Long-range and Secure Communication System for Remote Data Logging and Monitoring of Micro-grids

By

Amjad Iqbal

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Engineering

Faculty of Engineering and Applied Sciences
Memorial University

May, 2019

St. John's, Newfoundland and Labrador, Canada

Abstract

With the increasing renewable energy penetration in the grid system, the number of distributed generation and remote micro-grids are also increasing. While each remote micro-grid requires to have a sophisticated (secure, long range, low power and low cost) communication system for the Suspervisory Control and Data Acquision (SCADA) system, creating a research gap for devising a system with desired attributes. This research was conducted to fill that gap by implementing a low power, low cost, secure, and a long-range communication system for remote micro-grids. The literature review comprising the study of twelve wireless technologies and three wired technologies was done to achieve this. After the comparison of those technologies, LoRa communication was chosen for this purpose. Different encryption algorithms were studied, implemented, cross-checked, and finally advanced encryption algorithm was implemented to achieve the security of the communication system. An algorithm was developed to generate a unique message authentication code for each message. It enabled to identify a bit-level alteration in the message. Further, to monitor the system remotely, data was uploaded to the server. It was achieved by programming and configuring different gateways for this purpose. LoRa range was improved by implementing LoRa nodes in a mesh-network structure. A hybrid, LoRa and radio-set based system was implemented to extend its range up to 40km. Three different system topologies were designed and implemented, and the final one was recommended based upon the SG SCADA system requirements. The results of step-by-step designing and implementation of the system show that this research work has significantly contributed to develope a low-cost, secure and long-range communication system for remote micro-grids.

Acknowledgement

The author would like to express profound gratitude to his supervisor Prof. Dr M. Tariq Iqbal for his supervision and encouragement throughout the research program. It was possible only due to his guidance, technical and moral support.

The author is also thankful to all other faculty members and students of Memorial University of Newfoundland who played their role to enable the author to achieve this.

The author would also like to thank the Natural Sciences and Engineering Research Council (NSERC) of Canada for providing graduate student funding for this research.

Table of Contents

1 Chapter # 1 Introduction and Literature Review 1

1.1 Introduction 1

1.1.1 Security 2

1.1.2 Low Power Consumption 2

1.1.3 Confidentiality 3

1.1.4 Low Cost 3

1.1.5 Authorisation 3

1.1.6 Authentication 4

1.1.7 Scalability 4

1.2 Literature Review 4

1.2.1 Wireless Communication Technologies 4

1.2.2 Cellular 5

1.2.3 Zigbee 6

1.2.4 WiFi WLAN Series IEEE 802.11 6

1.2.5 Bluetooth 7

1.2.6 Radio Teletype 8

1.2.7 VHF/UHF 8

1.2.8 Optical Wireless Communication 9

1.2.9 Satellite Communication 10

1.2.10 WiMAX Series IEEE 802.16 11

1.2.11 SigFox 11

1.2.12 LoRa 12

1.3 Wired Communication 13

1.3.1 Power line carrier 14

1.3.2 Optical Fibers 16

1.3.3 DSL Based Wired Communication 17

1.4 General Analysis of different technologies 18

1.5 System Security and Encryption 21

1.5.1 Generating Artificial Noise Signal 21

1.5.2 Physical Layer Method 22

1.5.3 Cryptography 22
1.6 Research Goals and Objectives……...23
1.6.1 Secure Communication 23
1.6.2 Low Power 24
1.6.3 Regular Updates 24
1.6.4 Local and Remote Data Logging 24
1.6.5 Implementation 24
1.7 Thesis Overview……..25
2 Chapter # 2 Encryption Algorithms to Secure Communication System and Their Implementation on DRF1276G LoRa Module 27
2.1 Introduction…….27
2.1.1 Privacy 28
2.1.2 Message Authentication 28
2.1.3 Integrity 28
2.1.4 Nonrepudiation 28
2.2 Hardware Setup……...30
2.3 Cryptographic Algorithms on Arduino with DRF1276G LoRa Module……34
2.3.1 Shift Cipher 34
2.3.2 Affine Cipher 36
2.3.3 Transposition Cipher 38
2.3.4 Hill Cipher 38
2.3.5 Substitution Cipher 39
2.4 Breaching the Encryption Algorithms…….39
2.5 Conclusion……...43
3 Chapter # 3 Implementation of AES and MAC to Secure Communication System for Remote Micro-grids 45
3.1 Introduction…….45
3.2 Implementation of AES Algorithm using ESP32 and LoRa Module…….47
3.2.1 Adding the round key and generating a new key 49
3.2.2 Substitute Bytes 50
3.2.3 Shift Rows 50
3.2.4 Mix Column 51
3.3 Results…….52

3.4 Conclusion……..59
4 Chapter # 4 Data Logging Using Different Gateways and Mesh-Network Implementation to Improve LoRa Range……...60
4.1 Introduction……60
4.2 Local Data Logging……61
4.3 ESP-32 vs Dragino Gateways……62
4.4 Implementation of Mesh Network and Range Testing……69
4.5 Complete System Flow Diagram……73
4.6 Conclusion……...75
5 Chapter # 5 Radio-set Based System Topologies for Longer Range Setup 76
5.1 Introduction……76
5.2 System Structure-I……...77
5.3 Hardware Setup and Interfacing……..79
5.3.1 RA30H1317M..... 79
5.3.2 DC Power Supply LRS-150-12..... 80
5.3.3 Building MAX 232 circuit 81
5.1.1 Setting USR-TCP232..... 83
5.3.5 Ethernet and USR-TCP232 setting 85
5.4 System Structure-II……..87
5.5 System Structure-III……89
6 Chapter # 6 Summary and Recommendations.....93
6.1 Summary of Work……...93
6.2 Future Work Recommendations……..95
6.3 List of Publications……..95
6.3.1 Submitted 95
6.3.2 Published/Accepted..... 95
6.3.3 Poster Presentation..... 96
7 References.....97
8 Appendix A *C++ code to Breach the Hill Cipher* 104
9 Appendix B *Arduino code for LoRa duplex communication*.....109
10 Appendix C *Flexible Arduino-code for all level nodes of mesh-network to display/encrypt/decrypt/forward message* 111
11 Appendix D *Configuring SD card and logging data locally*.....145

12 Appendix E *Send email under abnormal conditions* 153
13 Appendix F *LoRa Simple server for Dragino-yun using RF95* 156
14 Appendix G *Dragino-Yun as a Gateway with partial_decryption* 159
15 Appendix H *ESP_Collector Dragino Side* 164
16 Appendix I *Configuring ESP32 as LoRa Gateway* 166

List of Tables

Table 1-1 Comparison of Data Rate for Different Channel Length in DSL [48] 18
Table 1-2 Comparison of different communication technologies [5],[6],[8],[9],[49] 19
Table 2-1 Alphabets with assigned numbers for mathematical calculations in encryption algorithms 37
Table 5-1 Specifications of the power source 80

List of Figures

Figure 2-1 Eavesdropper Masquerading the SCADA Network 30
Figure 2-2 System block diagram with LoRa based wireless communication 31
Figure 2-3 Arduni-uno with DRF1276G Lora module 32
Figure 2-4 Pin diagram of arduino-uno 33
Figure 2-5 Pin diagram of arduino uno with DRF1276G LoRa module 33
Figure 2-6 Occurrence probability of alphabets in any sentence 35
Figure 2-7 Highest to the lowest occurrence probability of alphabets in any sentence 36
Figure 2-8 Encrypted message sent from RED side Lora module 42
Figure 2-9 Two-way communication between two LoRa modules 43
Figure 3-1 Flow chart of the implemented communication process 46
Figure 3-2 Step by step 10 round AES encryption and decryption [72] 48
Figure 3-3 Addition of round key for AES encryption [72] 49
Figure 3-4 Generating new key from the previous round key and data string [72] 50
Figure 3-5 Rotation of bytes in shift row operation [72] 51
Figure 3-6 Mix column operation to increase the confusion and non-linearity [72] 52
Figure 3-7 Implementation of AES encryption using DRF1276G with LoRa module 54
Figure 3-8 ESP32 with LoRa module used for successful implementation of AES encryption 54
Figure 3-9 Implementation of AES encryption using ESP32 with LoRa module 55
Figure 3-10 A hint about large numbers to understand the size of keyspace having 2^128 elements 55
Figure 3-11 Implementation of AES algorithm with MAC address 55
Figure 3-12 Time lapsed in message receiving, verifying and decrypting under different SF 57
Figure 3-13 Time lapsed for message encrypting and sending under different SF 58
Figure 3-14 Difference of data rate and message authenticity under different SF 58
Figure 3-15 Bi-directional communication range testing of ESP32 with LoRa module 59
Figure 4-1 System structure for remote data-logging 61
Figure 4-2 Configuring SD card for local-data logging 62

Figure 4-3 Setting ESP32 as a gateway 64
Figure 4-4 Setting ESP32 as a gateway 64
Figure 4-5 Configuration results of ESP32 as a gateway 65
Figure 4-6 ESP32 gateway sending messages with AES encryption..... 65
Figure 4-7 ESP32 gateway uploading data to the Things Network without encryption..... 66
Figure 4-8 A dedicated dragino-yun LoRa gateway..... 67
Figure 4-9 Successful configuration results of dragino-yun working as a gateway 67
Figure 4-10 Dragino gateway real-time data load..... 68
Figure 4-11 Uploading data to the ThingSpeak server 68
Figure 4-12 ESP32 LoRa range testing..... 69
Figure 4-13 Process flow chart for an intermediate level node..... 70
Figure 4-14 Mesh network for improved LoRa range 72
Figure 4-15 Range testing in obstacle dense area after implementing a mesh network 72
Figure 4-16 Complete System diagram 74
Figure 5-1 System structure for topology I with broader range due to radio-set..... 78
Figure 5-2 Data logging by implementing system structure I..... 78
Figure 5-3 RA30H1317M based radio-set board used in this project..... 79
Figure 5-4 12V DC power supply to power-up the radio-set and protecting against surges 81
Figure 5-5 MAX232 driver circuit built to protect the controller from power surges and to provide common ground for signals..... 82
Figure 5-6 USR-TCP-232 used in the project for serial to ethernet conversion..... 84
Figure 5-7 USR-TCP-232 Serial port setting..... 84
Figure 5-8 Internal registration setting of USR-TCP-232..... 85
Figure 5-9 PC ethernet setting to interface with USR-TCP-232..... 86
Figure 5-10 Results and setting of USR-TCP232 test software..... 87
Figure 5-11 System structure for topology II with broader range due to radio-set..... 88
Figure 5-12 System structure for topology III with broader range due to radio-set 90
Figure 5-13 Dragino-yun low memory error 90
Figure 5-14 Data logging on the ThingSpeak server using system topology III 91

List of abbreviations

SG	Smart Grid
SCADA	Supervisory Control and Data Acquisition
LoRa	Long Range
RED	Remote End Device
EM	Electromagnetic
AMI	Automatic Metering Infrastructure
DG	Distributed Generation
CDMA	Code Division Multiple Access
TDMA	Time Division Multiple Access
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
RSS	Received Signal Strength
VHF	Very High Frequency
UHF	Ultra High Frequency
AES	Advanced Encryption Standard
MAC	Message Authentication Code
GSM	Global System Module
LSB	Least Significant Bit
MSB	Most Significant Bit
SF	Spreading Factor
SD	Secure Digital

Chapter # 1 Introduction and Literature Review

1.1 Introduction

Different techniques and setups which are being used for communication between Remote End Devices (RED) and Supervisory Control and Data Acquisition (SCADA) unit, can be classified in two main classes based on the medium of communication; one is wired, and other is wireless. Further, each class has many sub-categories based on the technique implemented for communication. Every technique has many limitations and scopes associated with it based upon the communication medium, power consumption factor, cost factor, security of information, reliability, authenticity, availability, strength against electromagnetic (EM) interference and many other such factors [1].

The basic requirements of Smart Grid (SG) for adopting any specific communication system are categorised based upon the secrecy of the data communicated [2]. The communication may be required between Automatic Metering Infrastructure (AMI) and distribution grid, between distribution and transmission network, between transmission section and SCADA System (Central Control Unit) or is between the set of virtual generation plants and SCADA System. In this thesis, it is intended to build a secure two-

way communication between Distributed Generation (DG) units and local data collector, and then from the local data collector to the remote SCADA System. For this purpose, different communication techniques have been discussed here and compared in Table 1.2 at the end of the chapter. The selection and implementation of communication setup have been made based upon below given features. Due to the critical nature of control messages and their importance in SG network, a communication system with the following features is required [3-5]

1. Security
2. Low Power Consumption
3. Confidentiality
4. Low Cost
5. Authorisation
6. Authentication
7. Scalability

1.1.1 Security

A secured communication and data storage is required for utility companies to ensure the controlled grid operations [5-6]. Grid control and critical information need to be protected through encryption to resist against hackers and cyber attacks.. In the coming chapters, it has been discussed in the details and how it has been achieved.

1.1.2 Low Power Consumption

In DG and SG, SCADA System communicates with every RED to ensure system health, voltage profile, current value, active power and reactive power along with the other miscellaneous messages. In this way, the communication system remains energised all the

times, and it must give optimal performance over the cost of minimum power consumption. That is why efficiency (power consumption) is among the top deciding factors [3].

1.1.3 Confidentiality

Confidentiality is also among the prime deciding factors for the selection of a communication system. By losing confidentiality of information a third person may get access to edit and can corrupt the data leaving the whole system prone to hackers. Therefore, a message signal must remain confidential between the sender and the receiver [2-3].

1.1.4 Low Cost

In the selection and the implementation of the communication system for the SCADA system and SG, capital/installation cost, running and maintenance cost should also be nominal. Therefore, along with the optimal performance, the communication system project is to be cost-effective [1,3].

1.1.5 Authorisation

After the precise selection of the communication system setup, it is necessary to authorise only concerned personnel to access it and process the messages. The unauthorised usage may give access to any eavesdropper to hack the system or infest any kind of interference [5-6]. Therefore, despite having encrypted data, the channel connection must be protected by passwords/IDs and only concerned users should have the privilege to access the data.

1.1.6 Authentication

Identification of the user account is also important to ensure whether the user is a registered one in the system or any unauthorised user is trying to access the system information. For which system should have a secured database of registered users [1,4].

1.1.7 Scalability

With the growing trend of Smart Grid and expanding integration of DG units in the system, up-gradation and scalability are very common. Therefore, a communication network must have the capability to accommodate incoming DG units and grids ensuring the security and self-configuration [5].

1.2 Literature Review

To find and implement an optimal mean of communication which meets previously stipulated requirements, critical analysis and comparison of all techniques is necessary. To do that in this literature review almost all sub-categories of wired and wireless technologies which have been either proposed or implemented in different publications/projects have been discussed, analysed and compared.

1.2.1 Wireless Communication Technologies

Both classes of communication, either wired or wireless technology have their own scopes and circle of applications. Here is the analysis of wireless technologies, and wired technology will be discussed in the next section.

1.2.2 Cellular

Global System for Mobile Communication (GSM) is a wireless technology. It is the widely used digital technology as compared to Code Division Multiple Access (CDMA), and it uses Time Division Multiple Access (TDMA) topology for digital communication. Currently available 2G, 3G, 4G and WiMAX which are its evolutionary forms are also considerable options for communication purpose in smart grid network especially for communication between RED and SCADA [7]. It will neither require any more capital investment in infrastructure nor installation time. The most important thing is, its accessibility in remote rural areas without additional infrastructure cost. This technology is in practice in few utility companies, e.g. Echelon's Energy Services Network has implemented this project in smart meter and is using GSM and T-Mobile to collect data accumulated in every 15 minutes interval [8]. Besides this several major cellular companies like Silver Spring Network (SSN), Verizon and Cisco are also implementing smart meters with embedded WiMAX chip for SG [9]. Some utilities are using smart meters with integrated GPRS module for communication purpose and in some SG projects Universal Mobile Telecommunication System (UMTS), Code Division Multiple Access (CDMA) and Wideband Code Division Multiple Access (WCDMA) are also being practised [9] but, they all have the common issue of the unavailability of the CDMA/GPRS signal [6]. The primary concern with cellular technology is of privacy loss. Using a cellular network, a third party gets access to the data and the probability of privacy breaching increases. An eavesdropper for the cellular network can also masquerade the system.

1.2.3 Zigbee

Zigbee is another wireless technology being used these days. It is comparatively cheaper in overall installation and simpler with lesser power consumption. But, at the same time, it has a shorter range (10-100m) and low data rate. Zigbee selection has been proposed in [10] and is very feasible for Automatic Meter Reading (AMR), home automation and for home appliances monitoring and control. National Institute of Standards and Technology (NIST) has approved Zigbee Smart Energy Monitoring (SEM) and Zigbee as the best option for the SG private network [11]. Zigbee technology has introduced three devices for its set which includes Zigbee remote device, Zigbee coordinator, and Zigbee router. Zigbee router works as a router and can also operate directly controlling the appliance just like Zigbee remote device controls. Multiple remote devices are controlled and interlinked through the Zigbee coordinator bridge which ensures authenticity and security code for each appliance. Smart meters integrated with Zigbee can control and communicate with the integrated devices in the range of 100m consuming only 1mW power for the radio signal transmission [9-10]. Zigbee is using 16 channels in the 2.4GHz band, and an unlicensed spectrum, which makes it low cost, simple and suitable for the SG network. But it cannot be used for the SCADA system of remote micro-grids only due to very short range.

1.2.4 WiFi WLAN Series IEEE 802.11

According to [12], for SG network, a communication system with high data rate and maximum reliability is required. To meet these requirements satisfactorily, a WLAN is a

good option because it provides high data rate with good reliability. But it is only discouraged due to the smaller coverage area. It ensures interference-free network for multiple users due to the use of Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS) Technologies which use specific access techniques of Carrier Sense Multiple Access Collision Avoidance (CSMACA). Using 2.4GHz frequency ISM band, it provides a high data rate of up to 11Mb/s by DSS modulation technique, and if, Orthogonal Frequency Division Multiplexing is introduced in it, then its data rate boosts up to 54Mb/s [12-13]. By implementing Multiple Input Multiple Output [MIMO] technique, its target is to achieve a data rate of 600Mb/s [12-13]. Despite all these features, it is limited to remote monitoring and Home Area Network (HAN), only due to the small coverage area of the only 100m. Therefore, it cannot be used for the SCADA system where the range of several kilometres is required.

1.2.5 Bluetooth

Bluetooth is a wireless communication technology which eliminates the need of wired network for short-range communication and ensures a high level of secured communication [14]. If the communication is intended between two devices, then the topology used for it is pico-net and integration of pico-nets leads to scatter-net which enables communication among multiple devices (up to eight devices). The devices are first to be set in the mode of transmitting/receiving by human intervention [14], to configure the Bluetooth communication among eight devices within the coverage.

The scatter-net topology makes the Bluetooth suitable to set HAN and to control the home appliances and home automation implementation in SG network. In [15], an attempt has been made to interconnect up to 255 nodes using RugBlue and building private server but, it also does not support more than 8 active devices at the same time. The authors of [16-17] also recommend GPRS and Bluetooth integration based communication system for SG units and vehicle to vehicle communication without considering its range limitations. This technology can be used for small data transfer between two devices within 20m but not for remote micro-grids where communication is carried out over the range of several kilometres.

1.2.6 Radio Teletype

This technology was introduced in the 20th century and was expected to surpass the Fax system and message sending/receiving via wireless rather than the wired network. It was used for the communication of two electromechanical gadgets having keyboards and was used not only sending the text messages but also for voice messages [18]. But, today this electromechanical technology has become almost obsolete with digital technology and micro/nano-electronics.

1.2.7 VHF/UHF

Radio Communication system has been using different frequencies for their carrier signal to transmit the message with minimum noise effect. Radio Frequencies in the band of 30-300MHz are Very High Frequency (VHF) waves, and those who lie in the band of 300-3000MHz are considered Ultra High Frequency (UHF) waves. The selection of carrier

frequency depends upon the mode of modulation (Frequency, Amplitude, and Phase) which is selected based upon the desired coverage area and data transmission rate [19]. Currently, the UK is using UHF for the communication of the SCADA system, and frequency allocation is supervised by UK Radio Communication Agency which assigns the carrier frequency based upon the message signal strength [20]. In UK sub-bands of 450-470MHz have been reserved for SCADA system communication and 183.5-184.5MHz has been reserved for metering purposes.

1.2.8 Optical Wireless Communication

Optical Wireless Communication (OWC) is a technology targeting to replace existing Radio Frequency communication [21]. Existing RF is using a frequency band of 30kHz to 300 kHz and is precisely regulated by international and national authorities. Due to the ubiquitous use of RF gadgets, there is band congestion despite the implementation of different delay-division multiplexing techniques due to which OWC is under consideration [22]. It uses sub-bands of Ultraviolet Radiation and Infrared Visible band, which are still unlicensed and due to which OWC is becoming the counterpart of RF [23-24]. Although OWC is still under research despite that, it is superior to RF in many aspects like broadband, inherent security features, unregulated, free from electromagnetic interference, low power consumption as compared to RF, free from multipath fading issue ad cheaper optical components are the most attractive features of OWC. In Oct 2013, OWC was tested for the ultra-long distance of 384600km (Moon to Earth) under the NASA's Lunar Laser Communication mission, and its results were amazing with the data rate of 622Mb/s [25].

Despite all such amazing features, it cannot be recommended for SG because it requires high and large antennas for every unit and this technology is also still in the experimental phase. Its implementation may lead to the bankruptcy of the project, the example of Boulder SG project is before us which faced the financial issues by implementing optical fibres [26-27].

1.2.9 Satellite Communication

Satellite Communication system is another technology being used for global communication and specifically with the use of Geo-Stationary Satellites and other dedicated satellites world has become like an interconnected village. Communication which required either repeaters, wires or large antennas to magnify and direct the communication signal has been bypassed in this technology, and in future it may surpass the terrestrial communication systems [28]. Due to this unique feature, it has been widely accepted mean of voice and data transmission, and these days few utility companies are also using it for communication with remotely located substations in rural areas as well.

On the basis of orbit and revolution period the satellites (being used for communication) can be categorized in three groups as Low Earth Orbit Satellite (LEOS), Middle Earth Orbit Satellites (MEOS) and Geostationary Earth Orbit Satellites (GEOS). GEOSs are at the height of 35,786km at the equator of the earth due to which a delay of 1/4th second takes place for one-way communication. LEOS are located at the height of about 1200km and are considered the best option for communication because their communication delay is not more than of terrestrial communication delay and that is why it is acceptable [29]. The

authors of [30] have discussed the pros and cons of the implementation of satellite communication for SCADA system and have discouraged only due to high cost.

1.2.10 WiMAX Series IEEE 802.16

Worldwide Interoperability for Microwave Access (WiMax) is widely accepted broadband technology with the target to establish Wireless Metropolitan Area Network (WMAN). It lies in the group of IEEE 802.16 with the unique feature of a high data rate for maximum coverage area. It has achieved 70Mb/s data rate for 50km coverage area [31-32]. Its coverage is sight dependent, whether the area lies on its Line of Sight (LOS) or not. Now another technique of Orthogonal Frequency Division Multiplexing is being implemented to improve the quality of coverage for off-line-of-sight areas. As WiMax is providing efficient service even for long distance using a frequency bandwidth of 1.25MHz to 20MHz and is also known as the Last Mile Technology for remote areas [12]. Due to its unique features, it is being considered the most reliable substitute of T1/E1 links, DSL and PCL. Despite all these features, it cannot be proposed for the SCADA system and SG network due to its lower penetration in the obstacles.

1.2.11 SigFox

SigFox is another popular technology used for Machine to Machine communication (M2M). It uses the license-free band of 868MHz in Europe and 915MHz in North America [33]. It uses Ultra Narrow Band (UNB) ranging from 0.1-0.3kHz bandwidth with Differential Binary Phase Shift Keying (DBPSK) modulation technique at the rate of 100b/s [34]. It transmits a sequence of three messages using random frequencies to ensure

successful reception of the message. Even if two transmissions are lost due to any reason (noise, interference or collision), the base station will receive a clear message [35]. But, SigFox also has certain limitations. Its frequency bands (whether is it of 868MHz or 915MHz) are restricted to 1% Duty Cycle, which means in one hour (3600s) it can communicate for only 36s. Further, for one package its time on air is of approximately 6s which means it can only transmit a maximum six messages in one hour. In this way, its 24 hourly transmissions, are restricted to 140 uplink messages with the size of not more than 12bytes each, and its downlink communication is limited to only four messages, with each message, not more than of 8 bytes [35-36]. Due to these features, it may be suitable for applications requiring very low data rate and a limited number of messages but is not promising for SG.

1.2.12 LoRa

In the emerging trend of M2M and Internet of Things (IoT) Low Power Wide Area Network (LPWAN) technologies are in front line to meet the specifications, and among these LoRaWAN is the most adopted one. It ensures outdoor connectivity of IoT devices with the simplest network structure due to star network of end devices, gateways and central control station and easy management [37].

LoRa uses Chirp Spread Spectrum (CSS) and GFSK modulation techniques [37]. Due to spectrum spreading it uses the whole band of 868MHz (in Europe) and 915MHz (in North America). To minimise the interference effect, it uses six spreading factors from SF-7 to SF-12. Each spreading factor has its different time on air. It uses a bandwidth of 125kHz

of unlicensed frequency band [35]. It provides the maximum data rate of 27kb/s using CSS and 50kb/s by FSK. In Star topology single gateway unit has the capability of communicating with thousands of nodes located several kilometres away from the gateway. Here are its main characteristics which make it a comparatively better option to be adopted for SG and especially fulfils the requirements of the communication system in SCADA system [38]. Due to the below-given features [8-49] it becomes an optimal option for setting up a two-way communication system:

- Long range
- Low power consumption
- Open nature and lesser path loss
- Receiver sensitivity
- Flexibility and low cost
- Reliability

1.3 Wired Communication

There are three main arts of communications for SCADA system which have a physical medium of communication. It includes power line carrier (PLC), digital subscriber lines (DSL) and Optical Fibers (OF). Although these are the most widely used till now because sometimes long power transmission lines are also used for communication purpose and at the receiving end communication signal is extracted using wave trappers. Along with the power transmission lines, an Earth Shield Wire (ESW) is used to protect power lines from lightning and sometimes ESW is also used for communication purpose. These techniques

are beneficial to use already existing infrastructure. But in SG, there is distributed generation, and that is why there are no more long power transmission lines due to which any dedicated communication system is required to encounter all problems of the communication system and to fulfil the requirements. For this, various means of communication have been discussed below.

