INTEGRATION OF UAVS WITH REAL TIME OPERATING SYSTEMS AND ESTABLISHING A SECURE DATA TRANSMISSION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Niranjan Ravi

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

August 2019

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF THESIS APPROVAL

Dr. Mohamed El-Sharkawy, Chair

Department of Engineering and Technology

Dr. Brian King

Department of Engineering and Technology

Dr. Maher Rizkalla

Department of Engineering and Technology

Approved by:

Dr. Brian King

Head of the Graduate Program

My beloved family

Ravi Krishnamoorthy and Meenakumari Ravi

ACKNOWLEDGMENTS

I would like to thank Dr. Mohamed El-Sharkawy for giving me constant support, motivation and guidance without whom this research would not have been possible.

To Dr. Brian King and Dr. Maher Rizkalla for being a part of thesis committee and their valuable inputs.

To Sherrie Tucker for her timely help in finishing up curriculum materials before the deadlines.

To NXP semiconductors for providing advanced hardware and software technology for this research.

Finally I would like to thank Dewant Katare, Siva Teja Reddy, Rajas Chitavis, Shivani Rachamalla and entire IoT laboratory for having trust in me and supporting me through the entire tenure.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Niranjan, Ravi. M.S.E.C.E., Purdue University, August 2019. Integration of UAVS with Real Time Operating Systems and Establishing a Secure Data Transmission. Major Professor: Mohamed El-Sharkawy.

In today's world, the applications of Unmanned Aerial Vehicle (UAV) systems are leaping by extending their scope from military applications on to commercial and medical sectors as well. Owing to this commercialization, the need to append external hardware with UAV systems becomes inevitable. This external hardware could aid in enabling wireless data transfer between the UAV system and remote Wireless Sensor Networks (WSN) using low powered architecture like Thread, BLE (Bluetooth Low Energy). The data is being transmitted from the flight controller to the ground control station using a MAVlink (Micro Air Vehicle Link) protocol. But this radio transmission method is not secure, which may lead to data leakage problems. The ideal aim of this research is to address the issues of integrating different hardware with the flight controller of the UAV system using a light-weight protocol called UAVCAN (Unmanned Aerial Vehicle Controller Area Network). This would result in reduced wiring and would harness the problem of integrating multiple systems to UAV. At the same time, data security is addressed by deploying an encryption chip into the UAV system to encrypt the data transfer using ECC (Elliptic curve cryptography) and transmitting it to cloud platforms instead of radio transmission.

1. INTRODUCTION

Internet has become a part of every day's life to gather and process the information collected across the globe. The development in various fields of technology fostered rapid growth in the area of research and paved a new path for homogeneous as well as heterogeneous modes of communication between humans and things[1]. This helped researchers to build new solutions for problems faced by human society. The advent of WSN, the applications of embedded systems have been more feasible. A WSN is a small network of spatially distributed autonomous nodes that communicate with all others in the network using radio signals. WSN consists of many independent low costs, micro-sensor nodes capable of sensing, processing the data and communicating them further. The micro-sensor node would consist of a sensor, a microcontroller or a microprocessor, a wireless transmitter with an inbuilt power source[2][3]. One of the ideal objectives of a WSN is sensing the environmental data like temperature, air pollution level in their respective locations. Employing WSN for data collection, a variety of approaches such as BLE, Thread, and ZigBee are used to transmit the data wirelessly between the WSN nodes without the use of the internet. This would help to develop more low powered architecture since it would work even if there is no internet. Wireless Mesh Network like Thread protocol is efficient when information has to be transmitted from remote WSN to other locations because it provides no single point of failure, which ensures no data loss during the process[4]. Also, the WSNs are characterized by diversity because there are many different proprietary and non-proprietary solutions. At the same time, the advent of Unmanned Aerial Vehicle (UAV) systems has drastically helped human kind in various ways. These remotely piloted aircrafts are employed in various types of applications from being a surgical strike drones in military sector to commercial deployments like agriculture, public safety and data collection from various terrains.

1.1 Motivation

There has been a pressing need from industrial and medical sectors to make this WSN data readily available for advanced research activities in the areas where it is onerous to deploy human resources for data collection. For example, during the times of natural calamities, it is difficult to deploy human resources to the affected area. The real challenge arises when the need to transmit WSN data from remote locations to research destinations without the usage of the internet cause the signal towers may get affected during crisis. On the other hand, when any sensor network collects the sensitive information like data from radioactive elements in a nuclear power plant or any transmission of security credentials, it has to be transmitted without any data loss or any data leakage. Secure data transmission still remains as one of the unconquerable tasks in many developed countries. And more importantly, when WSN collects the data, it is challenging to implement algorithms for increased data security on WSN nodes owing to their low processing power. The range of data transmission and establishing a secured data transmission is one of the significant questions that arise in the minds of developers all around the world.

1.2 Research Objectives

The ideal aim of this research is to propose a system with the help of UAV and ECC to address these above-mentioned challenges. UAV systems are growing at a rapid rate in various aspects of life, including the dispatch of medicines and undergo video surveillance during an emergency due to less air traffic. The ability of UAV to follow a pre-programmed flight path also aids in the design since they give the ability for UAV to travel without any barriers and irrespective of the terrains. UAVs are used to collect the data which has been gathered by WSN at remote locations where internet connectivity is not predominant. The architecture consists of a UAV system appended with a Real-Time Operating System (RTOS) using UAVCAN. This enables the UAV system to use Thread Protocol enabled in RTOS to receive the data wirelessly without internet. With regards to the challenge of secure data transmission, the advancements in the field of Internet of things (IoT) are used. The data from WSN are sent to destined locations with the help of cloud platforms. The passage of the data from micro-sensor nodes to cloud is secured with ECC which has been devised for low power IoT devices. An encryption chip was attached to the UAV system to encrypt the data using cryptographic algorithms and upload them to the cloud. That would avoid data leakage during radio transmission and no data loss owing to real-time data upload.

2. BACKGROUND STUDY

2.1 Background Study On Wireless Sensor Networks

2.1.1 Overview of IEEE 802.15.4 Standard

IEEE 802.15.4 is a standard that allows low cost and low power communications. This standard specifies many Physical Layers (PHYs) operating in many different frequency bands with an efficient Medium Access Control (MAC). They also define information Elements (IEs) that can be used to extend the standard practice. An IE provides an extensible, flexible, and easily implementable method of encapsulating information. The general frame format of IEEE 802.15.4 consists of an identification (ID) field, a length field, and a content field. IE acts as a flexible container for information that allows the addition of new IE definitions in further versions of the standard[5].

2.1.2 IPv6

Internet Protocol version 6 (IPv6) is an updated version of Internet Protocol, and it is a successor to IP version 4.IPv6 increases the IP address size from 32 bits to 128 bits. This would support more levels of addressing hierarchy, a more significant number of addressable nodes can be present at the network, and provide a simple auto-configuration of the addresses. Some IPv4 header fields have been dropped or made optional to reduce the common-case processing cost of packet handling and to limit the bandwidth cost of the IPv6 header, a few IPv4 header fields have been removed[6].

2.1.3 6LowPAN

6LowPAN stands for IPv6 over Low Power Wireless Personal Networks. The main objective of 6LoWPAN is to send and receive IPv6 packets over the 802.15.4 link. This has to accommodate for sending the 802.15.4 maximum frame size over the air. In case of Ethernet links, a packet with the size of IPv6 Maximum Transmission Unit (MTU) can be transmitted in a single frame over the link. 6LoWPAN acts as an adaptation layer in between the IPv6 networking layer and the 802.15.4 link layer. This helps in solving the issue of transmitting an IPv6 MTU by fragmenting the IPv6 packet at the sender side and reassembling them at the receiver. 6LoWPAN also provides a compression mechanism which reduces the IPv6 header size which is sent over the Air, in turn, reducing the transmission overload. If fewer bits are sent over the air, less energy consumed by the device. Thread network makes full use of this potential to transmit the packets over the 802.15.4 network. Header compression is used within the Thread network and devices which are sending the messages and compressing the IPv6 header to reduce the size of the transmitted packet. Mesh header supports for much more efficient compression of messages inside a Thread network. Thread stack, used a route-over configuration method which helps the nodes in IP-level visibility into the underlying radio connectivity[8].

2.1.4 Constrained Application Protocol

For the resource constrained nodes and networks in the field of IoT, specialized web transfer protocol called a constrained application protocol (CoAP) is used. CoAP acts like a lighter version of Hyper Text Transfer Protocol (HTTP) and RESTful features to run on resource-constrained embedded devices/systems. CoAP uses User Datagram Protocol (UDP) protocol rather than TCP (Transfer Control Protocol). Confirmable requests (CON) from parent nodes get acknowledgments as a response from their respective destination nodes.

2.2 Background Study On Thread Protocol

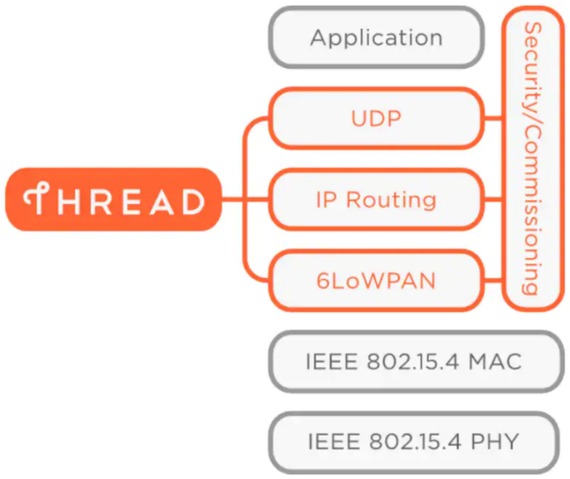


Fig. 2.1. Thread Network Stack[40]

Thread is an IP based wireless networking protocol targeting low power embedded IoT applications. The thread is built based on several standards by combining their advantages, cost-effectiveness, low-power consumption, and reliability. Having no standard application layer enables developers to use any necessary application layer in the Thread network. The Thread employs UDP for the network layer (NWK) on top of IP routing and 6LoWPAN. Thread also utilized IEEE 802.15.4 MAC and PHY wireless specification and supporting mesh network with a maximum speed of 250 Kbps in 2.4 GHz band.

2.2.1 Network Architecture of Thread

The advancements in the field of wireless communication enhanced the networks architecture of thread to support a wide range of platforms nowadays.

Developers/users can communicate within a Thread network from their devices like smartphone, tablet, or even a computer via Wi-Fi on their Home Area Network (HAN) or by using a cloud-based framework. The figure illustrates the key device types in the thread network architecture.

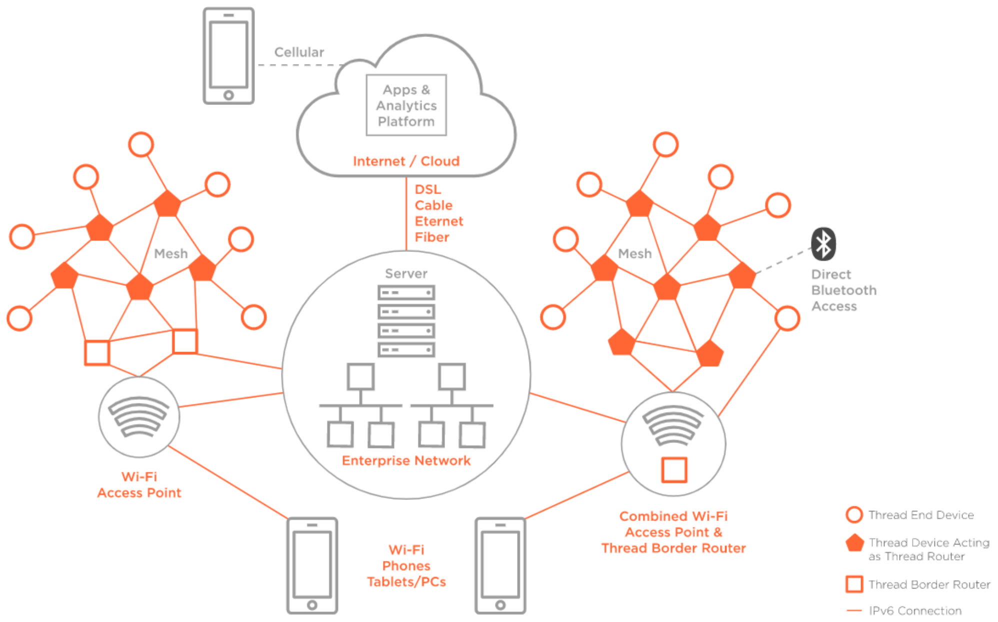


Fig. 2.2. Commercial Thread Network Topology[13]

2.2.2 Border Router

Border Router provides connectivity from network to adjacent networks on other physical layers such as Wi-Fi and Ethernet. They provide services for the devices within the network including routing services for off network-wide Operations. There are at least one border routers in a thread network, and it is possible to have many[9][10].

A leader manages a registry or entry of assigned router IDs and accepts request from router eligible end devices (REEDs) to become a router. The leader decides which devices should be routers and the end-devices. The Assigning and managing of router address is done using the Constrained Application Protocol (CoAP).

The routers and the leader share the thread network information, so if the leader fails or loses connectivity with the thread network, then another router is elected and takes over as a leader without any user intervention[11].

2.2.3 Thread Router

Thread Router provides routing services to the devices in the network. They also offer to join security services for devices trying to join the network. Although thread routers are not designed to sleep, they can downgrade their functionality and become REEDs.

2.2.4 Router Eligible End Devices(REEDS)

REED is router eligible end devices that can become a thread router or a leader but not necessarily a border router that has many properties such as multiple interfaces. Due to network topologies, REEDs do not act as routers[4] or provide security services for other devices in the network. The REEDs act as a remote end the device which is eligible to become a router under certain circumstances. The network manages itself and automatically promotes the REEDs to the router if necessary without any user intervention into the system.

2.2.5 Sleepy End Devices

Sleepy end devices can transmit messages and communicate with their thread router parent. However, they cannot relay messages for other devices in the network. Sleepy end devices in a thread network can give an extended battery life[12].

2.2.6 No Single Point of Failure

The thread stack or network is designed in such a way that they do not have a single point of failure. The devices in the network can perform various functions and are designed in such a way that thread can be replaced without replacing the ongoing operation of the network or devices. For instance, the sleepy end devices need a parent to communicate and transmit data, but if the parent fails due to connectivity issues or any other losses, the sleepy end devices can select a new parent to communicate. This mode of transition is invisible to the user. This allows the thread stack to work with no single point of failure. For example, if a system has a single border router and if the border router loses power, then there is no means to switch to an alternative border router. In this scenario, a reconfiguration of the border router must take place to keep the system stable. We can avoid this scenario by having more than one border router, as shown in the Fig. 2 to have a foolproof system[13].

2.2.7 Security Layers in Thread Network

Security is being provided at the network layer and can be applied at application layer as well. New devices can join a thread network after authentication procedures. All devices in a thread network are authenticated using a simple smartphone-era authentication scheme and advanced encryption standard (AES) encryption[7]. This makes the thread more secure compared to other existing wireless networks[11].

2.3 Background Study On Real Time Operating Systems

The field of embedded systems has been growing at an exponential rate over the last few years. The number of embedded systems employed in our day to day life is increasing at a rapidly ranging from a coffee maker to autonomous cars development of small, compact embedded systems capable of performing various operations in real time. This is called multi-tasking. But owing to the processing of multiple processes at the same time, a need for an operating system to work in real time had developed and this paved way to the development of RTOS. RTOS is defined to be a software component that rapidly between the tasks, which gives an impression that multiple programs are executed simultaneously on a single processing core. But on the contrary, the processor can run a single thread at a time, but RTOS decides the priority of the task, which aids the processor to switch between various tasks[14].

2.3.1 FreeRTOS

FreeRTOS is designed to be small to run on a microcontroller while the scope of FreeRTOS is not restricted to microcontroller applications alone. A microcontroller has a read-only memory (ROM or flash) to hold the executing program and random access memory (RAM) which contains the memory of the instructions during the execution. The microcontroller is a part of deeply embedded applications that has a specified task to perform. This dedicated end application requires the use of a full RTOS implementation. RTOS provides core real-time - scheduling, timing with synchronization, and inter-task scheduling, which is working in the background. This is described as a real-time kernel[15]. RTOS is selected in systems which requires a quick, highly deterministic response to external events. Critical features of RTOS are discussed below:

1. Interrupt Latency - Real-time systems are designed to carry out tasks/thread within an absolute worst-case scenario in terms of time complexity. This would

result in creating a system with a higher degree of reliability when compared to an operating system[16].

2. Resource-constrained processors - Microkernels in RTOS systems deliver a hard real-time response which places a crucial role in embedded microprocessors with limited RAM/ROM[14].

