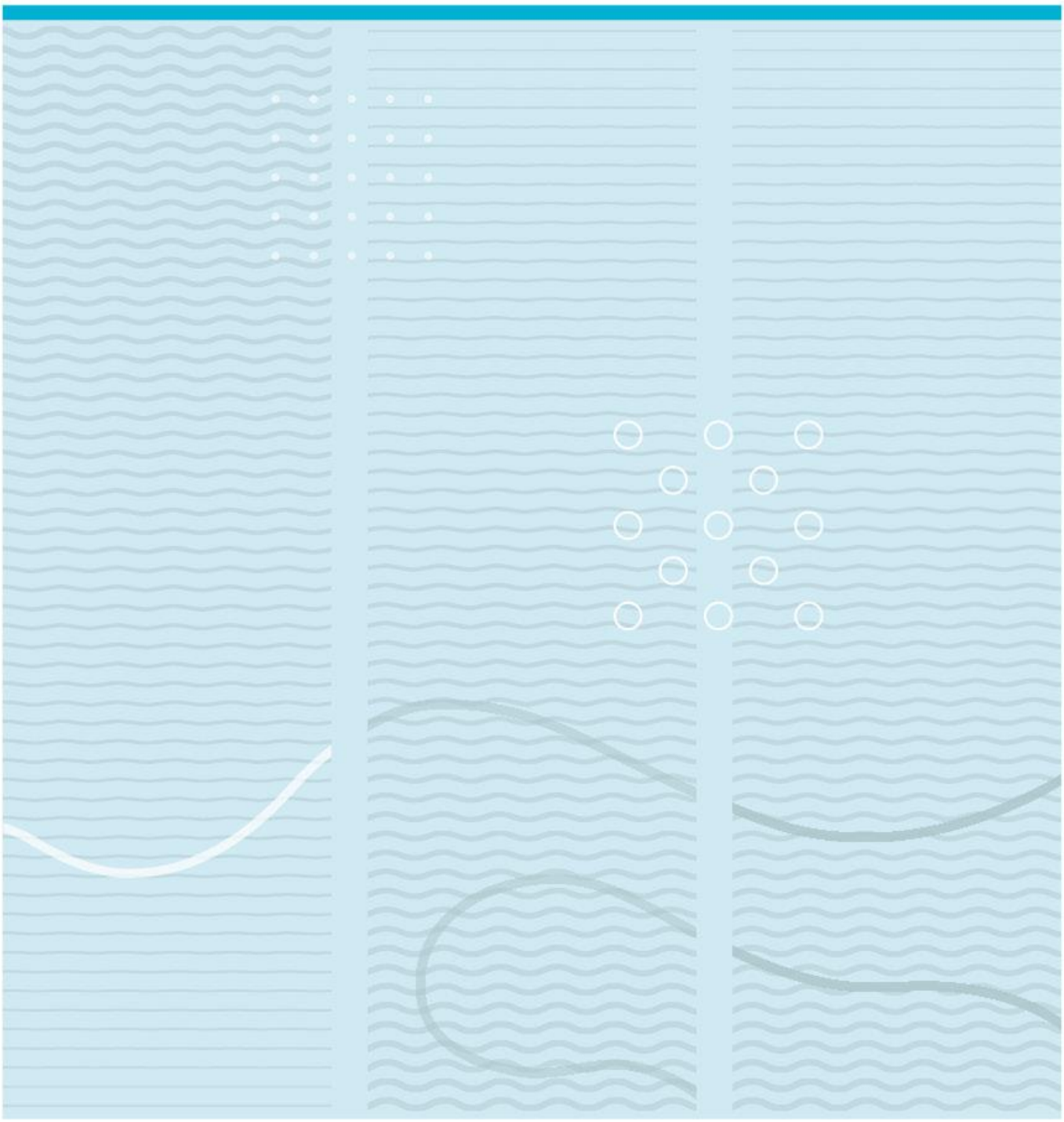
University College of Southeast Norway
Faculty of Technology and Maritime Sciences
–
Master Thesis in System Engineering with Embedded Systems
Kongsberg Department of Engineering
10 June 2021

SANDEEP SHIVAKOTI

UAVs communication over the Internet Of Things (IoT)



University of South-Eastern Norway
Faculty of Technology and Maritime Sciences

Institute of Technology and Engineering
PO Box 235
NO-3603 Kongsberg, Norway

http://www.usn.no

This thesis is worth 30 study points

Contents

Summary

Unmanned Aerial vehicles and Autonomy have gained much popularity in both industry and academia. Moreover, with the advancement in battery quality and sizes, progress in a communication system from mere short-range Radio Frequency to current 5G and significant improvement in global positioning system; have revolutionalized improvements in the commercialization of drones. However, constantly communicating with and monitoring multiple drones is a potential concern, mainly in denser aerial activity areas. Therefore, drone management urges a highly reliable communication network to maintain a seamless information exchange.

This research is mainly concerned with establishing constant and reliable communication of multiple drones with the web-based control station. Furthermore, implementing and establishing communication using IoT protocol within a cellular network, primarily within urban airspace. More than one base station might constantly monitor all the drones requesting take-off or which are in the air based on traffic. Simulation handling multiple drones is carried out, demonstrating the proposal's feasibility. Different use cases and scenarios concerned with the safety and security of GDPR are primary concentric points in maintaining the regulations. The simulation results show that MQTT can handle multiple drones within a network; four drones in simulation over a single network. Widgets in the thingsboard provide the visual realization of the drone's current position and parameters.

Foreword

First of all, I would like to extend my sincere and hearty gratitude to my supervisor Dr. Aurilla Aurelie Arntzen, and co-advisor, Dr. Serkan Güldal, for their guidance and inspiration through the thesis. I am thankful for all the meetings and valuable feedbacks, making the learning and development process much more interactive and effortless. I am also grateful to Dr. Aurilla Aurelie Arntzen for her suggestion of this topic, which I enjoyed working on and learned a lot from. I also would like to thank our head of department, Dr. Elisabet Syverud, for her constant efforts in making the labs accessible to students.

I would also like to remember all the Professors in school for their efforts, guidance, and their availability to students. Last but not least, I would like to thank my friends and colleagues at the University of South-eastern Norway (USN), Kongsberg, especially Mr. Biplav Karna, who introduced me to Python and the Linux Operating system, Tushal Shivale, who suggested me to use Thingsboard IoT platform.

Sandeep Shivakoti

Kongsberg, Norway, 10th June 2021

List of Abbreviations

IoT- Internet of Things

GPS- Global Positioning System

GNSS- Global Navigation Satellite System

VTOL- vertical take-off and Landing

BVLOS-Beyond Visual Line of Sight

VLOS- Visual Line of Sight

USAF- United States Air Force

GLONASS- Globalnaya Navigatsionnaya Sputnikovaya Sistema

FRPA- Fixed Reception Pattern Antenna

UAS-Unmanned Aircraft Systems

UAV-Unmanned Aerial Vehicle

GDPR- General Data Protection Regulation

WWI – World War I

UTM - Unmanned Traffic Management

VoLTE - Voice over Long-Term Evolution

M2M - Machine to Machine

RS - recommended-standard

PHY - Physical Layer

MAC - Medium access control

FFD - Full-function device

IETF - Internet Engineering Task Force

CON - Confirmable

NON - Non-Confirmable

ACK - Acknowledgment

RST – Reset

SD - smart dust

MEO - medium Earth orbit

Chapter 1

1. Introduction

Figure 1 Cargo Drone[1]

Unmanned Aerial Systems have been around much longer than most of us realize. UAS trace their modern origin back to the development of aerial torpedoes almost 95 years ago [2]. The Unmanned Aerial System was first designed for surveillance by the USA military. During WWI, both the USA Navy and the Army experimented with aerial torpedoes and flying bombs with very little success. Since then, some of the world's best minds have made their lifetime contributions to improving aerial technology. UAS has now gained a more comprehensive range of popularity in different sectors besides surveillance, which drew most of the focus. Some of the vastly affected areas include package delivery [3], photography [4-6], agriculture [7, 8], disaster management [9]. The shift in focus can be traced back to advancements in other sectors, including battery technology, improvement in motor power concerning sizes, communication advancements, GNSS positioning accuracy, metallurgy, etc., which help design low-weight chassis and improve other different factors affecting drone flight. Significant improvements in technology greatly revolutionized different sectors of the market. Therefore, advances in drone technology initiated a chain reaction revolutionizing other sectors along with it.

However, with the increasing air traffic density, security and safety have become the government's primary concern and the drone developing industry. Therefore, the government laid down drone regulations and their operability in the sky for safer operations. These regulations are country or state-dependent. The European Union also has commissioned these regulations under the "European Drones Regulation" [10]. Also, a wide range of private sector companies coordinating with the EU has joined hands together to make the sky safer. EuroDRONE [11] is an Unmanned Traffic Management (UTM) demonstration project funded by the EU's SESAR organization, and it aims to test and validate key UTM technologies for Europe's 'U-Space' UTM program. The proposed EuroDRONE system is a Highly Automated Air Traffic Management System for Small UAVs Operating at Low Altitudes. Moreover, major countries like the USA and Japan have their UTM program as NASA UTM [12, 13] and Japan as JUTM [14].

1.1 Problem Statement

The utility of drones has increased drastically. Grand View Research report forecast an estimated revenue of 129.23 billion USD, an approximate CAGR of 56.5 %[15]. Therefore, the traffic density of drones is expected to increase in terms of hundreds of thousands over a few years.

Hence. a constant and reliable communication system is of utmost importance. Drones primarily utilize a radio frequency spectrum between 900 MHz and 5.8 GHz[16] for communication. Drones with 2.4 GHz are enabled with live video streaming with a maximum range of 1-4 miles over a VLOS.

This research attempts to bridge a gap between the IoT and the drone operation specific to communication establishment. It gives a better picture of implementing the communications in drones beyond the VLOS. It tries to ensure constant and reliable communication over a web-based IoT server.

The thesis focuses explicitly on the IoT protocol -MQTT to establish bidirectional communication between a web server that acts as a hub and all the drones within the network. All the messages are routed to the respective drones through the server.

"Can IoT protocol- MQTT be the future of drone communication?"

1.2 Context

More and more drones are being inducted into commercial and other public sectors, increasing the drone traffic density exponentially. However, one thing in common is the technology of communication exchange between the drones and the Ground Control Station. They use radio frequencies- 900MHz and 2.4GHz for real-time video streaming and message exchange. Moreover, these drones communicate in point-to-point nature, i.e., a single user controls a drone at a time. Flying multiple drones by a single user is almost impossible.
This thesis uses a WiFi network with a presume maximum drone flight altitude of 300m to establish simulated communication. Therefore, it facilitates the possibility of controlling drones beyond the VLOS or a drone in any part of the world within a 4/5G network.

Moreover, the messages are routed from all the drones within the network to a web server. It provides the opportunity to monitor the real-time position of drones. Since MQTT supports bidirectional communication, it allows dynamic routing of drones in case of any chances of collision. A Webserver, as a hub, is used to relay the commands and messages to the respective Drone and Ground Control Station, if any. Hence, a drone can either publish a message or subscribe to a particular message type, thereby reducing message filtering.

1.3 Existing problem

There is a wide variety of UAS taking to the sky. The most common point in all the UAVs is that they utilize Radio Frequencies as the primary means of information exchange between the drone and the Ground Control Station. Globally, these frequencies vary over a wide range of few kilohertz to 59 GHz - 64 GHz [17]. Radio communications modules like WiFi, Bluetooth, Zigbee, and LoRa are the most common forms of RF connectivity. However, based on the modules used, the operational range and security are highly affected.
The operational range is a significant challenge that UAVs face in radio communication. For instance, WiFi has an operation range of 46 m indoor and 92 m outdoor. Bluetooth operates in a range of 10 m. Zigbee has 10-75 m of operation, while LoRa works in a range of 50 m indoors and 165 m outdoors. All these communications in the maximum fields are at the VLOS. The communication range is dramatically affected by objects between the LOS.

Another challenge is the security of the system itself. The plans are designed primarily for point-to-point communication. Very few securities are incorporated, making it very vulnerable to the system safety itself. Only a few military-based drone/RPAs are registered with Air Traffic Control. While small recreational and non-commercial drones need not be registered and need no prior permission to fly. However, European Union Aviation Safety Agency (EASA) has set a strict framework within EU and EASA member countries to fly drones in the European sky.
Since the drones taking to the sky are neither registered nor monitored. Therefore, it is almost impossible to know the identity of any drone flying in the sky. For security reasons, it becomes vital that all the manned and unmanned machines be closely monitored and kept track of.

Problems

1. Limited range of operation.
2. Point-Point communication.
3. Identification of any flying drones.
4. Path planning.
5. Data collection beyond VLOS.
6. Limited/No take-off and Landing permissions

1.4 Benefits and Drawbacks

The availability of multiple communication protocols provides broad flexibility in implementing communication in different applications and scenarios. For short-range indoor and outdoor communication, fewer overheads need to be considered compared to long-range. Moreover, when the long-range communication further extends to BVLOS. It entirely changes the whole picture in the communication system. Communication BVLOS provides a more fantastic connectivity range, however, with a high cost. Some of the advantages and disadvantages of IoT protocols BVLOS are listed

1.4.1 Benefits

- Manageable data cost
- The real-time operation BVLOS.
- Lightweight
- Secured and ensured message delivery
- Longer battery life
- Carry payload of any data type

1.4.2 Drawbacks

- Low data transmission
- No live video streaming
- Centralized broker failure results in complete system failure.

1.5 Target UAV

There are no standards defined to classify drones. Hence, there are broad parameters to be considered in the classification. Drones are categorized in different aspects commonly based on weight[18, 19], weight and flight time [18] , size [18, 20]s, range and endurance [20], aerodynamics [19], landing types[19] , rotors[19]and applications[19] by different authors. M. Hassanalian and A. Abdelkef in [18] have classified UAVs in an unconventional category of drones, including drones with a maximum wingspan of 61m and weight of 15,000 kg to SD with a minimum size of 1mm and weight of 0.005g.

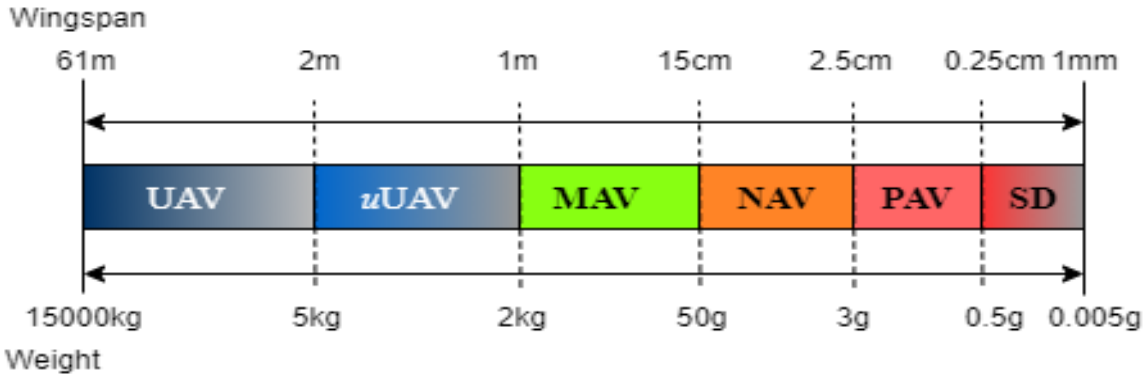


Figure 2 Spectrum of drones from UAV to SD

Figure 2 [18] represents the broad spectrum of drones from UAV to SD. In general, all the drones fall under this category, although the categorization parameters could be different.

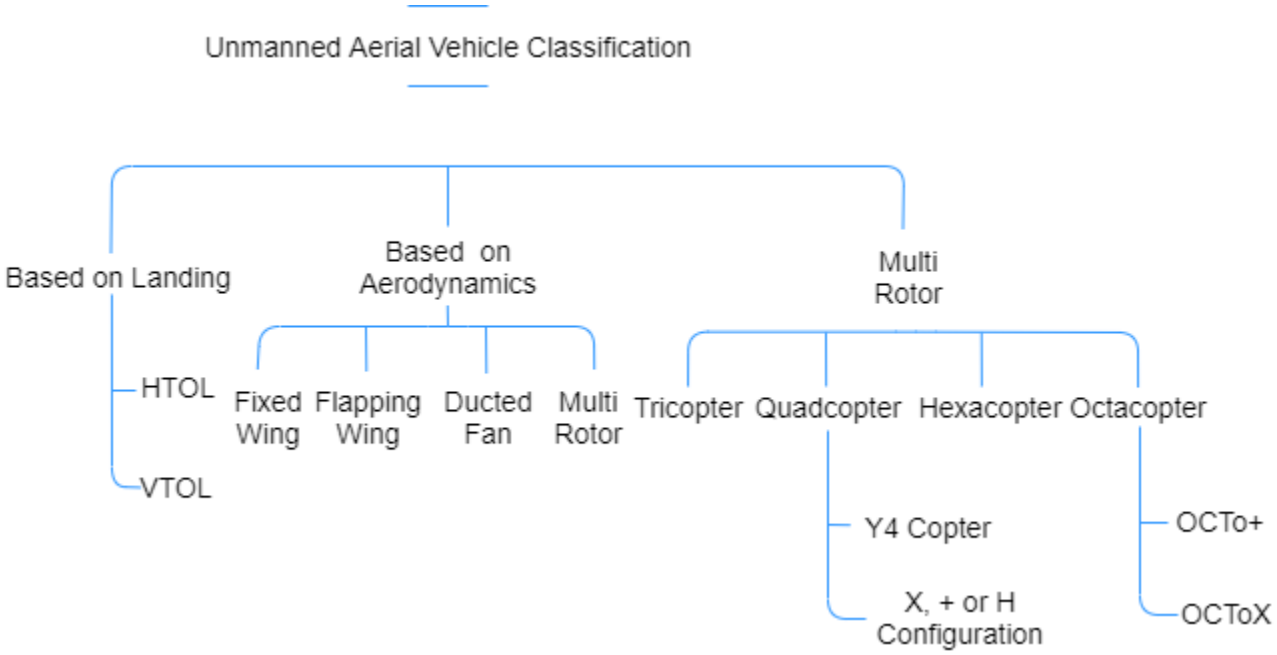


Figure 3 Classification of UAV based on Landing, Aerodynamics, and rotors

Figure 3 [19] represents the classification of drones based on Landing, Aerodynamics, and weight. In consideration of drones' application as cargo, figure 3 best fits the categorization. The drone in an urban setting is expected to have a short or no runway to be airborne; therefore, VTOL best fits the requirement. Moreover, aerodynamics forces are solely generated in multirotor by rotors with the best fit in X, +, or H configuration.

Selected Drone for the project

1. Categorization: Landing, Aerodynamics, and Rotors.
    a. Landing: VTOL
    b. Aerodynamics: Multirotor
    c. Rotor: Quadcopter

d. Quadcopter: H, + or H configuration

2. Application: Cargo/Delivery drone
3. Weight: 1kg ≤ Drone ≤5kg

1.6 State of the Art

There is an ever-growing need to connect small devices to the internet, collecting the data at a shared server acting as a hub. These data from the sensors are relatively small and do not require a broad spectrum to push the data to the cloud-based server. Similarly, for close monitoring of the drone, i.e., the real-time location and various onboard sensors reading, the data size is relatively small on a few kilobytes scale. Therefore, MQTT, a lightweight IoT protocol, can push the drone data to the webserver over a narrow bandwidth.

Drone highway: It is an imaginary highway in the air. It is pretty analogous to terrestrial routes. Except for they are aerial.