1.3.1 Power line carrier

PLC is an efficient communication technique which is free from the EM interference and Coloumb's Cage like issues and has been in practice since the advent of the telephone in the late 19th century [39]. In a power system, it has proven effective due to the use of already existing infrastructure and many other attributes associated with it. Few are listed below [40].

- No need of separate communication infrastructure, the existing electrical infrastructure can be used for communication, whether is that an underground or undersea.
- It is efficient and cost effective especially in the power system because no additional infrastructure is required.
- It is free from antenna tempering because it remains energized at 230V
- It faces no setting issues like antenna alignment setting
- Once the electrical infrastructure commissioning is completed the communication system can be energized.

- Today, PLC can be classified into four classes based upon its modulation techniques, data transfer rate, and PLC regulations.
    - Ultra-Narrow Band PLC
    - Narrow Band PLC
    - Quasi Band PLC
    - Broadband PLC

UNB-PLC is the oldest technique, and in the mid of 20th century, this technique was implemented using High Voltage Power transmission lines with frequency ranging 15kHz-500kHz [41]. The limitation of this technique was its one-way communication. The NB-PLC was introduced in the late 20th century, and the amplitude modulation technique was used with a frequency range of 148kHz-500kHz to transmit the message signal through NB-PLC. Its limitations include high power consumption, lack of reliability, and frequently repeaters required to strengthen the signal. China has recently implemented the QB-PLC. In this, NB-PLC and BB-PLC have been merged to achieve the optimal mean of communication. Its carrier frequency ranges from 1MHz-10MHz with a high data rate of approximately 2Mb/s, and it supports two-way communication which makes it suitable for AMI and HAN purposes. The BB-PLC uses a broad band of frequency ranging from 1.8MHz-250MHz and supports a very high data rate of up to 200Mb/s. The problem associated with QB-PLC and BB-PLC is their broader frequency band [42]. In SG PLC has been implemented for automatic metering infrastructure (AMI) in urban areas while other techniques and technologies are still under prototyping.

In [43] the author discusses the different areas for previously mentioned four categories and their standards. According to that, each category can give optimal performance provided that is used for a specific area of operation. For instance, LV PLC, a 3-95kHz band is reserved for power utilities, 125-140kHz for HAN, 140-148.5kHz for security and alarming. NB-PLC can be reserved for audio/voice communication, UNB for AMR and BB-PLC for internet access. But, it did not address the issues of these individual bands affected by the attenuation/amplification factor of transformers in the SG system. Its alternate to overcome a transformer effect is to use wave trapper/filters to extract communication signal from AC (50/60Hz) power signal but, it will not be cost effective for a distributed generation system.

1.3.2 Optical Fibers

Optical Fibre (OF) is another wireline communication technology and is unique in many features. It does not use any copper conductor and has a double-layer cylindrical structure with two parts; core and cladding. Message signal (light rays) undergo from the different angle of refraction based upon the wavelength and frequency obeying Snell's Law for Refractive Index and Total Internal Reflection. A light signal is lesser vulnerable to EM interference and retains strength for relatively larger distance compared to copper-wired PLC. A signal can be transmitted with the same quality up to 100km without any repeater using wavelength division multiplexing technique [44]. Its future is evident in SG as well provided it becomes cost effective.

Data transmission capability of this technique is exponentially growing with everyday innovation. In 1995 its data transmission rate was 10Gb/s, in 2002 up to 40Gb/s [45] and in 2010 data transmission at the rate of 100Gb/s was achieved [46] and in 2015 up to 200Gb/s has been accomplished. In this way OF is rich in data rate and speed in the communication system and has a promising future but, its installation cost is too high to be implemented on a small scale. Usually, its cost goes high due to the extremely small diameter of fibre core [47]. Although, different techniques have been used to make it reasonable for a large scale system but, it is not a financially viable option for the SCADA system of remote grids or SG network.

1.3.3 DSL Based Wired Communication

In general, DSL can be defined as the high speed wired communication system using similar like broadband PLC (BB-PLC) with the data rate of a few kb/s to Mb/s. It uses the existing infrastructure of copper wires of Graham Bell telephone network. Due to that, it has been a popular mean of communication and sometimes was opted for SG as well. Its performance is affected with increasing distance and the impact of length on data rate can be seen from table 1.1. Due to this impact, it is discouraged for the SG/SCADA communication system.

Table 1-1 Comparison of Data Rate for Different Channel Length in DSL [48]

Frequency (MHz)	Bit Rate	Maximum Distance (m)
30	400Mb/s	200
100	1.3Gb/s	120
200	2.5Gb/s	85
300	3.5Gb/s	70

1.4 General Analysis of different technologies

After doing the study of three different wirelines and twelve wireless technologies, a comparison table 1.2 has been made. It can be inferred that the selection of any technology for SCADA system is based upon the requirements of the system and considering the merits and demerits of the communication technology. Selection of wired technology is not a good choice for SG/SCADA especially for remote microgrids and in rural areas where installation and continuous maintenance of the network are required. Its least flexibility toward addition and removal of any node also discourages its implementation. After doing a critical analysis of different wireless technologies, LoRa seems the optimal choice for the SCADA system of remote micro-grids. Furthermore, the security of wireless communication can be improved by implementing any encryption algorithms discussed in the next section.

Table 1-2 Comparison of different communication technologies [5,6,8,9,49]

Technology		Data Rate	Coverage	Remarks
Copper Wired PLC		2-3Mb/s	1km to 3 km	• Unreliable and noisy due to harmonics
DSL	Wired Internet	Max 1Gb/s	100m	• High capital cost for installation and least flexible
	Fiber Optic	Max 14Tb/s	160km	• Extremely high capital cost for installation and least flexible for SG
	Wireless Local Area Network	54Mb/s	200m to 400m	• Short Range • Vulnerable to EMI • Easy Installation
	GSM	14.4kb/s	1km to 10km	• Poor Data rate • Monthly cost • Low availability in remote locations
	BlueTooth	250Mb/s	70-100m	• Limited Coverage • A limited number of nodes
	WifiWLAN	600Mb/s	100m	• Small Coverage • Inherent drawbacks of Wireless Mesh Network
	Radio Teletype	100b/s	Input supply dependent	• Outdated and analogue • Uses

Wireless			(0.7mV/m)	Electromechanical setup
	Optical Wireless Communication	622Mb/s	Unlimited	• Costly setup • Under the experimental phase
	WiMAX	75Mb/s	10-50km (Line of Sight) 1-5km (OFF LOS)	• Poor penetration in obstacles due to HF • Inherent drawbacks of HF
	Satellite Communication	1Gb/s	Unlimited	• High Cost • Signal fading due to snow and rain • Signal latency
	GPRS	170kb/s	1km to 10km	• Poor Data rate • Less reliable due to voice traffic
	Lora	Depends upon SF	2km to 15km	• Low cost • Power saver • Easy Installation • Low data rate
	Zigbee	250kb/s	30m t0 50m	• Short Range • Poor Data rate • Easy Installation
	SigFox	100b/s	3-10km (Urban)	• Low cost • High Scalability • Low data rate • A restricted number of messages per day
	GPRS-(2G)	170kb/s	1km to 10km	• Poor Data rate • Licensed frequency band

	Cellular				• Easy Installation
		High-Speed Downlink Packet (3G)	384kb/s to 14.4Mb/s	1-10km	• Licensed frequency band • Easy Installation • Limited availability
		Long Term Evaluation-LTE-(4G)	Max 42 Mb/s	1-10km	• Licensed frequency band • Easy Installation • Limited availability

1.5 System Security and Encryption

To ensure the privacy and security of data from the eavesdropper, free rider, forger and many other intruders like these is also of the primary concern. Specifically, for SG SCADA system, control commands are very critical and need to be protected from every kind of intruders and eavesdroppers to run the system smoothly and rectify abnormal condition. Different techniques have been implemented for data security and here is the overview of most ubiquitous ones.

1.5.1 Generating Artificial Noise Signal

In this technique, a noise signal is also generated at the same frequency but with lower power as compared to the message signal carrier wave. Transmitter uses two antennas each dedicated for message signal and noise signal separately. At the receiver end and repeaters, the message signal is extracted by eliminating noise set power [50-51].

In this method, an additional power consumption takes place for noise signal. The maximum efficiency of an antenna is 50% due to which a lot of power is wasted in both antennas and specifically in noise generator. In SG, the target is to achieve power efficient and a secure communication system due to which this technique cannot be implemented because of its poor power efficiency.

1.5.2 Physical Layer Method

In this technique, multiple cooperative relays are used between transmitter and receiver. There are two cooperative protocols under use, one is Amplify-and-Forward (AF), and the other is Decode-and-Forward (DF) [52]. AF relay node is just like a repeater; it receives a signal and extracts from the noise and amplifies to retransmit. While in DF, the received signal is decoded, encrypted with any other encryption algorithm and amplified for transmission [52-53]. In this way, security is increased at every DF relay. But, after a certain number of DF relays, the security of the encrypted message declines exponentially because the plain text is extracted from the ciphertext at every DF node which makes it vulnerable to the attack of an eavesdropper.

1.5.3 Cryptography

Cryptography is the most optimal way to secure the data by applying different encryption algorithms and complicating the extraction of real message from the ciphertext. In cryptography, a message (plaintext) is encoded in the ciphertext. Each character of plaintext is assigned any other alphabet to complicate its decoding for the eavesdropper. To do this, the characters of the plaintext can be:

a. Further shifted left/right
b. Assigned different characters for different combinations
c. Shuffled by applying a vigenere cipher technique
d. Combined in rectangles of matrices by applying playfair cipher technique.

Different encryption techniques will be applied to secure the LoRa communication in the SCADA system based upon the security, processing time, required memory, and efficiency of the algorithm. Anyhow a detailed description will be given in the chapter of Encryption Algorithms.

1.6 Research Goals and Objectives

After the literature review, it becomes clear that a communication system for SG SCADA system is required to improve in its security and use-ability. To monitor it remotely, a remote data logging and wider range communication is also necessary to set up. Therefore, the main goals of the thesis project are as follows

1.6.1 Secure Communication

Secured Communication is the prime objctive to achieve using LoRa based communication for the SCADA system. Without security, the critical information of SG network becomes accessible for every intruder. To achieve the security different encryption algorithms will be implemented and finally, most compatible one will be implemented taking the security requirements, processing time, and memory consumption under considerations. Different µC, e.g. arduino, ESP32 will be tested to achieve this task.

1.6.2 Low Power

To build a low power system is the second objective to achieve. It will be achieved by trying low power ICs and operation at low voltage. Finally, the combination of low power controllers and other low power components-based setup will be devised.

1.6.3 Regular Updates

Regular updates about the system condition are also important to ensure the proper system functionality. It will be achieved to inform the concerned personnel about system conditions and specifically in the abnormal conditions which may be due to any fault or harbinger of any failure.

1.6.4 Local and Remote Data Logging

Local and remote data logging is also as important as any other feature. Remote data logging is necessary for remote monitoring, and local data logging is for data backup. It will be achieved by logging data on any remote server, e.g. Thingspeak, and local data logging will be achieved using SD card.

1.6.5 Implementation

Implementation of the designed system with all previously mentioned features will be done on a prototype system. All this will be integrated after devising coding/programming and designing optimal algorithms to achieve the requirements of the SCADA system. Research

outcome of this work will be a design and demonstration of a secure, low cost, low power LoRa based communication system for remote monitoring of micro-grids.

1.7 Thesis Overview

This research work is the part of the project of NSERC Energy Storage Technology Network (NESTnet) to address the challenges of renewable sources with the collaboration of 15 other universities and many industries. Out of these 15 universities MUN is supposed to work on the design and implementation of a low cost, secure and low power SCADA system. In the research work behind this thesis, a secure, low-cost and low power communication system with coverage of 35-40km has been developed. The thesis has the following chapters to achieve all the features listed in section 1.7,

Chapter 1 is about the introduction of the SG SCADA system, advantages and different challenges in the implementation of the SG SCADA system with primary focus on the communication system. Mostly it comprises on the literature review of different communication technologies which are in practice today. Chapter 2 is about the setting up LoRa based bi-directional communication, encryption algorithms and their implementation, security, and complexity to breach the algorithms. Chapter 3 is on the implementation of Advanced Encryption Standard (AES) Algorithms, implementation of Message Authentication Code (MAC), range testing and data rate and the results of all these steps. Local and remote data logging, regular updates, and implementation and interfacings of different gateways have been explained in chapter 4. Radio-based broader range setup, serial/ethernet communication, configuration, their results and finally three

different system topologies have been explained in chapter 5. Chapter 6 summarizes the project work with the merits/demerits and limitations of the devised system and identify few methods which can help to improve the system.

Chapter # 2 Encryption Algorithms to Secure Communication System and Their Implementation on DRF1276G LoRa Module

2.1 Introduction

In this chapter, LoRa based bi-directional communication system, encryption algorithms and their implementation, security and complexity to breach the algorithms have been discussed. Complete design and lab test results have been included.

With the growing penetration of renewable energy in grid supply, distributed generation is increasing. Specifically, for remote communities who have added renewable energy to their energy mix have a lack of secure and authentic communication system [54-56]. There are many cases in which spies and enemies controlled the SCADA systems and disrupted the power system. In 2008, the Russian army took charge of the Georgian electric grid by controlling the SCADA system of their electric grid. According to the Wall Street Journal report, in 2009, spies hacked the control system of the U.S electrical grid and disrupted the system [57]. Many other articles like [58] have also highlighted the concerns about smart-grid cybersecurity. Therefore, the communication system of an electrical grid, specifically,

the setup related to the SCADA system, must have strong resistance against eavesdroppers and masqueraders. Usually, a communication system is regarded as secure if it satisfies the following four features [59-60].

2.1.1 Privacy

The message should be encoded or encrypted such that the only authorized receiver can read the message.

2.1.2 Message Authentication

The message should be authentic, and only the privileged nodes should be able to send the message. Furthermore, no eavesdropper should be able to masquerade the receiver by sending fake messages.

2.1.3 Integrity

The message received at the receiver side must exactly be the same as the sender sent.

2.1.4 Nonrepudiation

If there is any alteration in the message, whether due to the channel error or attacker's interference, the receiver must be able to recognize that and decline the message.

In [61-63], a few techniques have been discussed to address communication security issues. In their proposed methods a third party is involved to ensure the security of the communication network or setup, which depends upon the third-party network to communicate with Remote End Devices (RED). Different encryption algorithms have been proposed in [64-65] to secure the communication system using different cryptographic techniques like; shift-cipher and substitution-cipher but, for a cryptanalyst, they are too

simple to break or other encryption algorithms proposed in [66],[67]. A cryptanalyst can easily take control of the system and can modify the control messages as demonstrated in Figure 2.1. In the Fig. 2.1, an eavesdropper receives the message from the SCADA unit, and modifies the messages and control command and sends that to the remote end device (RED) pretending to be the SCADA unit and hacks the system. In this way control information becomes prone to the eavesdropper and loses authenticity and security. Specifically, in a smart grid network, secure communication between the energy meters and the SCADA system requires a low cost and a secure communication setup with improved power efficiency. A raspberry-pi could also have been used for gateway purposes to provide remote access, just like [69],[71] but, that consumes 3-4 times more power than the tiny DRF1276G. In this chapter, different encryption algorithms have been implemented to secure the communication system. After the implementation of these algorithms, C++ programs were written to breach the encryption, and their results have also been discussed at the end.

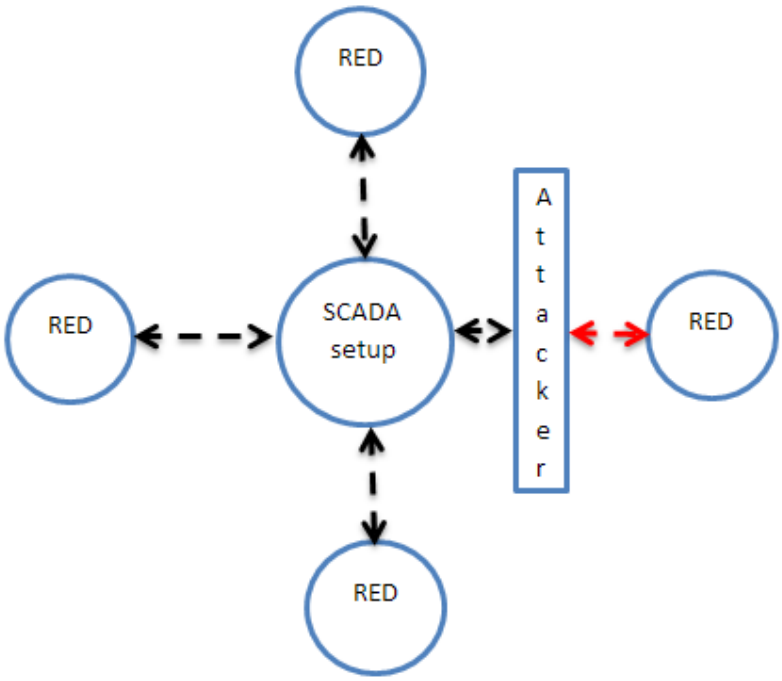


Figure 2-1 Eavesdropper Masquerading the SCADA Network

2.2 Hardware Setup

The layout of the system has been shown in Figure-2.2. In this, each Remote End Device (RED) has sensors and Arduino-uno with DRF1276G LoRa module. It receives data from the sensors and encrypts. After encryption, it adds node ID in message string and sends. The local collector node receives the LoRa messages from RED nodes and confirms message authenticity. If the message is authentic, the collector node adds its ID along with the RED node ID as its signatures and forwards to the SCADA unit side data collector.

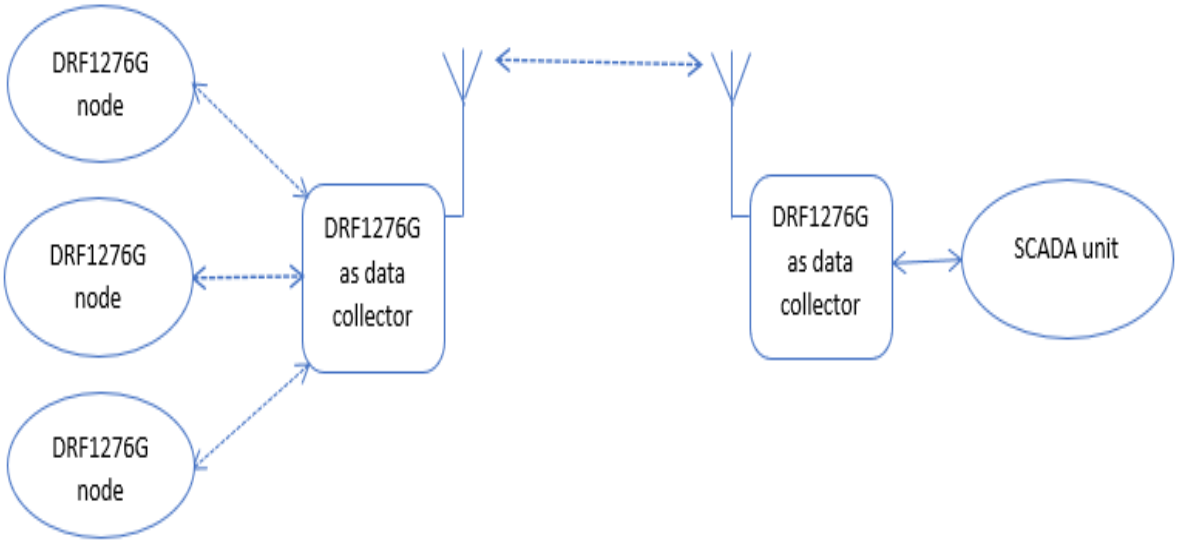


Figure 2-2 System block diagram with LoRa based wireless communication

On SCADA side, the collector unit receives the LoRa message, and after verifying the signatures of remote end side data collector, it decrypts the message and sends to the SCADA unit for processing. Similarly, for downlink communication, the SCADA side data collector receives the message from the SCADA unit and encrypts the message. The encrypted message is sent after the addition of the sender signatures and the target RED node ID. The data collector DRF1276G of RED node side receives the LoRa message verifies the signatures of the sender (SCADA side sender). After signatures verification, the message is forwarded to the targeted RED node. If any node other than the target node receives the message, that will ignore the message because of different IDs, and if it is the target node, then the message will be decrypted and processed. In this way, two-way encrypted communication is set to continue. This system structure was implemented using Arduino-uno with LoRa module shown in Figure-2.3. It was chosen because of its user-

friendly nature and easy to program in Arduino IDE software. The other main advantage of this tiny controller is that it has onboard LoRa module, unlike regular Arduino-uno for which a separate LoRa shield is required for LoRa module interfacing. The pin diagram of the arduino-uno and with LoRa chip have been shown in Figure 2.4 and 2.5. It has six analog input pins to connect sensors and fourteen digital I/O pins (with six pins dedicated for PWM). Its operating voltage is 3.3-5V.

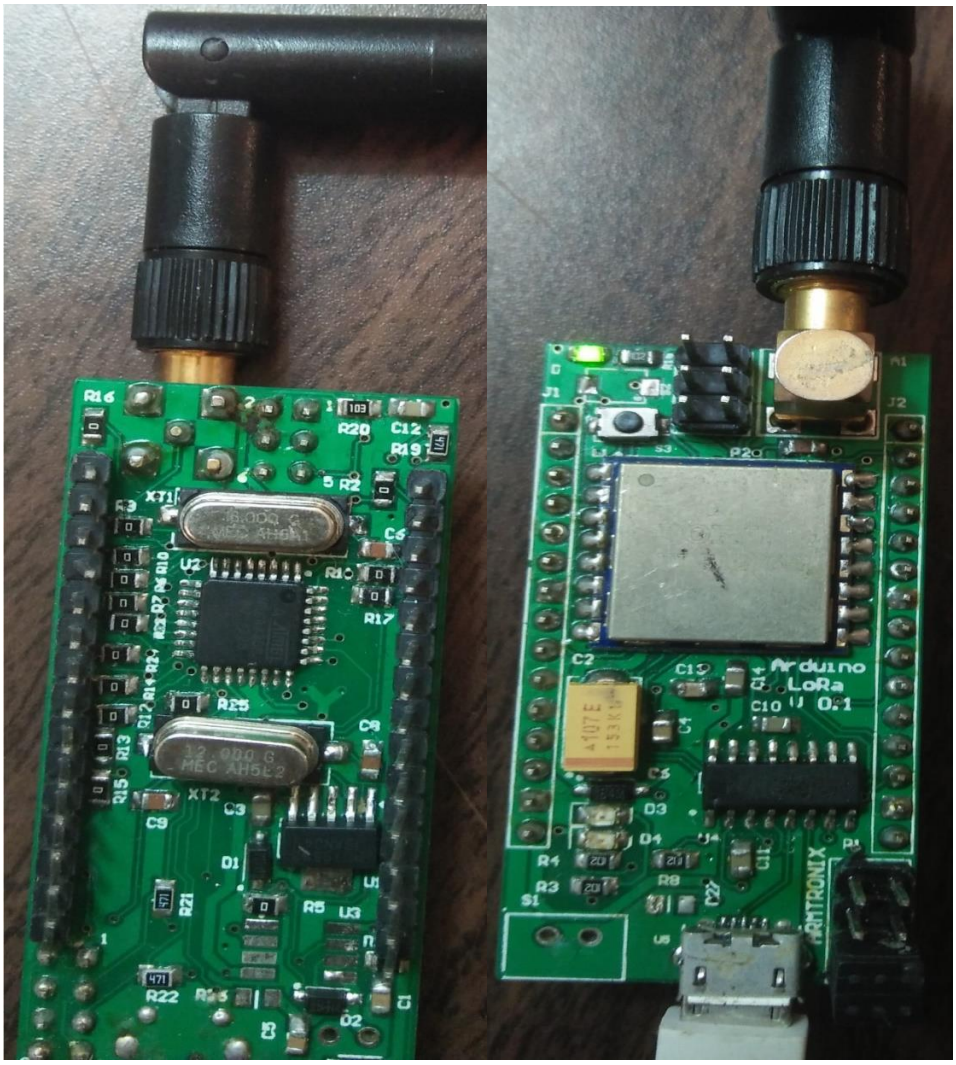


Figure 2-3 Arduni-uno with DRF1276G Lora module

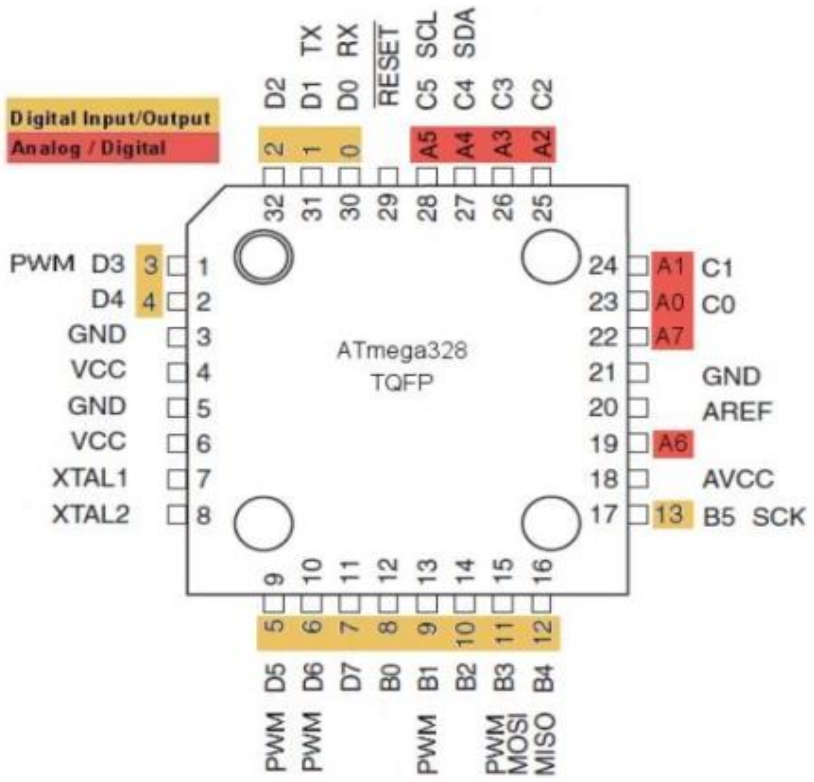


Figure 2-4 Pin diagram of arduino-uno

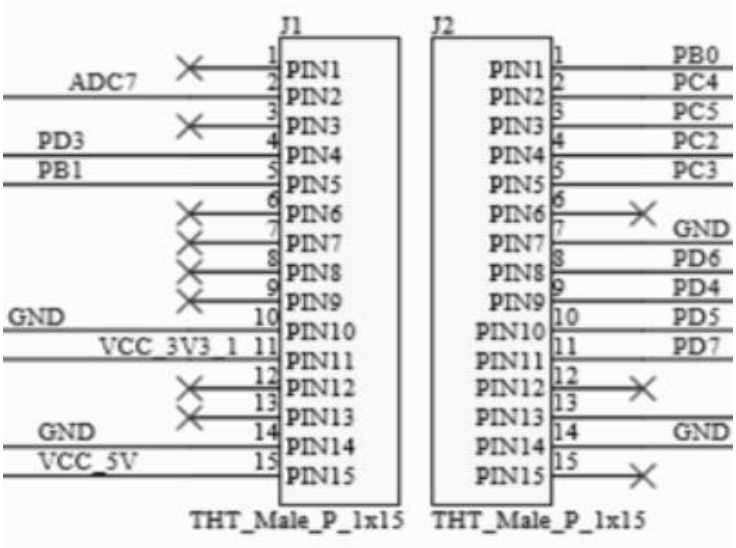


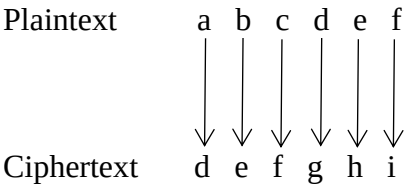
Figure 2-5 Pin diagram of arduino uno with DRF1276G LoRa module

2.3 Cryptographic Algorithms on Arduino with DRF1276G LoRa Module

Multiple encryption algorithms were implemented on the arduino DRF1276G LoRa module to achieve the previously discussed four features of a secure and authentic communication system for microgrid but, all cryptographic algorithms do not provide the equal secrecy level. Their secrecy level is proportional to the key size required to breach the encrypted text. Below are the details of the encryption algorithms which were implemented and their resistance against any attack has also been discussed depending upon the size of their keyspace.