3. Priority Scheduling - Real-time operating systems schedule the tasks based on the priority level decided by designers. High priority tasks are given about 100 percent importance compared to a low priority task[16]. RTOS system is also programmed carefully to avoid catastrophic situations like deadlock when two or more tasks/threads are each waiting for others to finish their tasks.

4. Improved Efficiency - RTOS is an event-driven system. These systems would increase the efficiency by reducing the processing time on the tasks which never occurred during the run-time of the processor[16].

2.4 Background Study On Internet Of Things

2.4.1 Advancements in the Field of Embedded Systems

The field of embedded systems is advancing at a constant pace in day to day's life. While designing an embedded system could be relatively simple, but the features and results provided by embedded systems are beyond what a human could offer. For example, aviation systems are constructed as an embedded system to integrate sensor data and provide real-time sensor output. This approach created a path to explore the data from embedded systems by a new feature called the Internet of things (IoT).

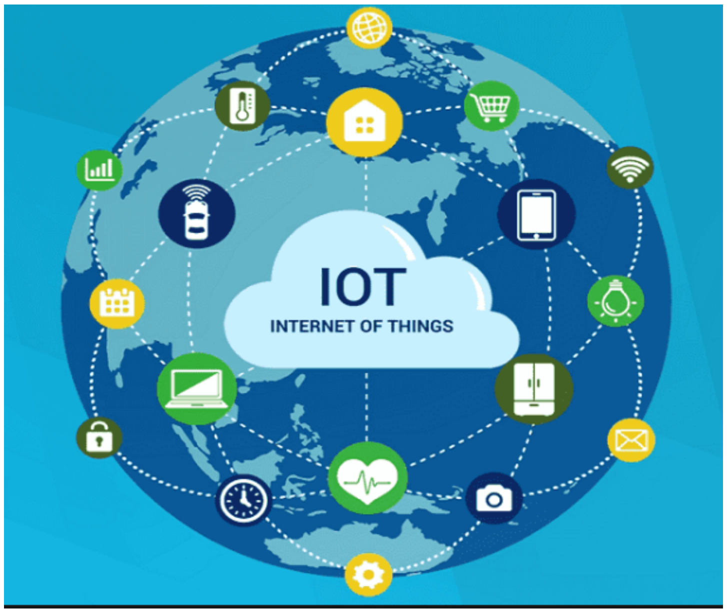


Fig. 2.3. Internet of Things[43]

2.4.2 Internet Of Things

The term IoT was coined initially by Auto-ID center in 1999, which visualized the future where every physical object would be tagged by installing a radio frequency identification (RFID) tag which would have a unique identifier globally[1].

Internet of things is a platform for a network of embedded physical devices are built by electronic components, software, sensors and establishing a communication between the connected devices in a system or with the manufacturer. The term IoT commonly refers to a collection of devices that are generally resource-constrained in terms of computation and communication capabilities. This structure has been dramatically modified by establishing a connection between all IoT devices to the internet and various services, and features. This would result in the development of more machine to machine communication using the internet without any human intervention. IoT also paves the way for connecting many heterogeneous networks in which each system adopts a different communication pattern such as human-to-human (H2H), thing-to-things (t2T) or thing-to-thing (t2T). So, this would give an advantage of monitoring the devices in case of hardware and software issues remotely and also aids in analyzing the data over the internet. The concept of IoT has widely expanded, and now it encloses a wide variety of technologies and communication protocols.

The primary reason for the interest of the research community to redesign and deploy new enhancements into the field of IoT. The embedded systems which are a part of IoT systems can understand and act according to the environment where they are located. They are referred to as smart objects or smart devices. The introduction of IPv6 and CoAP as basic blocks for IoT applications allows the establishment of the connection between IoT hosts to the internet[1]. That would amplify the current advantages to the existing systems, including the development of various devices that vary in computational capacities, different systems can be integrated and work towards a common goal, an integrated interface for applications that remove the needs for application-level procedures.

2.5 Background Study On Cloud And Encryption

2.5.1 Cloud Computing

Cloud computing platform holds a specific position in today's IoT market. We could understand that the IoT devices are resource-constrained. Hence, complex algorithm calculations and implementations on them would be difficult and would result in the modification of hardware with advanced features an increase in the size and cost of manufacturing an IoT device. An alternate approach is establishing a cloud platform support for IoT devices. Cloud platform comes with advanced features for data handling, storage, and unlimited distributions. There are many cloud platforms available at a little cost, which help a researcher to expand the scalability of his existing IoT product[17]. This integration brings up many unanimous advantages for IoT devices such as flexibility and capacity to store unlimited data. Another edge of the cloud platform is that it helps the user to understand the characteristics of his IoT device located in a remote region by simple accessing his laptop or mobile phone. This approach would avoid false alarms and saves human resources. The most important feature of a cloud platform is that they help the developer to witness the results of his product in real time and deploy complex machine learning algorithms to analyze the pattern on them.

2.5.2 Elliptic Curve Cryptography

Elliptic curve cryptography is a public-key technology offers improved performance for resource-constrained devices at higher security levels.

This includes the elliptic curve version of the Diffie-Hellman key exchange and the elliptic curve versions of the ElGamal signature algorithm. This provides a two-way authentication scheme for IoT based devices on pre-existing internet standards. In today's environment, we could visualize that the IoT with a wireless sensor networks being used in a huge magnitude. For these devices, the goal is to make the collected

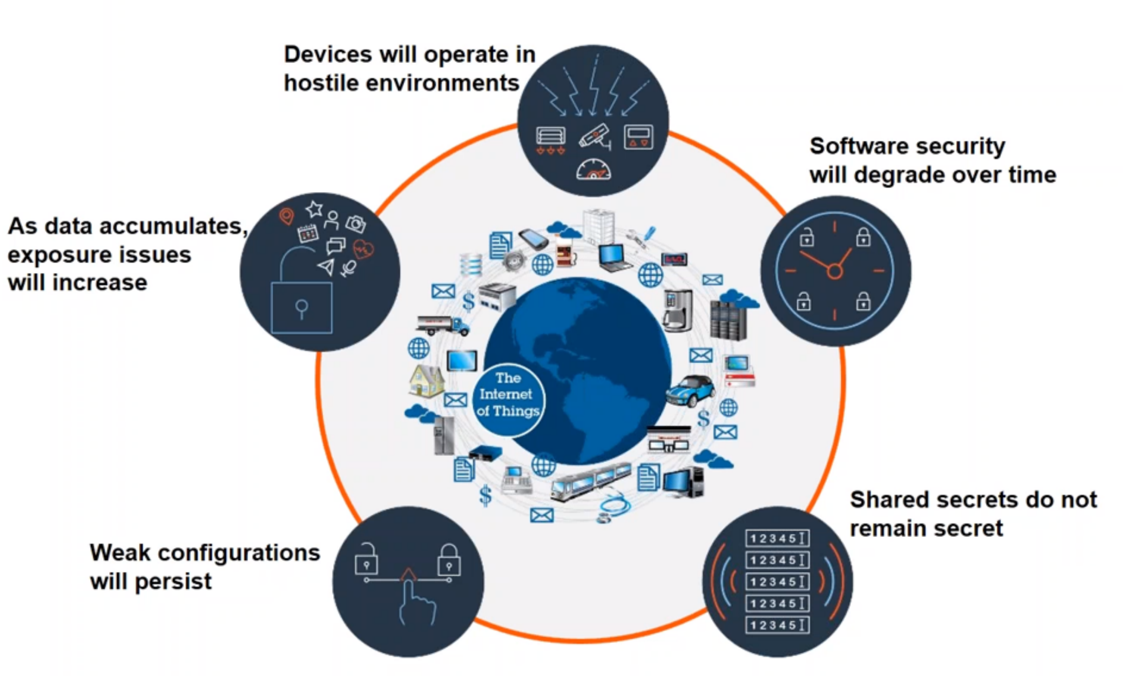

Devices will operate in hostile environments
Software security will degrade over time
Shared secrets do not remain secret
Weak configurations will persist
As data accumulates, exposure issues will increase
The Internet of Things
12345
12345
12345
12345
12345

Fig. 2.4. The Need For Data Security

data globally available to all users. These data vary from a temperature value from a sensor to crucial data like medical records. While these are being transmitted to cloud platforms or exchanged between two wireless sensor networks, a pressing need for security solutions always arises. Therefore, an efficient end to end security is required to achieve an adequate level of security between the applications. This would result in development of a system in which data breach can be considerably reduced[18]. ECC is a public-key technology that offers improved performance at the higher security levels for Transport Layer Security (TLS). ECC provides smaller key sizes that would help in saving the computational cost, power and memory which are the core features of resource-constrained devices[45]. TLS Handshake is responsible to responsible for authentication and key exchange to establish secured connection. TLS protocol version and encryption techniques like asymmetric or symmetric encryption and other factors are managed by the TLS Handshake protocol. The full specifications can be found in [37][45].

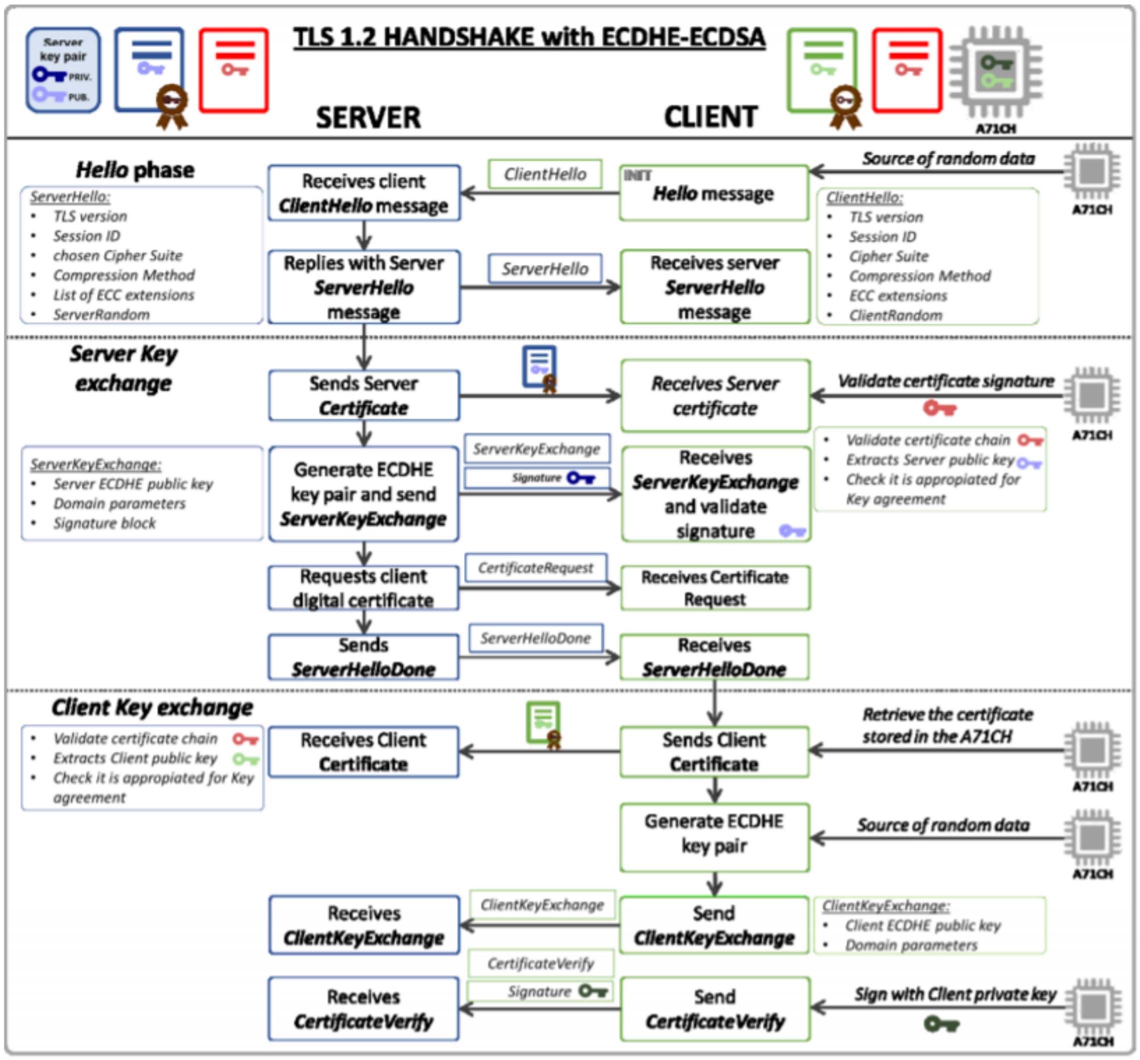


Fig. 2.5. TLS Handshake with ECDHE-ECDSA[37]

2.6 Related Study On UAV Systems

The rapid technological developments in the field of electronics and sensors, gave rise to the usage of UAV in the field of both civilian and military operations[20]. UAV systems deployed in an area where it is challenging to deploy defense forces. They can be used for surveillance of a city or could also use as a mode of air-strike on enemy targets. Scientists all over the world work on the usage of UAV systems in various aspects of life. This range from deploying a drone during a time of disaster to look for human survivors or even to have secret surveillance on enemy camp during the time of war. The primary reason for the uprising scalability of UAV systems is that they are easy to construct, economical, and their ability to predetermined work. The principal activities of drones include:

- Environmental and scientific research.
- As a long-range weapon in military air strikes.
- Aerial surveillance.
- Public safety.

A typical UAV system would consist of a GPS module to aids in navigation of the drone to the destined path and a telemetry kit which allows observing the flight data wireless from the ground/base station. To obtain the flight dimensions in real time, the flight controllers are embedded with an accelerometer, gyro meter and magnetometer[10]. They also include a ground station such as a PC/laptop which would send and receive data from the drones through telemetry device. The ground control station gives an option to upload a predefined path for the UAV system. Other option is operating the drone in manual mode for testing and trial purpose.

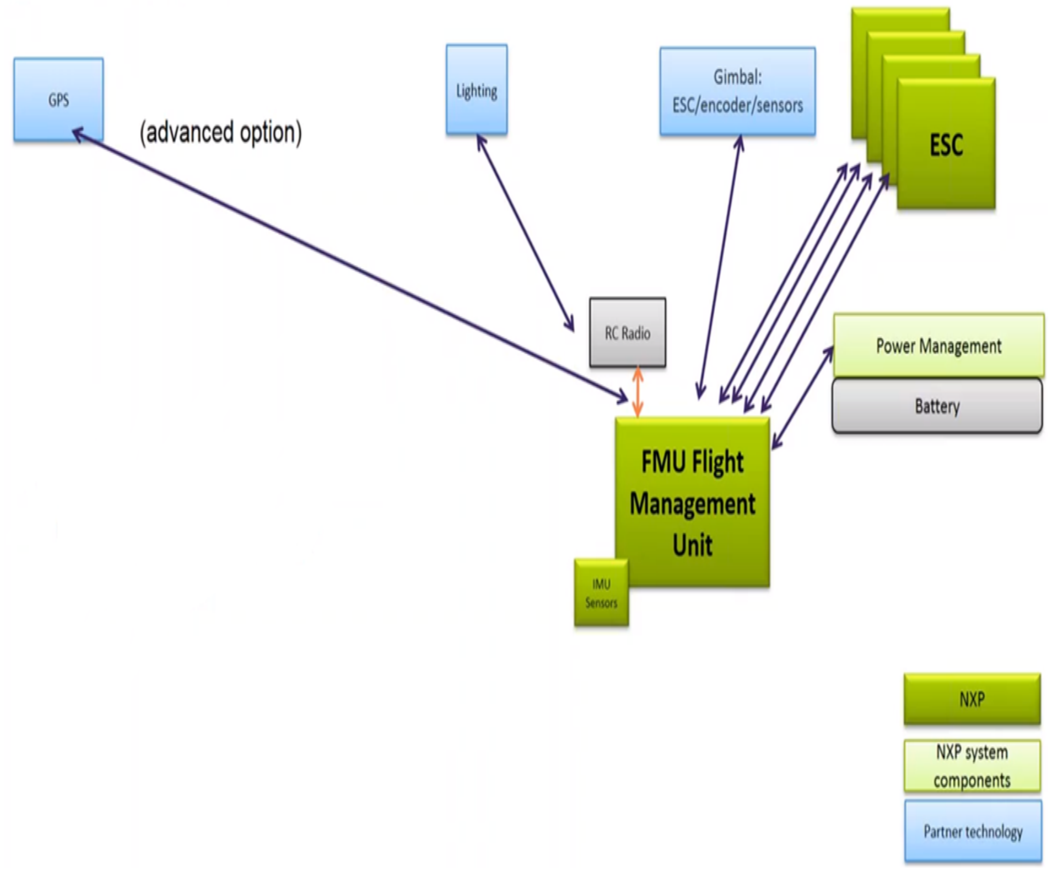


Fig. 2.6. Traditional UAV Design[10]

2.6.1 Flight Controller

Flight controller acts as a central part of a UAV system. The flight control method of aircraft gradually developed from traditional mechanical and hydraulic systems to ”Fly-By-Wire (FBW)” owing to advancements in the field of electronics. The use of fiber optics in aircraft provides considerable advantages[21]. FBW provides an electronic interface to control the flight systems. The flight movements are, in turn, converted into electric signals and are transmitted to the base station to monitor the fly activity. It also provides automatic signals to be sent to the aircraft unit without the pilot’s input, which helps in stabilizing the systems. FBW systems are lightweight, easier to maintain as compared to its predecessor methods. There are various flight controllers available in the market on a current day, and it varies depends on the scope of the application. Since the flight controller acts as a heart of the UAV system, the entire operation is navigated through it[22]. Pixhawk and Ardupilot are predominately used flight controllers for a UAV system[23].

