
set new_topic 0
set length_topics 0
data topic1 ACCIDENT $new_topic $length_topics
data topic2 ILLNESS $new_topic $length_topics
data topic3 FIRE $new_topic $length_topics
vset $topic1 sys_topic 0
vset $topic2 sys_topic 1
vset $topic3 sys_topic 2
//level 1
//predefined topic integrated in broker publish side
//we had proposed 3 per-defined topics ACCIDANT ILLNESS FIRE
// | Topic_name(ex ACCIDENT)|mssgId|sensor_ID|time_of_action|longitude|latitude|
vec Topic_ACCIDENT 100
set count_Topic_ACCIDENT 0

vec Topic_ILLNESS 100
set count_Topic_ILLNESS 0

vec Topic_FIRE 100
set count_Topic_FIRE 0

//-------------------------------------------------------------------------------------------------
//--------------------------------------Start the loop ----------------------------------
```

```
loop
time t
int cloock $t
wait 500
getpos lo la
read rp
if($rp != \)
 rdata $rp p rid len type
  if($p == $PORT1)
//-----------------------Receive the GW for publish the data (events)----------
    if($type == $SEARCHGW)
      set access_SEARCHGW 1
    end
//-------------------------------------------------------------------
//-------------------------- Receive connection confirmation-----------------------------
    if($type == $CONNECT)
      set access_CONNECT 1
    end
//-----------------------------------------------------------------------------------------------
//------------------------------receive publish data-----------------------------------------
    if($type == $PUBLISH)
      set disponible 0 // to confirms  there is the topic that need or no
      rdata $rp p rid len type DUP QoS retain will cleanSession TopicIdType Topic_name
mssgId sensor_ID time_of_action lo la
       set return_code $Accepted
      if(($Topic_name == ACCIDENT)||($Topic_name == ILLNESS)||($Topic_name ==
FIRE))
       set disponible 1
      end

      if($disponible == 0)
       set return_code $Rejected_invalid_topic
      else
//in this section broker will manage topics received
//-------------------Find the topic and save it the data published-------------------------
       for i 0 3 1
        vget sys sys_topic $i
        vdata vsys $sys
        vget index0 vsys 0
        vget index1 vsys 1
        vget index2 vsys 2
        if($index2 < 100)
         if($index0 == $Topic_name)
          data sv $Topic_name $mssgId $sensor_ID $time_of_action $lo $la
          printfile DATA Published--> $sv  in $t
           if($i == 0)
            //printfile broker_pub_side<--publish: $mssgId ---Publisher in $t
            vset $sv Topic_ACCIDENT $count_Topic_ACCIDENT
            inc count_Topic_ACCIDENT
```

```
              data update_sys $index0 1 $count_Topic_ACCIDENT
              vset $update_sys sys_topic $i
              delay 10
            else
               if($i == 1)
                //printfile broker_pub_side<--publish: $mssgId ---Publisher in $t
                vset $sv Topic_ILLNESS $count_Topic_ILLNESS
                inc count_Topic_ILLNESS
                data update_sys $index0 1 $count_Topic_ILLNESS
                vset $update_sys sys_topic $i
                delay 10
               else
                 if($i == 2)
                  //printfile broker_pub_side<--publish: $mssgId ---Publisher in $t
                  vset $sv Topic_FIRE $count_Topic_FIRE
                  inc count_Topic_FIRE
                  data update_sys $index0 1 $count_Topic_FIRE
                  vset $update_sys sys_topic $i
                  delay 10
                 end
               end
             end
           end //test topic
          else //test lenght_of list
           cprint full $i
         end //test lenght_of list
       end
       set l 22
       data sp $p $id $l $PUBACK $Topic_name $mssgId $return_code
       send $sp $rid
       //printfile broker_pub_side--puback: $mssgId --->Publisher in $t
       delay 100
     end //test disponibal
//-------------------------------------------------------------------------------
   end
//-------------------------------------------------------------------------------
  end
  if($p == $PORT2)
   if($type == $SEARCHGW)
     set access_SEARCHGW 1
   end

   if($type == $CONNECT)
    set access_CONNECT 1
   end
 //------------Receive a subscription to get topic content ----------
   if($type == $SUBSCRIBE )
     set disponible2 0
     rdata $rp p rid len type dup qos Retain Will Cleansession topictype
```