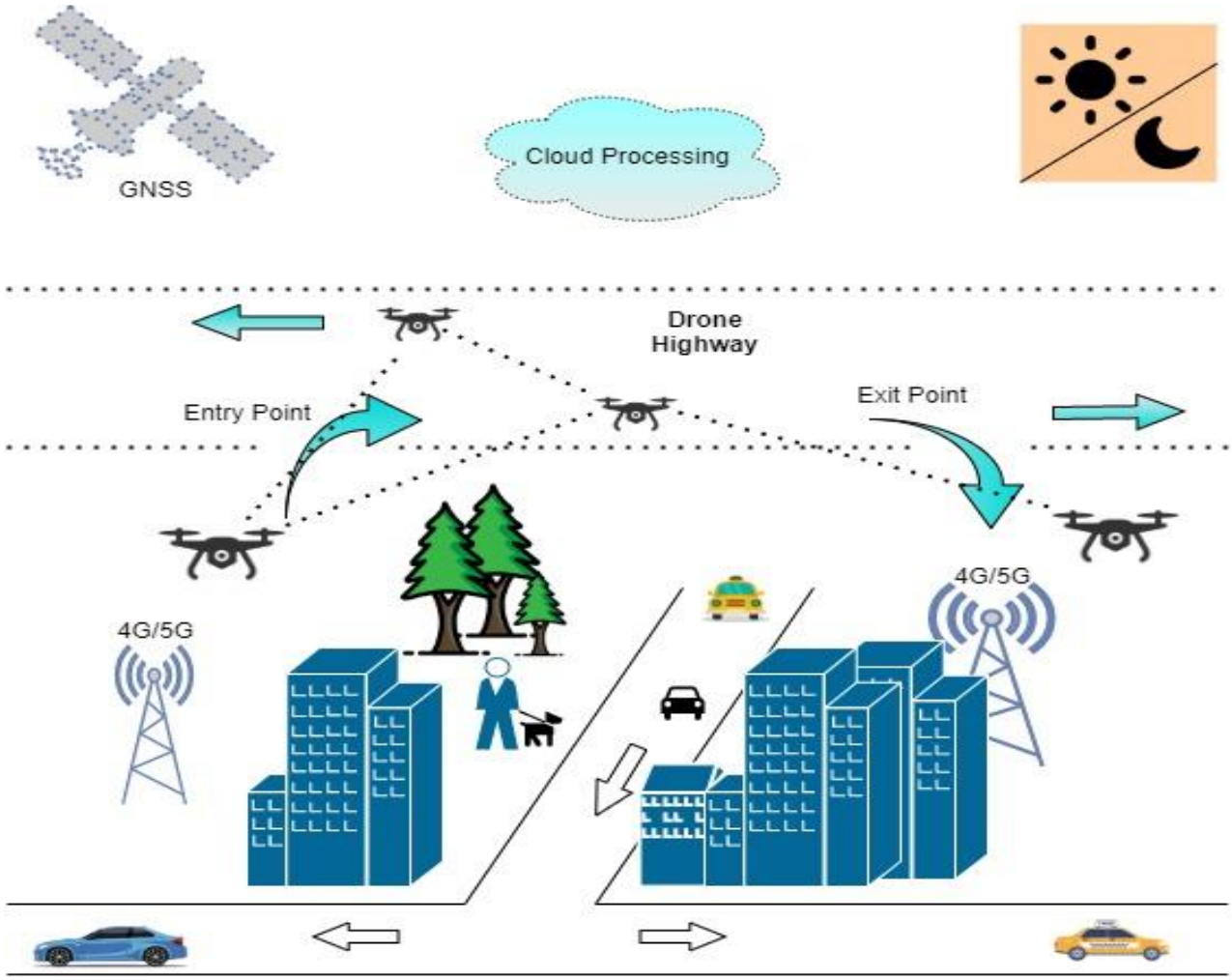


Figure 4 ConOps

Figure 4 represents the concept of operation for the proposed design concept. The figure shows drones following the drone highways, also the drones entering and leaving the drone highways. The whole system is backed with a 4/5G network.

1.7 Expected Result

- Real-time tracking of drones BVLOS
- Enable dynamic routing.
- Handle high traffic density.
- Monitoring and authentication of drones

Chapter 2

2. IoT Protocols and Issues

The introduction of wireless communication and the first radio transmission 2000 miles across the Atlantic ocean by Italian physicist Guglielmo Marconi in 1901[21] gave birth to a blueprint to modern wireless long-range communication. A drastic shift to digital communication systems from analog led to higher data transmission over the same communication channel. Indoor and Outdoor communication protocols ranging from few meters to several kilometers gradually led to the notion of M2M development connecting remote devices wirelessly. However, M2M communication mainly concerns connecting point-to-point devices, thereby restricting data sharing, which forces the shift into IoT, making devices central to an IoT platform, enabling data sharing. Figure 5 represents different wireless IoT communication protocols, both indoor and outdoor, with featured coverage.

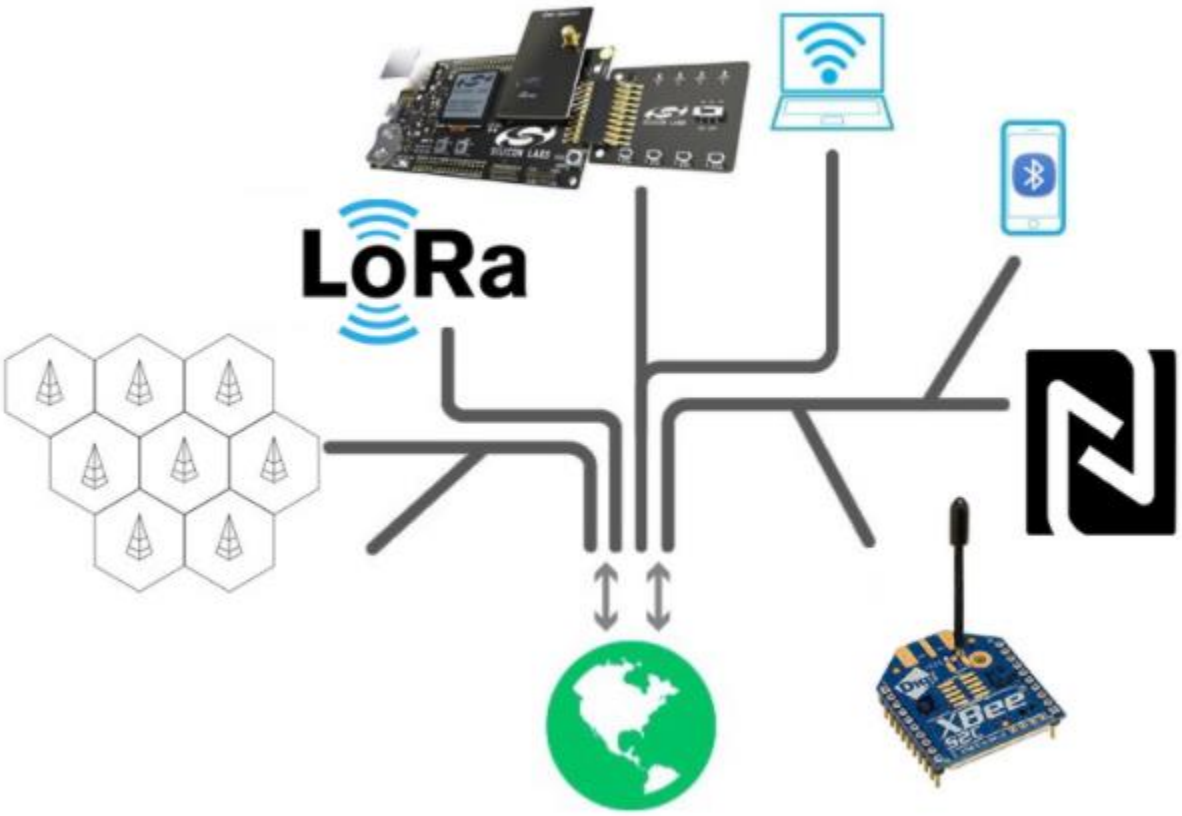


Figure 5 Wireless Communication Protocols in IoT[22]

2.1 IoT Protocols

Wireless communication is the most common and widely used communication system for small and remote devices. A wide variety of Radio modules facilitate this feature. Some of them are listed below.

2.1.1 Bluetooth

"Bluetooth is a short-range, low-power IEEE open standard for implementing wireless personal area networks" [23]. Bluetooth operates in the globally unlicensed 2.4GHz short-range radio frequency spectrum meant for short-range communications. Being the most commonly used radio frequency protocol, Bluetooth also has a wide range of applications in IoT. Bluetooth defines a radio interface and allows the devices to discover each other within the vicinity.

Figure 6 Bluetooth

In 2003 with its limited range and small data rate, Bluetooth technology was on the verge of the dead[24]. However, with its wide applications connecting devices such as audio communications and stereo streaming, Bluetooth highly regained its popularity. Bluetooth was mainly popular in connecting short-range devices. However, with the introduction of BLE in Bluetooth 4.0. It has gained much popularity in IoT and Machine-to-Machine(M2M) communication. Nevertheless, now the new challenge is reducing the power consumption for battery-powered devices for an extended period. With the introduction of Bluetooth 5.0, a wide range of distances can be achieved at the expense of low data transmission. However, before the launch of Bluetooth 5.0, the only way to increase the scope of its BLE product was by increasing the transmit power.

2.1.1.1 Bluetooth data frame

Bluetooth supports two types of data frame formats[20]; the basic frame format carries a lower payload with a maximum baud rate of 1Mbps, while the enhanced frame format supports a maximum of 2-3 Mbps depending on the modulation technique incorporated. The main difference between the two formats is the payload and the additional fields in the enhanced frame format. Since the enhanced frame format supports up to 3Mbps, the Guard/sync field synchronizes the two data-sharing devices when the transfer rate is switched while the trailer field identifies the end of the data field.

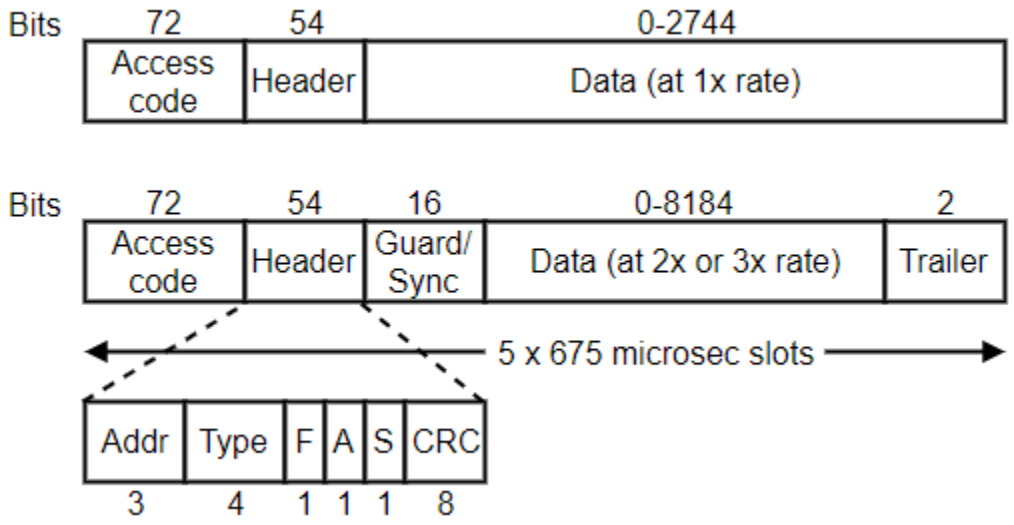


Figure 7 Bluetooth Frame Format at Basic and Enhanced rates

2.1.1.2 Network Architecture

2.1.1.2.1 Piconets

Piconets[25] are small Bluetooth networks consisting of more than two Bluetooth-enabled devices at the most eight nodes forming a PAN network. In a Piconet of 8 nodes in figure 8[25], one acts as a master while the rest of the seven nodes act as slaves. No two slaves can independently communicate with each other. The master as a primary station acts as a hub that manages the communication within the network. All the information is routed through the master node. Communication between a master and a slave can be either one-to-one or one-to-many. Besides the seven active slaves in a Piconet network, there can be up to 255 parked nodes. [26]However, parked nodes are in the currently active network in a dormant state. They can listen to the beacons from the master and rejoin the network as an active state. The node is assigned an address before it attains the active slave state and gives up its 8 bit parked member address.

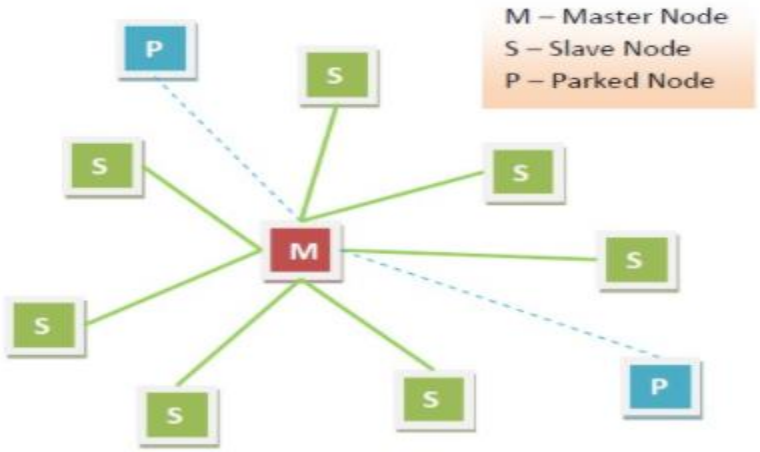


Figure 8 Piconet

2.1.1.2.2 Scatternet

A scatternet[27] is a network of two or more interconnected Piconet networks. A piconet node within a scatternet network, either a master or a slave, acts as a slave to another piconet in the same scatternet network. This node acts as a bridge between the two piconet networks and is called a bridge node. Hence, if there are one or more bridge nodes in a piconet network, the overall network is a scatternet. The bridge node relays the data between the neighboring piconet networks. Therefore, bridge nodes play a crucial role in information exchange and the formation of the scatternet.

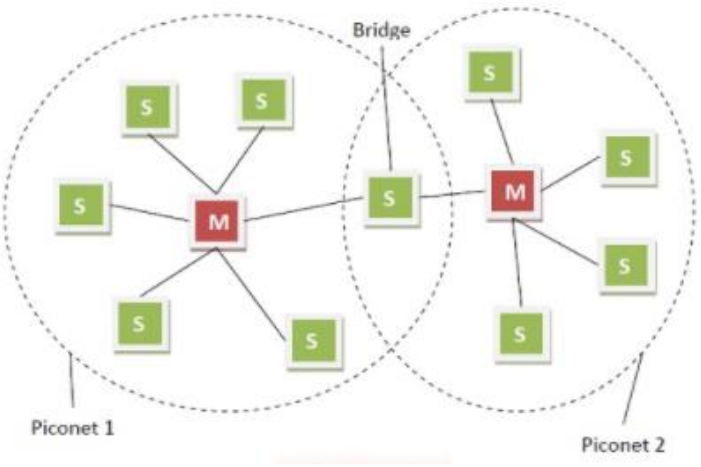


Figure 9 Scatternet

2.1.2 Zigbee

IEEE 802.15.4 committee started low data rate ZigBee technology and joined forces with ZigBee Alliance to further development[28]. ZigBee malleable devices can transmit data over a range of 10-75metres in 2.4 GHz frequency (915MHz Americas or 868 MHz Europe). The data rate is 250kbps at 2.4GHz, 40kbps at 915MHz, and 20kbps at 868MHz[28]. Figure 10 represents the ZigBee symbol.

Figure 10 Zigbee

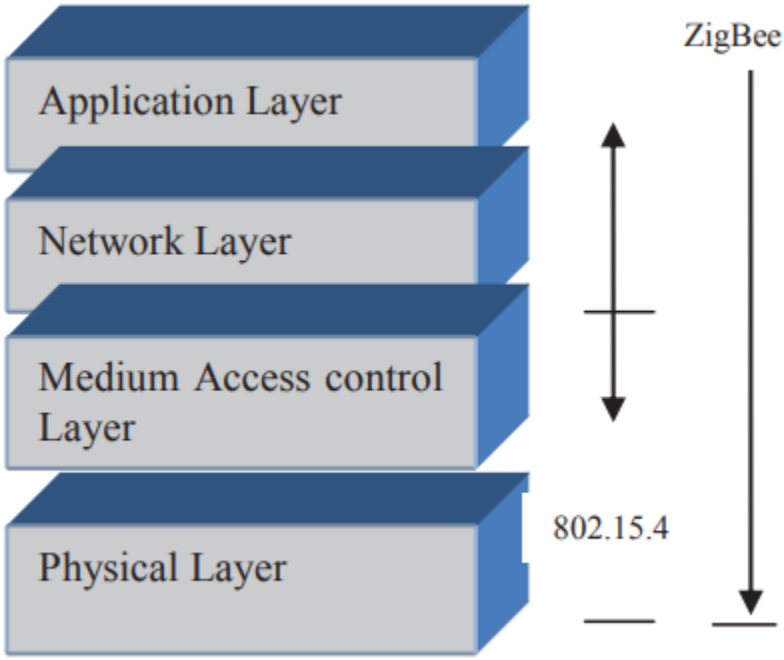


Figure 11 ZigBee Protocol stack

OSI model captures the ZigBee Protocol stack in figure 11[29] that occupies four layers. IEEE 802.15.4 defines the PHY layer, MAC layer and network layer, and Application layer by ZigBee Alliance[29].

2.1.2.1 Network Architecture

ZigBee facilitates three different network topologies; Star, Peer-to-Peer, and cluster.

2.1.2.1.1 Star Topology

Start topology is the basic topology where the central node, called PAN controller, and multiple end devices are in star formation. The coordinator routes any message between the two end devices. A star topology is easy to implement, and it takes only two hope to route the message to any end device. However, it creates a bottleneck, increasing the load on the coordinator.

When powered up, a device may establish its network and become the PAN coordinator in FFD mode. A device in a star network can select a PAN identifier currently unused by any other network within the vicinity. A unique PAN identifier allows each star network to operate independently[28].

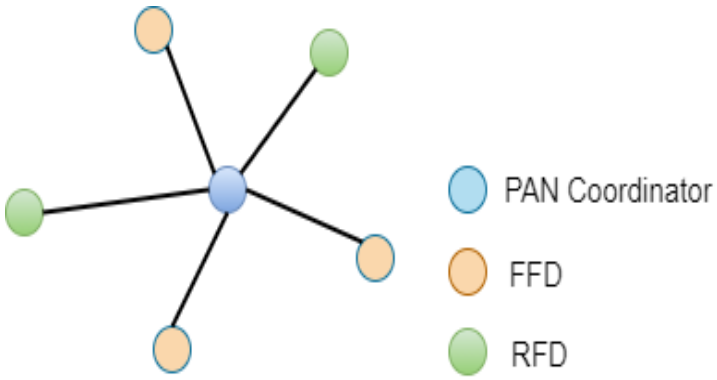


Figure 12 Start Topology

2.1.2.1.2 Peer to peer Topology

The device connected can directly communicate without a coordinator in peer topology, although at least one coordinator is present. "A peer-to-peer network can be ad hoc, self-organizing, and self-healing" [28]. A peer-to-peer topology is more demanding to maintenance; however, it is more robust and fault-tolerant. It supports multi hoping to route the message to the target device.

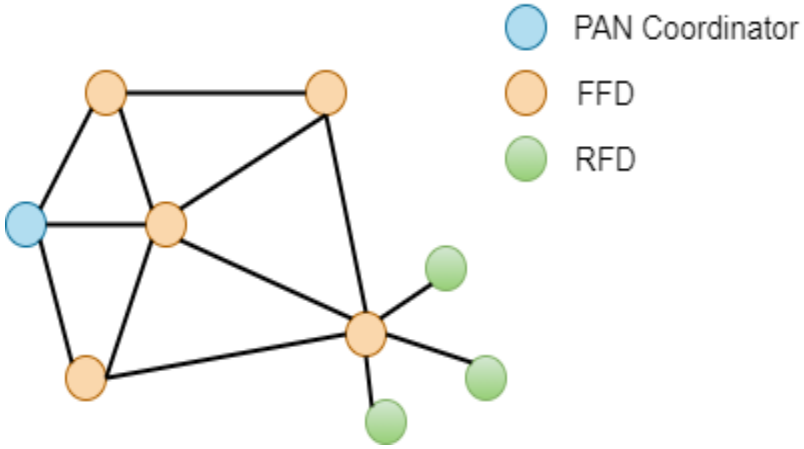


Figure 13 Peer-to-peer Topology

2.1.2.1.3 Cluster-tree Topology

A cluster-tree topology is a combination of two or more peer-to-peer topologies. It consists of FFD and RFD, while FFD outnumbers RFD in the network. Only an RFD device has the power to add a new peer-to-peer network to the existing cluster-tree network. "Any of the FFD can act as a coordinator and provide synchronization services to other devices and coordinators. However, only one of these coordinators is the PAN coordinator" [28]. As the network increases, the number of messages hoping increases accordingly.