2.6.2 QGroundControl

QGroundControl provides a complete vehicle setup and flight control for PX4 and other Ardupilot powered vehicles. It provides full setup and configuration for PX4 software. Currently, PX4 is predominantly used for developing the UAV applications. QGC delivers the ability to design autonomous fight path for autonomous missions. The flight map displays waypoints and flight track, which are essential to plan the mission/path. It also provides a simple User Interface (UI). QGC aids in updating the firmware in the flight controller over a Universal Serial Bus (USB), an air-frame selection that helps to choose a particular air-frame and calibrate the sensors before the takeoff. This could prevent unwanted crash of the drone owing to calibration issues. Developers can monitor the battery level in the UAV system through QGC[24].

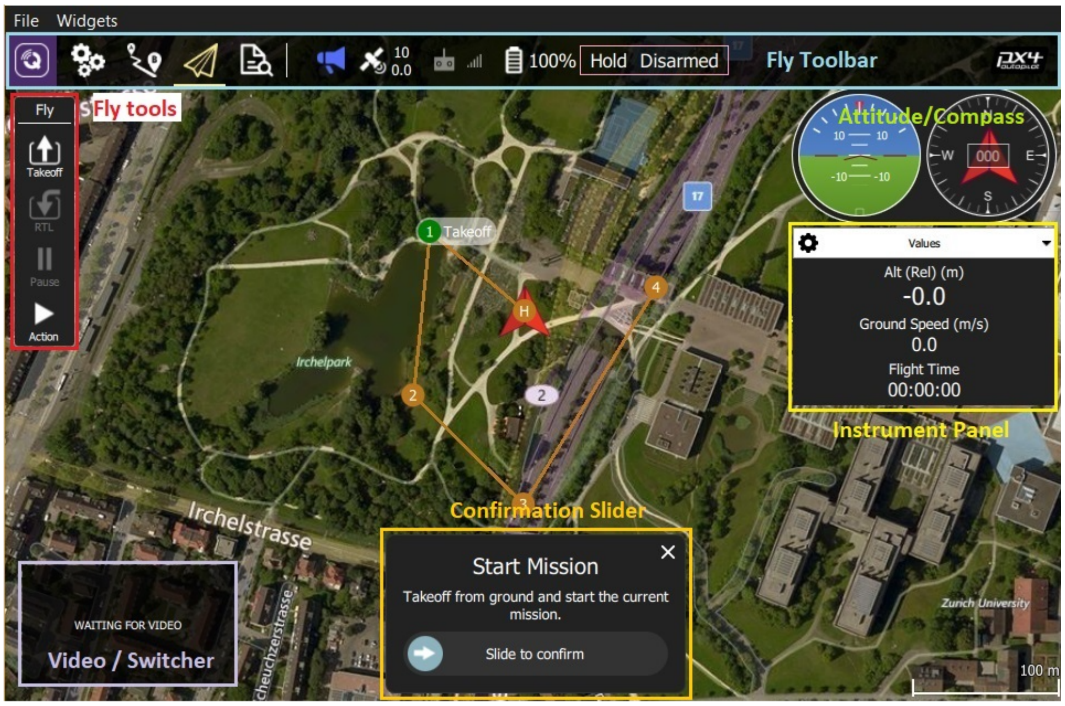
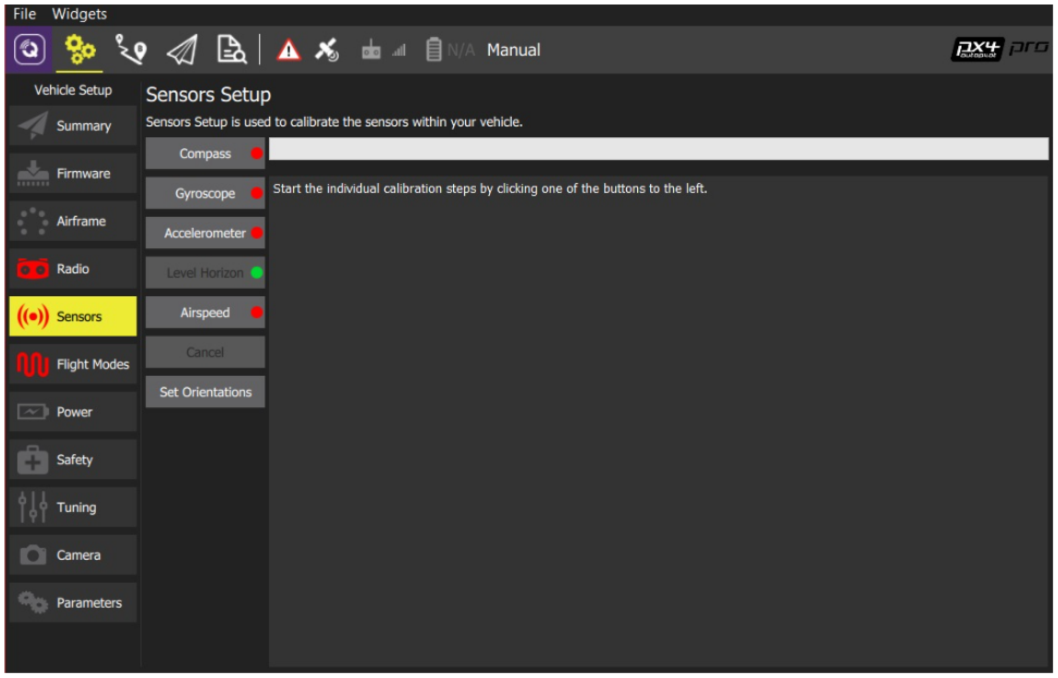


Fig. 2.7. QGroundControl - Ground Control Station[41]



Sensors Setup

Sensors Setup is used to calibrate the sensors within your vehicle.

Start the individual calibration steps by clicking one of the buttons to the left.

Fig. 2.8. QGroundControl - Sensor Calibration[42]

2.7 Controller Area Network

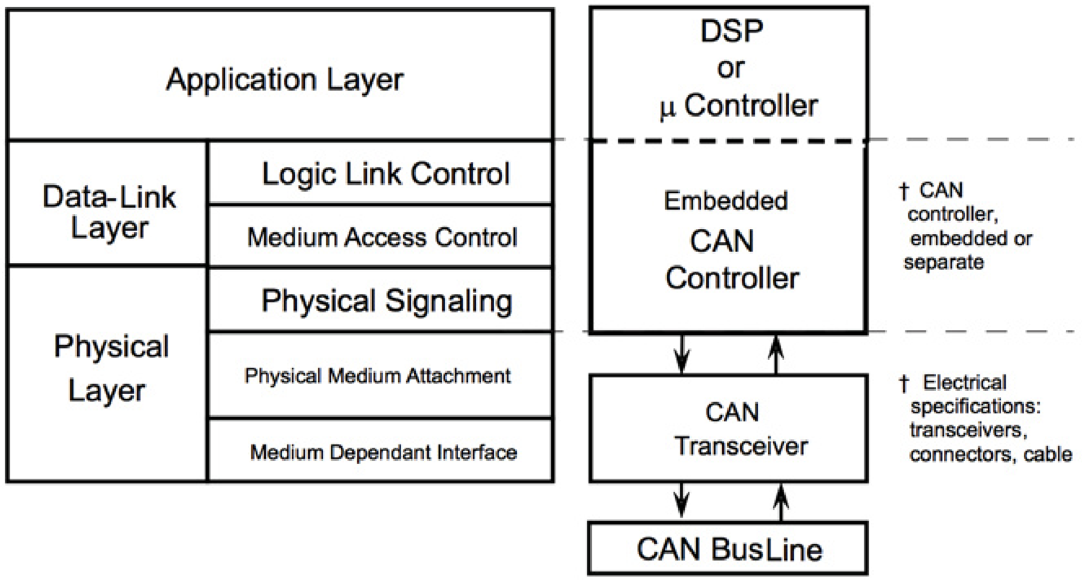


Fig. 2.9. ISO specification For CAN Bus[25]

Robert Bosch GmbH, German engineering, and electronics company established CAN bus standard in the automotive industry[25], to replace complex wiring harness with two wire bus. The CAN bus has immunity to electrical noise and repair data error; this standard gained popularity in various industries ranging from automobile to medical, industrial sectors[26]. CAN bus is a multi-master, message broadcast system that has a maximum signaling rate of about 1 Mbps. The messages/CAN frames broadcasted to the entire network, which provides data consistency in every node of the system. There are two types of the CAN standard:

1. Standard CAN.

2. Extended CAN.

CAN communication protocol is a Carrier-Sense Multiple Access (CSMA) protocol with Collision Detection and Arbitration on Message Priority (CD+AMP). It shows every node in the network should wait for a particular period before transmitting a message. The collisions are resolved by a bit-wise arbitration, based on the priority of individual information/message in the identifier field of the CAN frame. The standard CAN have an 11-bit identifier with a signaling rate of about 125Kbps to 1 Mbps[25]. Later, was introduced extended CAN frame with a 29-bit identifier. The standard CAN provides 2048 unique message identifiers while the extended CAN provide 537 million identifiers. They both are compatible with each other and can work on the same bus.

2.7.1 Can Bus

In a system, logic-high associated with one and logic low associated with zero. But on the CAN bus, it works oppositely. The highest priority is given to the lowest bit of ID. In the bus, the CAN-High and CAN-Low wires should be terminated by the resistance of about 120 ohms.

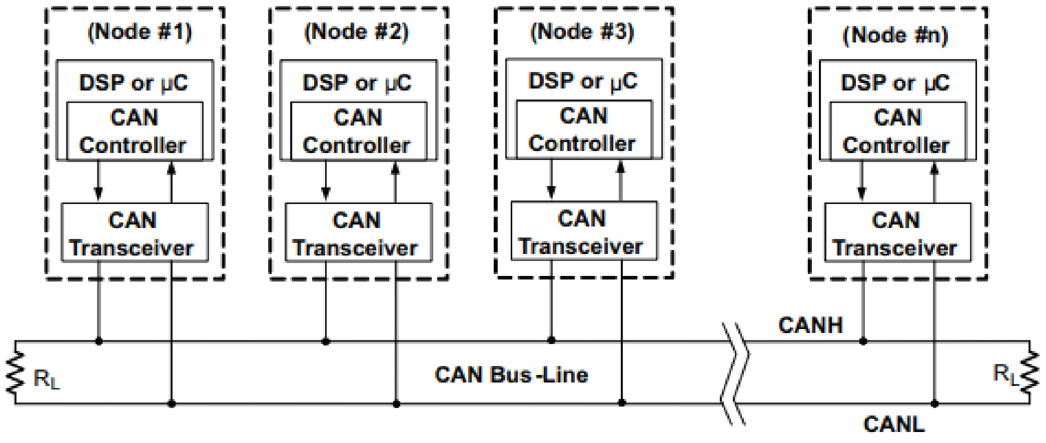


Fig. 2.10. CAN Bus Network Configuration[25]

2.7.2 UAVCAN

Unmanned Aerial Vehicle Controller Area Network (UAVCAN) protocol designed for reliable communication for aerospace and robotic applications over the robust vehicular networks such as CAN bus. The below concepts of UAVCAN are greatly utilized in this research[27]. The background study on the research article[44] aided in better understanding of UAVCAN principles for this research. The design goals of on which UAVCAN is focused on are below:

- Democratic network - The UAVCAN is designed similar to CAN bus where there exist no master node, and all the nodes possess equal communication to ensure no single point of failure in the network.
- Nodes can exchange long payloads - Nodes aid in the exchange of payloads in the bus. When the size of the payload is small, it is sent as a single CAN frame and for large data structures, multiple CAN frames are used in which the decomposition and reassembly of payload occur at the protocol level.
- Support for redundant interfaces and redundant nodes - One of the requirements for safety-concerned applications.
- High throughput, low latency communication - Several applications which are relying on high-frequency, hard real-time control loops require communication which has a high-throughput and a low-latency.
- Simple logic, low computational requirements - UAVCAN are selected for high-performance embedded systems for high-level data processing to devices which are resource constrained.
- Common high-level functions should be clearly defined - UAVCAN standards, messages and services are used for higher level functions such as node firmware update, network discovery, network-wide time synchronization, and the dynamic node ID allocation.

Basic Concepts

In UAVCAN network each node has a unique numeric identifier known as node ID. Two major communication methods are present in UAVCAN:

1. Single Frame transfer - The payload is entirely transmitted in a single CAN frame up-to 8 bytes.

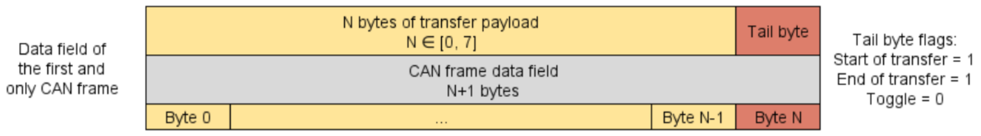


Fig. 2.11. Single Frame Transfer in UAVCAN[27]

2. Multi Frame transfer -The payload is divided and sent using CAN different frame.

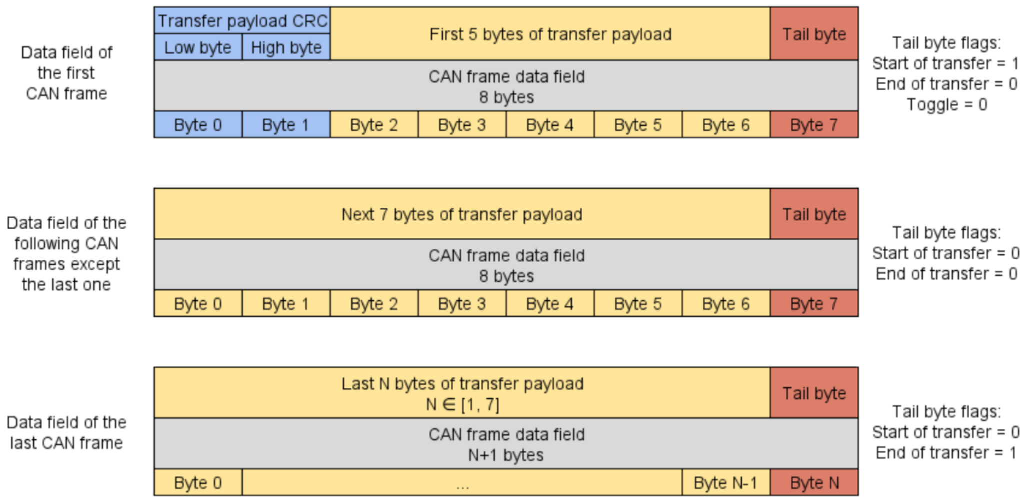


Fig. 2.12. Multi Frame Transfer in UAVCAN[27]

Each mode of communication has a predefined data structure where data structure consists of a unique identifier knows as the data type ID. There are two types of data structures:

1. Data structures defined by protocol specification.
2. Vendor-specific data structures or application specific data structures.

Every published message has a unique data type ID, and each node in the network would possess a unique node ID. A pair of unique node ID and data type ID is helpful to support redundant nodes with same functionality inside the same network. The data structures are defined by data structure description language (DSDL). DSDL is used to specify the data structures which have to be transmitted as a CAN frame. Each DSDL definition file would have a unique identifier followed by the data type name. Vendor-specific data types are also initialized in the system. A new data type name has to is added for each customer specific applications such that it does not affect the existing system functionality. DSDL helps optimize the implementation of the protocol in terms of performance and memory consumption by allowing the compilers to determine the size of data structures statically. This is critical in deeply embedded systems where dynamic memory allocation may be not accepted. Serialized messages and service data structures are exchanged through CAN bus transport layer and which automatically decomposes long transfers of CAN frames enabling the nodes to exchange data structures of arbitrary size[27].