```
          MSSGID   Topic_name
          set return_code $Accepted
       if(($Topic_name == ACCIDENT)||($Topic_name == ILLNESS)||($Topic_name ==
FIRE))
           set disponible2 1
       end

         if($disponible2 == 0)
          set return_code $Rejected_invalid_topic
         else
          for i 0 3 1
            vget sys sys_topic $i
            vdata vsys $sys
            vget index0 vsys 0
            vget index1 vsys 1
            vget index2 vsys 2
           if($index1 == 1)
             if($index0 == $Topic_name)
    //In this section will search the topic wanted and get its data (the same as i=1 and i=2)-
                 if($i == 0)
                printfile broker_pub_side $id <--subscribe: $MSSGID ---broker_sub_side in $t
                   set limit $count_Topic_ACCIDENT-1
                    for j 0 $count_Topic_ACCIDENT 1
                    vget content Topic_ACCIDENT $j
                set last 0
                    if($j == $limit )
                 set last 1
                     set st ST2
                     data indexes $index0 0 $index2
                     vset $indexes sys_topic $i
                     vec Topic_ACCIDENT 100
                     set count_Topic_ACCIDENT 0
               end
                   set l 28
              data Flags $DUP $QoS1 $retain $will $CleanSession $TopicIdType
              data sp $PORT2 $id $l $SUBACK $Flags  $Topic_name
                  $MSSGID $st $ $content $return_code
              send $sp $rid
                  //printfile broker_pub_side--suback: $MSSGID --->broker_sub_side in $t
             delay 100
            end
   //-------------------------------------------------------------------------------------------------
                else
                   if($i == 1)
                    //printfile broker_pub_side<--subscribe: $MSSGID ---broker_sub_side in $t
                     set limit $count_Topic_ILLNESS-1
                     for j 0 $count_Topic_ILLNESS 1
                       vget content Topic_ILLNESS $j
                   set last 0
```

```
            if($j == $limit )
             set last 1
                 set st ST2
                 data indexes $index0 0 $index2
                 vset $indexes sys_topic $i
                 vec Topic_ILLNESS 100
                 set count_Topic_ILLNESS 0
            end
                set l 23
            data Flags $DUP $QoS1 $retain $will $CleanSession $TopicIdType
            data sp $PORT2 $id $l $SUBACK $Flags $Topic_name
                $MSSGID $st $last  $content $return_code
             send $sp $rid
                //printfile broker_pub_side--suback: $MSSGID --->broker_sub_side in $t
             delay 100
          end
            else
               if($i == 2)
             //printfile broker_pub_side<--subscribe: $MSSGID ---broker_sub_side in $t
                 set limit $count_Topic_FIRE-1
                  for j 0 $count_Topic_FIRE 1
                    vget content Topic_FIRE $j
                set last 0
                if($j == $limit )
                  set last 1
                     set st ST2
                     data indexes $index0 0 $index2
                     vset $indexes sys_topic $i
                     vec Topic_FIRE 100
                     set count_Topic_FIRE 0
                end
                   set l 20
                data Flags $DUP $QoS1 $retain $will $CleanSession $TopicIdType
                data sp $PORT2 $id $l $SUBACK $Flags $Topic_name
                   $MSSGID $st $last $content $return_code
                 send $sp $rid
                 //printfile broker_pub_side--suback: $MSSGID --->broker_sub_side in $t
                delay 100
               end
                 end
              end
           end
        end //test topic
      else
      set return_code $Nothing // not forget you must declar
  end
    end
  end
end
```