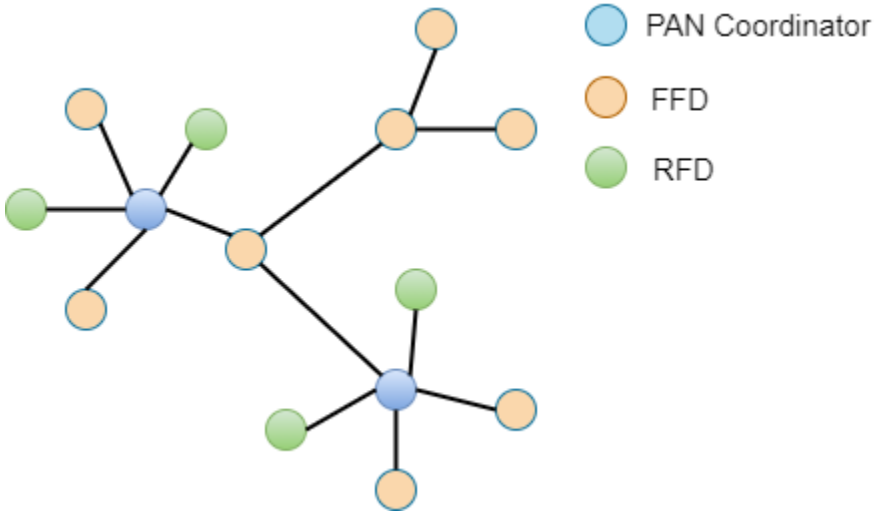


Figure 14 Cluster topology

2.1.3 LoRaWAN

2.1.3.1 Network Architecture

A typical LoRaWAN network consists of end-devices commonly known as motes, gateways, and servers. Motes are sensors and actuators which collect data and send it to the server via a gateway. "The network topology is a "star of stars," which means that groups of motes are connected to the gateway via LoRa wireless links while the gateways connected to a remote server via IP network" [30]. Three different classes[30] categorizes a mote.

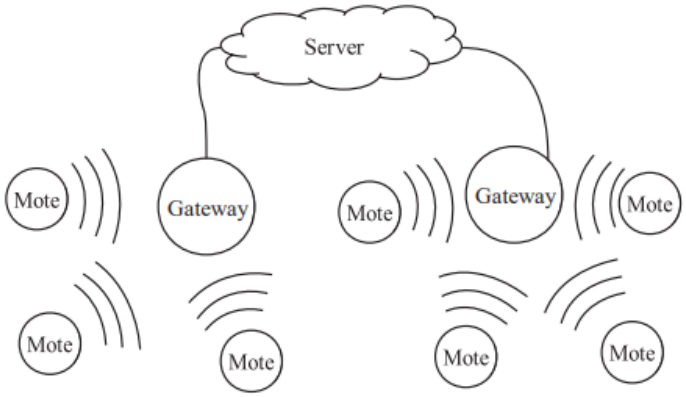


Figure 15 LoRaWAN Network Topology

2.1.3.2 Data frame

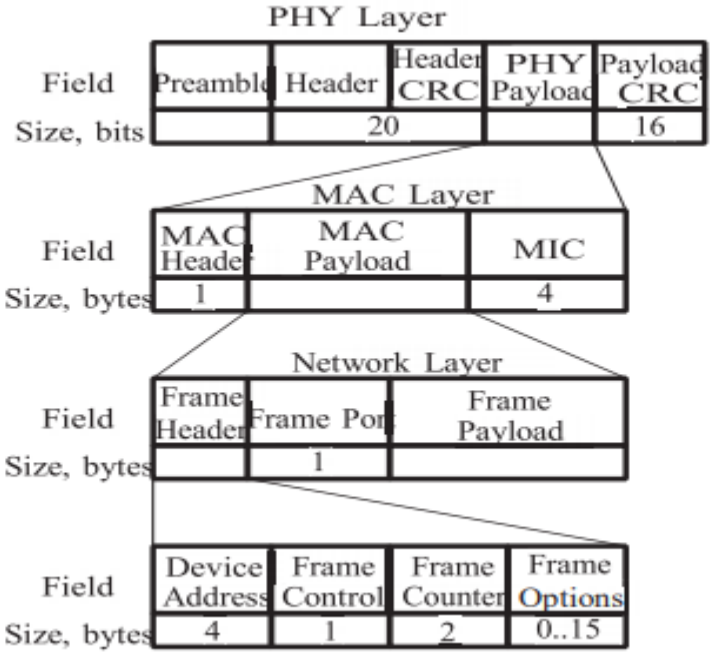


Figure 16 LoRaWAN Frame Format

Figure 16[30] represents the LoRaWAN frame format. LoRaWAN is a lightweight protocol designed for low-powered and low-rate communication. In the PHY layer, the LoRaWAN frame starts with a preamble that synchronizes the transmitter and receiver. Moreover, it also defines the data packet modulation scheme, being modulated as the rest of the packet. A header field follows the preamble, identifies the payload length in bytes, payload, and payload CRC[30, 31].

2.1.4 CoAP

CoAP was made a full IETF Internet standard in 2014 officially[32]. "CoAP is a specialized web transfer protocol Analogous to HTPP designed for use with constrained nodes and networks in the IoT. The protocol design is extremely M2M application-oriented to deliver low data packets."[33]. Similar to HTTP, CoAP works on a request/response approach.

2.1.4.1 CoAP message exchange

Message exchange in CoAP can broadly be classified as messaging in two layers, namely message layer and Request/Response layer[34].

2.1.4.1.1 Message layer

CoAP supports four message types that identified the state of sent/receive messages. The message type includes:

1) Confirmable

2) Non-Confirmable
3) Acknowledgment
4) Reset

2.1.4.1.2 Message delivery

However, the message delivery type can be divided into two basic types

2.1.4.1.2.1 Reliable Message

CON marks a reliable message. In a message, the sender also sends a confirmable message followed by the data message. The CON message is sent by default timeout and exponential back-of until an acknowledgment is received. On message received, the server on the other end sends an ACK in response to the confirmable message type. Each message has a unique id for redundancy check. A reset message replaces ACK if a reply is unsuccessful. Figure 17 [34] represents a reliable message mode with CON and ACK messages.

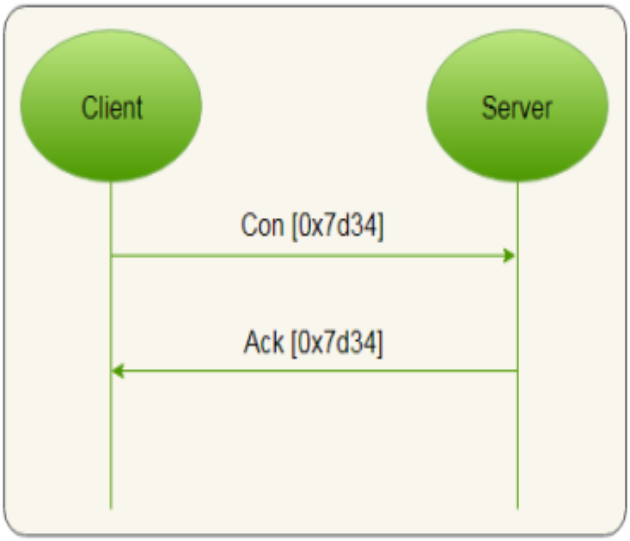


Figure 17 Reliable Message

2.1.4.1.2.2 Unreliable Message

An unconfirmable message is unreliable. A message in an unreliable type is not acknowledged in any way. However, the messages themselves carry a unique identification code that checks for any redundancy in messages. A reset message is sent to the client if the recipient fails to respond to the non-confirmable message. Figure 18 [34] represents an unreliable message mode with no acknowledgment.

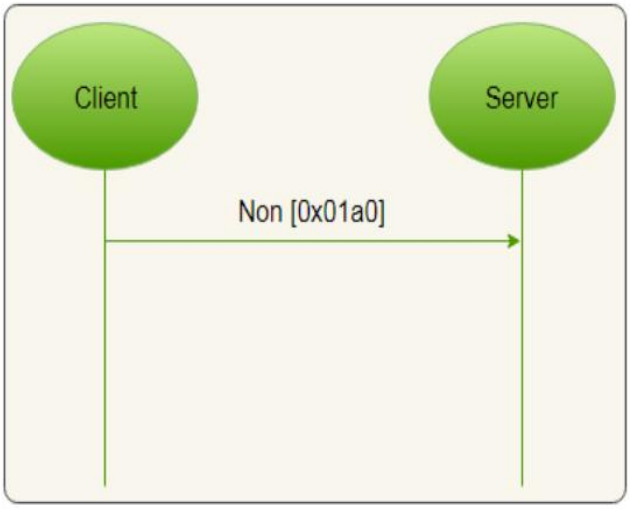


Figure 18 Unreliable Message

2.1.4.1.3 Request/Response Layer

Messages are divided in the request/response layer into two types:

1) Piggybacked
2) Separate
3) Non-Confirmable

2.1.4.1.3.1 Piggybacked

When a client sends a confirmable message, if a response immediately follows the message, the response message is in the acknowledge message that acknowledges the message request. It is known as a piggybacked response [34]. Figure 19[34] represents Successful and Fail Piggyback Messages from the server.

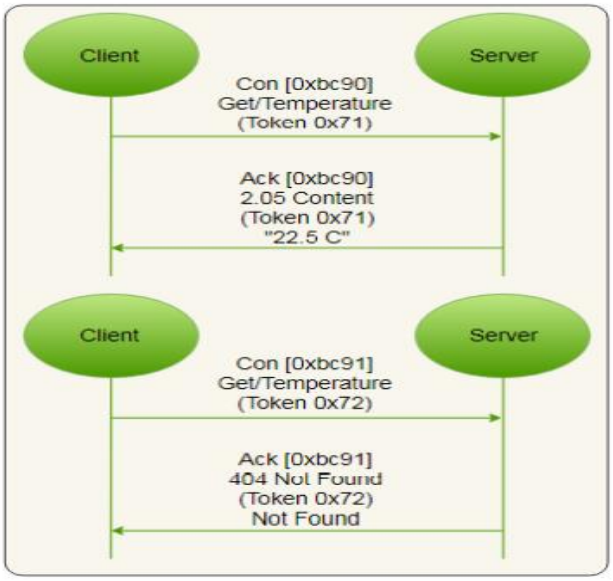


Figure 19 Successful and Failure Piggyback Messages

2.1.4.1.3.2 Separate

When a remote device sends a message to the server, the response message is empty rather than acknowledging. It is in a separate response mode. It helps the client to stop resending the message. Once the server is ready for new messages, it sends a confirmable message to the client. The client acknowledges the confirmable message with an ACK. Figure 20[34] represents a Message with a separate response

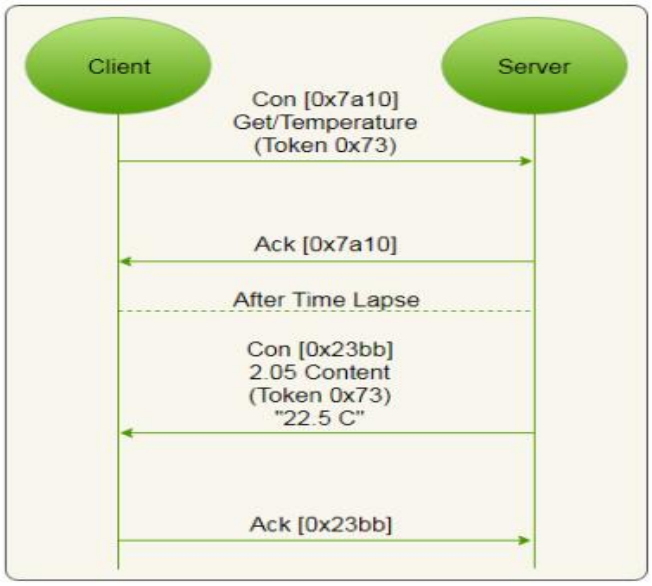


Figure 20 Message with a separate response

2.1.4.1.3.3 Non-Confirmable

When a client sends a non-confirmable message, the response could be a non-confirmable or confirmable message if the server sends a confirmable message. it is called Non-Confirmable mode. A Non-confirmable type request/response message is shown in Figure 21[34].

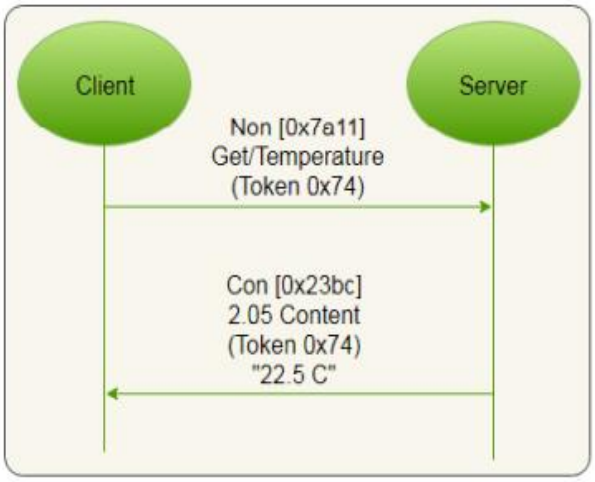


Figure 21 Non-confirmable request/response message

2.1.5 MQTT

MQTT is a lightweight publish/subscribe messaging protocol designed and developed by Andy Stanford-Clark (IBM) and Arlen Nipper in 1999 for M2M (machine to machine)[35]. MQTT stands for Message Queuing Telemetry Transport. It was specifically designed for connecting Oil Pipeline telemetry systems over satellites in very low bandwidth[36]. The goal was to develop a lightweight messaging protocol for remote devices to save energy that could operate over the years.

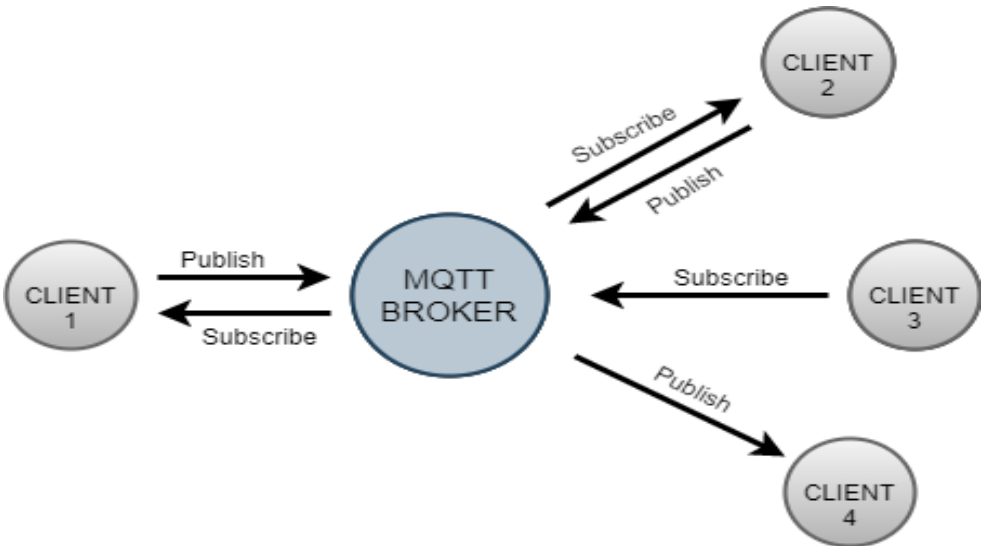


Figure 22 MQTT Architecture

MQTT has three essential constituent components.

1. Publisher (An MQTT Client)

MQTT works on publish/subscribe methodology. There could be one or more clients publishing on a particular topic in an MQTT network. Moreover, a client is entitled to either publish or subscribe to any topic of interest. A client can be any remote device, from an entire server to a bare microcontroller connected to an MQTT broker.

2. A broker (An MQTT Server)

A broker is a medium that bridges the gap between any two remote devices or a server. It acts as a hub where publishers and subscribers are linked together. A broker temporarily stores the messages from a publisher and routes/forwards them to the respective subscriber/subscribers.

3. Customer/subscriber.

A subscriber is any remote device, a simple microcontroller to a whole server connected to an MQTT broker. A subscriber need not subscribe to a valid topic. It could be subscribed to an invalid topic and yet be a valid subscriber.

Table 1 Message Format

Fixed Header	Variable header	Payload
2 Bytes size	Variable size	Variable size

MQTT message exchange

1. Establish connection

On successful network connectivity, the client sends a connect request to the MQTT server. The server acknowledges the client's connection request with a CONNACK packet with a return code denoting the connection status. Error codes are listed in [37].

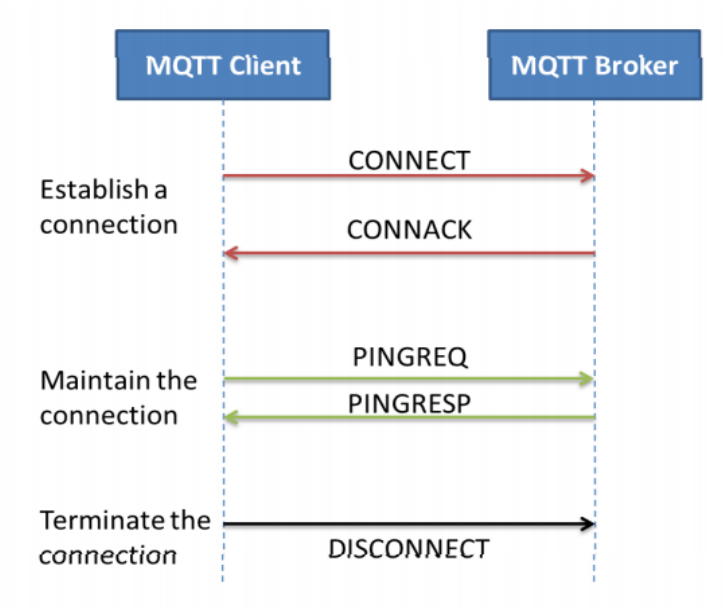


Figure 23 Establishing, maintaining, and terminating MQTT connection

2. Subscribe

   All messages are published on a specific topic. Therefore, an MQTT client sends a SUBSCRIBE packet along with the topic name. The server acknowledges the subscription request with SUBACK packet along with return code

3. Publish

   If a client desires to be a publisher, it sends a PUBLISH command to the server. The packet contains details, including QoS[38], topic name, payload. Figure 24[38] represents the package exchange in three different QoS levels. QoS 3 is the most reliable, however, with latency cost.

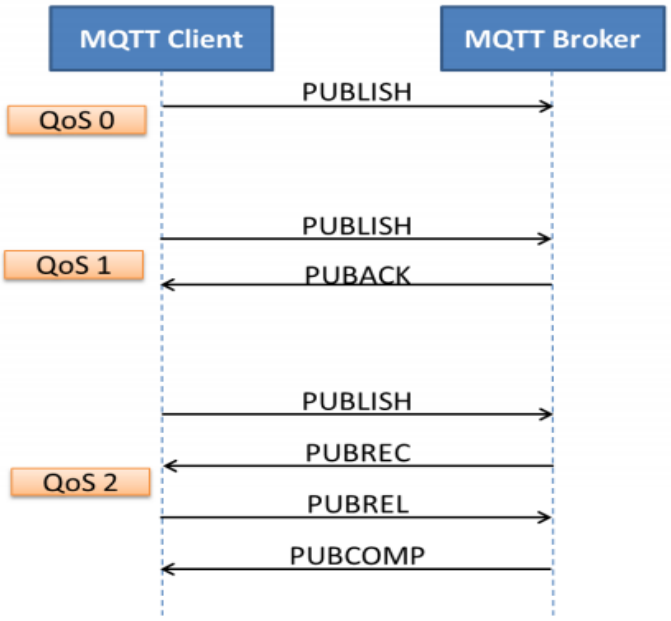


Figure 24 Client publishing messages to the server with various QoS

4. Disconnect connection

To disconnect the connection, the MQTT client sends a DISCONNECT packet to the server. The server does not acknowledge the DISCONNECT packet. All the information and topics related to the client are flushed off and disconnected from the server.