Message Broadcasting

This refers to the serialized transmission of data structure over the CAN bus. This is adopted as one of the primary methods of UAVCAN data exchange. A broadcasted message includes the following:

1. Payload - Data structure to be transmitted or received.
2. Data type ID - Numerical identifier indicating how the data structure should be interpreted.
3. Source node ID - Node ID of transmitting node.
4. Transfer ID - An integer getting incremented with every transfer of this message from a given node.

Nodes which does not have unique node ID can publish anonymous messages.

Service Invocation

This a two-step data exchange between a server and a client.

1. The client initially sends a service request to the server.
2. Server undertakes appropriate actions and transmits a response to the client.

Service invocation has all features of message broadcasting with an addition of another feature.

Client node ID - The source node ID while requesting transfer and destination node ID during the response.

Data Structure Description Language

DSDL is used to define data structures for exchange via the CAN bus. DSDL is automatically generate message serialization/deserialization code for programming language. Every DSDL file specifies one data structure that can be used for message broadcasting or a pair of structures for service invocation method. The source file must follow the below format:

[default data type ID].data type name.uavcan.

The default data type ID is an integer number which would be a part of message transfer ID.

Syntax of DSDL

DSDL consists of attributes and directives. All definition files should contain one attribute definition or at most one directive. An attribute can be:

1. Field - Variable that can be altered by the application and exchanged.

2. Constant - A constant value that does not participate in the network exchange.

A DSDL definition may contain the following entities:

1. Service response marker.

2. Comments.

A DSDL definition for a message data type may contain only the following:

1. Directives (zero or more).

2. Attribute definitions (zero or more).

3. Comments (optional).

Vendor-Specific Data Types

Vendors should define their specific data types in a namespace which should match their company name. A new data type name has to is added for customer specific applications such that it does not affect the existing system functionality. It should follow the naming requirements; the name of the DSDL namespace should start with an alphabetic character. As we understood the data type ID should be an integer value so in-case of vendor-specific data type, the integer number assigned to company must be used to avoid confusion.

CAN Bus Transport Layer

UAVCAN supports tow modes of transfer:

1. Message transfer - A broadcast that contains a serialized message.
2. Service transfer - A unicast transfer that contains either a service response or request.

They can be further distinguished into:

1. Payload - Serialized data structure.
2. Data type ID - Numerical identifier indicating how the data structure should be interpreted.
3. Priority - An integer indicating the priority of the message. The highest priority is given to zero.
4. Transfer ID - An integer getting incremented with every transfer of this message from a given node.

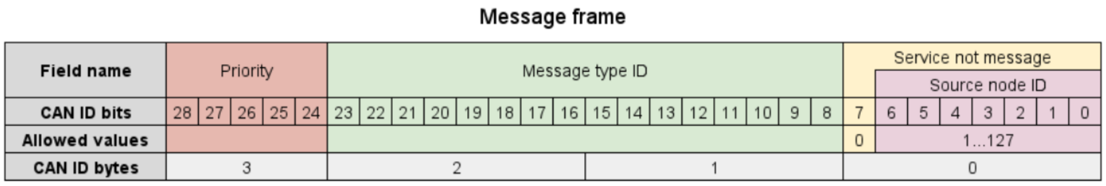
The properties which are common to all types of transfers are:

1. Single Frame transfer - The payload is entirely transmitted in a single CAN frame up-to 8 bytes.
2. Multi Frame transfer -The payload is divided and sent using CAN different frame.

CAN Frame Format

UAVCAN uses only CAN 2.0B frame format (29-bit identifiers). But it can share the same bus with other protocols based on CAN 2.0A (11 bit identifier). In the event of message broadcast, the CAN ID field of every frame of transfer will contain the below fields:

- Priority -5 bits representing the priority of the message.
- Message type ID -16 bits containing the data type ID of the encoded message.
- Service not message -0 for message broadcast and 1 for service invocation.
- Source node ID -7 bits representing the node ID of the source.



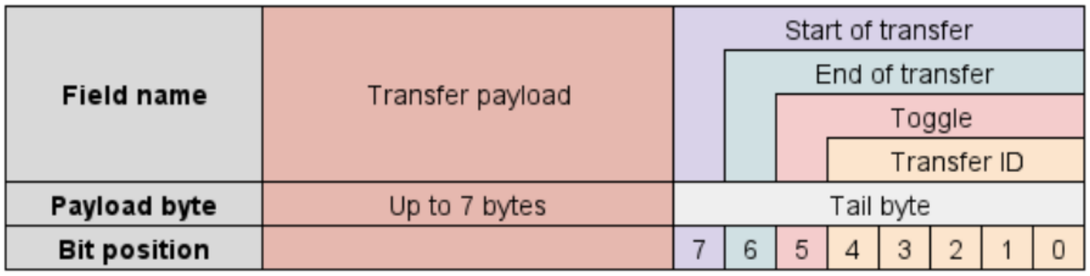
Message frame

Field name	Priority					Message type ID																Service not message	Source node ID						
CAN ID bits	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Allowed values																						0	1...127						
CAN ID bytes	3					2								1								0							

Fig. 2.13. CAN Frame Format[27]

Payload

The payload of the CAN frame is 7 bytes and it is followed by a single tail byte.



Field name	Transfer payload	Start of transfer	End of transfer	Toggle	Transfer ID			
Payload byte	Up to 7 bytes	Tail byte						
Bit position		7	6	5	4	3	2	1	0

Fig. 2.14. CAN payload[27]

The tail byte consist of following field structure:

- Start of transfer - The value of the field is 1 for single frame transfers.
- End of transfer - The value of the field is 1 for single frame transfers.
- Toggle bit - Field value is 0 for single frame transfers.
- Transfer ID - Contains the transfer ID value of the current transfer[27].

2.7.3 FlexCAN

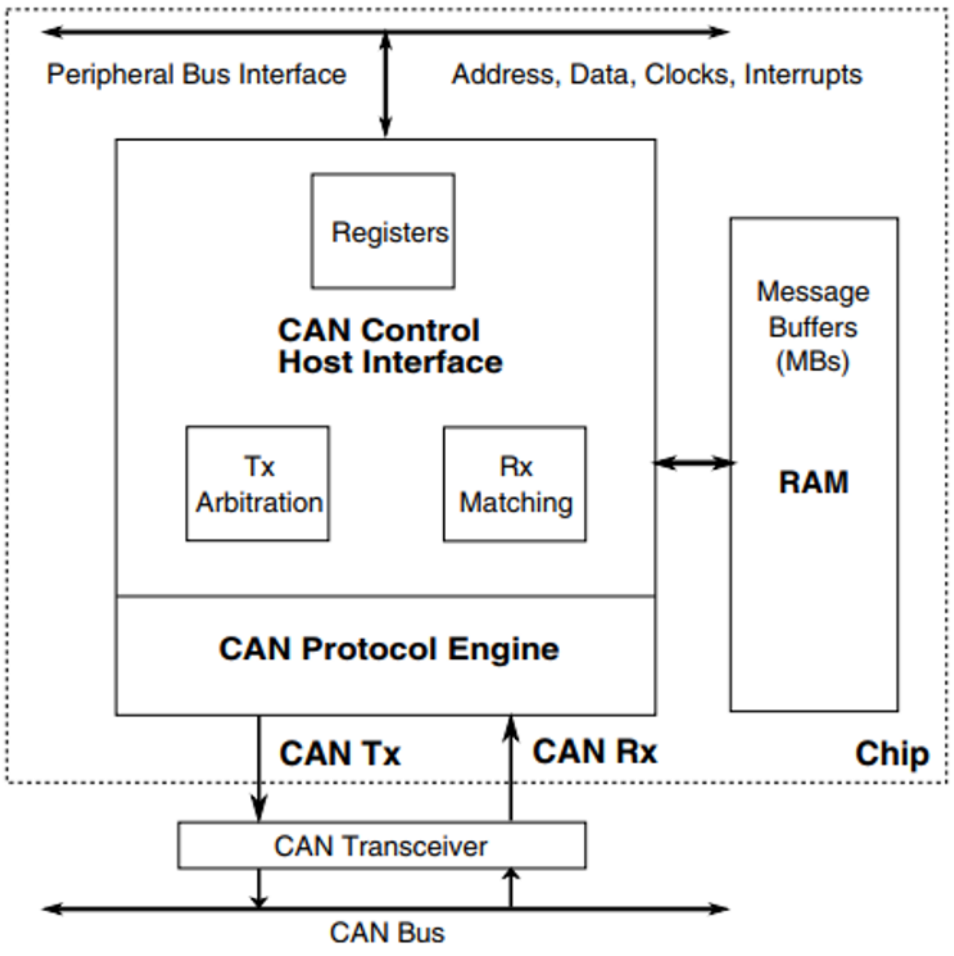


Fig. 2.15. FlexCAN Architecture For Embedded Systems[28]

FlexCAN architecture is designed for highly dependable embedded applications. This architecture is based on CAN protocol and targeted on safety-critical embedded applications such as vehicles like trucks, cars, and boats. FlexCAN is building upon native CAN protocol by addition of a specific protocol termed as SafeCAN to address stringent levels of dependency[28]. FlexCAN contains all the basic CAN features, and in addition to it, it also contains below:

1. Helpful in redundancy management by detecting node errors and manage a set of replicated nodes as well.
2. Sequence numbers.
3. Timed messages.
4. Replicated CAN channel management.
5. Keeps track of various applications.
6. Has features to enable fault and failure management.
7. Enable application fail-safe behavior.

FLexCAN offers node replication and CAN channel replication. In addition to the above features, a mechanism is needed to address various replicated components in the network. This is aided by SafeCAN protocol. FlexCAN provides a simple mechanism that allows end users to write safe applications. The CAN messages are transferred as frames. The number of frames transferred and received depends on the hardware capacity of the processor in which protocol is implemented. In addition to it, the end safe applications should be carefully defined, developed, tested, and verified before deploying it in real-time applications[29].

3. HARDWARE AND SOFTWARE USED

3.1 Rapid IoT Prototyping Kit

Fig. 3.1. Rapid IoT - Prototyping[30]

NXP's rapid IoT prototyping kit is a power optimized IoT end node solution, comprehensive and secure with user friendly development environment features that enables ideas to transform to proof of concept (POC)[30]. Rapid IoT includes pre-programmed applications that enables users to have a quick understanding about the hardware and software architectures. It also provides a web-based IDE and GUI tool which would help researchers to work on applications without help of programming languages. MCU Xpresso IDE which is useful to validate code and develop various applications with the existing architecture[31]. It supports several wireless protocols such as BLE, Thread and ZigBee which would help in data exchange with any Wireless Network. Rapid IoT targets applications like smart cities, smart homes and smart applications which constitute a major portion of IoT field[32].

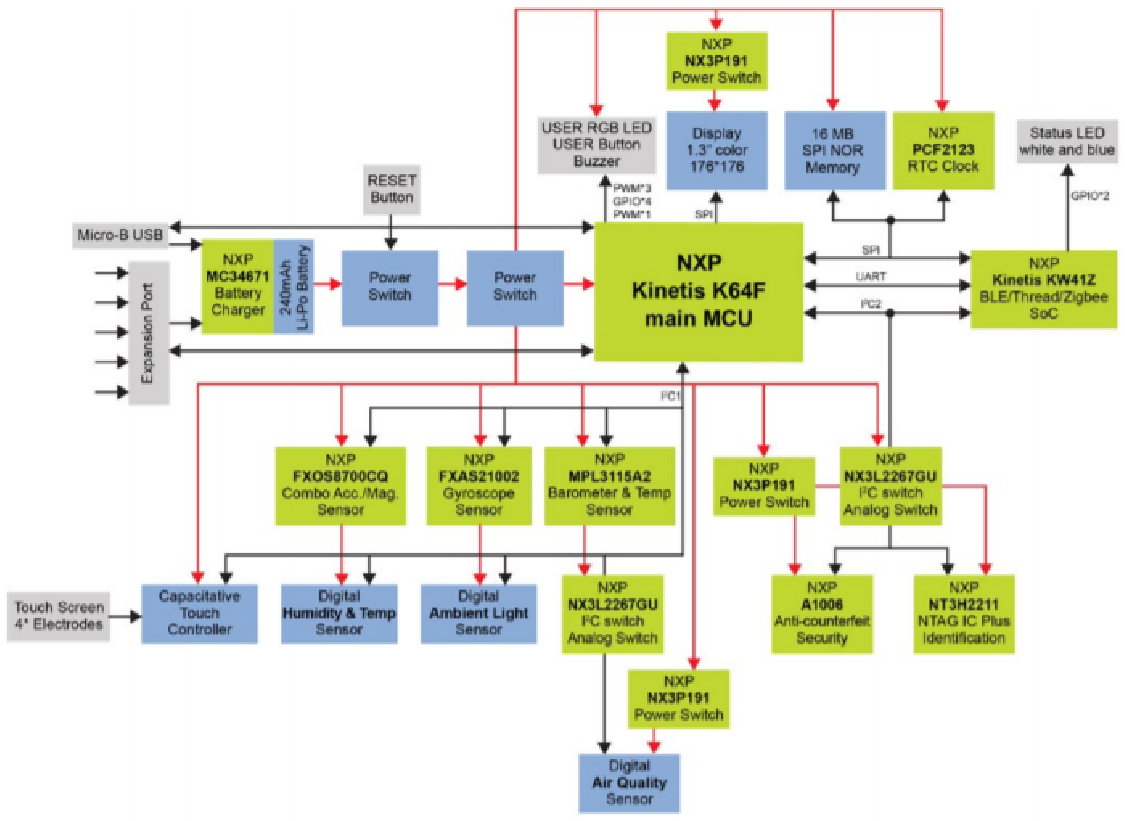


Fig. 3.2. Rapid IoT Architecture[30]

Wireless protocols existing in rapid IoT system helps in integrating rapid IoT with UAVs and help in wireless data transfer applications. Owing to low cost and low weight of the system, it is easy to be integrated with existing applications to provide advanced features.

3.2 Features of Prototyping kit

The key features of rapid IoT prototyping kit are:

1. MCU based on ARM Cortex-M4 Core for application processing.
2. BLE, Thread and ZigBee enables connection to a gateway or a phone.
3. Enhanced security features using A1006 secure authentication and NFC forum type.
4. Software enablement including RTOS drivers, middleware and cloud platforms connectivity.
5. Multiple sensor options like Gyroscope, Accelerometer/Magnetometer used as a part of the system.
6. System is enabled with advanced sensors like digital temperature, humidity, ambient light sensor and digital air quality sensors[30].

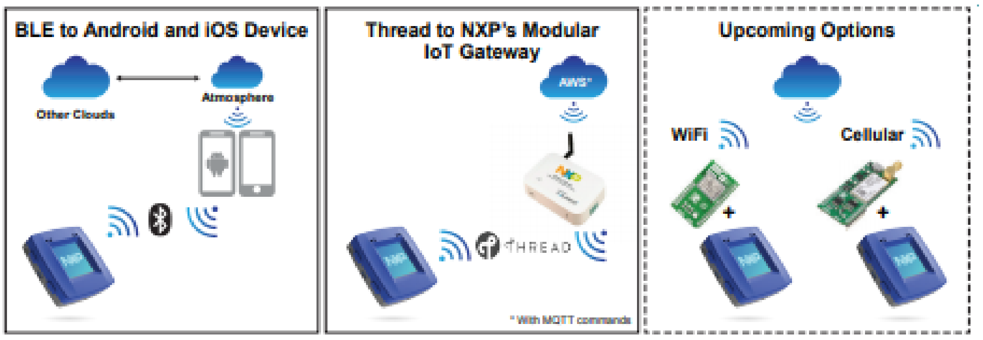


Fig. 3.3. Rapid IoT - System Blocks[30]

3.3 Pixhawk/PX4

Pixhawk is an open-source project developed initially by Swiss Federal Institute of Technology at Zurich for Micro Air Vehicle transportation[35]. Pixhawk is based on 32bit STM32F427 Cortex M7 with 2 MB of memory and 512KB RAM. Further it has inbuilt sensors like ST Micro L3GD20H 16 bit gyroscope, ST Micro LSM303D 14 bit accelerometer / magnetometer,Invensense MPU 6000 3-axis gyroscope/accelerometer, MEAS MS5611 barometer and interfaces like 5x UART (serial ports), 2x CAN, and a Spektrum DSM / DSM2 / DSM-X, RSSI, S-BUS, SPI, I2C, ADC.