```
  end

   if($access_SEARCHGW== 1)
    rdata $rp p rid len type raduis
    set Raduis $raduis
    set l 11
    set access_SEARCHGW 0
    data sp $p $id $l $GWINFO $id $p
    send $sp $rid
    delay 100
   end

   if($access_CONNECT == 1)
    rdata $rp p rid len type DUP QoS retain will cleanSession TopicIdType protocol
   duration clientId
    set return_code $Accepted
    set keep_alive_timer $duration
     if($protocol == $protocolId)
      set return_code $Rejected_not_supported
      if($cloock > $duration)
       set return_code $Rejected_congestion
      end
     end
    set l 7
    data sp $p $id $l $CONNACK $return_code
    set activate_keep_alive 1
    set access_CONNECT 0
    send $sp $rid
    delay 100
   end
end
//----------------------allow to count the time of connection -----------
if($activate_keep_alive == 1)
 if($keep_alive_timer < $cloock)
  cprint disconnect_with_keep_alive in $cloock
  set activate_keep_alive 0
  set keep_alive_timer 0
 end
 delay 10
end
//-------------------------------------------------------------------------------
if($cloock == 255)
 for i 0 $count_Topic_ACCIDENT 1
  vget t10 Topic_ACCIDENT $i
  cprint ======= $t10 ==========
  delay 10
 end
end
delay 100
```

```
    //-------------------------------------Next Loop ----------------------------
 //*************************************************************************
 //******************BROKER SUBSCRIBE SIDE SENSOR**********************
    atget id id // the id of this sensor
    //---------------------------type of control packet----------------------------
    set SEARCHGW 1 //0x01
    set GWINFO 2 //0x02
    set CONNECT 4 //0x04
    set CONNACK 5 //0x05
    set SUBSCRIBE 18 //0x12
    set SUBACK 19 //0x13
    set MISSION 33 // 0x21 using for receive the orders from Subscriber
    set MISSIONACK 34 //0x22 the reply of MISSION packet
    set END_MISSION 35 // 0x23 using where arrive uav to its place
    set END_MISSIONACK 36 //0x24 is the replay of FIN_MISSION
    vec Data_of_topics 150 // vector of storge
    set count_Data_of_topics 0 // its count of Data_of_topics vector
    //-------------------------------------------------------------------------
    //-------------------------------Auxiliary variables ----------------------
    set step 0
    set PORT1 1885 // the port used between broker pub side and broker sub side
    set PORT2 1886 // the port used between broker sub side and subscriber
    set SEARCHGW_luck 0
    set CONNECT_luck 0
    set CONNECT_timer 0
    set broker_sub_side \
    set place place1
    set topic_name \
    set missionId 0
    set nextstreet \
    set position 0
    set subID -1
    set GW 0
    set sub_key $id*100
    //return code value
    set Accepted 0
    set Rejected_congestion 1
    set Rejected_invalid_topic 2
    set Rejected_not_supported 3
    set Nothing 4 // new proposed  return code pointing to topics does not exist
    //--------------------position of the start point of the street -------------------
    set 1_lo 2.344060242176056
    set 1_la 48.861242179664664
    set 2_lo 2.3412680625915527
    set 2_la 48.86175390660024
    set 3_lo 2.342483103275299
    set 3_la 48.8613357025641
    set final_lo 2.3469221591949463 // position of the end point of the street
    set final_la 48.854373944975215
```

```
//-------------------------------------------------------------------------------
//flag parameters
set DUP \ //set 0 if send first time else 1 using in publish packet
set QoS1 2817 // 0x0b01
set retain \
set will \
set CleanSession \
set TopicIdType 2817 //0x0b01
set try_con 0 // help us for get connrction
set protocolId MQTT_SN
set Duration 0 //keep alive
set ClientId 0
set CONNECT_luck 0 //test of connection
set CONNECT_timer 0 // if don't receive a CONNACK packet control
//-------------------------------------------------------------------------------------
//----------------------------------Start the loop ----------------------------------