2.2 Comparison of Protocols:

2.2.1 Application Layer Protocols

Table 2 Comparison between IoT Application Layer Protocols[16]

#	Parameter	CoAP	MQTT
1	Design Goal	To Keep message overhead small and thus limiting the need for fragmentation	To reduce network bandwidth
2	Types of application supported	Machine-to-machine applications	Machine-to-machine and mobile applications
3	Mode of message exchange	Both synchronous and asynchronous	Asynchronous
4	Responsiveness /Real-time	Real-time protocol	Real-time protocol
5	Reliability	Built-in reliable mechanism	Reliability of message delivery through different QoS levels
6	Security	DTLS	SSL/TLS
7	Quality of Service (QoS)	Provides QoS	Provides QoS
8	Architecture	Request/ Response	Publish/Subscribe
9	Transport	UDP	TCP
10	Power consumption	Decrease power consumption	Decrease power consumption
11	Applicability	Web Services	Small size, small power mobile applications
12	Network bandwidth	Increase Network bandwidth	Decrease network bandwidth

2.2.2 Data Link Layer Protocols

Table 3 Comparison of IoT Data Link Layer Protocols[16]

#	Parameter	Bluetooth	ZigBee	LoRa
1	Standardized by	IEEE 802.15.1	IEEE 802.15.4	IEEE 802.15.4
2	Networks supported	WPAN	LR-PAN	LPWA
3	Frequency	2.4 GHz ISM	2.4GHz ISM	2.4 GHz frequency (915MHz Americas or 868 MHz Europe).
4	Designed for	Low-powered devices with less data usage	Small data packets over < 10-75metres	Small data, long-distance VLOS
5	Data rate	1 Mb/s	250 Kb/s	27 Kb/s
6	Throughput	270 kbps	250 kbps	5.5 Kbps
7	Topology	Mesh and star	Mesh only	star-of-stars

2.3 Generic Issues

1. Short Communication range
2. Communication Only within VLOS
3. Large bandwidth requirement.
4. Higher power consumption
5. No data filter specification
6. Critical data security concern
7. Pairing and system compatibility

Chapter 3

3 Methodology

Due to the simplicity of design, modular system, robustness flight (VTOL), and agility are characteristics that make quadcopter very popular nowadays. Quadcopter maneuvering is both simple and complex at the same time. Simple in the sense that the four rotors control maneuvering. While difficult, it is complicated to determine the rpm to engines to facilitate the forward, backward, up, down, and banking maneuvering.

3.1 Quadcopter Modeling

Quadcopters have gained much popularity because of their various features and applications[39]. However, the first challenge in flying a Quadcopter is obtaining stability and reliable maneuvering. Therefore stability plays a vital role in the success of quadcopters. It is essential to know the different forces acting on a quadcopter. Various forces act on any flying object or a free body in space. The most common force acting on any object is the gravitational force, while the object itself generates other forces to achieve various maneuverings. Some of the fundamental forces acting on a UAV, particularly on a quadcopter, are discussed:

3.1.1 Flight dynamics

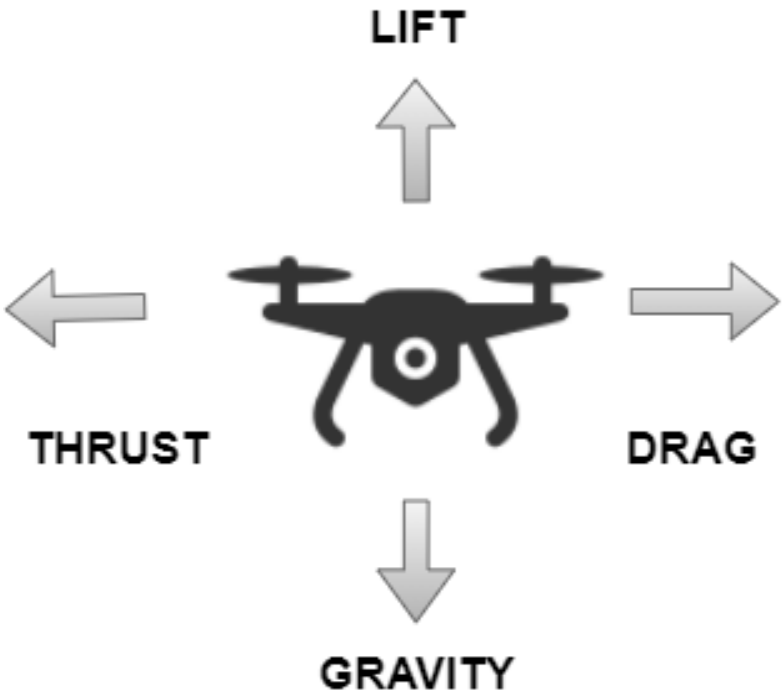


Figure 25 Forces acting on a Quadcopter

Figure 25 represents all the forces acting on a quadcopter under ideal conditions. It can be called the free body diagram representation of a quadcopter. However, free body conditions can only generate rotation (Left and

Right) and lift(Up and Down). The quadcopter needs to tilt at a certain angle to achieve forward, backward, and banking movement. Therefore there can be two configurations to study all the forces in a quadcopter.

Assumptions made:

1. Any external force not considered
2. Mass is equally distributed
3. The center of gravity lies at the middle of the quadcopter
4. The quadcopter is ideally a free body

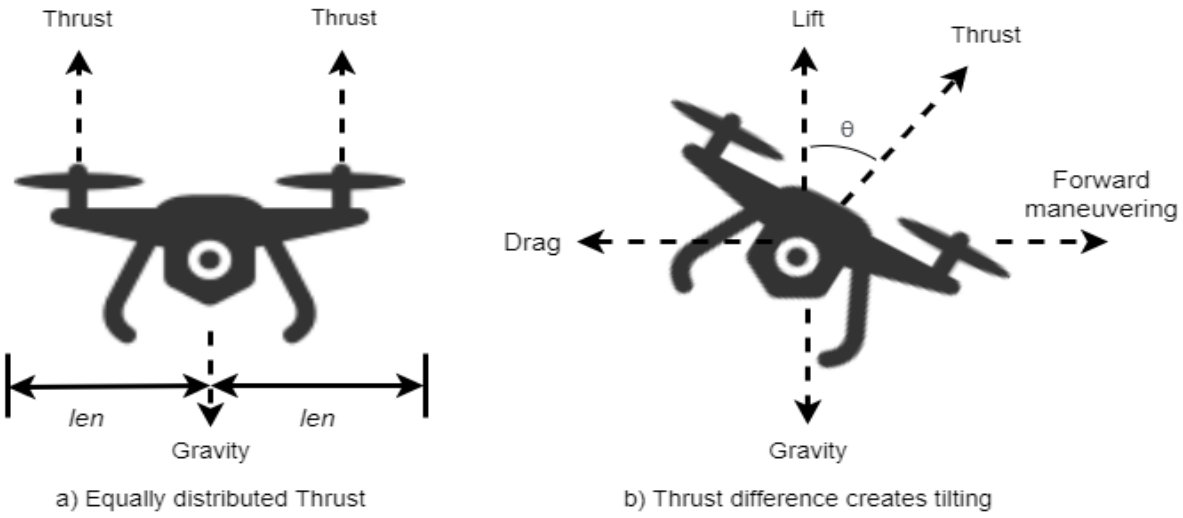


Figure 26 Forces on the quadcopter

3.1.1.1 Equal thrust distribution

Equal thrust distribution creates hovering of quadcopter at a constant point in the air. Hence, hovering means that all four rotors rotate at the same constant speed[40]. Figure 26.a represents the free body diagram of the quadcopter in equally distributed thrust conditions and hovering state. The forces on the X-axis and Z-axis are active while the Y-axis forces cancel out each other.

Observations:

1. Lift is equal to the gravitational pull
2. Drag and thrust balance each other
3. All rotors have the same and constant RPM

3.1.1.2 Unequal thrust distribution

In differential thrust distribution, the motor rotates at different RPMs. The difference in RPM is primarily responsible for generating unequal thrust at other ends of the quadcopter. In this state, all the forces in the 3D axis

are active. Tilting occurs as the rear motors increased the RPM, while the RPM's front motors are reduced. The differences of thrust at the rear and front motors create an angle (θ). Simultaneously, it produces forward motion while the tip (θ) kept increasing due to the moment.[40]

Observations:

1. Higher RPM of one pair of motors creates thrust
2. Thrust is directly proportional to the difference in motor pair RPM
3. The higher the angle(θ), the greater is thrust.

Six basic maneuverings must be mastered for a clean flight: up, down, left turn, right turn, forward, and backward movement. However, complex actions like banking left, pure banking, downward and upward movement when in motion. These movements are achieved by combining three techniques; Pitch, Roll, and Yaw.

Pitch: Pitch is the angular rotation along the Lateral Axis. The pitch angle is measured along the horizontal plane. Pitch facilitates achieving the banking trajectory.

Roll: Roll is the angular rotation along the Longitudinal Axis.

Yaw: Roll is the angular rotation along the Vertical Axis.

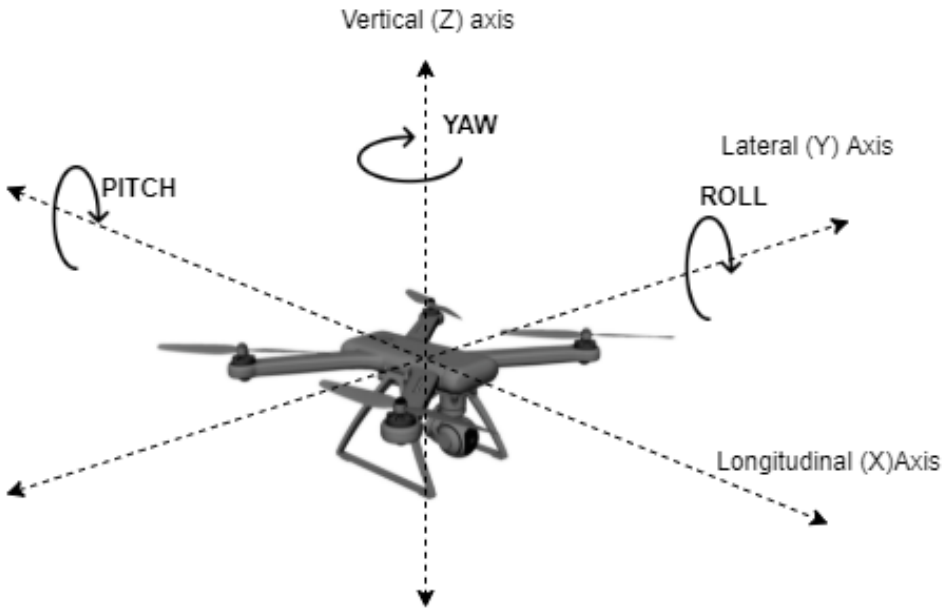


Figure 27 Pitch, Roll, and Yaw angle of a quadcopter[41]

$$R = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\Phi - S_\psi C_\Phi & C_\psi S_\theta C_\Phi - S_\psi C_\Phi \\ S_\psi C_\theta & S_\psi S_\theta S_\Phi - C_\psi C_\Phi & S_\Phi S_\theta C_\Phi - C_\Phi S_\Phi \\ -S_\theta & C_\theta S_\Phi & C_\theta C_\Phi \end{bmatrix} \ldots\ldots \text{I}$$

; C_x is cos(x) and S_x is sin(x) respectively.

The thrust which the quadcopter propellers push the air downwards for gaining lift:

$$T = k.\omega^2 \quad \ldots\ldots \text{I}$$

; T is the thrust generated by a single propeller

k is the thrust constant

ω is the angular velocity of the motor's shaft.

Summation of all propellers' thrust gives overall thrust of the entire body (Drone)

$$T_B = \sum_{i=0}^{4} Ti = k \begin{bmatrix} 0 \\ 0 \\ \sum_{k=0}^{n} \omega_i \end{bmatrix} \ldots\ldots \text{II}$$

Linear friction also acts on each axis of the quadcopter, being proportional with the corresponding linear velocity:

$$F_D = \begin{bmatrix} -k_d . \dot{x} \\ -k_d . \dot{y} \\ -k_d . \dot{z} \end{bmatrix} \ldots\ldots \text{III}$$

The inertia of the body frame assuming equal arms in the X, Y, and Z-axis, is represented by the Inertia matrix.

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \ldots\ldots \text{IV}$$

Ixx= Intertia along X-axis

Iyy= Intertia along Y-axis

Izz= Intertia along Z-axis

The relationship between angular velocity and the derivatives of Roll (ϕ), Pitch (θ), Yaw (ψ) angles of the drone as a whole is:

$$\acute{\omega} = \begin{bmatrix} \acute{\omega}_x \\ \acute{\omega}_y \\ \acute{\omega}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & S_{\mathrm{q}} \\ 0 & C_{\mathrm{f}} & C_{\mathrm{q}}S_{\mathrm{f}} \\ 0 & -S_{\mathrm{f}} & C_{\mathrm{q}}S_{\mathrm{f}} \end{bmatrix} . \begin{bmatrix} \mathrm{f} \\ \mathrm{q} \\ \mathrm{y} \end{bmatrix} \quad \ldots\ldots \text{ V}$$

; C_x is cos(x) and S_x is sin(x) respectively.

$\acute{\omega}_x$, $\acute{\omega}_y$ and $\acute{\omega}_y$ Are angular velocity in respective angles.

$\acute{\omega}$ body frame overall angular velocity.

3.2 Positioning

A reliable Positioning system plays a crucial role in the success and safe operation of the system in a densely populated environment. Therefore, the real-time tracking of a drone is highly dependent on live coordinates, which are very precise.

3.2.1 GNSS

GNSS, defined as the "worldwide position and time determination system that includes one or more satellite constellations" [42, 43]. GNSS can be considered a broad terminology that defines all the satellites in space. Today, GNSS comprises two major constellations: (1) the United States Global Positioning System (GPS) and (2) the Russian Federation's Globalnaya Navigatsionnaya Sputnikovaya Sistema (GLONASS). There are more constellations, namely Compass, Quasi-Zenith Satellite System (QZSS), Indian Regional Navigation Satellite System (IRNSS), and Galileo[44]. Galileo is a European satellite navigation system. Galileo is designed explicitly for civil and commercial purposes to be interoperable with the other radio-navigation systems. This benefits all users as they could use more satellites for redundancy and higher accuracy. The fully established constellation consists of 30 satellites (24 operational and six active spares) distributed over three orbital planes[45].

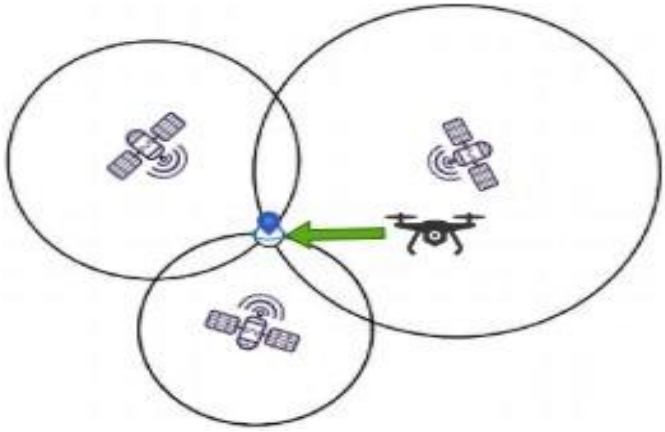


Figure 28 GNSS Positioning Principle

3.2.2 GNSS Antenna

A wide variety of GPS/GNSS antennas are available in the market, suitable for different navigational applications. They vary from simple miniaturized antennas used in headsets and smartphones to complex adaptive and beamforming arrays used to overcome interference and jamming.

3.2.2.1 Classification of antenna

Based on design and performance, GNSS Antennas are Categorized

1. FRPA
2. High-gain directional antenna
3. GPS adaptive antenna
4. Multiband antenna
5. Handset antenna
6. Active antenna

Most GNSS modules have internal antennas capable of receiving signals in Visual Line of sight (VLOS). However, when beyond the visual line of sight (BVLOS), the frequencies are impaired. Hence the on-chip antennas are unable to pick the signals. Therefore, it is imperative to select an application-specific antenna based on the Selection Criteria(GNSS)[46, 47]. Based on the selection criteria, we choose the FRPA antenna. According to Alison Brown and Huan-Wan Tseng, NAVSYS Corporation [48], many DoD aircraft currently use FRPA antennas. Moreover, some host aircraft within Airforce and Navy has elected to install FRPA antennas to provide the anti-jamming protection needed in many tactical environments."UTM Programs such as the Joint Direct Attack Munition (JDAM), Joint Air-to-Surface Standoff Missile (JASSM), and the Joint Standoff Weapon (JSOW) can benefit from the reduced size but the full performance of the mini-array technology"[49].

3.2.2.2 FRPA

An FRPA has a nearly omnidirectional beam pattern in the upper hemisphere(5^o-10^o). It is designed for acquiring almost all, but at least a minimum of four in the VLOS above the masking angle. FRPA is specially designed for an airborne vehicle. It is small in size, weight, and accuracy, making it ideal for UAVS applications.

Table 4 GNSS Antennas Classified

Platform	Application	Bands	Inherent bandwidth	Gain Pattern	Multiple rejection	Interference rejection	Phase center stability	size	Weight	Cost
Large	Geodetic, ships etc	2 or more	>40MHz	Very strict*	High	High	Good	Dinmeter > 15 cm	Heavy*	High*
Medium	Car, truck, train, aircraft	1-2	>40MHz	Somewhat strict*	Medium	Medium	Fair	Dinmeter > 3 cm	Medium	Medium

Small	Bodtwear able, laptops	1	>40MHz	Not strict	Low	Low	Poor	Small and conforma l	Light	Low
Handhel d	Cellphone , GNSS receiver	1	>2MHz	Ignored	None	None	Very poor	Very small	Very light	Very low

Table 3 represents GNSS Antennas Classified Based on Intended Platform, With their general Characteristics and Anticipated Changes

3.2.3 GNSS Constellation: Galileo

In 2002, the European Union commenced developing Galileo, GNSS constellation. Galileo is designed explicitly for civil and commercial purposes and can interoperable with the other radio-navigation systems. This benefits all users as more satellites are used for redundancy and higher accuracy[43].

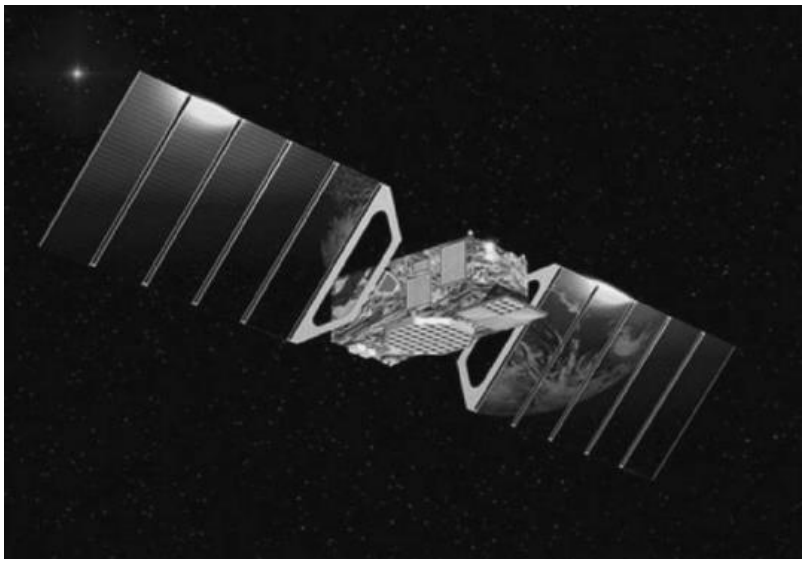


Figure 29 Artist's view of a Galileo satellite. European Space Agency, J. Huart (reprinted)

Features of Galileo constillation[43] :

A. Services

1. Four navigational services with an added service to support search and rescue operations have been recognized to facilitate the broadest range of user requirements, including professional users, scientists, market users, life safety, and the regulated public domains.
2. The Galileo navigational services can benefit on a local basis through collaboration with local elements for applications that demand a higher rate of precision within environmental conditions

B. Infrastructure:

1. The Galileo system is entirely on a global constellation of 27 satellites in three MEO orbital planes at an angle 56º to the equator at 23 000 km altitude in a Walker 27/3/1 configuration[28].