2.3.1 Shift Cipher

In shift-cipher, all characters of the message are shifted by the same number. For example, if the message is ‘abcdef’ and each character is shifted right by three characters, then after shift ‘a’ will go to ‘d’, ‘b’ to ‘e’ and so on as shown below.



As there are only 25 possible shifts (if the message contains simple alphabets only), it means that its key set has only 25 elements and cipher can easily be decrypted within 25 attempts [68].

In this encryption, although cipher text becomes different than actual message, it is not difficult to break because of the limited keyspace. In the English language, all characters have different occurrence probability in the plaintext messages. Their occurrence

probability helps in breaching the message. From the below-given charts in Figure 2.6 and 2.7, it can be seen that the letter ‘e’ has the highest occurrence probability followed by the ‘t’, and the letter ‘z’ has the least occurrence probability after letter ‘q.’ Attempt was made to break the encrypted message by counting the occurrence frequency of the letters and then making some assumptions based upon those. It did not give much resistance and was hacked after some mathematical calculations. Therefore, this algorithm has been discouraged for the security of SCADA system.

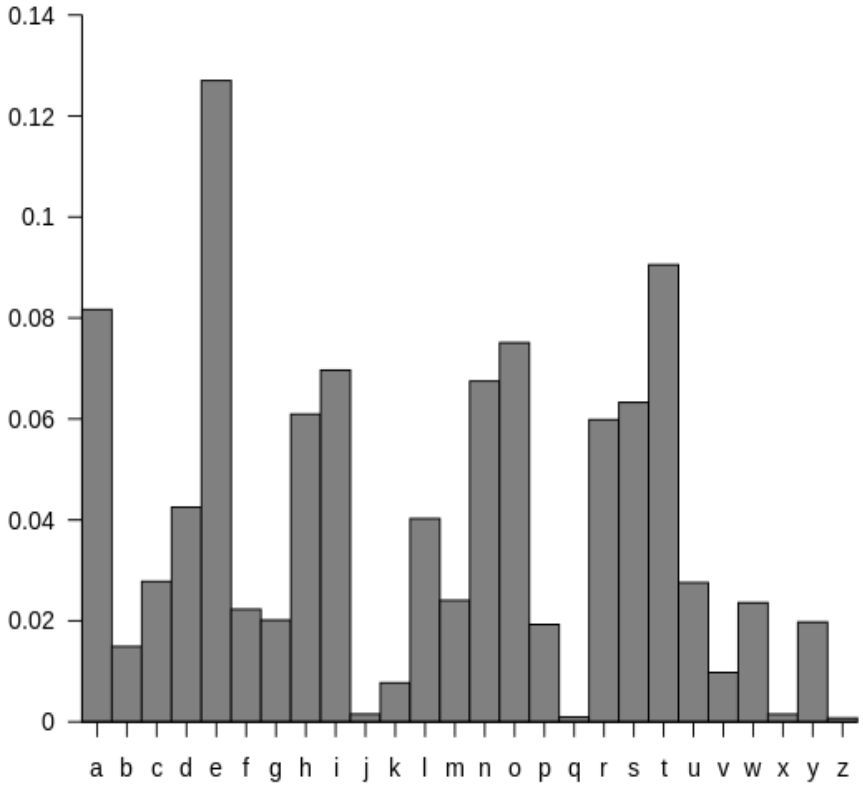


Figure 2-6 Occurrence probability of alphabets in any sentence

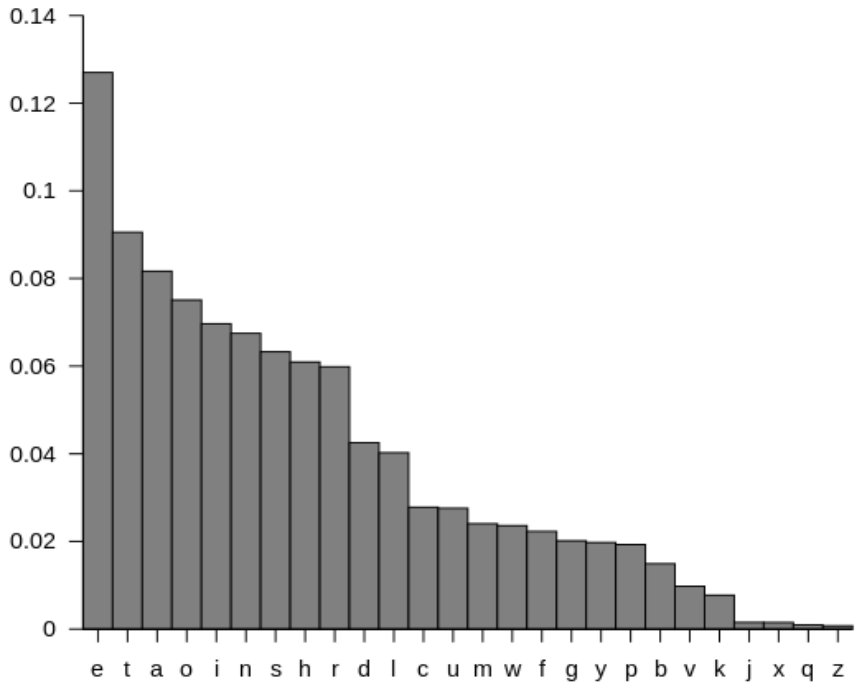


Figure 2-7 Highest to the lowest occurrence probability of alphabets in any sentence

2.3.2 Affine Cipher

The keyspace of affine-cipher is a little bit larger than of shift-cipher. In this technique, the ciphertext is calculated by solving linear equations under modulo 26 (because there are only 26 alphabets). Each alphabet is assigned a certain number. Most commonly, the values are assigned as shown in Table-2.1. There are two common conventions for assigning number to alphabets, sometimes values are assigned from 1 to 26 and sometimes from 0 to 25. The equations used for this algorithm are first-order algebraic equations

For instance, ‘y’ indicates ciphertext number and ‘x’ indicates plaintext number then the linear equation for this cipher becomes like:

y=a*x+b

where ‘a’ and ‘b’ are constants but less than ‘m’, where ‘m’ is the integer of the modulus. Its keyspace depends upon the possible values for ‘a’ and ‘b’ and it will be:

$K=\{(a,b) \epsilon Z_m \times Z_m \mid \gcd(a,m)=1\}$

To ensure that each ciphertext is for only one plaintext character, the gcd(a,m) must be equal to 1 for injective ciphertext. For regular alphabets, the ‘m’ is 26 so,

Keyspace=(# of possible values for ‘a’)x(# of possible values of ‘b’)

- ⇨ Keyspace = 12*26=312

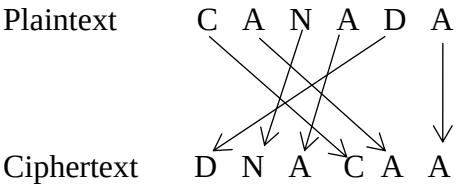
The possible values for ‘a’ are twelve and for ‘b’ are twenty-six, and the encryption is injective in nature. So, this cipher could be decrypted within 26x12=312 attempts [68].

Table 2-1 Alphabets with assigned numbers for mathematical calculations in encryption algorithms

Letter	Plaintext/Ciphertext number	Letter	Plaintext/Ciphertext number
A	0	N	13
B	1	O	14
C	2	P	15
D	3	Q	16
E	4	R	17
F	5	S	18
G	6	T	19
H	7	U	20
I	8	V	21
J	9	W	22
K	10	X	23
L	11	Y	24
M	12	Z	25

2.3.3 Transposition Cipher

In this cipher, the characters of a chosen block size are not substituted, instead they are randomly shuffled with each other within the plaintext block e.g.



The security of the algorithm depends upon the size of the encryption block, and with increasing block size, security also increases. If a block has 'n' characters, then the key set will have total n! possible values [68].

2.3.4 Hill Cipher

Hill-cipher is based upon linear algebra, and its feature is that it is not an injective cipher. It is similar to the affine cipher in the encryption/decryption structure. The only difference is that it works on matrixes and columns of plain/ciphertext unlike characters used in the affine-cipher. In this cipher, all characters are assigned numbers, eg. a=0, b=1 and similarly y=24 and z=25 similar to affine cipher table. It uses an nxn square matrix as a key matrix to get the column matrix of ciphertext from the column vector of plaintext. For example, if the key matrix is $\begin{bmatrix} 5 & 19 \\ 24 & 3 \end{bmatrix}$, and 'GA' is to be encrypted. The respective plaintext column will be $\begin{bmatrix} G \\ A \end{bmatrix} = \begin{bmatrix} 6 \\ 0 \end{bmatrix}$, and after encryption the ciphertext will be $\begin{bmatrix} 4 \\ 14 \end{bmatrix} = \begin{bmatrix} E \\ O \end{bmatrix}$ 'EO' as shown below.

$$\begin{bmatrix} G \\ A \end{bmatrix} => \begin{bmatrix} 5 & 19 \\ 24 & 3 \end{bmatrix} * \begin{bmatrix} 6 \\ 0 \end{bmatrix} = \begin{bmatrix} 30 \\ 144 \end{bmatrix} = \begin{bmatrix} 4 \\ 14 \end{bmatrix} = \begin{bmatrix} E \\ O \end{bmatrix} \ (mod\ 26)$$

The only condition for a key matrix is that it must be an invertible matrix to ensure that the respective decryption key matrix also exists. Although, hill-cipher key space is m^{n^2} where 'm' is the modulo and 'n' is the size of the matrix but, it is vulnerable to chosen plaintext attack [68].

2.3.5 Substitution Cipher

Substitution-cipher gives much better security than the shift and affine-cipher due to large key size. In the implementation, it is quite similar to the shift cipher but, each plaintext character is not shifted by the same number, eg.

Plaintext a b c d e f

Ciphertext d z h k a f

The first character can be substituted by any of the other 25 characters, second character by any of the rest of 24 characters and so on. In this way possible key size becomes

$$|K|=25x24x23\ldots\ldots1=25!$$

2.4 Breaching the Encryption Algorithms

After implementing these encryption algorithms, few C++ codes were formulated to verify the resistance of encryption algorithms. In these breaching algorithms, the advantage of certain loopholes (e.g., the occurrence frequency of the characters and pairs) and certain logical assumptions (e.g., type of encryption algorithm/ size of encryption block) were made.

From the keyspace of different algorithms, it has been seen that hill-cipher is relatively more secure than all other algorithms. Although, the keyspace of substitution is larger than

all others but, it is weaker than the hill-cipher due to its injective nature. Hill cipher using 2x2 matrix size has the keyspace of m^{n^2}=26^4=456976 possible keys. Although, it seems very large keyspace and difficult to breach but, it was breached using C++ code given in appendix A. Below is the example of a message encrypted using Hill Cipher and the decrypted plaintext extracted after applying the code of appendix A.

Cipher Text:-

FUGOMRPKWODRDTJHDTHPQTJDADFKLZDTYFWCDLJDALRPNNQVPHZDR

FMIPPSXWOSWDFUXELKMMGVOGBODEACSZJJDNDZROFMRAHAERPEAFJT

NCZXPUDKQDITNYPFKNFBKOODBZPECFKWYXPGBEACFUQNQECCXSTGCZ

DCKGBZRQVFKBKVRIIVRBKZJECUVANJQVRBKQGZRBKDLECNSMGULBKV

RIIVRGBFKKLBRGBANKOMGLCECIZRPNNGJQZDTUVKLRFECDSSFSXKLCA

HLTRWUCOFGLNSXWOBKSYKLKDXPNLWUELQEFVQMQEVSJVQMUYKLLD

HVXPBKCKCZMEFGDTNFGENYYIDOZDBHOSSCGPCKUVODBKIUFGNFYKQ

VYMJQYVBXWRPQKELWGBQTWYFJ

Plain Text:-

Themonkeyspawpartiwithoutthenightwascoldandwetbutinthesmallparlouroflaburnumvilla

theblindsweredrawnandthefireburnedbrightlyfatherandsonwereatchesstheformerwhoposse

ssedideasaboutthegameinvolvingradicalchancesputtinghiskingintosuchsharpandunnecessa

ryperilsthatitevenprovokedcommentfromthewhitehairedoldladyknittingplacidlybythefireh

arkatthewindsaidmrwhitewhohavingseenafatalmistakeafteritwastoolatewasa

Figure-2.8 and 2.9 also show the results of the Hill Cipher algorithm-based encryption and the two-way communication between two LoRa modules. Figure-2.8 shows the encrypted message sent from the LoRa node of inverter side (message sent from remote end device side). Figure-2.9 shows the plaintext message of SCADA side and the decrypted received message.

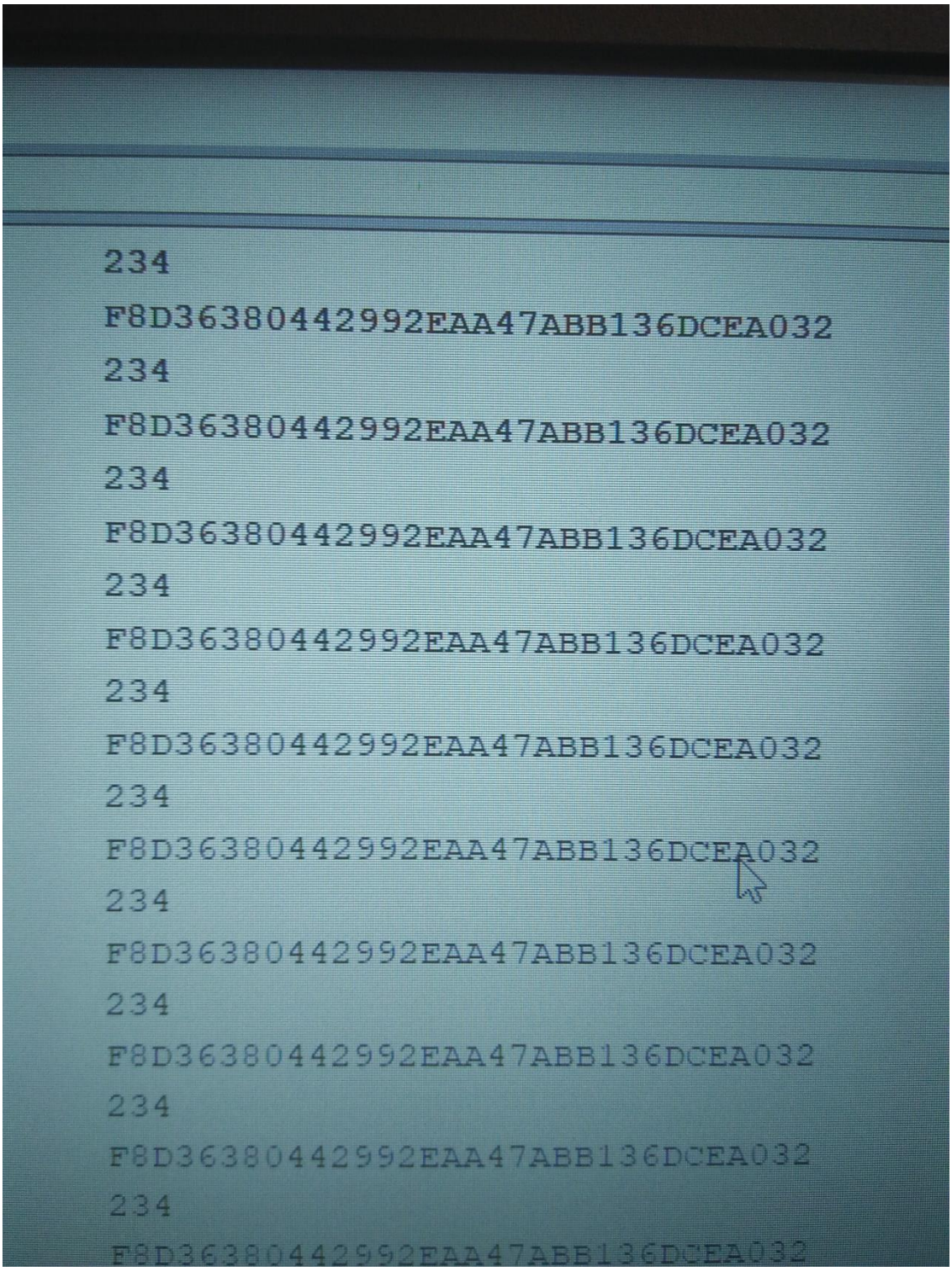


Figure 2-8 Encrypted message sent from RED side Lora module

Figure 2-9 Two-way communication between two LoRa modules

2.5 Conclusion

All previously discussed cryptographic techniques can be implemented for simple communication purposes. This setup of duplex communication gave the range of 5-15km (on/off-site) with power consumption of 500Mw (2x250mW) and cost not more than CAD60 for two units to set up a two-way communication. It was planned to use Hill Cipher (relatively secure one) for the encryption of SCADA system messages but, after finding some loopholes and successful attempts of breaching Hill-cipher (developing C++ code given in the appendix) its implementation for the SCADA system has been discouraged. It is also vulnerable to attack by any cryptanalyst due to the limited number of keys in

keyspace. That is why the implementation of the Advanced Encryption Algorithm (AES) has been chosen to ensure the security of the SCADA system which not only has large key space of 2^{128} possible keys but also is non-linear in nature and is flexible to change the pattern of output by changing the number of cascaded encryption rounds. Usually, it is used for military purposes and where extreme security is required. The arduino with DRF1276G LoRa module cannot support the complexity of the algorithm due to its small flash size (32kb) and CPU limitations. To implement AES, ESP32 has been used which not only has enough flash, better CPU and low cost but also has very little power consumption 3.5-5mW.

Chapter # 3 Implementation of AES and MAC to Secure Communication System for Remote Micro-grids

3.1 Introduction

After implementing various encryption algorithms (explained in the previous chapter), attempts were made to break them to check their security. Some C++ programs were developed, and some online softwares (ESEnfCVPOA, Crypto Corner, CodeBreaker etc.) were used to check their resistance. From such cross-checks and their keys-pace it was inferred that those are not perfectly secure algorithms. Therefore, Advanced Encryption Standard (AES) Algorithms were studied, learned and implemented. Here, AES has been implemented first using DRF1276G LoRa module but, it did not work due to controller limitations. Then, ESP32 was used to secure the LoRa communication by implementing AES algorithm. The message authenticity and non-repudiation has been achieved by implementing a 64 bit MAC address for each message regardless of message size. It gives

the ability to identify even a single bit change in the message which could be due to any eavesdropper or strong electromagnetic interference.

Figure 3.1 shows the flow diagram of the process. Before sending any message, first that is encrypted using the AES encryption algorithm, and then a 64bit unique MAC is generated from the plaintext. Finally, the ciphertext and the MAC are concatenated and sent. Similarly, on the receiver side, first the MAC and the ciphertext are separated, then the MAC is verified, and then ciphertext is decrypted to process further. If the received MAC address at the receiver end does not match with the MAC calculated from the message then received packed would be considered suspicious and will not be executed.

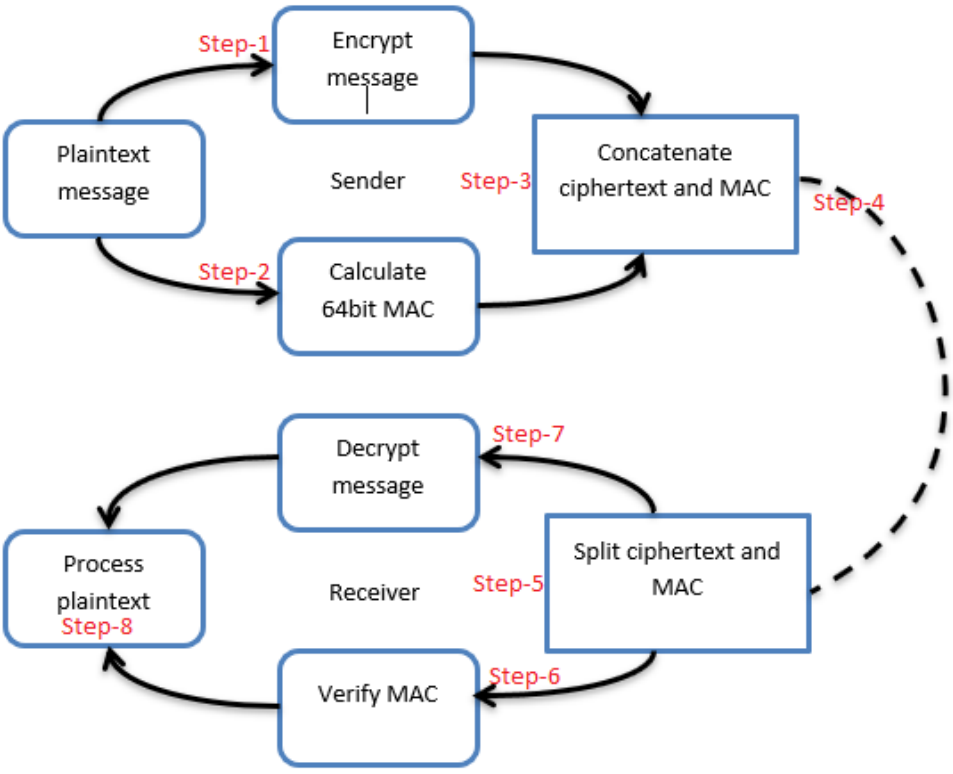


Figure 3-1 Flow chart of the implemented communication process

3.2 Implementation of AES Algorithm using ESP32 and LoRa Module

The AES algorithm not only has a large key set (2^{128} possible keys), but also is secure from many cryptanalysis algorithms like differential cryptanalysis, integration, linear, multiset and many others like these. So far, it is regarded as the most secure one to all known attacks [72],[73]. The flow of the AES algorithm has been shown in figure 3.2. The left half of the figure shows the encryption flow and the right half shows the decryption flow. Further, these parts have been explained in the coming section. Each encryption round has the five steps which go sequentially;

- Add round key
- Substitute bytes
- Shift rows
- Mix columns
- Add round key

Similar to encryption, each decryption round also has the five steps

- Add round key
- Inverse mix columns
- Inverse shift rows
- Inverse substitute bytes
- Add round key

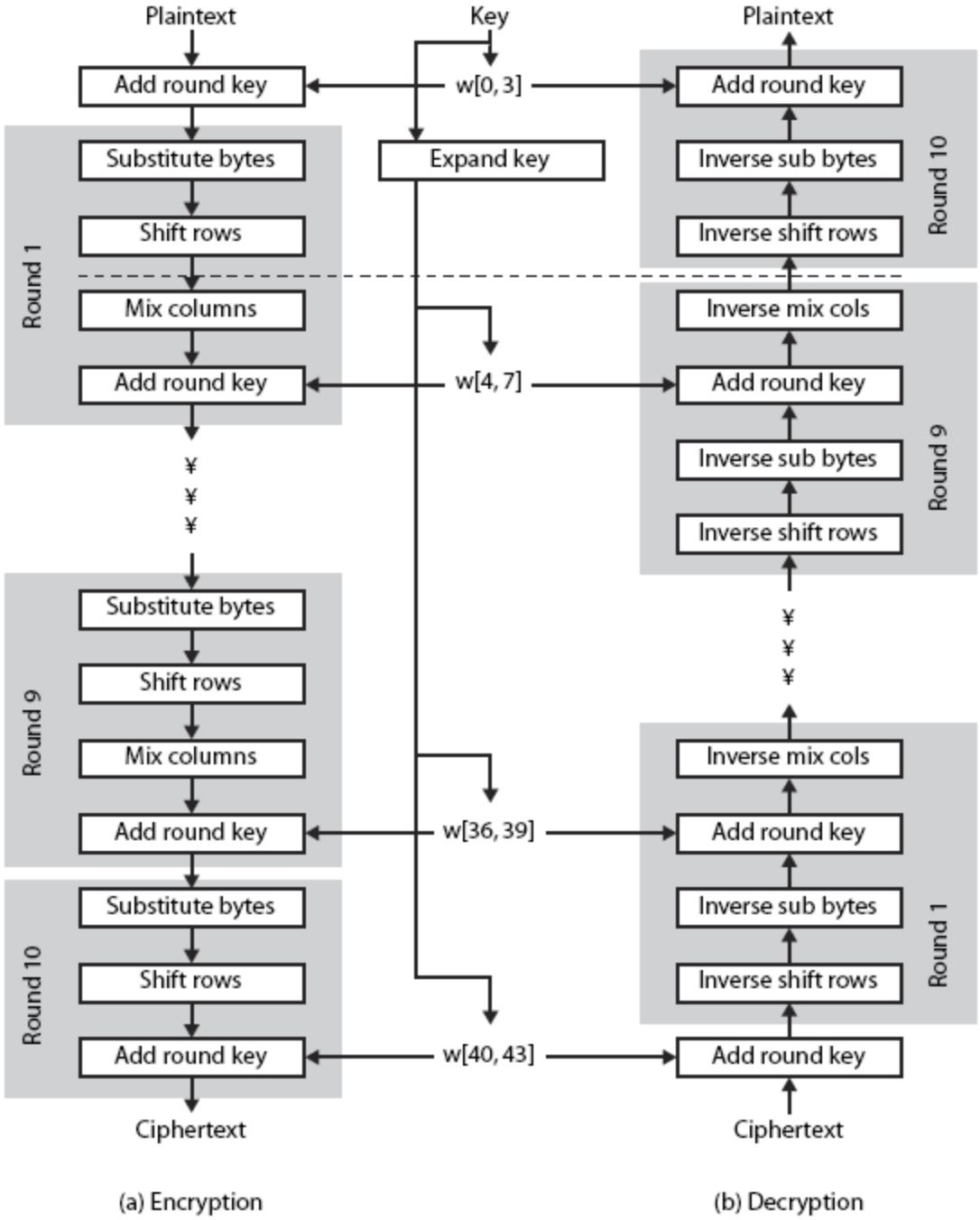


Figure 3-2 Step by step 10 round AES encryption and decryption [72]

3.2.1 Adding the round key and generating a new key

The addition of the round key has been shown in figure 3.3. It is the XOR sum of the binary strings of the message and the key. The condition for this addition is that the length of the message string and the key must be equal. Further, the AES encryption maps the message into a square matrix (4x4 bytes or 32x32 bits). In key addition operation, the first 32-bits of the message are XORed with the first column of the key matrix. The addition of key gives the sub-round ciphertext. In figure-3.4 the generation of the round key has been shown. For the first round, any default or input key is used, and the key for the rest of the rounds are calculated from the previous round key, plaintext message and the seed (any word of 4-bytes or binary string of 32-bits). The round key of the previous round is mapped into a square matrix of 4x4 byte size elements. In the next step, each column of the key matrix is XORed with the respective column of the word.