loop
getpos2 lo la
//-----------------------------------try search GW-------------------------
if($SEARCHGW_luck > 3)
  set step 0
  set SEARCHGW_luck 0
  route PR
  delay 50
end
//-------------------------------------------------------------------------------------
//----------------------section for guider the UAV Broker ------------------
if(($1_lo == $lo)&&($1_la == $la))
  if($try_con < 2)
    route ST1
    set step 1
  else
     rmove 840 //speed of uav  840 ms between 2 market (7m) see simulation map
     route PR  //change the route
     route ST2
     set step 0
     set try_con 0
  end
 delay 50
end
if(($2_lo == $lo)&&($2_la == $la))
 if($try_con < 2)
    route ST2
    set step 1
  else
     rmove 840
     route PR
     route ST3
```

```
        set step 0
        set try_con 0
       end
     delay 50
     end

    if(($3_lo == $lo)&&($3_la == $la))
     if($try_con < 2)
       route ST3
       set step 1
       else
        rmove 840
         route PR
        set step 0
        set try_con 0
       end
     delay 50
     end
    //-------------------------------------------------------------------------------
    time t
    int cloock $t
    wait 500
    read rp
    if($rp != \)
    rdata $rp p rid len type

      if($p == $PORT1)
      //------------Receive the GW for publish the data (events) ------------
        if($type == $GWINFO)
         rdata $rp p rid len type GWID GWADD
         set broker_sub_side $rp
         set SEARCHGW_luck 0
         set step 2
         delay 10
        end
    //-------------------------------------------------------------------------------
    //-----------------Receive connection confirmation----------------------
        if($type == $CONNACK)
         set CONNECT_luck 0
         cprint broker_sub_received_connack
         set step 3
         delay 10
         end
    //-------------------------------------------------------------------------------
    //---------------- Receive and save  the data of topic are wanted--------------
         if($type == $SUBACK)
           rdata $rp p rid len type Dup QoS Retain Will cleanSession topicIdtype Topic_name   MSSID
street last Topic_id mssgId sensor_ID time_of_action LO LA RC
            cprint $RC   $Accepted
```

```
        if($RC == $Accepted)
         if($sub_key == $MSSID)
           printfile broker_sub_side<--suback: $mssgId ---broker_pub_side in $t
           if($Topic_name == $topic_name)
             if($count_Data_of_topics < 150 )
                 dec try_con
                    data save $Topic_id $mssgId $sensor_ID $time_of_action $LO $LA
                    vset $save Data_of_topics $count_Data_of_topics
                 printfile DATA--> $save  in $t
                    inc count_Data_of_topics
                    if($last == 1)
                   set nextstreet $street
                   inc sub_key
                     set position 1
                    end
               delay 10
                 else
                 cprint plain list
          end
         end
        end
      end
   end
 //-------------------------------------------------------------------------------
  end

  if($p == $PORT2)
 //------------------Receive order from Subscriber----------------------------
    if($type == $MISSION)
      rdata $rp p rid len type Topic_name missionID street
      set subID $rid
      set missionId $missionID
      set nextstreet $street
      set subID $rid
      set position 1
      set topic_name $Topic_name
      data sp $p $id 4 $MISSIONACK
      send $sp $rid
      //printfile broker_sub_side<--mission: $missionID ---subscriber in $t
      delay 100
    end
//-------------------------------------------------------------------------------------
//---Receive confirmation that mission flinched with successfully ---------
    if($type == $END_MISSIONACK)
      rdata $rp p rid len type street
      printfile broker_sub_side<--end_missionack---subscriber in $t
      set nextstreet $street
      vec Data_of_topics 150
      set count_Data_of_topics 0
```