C. Signals

1. The Galileo navigational system can transmit signals in the four frequency bands referred to as E5a, E5b, E6, and E1, indicated in Figure 30. Galileo uses CDMA within each frequency band. A user may find detailed descriptions of the Galileo signals in [50, 51]

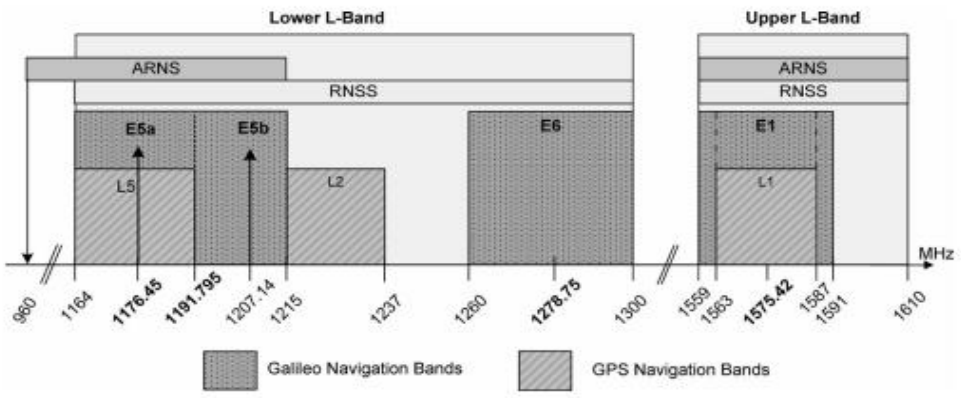


Figure 30 Galileo frequency bands

Table 1: Performance Requirements for the GALILEO Open and Safety-of-Life Services

Galileo	Open Service	Safety-of-Life Service
Coverage	Global	Global
Accuracy(95%)	Horizontal: 15m Vertical:35 m (single frequency) Horizontal: 4m Vertical:8m (dual frequency)	Horizontal: 4m Vertical:8m (dual frequency)
Availability	99.8%	99.5%
Alert Limit	N/A	Horizontal alert limit: 40m Vertical alert limit: 20m

Time to Alert	N/A	6s
Integrity Risk	N/A	2x10-7/150 s
Continuity Risk	N/A	8x10-7/15 s
Certificate and Service Guarantees	No	Yes

3.2.4 Geographical classification

The density of drones varies entirely on different factors like population density, number of deliveries, government regulations, etc. Therefore, other areas are expected to have different traffic densities. The areas with lower traffic density do not require swift process monitoring. However, in some areas, the thickness could be so high that managing the density could be tedious. Hence we categorize a geographical location into different blocks. Blocks with high traffic density are further divided into subblocks filtering the density. Figure 31 shows various blocks and subblocks in Norway. The main reasons for dividing the space into blocks are :

Figure 31 Geographical distribution

1. Limiting the publish/subscribe topic within a region
2. Efficient handling of traffic
3. Reduces routing overhead to the server

4. Efficient data management
5. Effective zone separation

3.3 Speed Tracking

The tracking system is entirely dependent on various factors, including weather. However, accuracy is one of the most critical factors affecting the performance and reliability of the system as a whole.

The raw data from a GNSS module provides Latitude, Longitude, Altitude concerning sea level, time of data fetch, and much more. For the tracking purpose in 3D space, Latitude, Longitude, and Altitude are the slightest requirements. However, the system can use other data to improve system reliability and improve performance.

There are multiple ways to track the speed of a drone. The most simple is to use a speedometer onboard in the drone. However, this increases both the cost and the weight of the system. Therefore, it is better to calculate the speed based on the available design on board. Hence, one of the most feasible ways to monitor speed is by utilizing the coordinates from the GNSS module itself. However, using the coordinate system increases the complexity of the system. It needs to handle different use cases

3.3.1 Same height

When both the initial point A and final point B are at the same height, only the two parameters, i.e., Latitude and Longitude, change while the attitude remains the same. Consideration of only two parameters considerably reduces the complexity of the distance. It also highly reduces the travel time as the displacement is zero in a straight path. However, it is almost impossible to have the ideal situation of origin and destination at the same point. Straight paths usually exist in fragments between the initial and final positions.

$$Distance\ (A,B) = \sqrt{(x2-x1)^2 + (y2-y1)^2}$$

Where: x = Latitude

y = Longitude

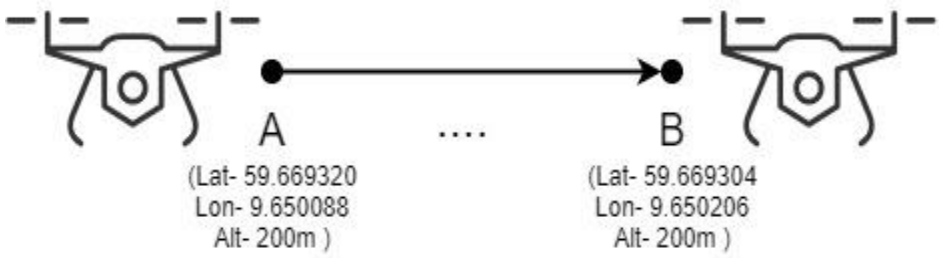


Figure 32 Drone path at the same height

3.3.2 Different height

Almost no initial and final positions of a specific path are in the same height in practice. Therefore differential height describes a more realistic path scenario.

3.3.2.1 Euclidian Distance

Euclidian distance is the shortest distance between any two vertices in 3D space. Figure 33 represents a Euclidian distance which, can be considered as a diagonal and has zero displacements. Therefore Euclidian distance is most feasible in terms of the shortest distance within a minimum time interval. However, the complexity of the Euclidian path is much higher than compared to Manhatten. Unlike in Manhatten, where the drone either attains/nullifies the height and horizontal distance. Euclidian path advanced in both horizontal and vertical altitude, drastically increasing the complexity and possibility of collision.

$$\text{Distance } (A,C) = \sqrt{(x2-x1)^2 + (y2-y1)^2 + (z2-z1)^2} \quad \ldots\ldots \text{ VI}$$

Where: x = Latitude

y = Longitude

z = Altitude

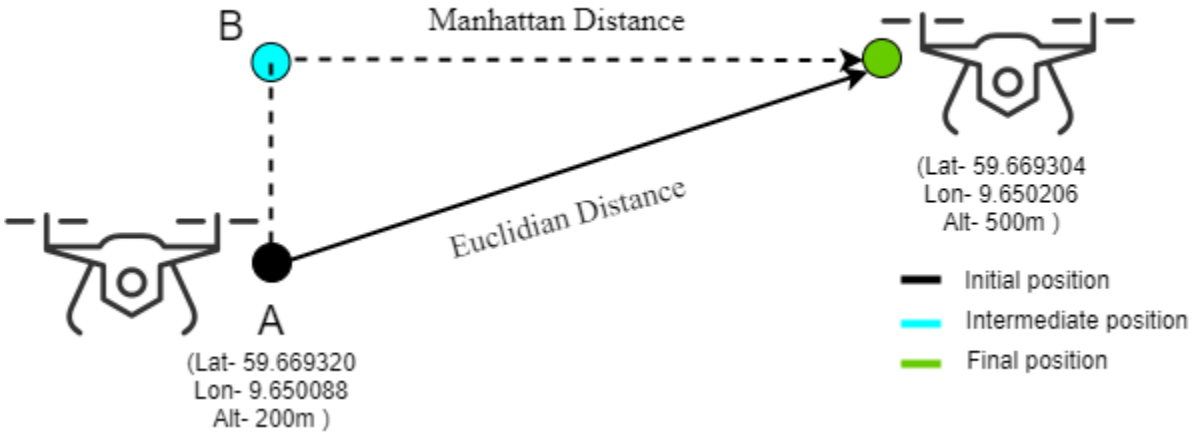


Figure 33 Euclidian Path

3.3.2.2 Manhattan Distance

Distance between any two vertices in 3D space can be traced using the Manhattan path. Since the points in space are in different heights and the path followed is AB and BC to reach its final destination as per figure 34. The path followed can be subdivided into two; AB and BC, respectively. Manhattan path is simplistic in nature with better traceability, however with a cost of longer flight time.

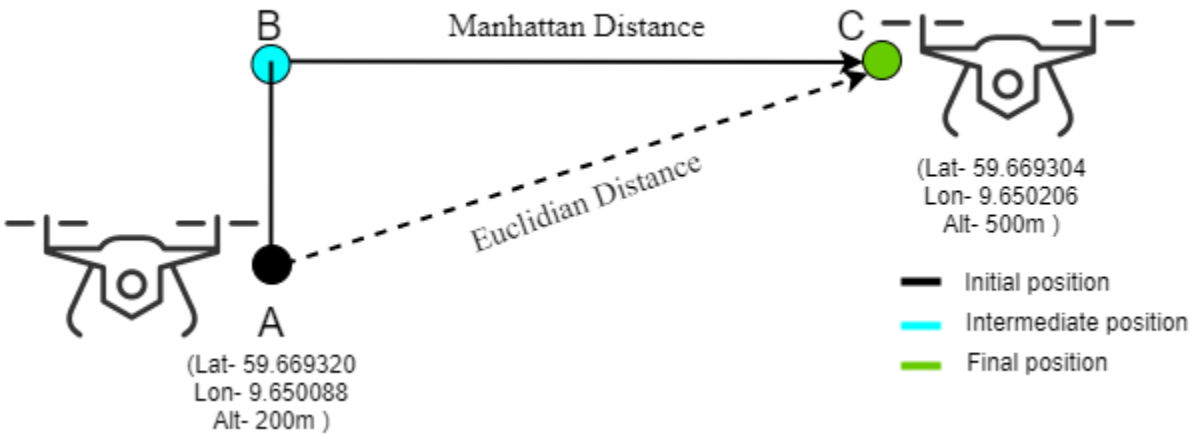


Figure 34 Manhattan Distance

3.4 Layer Separation

Research focusing on increasing Air Traffic Control [52, 53] and the relation between traffic density and capacity[54, 55] are underway for few decades. With the separation of the airspace layer, each layer can be considered a highway in the air, analogous to terrestrial highways[56].

When a drone needs to go from one point to a particular location, the drone makes an entry request to the central server. When the request is approved, the drone proceeds to the nearest entry point. An entry point is an imaginary portal where a drone can enter the highways. And when the drone reaches its zone, it requests the server for an exit. Based on the drone's current location, the central server guides the drones to the nearest exit point and then to the final destination on the ground.

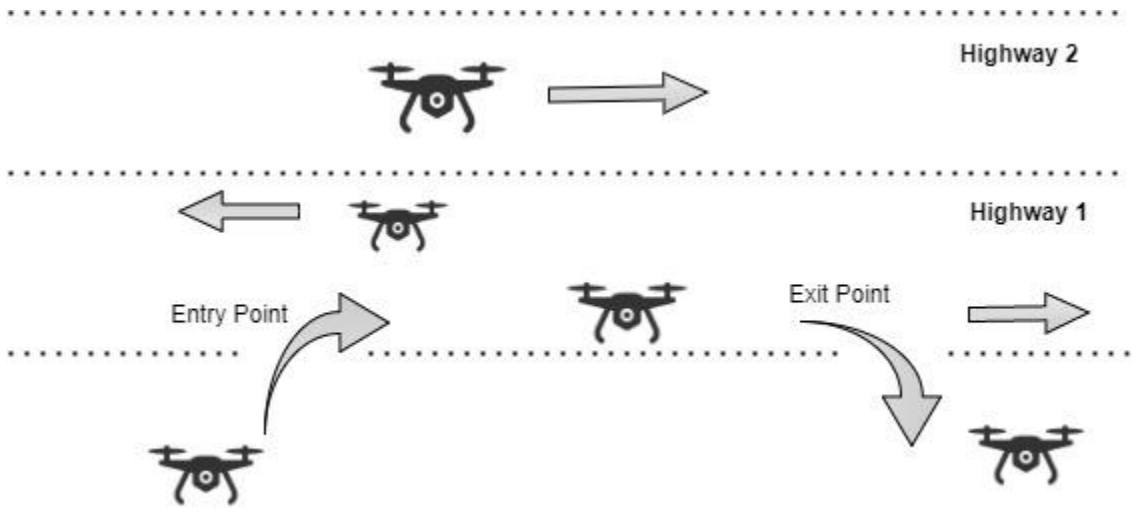


Figure 35 Model of Drone Highway

Features

2. Drone highways are flexible.
3. They are imaginary and require no cost compared to terrestrial highways.
4. Assignment of highway based on distance and time of flight.

3.5 Cellular Connectivity

Quectel, the leading cellular IoT modules supplier, provides a wide range of cellular modules (GPRS to 5G) with GNSS enabled tracking system.

Figure 36 Quectel Modeule 2G/5G modules[57, 58]

3.5.1 Specification of MC60 [59, 60]

1. Ultra-compact size: 18.7mm × 16.0mm × 2.1mm
2. Support Bluetooth 4.0 and Bluetooth 3.0 specifications
3. Easier soldering process with LCC package
4. Low power consumption: 1.2mA @DRX=5
5. Support voice, data, SMS, and QuecFOTA™ functions
6. Embedded abundant Internet service protocols
7. Built-in LNA for higher sensitivity: -149dBm @Acquisition
    1. -167dBm @Tracking
8. Multi-GNSS system: GPS, GLONASS, Galileo, and QZSS
9. GNSS receiver channels: 99 acquisition/33 tracking/210 PRN channels

10. Support advanced technologies: EASY™/LOCUS™/EPO™/AlwaysLocate™/GLP/SDK/QuecFastFix Online
11. Great anti-jamming performance due to multi-tone active

3.5.2 Specification of RM502Q-GL [61, 62]

1. 5G/4G/3G multi-mode module with M.2 form factor, optimized for IoT and eMBB applications
2. Worldwide 5G and LTE-A coverage
3. Both NSA and SA modes supported
4. Multi-constellation GNSS receiver available supporting fast and accurate fixes in any environment
5. Feature refinements: DFOTA and VoLTE (optional)

3.6 Network Connectivity

3.6.1 MQTT Network Topology

MQTT is a lightweight publish/subscribe protocol designed around a central broker. Hence MQTT follows a simple start network topology, as shown in figure 37. The broker acts as the centralized hub.

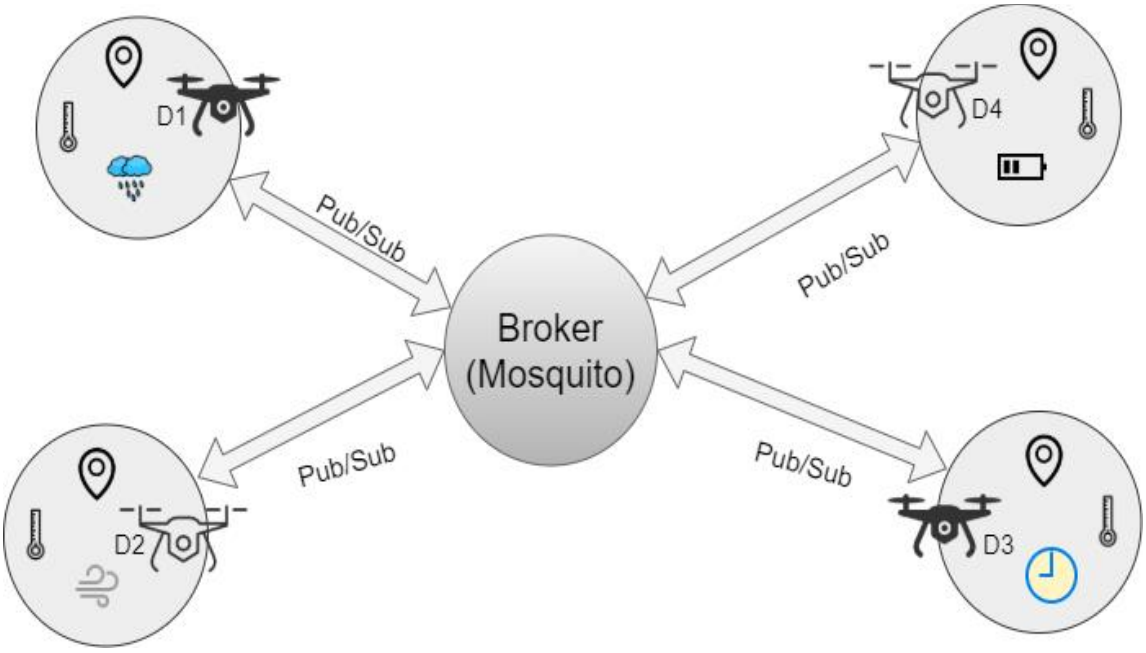


Figure 37 MQTT Star network topology

3.7 Implementation

The heart of implementing an IoT service is reliable network connectivity. Therefore, the 5G cellular network is used for reliable connectivity [63-66] within urban airspace.

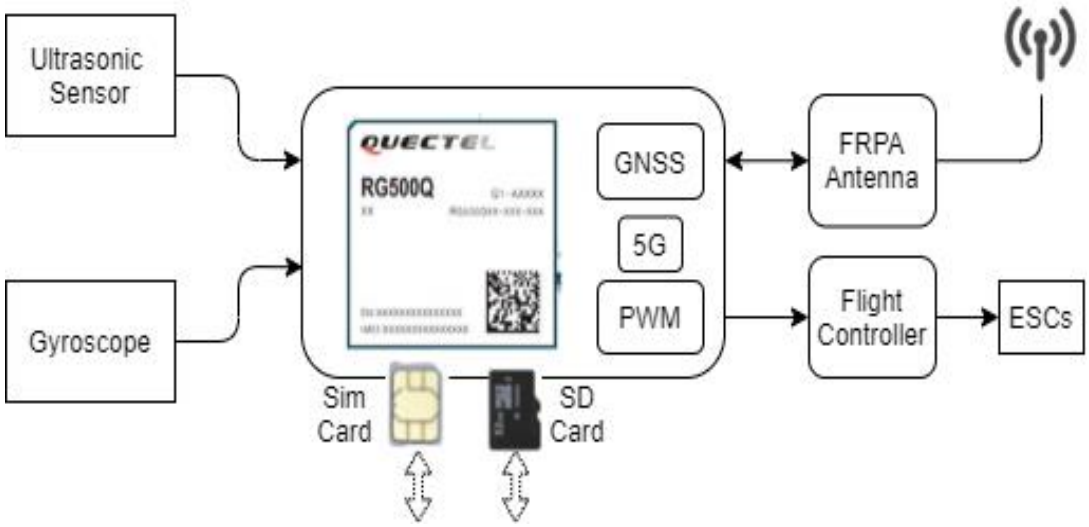


Figure 38 Interface Block Diagram

Ultrasonic sensors :

An ultrasonic sensor detects any nearby objects within the range. The scan distance depends on the specification of the sensor. Sensor-ranging detection capability from 1 cm to 20 m are commonly available types[67]. The sensor emits and detects a sonic pulse reflected by the object. HRLV-MaxSonar-EZ, an ultrasonic sensor product from MaxBotix with a range of 5000 mm, is implemented in the practical. Any object in the range lesser than 30 cm accounts for 30 cm. Since the sensor has a narrow beam angle of ~8^{o}, the sensor is placed in the servo to scan 360^{o}. The sonic pulse, time of flight between the emitter and the object, and back to the receiver is measured. Distance is calculated using the formula 1.

$$\text{Distance} = \frac{\text{speed of sound X sonic time of flight}}{2} \quad \ldots. \text{VII}$$

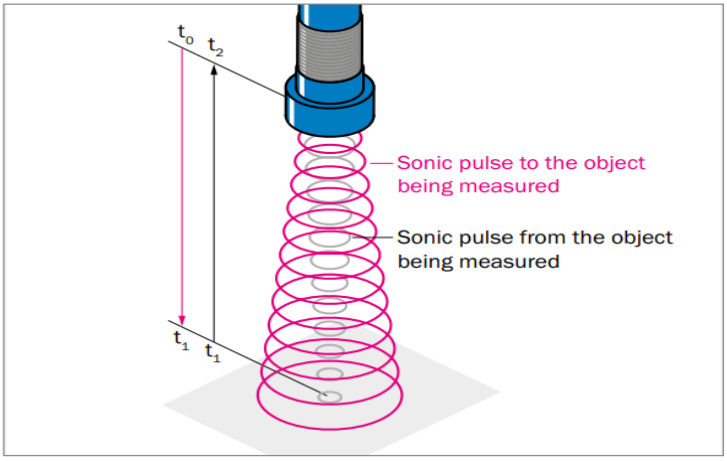


Figure 39 Time-of-flight measurement

Gyroscope:

A gyrosensor detects any angular tilt in the vertical and horizontal orientation of the drone. It generates an event-based trigger each time the device angle changes. Some advanced gyrosensors have a programmable average window and a programmable average threshold.