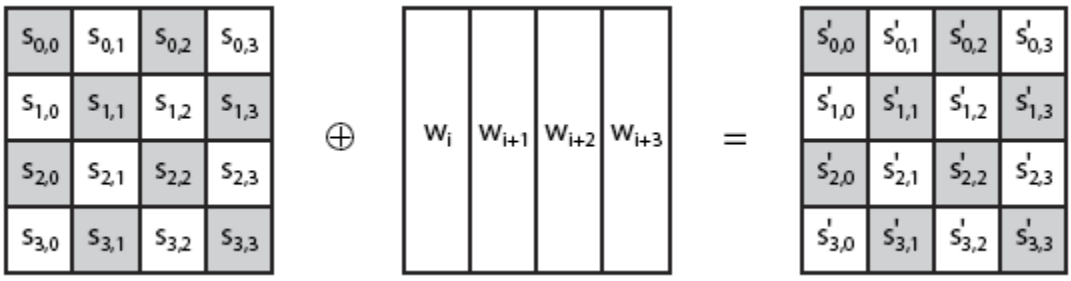


Figure 3-3 Addition of round key for AES encryption [72]

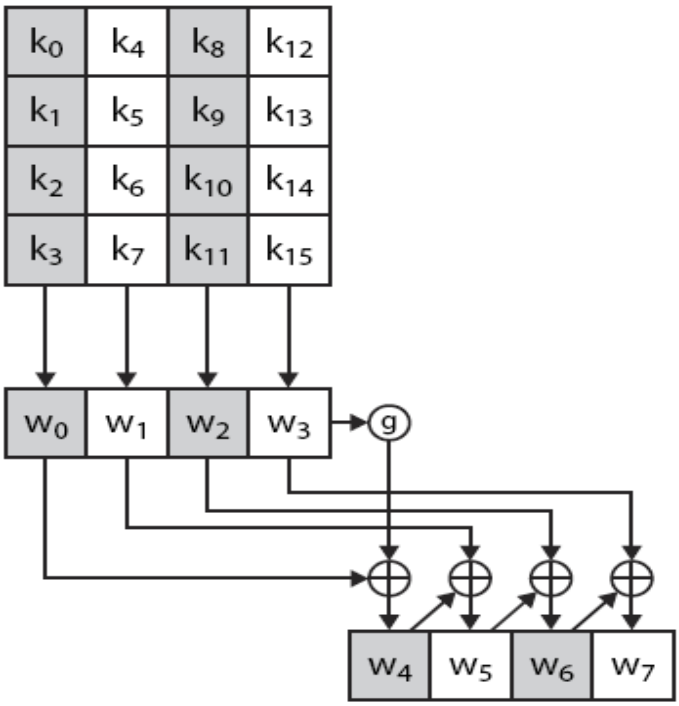


Figure 3-4 Generating new key from the previous round key and data string [72]

3.2.2 Substitute Bytes

After calculating the next round key and the addition of key in the previous round ciphertext (or plaintext for the first round), each byte (the pair of HEX characters) is replaced with the respective Rijndael's table (a standard table of 256 values) to increase the confusion.

3.2.3 Shift Rows

After substitution, all 16 bytes are distributed to construct a 4x4 square matrix. In the resultant matrix, the row rotation operation is applied. It has been shown in figure 3.5

- The first row of the matrix remains unchanged
- 2nd row of the matrix is rotated left by one byte

- 3rd row is rotated left by two bytes
- 4th row is rotated left by three bytes (or rotated right by one byte)

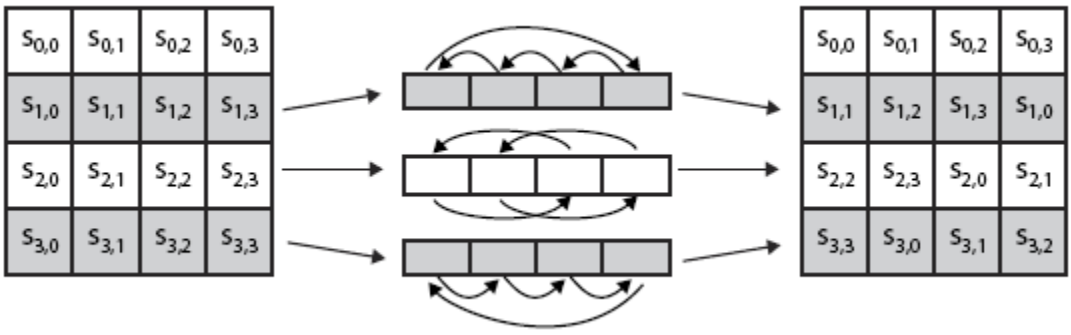


Figure 3-5 Rotation of bytes in shift row operation [72]

3.2.4 Mix Column

As shown in figure 3.6, the matrix left multiplication is applied using a 4x4 matrix on the results of the shift row operation. To get the results of multiplication by two without data information loss is achieved in two steps. Apply left shift on the binary value of the data and add 0 on LSB side. The second step depends upon whether the MSB was 0 or 1. The second step has two ways depending upon the MSB value

- If the MSB of the data was 0, then the final result of 2s multiplication remains the same as of left shift by one bit
- If the MSB of the data was 1, then the final result of 2s multiplication is achieved calculating XOR sum of the results of left shift and (00011011)

The multiplication of the data by 3 is quite simple. It is the XOR sum of the 2s multiplication and the data.

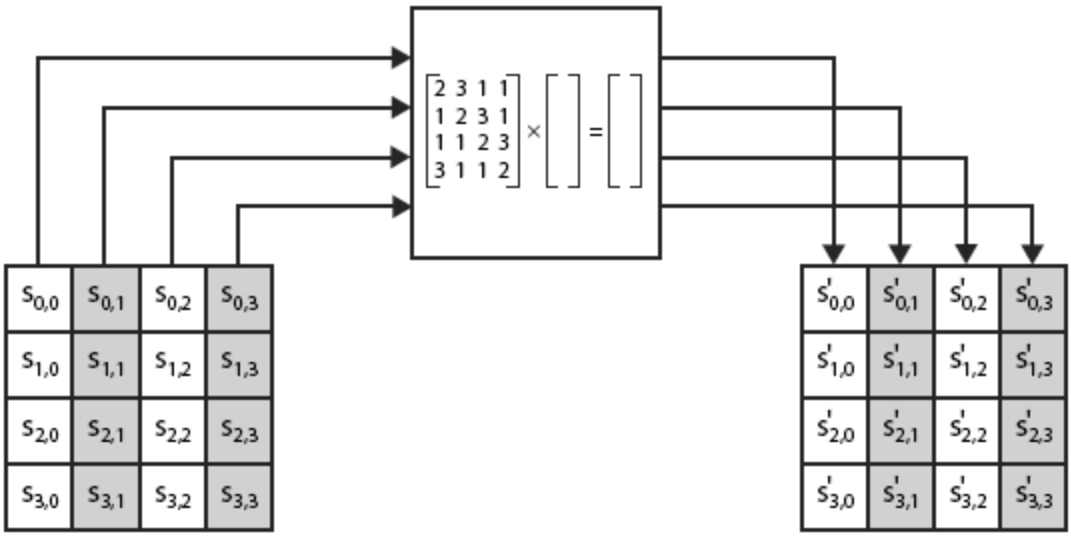


Figure 3-6 Mix column operation to increase the confusion and non-linearity [72]

This algorithm not only has a large key set (2^{128} possible keys), but also is secure from many cryptanalysis algorithms like differential cryptanalysis, integration, linear, multiset and many others like these.

In the AES algorithm, for each round, a new key is derived from the previous round key and the ciphertext of the previous round. From the test results, explained in the next section, it could be seen that each round ciphertext is entirely different from all others which is due to the implementation of binary level encryption. The confusion created at each round and the propagation of confusion from one round to the next round makes it more secure. Its complete code is provided in appendix B&C.

3.3 Results

The AES algorithm was implemented on DRF1276G, and the results have been shown in figure 3.7 but, it did not support the AES algorithm due to small flash size (32kb), slow

processing and other processor limitations. It could not even execute the single round for encryption. Therefore, ESP32 shown in figure 3.8 was used which not only supported this algorithm but also the implementation of Message Authentication Code (MAC) as well. The MAC has been implemented in the later sections, and the results of AES implementation ESP32 have been shown in figure 3.9.

It can be seen from the results of the AES implementation on ESP32 that each round ciphertext is different from the previous round. It is due to the bit level encryption. In each block of encryption, there are 128 bits due to which 2^{128} possible key sets would be required for a cryptanalyst to break this successfully. Moreover, if, any computer program is developed to break this cipher that would also take thousands of years to perform 2^{128} calculations. Figure 3.10 gives a brief insight about such large numbers. Even if a supercomputer which performs single encryption in 1ns will also take more than 60 billions of years to break it. Therefore, its implementation for SCADA system resolves the security-related issues and gives perfect security against eavesdropper. Figure 3.11 shows the received message with the MAC address and the status of its authenticity. On the receiver side, the received message is parsed into ciphertext and the MAC address then the MAC is calculated from the received message and is compared with the received MAC. If both MACs are equal then the message is considered authentic one and is processed; otherwise the message is declared suspicious and is not processed.

```
pre-encrypt(plain_text):        0123456789ABCDEF0123456789ABCDEF
Ciphet-text after 1
Ciphet-text after 2 rounds:
Ciphet-text after 3 rounds:
Ciphet-text after 4 rounds:
Ciphet-text after 5 rounds:
Ciphet-text after 6 rounds:
Ciphet-text after 7 rounds:
Ciphet-text after 8 rounds:
Ciphet-text after 9 rounds:
final ciphertext:
```

Figure 3-7 Implementation of AES encryption using DRF1276G with LoRa module

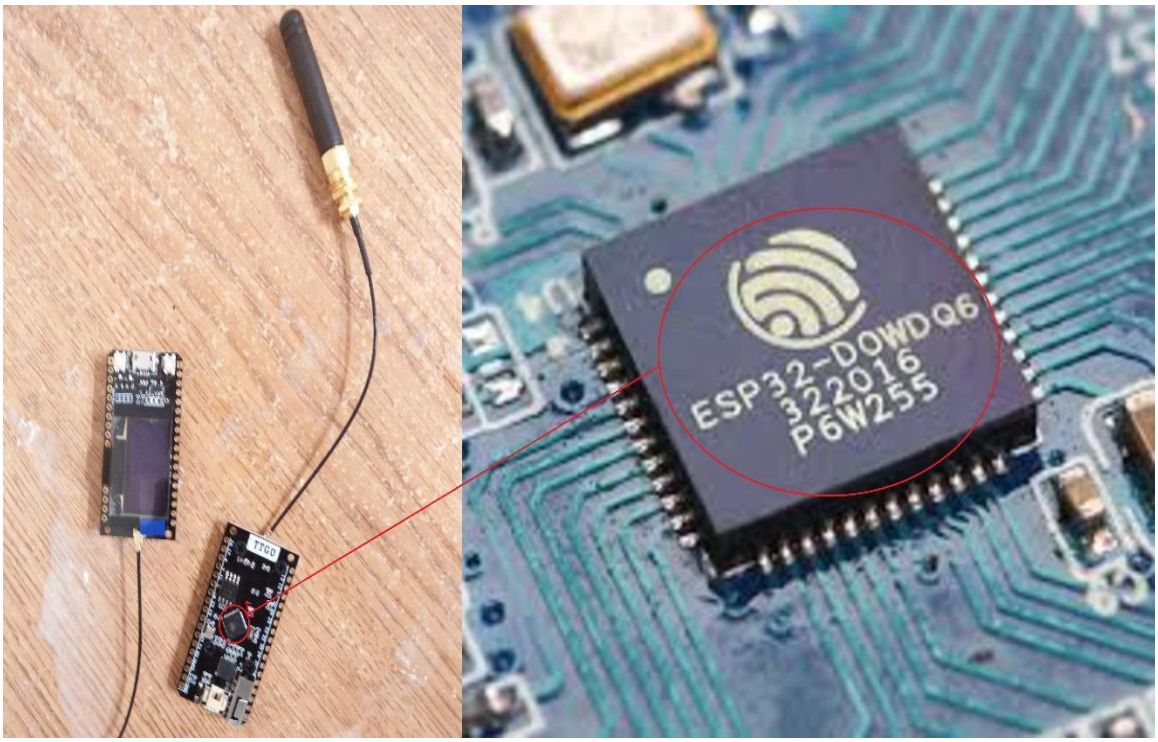


Figure 3-8 ESP32 with LoRa module used for successful implementation of AES encryption

```
pre-encrypt(plain_text):        0123456789ABCDEF0123456789ABCDEF
Ciphet-text after 1 rounds:     4023CABB284333AC463817F42893333D
Ciphet-text after 2 rounds:     4716512657507AE6F0BFB407EBEC5817
Ciphet-text after 3 rounds:     89C5F20861099F9EEB73639A0BE8D3EA
Ciphet-text after 4 rounds:     F1D452604C3119F03577B2790FF6A34D
Ciphet-text after 5 rounds:     92A6C59992FDB6A8A8452D28E3764214
Ciphet-text after 6 rounds:     A0DAD27C43FE5684D8349F6EFA113858
Ciphet-text after 7 rounds:     D71F17D9383AFE1FF08EA6657040EED5
Ciphet-text after 8 rounds:     512E95839F96762F9C544D00A66FFA17
Ciphet-text after 9 rounds:     FF75BAA2661F246560821DBD47D73AB2
final ciphertext:               FF75BAA2661F246560821DBD47D73AB2
```

Figure 3-9 Implementation of AES encryption using ESP32 with LoRa module

Time for evolution of a species $\approx 2^{20}$ years
Age of the earth $= 2^{32}$ years
Bits in a terabyte drive $= 2^{43}$
Cells in the human body $\geq 2^{46}$
Amount of water in the Great Lakes $= 2^{53}$ gallons
Estimate of atoms in observable universe $\approx 2^{265}$

Figure 3-10 A hint about large numbers to understand the size of keyspace having 2^128 elements

```
pre-encrypt(plain_text):        0123456789ABCDEF0123456789ABCDEF
Ciphertext without MAC is:      F9F026E7CD825F05A559FE74E4656FE9
Decrypted Plaintext:            0123456789ABCDEF0123456789ABCDEF
ciphertext with MAC:            F9F026E7CD825F05A559FE74E4656FE9410DFC5C95DFF8CE
Received_MAC:                   410DFC5C95DFF8CE
Calculated_MAC:                 410DFC5C95DFF8CE
MAC verification status:        Message is authentic.
Verified decrypted message is: 0123456789ABCDEF0123456789ABCDEF
```

Figure 3-11 Implementation of AES algorithm with MAC address

The data rate for two-way communication between two ESP32 modules depends upon the spreading factor of the LoRa communication. Spreading factor is basically the frequency hopping, and its range could be configured by setting the spreading factor from 7-12 for LoRa communication. The data rate is inversely proportional to the spreading factor. However, a higher spreading factor gives better range. Figure 3.12 and 3.13 show the graph between the time elapsed on the sender side and receiver side. The time sampled on sender side for the data readings includes the time taken in message encrypting and sending and for the receiver side, the time sample includes the message air time, time taken in message verification and decryption as well. In these graphs, the relation between spreading factor and the data rate can be seen vividly.

Figure 3.14 shows the impact of spreading factor on the range. The range of LoRa coverage is directly proportional to the spreading factor. From the given figure, it can be seen that when spreading factor is seven the number of messages received and sent is equal and it indicates three advantages

- No data loss
- Authentic messages
- No delay in messages

On the other side, when the spreading factor is twelve, the number of sent messages is much higher than the number of received messages, and it has two drawbacks

- Messages are prone to EMI/data loss
- Messages are delayed due to a longer message on air time.

At the same time when the range testing was done, the higher spreading factor gave better range coverage. For SF equal to twelve there was a successful communication link up to 4km (low range due to obstacles) despite minor obstacles as shown in figure 3.15. When the SF was set equal to seven, the communication link was not good for the same range and similar obstacles.

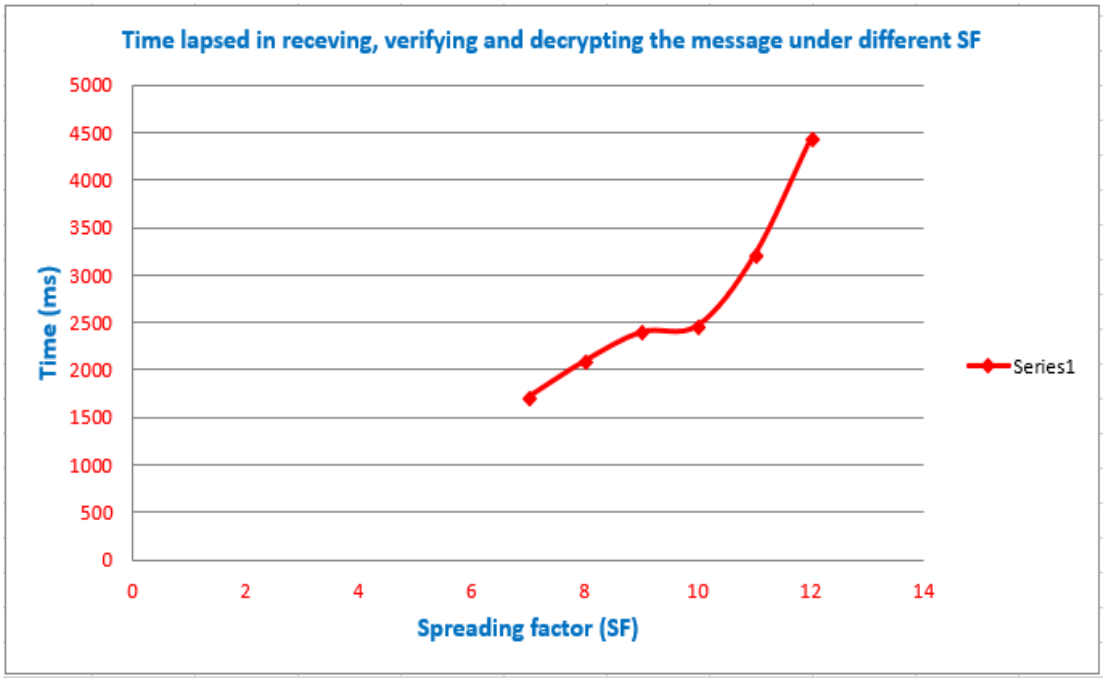


Figure 3-12 Time lapsed in message receiving, verifying and decrypting under different SF

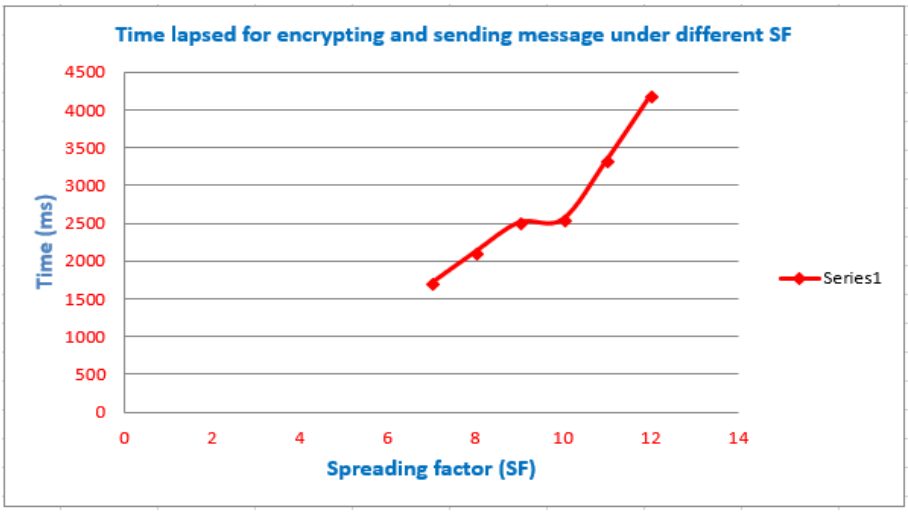


Figure 3-13 Time lapsed for message encrypting and sending under different SF

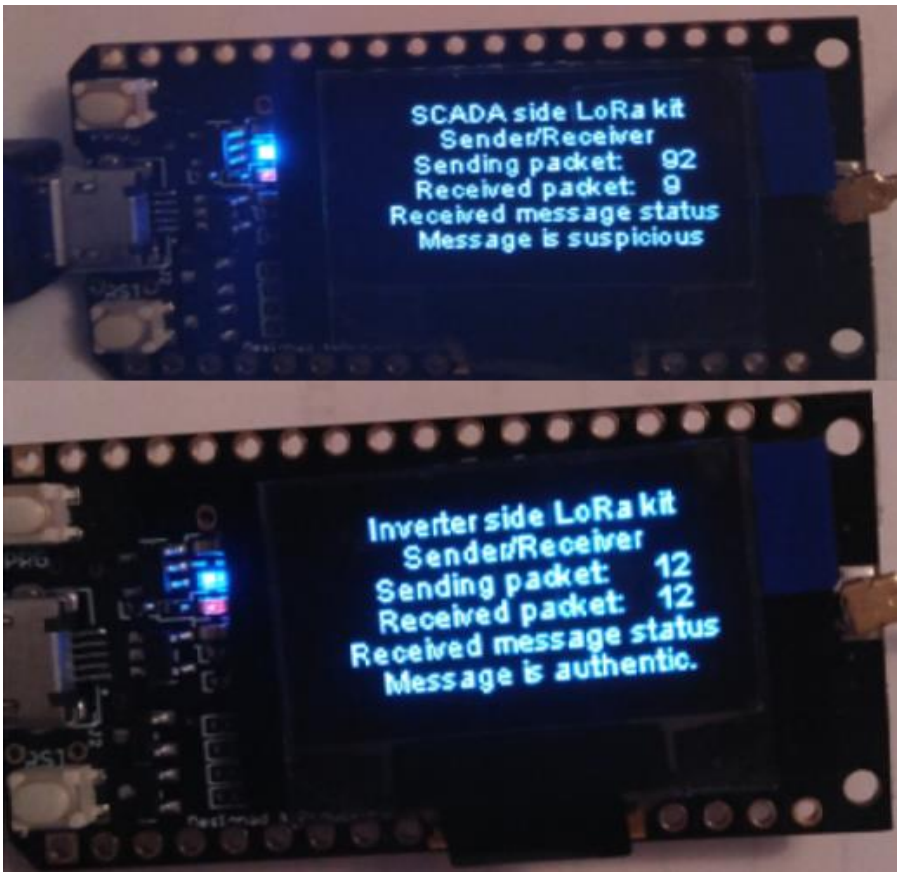


Figure 3-14 Difference of data rate and message authenticity under different SF

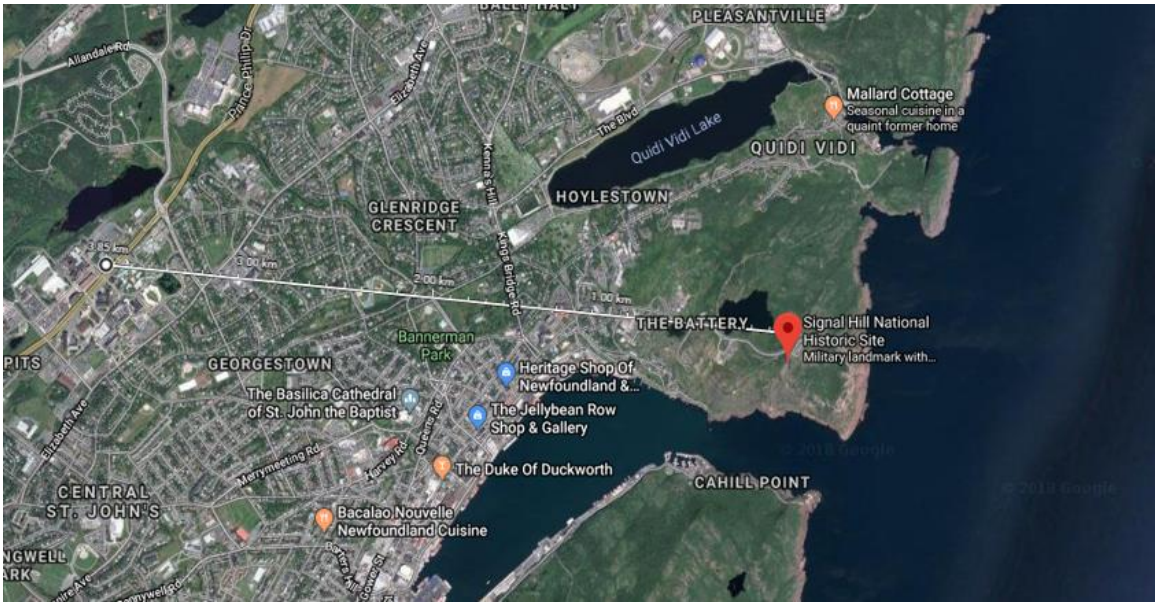


Figure 3-15 Bi-directional communication range testing of ESP32 with LoRa module

3.4 Conclusion

The implementation of the AES algorithm on ESP32 gives a secure, low cost (of not more than $50 for two-way communication link), low power (with power consumption of 200mW/unit) and authentic communication setup. Its implementation with MAC address, addresses the all requirements of the SCADA system for remote micro-grids. Further, its on-site range is up to 15km and off-site up-to 5km which gives much better coverage as compared to other conventional wireless communication systems based upon Zigbee, SigFox, WiFi and others like these. Therefore, its implementation on a large scale will be economically viable, and also completely secure from all kind of eavesdropper.