```
      set position 2
      // charge the battery 4000mah * 5v = 20000 joule
      battery set 20000
      delay 10
    end
  end
//-------------------------------------------------------------------------------------
//-------------------------switching between the routes ----------------------

 if($position == 1)
   set position 0
   rmove 840
   route PR
   route $nextstreet
   delay 50
  end

  if($position == 2)
   rmove 840
   set position 0
   route $nextstreet
   delay 50
  end
//-------------------------------------------------------------------------------------
end
//-------------------------send a message to search GW----------------------------
if($step == 1)
  data sp $PORT1 $id 4 $SEARCHGW $raduis
  inc SEARCHGW_luck
  //set step 0
  send $sp
  delay 100
end
//-------------------------------------------------------------------------------------
//------------------- send a message to connect with the GW ---------------------
if($step == 2)
  vdata info $broker_sub_side
  vget gwid info 4 // get id of broker pub side
  vget gwadd info 5 // get port of broker side
  //create connect packet
  set GW $gwid
  inc CONNECT_luck
  set CONNECT_timer $cloock
  set Duration $cloock+50
  set ClientId $id
  data Flags $DUP $QoS1 $retain $will $CleanSession $TopicIdType
  data sp $gwadd $id 16 $CONNECT $Flags $protocolId $Duration $ClientId
  set step 0
  send $sp $gwid
```

```
  delay 100
 end
 //-------------------------------------------------------------------------------------
 //--------------- send a message to subscribe in topic--------------------------
 if($step == 3)
  data Flags $DUP $QoS1 $retain $will $CleanSession $TopicIdType
  data sp $PORT1 $id 16 $SUBSCRIBE $Flags $sub_key $topic_name
  set step 0
  send $sp $GW
  inc try_con
  printfile broker_sub_side--subscribe: $sub_key --->subscriber in $t
  delay 100
 end
 //-------------------------------------------------------------------------------------
 //---------------allowed to send 4 packet to try the connection----------------
 if(($CONNECT_luck > 0)&&($CONNECT_luck < 4))
  minus taux $cloock $CONNECT_timer
   if($taux > 10)
    cprint time_out
    set step 2
   end
  delay 10
  end
 //-------------------------------------------------------------------------------------
 //----------------------send a finished mission message -------------------------
 if(($final_lo == $lo)&&($final_la == $la))
   set limit $count_Data_of_topics-1
  for j 0 $count_Data_of_topics 1
   vget content Data_of_topics $j
   set last 0
        if($j == $limit )
         set last 1
        end
   data sp $PORT2 $id 10 $END_MISSION $Topic_name $missionId $content $last
   send $sp $subID
   printfile broker_sub_side--end_mission:missionId --->subscriber in $t
   delay 100
  end
  if($count_Data_of_topics  == 0)
   data sp $PORT2 $id 10 $END_MISSION $Topic_name $missionId \ \ \ \ \ 1
   send $sp $subID
   printfile broker_sub_side--end_mission:missionId --->subscriber in $t
   delay 100
  end
 end
 //-------------------------------------------------------------------------------------
 delay 100
 //-----------------------------------Next Loop ---------------------------------
//*************************************************************************
```

```
//*****************************SUBSCRIBER ******************************
    atget id id // the id of this sensor
    //--------------------------type of control packet----------------------------
    set MISSION 33 // 0x21
    set MISSIONACK 34 //0x22
    set END_MISSION 35 // 0x23
    set END_MISSIONACK 36 //0x24
    set HELP 37 //a secondary message type for help us in simulation
    set HELPACK 38
    //-------------------------------------------------------------------------------
    //--------------------------------Auxiliary variables -------------------------
    set step 0
    set PORT 1886 // the port used between broker_sub_side and subscriber
    set Topic_name ACCIDENT // the topic that we want
    set missionId $id*1000 // for more security
    set nextstreet ST1
    set timer 100
    vec Events 150 // list of the data collected
    set count_Events 0
    set count_help 0
    set en 1
    //-------------------------------------------------------------------------------------
    //-----------------------------------Start the loop--------------------------------
    loop
    getpos2 lo la
    time t
    int cloock $t
    wait 500
    read rp
    if($rp != \)
    rdata $rp p rid len type
        if($p == $PORT)
    //------------------------------Confirmed the mission  -------------------------
            if($type == $MISSIONACK )
            set step -1
            delay 10
            end
    //-------------------------------------------------------------------------------
            if($type == $HELPACK)
            set count_help $count_help+1
            set en 1
            delay 10
```