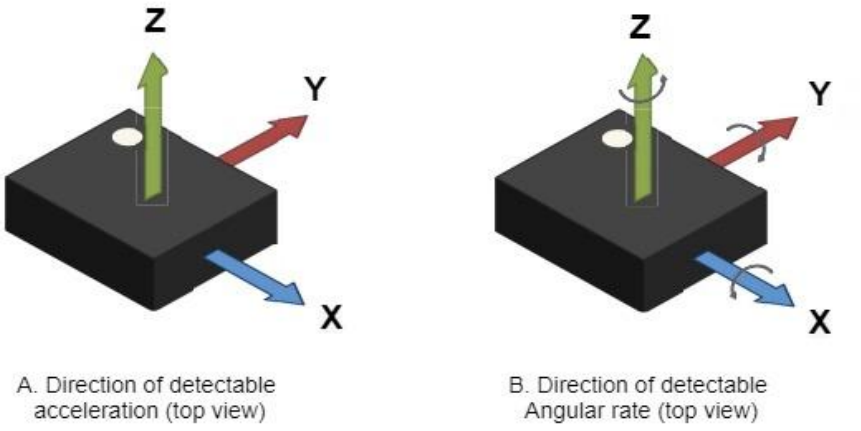


Figure 40 Lateral/Angular tilt Detection[68]

3.8 Simplified Ideal Case Operating algorithm

1. Request for take-off.
2. Case 1. If approved, follow the trajectory and head towards the assigned nearest entry point
   Case 2. If denied, resend the request command, wait for the approval.
3. Notify the server of the entry to the drone highway
4. Continuously follow the assigned trajectory and keep uploading the real-time location.
5. Check if the drone is out of trajectory, correct the location within trajectory error limits.

6. Check if the drone receives the highway switch command; if yes, change to the specified highway or continue the forward journey.
7. Check if the drone receives the hovering command; if yes, hover, post the server, and wait for a command from the server; continue the forward journey.
8. If the drone is in the destination geographical block (figure 31), request an exit from the drone highway.
9. Follow the trajectory to the destination.
10. When reached the destination, update the server and wait for a backward route post from the server.

4 Simulation and Result

4.1 Simulation setup

4.1.1 Raspberry Pi

The simulation of the drone communication is carried out using Raspberry pi 3b+. Features[69] of raspberry pi; 5V TTL logic level, WiFi, GPIO's, high processing power make it ideal for simulation usage. Each raspberry pi is treated as a removed drone connected to the server. Sensors onboard to the raspberry pi continuously read the ambient data and continuously push the reading to the thingsboard. The data can be visualized in the thingsboard dashboard. The raspberry pi acts as a bridge linking the sensor data to the thingsboard. Therefore, it requires softwares that support sensing the data from it to the server.

Paho MQTT :

Paho is an Eclipse foundation project which helps to create an mqtt client. The client can both subscribe and publish data on any particular topic. Therefore, Paho mqtt client library is used for procuring the data to and from the thingsboard server.

Linux Commands :

pip –version; check the package installer

sudo apt-get install python3-pip; if pip not preinstalled

pip install paho-mqtt; install paho mqtt

Figure 41 Raspberry Pi 3b+ [69]

4.1.2 Thingsboard

"Thingsboard[70] is an open-source IoT platform for data collection and processing, visualization, and device management". It enables connectivity via industry standard IoT protocols - MQTT, CoAP, and HTTP facilitating both cloud and local deployments. ThingsBoard leverages device scalability, data visualization, cloud data processing, ensuring data protection. Features of thingsboard can be referred to [70].

4.1.2.1 Device setup

A device in thingsboard is analogous to an actual remote device. All the incoming telemetry data are received at the device—steps in adding a device[70].

1. Within the devices, click + to add new devices
2. Enter the Device name and transport type (Protocol).
3. Enter client, server, and shared attributes in the attributes tab.

4.1.2.2 Dashboard setup

Dashboard represents all visuals of all the telemetry data pushed to the IoT server. A wide variety of widgets are available in thingsboard, making the dashboard clean and attractive. The same data can be

1. Within the dashboard, click + to create a new dashboard
2. Assign the dashboard alias and device to receive the data from
3. Within the assigned alias, click + to add a new widget
4. Add widget data source as the telemetry parameters from the device

4.1.2.3 Rule Chain Setup

A Rule Chain in thingsboard is highly configurable with a drag-and-drop feature and minimal programming. The drag-and-drop, referred to as Rule Node, is a primary building block of the rule chain that handles and processes the telemetry data one at a time and forwards it to the next node based on its decision. The input node in figure 42 is, by default, present in all the rule chains, while figure 43 represents the default root chain in the thingsboard. The primary task of the input node is to collect all the data and events relative to the current alias. Before any event, the desired Rule chain must be assigned as the Root Rule Chain.

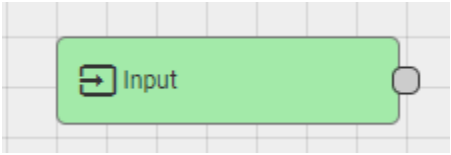


Figure 42 Input Node

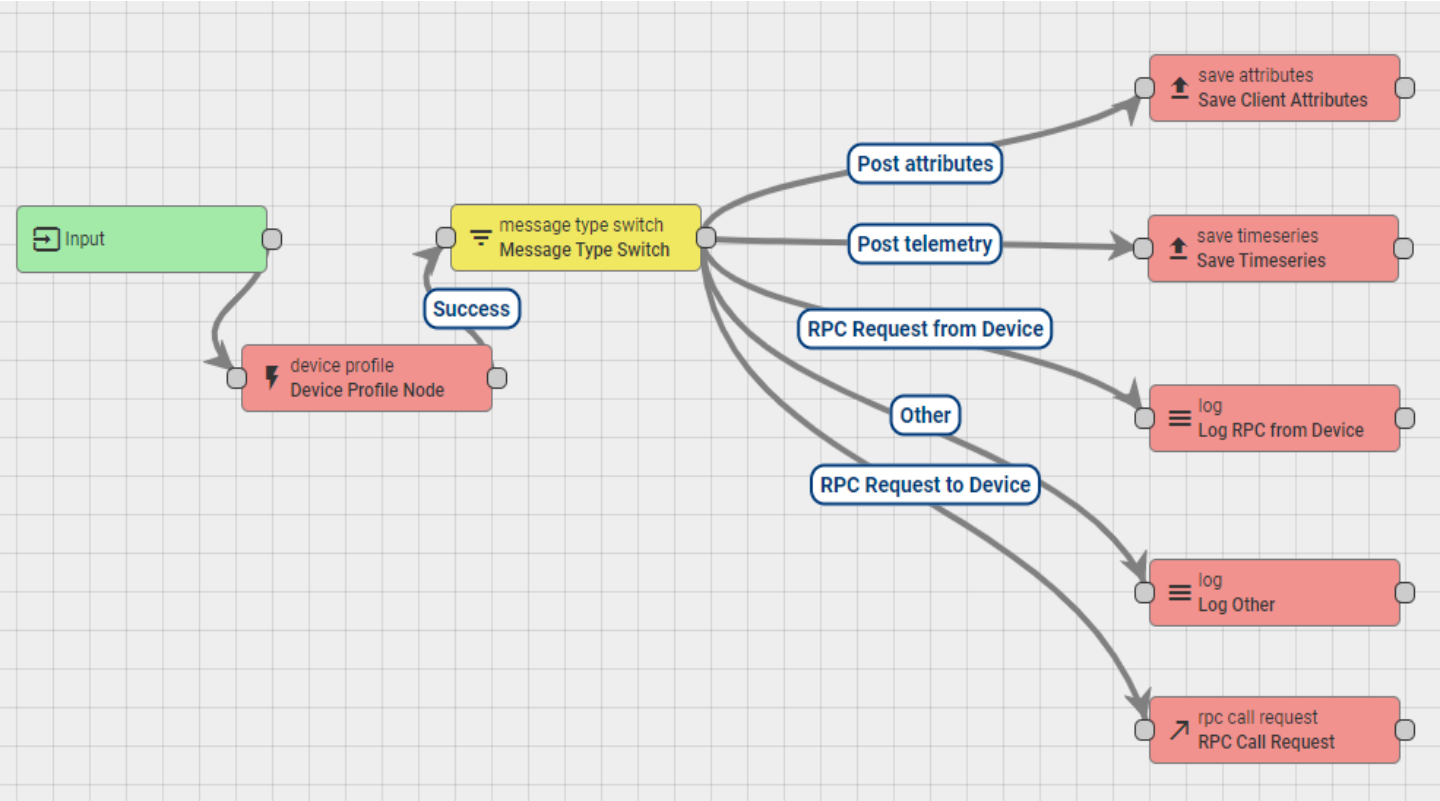


Figure 43 Root RuleChain

4.2 Simulation

4.2.1 Take off approval

Only the authenticated drones must take to the sky. Therefore, an approval system is of utmost importance concerning safety. Each drone connected to the thingsboard has a unique access token and device id. On creating a device, the access token and device id are autogenerated. Connection to thingsboard requires an access token, or the drone connection to thingsboard and the authentication fails.

```
pi@raspberrypi:~/thesis/demo.thingsbaord/integration $ python integration.py
Traceback (most recent call last):
  File "integration.py", line 19, in <module>
    client.connect(THINGSBOARD_HOST, 8080, 60)
  File "/usr/local/lib/python3.7/dist-packages/paho/mqtt/client.py", line 941, in connect
    return self.reconnect()
  File "/usr/local/lib/python3.7/dist-packages/paho/mqtt/client.py", line 1075, in reconnect
    sock = self._create_socket_connection()
  File "/usr/local/lib/python3.7/dist-packages/paho/mqtt/client.py", line 3546, in _create_socket_connection
    return socket.create_connection(addr, source_address=source, timeout=self._keepalive)
  File "/usr/lib/python3.7/socket.py", line 727, in create_connection
    raise err
  File "/usr/lib/python3.7/socket.py", line 716, in create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused
pi@raspberrypi:~/thesis/demo.thingsbaord/integration $
```

Figure 44 Failed Connection

For any reason, like port number, incorrect URL, the compiler throws an error with error code 111. The error represents the socket connection error.

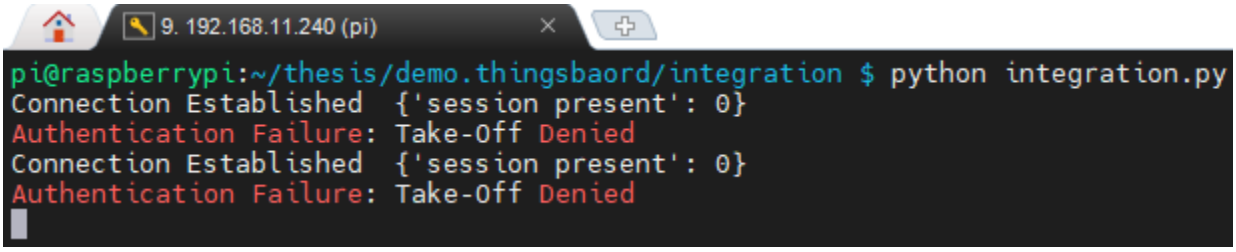


Figure 45 Connection attempt; Access denied

It is evident from console figure 45 that there is a connection of the drone with the thingsboard server. The session code 0 represents the connection with the thingsboard that has been established. However, the drone credential, i.e., the access token, is incorrect. Hence, the drone take-off is denied. The drone tries to reconnect to the server repeatedly with failed attempts until a valid access token is provided.

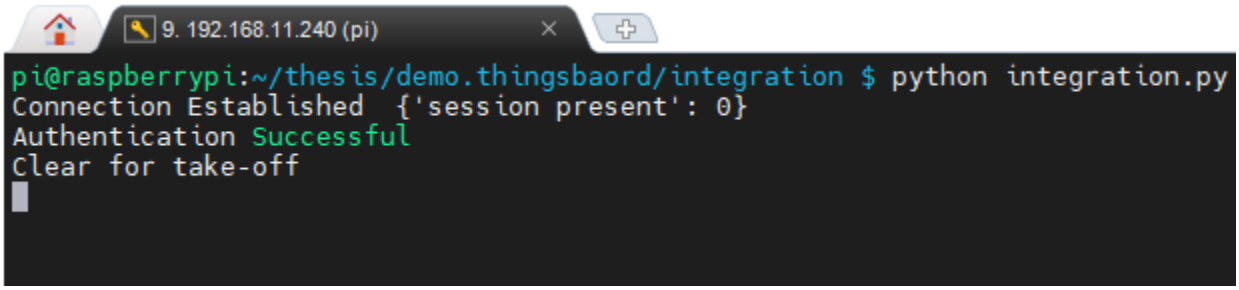


Figure 46 Connection success

When the access token is authenticated, with connection session 0, the drone is granted permission to take off. The drone then follows the path instructed by the path planning algorithm. Static GPS coordinates are fed to the system in advance. The drone follows the static coordinates. The drones, on approved take-off, attain vertical height 250m which is considered default before any forward motion.

4.2.2 Routing

The routing of a drone is divided into three different components.

4.2.2.1 Ascending

Once the drone is authenticated and approved for take-off, the drone first attains a default height of 250M, considering its initial position as the reference. Once the default height is attained, the drone makes any further moves.

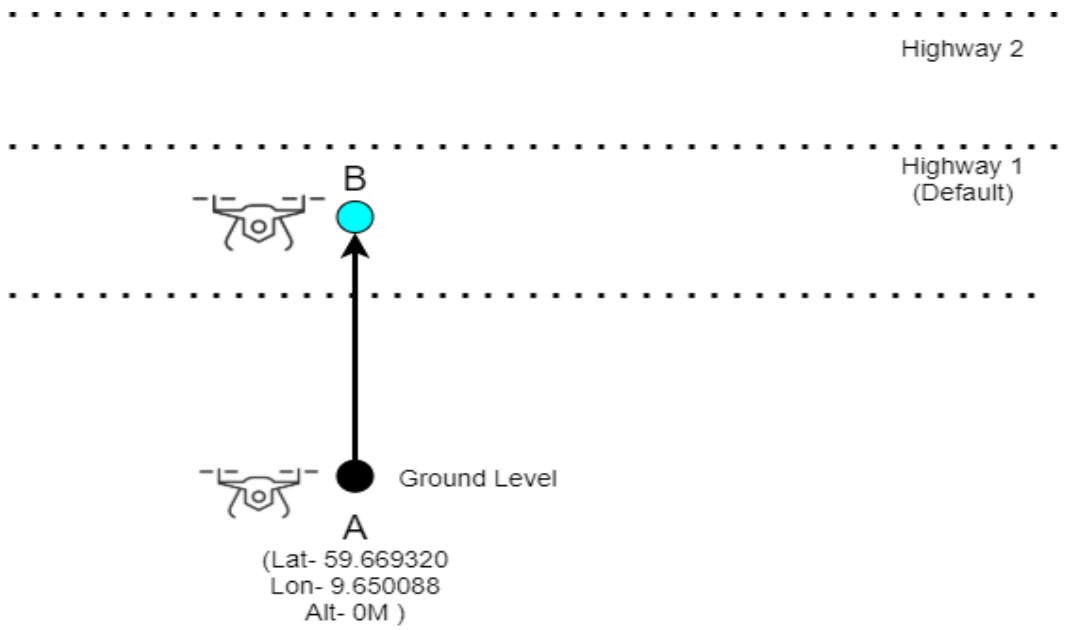


Figure 47 Drone Ascending

```
Connection Established  {'session present': 0}
Authentication Successful
Clear for take-off
Current height = 0 M
Climbing to 250M altitude....
Current height = 1 M
Current height = 2 M
Current height = 4 M
Current height = 8 M
Current height = 16 M
Current height = 32 M
Current height = 64 M
Current height = 128 M
Altitude Attained = 250
```

Figure 48 Drone Ascending Console

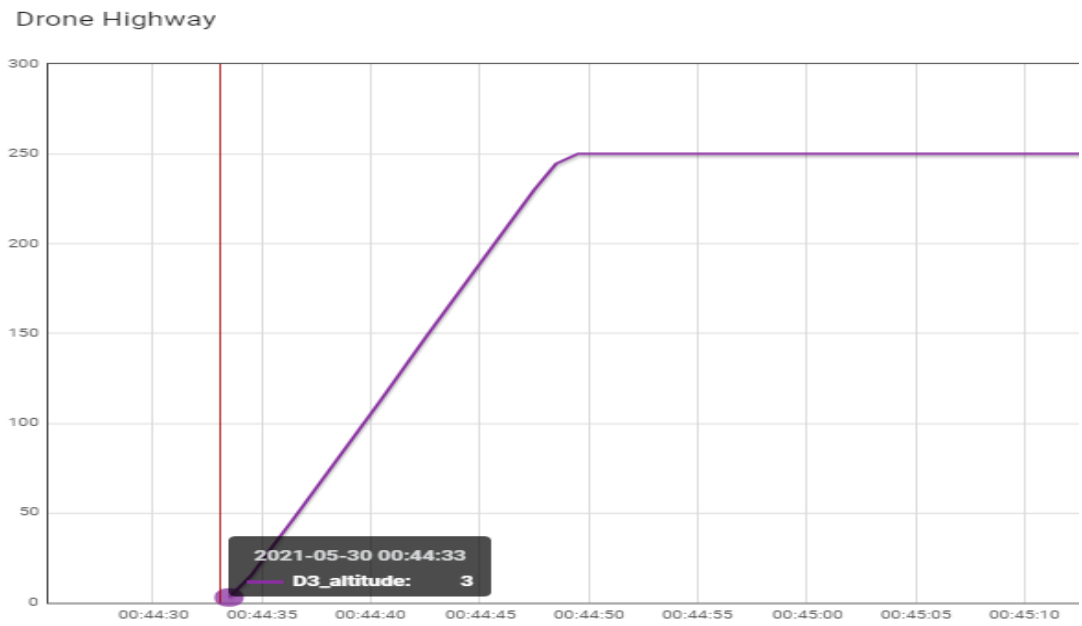


Figure 49 Drone Ascending Graph

Drone

Drone name	status	speed	battery
Drone 1	on route	89	100
Drone 2	arrived	0	100
Drone 3	climbing	40	20.0
Drone 4	decending	101.5	100

Figure 50 Drone Status

Figure 48 depicts that once the take-off is clear, the current altitude of the drone is considered 0M. It is an assumed altitude; the initial height of the drone varies based on its area/region of operation. GPS sensor is read before in industrial practice. The drone slowly gains altitude with reference to its initial position coordinates. It follows initial reference coordinates until the drone attains the default height of 250 m. From 0 altitude to 250M, the whole process is considered as ascending of a drone.

4.2.2.2 On route

When the drone attains the default height of 250M, the drone then starts its forward journey towards its destination. The route from the end of ascending to the start of descending is defined as on route. In this stage, the drone makes an onward journey to the destination, continuously following the route as instructed by the

routing algorithm .the GPS continuously reads the current position of the drone and continuously updated the server for any drift in position and correction of the same. Figure 51 represents the current drone status, drone 3 status as on route.

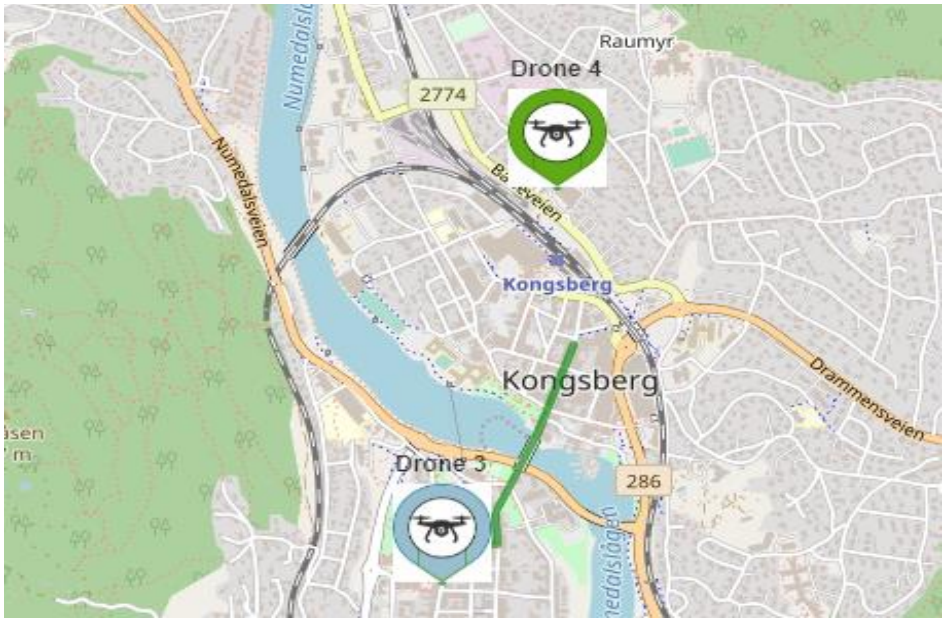


Figure 51 Drone On Route

4.2.2.3 Descending

Descending is analogous to ascending, where the drone gradually climbs down the altitude from the current highway to the ground level. When the drone reaches its final GPS coordinate, the status of the drone is updated to "descending.". It then slowly climbs down, following the Euclidian path until it reaches ground level. Figure 52, drone 3 represents the descending status of the drone. On descending, the drone's speed is maintained at 5 kmph to 0 concerning its descend height for the safety of the drone and the payload it carries.