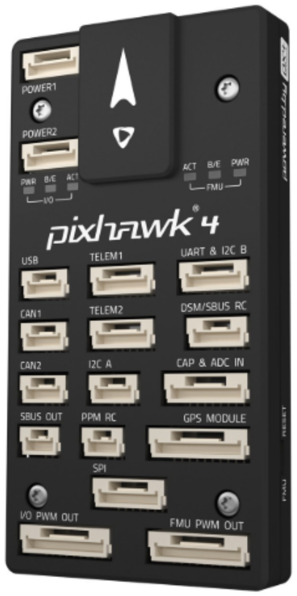


Fig. 3.4. Pixhawk Autopilot Board[35]

3.3.1 PX4 Firmware

PX4 works on a lightweight and an energy efficient real-time operating system called NuttX, which provides a Portable Operating System Interface (POSIX) style environment. NuttX is designed for resource-constrained systems.

It provides a command line interface called NuttShell (nsh) which can be accessed by ground Control station to access the system[33]. PX4 middleware also provides a micro Object Request Broker (uORB) for an asynchronous mode of communication between tasks. uORB can help in data transfer between threads in a simple publish-subscription pattern[34].

3.4 A71CH - Plug and Trust for secure IoT applications

3.4.1 Overview of A71CH

The data acquired by IoT devices is increasing each day. The need to improve the security for the data exists as a common question across the research community[39]. A71ch is a ready-to-use solution that provides root of trust at the IC level and delivers proven, chip to the cloud security out of the box so connection can be established with cloud platforms such as IBM, AWS and google cloud without the need to write security keys or exposing certificates/credentials. The A71ch solution provides basic security measures protecting the IC against many physical and logical attacks[37]. This can be used with various host platforms and host operating systems to establish a secured communication between a broad range of systems. The key benefits of this chip are:

1. Zero Touch Key management
   Generation of keys and credentials are often visualized as complex process and it increases the cost of ownership since they increase the vulnerabilities if not properly done. On other hand, provisioning a large number of devices is a tiring process and would require a lot of man power. In order to eliminate the above difficulties, A71ch secure trust provisioning service is implemented at chip level and it offloads the cost of ownership and complexity of key management from OEMs to have a smooth procedure for IoT development.

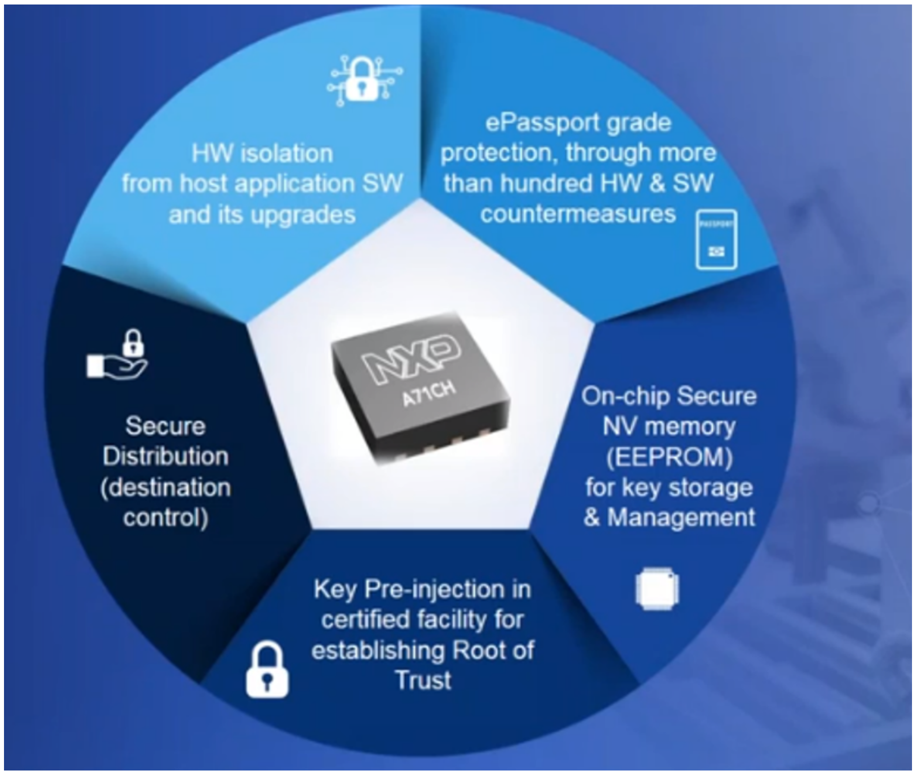

HW isolation from host application SW and its upgrades
ePassport grade protection, through more than hundred HW & SW countermeasures
Secure Distribution (destination control)
NXP
A71CH
On-chip Secure NV memory (EEPROM) for key storage & Management
Key Pre-injection in certified facility for establishing Root of Trust

Fig. 3.5. A71ch - Plug and Trust for Secure IoT

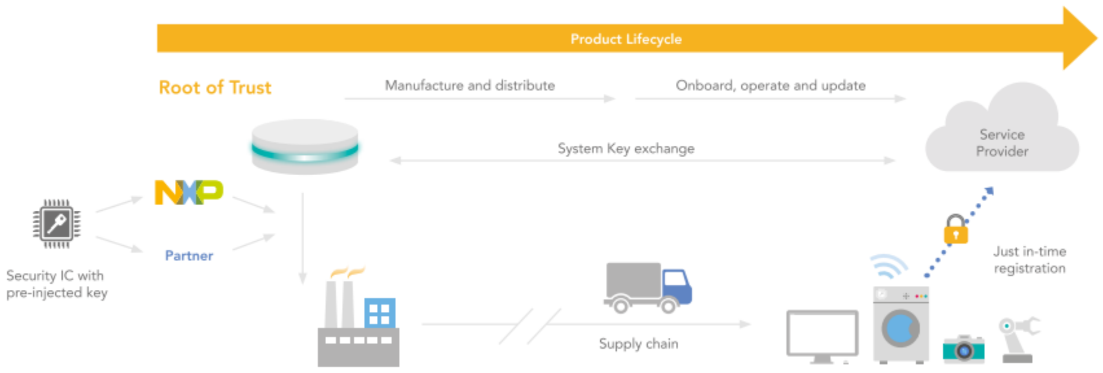

Product Lifecycle
Root of Trust
Manufacture and distribute
Onboard, operate and update
System Key exchange
Service Provider
NXP
Partner
Security IC with pre-injected key
Just in-time registration
Supply chain

Fig. 3.6. A71ch - Root Of Trust

2. Quick Integration

   The design integration is provided with simple connectivity stacks and with sample code to get a better understanding about major use cases. The system is compatible with MX and kinetic MCU[38]. Cloud support with platforms like Amazon, Alibaba aids in providing complex security solutions. This feature would highly reduce the deployment time for secure IoT devices and deliver protection throughout the extended ecosystem.

3.4.2 Features and Application sectors of A71CH

Cryptographic features include:

1. Protected Access storage, generation, insertion or deletion of 4 key pairs.
2. Systematic enforced authentication.
3. Protected Access storage, insertion or deletion of 3 public keys.
4. Signature generation and verification (ECDSA).
5. Shared secret calculation for Key Agreement (ECDH or ECDH-E).
6. A unique chip ID (18 bytes).
7. Freezing of credentials (= OTP behavior).
8. Possibility to lock the A71CH module as transport lock mechanism.
9. HMAC SHA256 calculation in one shot or sequential[36].

The target applications[38] are:

1. Connected industrial devices handling secure and confidential data.
2. Sensor networks gathering medical records.
3. Secure key management.

4. IP cameras for vision processing or object detection.
5. Home gateways.
6. Home appliances.

4. PROPOSED METHOD

4.1 Integrate an RTOS system with UAV

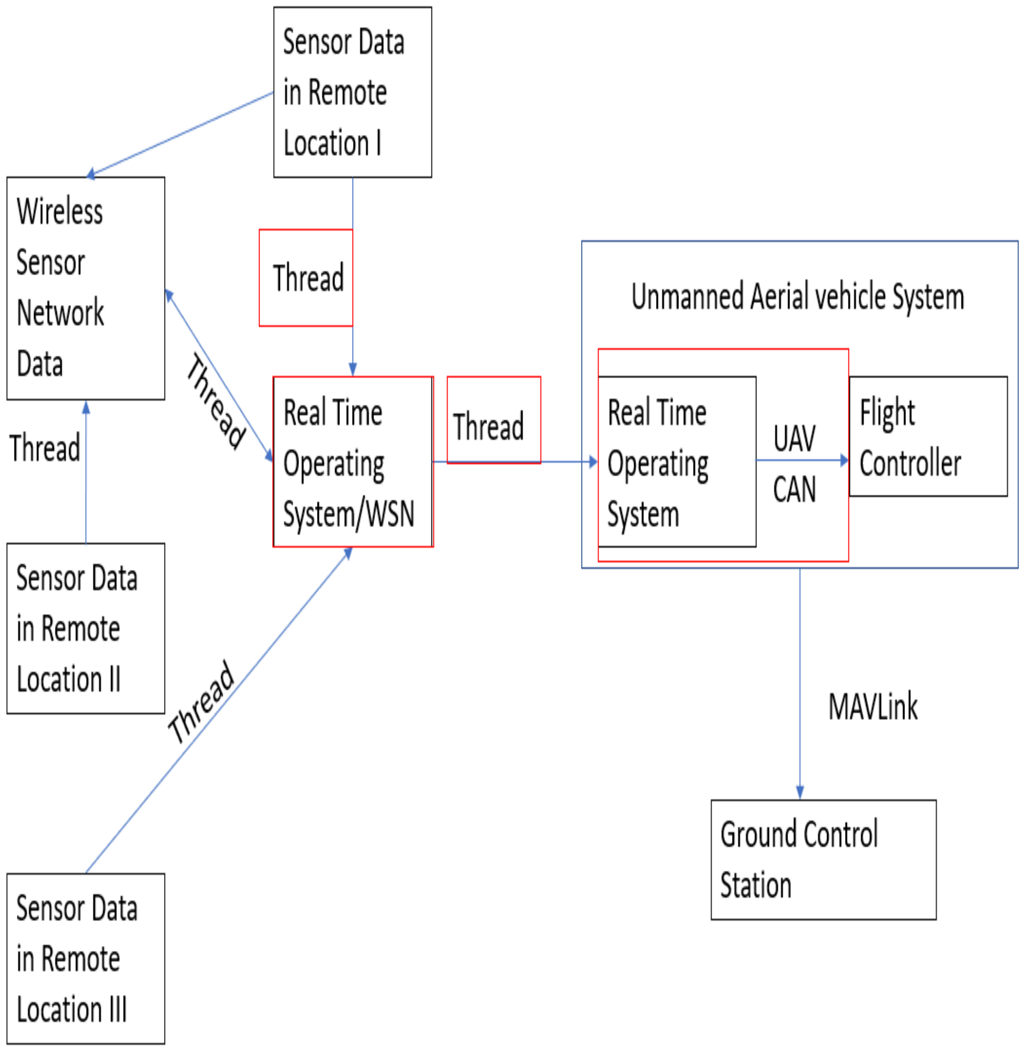

Sensor Data in Remote Location I
Wireless Sensor Network Data
Thread
Thread
Thread
Real Time Operating System/WSN
Thread
Unmanned Aerial vehicle System
Real Time Operating System
UAV CAN
Flight Controller
Sensor Data in Remote Location II
Thread
MAVLink
Ground Control Station
Sensor Data in Remote Location III

Fig. 4.1. Proposed Block Diagram For First System

Almost all physical system devices which are being today are embedded with sensors. These sensors help in the process of data collection like a temperature sensor to sense the temperature of the surrounding environment and a weight sensor to monitor to weight of an object. The information gathered by the sensor are often processed by the processor and are transmitted to other physical devices in the same network either through wired communication like Serial, CAN bus protocols or with the help of various modes of wireless communication and data transfer in IoT devices these days. BLE and Thread network is used as a predominant wireless method of data transfer owing to the low powered architecture and no single point of failure features. Significance of using BLE and Thread in any device is that they are cheap and can transfer data wirelessly up to 300 m without any internet. Having a clear understanding of RTOS system comes handy because nowadays the majority of RTOS is enabled with BLE/Thread protocols. It would provide a crucial advantage during the times of crisis or disaster for data transmission between remote networks because the possibility of getting internet is very narrow. But the data can be exchanged only for a particular range if there is no internet. This remains one of the constraints in hilly or rough terrains where chances of getting internet connectivity is always low. In order to extend the range of data transmission, UAV systems were deployed to collect information from remote wireless networks through wireless data transfer. In this research, a freeRTOS is appended to the UAV system through a specialized CAN protocol called UAVCAN. At the same time, in the remote location all the sensor nodes at different points are connected in same network through Thread network. Once the UAV system reaches any remote destination, the thread network in the rapid IoT would get activated and it would acquire data from other wireless sensor networks in that locality. UAV system can acquire data even from the nodes which are very far away from the drone cause all the nodes are in a single network and it works like a chain reaction. This received data is in turn communicated to flight controller module. Further, the flight controller transmits the information through telemetry, as radio signals to ground station located miles away.

4.2 Establishing a secure data transmission

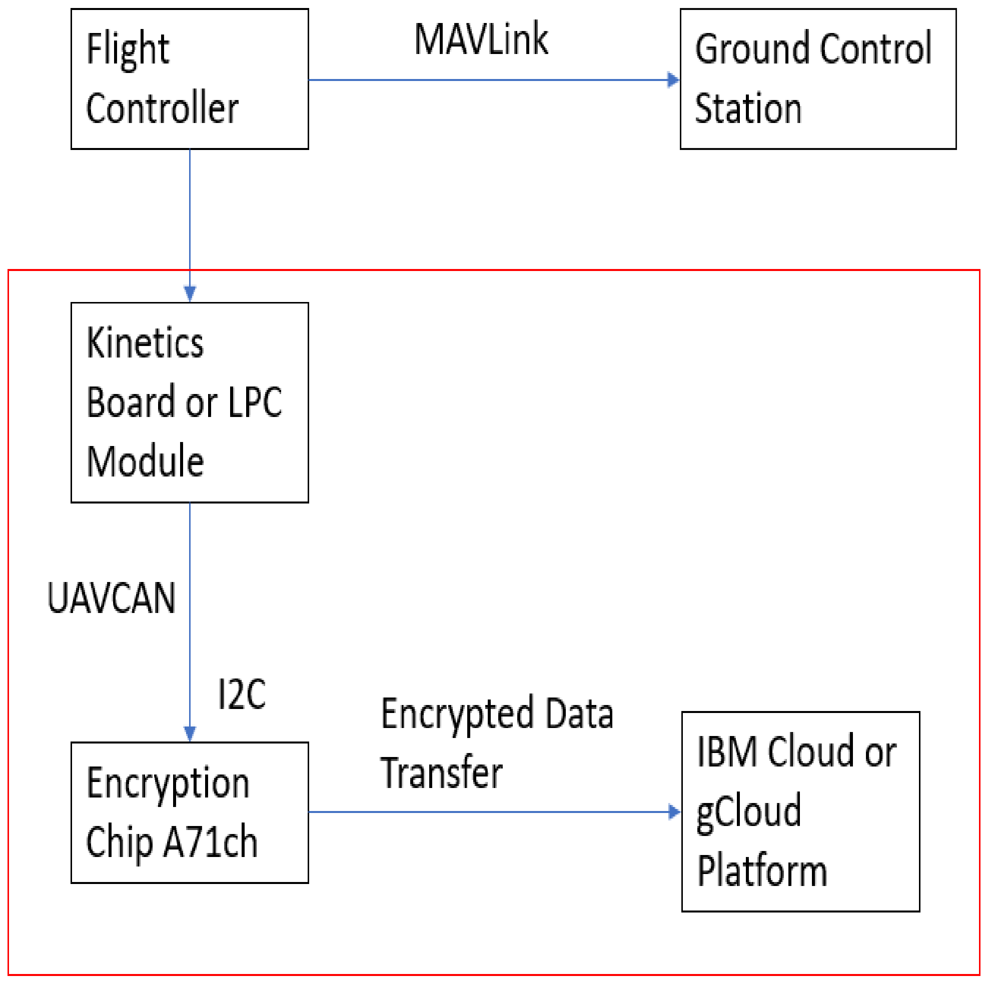


Fig. 4.2. Proposed Block Diagram For Second System

UAV systems are highly helpful in various applications. Owing to less air traffic, UAV are vital in rescue operations since they can reach the locations in a short span of time irrespective of the terrains. For instance, during the time of vehicular accident UAVs can reach the accident location faster than rescue team. They can be used for video surveillance to estimate the impact of the accident. UAVs are also used in collecting sensitive data which range from patients medical data to industrial readings.