```
            end
    //--------------Receive all the data collectd from broker sub side --------
            if($type == $END_MISSION)
            rdata $rp p rid len type Topic_n missionID Topic_id mssgId sensor_ID
        time_of_action LO LA last
                if($Topic_n == $Topic_name )
                    if($missionID == $missionId)
                        if($Topic_id != \)
                        data content $Topic_id $mssgId $sensor_ID $time_of_action $LO $LA
                        vset $content Events $count_Events
                        set timer $t+100
                        set d $cloock-$time_of_action
                        set sum $sum+$d
                        set en 1
                        inc count_Events
                        end
                        if($last == 1)
                        set timer $t+10
                        set step 0
                        data sp $p $id 4 $END_MISSIONACK place2
                        send $sp $rid
                        delay 100
                        end
                    else
                    cprint a naughty uav
                    set step -1
                    data sp $p $id 4 $END_MISSIONACK place4
                    send $sp $rid
                    delay 100
                    end
                else
                cprint a naughty uav
                set step -1
                data sp $p $id 4 $END_MISSIONACK place4
                send $sp $rid
                delay 100
                end
            end
//-------------------------------------------------------------------------------------
        end
    end
//----------------------------the time between the missions-----------------------
    if(($step == 0)&&($timer < $cloock))
    set step -1
    data sp $PORT $id 10 $MISSION $Topic_name $missionId $nextstreet
    send $sp
    delay 100
    end
    //-------------------------------------------------------------------------------
```