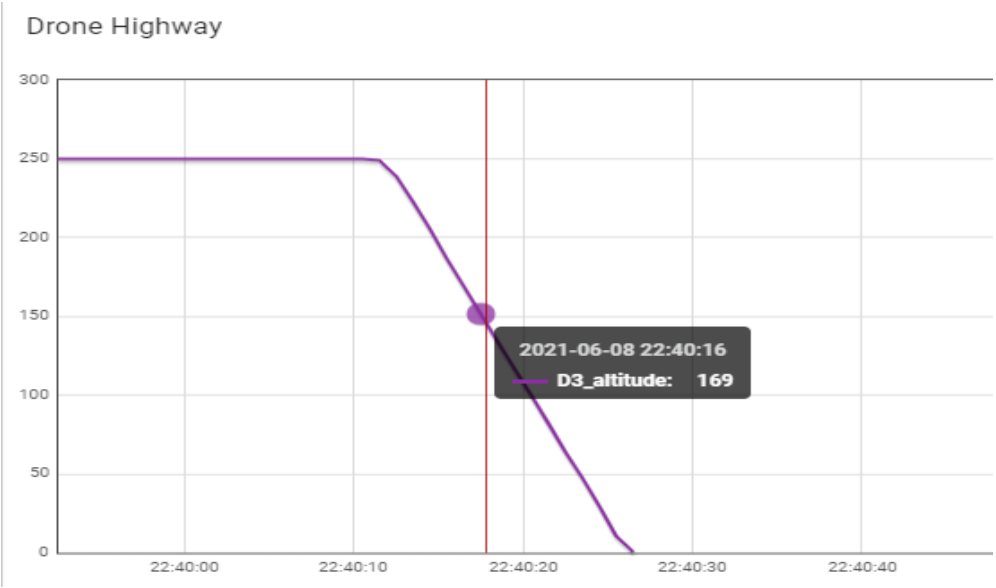


Figure 52 Drone Ascending Graph

4.2.3 Hovering

Hovering is a state where a drone maintains its current coordinates in the air, maintaining the same height. Hovering conditions, in general, can be programmed or autogenerated to prevent a collision. During the hovering state, the algorithm gets a chance to route/manage the traffic avoiding collisions or long flight time.

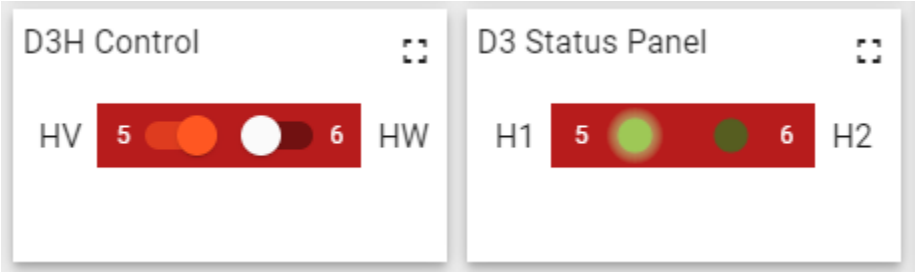


Figure 53 Hovering command and status panel

Figure 53 shows that the hovering command is sent (HV), and the same is reflected as the drone's acknowledgment. This is the server-side command for hovering. However, an automated system is also incorporated into the project. The ultrasonic sensor onboard the drone senses the surrounding distance and objects' presence. If an object is closed than 5M, the drone updates its current status to hovering. When the other drone safely passes the shared route. The status of the hovering drone is changed back to "on the route," and the drone heads its further journey.

Drone

Drone name	status	speed	battery
Drone 1	decending	123	100
Drone 2	arrived	0	100
Drone 3	hovering	92	23.0
Drone 4	decending	101.5	100

Figure 54 Drone Status

4.2.4 Object detection

HRLV-MaxSonar-EZ from MaxBotix is successfully implemented to detect any object in the drone trajectory. Two experimental cases were carried out in [71]. Figure 55[71] is the trajectory of the drone. The blue line is the actual drone trajectory, while the line represents the ideal drone trajectory, and the red block is the object in the trajectory of the drone. The sensor continuously scans for any object in its vicinity.

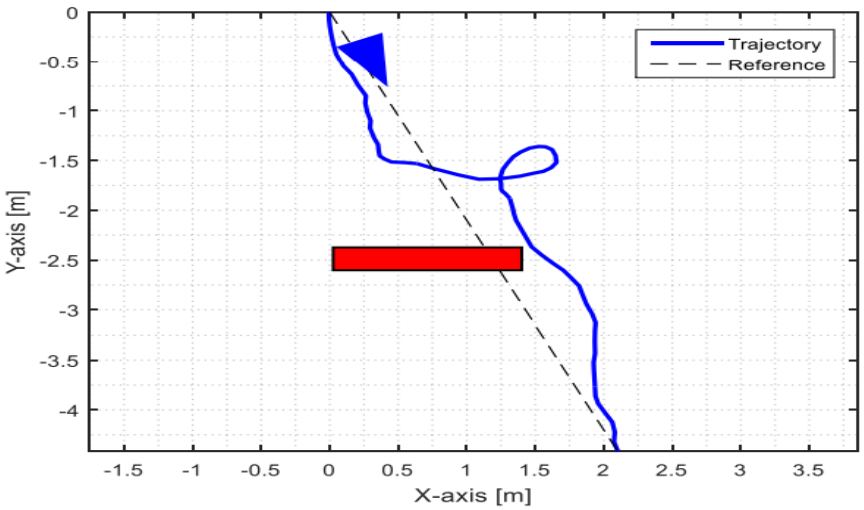


Figure 55 Drone trajectory

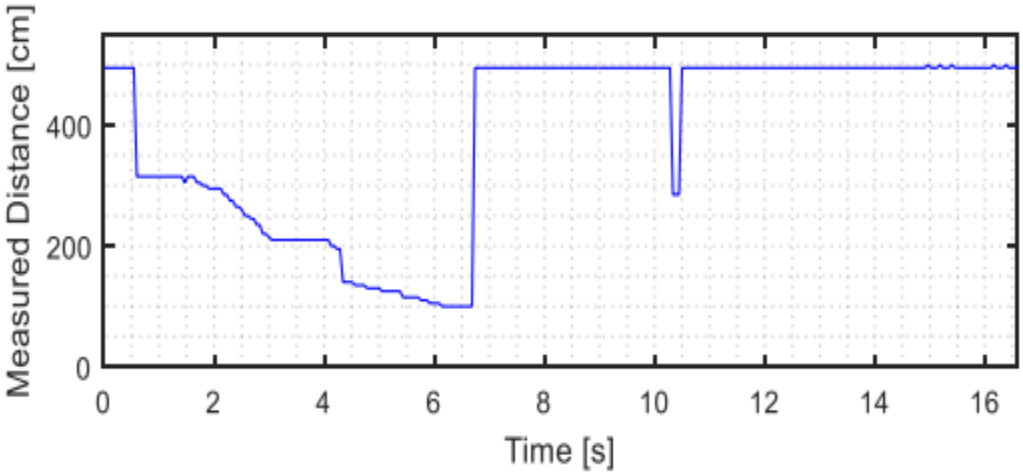


Figure 56 Distance v/s Time

Figure 56[71] indicates that the drone reads the object in its trajectory and pushes itself back in ~6.4 s. Figure 55 then slightly changes its trajectory and successfully avoids the object in its flight plan.

4.2.5 Dashboard visualization

The dashboard is where all the parameters of drones are displayed. Some parameters include current height, battery, and status that represent that the drone is ascending, on the route, hovering, or descending mode.

The dashboard can be broadly classified into two

4.2.5.1 Admin dashboard

An admin dashboard can closely monitor all the drones in the network. The admin has the liberty to change the route, highway, and other parameters manually. It can be considered that when the autonomous mode fails, the admin can handle the conflicts manually through the admin dashboard.

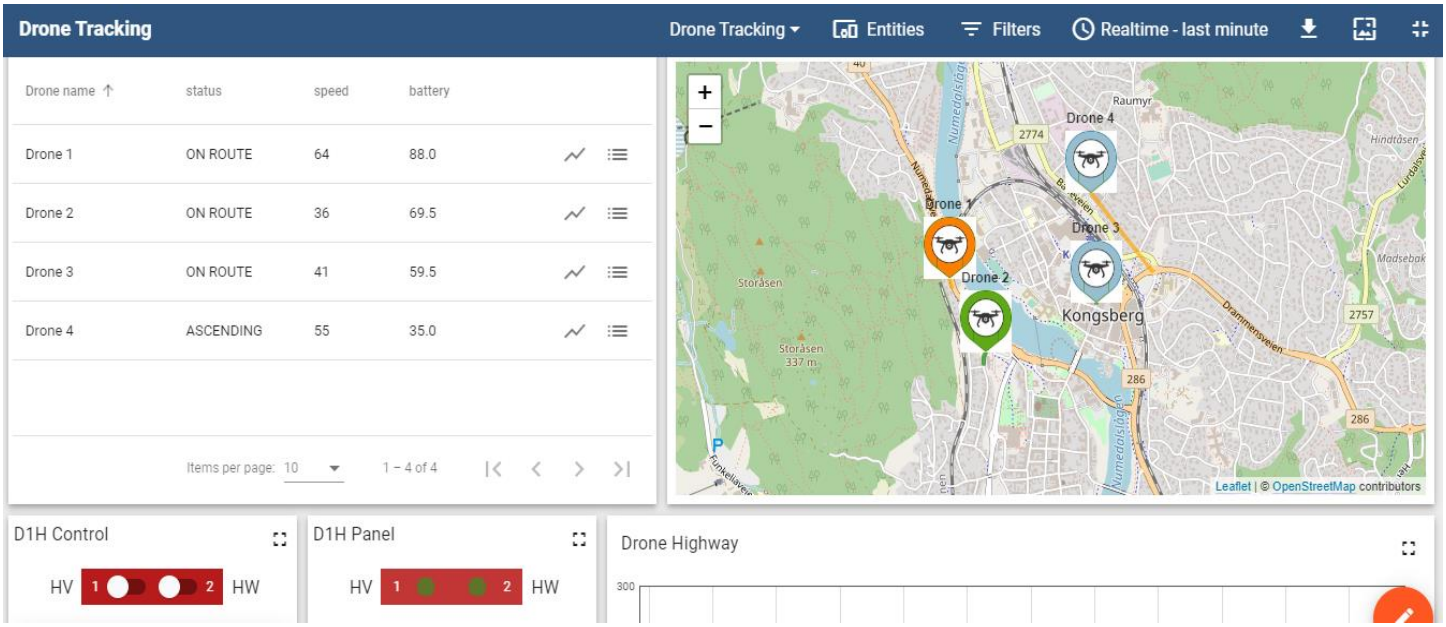


Figure 57 Combined graph

4.2.5.2 Customer dashboard

In a customer dashboard, the parameters of only the specific customer-owned drones are visualized. All the other drones which do not belong to the customer are inaccessible. It entirely depends on the admin to give control access to the customer.

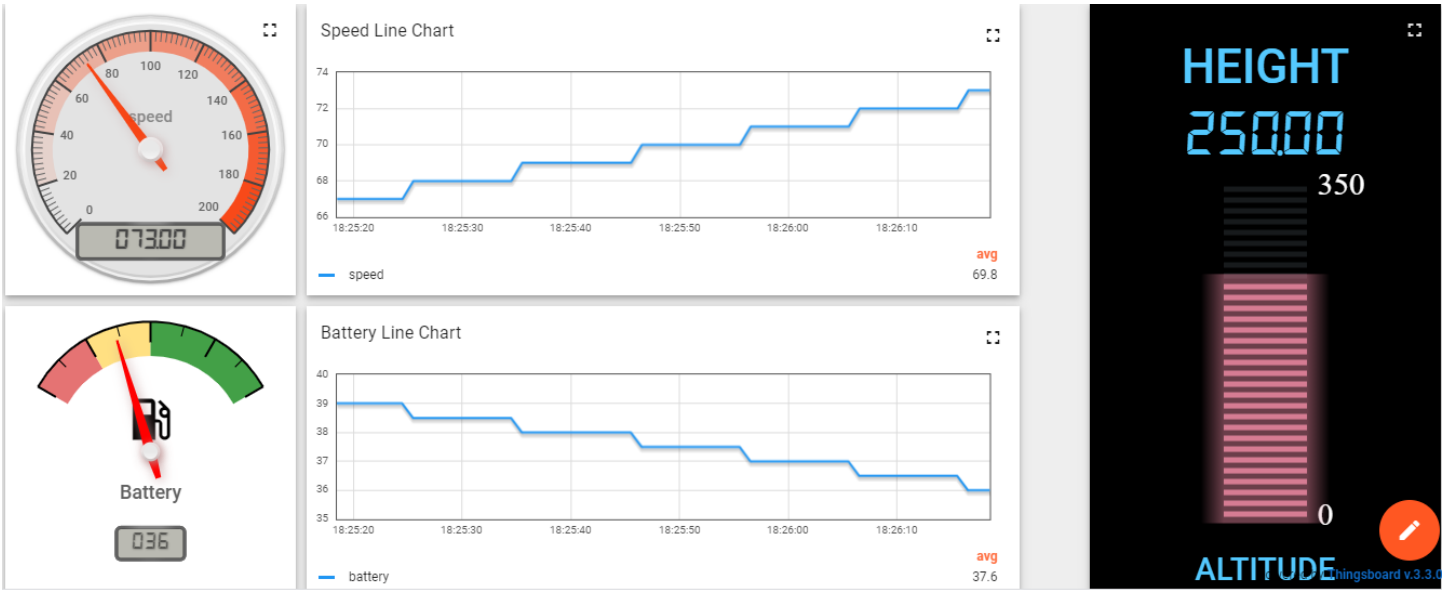


Figure 58 Individual drone graph

Chapter 5

5 Conclusion and Future Work

This work mainly seeks to understand the feasibility of implementing MQTT communication for multi UAVs' communication. The simulation results provide vital insights into UAV communication incorporating MQTT protocol. Following are the conclusion based on the results obtained and the potential future:

5.1 Conclusion

Autonomous aerial vehicles have always been the interest of scholars and researchers. Operating them beyond a certain meter to a few kilometers was an outstanding achievement until the idea of BVLOS. Significant advancements in the BVLOS are made in the past decades, facilitating communication and information exchange. This thesis focuses explicitly on operating a drone over a drone network beyond the VLOS in urban airspace.

MQTT provides a promising service in connecting multiple devices to a central hub.

Number of devices

Theoretically, a single MQTT broker can handle millions of devices over a single network. As seen from the implementation results, on thingsboard, a web-based IoT platform efficiently handles four drones in a single network.

Data delivery

A highly flexible data delivery and acknowledgment feature of MQTT proves to be very beneficial for different applications. Three MQTT QOS provides high flexibility, unlike any other protocol.

Visual Confirmation

A large number of widgets in the thingsboard provide visual confirmation. Therefore, thingsboard proves to be very user-friendly with an attractive dashboard.

Object detection

Object detection by Ultrasonic limits to a short range with limited information of the type, structure, and property of their object detected.

5.2 Future Work

The thesis laid the essential ground implementation of Multi UAV communication over the IoT using MQTT. It uses a basic time-based routing algorithm to send data from a drone to the IoT platform. On taking the encouraging simulation results into account, it is motivating to implement the communication to a more advanced level. Therefore, the following issues that need immediate attention can highly take the project to new heights are listed below.

- Minimal considerations regarding connection failure. A better security and emergency landing system in case of connection failure could highly improve the overall safety.
- An emergency delivery route needs a good focus.
- An optimum and efficient message routing algorithm using MQTT can optimize payload.
- Factors like the wind, atmospheric pressure are not compensated.
- Incorporate LIDAR
    - Ultrasonic sensors can only detect the presence of an object. However, LIDAR can sense other properties of objects such as shape, size.
    - Ultrasonic sensors have a limited range of detection of ~20m. In contrast, LIDAR has a detection range of 75–660m[72].

Reference

[1] S. B. a. M. Hader. "CARGO DRONES: THE FUTURE OF PARCEL DELIVERY." Roland Berger. https://www.rolandberger.com/en/Insights/Publications/Cargo-drones-The-future-of-parcel-delivery.html?country=null (accessed 2021).

[2] J. F. Keane and S. S. Carr, "A brief history of early unmanned aircraft," Johns Hopkins APL Technical Digest, vol. 32, no. 3, pp. 558-571, 2013.

[3] S. R. R. Singireddy and T. U. Daim, "Technology Roadmap: Drone Delivery–Amazon Prime Air," in Infrastructure and Technology Management: Springer, 2018, pp. 387-412.

[4] S. Hamilton and J. Stephenson, "Testing UAV (drone) aerial photography and photogrammetry for archaeology," Lakehead University, Tech. Rep., 2016.

[5] S. Hamilton, "Drone mapping and photogrammetry at Brandon House 4," Historical Archaeology, vol. 51, no. 4, pp. 563-575, 2017.

[6] B. Prodanov, I. Kotsev, T. Lambev, L. Dimitrov, R. Bekova, and D. Dechev, "Drone-based geomorphological and landscape mapping of Bolata Cove, Bulgarian coast," Proc. of IMAM, pp. 592-598, 2019.

[7] S. Ahirwar, R. Swarnkar, S. Bhukya, and G. Namwade, "Application of drone in agriculture," International Journal of Current Microbiology and Applied Sciences, vol. 8, no. 01, pp. 2500-2505, 2019.

[8] D. Murugan, A. Garg, and D. Singh, "Development of an adaptive approach for precision agriculture monitoring with drone and satellite data," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 10, no. 12, pp. 5322-5328, 2017.

[9] K. M. Hasan, S. S. Newaz, and M. S. Ahsan, "Design and development of an aircraft type portable drone for surveillance and disaster management," International Journal of Intelligent Unmanned Systems, 2018.

[10] E. Bassi, "European drones regulation: Today's legal challenges," in 2019 International Conference on Unmanned Aircraft Systems (ICUAS), 2019: IEEE, pp. 443-450.

[11] V. Lappas et al., "Eurodrone, a European utm testbed for u-space," in 2020 International Conference on Unmanned Aircraft Systems (ICUAS), 2020: IEEE, pp. 1766-1774.

[12] P. Kopardekar, J. Rios, T. Prevot, M. Johnson, J. Jung, and J. E. Robinson, "Unmanned aircraft system traffic management (UTM) concept of operations," in AIAA aviation forum, 2016.

[13] A. S. Aweiss, B. D. Owens, J. Rios, J. R. Homola, and C. P. Mohlenbrink, "Unmanned Aircraft Systems (UAS) Traffic Management (UTM) National Campaign II," in 2018 AIAA Information Systems-AIAA Infotech@ Aerospace, 2018, p. 1727.

[14] H. Nakamura, K. Harada, and Y. Oura, "UTM concept demonstrations in Fukushima; overview of demonstration and lesson learnt for operation of multiple UAS in the same airspace," in 2018 International Conference on Unmanned Aircraft Systems (ICUAS), 2018: IEEE, pp. 222-228.

[15] Rahul. "Commercial Drone market explored in the latest research." Semiconductor & Electronics Market Research News. https://www.whatech.com/markets-research/semiconductor-and-electronics/602522-what-will-the-commercial-drone-market-size-be-in-2019-2025-and-what-will-the-growth-rate-be (accessed 2021).

[16] J. Barnard, "Small UAV Command, Control and Communication Issues," in Communicating with UAV's, 2007 IET Seminar on, 2007, pp. 75-85.

[17] O. Tervo, T. Levanen, K. Pajukoski, J. Hulkkonen, P. Wainio, and M. Valkama, "5G new radio evolution towards sub-THz communications," in 2020 2nd 6G Wireless Summit (6G SUMMIT), 2020: IEEE, pp. 1-6.

[18] M. Hassanalian and A. Abdelkefi, "Classifications, applications, and design challenges of drones: A review," Progress in Aerospace Sciences, vol. 91, pp. 99-131, 2017.