Chapter # 4 Data Logging Using Different Gateways and Mesh-Network Implementation to Improve LoRa Range

4.1 Introduction

In this chapter local and remote data logging has been achieved along with the implementation of Mesh-network to improve the LoRa range.

The ESP32 board has only a single channel SPI bus which does not let the simultaneous configuration of LoRa communication and the configuration of an SD card to log data locally. Therefore, another ESP32 board was used to collect data through Rx/Tx and store on a separate SD card. Remote data logging has been achieved by uploading data to a server. Its block diagram has been shown in fig-4.1. In that, remote nodes send data to the gateway to upload it to the cloud for remote access. To upload the data to a server requires a gateway, for that ESP32 board, was programmed to work as a gateway but, it affected the functionalities of ESP32. Further, it slowed down its processing time due to which a dedicated Dragino-yun based gateway was tried for uploading data to a server, and it worked fine.

Finally, to improve the LoRa communication range, a mesh-network was planned and implemented. The implementation of mesh-network significantly improved the LoRa range. The results and detailed implementations have been explained in the coming sections.

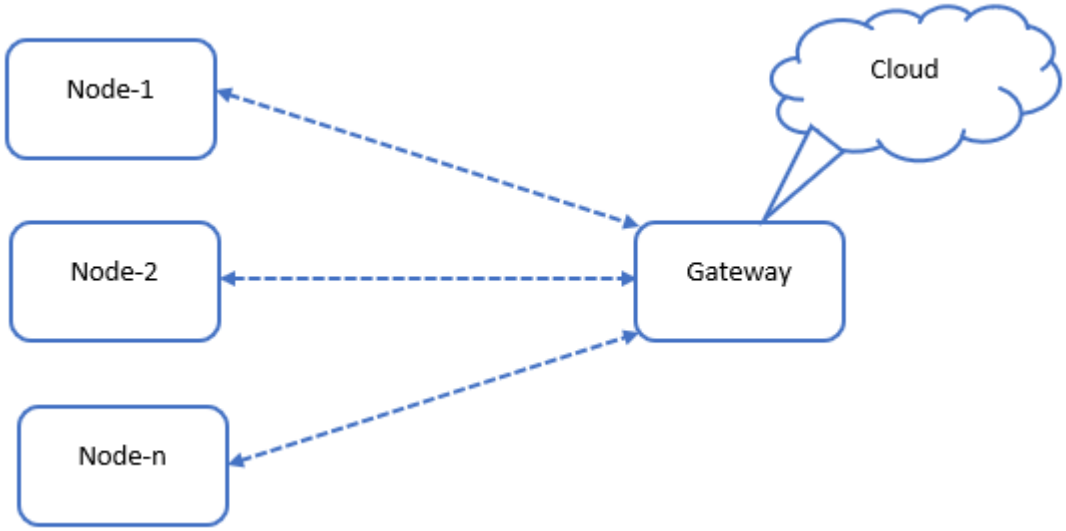


Figure 4-1 System structure for remote data-logging

4.2 Local Data Logging

The feature of local data logging was added into this system in order to have a local-data backup mechanism and to avoid data loss due to any accidental failure in the communication system. The data which is supposed to be coming from a local inverter/wind turbine is time-stamped and is stored after applying the AES encryption. Subsequently, the received data after extracting sender/receiver identity passwords and MAC is decrypted and verified to ensure that the received message is authentic. Finally, the data is time-stamped and stored in a separate received data file in CSV format. For the

local data storage, the configured SD card console has been shown in fig-4.2, and the logged data depends upon storage size and received data rate.

```
SD Card Size: 7580MB
Listing directory: /
  FILE: /test.txt  SIZE: 1048576
  FILE: /foo.txt  SIZE: 13
  DIR : /my_new_directory_1
Creating Dir: /mydir
Dir created
Listing directory: /
  FILE: /test.txt  SIZE: 1048576
  FILE: /foo.txt  SIZE: 13
  DIR : /my_new_directory_1
  DIR : /mydir
Removing Dir: /mydir
Dir removed
Listing directory: /
  FILE: /test.txt  SIZE: 1048576
  FILE: /foo.txt  SIZE: 13
  DIR : /my_new_directory_1
Listing directory: /my_new_directory_1
Writing file: /bytes.txt
File written
Appending to file: /bytes.txt
Message appended
1048576 bytes read for 2905 ms
1048576 bytes written for 4913 ms
Total space: 7563MB
```

Figure 4-2 Configuring SD card for local-data logging

4.3 ESP-32 vs Dragino Gateways

The collected data is uploaded to a server for analysis and storage. Two different gateways, based upon ESP32 and dragino were tried to upload the data to a server but, both had certain limitations. The programming and setting of an ESP32 board to work as a gateway is relatively difficult because it is to be configured as a gateway through coding while a

dedicated dragino gateway is already available in the market with complete configuration and is more user-friendly. On the other hand, an ESP32 based gateway is much more cost effective and power efficient. It hardly consumes 230-300mW of power [74].

Figure 4.3 shows the successful configurations of the ESP32 board as a gateway. The configuration and setting of an ESP32 board to operate as a gateway have been shown in fig-4.4 and fig 4.5. In fig-4.4 is the information about the spreading factor used for data uploading and fig-4.5 shows the general settings about which feature to be turned on or off. The Arduino program for its configuration has been given in the appendix. The major limitation of using ESP32 as a gateway is that when it is used as a gateway and to encrypt/decrypt data simultaneously, its data uploading rate drastically decreases. It is vivid from the fig-4.6 and fig-4.7. Fig-4.6 shows the message sent to the server when ESP32 was being used to decrypt the received data and then to send to the server. It can be seen that there is a delay sending messages to the server due to the board limitations and it causes a lot of data loss. It could be used as a gateway for the applications where the data rate is low but, could not be used for the SCADA system. Further, it does not have ethernet port and connection is through WiFi. Therefore, if intended to upload decrypted data to the server it will lose the security. Later in fig-4.7 the console window of The Things Network has been shown in which data is being uploaded to the server. It can be seen that this window does not show any delay because data was being uploaded without decrypting.

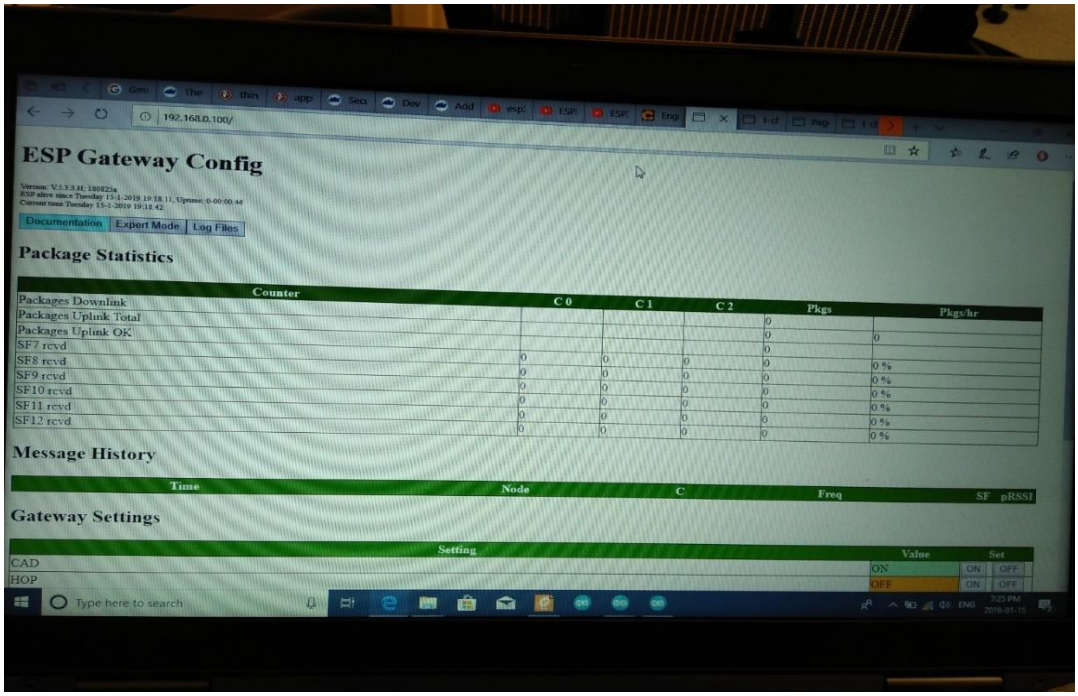


Figure 4-3 Setting ESP32 as a gateway

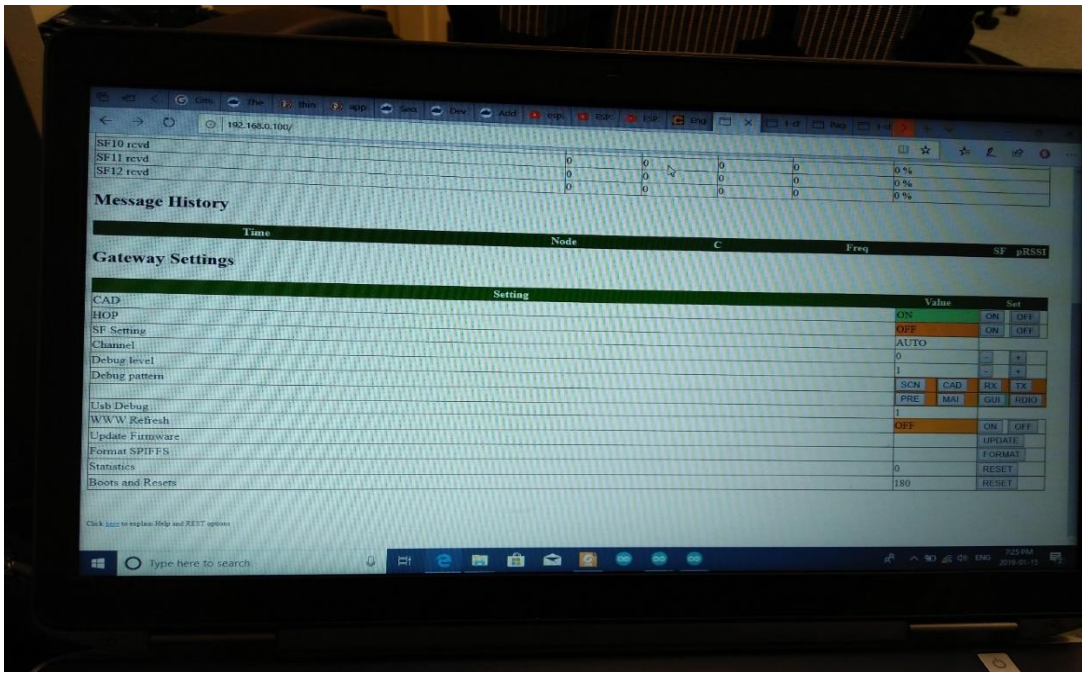


Figure 4-4 Setting ESP32 as a gateway

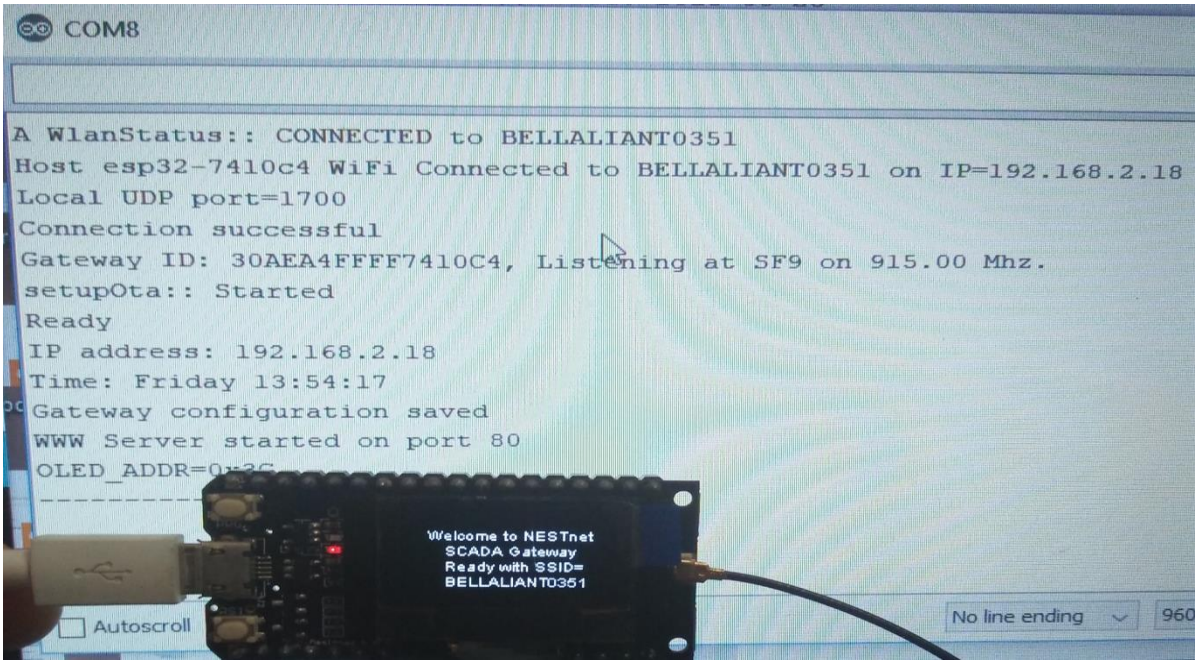


Figure 4-5 Configuration results of ESP32 as a gateway

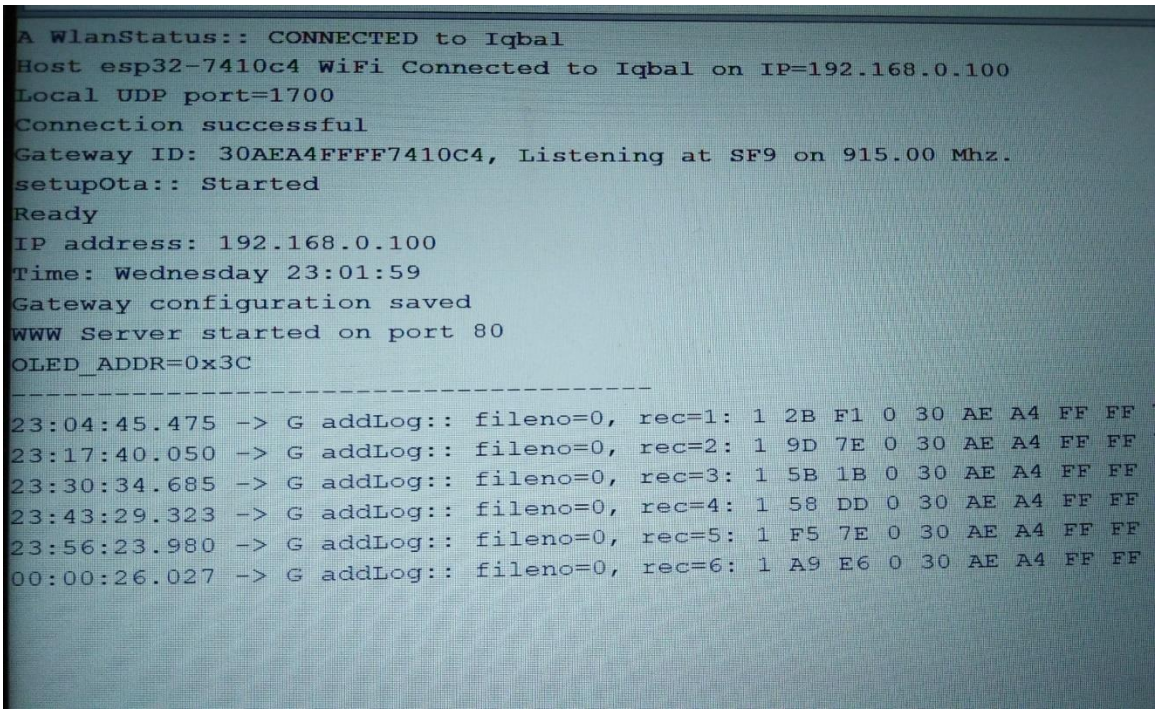


Figure 4-6 ESP32 gateway sending messages with AES encryption

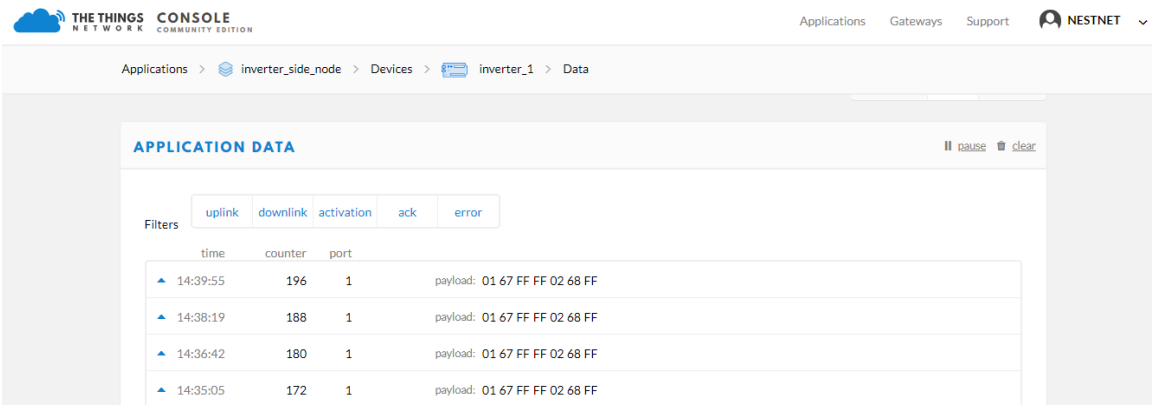


Figure 4-7 ESP32 gateway uploading data to the Things Network without encryption

Fig-4.8 shows the dragino gateway, used for uploading data to the ThingSpeak server. Its successful configuration and real-time load have been shown in fig-4.9 and fig-4.10. It has WAN and LAN ethernet ports and consumes 12W power under loading conditions in active mode and requires 12V DC for proper functioning. Dragino compensates for this excess power consumption in terms of many other features. For example, it can serve up to 8 nodes simultaneously by communicating with each node at a different frequency [74], while ESP32 can support only three such nodes simultaneously. Unlike ESP32, it does not face any processing issue due to encryption/decryption which makes it a better choice for reliable operations.

Figure 4.11 gives the graphical view of data uploaded on the ThingSpeak server using a dragino gateway. Data is uploaded to the ThingSpeak server once in every fifteen seconds. ThingSpeak server gives up to eight free data fields for a single account. If the usage exceeds that then it requires a premium account. Here in the prototype setup, three fields were used for data logging.

Figure 4-8 A dedicated dragino-yun LoRa gateway

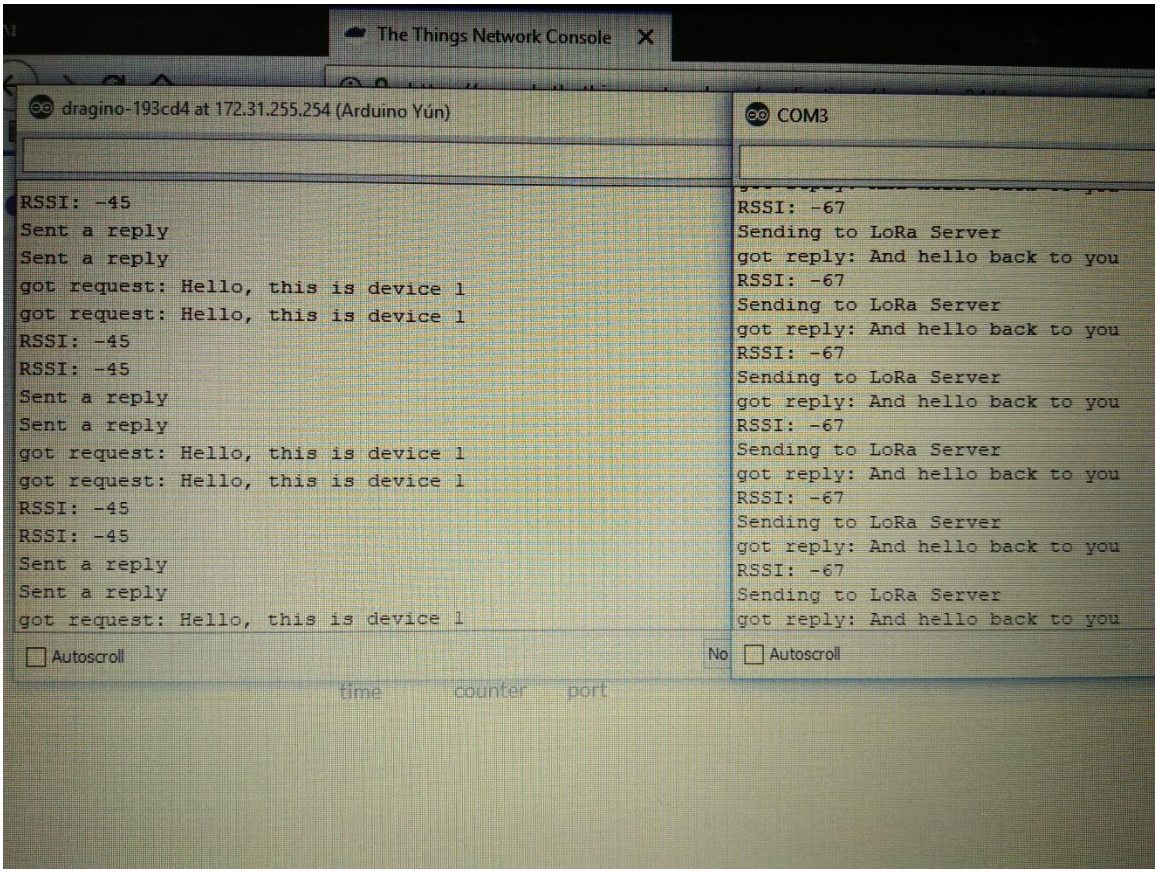


Figure 4-9 Successful configuration results of dragino-yun working as a gateway

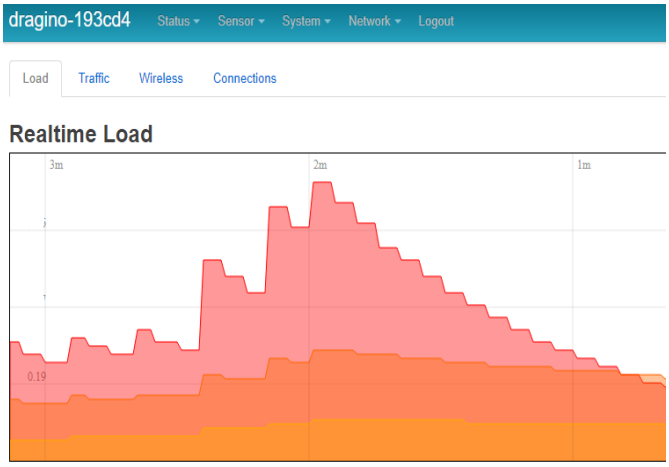


Figure 4-10 Dragino gateway real-time data load

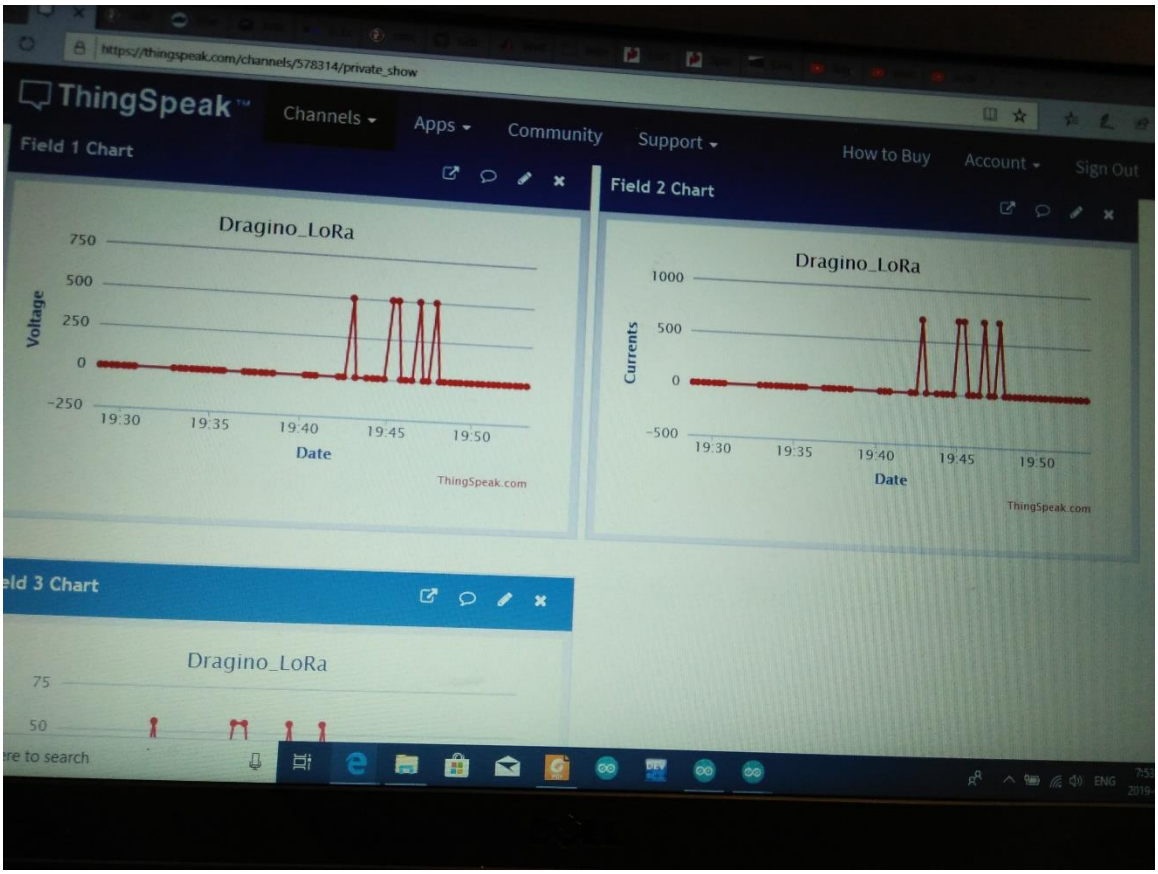


Figure 4-11 Uploading data to the ThingSpeak server

4.4 Implementation of Mesh Network and Range Testing

The LoRa range was tested deploying one ESP32-LoRa at Memorial University and taking other EP32-LoRa to the Signal Hill as shown in fig-4.12. This setup supported a noise and error-free communication for the distance of 3.85km. Although, its range is usually obstacle dependent and during another testing it was observed that if the transmitter is at ground floor in the house window and a receiver is taken outside in neighbouring streets, then the communication range drastically goes down and they can communicate only up to the distance of 500-700m (with obstacles). Later, a mesh-network topology was implemented to address this issue, and it gave significant improvements in range.