The existing system is designed in such a manner that, when drone is performing its task, with the help of a telemetry device these data are transmitted to ground control station using radio transmission by the help of telemetry. The base station and the drone can be miles apart at times but the data would still be transmitted using radio transmission. But if the telemetry port of the UAV system is damaged or if the drone crashes due to bad weather condition, the data transmission would be affected. This could lead to loss of sensitive/essential information. Also, data leakage still remains a threat since the path of data transmission is not completely secure. Potential threats might infect the data.

To address this issue in UAV system, the research uses an encryption chip which supports ECC to encrypt the data and uploading it to cloud platform using public and private keys exchange in the areas which has internet accessibility. By using the cryptography algorithms illegal accessing of data could be highly minimized and only authorized personnel could visualize the data in the cloud transmitted from the UAV system. The cloud platform would serve as a data backup for each instance of data transmitted and would save the data for later uses even if the UAV system has been damaged. Another advantage of cloud platform is that, it provides access to witness the data to any location in the globe.

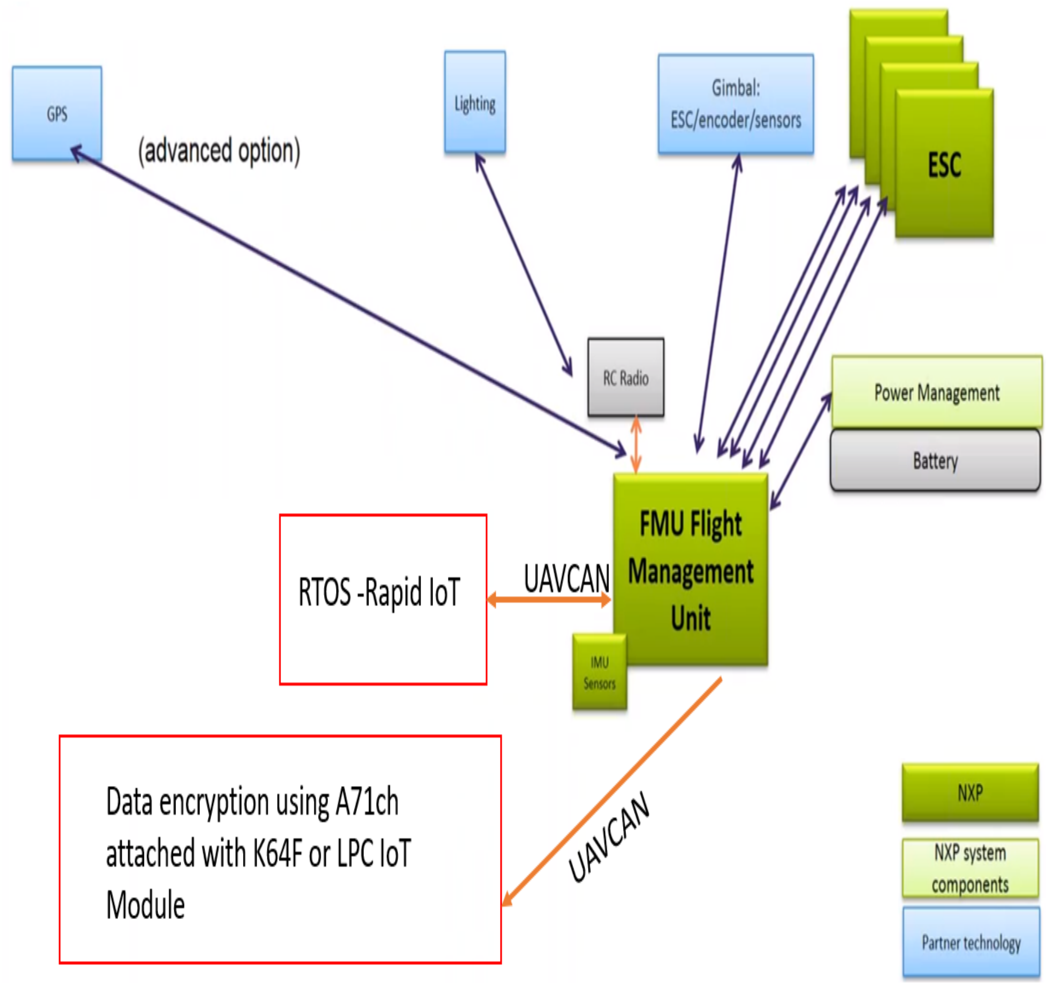


Fig. 4.3. Proposed UAV Design

5. IMPLEMENTATION

5.1 Initial Setup and Configuration

Pixhawk,an autopilot module is updated with latest version PX4 firmware from git repository. Two message files and two c files were created inside px4 firmware to test uORB communication using simple publish-subscribe method. One of the program files was used to send incremental integers every second and other file was used to receive the integers and store them in a variable. This behaviour is witnessed through Nuttx shell. Two new UAVCAN messages were created in px4 firmware. One was used to send data as a CAN frame to the external hardware and another messages was used to receive the incoming the CAN frames. Necessary changes were made in controller files such that it would not affect the existing UAVCAN functionality. To visualize the CAN frames in the flight controller, a UAVCAN GUI tool was used. The bus monitor feature was helpful to witness and decode the CAN frames in the device. Ardupilot software stack can also be used to debug errors in the CAN frames but GUI tool was adopted in this research for testing frames.

To extend the range of data transmission, NXP's Rapid IoT, a freeRTOS system is used as a Thread Radio Module in this research. NXP's FRDM KW41Z is assumed to be the Thread Radio Module on the remote locations. Both these devices are ensured that they both exist on the same network. Rapid IoT being the leader and other being an remote end device which would transmit sensor data.

In order to ensure secure data transmission, NXP's FRDM K64F, a Cortex M4 is used as a debugger for encryption chip, A71ch. A71ch is connected to K64F using I2C communication protocol. Using the USB port as debugger, security credentials are generated for the encryption chip.

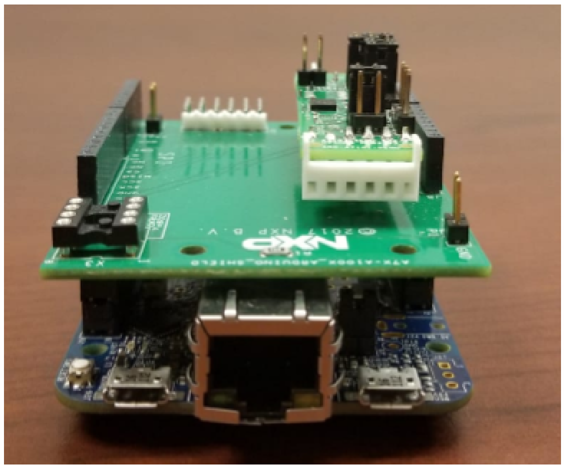


Fig. 5.1. A71ch with K64F

A pair of public and private keys is produced for each device. This remains unique to the encryption chip irrespective of the debugger platform used. IBM cloud/google cloud can be used as a cloud platform to visualize the data which is being sent from encryption chip.

IBM Bluemix was utilized in this approach. It can support up to 500 devices. An user account was created in IBM cloud platform. CA certificates generated for the encryption chip are uploaded to the cloud. The corresponding device ID and the organization name are modified in IBM cloud platform. Multiple devices can exist for a single organization provided each device has unique ID name. These credentials should be match the credentials in the encryption chip to make the data transmission successful. A python script was written to do a initial testing to ensure the data corresponding to the particular device was getting updated in the cloud without the using the hardware. When this approach was successful, the values were getting updated under recent events section in the cloud, the script was modified to fetch back the values from the cloud as well. Both of these scripts were tested in raspberry pi to ensure it was working in an independent system and to obtain full understanding of the cloud section.

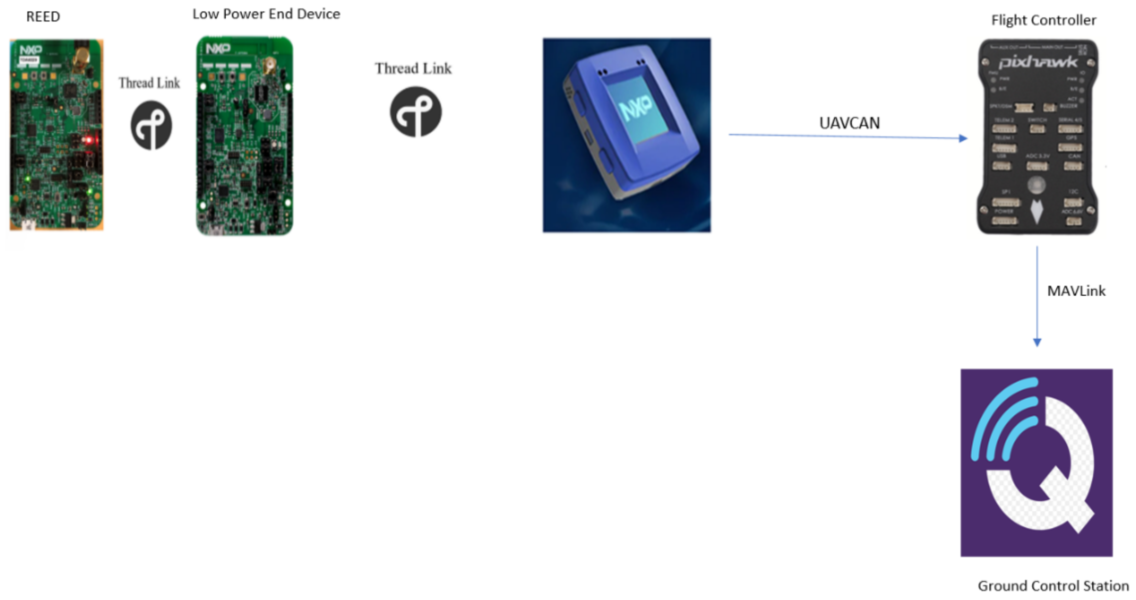


Fig. 5.2. Proposed System Design For First System

5.2 Extending the range of data transmission

5.2.1 System Design

The proposed system comprises of Cortex M7 based flight controller, Pixhawk and Thread radio modules from NXP and Cortex M4 based Rapid IoT which works on freeRTOS. Pixhawk is an autopilot module running PX4 flight stack on NuttX RTOS, which allows to plan autonomous missions to perform sensor data collection at the remote locations. The interfaces in the Pixhawk are widely used to connect with telemetry, GPS module and battery management unit to power up the board. The CAN port was used in this research to establish a serial communication with freeRTOS system. Battery management unit was utilized to supply the hardware with 5V power supply. The 41Z boards are placed at a distance of about 80m connected together with a thread network.

5.2.2 Hardware Setup

The setup of the UAV system is carried out by addition of GPS for navigation of the drone, telemetry to send and receive information with ground control station by radio transmission. GPS and telemetry are connected to the respective connection ports of flight controller. The rapid IoT kit is programmed with the required network credentials mentioned in the initial setup and it is mounted on the UAV system with the help of hexiwear docking station. This docking station provides ports for external interfaces to rapid IoT module. It is also fitted with external CAN transreceiver to convert CAN TX and CAN RX data into CAN High and CAN Low signals. The CAN termination resistor is already existing in the transreceiver. The UAV system is pre-programmed with an autonomous flight path to follow programmed travel directions.

On the other hand, the KW41Z board located on remote location is programmed with the same set of network credentials that include unique network ID and network name. The remote thread module would receive the sensor data from remote wireless sensor networks. Once the UAV system reaches the remote locations such as a hilly area or a disaster location, this thread network in UAV system gets activated would request data from remote thread module within its range. The data packets are transferred wirelessly using UDP. The packets are transferred even from thread nodes which are not present in the immediate range of UAV system. Once the freeRTOS receives the data, it uses UAVCAN communication bus to transmit the data to flight controller which in turns sends the data through telemetry to ground station which can be about 2 miles away. The data can be seen in the ground Control shell terminal as well.

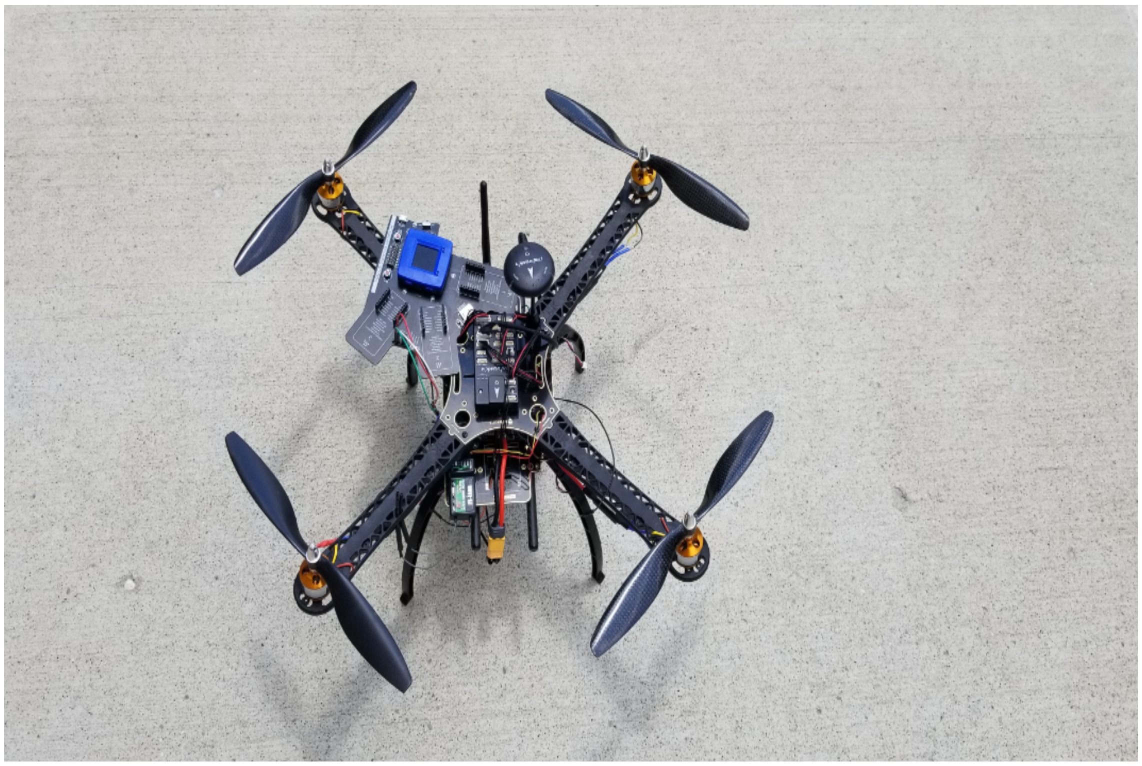


Fig. 5.3. UAV System with Rapid IoT

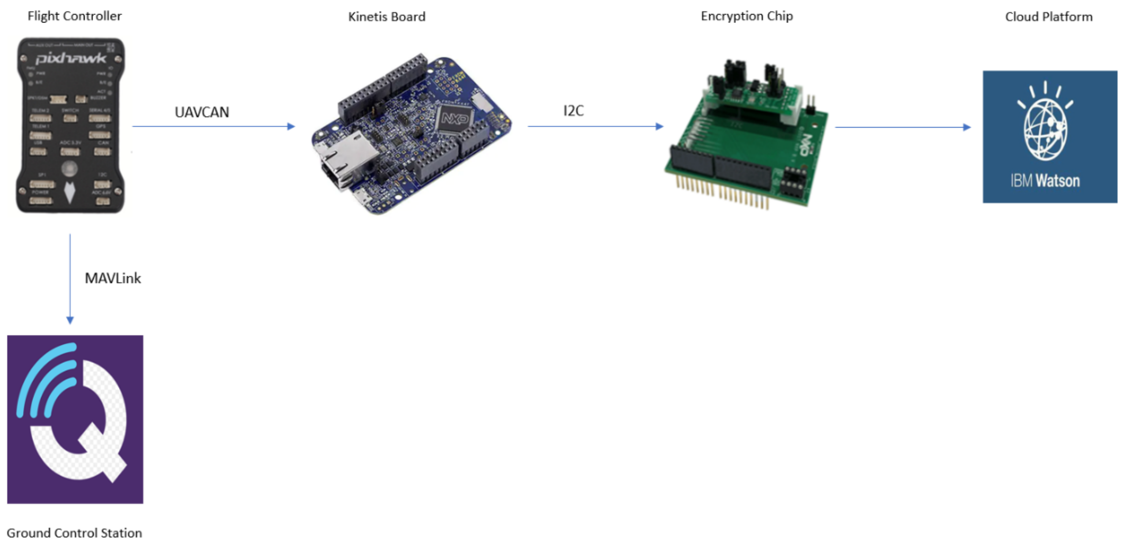


Fig. 5.4. Proposed System Design For Second System

5.3 Secure Data Transmission

5.3.1 System Design

The system design consists of the UAV with Pixhawk as flight controller and A71ch,an encryption chip. The flight controller transmitting data as CAN frames to kinetis board using UAVCAN protocol. Pixhawk was able to transmit up to 1200 frames and each frame of about 8 bytes of data. With the help of external CAN transmitters and CAN port in Pixhawk, the encryption chip is appended to the flight controller. UAVCAN protocol is used to exchange CAN data frames. When the Pixhawk receives any external data, the data is transmitted as CAN frames to the encryption chip. This chip encrypts the data using cryptography algorithms and sends the data to cloud platforms through Wi-Fi. It utilizes the network at that location to transmit to the cloud.