[19] G. Singhal, B. Bansod, and L. Mathew, "Unmanned aerial vehicle classification, applications, and challenges: A review," 2018.
[20] Q. A. Abdullah, "Classification of the unmanned aerial systems," ed: Geospatial, 2014.
[21] G. Falciasecca and B. Valotti, "Guglielmo Marconi: The pioneer of wireless communications," in 2009 European Microwave Conference (EuMC), 2009: IEEE, pp. 544-546.
[22] IoTDesignPro. "Types of Wireless Communication Protocols in IoT." https://iotdesignpro.com/articles/different-types-of-wireless-communication-protocols-for-iot (accessed 2021).
[23] P. McDermott-Wells, "What is Bluetooth?," 20 December 2004, Dec. 2004-Jan. 2005.
[24] M. Krzysztoń and M. Marks, "Simulation of watchdog placement for cooperative anomaly detection in Bluetooth Mesh Intrusion Detection System," Simulation Modelling Practice and Theory, vol. 101, p. 102041, 2020.
[25] Moumita. "What is piconet?" TutotialsPoint. https://www.tutorialspoint.com/what-is-piconet (accessed 2021).
[26] V. K. Garg, "CHAPTER 19 - Wireless Personal Area Network — Bluetooth," Wireless Communications & Networking, pp. 653-674, 2007. [Online]. Available: https://doi.org/10.1016/B978-012373580-5/50053-3.
[27] Moumita. "What is scatternet?" TutotialsPoint. https://www.tutorialspoint.com/what-is-scatternet (accessed 2021.
[28] R. Zandbergen, S. Dinwiddy, J. Hahn, E. Breeuwer, and D. Blonski, "Galileo orbit selection," in Proceedings of the 17th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2004), 2004, pp. 616-623.
[29] J. Malhotra, "ZigBee technology: Current status and future scope," in 2015 International Conference on Computer and Computational Sciences (ICCCS), 2015: IEEE, pp. 163-169.
[30] D. Bankov, E. Khorov, and A. Lyakhov, "On the Limits of LoRaWAN channel access," in 2016 International Conference on Engineering and Telecommunication (EnT), 2016: IEEE, pp. 10-14.
[31] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of LoRaWAN for IoT: From technology to application," Sensors, vol. 18, no. 11, p. 3995, 2018.
[32] L. Coetzee, D. Oosthuizen, and B. Mkhize, "An analysis of CoAP as transport in an Internet of Things environment," in 2018 IST-Africa Week Conference (IST-Africa), 2018: IEEE, pp. Page 1 of 7-Page 7 of 7.
[33] C. Bormann. "CoAP, RFC 7252 Constrained Application Protocol." Coap Technology. https://coap.technology/ (accessed 2021).
[34] D. B. Ansari, A. Rehman, and R. Ali, "Internet of things (IoT) protocols: a brief exploration of mqtt and coap," International Journal of Computer Applications, vol. 179, no. 27, pp. 9-14, 2018.
[35] M. Collina, G. E. Corazza, and A. Vanelli-Coralli, "Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST," in 2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications-(PIMRC), 2012: IEEE, pp. 36-41.
[36] O. Akintade, M. Okpako, A. Nasir, and L. Kehinde, "Development of an MQTT Based Collaborative-Aware and Ubiquitous Smart Home Services over IEEE 802. 11: A Safety and Security Case Study," in Proceedings of the OAU Faculty of Technology Conference, Ile-Ife, 2017, pp. 24-27.
[37] VTScada. "MQTT Client Error Codes." VTScada. https://www.vtscada.com/help/Content/D_Tags/D_MQTT_ErrMsg.htm (accessed 2021).
[38] R. K. Kodali and S. Soratkal, "MQTT based home automation system using ESP8266," in 2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), 2016: IEEE, pp. 1-5.
[39] G. J. L. a. S. J. K. Maryam Torabbeigi1, "Drone Delivery Scheduling Optimization Considering Payload-induced Battery Consumption Rates," Journal of Intelligent & Robotic Systems (2020), p. 17, 10 May 2019.

[40] I. M. Arrif, W. Kuntjoro, M. R. Abdullah, and R. E. M. Nasir, "Dynamics and Simulation of Thrust Differential Based Quadcopter," in 2020 IEEE 8th Conference on Systems, Process, and Control (ICSPC), 2020: IEEE, pp. 19-24.
[41] D. C. Browser. "Mi Drone 3D Model." https://www.3dcadbrowser.com/3d-model/mi-drone (accessed 2021).
[42] R. Abeyratne, Convention on International Civil Aviation. Springer, 2014.
[43] C. J. Hegarty and E. Chatre, "Evolution of the global navigation satellite system (GNSS)," Proceedings of the IEEE, vol. 96, no. 12, pp. 1902-1917, 2008.
[44] N. Nadarajah, A. Khodabandeh, and P. J. Teunissen, "Assessing the IRNSS L5-signal in combination with GPS, Galileo, and QZSS L5/E5a-signals for positioning and navigation," GPS solutions, vol. 20, no. 2, pp. 289-297, 2016.
[45] A. Kanj, J. Delporte, N. Suard, B. Bonhoure, and P. Defraigne, "Galileo open service time performance," in 2018 IEEE/ION Position, Location and Navigation Symposium (PLANS), 2018, pp. 1409-1413.
[46] I. J. Gupta, I. M. Weiss, and A. W. Morrison, "Desired features of adaptive antenna arrays for GNSS receivers," Proceedings of the IEEE, vol. 104, no. 6, pp. 1195-1206, 2016.
[47] Y. Wan, F. Chen, J. Nie, and G. Sun, "Optimum reference element selection for GNSS power-inversion adaptive arrays," Electronics Letters, vol. 52, no. 20, pp. 1723-1725, 2016.
[48] A. Brown and H.-W. Tseng, "Miniaturized GPS antenna array and test results," NAVSYS CORP COLORADO SPRINGS CO, 2006.
[49] D. Reynolds, A. Brown, and A. Reynolds, "Miniaturized GPS antenna array technology and predicted anti-jam performance," in Proceedings of the 12th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 1999), 1999, pp. 777-786.
[50] E. Kaplan and C. Hegarty, Understanding GPS: principles and applications. Artech House, 2005.
[51] I. Galileo, "Galileo open service, signal in space interface control document (OS SIS ICD)," ed: European space agency/European GNSS supervisory authority, 2008.
[52] R. Irvine, N. Unit, and A. Seminar, "Investigating the capacity benefit of airborne speed adjustment," in FAA/Eurcontrol ATM Seminar, 2015.
[53] S. Ruiz and M. Soler, "Conflict pattern analysis under the consideration of optimal trajectories in the European ATM," in Proceedings of the 11th USA/Europe Air Traffic Management Research and Development Seminar, ATM 2015, 2015.
[54] P. Brooker, "Controller workload, airspace capacity, and future systems," 2003.
[55] G. Tobaruela, A. Majumdar, and W. Y. Ochieng, "Identifying Airspace Capacity Factors in the Air Traffic Management System," in Proceedings of the 2nd International Conference on Application and Theory of Automation in Command and Control Systems, 2012, pp. 219-224.
[56] L. Berntzen, A. Florea, C. Molder, and N. Bouhmala, "A strategy for drone traffic planning dynamic flight paths for drones in smart cities," in SMART 2019, The Eighth International Conference on Smart Cities, Systems, Devices and Technologies, 2019.
[57] 4gltemall. "Quectel MC60 GSM/GPRS/GNSS Module." https://www.4gltemall.com/quectel-mc60.html (accessed 2021).
[58] Quectel. "5G RM502Q-GL." Quectel https://www.quectel.com/product/5g-rm502q-gl/ (accessed 2021).
[59] Quectel. (2017-05-15). MC60 Series Hardware Design.
[60] Quectel. (2019). Quectel_MC60E_GSM_Specification_V1.0.
[61] Quectel. (2020). Quectel_RM500Q-GL_5G_Specification_V1.0_Preliminary.
[62] Quectel. (2020). EM120R-GL&EM160R-GL Hardware Design V1.0.
[63] V. Petrov et al., "Achieving end-to-end reliability of mission-critical traffic in softwarized 5G networks," IEEE Journal on Selected Areas in Communications, vol. 36, no. 3, pp. 485-501, 2018.
[64] P. Popovski, "Ultra-reliable communication in 5G wireless systems," in 1st International Conference on 5G for Ubiquitous Connectivity, 2014: IEEE, pp. 146-151.

[65] Q. Zhang and F. H. Fitzek, "Mission-critical IoT communication in 5G," in Future Access Enablers of Ubiquitous and Intelligent Infrastructures, 2015: Springer, pp. 35-41.
[66] P. Skarin, W. Tärneberg, K.-E. Årzen, and M. Kihl, "Towards mission-critical control at the edge and over 5G," in 2018 IEEE international conference on edge computing (EDGE), 2018: IEEE, pp. 50-57.
[67] S. S. Intelligence, "ULTRASONIC SENSORS: ULTIMATE ULTRASONIC SENSOR SOLUTION FROM SICK," Online, 2019-10-15 2019.
[68] G. Allegato, C. Valzasina, and L. Zanotti, "Gyroscopes," in Handbook of Silicon-Based MEMS Materials and Technologies: Elsevier, 2020, pp. 899-914.
[69] R. P. Foundation. (2019). Raspberry Pi Compute Module 3+ / Raspberry Pi Compute Module 3+ Lite.
[70] Thingsboard. "Devices." https://thingsboard.io/docs/user-guide/ui/devices/ (accessed 2021).
[71] M. F. Rahman and R. A. Sasongko, "Obstacle avoidance for Quadcopter using Ultrasonic sensor," in Journal of Physics: Conference Series, 2018, vol. 1005, no. 1: IOP Publishing, p. 012037.
[72] C. Edson and M. G. Wing, "Airborne light detection and ranging (LiDAR) for individual tree stem location, height, and biomass measurements," Remote Sensing, vol. 3, no. 11, pp. 2494-2528, 2011.
[73] m.-l. a. a. ikulikov. "simple-mqtt-client." https://github.com/thingsboard/thingsboard/blob/master/tools/src/main/python/simple-mqtt-client.py (accessed 2021).
[74] Gus. "Raspberry Pi Distance Sensor using the HC-SR04." https://pimylifeup.com/raspberry-pi-distance-sensor/ (accessed.

List of Figures

List of Tables

Annexes

A.1 Python Code Listing for MQTT, Ultrasonic sensor integrated[73, 74]

```
import os
import RPi.GPIO as GPIO
import time
import sys
import paho.mqtt.client as mqtt
import json
from array import *

### Pi setup ###
GPIO.setmode(GPIO.BOARD)
Ult_Trigger = 16
Ult_Echo = 18
GPIO.setup(Ult_Trigger,GPIO.OUT)
GPIO.setup(Ult_Echo,GPIO.IN)
GPIO.output(Ult_Trigger, False)
time.sleep(1)


# Thinsgboard Credentials #
THINGSBOARD_HOST = 'demo.thingsboard.io'
ACCESS_TOKEN = 'qUCpd0fBPfbYWQ9vuWcn'

#create a client
client = mqtt.Client()

# Set access token
client.username_pw_set(ACCESS_TOKEN)

# Connect to ThingsBoard using default MQTT port and 60 seconds keepalive interval
client.connect(THINGSBOARD_HOST, 1883, 60)

gPS_CoOrd=[ ## Static GPS Coordinates ## ]

# We assume that all GPIOs are LOW
highway_status = {5:False, 6:False}
highway_temp=0
connection_flag =0
highway_temp_height=0

# The callback for when the Drone receives a CONNACK response from the server.
def on_connect(client, userdata, rc, *extra_params):
###check connection ####
    print("Connection Established  "+ str(rc))
    global flag_connection
    flag_connection =1
```

```
### Authentication ###
    if (ACCESS_TOKEN == 'qUCpd0fBPfbYWQ9vuWcn'):
        print('Authentication Successful')
        time.sleep(1)
        # Subscribing to receive RPC requests
        print("Clear for take-off")
        client.subscribe('v1/devices/me/rpc/request/+')
        # Sending current GPIO status
        client.publish('v1/devices/me/attributes', get_highway_status(), 1)
    else:
        print('Authentication Failure: Take-Off Denied')
        time.sleep(1)
        client.connect(THINGSBOARD_HOST, 1883, 60)



# call back on disconnect
def on_disconnect(client, userdata, rc):
    global flag_connection
    flag_connection = 0



# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
#    print( 'Topic: ' + msg.topic + '\nMessage: ' + str(msg.payload))
    print( 'Topic: ' + str(msg.payload))
    #print("switching Drone to "+ str(data['pin']+ " highway")
    # Decode JSON request
    data = json.loads(msg.payload)
    height=set_height(int(data['params']['pin']), data['params']['enabled'])
    put_height={"height":height}
  #  highway_temp=json.loads(msg.payload)
    # Check request method
    if data['method'] == 'getHighwayStatus':
        # Reply with GPIO status
        client.publish(msg.topic.replace('request', 'response'), get_highway_status(), 1)
        client.publish('v1/devices/me/attributes', get_highway_status(), 1)
        client.publish('v1/devices/me/telemetry',json.dumps(put_height), 1)
    elif data['method'] == 'setHighwayStatus':
        # Update GPIO status and reply
        set_highway_status(data['params']['pin'], data['params']['enabled'])
        client.publish(msg.topic.replace('request', 'response'), get_highway_status(), 1)
        client.publish('v1/devices/me/attributes', get_highway_status(), 1)
        client.publish('v1/devices/me/telemetry',json.dumps(put_height), 1)

def Ultrasonic_read():
    GPIO.output(Ult_Trigger, True)
    time.sleep(0.00001)
    GPIO.output(Ult_Trigger, False)

    while GPIO.input(Ult_Echo)==0:
      pulse_start = time.time()
```

```
    while GPIO.input(Ult_Echo)==1:
      pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start

    distance = pulse_duration * 17150

    distance = round(distance+1.15, 2)
    if distance<=20 and distance>=2:
##        print "Object at distance = ",distance,"m"
        read_flag=1
    else :
        read_flag=0
    return distance, read_flag


def get_highway_status():
    # Encode GPIOs state to json
    return json.dumps(highway_status)


def set_highway_status(pin, status):
    # Output GPIOs state
#    GPIO.output(pin, GPIO.HIGH if status else GPIO.LOW)
    # Update GPIOs state
    highway_status[pin] = status
    #print(pin)

def set_height(pin, stat):
    global highway_temp
    if pin ==5:
        if stat==0:
            print("Switching to ON ROUTE state")
            print("current Highway = "+str(highway_temp))
        else:
            print("Drone in HOVERING STATE")
            print("current Highway = " + str(highway_temp))
    elif pin==6:
        print("Switching Drone highway")
        if (stat==0):
            highway_temp = 250
            print("Drone in Highway 1" + str(highway_temp))
        else:
            highway_temp = 300
            print("Drone in Highway 2" + str(highway_temp))
    return highway_temp


# Register connect callback
client.on_connect = on_connect
# Registed publish message callback
client.on_message = on_message
# register disconnect call back
client.on_disconnect = on_disconnect
```

```
speed =0 ## Dependent on rotors
battery =60 ## ADC read using voltage divider.
client.loop_start() ## starts/initiates the Mqtt connection
CoOrd_numbers= len(gPS_CoOrd)
##print(CoOrd_numbers)
gPS_CoOrd_Track=0
time.sleep(2)

###     Function to get latitude    ###
def get_latitide(gPS_CoOrd_Track):
    latitude_ret =gPS_CoOrd[gPS_CoOrd_Track][1]
    return latitude_ret

###     Function to get Longitude   ###
def get_longitude(gPS_CoOrd_Track):
    longitude_ret= gPS_CoOrd[gPS_CoOrd_Track][0]
    return longitude_ret

###     Function to print the Drone parameters ###
def print_parameter(gPS_CoOrd_Track):
    latitude_ret =gPS_CoOrd[gPS_CoOrd_Track][1]
    longitude_ret= gPS_CoOrd[gPS_CoOrd_Track][0]
    print("latitude = "+ str(latitude_ret)+" longitude = "+str(longitude_ret))
    print("battery = "+str(battery)+("  speed = "+str(speed)))

def grounded():
    put_data = {"latitude":latitude,"longitude":longitude,"battery":
battery,"speed":0,"status":"ARRIVED","D3_altitude":" - ","connection":" 0 ","height": " 0
"}
    client.publish('v1/devices/me/telemetry',json.dumps(put_data), 1)

mycount=0


def attin_height(highway_temp,speed):
    temp = 0
    print("Current height = "+str(highway_temp)+" M")
    put_data = {"latitude": gPS_CoOrd[0][1], "longitude": gPS_CoOrd[0][0], "D3_altitude":
highway_temp,
                "status": "ASCENDING", "height": highway_temp, "speed": speed}
    client.publish('v1/devices/me/telemetry', json.dumps(put_data), 1)
    print("Climbing up to 250M altitude....")
    temp =1
    while highway_temp < 250 :
        if (highway_temp%35)==0:
            speed+=5
        time.sleep(0.05)
        highway_temp+=1
        put_data = {"latitude":gPS_CoOrd[0][1],"longitude":gPS_CoOrd[0][0],"D3_altitude":
highway_temp,"status":"ASCENDING","height":highway_temp,"speed":speed}
        client.publish('v1/devices/me/telemetry', json.dumps(put_data), 1)
        if (highway_temp % temp) == 0:
            print("Current height = "+str(highway_temp)+" M")
```

```
            temp+=temp
    print("Altitude Attained = "+str(highway_temp))
    return highway_temp, speed

def ground_height(highway_temp,speed):
    print("Climbing Down to 0M")
    while highway_temp >0 :
        if(speed>5):
            speed-=0.5
        highway_temp-=1
        time.sleep(0.05)
        put_data = {"latitude":latitude,"longitude":longitude,"D3_altitude":
highway_temp,"status":"DESCENDING","height":highway_temp}
        client.publish('v1/devices/me/telemetry', json.dumps(put_data), 1)
    print("Current altitude "+ str(highway_temp))
    return highway_temp,speed


highway_temp,speed=attin_height(highway_temp,speed)
try :
    while ACCESS_TOKEN == 'qUCpd0fBPfbYWQ9vuWcn':
        print_flag=0
        distance, flag_dist = Ultrasonic_read()
        while((highway_status[5]==True) or (flag_dist == True) ):
            print("Hovering " +str(flag_dist))
            print ("latitude : "+str(latitude),"longitude : "+str(longitude))
            if highway_status[5]==True :
                if gPS_CoOrd_Track==0 & print_flag==0:
                    put_data = {"latitude":59.674,"longitude":9.65052,"battery":
battery,"speed":speed,"status":"HOVERING","D3_altitude":highway_temp,"connection":flag_co
nnection,"height":highway_temp}
                    client.publish('v1/devices/me/telemetry',json.dumps(put_data), 1)
                elif gPS_CoOrd_Track!=0 & print_flag==0:
                    put_data = {"latitude":latitude,"longitude":longitude,"battery":
battery,"speed":speed,"status":"HOVERING","D3_altitude":highway_temp,"connection":flag_co
nnection,"height":highway_temp}
                    client.publish('v1/devices/me/telemetry',json.dumps(put_data), 1)
            elif flag_dist == True  :
                print ("Object at distance = ", distance, "m")
                time.sleep(0.1)
                put_data = {"status":"HOVERING","speed":" 0 "}
                client.publish('v1/devices/me/telemetry',json.dumps(put_data), 1)
            else:
                break
            distance, flag_dist = Ultrasonic_read()
            time.sleep(1)
        mycount += 1
        #Print(gPS_CoOrd_Track)
        latitude = get_latitide(gPS_CoOrd_Track)
        longitude = get_longitude(gPS_CoOrd_Track)
        put_data = {"latitude":latitude,"longitude":longitude,"battery":
battery,"speed":speed,"status":"ON
ROUTE","D3_altitude":highway_temp,"connection":flag_connection,"height":highway_temp}
        client.publish('v1/devices/me/telemetry',json.dumps(put_data), 1)
```

```
        #print(sys.getsizeof(put_data))
        time.sleep(1)
        if (gPS_CoOrd_Track %10) == 0 :
            speed += 1
            battery-=0.5

        if gPS_CoOrd_Track >= CoOrd_numbers-1:
            print ("Arriving to Destination...")
            highway_temp = ground_height(highway_temp,speed)
            break
        else:
            gPS_CoOrd_Track +=1
            print_parameter(gPS_CoOrd_Track)
            if (gPS_CoOrd_Track % 5) == 0:
                print ("ON ROUTE")
    time.sleep(1)
    grounded()
    time.sleep(1)
    print ("Destination Arrived ")
    client.disconnect()
except KeyboardInterrupt:
    print ("Emergency shutdown")
    put_data = {"latitude":latitude,"longitude":longitude,"battery":
battery,"speed":0,"status":"EMERGENCY SHUTDOWN","D3_altitude":" - ","connection":" -
","height": " - "}
    client.publish('v1/devices/me/telemetry',json.dumps(put_data), 1)
    client.disconnect()
    GPIO.cleanup()
```

S