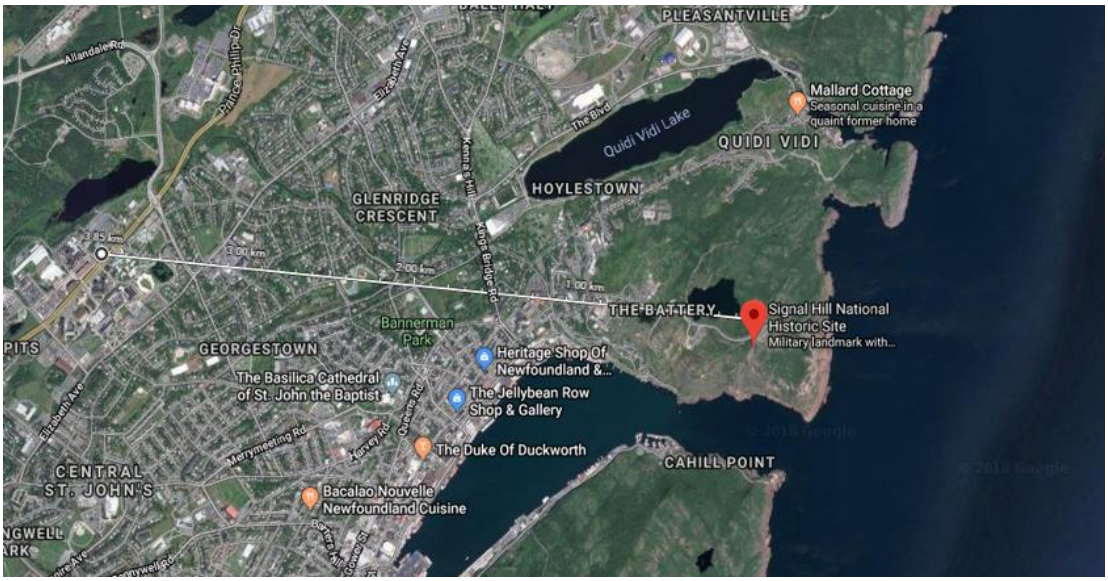


Figure 4-12 ESP32 LoRa range testing

Each ESP32-Lora module of the network was assigned a unique identity code to implement a mesh topology and was also fed with the directory of all other units' identity code directory. Before sending a message, the message is encrypted, and MAC address is added

to ciphertext string. When a ciphertext string with MAC is ready to be sent, the sender adds the target node identity code and its identification code in the string. A simple flow chart of the processing steps at the receiver end has been shown in fig-4.13. The main steps of this flow are:

a) Receive message

b) Compare received ID and with node ID

c) If true, then verify MAC address, decrypt and execute

d) If false, then tranmit message forward

The receiver comes out of the sleep mode and receives the message. Whenever there is a message the receiver receives the message and parses it into three parts

a) sender identity code

b) the targeted receiver identity code

c) message packet with data and MAC information

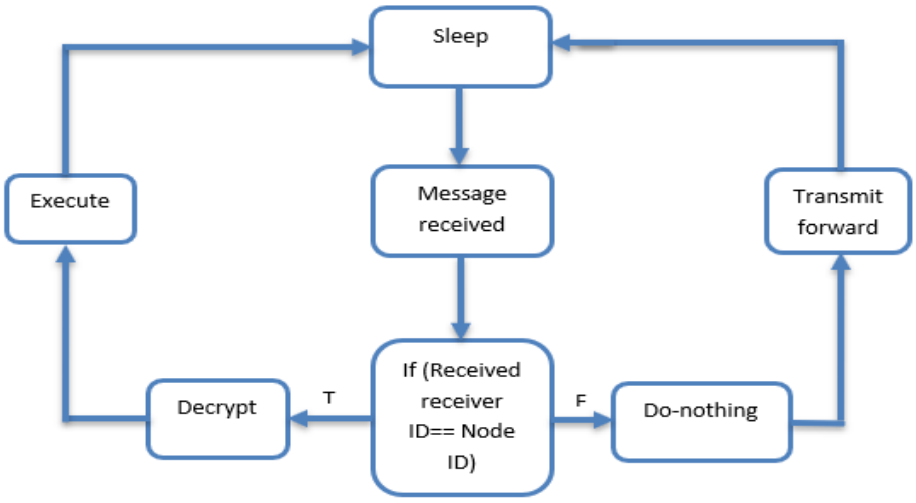


Figure 4-13 Process flow chart for an intermediate level node

In the first step, the ID of the node is compared with the targeted receiver ID. If they are equal, this means that the node is the targeted receiver node and then the received message packet is further parsed into an encrypted message and MAC to process further. If the receiver ID is not equal to the targeted receiver ID, it shows that the message is for any other node then, the message is again packed in a single string as it was received and is forwarded to the other nodes lying in the range.

In this way, if a message is sent from the central control unit for a node which does not lie in the range of that unit, then a node in the vicinity of the sender will receive that message and will forward to the next nearby nodes.

A complete system flow chart has been shown in the fig-4.14 to make it easy to understand. In this figure, a node of level-2 lies out of the range of the central node and a node of level-1 acts as a bridge for two-way communication between the control node and level-2 nodes. After implementing mesh topology, its range was tested for two levels of nodes, and a significant improvement was observed. The nodes whose range was limited to 900-1000m in the street with obstacles achieved another 500m in their coverage area, and its results can be seen in fig-4.15. In this way, the range which was only obstacle dependent became the function of the number of levels between the sender and the receiver as well.

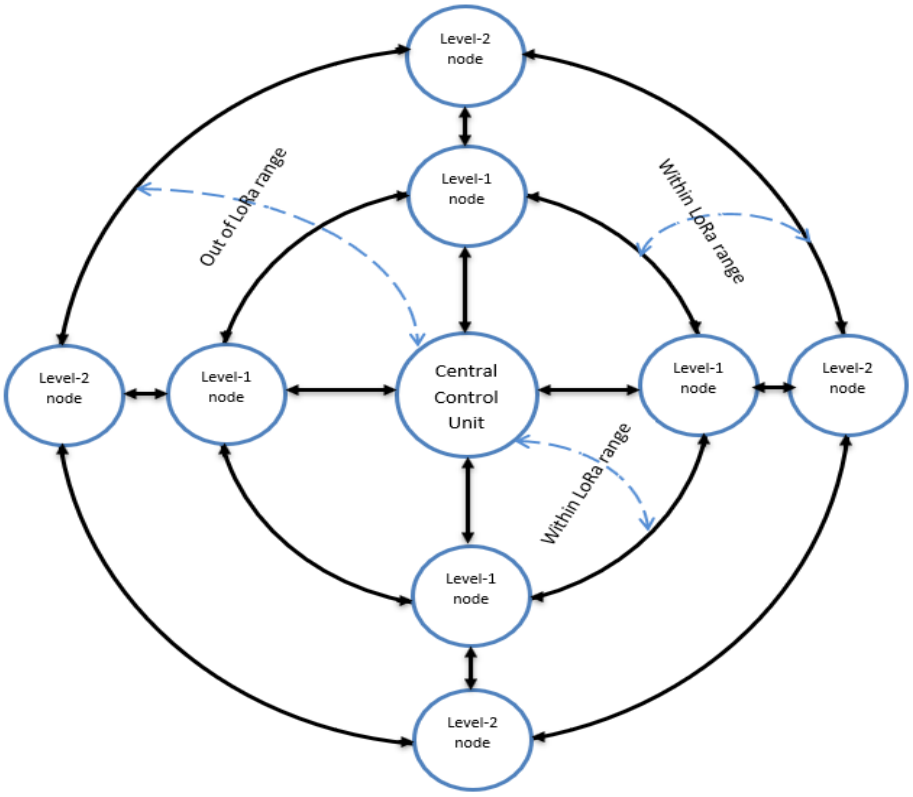


Figure 4-14 Mesh network for improved LoRa range

Figure 4-15 Range testing in obstacle dense area after implementing a mesh network

4.5 Complete System Flow Diagram

Figure 4.16 shows a complete system flow diagram for a prototype micro-grid network. It has distributed generation shown as two different wind turbines and two solar panels which represent the distributed power generation units. Each power generation unit has sensors for certain measurements (current, voltage, power etc.), and is also associated with ESP32 based LoRa module which acts as a remote-end-device. Data is collected from the sensors installed on RED and is serially transmitted to the local ESP32 modules. The ESP32 encrypts the data applying AES encryption algorithm and calculates the 64bit unique MAC for the message and adds to the ciphertext string. After encryption and MAC addition, one copy of the data is stored on local SD card for local data-backup, and the other copy is transmitted to the SCADA side through LoRa communication after the addition of sender and receiver node IDs. LoRa communication gives it better range through the mesh-network structure and transmits to the SCADA unit. On SCADA side, the received message is parsed into sender/receiver IDs, MAC address and the ciphertext of the message. Before decryption and further processing first, the authenticity and integrity of the message are verified by comparing node IDs and calculated/received MAC addresses. After ensuring the message authenticity, the message is decrypted and processed. For remote data logging, a copy of the data is sent to the gateway (ESP32/Dragino-yun) to upload data on the web. In this way, a secure and authentic communication system for remote micro-grids has been achieved, and data has been logged on a server to view, analyze and store data remotely.

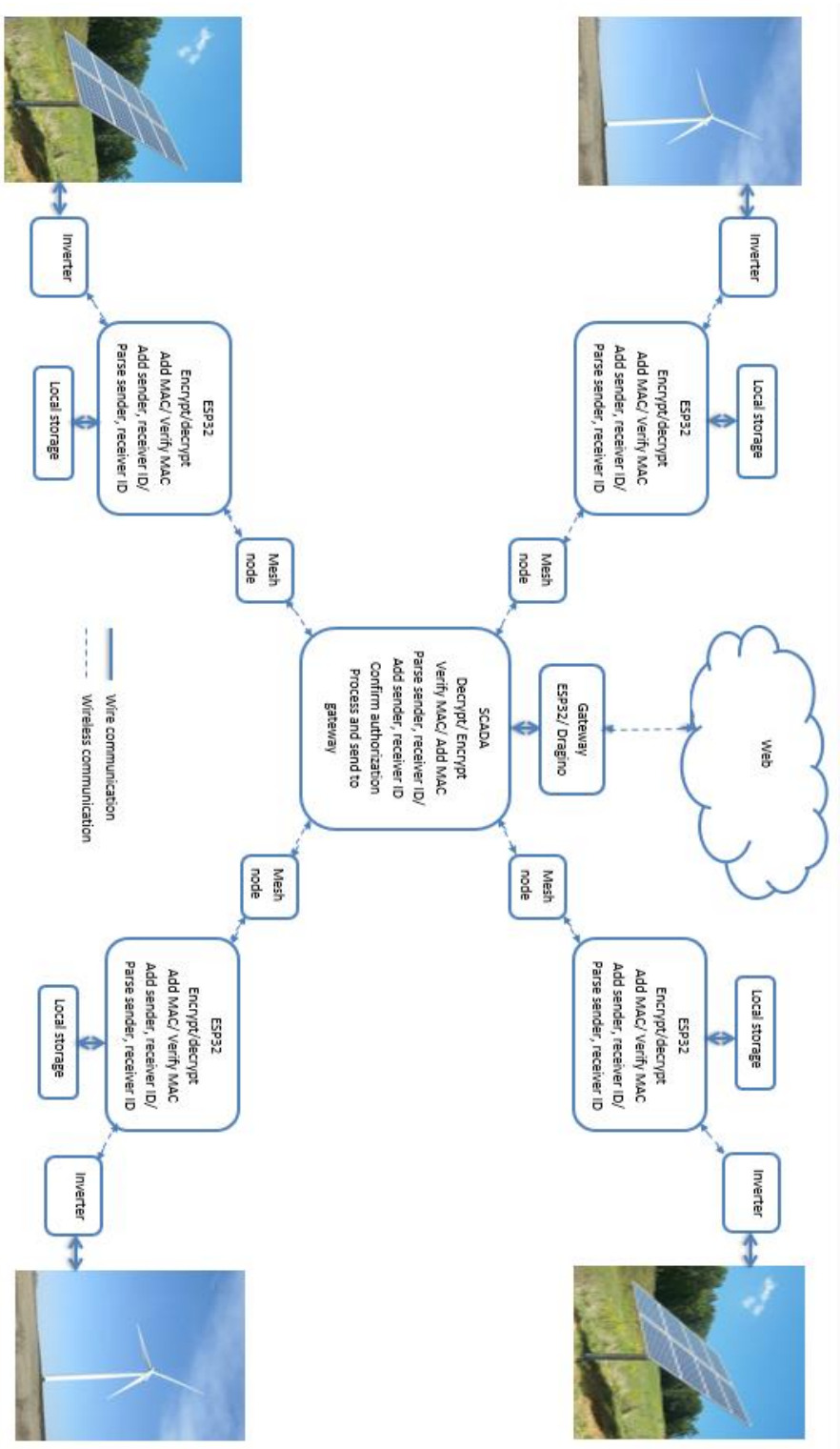


Figure 4-16 Complete System diagram

4.6 Conclusion

Here, two different gateways were programmed and configured to work as a gateway to upload data to the server. It was seen that ESP32 based gateway has less power consumption but also has many limitations, slow processing, delay in data uploading and loss of the data. Due to which Dragino-yun was tried to upload data to the server, and it worked nicely without any data loss. For local data logging, an SD card was configured, and data was stored on that but, that requires an additional ESP32 board due to the limitations of board structure (overlapping of communication buses). Finally, to meet applications requirements the dragino-yun based gateway was selected to upload data to the server.

A successful mesh-network structure was implemented to improve the range of LoRa communication and to collect data from the remote end devices. It significantly improved the LoRa range (5km off-site and 15km on-site) and tackled with the challenge of data collection from scattered remote end devices.

In this way a secure communication with remote data logging was achieved with cost not more than $82 with additional $26 for every new node into the system. Single ESP32 consumes power of 200mW and the Dragino gateway consumes 4.5W and overall power consumption was not more than 4.7W with additional 200mW for every new node. In this way, system not only meets the requirements but also gives low cost and low power setup.

Chapter # 5 Radio-set Based System Topologies for Longer Range Setup

5.1 Introduction

After the successful implementation of a secure communication system for remote microgrids, and improved LoRa range by implementing mesh-network, the idea came about extending its range beyond 30km to enhance the circle of applications. To achieve this goal, a radio set was used on both sides. One on the SCADA side and other on the remote end with the local data collector. The local data collector collects the data from distributed Lora units, and stores data for local backup. After applying encryption and other data security steps discussed in previous chapters, the data is serially transferred to the MAX232 (driver for serial communication between radio-set and controller) to feed it to the radio set. The radio-set takes 12V DC due to which MAX232, a serial communication driver has been used to provide common ground for different voltage level signals. The radio-set transmits the received data improving transmission range. A single layer of LoRa modules (without mesh-network) has the range of 15km, and the radio-set has the range of up to 25km. In this way, the integration of LoRa module and radio-set give transmission

range of more than 40km without the use of any mesh-network or repeater for radio-sets. Here, three different system topologies

1) ESP32 with LoRa based communication system
2) Remote data logging using ESP32 based gateway
3) Data logging on a web-server using Dragino-yun

have been tried using radio-set.

5.2 System Structure-I

A simple diagram of the system topology applied at this stage has been shown in fig 5.1. In this system structure, the data collected from remote sensors is fed to the dedicated ESP32 module. The ESSP32 with LoRa, repeats the same steps of encryption, MAC calculation, and the addition of sender/receiver IDs as was done in system structure explained in previous chapters. After repeating those steps, the ESP32 data collector does not transmit the message over LoRa communication; instead serially transmits to the radio-set through MAX232. The radio set boosts the range and sends it to the far away located other radio-set. On the receiver end, the radio-set receives the data and sends it to the USR-TCP-232 serial to ethernet converter. The output of this converter goes to the PC where the message is further processed. The results of data collected through this kind of system-scheme have been shown in fig 5.2. The configuration of USR-TCP-232 has been explained in the coming sections.

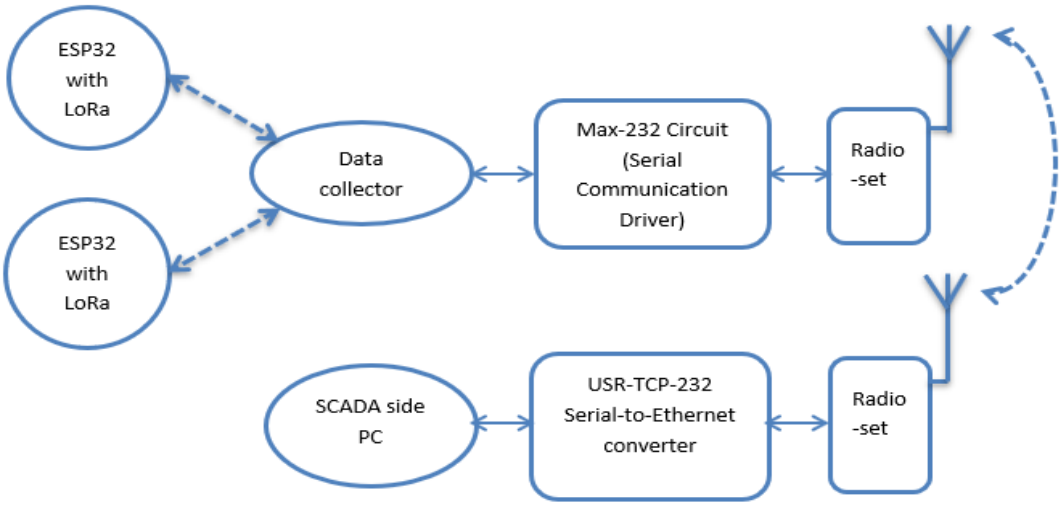


Figure 5-1 System structure for topology I with broader range due to radio-set

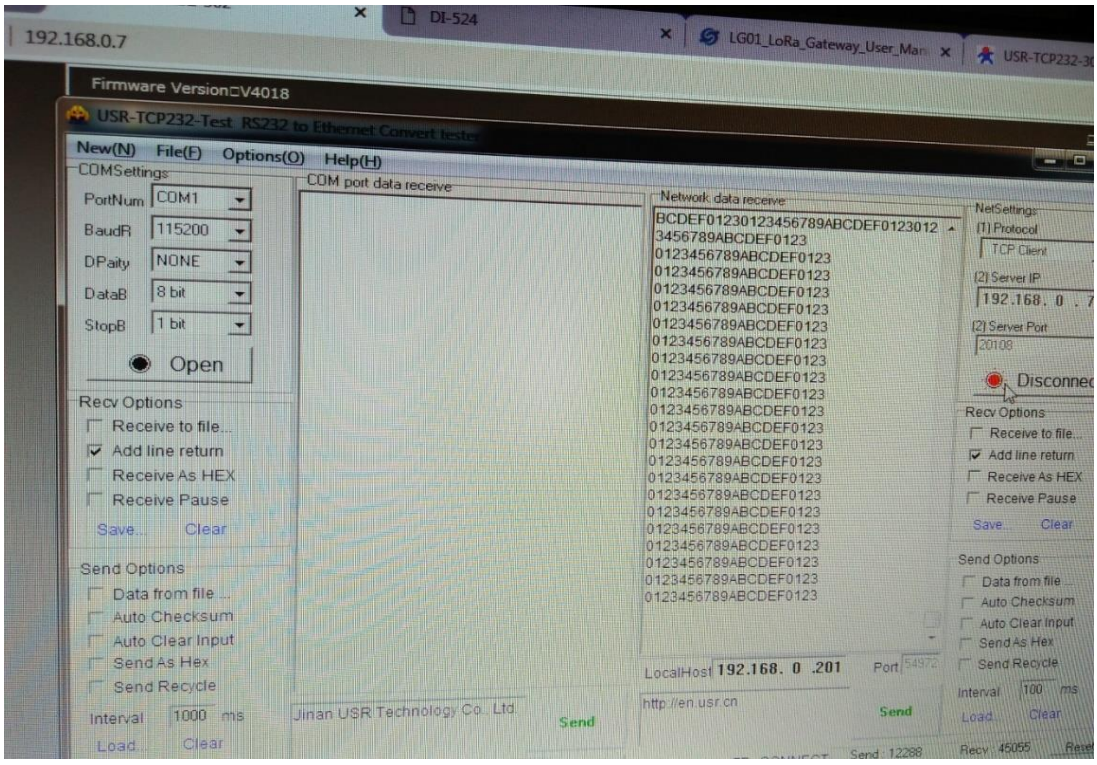


Figure 5-2 Data logging by implementing system structure I

5.3 Hardware Setup and Interfacing

5.3.1 RA30H1317M

The circuit of the mobile radio set used in the project has been shown in fig 5.3. This mobile radio requires 12.5V DC and consumes 25W power to transmit messages over up to 30km in the frequency band of 135MHz-170MHz. It was chosen because it has the following features

- Efficiency >40%
- Broadband frequency range 135-170MHz
- Pout=30Watt
- 2nd harmonic power = -35dB

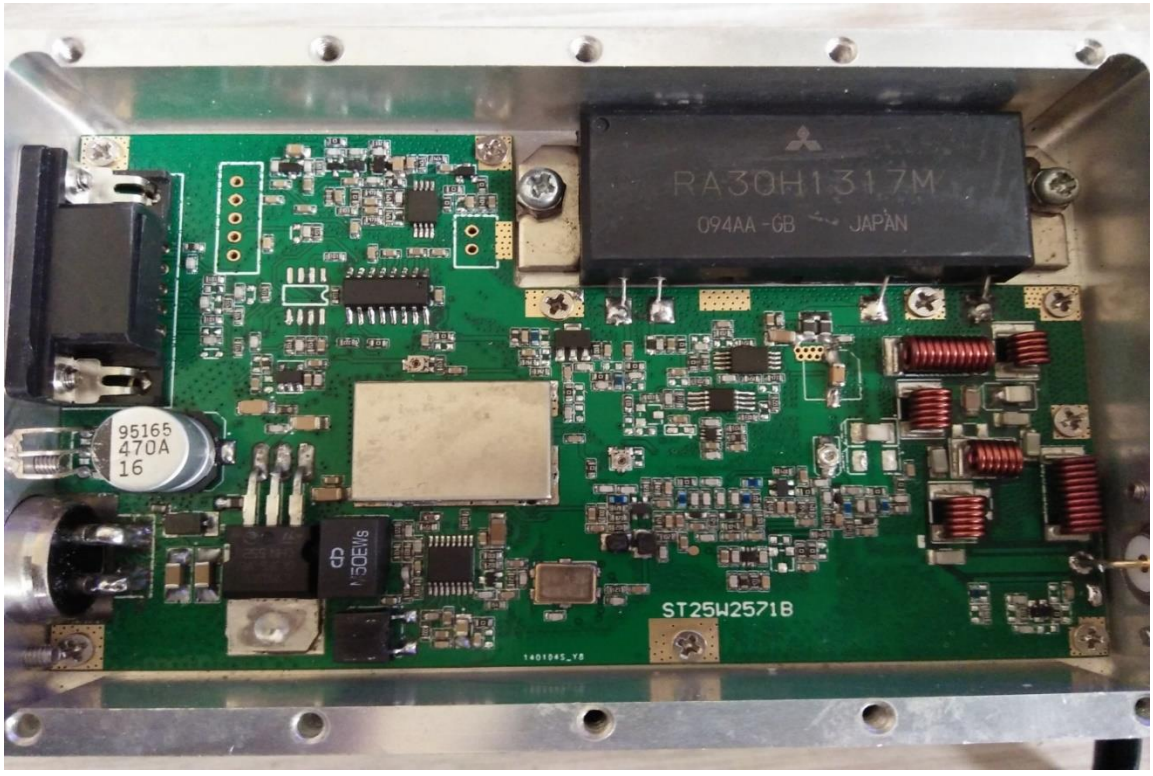


Figure 5-3 RA30H1317M based radio-set board used in this project

5.3.2 DC Power Supply LRS-150-12

Radi0-set RA30H1317M requires a power supply which can give protection against all kind of surges and minimum voltage regulation. For message sending and receiving radio-set draws very high current (up to 6Amp) than its normal energized state (110mA) therefore, a DC power supply must also be able to deliver that current without voltage dip. The DC power supply LRS-150-12 was chosen because its rated current is 12.5A, and at full load the voltage regulation is not more than 1%. Its further specifications are given in table 5.1, and its circuit has been shown in fig 5.4.