```
        for i $count_help $count_Events 1
        vget e Events $count_help
        vdata c $e
        vget lo1 c 4
        vget la1 c 5
        data sp $PORT $id 10 $HELP $lo1 $la1
        send $sp
        delay 100
        //set i $count_Events
        end
    delay 1000
//----------------------------------------Next Loop-------------------------------
//*************************************************************************************
```

11 Total vision in more grand scenario:

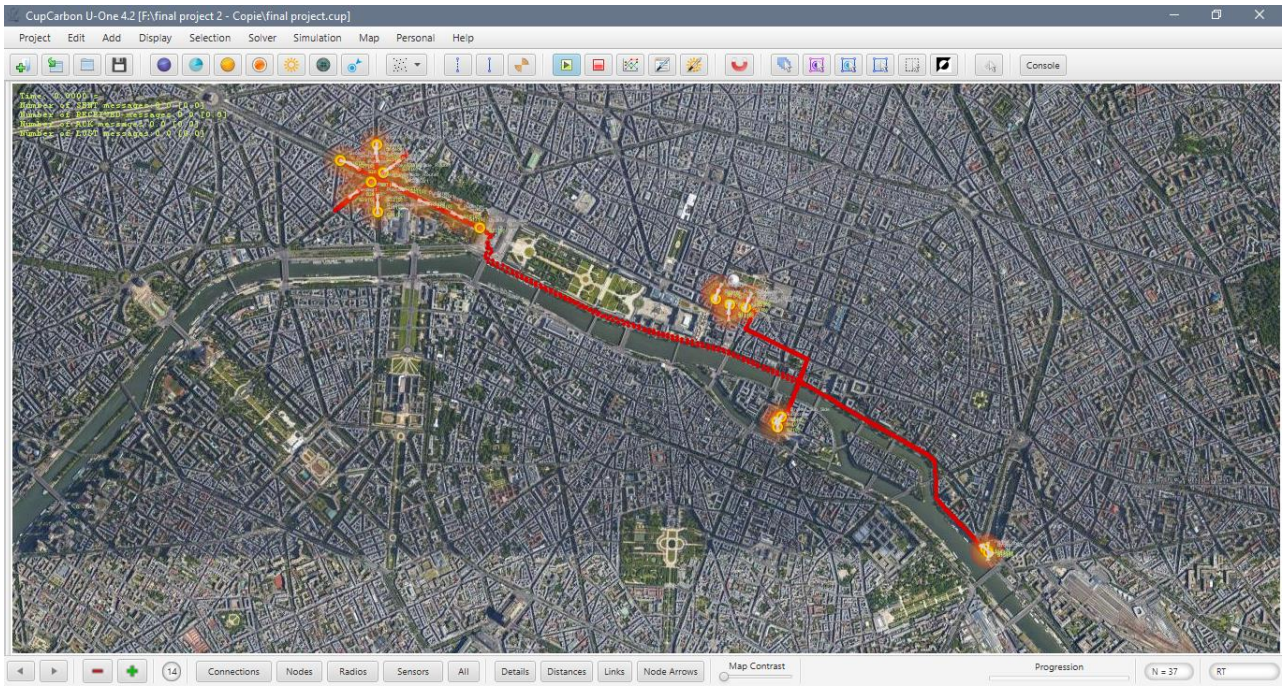

Figure 4. 12: Total vision of a more grand scenario.

12 Conclusion:

In this chapter, we have presented the evaluation context of our solution, as well as the obtained assessment results with Cupcarbon simulator. The results confirm that our solution is efficient enough and well adapted to smart city scenarios. Hence, the publish/subscribe communications with drone-based broker provide energy consumption, latency, and packet delivery ratio. In our simulations, we focused an optimised deployment scenario because we aimed to achieve good coverage of a given area (smart city) at low costs and fast network deployment.

General Conclusion:

Throughout this work, we have presented generalities about the IoT. Then, we have studied opportunistic and publish/subscribe communications, especially MQTT-based ones. Thereafter, we have presented our realized work that is about an effective solution that handles opportunistic and UAV-based broker for publish/subscribe communications in a smart city context. The evaluation results have confirmed the efficiency of our solution.

In fact, our solution can be useful for large amount of smart city scenarios, it could be also effective in case of a disaster as what happened with Beirut harbor explosion that cause huge damage on wide area of the city.

So, to recover communication with the harbor in fast time. We just should have installed an opportunistic communication, quite like the one we proposed, in order to ensure area coverage and exchange of sensitive data without any problem. This can be considered as practical use case for our application.

As perspective and in order to give an insight about the improvement directions that can be approached in our solution, we suggest the followings:

- Implement a dynamic QoS to ensure better latency.
- Create an algorithm for switching the channels.
- Integrate the ground sensor with publisher nodes.
- Integrate a real time GPS tracker for more security.
- Add a new architecture enabled by Software Defined Networking (SDN) to provide dynamic and flexible networking capabilities, to our application.
- Adopt a cooperative broker subscriber side to provide enhanced end to end delivery ratio.

List of references

[1] B. Dorsemaine, J.-P. Gaulier, J.-P. Wary, N. Kheir, and P. Urien, “Internet of things: a definition & taxonomy,” in Next Generation Mobile Applications, Services and Technologies, 2015 9th International Conference on. IEEE, 2015, pp. 72–77.

[2] Christian, Maeder.The History of IoT. February 24, 2017. Internetofthingsrecruitment.com/ The-History-of-IoT provided by FINDTECH .SINGAFORE. Available online at: https://iotnews.asia/92/iotinfographics/infographic-history-iot/

[3] Dr. Ovidiu Vermesan SINTEF, Norway, Dr. Peter FriessEU, Belgium, “Internet of Things– From Research and Innovation to Market Deployment”, river publishers’ series in communications, 2014.

[4] Klara Nahrstedt, H. L. (4-8 April 2016). Internet of Mobile Things: Mobility-Driven Challenges, Designs and Implementations. IEEE (pp. 25-36). Berlin, Germany: University of Illinois Urbana-Champaign.