5.3.2 Hardware Setup

In this setup, FRDM k64F board along with A71ch is used as mentioned in above Figure 11.1. With the help of external CAN transreceivers, CAN pins of K64F is connected to CAN communication port of flight controller. Once the UAV system is armed, the commands to send data as CAN frames are sent from the qground control. Telemetry transmits this command to UAV system. The data from the flight controller is transmitted as CAN frame to k64F board. UAVCAN GUI tool is used to ensure all the frames were sent.

K64F on receiver end would receive all the CAN frames transmitted by the flight controller but it filters out the CAN frame based on a unique Extended CAN ID and rest of the IDs are ignored. The data is then encrypted using ECC, and it is transmitted to the IBM cloud platform. Data is witnessed on the IBM cloud platform to ensure no data is lost during the transmission. The information is uploading every second during this transmission. A raspberry pi is also added to the system to subscribe to the data from the IBM cloud. This architecture would make the data available to anywhere in the world with internet connectivity. This approach can be implemented using NXPs LPC 54018 board, which has Wi-Fi connectivity along with A71ch.

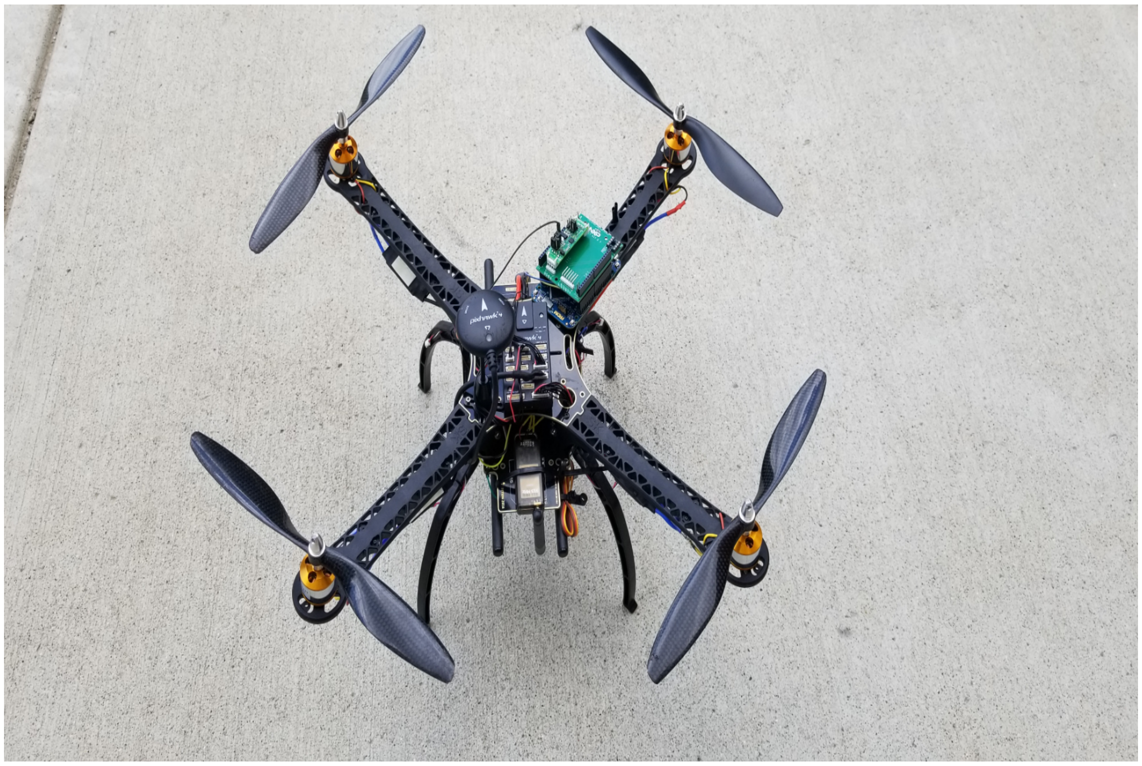
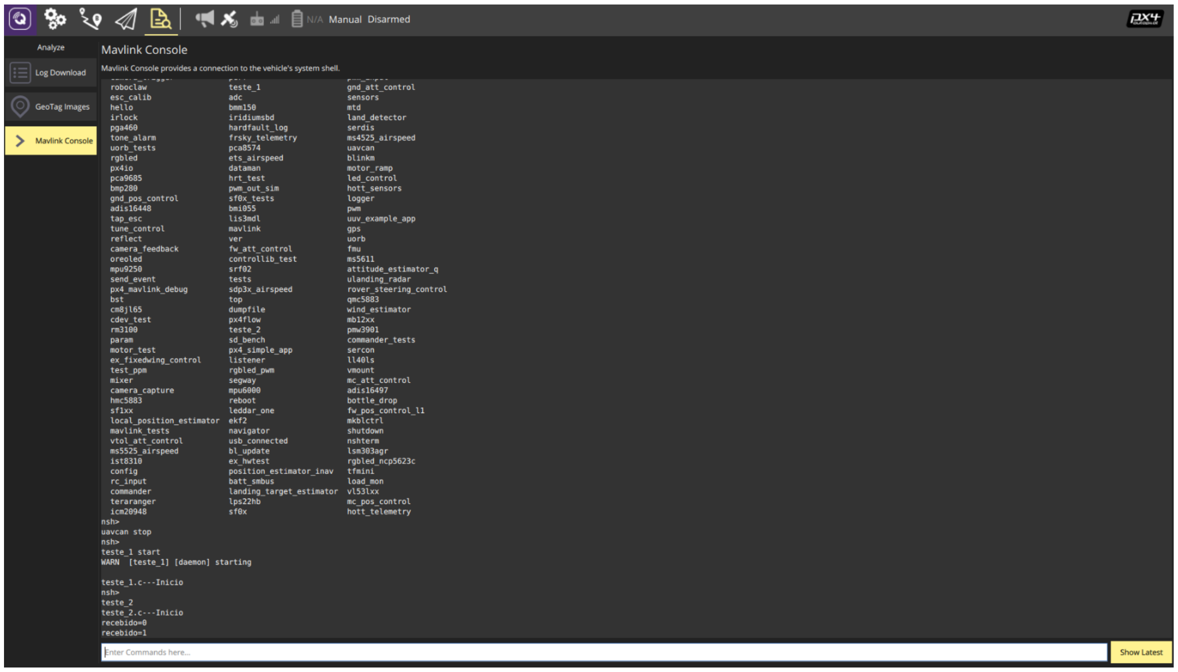


Fig. 5.5. UAV system with Security IC

6. RESULTS AND DEMONSTRATION

Extending the range of data transmission of wireless sensor networks by incorporating thread and UAV design and secured data transmission with the help of latest hardware like NXPhlite, Pixhawk, NXP KW41Z and NXP rapid IOT, NXP Security IC is tested in real-time environment.

A publish-subscribe uORB messaging was tested initially and Figure 12.1 depicts it. The proposed system design One has flight controller which receives the data from rapid IoT system for each second through UAVCAN communication protocol. The status of UAVCAN frames is observed through qGroundControl. Fig 12.1 and 12.2 show the status of UAVCAN and it gets activated by passing UAVCAN the start command through the telemetry.

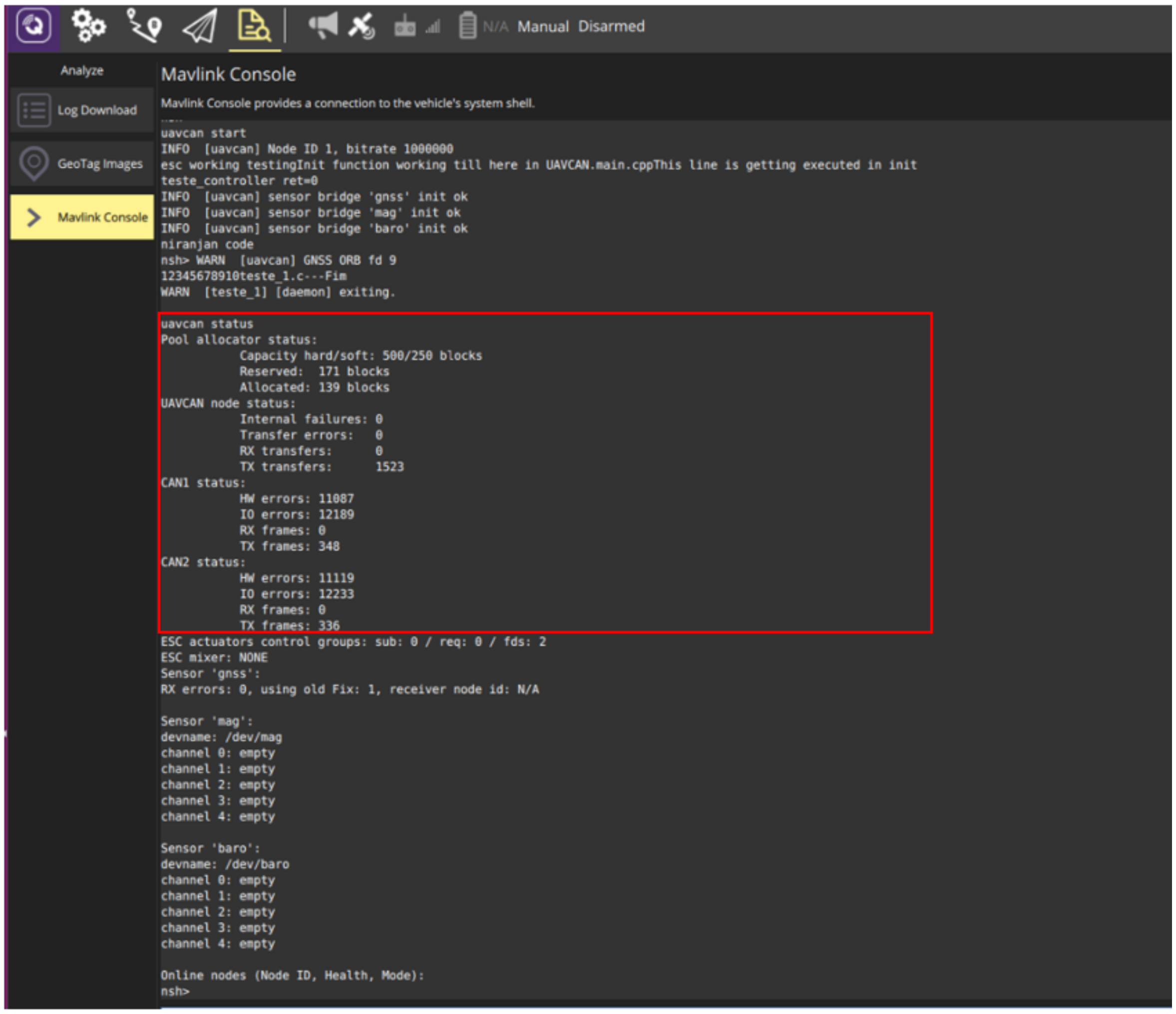


Mavlink Console

Mavlink Console provides a connection to the vehicle's system shell.

```
  roboclaw                 teste_1                  gnd_att_control
  esc_calib                adc                      sensors
  hello                    bmm150                   mtd
  irlock                   iridiumsbd               land_detector
  pga460                   hardfault_log            serdis
  tone_alarm               frsky_telemetry          ms4525_airspeed
  uorb_tests               pca8574                  uavcan
  rgbled                   ets_airspeed             blinkm
  px4io                    dataman                  motor_ramp
  pca9685                  hrt_test                 led_control
  bmp280                   pwm_out_sim              hott_sensors
  gnd_pos_control          sf0x_tests               logger
  adis16448                bmi055                   pwm
  tap_esc                  lis3mdl                  uuv_example_app
  tune_control             mavlink                  gps
  reflect                  ver                      uorb
  camera_feedback          fw_att_control           fmu
  oreoled                  controllib_test          ms5611
  mpu9250                  srf02                    attitude_estimator_q
  send_event               tests                    ulanding_radar
  px4_mavlink_debug        sdp3x_airspeed           rover_steering_control
  bst                      top                      qmc5883
  cm8jl65                  dumpfile                 wind_estimator
  cdev_test                px4flow                  mb12xx
  rm3100                   teste_2                  pmw3901
  param                    sd_bench                 commander_tests
  motor_test               px4_simple_app           sercon
  ex_fixedwing_control     listener                 ll40ls
  test_ppm                 rgbled_pwm               vmount
  mixer                    segway                   mc_att_control
  camera_capture           mpu6000                  adis16497
  hmc5883                  reboot                   bottle_drop
  sf1xx                    leddar_one               fw_pos_control_l1
  local_position_estimator ekf2                     mkblctrl
  mavlink_tests            navigator                shutdown
  vtol_att_control         usb_connected            nshterm
  ms5525_airspeed          bl_update                lsm303agr
  ist8310                  ex_hwtest                rgbled_ncp5623c
  config                   position_estimator_inav  tfmini
  rc_input                 batt_smbus               load_mon
  commander                landing_target_estimator vl53lxx
  teraranger               lps22hb                  mc_pos_control
  icm20948                 sf0x                     hott_telemetry
nsh>
uavcan stop
nsh>
teste_1 start
WARN  [teste_1] [daemon] starting

teste_1.c---Inicio
nsh>
teste_2
teste_2.c---Inicio
recebido=0
recebido=1
```

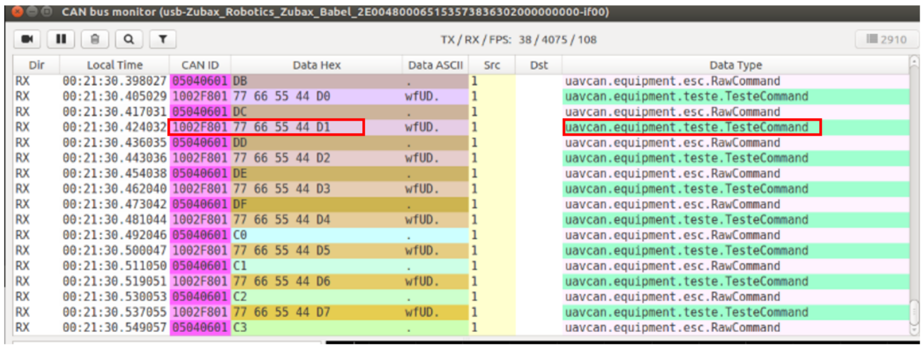
Fig. 6.1. uORB Testing



```
uavcan start
INFO  [uavcan] Node ID 1, bitrate 1000000
esc working testingInit function working till here in UAVCAN.main.cppThis line is getting executed in init
teste_controller ret=0
INFO  [uavcan] sensor bridge 'gnss' init ok
INFO  [uavcan] sensor bridge 'mag' init ok
INFO  [uavcan] sensor bridge 'baro' init ok
niranjan code
nsh> WARN  [uavcan] GNSS ORB fd 9
12345678910teste_1.c---Fim
WARN  [teste_1] [daemon] exiting.

uavcan status
Pool allocator status:
          Capacity hard/soft: 500/250 blocks
          Reserved:  171 blocks
          Allocated: 139 blocks
UAVCAN node status:
          Internal failures: 0
          Transfer errors:   0
          RX transfers:      0
          TX transfers:      1523
CAN1 status:
          HW errors: 11087
          IO errors: 12189
          RX frames: 0
          TX frames: 348
CAN2 status:
          HW errors: 11119
          IO errors: 12233
          RX frames: 0
          TX frames: 336
ESC actuators control groups: sub: 0 / req: 0 / fds: 2
ESC mixer: NONE
Sensor 'gnss':
RX errors: 0, using old Fix: 1, receiver node id: N/A

Sensor 'mag':
devname: /dev/mag
channel 0: empty
channel 1: empty
channel 2: empty
channel 3: empty
channel 4: empty

Sensor 'baro':
devname: /dev/baro
channel 0: empty
channel 1: empty
channel 2: empty
channel 3: empty
channel 4: empty

Online nodes (Node ID, Health, Mode):
nsh>
```

Fig. 6.2. QGroundControl - UAVCAN Frame Status

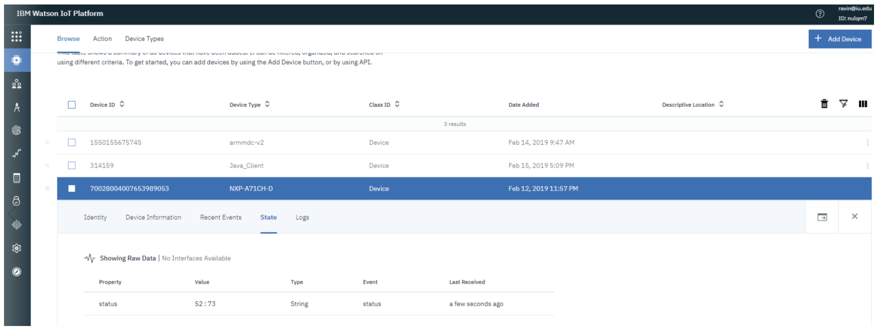
The data bytes is which being transmitted as UAVCAN frames are decoded by using UAVCAN GUI tool and the Figure 12.3 displays that. Anonymous node ID were chosen for the flight controller so the Source Node ID in GUI tool is one. The Data Hex displays the bytes of data that is being sent and received. The data type would display the incoming and outgoing data types and timestamp records transmissions during each instance. Figure 12.4 depicts the proposed system design two.