Table 5-1 Specifications of the power source

DC Voltage	12V
Rated current	12.5A
Current range	0~12.5A
Rated power	150W
Ripple & noise (max)	150m Vp-p
Output voltage adj. range	10.2~13.8V
Voltage regulation	+-1%
Input voltage	85~132VAC

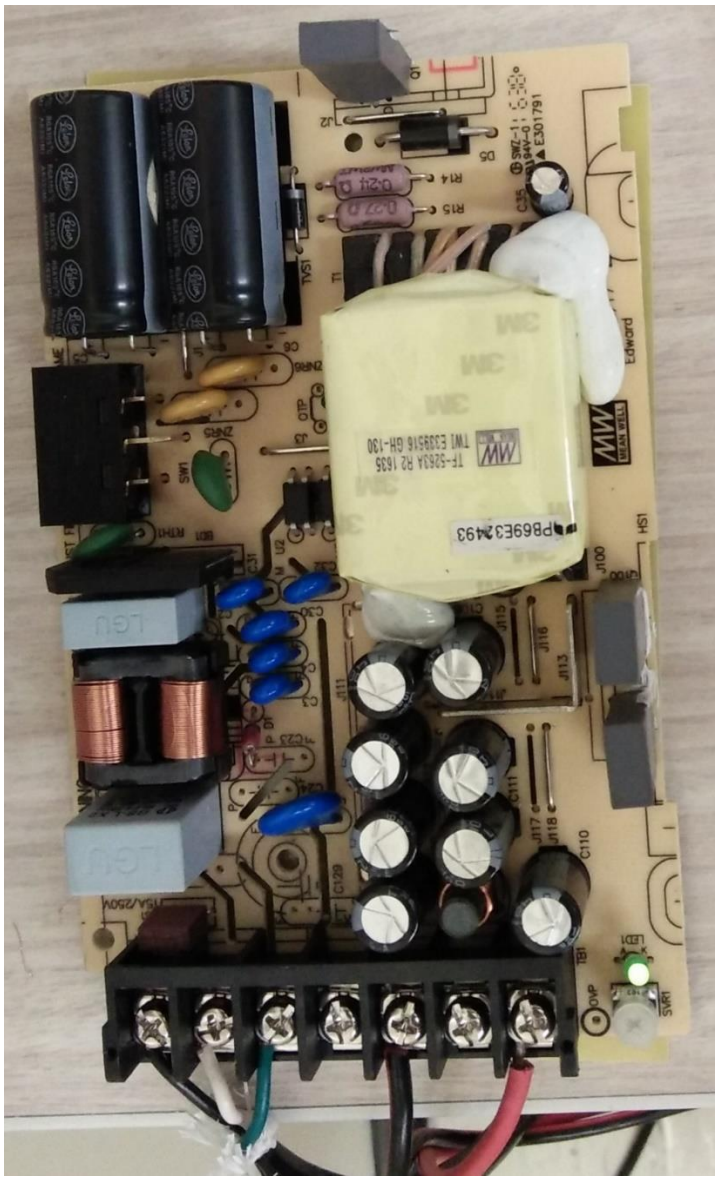


Figure 5-4 12V DC power supply to power-up the radio-set and protecting against surges

5.3.3 Building MAX 232 circuit

The controller boards used in this project are ESP32 with LoRa module. Its operating voltage range is 3.3-5V, and the radio-set which receives the message from ESP32 through serial Rx/Tx pins operates at 12V DC. Therefore, to provide common grounds for different

voltage signals and to protect the controller from any unwanted surge requires a driver. MAX232 was used to protect the boards and to provide common grounds. Its circuit was built as shown in fig 5.5 according to its data-sheet.

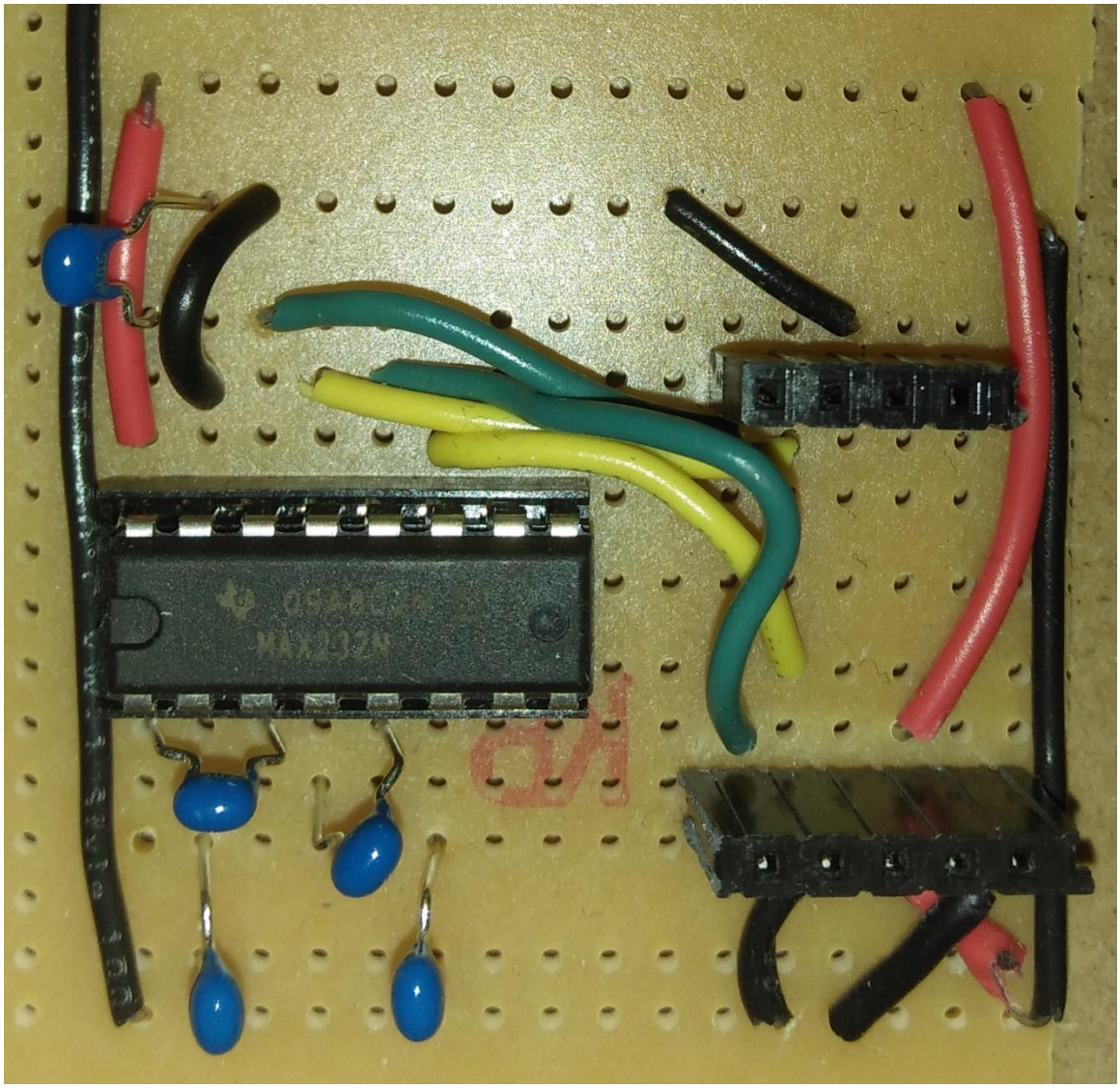


Figure 5-5 MAX232 driver circuit built to protect the controller from power surges and to provide common ground for signals

5.1.1 Setting USR-TCP232

On the receiver side, the received message by radio-set is sent to the PC through USR-TCP232 converter shown in fig 5.6. It works on 5V DC and converts DB-9 signal into ethernet. It has inbuilt TCP server and requires complete settings. After logging in to its setting page it requires setting in two domains. First, its serial port setting is to be done as shown in fig 5.7 mainly focusing on three categories

Baud rate:- It could be set according to the other system baud rate for proper communication without any delay or information loss. The system baud rate of 9600 was used in this communication setup.

Local port number:- USR-TCP232 requires its local and remote port numbers to be set. When it is being used as a TCP server, then its local port number is to be assigned a specific ID number from 0-65535, and when it is used as a TCP client, then its remote port number is to be set. Based upon the port number, it interfaces with the PC, and it should also match with the PC port number (explained in the next section) to log data.

TCP Server:- Lastly, its work mode is set by setting whether is it to work as a TCP server or TCP client. Here it has been used as a TCP server, and its setting has been done accordingly as shown in fig 5.7.

After its serial port setting, its internal registration setting is to be done by setting packet type and direction. Here communication between different units was in ASCII due to which it was set in ASCII mode shown in fig 5.8; otherwise it could be set into Binary Coded Decimal (BCD), hex or binary.

Figure 5-6 USR-TCP-232 used in the project for serial to ethernet conversion

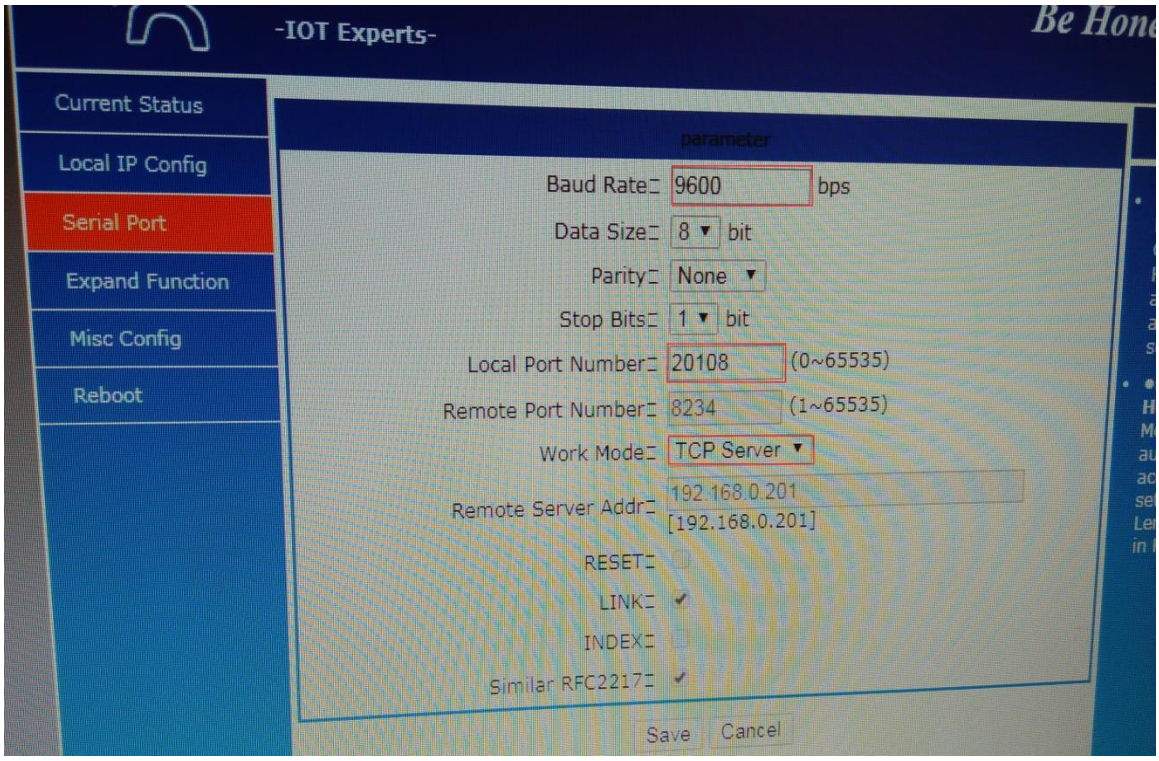


Figure 5-7 USR-TCP-232 Serial port setting

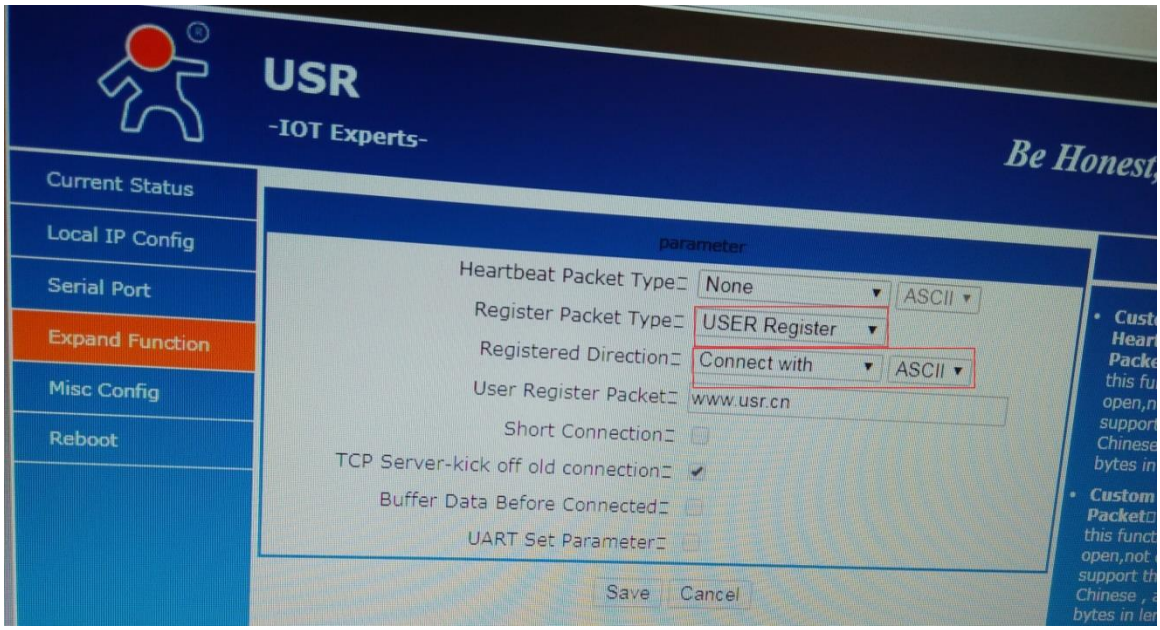


Figure 5-8 Internal registration setting of USR-TCP-232

5.3.5 Ethernet and USR-TCP232 setting

After the completion of USR-TCP232-302 setting, the PC ethernet port setting was done as shown in fig 5.9. To start logging data or sending messages from PC to remote end devices, the USR-TCP-232-test software proved very helpful. Its configuration setting has been shown in fig 5.10. For its successful setting three things are important to be considered. It should be set into the TCP Client (Server) mode because USR-TCP232 was configured in TCP Server (Client) mode. Secondly, its server IP must be the same which was set for TCP-232 device. Finally, its server port must be the same as was set in the local port ID of the TCP server. After these settings are correctly done, the connect option connects the serial channel for communication and data logging (message sending) starts. Further, this software gives many additional features, e.g. logging received data into a .text

file, reading messages from .text file to send over this channel in a cyclic way. The time interval between the outgoing messages can be set easily as shown in the fig 5.10.

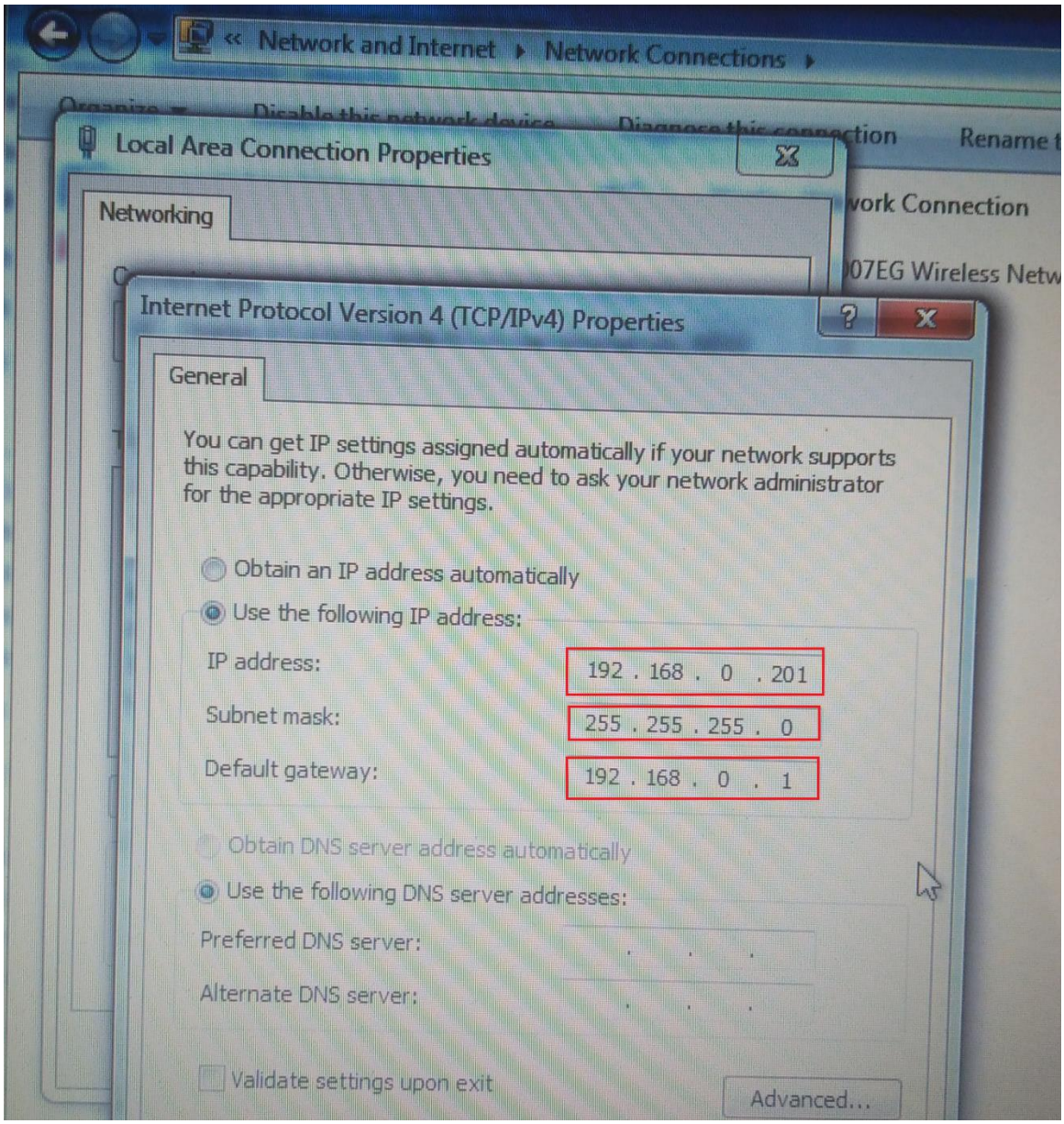


Figure 5-9 PC ethernet setting to interface with USR-TCP-232

Results

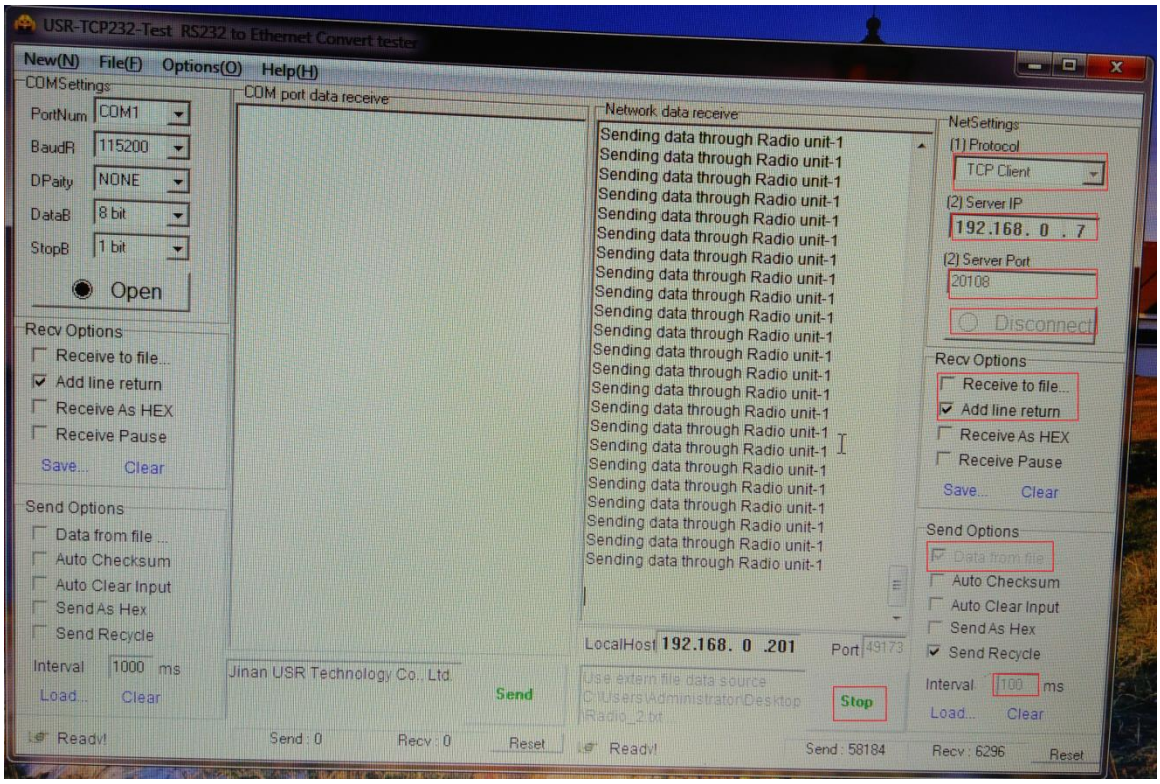


Figure 5-10 Results and setting of USR-TCP232 test software

5.4 System Structure-II

In this system structure, the remote-end sensor side system structure is almost the same as the system structure-I but, the SCADA side has been improved significantly. In the previous system, the Radio-set was serially connected with the PC through USR-TCP232-302, and data was being logged, but its drawback was that it reserves a whole PC running round the clock to log the data. In system-II, as shown in the fig 5.11, a driver of serial communication (MAX-232) collects the serial data from Radio set and feeds it to a tiny controller ESP-32 running as a gateway. It uploads data to the server through the wifi and eliminates the need of a PC.

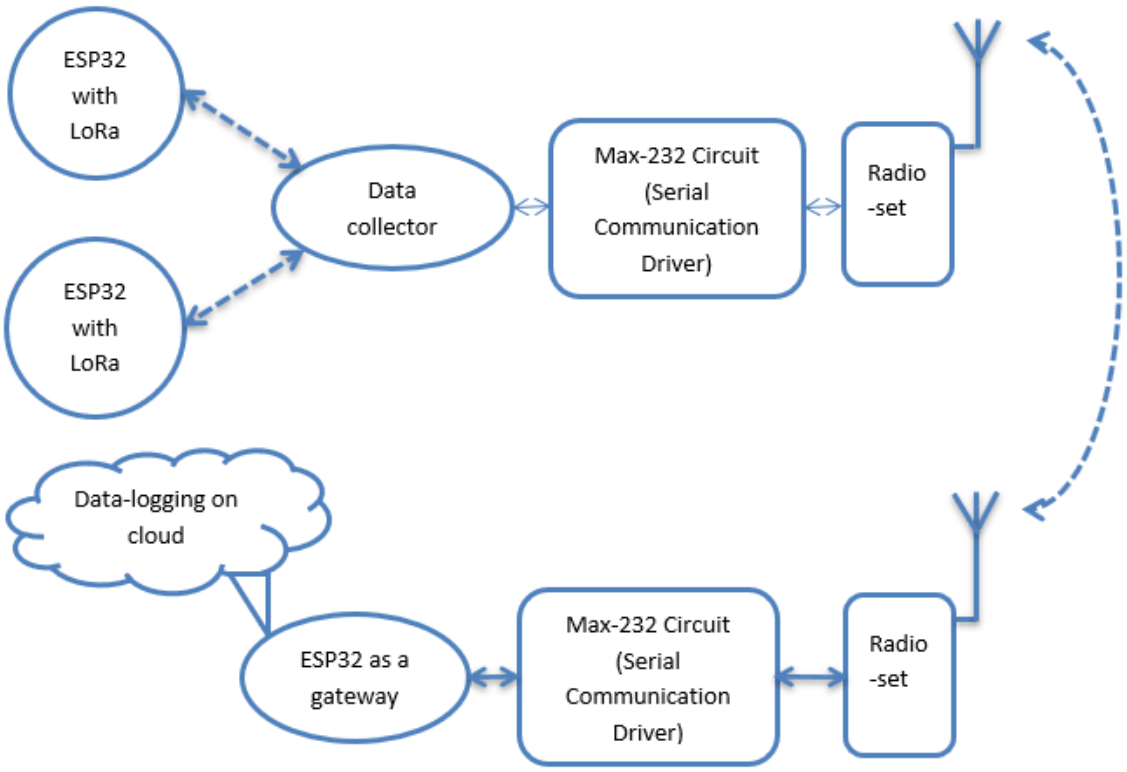


Figure 5-11 System structure for topology II with broader range due to radio-set

Although, the system structure-II seems very interesting because it simplifies the system by eliminating the PC and making system more power efficient but, there are few limitations of this system e.g.