[5] Pallavi,Sethia ;Smruti,R.Sarangi. Internet of Things: Architectures, Protocols, and Applications. 2017, Article ID 9324035, 25 pages.

[6] H. Ning and Z. Wang, “Future internet of things architecture: like man kind neural system or social organization framework?” IEEECommunicationsLetters,vol.15,no.4,pp.461–463,2011.

[7] Leila, Fatmasari Rahman;Tanir, Ozcelebi;Johan, Lukkien. Understanding IoT Systems: A Life Cycle Approach. Procedia Computer Science.2018.vol 130 .1057–1062.

[8] avsystem.com. iot-technology.2006. Available online at <://www.avsystem.com/blog/iot-technology/> consulted at 24/01/2020.

[9] Yusuf, Perwej; Mahmoud Ahmed ,AbouGhaly; Bedine ,Kerim; Hani Ali Mahmoud ,Harb. An Extended Review on Internet of Things (IoT) and its Promising Applications.Foundation of Computer Science FCS, New York, USA Volume 7– No. 26, February 2019 . page 22. Available online at< www.caeaccess.org >.

[10] Thomas ,Lagkas ;Vasileios ,Argyriou;Stamatia, Bibi ;Panagiotis ,Sarigiannidis. UAV IoT Framework Views and challenge: Towards Protecting Drones as “Things”.17 November 2018. 21 pages . Available online at < https://www.mdpi.com/journal/sensors>.

List of References

[11] Mustafa ,Kocakulak ;Ismail ,Butun. An Overview of Wireless Sensor Networks Towards Internet of Things.IEEE.2017.31 pages.

[12] Harshvardhan , Mishra. IoT Communication Models . iotbyhvm .09 October 2018. . Available online at <https://iotbyhvm.ooo/iot-communication-models/ >.

[13] Keyur ,K Patel; Sunil, M Patel;PG ,Scholar; Assistant ,Professor. Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges. Department of Electrical Engineering Faculty of Technology and Engineering-MSU, Vadodara, Gujarat, India.2016.vol 6.page 6122- 6131.

[14] John, Wilry; Sons ,The Atrium Southern Gate Chichester, West Susses. IoT Security For Dummies . INSIDE secure Edition pari.2016.53 page.

[15] S. U. Khan, "Mosaic-Net: A Game Theoretical Method for Selection and Allocation of Replicas in Ad Hoc Networks,"Journal of Supercomputing, vol. 55, no. 3, pp. 321-366, 2011.

[16] Jean Femia . introduction to the publish/subscribe communication. OptoPartner/Distributor. Chicago.Apr 17, 2018 2:00:00 PM.

[17] medium.com. M2M communication protocols MQTT. Available online at <https://medium.com/predict/an-era-of-iot-m2m-communication-protocols-mqtt-e68d81b93613> consulted at 2/02/2020.

[18] microsoft.com .publish subscriber. 07/14/2010 .https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649664(v=pandp.10). consulted at 2/02/2020.

[19] arothuis.nl. Director of page web :Alex Rothuis<https://arothuis.nl/posts/messaging-pub-sub/> consulted at 2/02/2020.

[20] hivemq.com. Written by The HiveMQ Team. Publish & Subscribe - MQTT Essentials.in January 19, 2015.

[21] AL-FUQAHA, Ala et al. Internet of things: A survey on enabling technologies, protocols, and applications. v. 17, n. 4, p. 2347–2376, 2015.

[22] Valentina E. Balas, Vijender Kumar Solanki, Raghvendra Kumar , Md. Atiqur Rahman Ahad . A Handbook of Internet of Things in Biomedical and Cyber Physical System. Raghvendra Kumar Department of Computer Science and Engineering LNCT Group of College Jabalpur, Madhya Pradesh, India. volume 165 .2020. ISBN 978-3-030-23983-1.

List of References

[23] LOCKE, Dave. Mqtt v3. 1 protocol specification. International Business Machines Corporation (IBM) and Eurotech, p. 42, 2010.

[24] STANFORD-CLARK, Andy; TRUONG, Hong Linh. Mqtt for sensor networks (mqtt-sn) protocol specification. v. 1, 2013.

[25] Andrew Chapman is the NSW Director of Operations for Australian UAV , the Australian DRONE magazine, issue 3 (June 2016)specialising in aerial mapping, survey and inspection work since 2013. <www.auav.com.au>.