CAN bus monitor (usb-Zubax_Robotics_Zubax_Babel_2E0048000651535738363020000000000-if00)
TX / RX / FPS: 38 / 4075 / 108
2910

```
Dir  Local Time       CAN ID    Data Hex          Data ASCII  Src  Dst  Data Type
RX   00:21:30.398027  05040601  DB                .           1         uavcan.equipment.esc.RawCommand
RX   00:21:30.405029  1002F801  77 66 55 44 D0    wfUD.       1         uavcan.equipment.teste.TesteCommand
RX   00:21:30.417031  05040601  DC                .           1         uavcan.equipment.esc.RawCommand
RX   00:21:30.424032  1002F801  77 66 55 44 D1    wfUD.       1         uavcan.equipment.teste.TesteCommand
RX   00:21:30.436035  05040601  DD                .           1         uavcan.equipment.esc.RawCommand
RX   00:21:30.443036  1002F801  77 66 55 44 D2    wfUD.       1         uavcan.equipment.teste.TesteCommand
RX   00:21:30.454038  05040601  DE                .           1         uavcan.equipment.esc.RawCommand
RX   00:21:30.462040  1002F801  77 66 55 44 D3    wfUD.       1         uavcan.equipment.teste.TesteCommand
RX   00:21:30.473042  05040601  DF                .           1         uavcan.equipment.esc.RawCommand
RX   00:21:30.481044  1002F801  77 66 55 44 D4    wfUD.       1         uavcan.equipment.teste.TesteCommand
RX   00:21:30.492046  05040601  C0                .           1         uavcan.equipment.esc.RawCommand
RX   00:21:30.500047  1002F801  77 66 55 44 D5    wfUD.       1         uavcan.equipment.teste.TesteCommand
RX   00:21:30.511050  05040601  C1                .           1         uavcan.equipment.esc.RawCommand
RX   00:21:30.519051  1002F801  77 66 55 44 D6    wfUD.       1         uavcan.equipment.teste.TesteCommand
RX   00:21:30.530053  05040601  C2                .           1         uavcan.equipment.esc.RawCommand
RX   00:21:30.537055  1002F801  77 66 55 44 D7    wfUD.       1         uavcan.equipment.teste.TesteCommand
RX   00:21:30.549057  05040601  C3                .           1         uavcan.equipment.esc.RawCommand
```

Fig. 6.3. CAN Frames - UAVCAN GUI Tool

The data from flight controller is transmitted to security IC and it shows the data in IBM cloud platform during every second of the transaction.



using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Device ID	Device Type	Class ID	Date Added	Descriptive Location
3 results				
1550155675745	armmdc-v2	Device	Feb 14, 2019 9:47 AM	
314159	Java_Client	Device	Feb 15, 2019 5:09 PM	
70028004007653989053	NXP-A71CH-D	Device	Feb 12, 2019 11:57 PM	

Showing Raw Data | No Interfaces Available

Property	Value	Type	Event	Last Received
status	52 : 73	String	status	a few seconds ago

Fig. 6.4. IBM Bluemix - Output Data Visualization

7. CONCLUSION

Technological growth and developments help mankind in different ways. It makes world a better place to live-in for all of us. But internet plays a major role in all these developments. In developed countries, having internet connectivity is a part of every-day's life and people depend on it almost for all activities in a day like using GPS for navigation to reach office using less traffic path and using internet even to order their favourite food. But it is not the same in developing or underdeveloped countries. There are places in the world, where getting a network signal in the place they live happens once in a while. Such places, not all technological advancements are helpful.

The ideal aim of this research is to use UAV systems to aid in development and provide immediate help to places where internet/network coverage not available. To make this idea come true, wireless sensor networks are deployed in areas of interest. UAV are capable of reaching the remote locations where it is difficult for humans to reach. By incorporating thread protocol, a low power architecture to transfer data wirelessly several advantages are brought into UAV system. On the contrary, having a data transmission alone would not be good enough when confidential details like medical records of a patients in terms of OTA (Over The Air) file transfer or any credentials of a private sector. The security IC aids in encryption of data especially when a secure data transmission is involved with UAV system.

An ideal advantage of this research is establishing both the applications using UAVCAN protocol. The CAN bus reduces the number of wiring. The future of UAV system is projected to have CAN bus for all sorts of communication with external hardware platforms and so by integrating the flight controller with freeRTOS system using CAN bus will act as a base for further research applications and ideas.

8. FUTURE SCOPE

This research lays the basic foundation for the new architecture of UAV systems. The future scope of this approach would be incorporating vision processing for machine learning and deep learning algorithms. Integration of Alexa, can be helpful to control the drone system using voice activation comments.

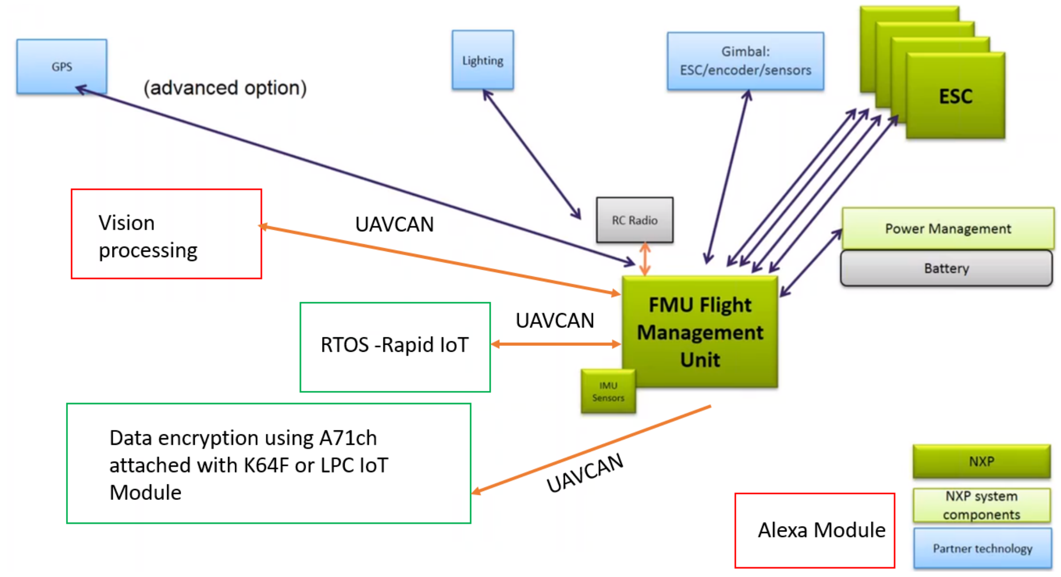


Fig. 8.1. Future UAV Design

REFERENCES

REFERENCES

[1] S. Kumar, and M. Sethi. "State-of-the-Art and Challenges for the Internet of Things Security", (Last Accessed: April 24, 2019). [Online] Available: https://tools. ietf. org/html/draft-irtf-t2trg-iot-seccons-05 (2018).

[2] L. Mainetti, L. Patrono, and A. Vilei. "Evolution of wireless sensor networks towards the internet of things: A survey", in SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks, September 2011, pp. 16.

[3] C. Rotariu, R. G. Bozomitu, V. Cehan, A. Pasarica, and H. Costin, "A wireless sensor network for remote monitoring of bioimpedance", in 2015 38th International Spring Seminar on Electronics Technology (ISSE), May 2015, pp. 487490.

[4] R. Niranjan and M. El-Sharkawy. "Collision avoidance and Drone surveillance using Thread protocol in V2V and V2I communications" (July 2019), pending publication.

[5] P. Kinney. "IEEE 802.15. 4 Information Element for the IETF. No. RFC 8137", 2017.

[6] W. Stallings. "IPv6: the new Internet protocol", IEEE Communications Magazine, 34(7), pp. 96-108.

[7] Thread, "Thread Fundamentals", (Last Accessed: September 5, 2018). [Online] Available: https://www.silabs.com/documents/public/user-guides/ug103-11- appdevfundamentals-thread.pdf

[8] C. W. Group, "Constrained application protocol (CoAP) draft-ietfcore-coap-18", (Last Accessed: October 22, 2018). [Online]. Available: https://tools.ietf.org/html/draft-ietf-core-coap-18

[9] R. Niranjan and M. El-Sharkawy, "Applications of Drones using Wireless Sensor Networks" (July 2019), pending publication.

[10] V. Sivateja and M. El-Sharkawy, "Remote Wireless Sensor Network Range Extension using UAVs with Thread Protocol", CSCI18 5th Annual Conference on Computational Science Computational Intelligence, Las Vegas, 2018.

[11] Thread, "Security Commissioning", (Last Accessed: August 3, 2018). [Online] Available: https://www.threadgroup.org/supportWhitepapers

[12] Thread, "Battery Friendly Design", (Last Accessed: March 7, 2019). [Online] Available: https://www.threadgroup.org/What-is-Thread

[13] Thread, "Commercial Network Topology", (Last Accessed: March 7, 2019). [Online] Available: https://www.threadgroup.org

[14] RTOS, ”What is an RTOS”, (Last Accessed: March 23, 2019). [Online] Available: https://www.highintegritysystems.com/rtos/what-is-an-rtos/

[15] RTOS, ”What is an FreeRTOS”, (Last Accessed: March 7, 2019). [Online] Available: https://www.freertos.org/about-RTOS.html

[16] RTOS, ”Introduction to RTOS”, (Last Accessed: February 14, 2019). [Online] Available: https://www.ni.com/en-us/innovations/white-papers/07/what-is-a-real-time-operating-system–rtos–.html

[17] Y. G. Hong. ”Problem statement of IoT integrated with edge computing”, (Last Accessed: April 24, 2019). [Online] Available: https://tools.ietf.org/id/draft-hong-iot-edge-computing-01.html

[18] B. Stiller and C. Schmitt. ”DTLS-based Security with two-way Two-way authentication for IoT”, (Last Accessed: April 24, 2019). [Online] Available: https://tools.ietf.org/html/draft-schmitt-two-way-authentication-for-iot-00

[19] NXP’s A71ch Plug and Trust -The fast, easy way to deploy secure IoT connections hardware overview, (Last Accessed: February 14, 2019). [Online] Available: https://www.nxp.com/docs/en/application-note/AN12135.pdf

[20] M. Farhan, ”Efficient data communication in unmanned aerial vehicles”, Thesis.35, United Arab Emirates University, 2015.

[21] C. Tonoy, I. L. Rewzana, and G. Atul, ”Evolution of flight control system flyby-light control system”, in International Journal of Emerging Technology and Advanced Engineering (IJETAE), vol. 3, 2013.

[22] Hackaday, ”Droning on: Choosing a flight controller”, (Last Accessed: April 29, 2019). [Online] Available: https://hackaday.com/2014/06/06/droningon- ight-controller-round-up/

[23] Pixhawk, ”What is pixhawk” (Last Accessed: April 29, 2019). [Online] Available: http://pixhawk.org/

[24] D. Gagne, ”Overview of qgc” (Last Accessed: April 29, 2019). [Online] Available: https://docs.qgroundcontrol.com/en/

[25] CAN, ”Introduction to Controller Area Network”, (Last Accessed: April 29, 2019). [Online] Available: http://www.ti.com/lit/an/sloa101b/sloa101b.pdf

[26] CAN, ”A simple Intro”, (Last Accessed: April 29, 2019). [Online] Available: https://www.csselectronics.com/screen/page/simple-intro-to-can-bus

[27] UAVCAN, ”Complete background on UAVCAN”, (Last Accessed: April 29, 2019). [Online] Available:https://uavcan.org/

[28] FlexCAN, ”FlexCAN: A Flexible Architecture for Highly Dependable Embedded Applications” (Last Accessed: April 29, 2019). [Online] Available: https://paws.kettering.edu/ jpimente/flexcan/FlexCAN-architecture.pdf

[29] FlexCAN, ”FlexCAN-K21 Sub-Family Reference Manual” (Last Accessed: April 29, 2019). [Online] Available: https://www.nxp.com/docs/en/reference-manual/K21P144M120SF5RM.pdf

[30] Rapid IoT, "Rapid IoT Prototyping kit: SLN-RPK-NODE", (Last Accessed: April 29, 2019). [Online] Available: https://www.nxp.com/docs/en/fact-sheet/IOTPROTOKITFS.pdf

[31] Rapid Iot, "SLN-RPK-NODE-SW-REL-NOTES-Rapid IoT", (Last Accessed: May 20, 2019). [Online] Available: https://www.nxp.com/support/developer-resources/rapid-prototyping/nxp-rapid-iot-prototyping-kit:IOT-PROTOTYPING

[32] Smarter Infrastructure, (Last Accessed: April 29, 2019). [Online] Available: https://www.nxp.com/docs/en/white-paper/SMRTRINFRASTRWP.pdf

[33] Dronecode, "PX4 architectural overview", (Last Accessed: April 29, 2019). [Online] Available: https://dev.px4.io/en/concept/architecture.html

[34] uORB, "uORB Messaging " (Last Accessed: April 29, 2019). [Online] Available: https://dev.px4.io/en/middleware/uorb.html

[35] L. Meier et al.,"PIXHAWK: A system for autonomous flight using onboard computer vision", Robotics and Automation (ICRA), 2011 IEEE International Conference on, 2011, pp. 2992-2997.

[36] NXP's A71ch Plug and Trust, "APDU Specification", (Last Accessed: April 24, 2019). [Online] Available: https://www.nxp.com/products/identification-security/authentication/plug-and-trust-the-fast-easy-way-to-deploy-secure-iot-connections:A71CH

[37] NXP's A71ch Plug and Trust, "A71CH Host software package documentation", (Last Accessed: April 24, 2019). [Online] Available: https://www.nxp.com/docs/en/application-note/AN12133.pdf

[38] NXP, "NXP IoT Security IC For Cloud Connections", (Last Accessed: April 29, 2019). [Online] Available: https://www.nxp.com/docs/en/fact-sheet/A71CH-LEAFLET.pdf

[39] A New Kind of IoT Security, (Last Accessed: April 28, 2019). [Online] Available: https://www.nxp.com/docs/en/brochure/A71CH-IOT.pdf

[40] L. Zimmermann, N. Mars, M. Schappacher, and A. Sikora, Development of thread-compatible open source stack, Journal of Physics: Conference Series, vol. 870, no. 1, p. 012001, 2017, (Last Accessed: November 29, 2018). [Online]. Available: http://stacks.iop.org/1742-6596/870/i=1/a=012001

[41] QGroundControl, "QGroundControl User Guide", (Last Accessed: April 24, 2019). [Online] Available: https://docs.qgroundcontrol.com/en/

[42] QGroundControl, "Sensor Calibration", (Last Accessed: April 3, 2019). [Online] Available: https://docs.qgroundcontrol.com/en/SetupView/Sensors.html

[43] IoT, "Overview Of Internet of Things", (Last Accessed: February 21, 2019). [Online] Available: https://www.alibabacloud.com/

[44] UAVCAN, "Background Study", (Last Accessed: June 5, 2019). [Online] Available: https://fenix.tecnico.ulisboa.pt/downloadFile/563345090414487/Tese

[45] B. Mller, "Elliptic curve cryptography (ECC) cipher suites for transport layer security (TLS)", (Last Accessed: April 24, 2019). [Online] Available: https://tools.ietf.org/html/rfc4492.