- It uses ESP32 as a gateway and uploads the decrypted data to the server through wifi due to which data becomes vulnerable to the eavesdropper.
- An ESP32 is not a dedicated gateway but is being used for serial communication, encryption, decryption, MAC generation, MAC verification and uploading data to the server which make it slow. This system structure may prove very helpful if there

is not high traffic of two-way messages over the channel (not more than 10 messages per hour).

But, for the SCADA system of remote micro-grids, it cannot be used due to the increased message on air time and data loss. Due to this drawback, a third system structure was designed and implemented. It has been explained in the next section.

5.5 System Structure-III

Quite similar to the previously discussed system structure, this system also has similar type of structure on remote end but, is different in system structure for the SCADA side. Its simplified structure has been shown in fig 5.12. There is a MAX-232 circuit which serially connects the radio set with the ESP32 collector-dragino side. This ESP32 has been used to collect the received data serially, to partially decrypt the received data and then send to the dragino gateway through LoRa communication. Dragino-yun is a dedicated LoRa gateway and does not face any time delay issues. It was also planned to feed the serial output of MAX2232 direct to the dragino gateway without the use of ESP32 collector but, due to the limitations given below it was not possible

a) Dragino gateway cannot decrypt the received message due to limited memory. When it was tried a low-memory error occurred as shown in fig 5.13
b) It is built on one dragino-yun shield and one other chip for the gateway, and both are connected through Rx/Tx pins. So, external use of Rx/Tx pins is not possible

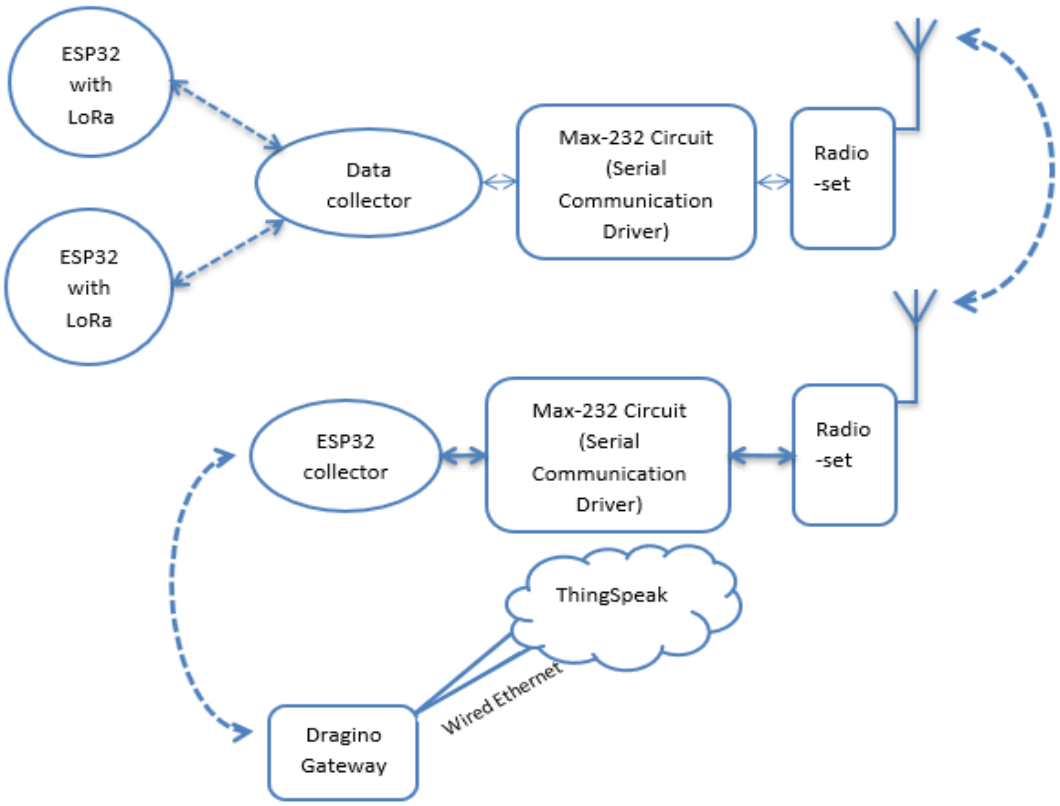


Figure 5-12 System structure for topology III with broader range due to radio-set

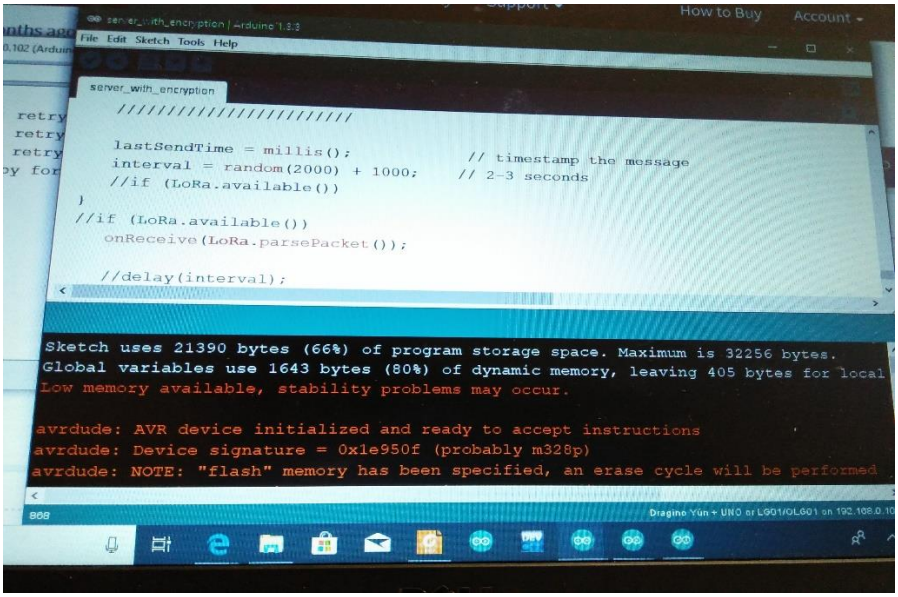


Figure 5-13 Dragino-yun low memory error

Therefore, an additional ESP32 has been used as a collector to decrypt/encrypt the received message and send the serial/LoRa received data on LoRa/Serial. The ESP32 decrypts the data partially, and extracts a string of HEX from the encrypted string and forwards to the dragino gateway. Finally, the dragino gateway has been used to extract the real data from the partially decrypted message and uploads that to the server.

A prototype of the message encrypted with AES was sent from the remote end node and a dummy data was logged on the ThingSpeak server. Data sample is logged once in every fifteen seconds and it offers eight different fields to log under single user ID without any charges.The results have been shown in fig 5.14.

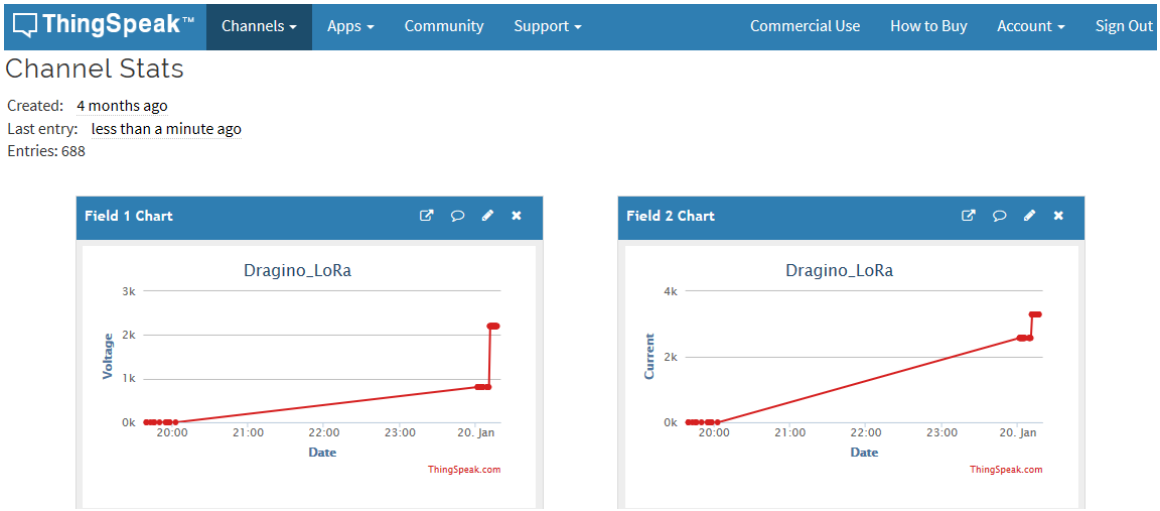


Figure 5-14 Data logging on the ThingSpeak server using system topology III

5.6 Conclusion

A radio-set based broader range communication system has been setup. It significantly improves the range up to 50km. System structure-1 costs not more than $325 (Three ESP32

each of $26, USR-TCP-232 of $28 and LSR-150-12 with two Radio-sets each of $109.53) including radio-sets and system structure-II&III cost $323 and $353 respectively. Here, three different topologies have been tried for secure communication and data logging. Each topology has its own circle of applications based upon merits and demerits. If data logging is required only at the main SCADA unit then first topology works very well. Second topology is useful for logging data on to the server to access remotely but, it has certain limitations. It will be nice option for data logging if there is very limited number of messages (roughly 10 messages/hr). The final one is useful if very frequent communication is required. Then, this topology seems the optimal solution for real-time communication and logging data on the server.

After implementing these three topologies and identifying their limitations, the System Structure-III shown in fig-5.12 was finalized for the SCADA system because, it meets all system requirements and has longer range (>50km) due to radio-sets. Cascaded mesh-topology cannot be implemented to achive this range because due to increase of cumulative message on air time, it becomes more vulnerable to data loss and electromagnetic interference. The finalized system consists of two radio-sets, three ESP32s, two MAX232s circuits and one dragino gateway and costs $353 and gives range of more than 50km. It has secure communication due to the implementation of AES algorithm. It gives message authenticity due to the implementation of MAC. Usage of the dedicated gateway makes it superior over System Structure-II and it supercedes the System Structure-I due to the feature of remote data logging.

Chapter # 6 Summary and Recommendations

6.1 Summary of Work

With increasing renewable energy penetration, the number of distributed generation and remote micro-grids are also increasing. While each remote micro-grid requires to have a sophisticated (secure, long range, low power and low cost) communication system for SG SCADA system. It creates a research gap for devising a system with desired attributes. This research was conducted to develop a system with the above listed features. This goal was achieved in five research phases given below

1. Finding a low power, low cost and long range communication system.

   To achieve this, the literature review comprising the study of twelve wireless technologies and three wired technologies was done. After the comparison of those technologies, a comparison table was constructed, and LoRa communication was chosen for this purpose.

2. Finding and implementing a secure communication mechanism.

   To achieve the security of the communication system, different encryption algorithms were studied, implemented and cross-checked by developing C++ code

and using online software. Encryption was implemented on using DORJI DRF1276G Arduino-LoRa moidules.

3. Authentication of the message signals and implementation of advanced encryption algorithms.

   To authenticate the message algorithm was developed to generate a unique message authentication code for each message. It enabled to identify a bit-level alteration in the message. Further, AES was implemented on ESP32 to secure the message from decryption.

4. Logging data on a cloud for remote monitoring and improving the LoRa range.

   Data was uploaded to the server to access remotely. It was achieved by programming and configuring two gateways for this purpose. Further, LoRa range was improved by bringing the idea of its implementation in mesh-network structure. This led to a low-cost ($78 +$26 for every new node and level), low power (600mW + 200mW for every new node and level), and improved the range by 5km on-site and 15km off-site by every new mesh-network level.

5. Extending the system to get longer range (50km) without data loss and electromagnetic intergference.

   It was achieved by implementing a hybrid system using LoRa and radio-system. Considering the requirements of the SG SCADA system, three possible different system topologies were designed and implemented. Final design of 3rd topology consists of ESP32s, Radio-sets and dragino-gateway with total cost of $353 and power consumption of 43W. And it was recommended based upon the SG SCADA system objectives.

6.2 Future Work Recommendations

There is always a possibility to make the existing work better and here are the few recommendations.

1. A private server can be developed to eliminate any possible way of intruder's intervention through server side and internet.
2. After the study of the SCADA system, local controllers can be given certain privileges to minimize the communication of critical messages and rank data based upon data priority.
3. Local graphical user interfacing (Human Machine Interfacing) and control can be introduced to make the system more flexible for adding/removing nodes.
4. Design a proper housing and power supply for the communication link.

6.3 List of Publications

6.3.1 Submitted

1. Amjad Iqbal and M. Tariq Iqbal, "Low-cost and Secure Communication System for SCADA System of Remote Micro-grids", accepted in the Journal of Electrical and Computer Engineering, April 2, 2019

6.3.2 Published/Accepted

2. Amjad Iqbal and M. Tariq Iqbal, "Low-cost and Secure Communication System for Remote Micro-grids using AES Cryptography on ESP32 with LoRa Module," presented in IEEE Electrical Power and Energy Conference (EPEC), 2018

3. Amjad Iqbal and M. Tariq Iqbal, “Design and Analysis of a Stand-alone PV System for a Rural House in Pakistan,” published in the International Journal of Photoenergy, April 24, 2019
4. Amjad Iqbal and M. Tariq Iqbal, “Thermal Modeling and Sizing of a Stand-Alone PV System for a Rural House in Pakistan,” presented at 27th IEEE NECEC Conference, 2018

6.3.3 Poster Presentation

5. Amjad Iqbal and M. Tariq Iqbal, “Low-cost and Secure Communication System for Remote Micro-grids using AES Cryptography and ESP32 with LoRa,” presented in poster session at Ryerson University, Toronto ON, during NESTNet 2nd Annual Technical Conference, June 19-20, 2018

References

[1] F. Qi *et al.*, "Optimal planning of smart grid communication network for interregional wide-area monitoring protection and control system," *Proceedings - 2nd IEEE International Conference on Energy Internet, ICEI 2018*. pp. 190–195, 2018.

[2] H. Lim, J. Ko, S. Lee, J. Kim, M. Kim, and T. Shon, "Security architecture model for smart grid communication systems," *2013 Int. Conf. IT Converg. Secur. ICITCS 2013*, pp. 1–4, 2013.

[3] P. Kansal, S. Member, A. Bose, and A. Infrastructure, "Smart Grid Communication Requirements for the High Voltage Power System," *2011 IEEE Power Energy Soc. Gen. Meet.*, pp. 1–6, 2011.

[4] M. F. Aziz and N. Abdulaziz, "Prospects and challenges of renewable energy in Pakistan," in *2010 IEEE International Energy Conference and Exhibition, EnergyCon 2010*, 2010, pp. 161–165.

[5] Q. Yang, J. A. Barria, and T. C. Green, "Communication infrastructures for distributed control of power distribution networks," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2. pp. 316–327, 2011.

[6] V. C. Güngör, D Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, "Smart grid technologies: Communication technologies and standards," *IEEE Trans. Ind. Informatics*, vol. 7, no. 4, pp. 529–539, 2011.

[7] B. Flerchinger, Robert Ferraro, Chris Steeprow, Michael Mills-Price, J.W. Knapek, "Third-Generation Cellular and Wireless Serial Radio Communications: Field Testing for Smart Grid Applications," *IEEE Ind. Appl. Mag.*, vol. 24, no. 5, pp. 10–17, Oct. 2018.

[8] S. Supriya, M. Magheshwari, S. Sree Udhyalakshmi, R. Subhashini, and Musthafa, "Smart grid technologies: Communication technologies and standards," *Int. J. Appl. Eng. Res.*, 2015.

[9] V. C. Güngör, D Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, "A Survey on smart grid potential applications and communication requirements," *IEEE Trans. Ind. Informatics*, 2013.

[10] P. G. Kate and J. R. Rana, "ZIGBEE based monitoring theft detection and automatic electricity meter reading," in *International Conference on Energy Systems and Applications, ICESA 2015*, 2016.

[11] P. Yi, A. Iwayemi, and C. Zhou, “Developing ZigBee deployment guideline under WiFi interference for smart grid applications,” *IEEE Trans. Smart Grid*, 2011.

[12] A. Mulla, J. Baviskar, S. Khare, and F. Kazi, “The Wireless Technologies for Smart Grid Communication: A Review,” 2015, pp. 442–447.

[13] Y. Gu, A. Lo, S. Member, and I. Niemegeers, “Wireless Personal Networks,” *Communications*, 2009.

[14] X. Xu, X. Hu, Z. Zhu, and X. Ji, “Application and research of Bluetooth technology in the development of the portable device for the evaluation of myodynamia,” *ITME 2011 - Proceedings: 2011 IEEE International Symposium on IT in Medicine and Education*, vol. 2. pp. 620–623, 2011.

[15] M. Aiello, R. De Jong, and J. De Nes, “Bluetooth broadcasting: How far can we go? An experimental study,” in *2009 Joint Conferences on Pervasive Computing, JCPC 2009*, 2009.

[16] K. A. T. Lasagani, T. Iqbal, and G. K. Mann, “Data Logging and Control of a Remote Inverter Using LoRa and Power Line Communication,” *Energy Power Eng.*, vol. 10, no. 08, pp. 351–365, 2018.

[17] M. Conti, D. Fedeli, and M. Virgulti, “B4V2G: Bluetooth for electric vehicle to smart grid connection,” 2011, pp. 13–18.

[18] ABB, “Utility Communications Narrowband SCADA Radio Modems,” *Power and Productivity for a Better World*. [Online]. Available: https://library.e.abb.com/public/73ca197a99448ddbc12576bd00458654/Narrowband SCADA Radio Modems.pdf. [Accessed: 02-Feb-2019].

[19] Aa. Shahzad, Y. G. Kim, and A. Elgamoudi, “Secure IoT Platform for Industrial Control Systems,” in *2017 International Conference on Platform Technology and Service, PlatCon 2017 - Proceedings*, 2017.

[20] B. Glushko, D. Kin, and A. Shar, “Gigabit optical wireless communication system for personal area networking,” *Proc. 2013 18th Eur. Conf. Netw. Opt. Commun. NOC 2013 2013 8th Conf. Opt. Cabling Infrastructure, OC I 2013*, pp. 145–148, 2013.

[21] M. Rahaim, Thomas D. C. Little, “Interference in IM/DD optical wireless communication networks,” *IEEE/OSA J. Opt. Commun. Netw.*, vol. 9, no. 9, pp. D51–D63, 2017.

[22] M. Uysal and H. Nouri, “Optical wireless communications - An emerging technology,” in *International Conference on Transparent Optical Networks*, 2014.

[23] P. J. Winzer and R. J. Essiambre, “Advanced optical modulation formats,” in *Optical Fiber Telecommunications V1B*, 2008.

[24] S. Arnon, J. R. Barry, G. K. Karagiannidis, R. Schober, and M. Uysal, *Advanced optical wireless communication systems*. 2012.

[25] “Laser Demonstration Reveals Bright Future for Space Communication,” *Dewayne Washington NASA’s Goddard Space Flight Center*, Washington, 04-Mar-2016.

[26] A. Nowicki, “Boulder’s Smart Grid Leaves Citizens in the Dark,” *A Wood Mackenzie Business*, 18-Mar-2013.

[27] D. Buterbaugh, Timothy Brown, “Boulder Strategic Networking for Power System,” Colorado, 2010.

[28] M. Jia, X. Gu, Q. Guo, W. Xiang, and N. Zhang, “Broadband hybrid satellite-terrestrial communication systems based on cognitive radio toward 5G,” *IEEE Wireless Communications*, vol. 23, no. 6. pp. 96–106, 2016.

[29] M. Guimarães Castello Branco, Augusto da Rocha Gomes, “Satellite Communication Challenges in a Fully Interconnected World,” in *2017 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC)*, 2017, p. 5.

[30] K. Sohraby, Daniel Minoli, Benedict Occhiogrosso, Wei Wang, “A Review of Wireless and Satellite-Based M2M/IoT Services in Support of Smart Grids,” *Mob. Networks Appl.*, vol. 23, no. 4, pp. 881–895, 2017.

[31] C. X. Wang, F. Haider, X. Gao, X. H. You, Y. Yang, H. M. Aggoune, H. Haas, S. Fletcher and E. Hepsaydir, “Cellular architecture and key technologies for 5G wireless communication networks,” *IEEE Commun. Mag.*, 2014.

[32] J. Ghetie, “Wireless Communications and Networking,” in *Fixed-Mobile Wireless Networks Convergence*, 2009.

[33] Y. Chung, Jae Young Ahn, Jae Du Huh, “Experiments of A LPWAN Tracking (TR) Platform Based on Sigfox Test Network,” in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, 2018.

[34] N. I. Osman, Esra B. Abbas, “Simulation and Modelling of LoRa and Sigfox Low Power Wide Area Network Technologies,” in *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, 2018.

[35] B. Vejlgaard, M. Lauridsen, H. Nguyen, I. Z. Kovacs, P. Mogensen, and M. Sorensen, “Coverage and Capacity Analysis of Sigfox, LoRa, GPRS, and NB-IoT,” in *IEEE Vehicular Technology Conference*, 2017.

[36] F. Bonavolontà, Annarita Tedesco, Rosario Schiano Lo Moriello, Antonio Tufano, "Enabling wireless technologies for industry 4.0: State of the art," in *2017 IEEE International Workshop on Measurement and Networking (M&N)*, 2017.

[37] L. Vangelista, "Frequency Shift Chirp Modulation: The LoRa Modulation," *IEEE Signal Processing Letters*, vol. 24, no. 12. pp. 1818–1821, 2017.

[38] A. Lavric, Valentin Popa, "Internet of Things and LoRa™ Low-Power Wide-Area Networks: A survey," in *2017 International Symposium on Signals, Circuits and Systems (ISSCS)*, 2017.

[39] A. U. Macrae and A. M. Noll, "Bell Telephone Laboratories, Incorporated: 1925-1984 New Jersey's innovation factory," in *Proceedings of the 2015 ICOHTEC/IEEE International History of High-Technologies and their Socio-Cultural Contexts Conference, HISTELCON 2015: The 4th IEEE Region 8 Conference on the History of Electrotechnologies*, 2015.

[40] G. Bumiller, L. Lampe, and H. Hrasnica, "Power line communication networks for large-scale control and automation systems," *IEEE Commun. Mag.*, 2010.

[41] S. Rinaldi, P. Ferrari, A. Flammini, M. Rizzi, E. Sisinni, and A. Vezzoli, "Performance analysis of power line communication in industrial power distribution network," *Comput. Stand. Interfaces*, 2015.

[42] K. Sharma and L. M. Saini, "Power-line communications for smart grid: Progress, challenges, opportunities and status," *Renewable and Sustainable Energy Reviews*. 2017.

[43] S. Galli, A. Scaglione, and Z. Wang, "For the grid and through the grid: The role of power line communications in the smart grid," in *Proceedings of the IEEE*, 2011.

[44] A. Leonov and V. Konyshev, "From the Revolution to the Evolution: The Change in the Character of Development of Fiber Optic Communications Technology — And the Record Performance of 100 Gbit/s Systems as a Marker of this Change," 2017, pp. 34–36.

[45] G. P. Agrawal, *Lightwave Technology: Telecommunication Systems*. 2005.

[46] R. J. Essiambre and R. W. Tkach, "Capacity trends and limits of optical communication networks," in *Proceedings of the IEEE*, 2012.

[47] C. DeCusatis, *Handbook of Fiber Optic Data Communication*. 2008.

[48] J. Vodrazka, "Potential use of gigabit digital subscriber lines in hybrid access networks," in *International Conference on Digital Technologies 2013, DT 2013*, 2013.

[49] D. Mandell, “LoRaWAN Assumes LPWAN Leadership for IoT Gateways.” VDC Research Insight for the Connected World, 2017.

[50] Y. Tang, W. Li, C. Xiong, J. Wei, and M. Ghogho, “Secure communications via sending artificial noise by both transmitter and receiver: optimum power allocation to minimise the insecure region,” *IET Communications*, vol. 8, no. 16. pp. 2858–2862, 2014.

[51] A. D. Harper and X. Ma, “MIMO Wireless Secure Communication Using Data-Carrying Artificial Noise,” *IEEE Transactions on Wireless Communications*, vol. 15, no. 12. pp. 8051–8062, 2016.

[52] Z. Chen, T. Jiang, and W. Zou, “A novel physical layer security communication method based on dual base station,” *2017 17th International Symposium on Communications and Information Technologies, ISCIT 2017*, vol. 2018–January. pp. 1–4, 2018.

[53] T. Q. Duong, “Keynote talk #1: Trusted communications with physical layer security for 5G and beyond,” 2017.

[54] C. Mavrokefalidis, D. Ampeliotis and K.Berberidis, “A study of the communication needs in micro-grid systems,” in *General Assembly and Scientific Symposium of the International Union of Radio Science (URSI GASS), 2017 XXXIInd*, 2017, pp. 1–4.

[55] A. García-Domínguez, “Enabling SCADA cluster and cloud for smart grid using hierarchical multicast; The PTMF framework,” in *Proceedings of the IEEE International Conference on Industrial Technology*, 2015, vol. 2015–June, no. June, pp. 218–225.

[56] H. H. and D. M. S. and M. G. and A. K. Safa, “Cyber security of smart grid and SCADA systems, threats and risks,” in *CIRED Workshop 2016*, 2016, pp. 1–4.

[57] E. Bou-Harb, C. Fachkha, M. Pourzandi, M. Debbabi, and C. Assi, “Communication security for smart grid distribution networks,” *IEEE Commun. Mag.*, vol. 51, no. 1, pp. 42–49, 2013.

[58] H. H. and D. M. S. and M. G. and A. K. Safa, “Cyber security of smart grid and SCADA systems, threats and risks,” in *CIRED Workshop 2016*, 2016, pp. 1–4.

[59] A. Tanenbaum, “Network Security,” in *Computer Networks*, 5th ed., PEARSON, 2011, pp. 767–790.

[60] H. Su, M. Qiu, and H. Wang, “Secure wireless communication system for smart grid with rechargeable electric vehicles,” *IEEE Commun. Mag.*, vol. 50, no. 8, pp. 62–68, 2012.

[61] D. NamdeoHire, “Secured Wireless Data Communication,” *Int. J. Comput. Appl.*, vol. 54, pp. 27–30, 2012.

[62] A. A. P. R. and R. F. S. Amiruddin, “A testbed implementation of secure and lightweight privacy preservation mechanism using scrambled Fibonacci and XOR for ZigBee,” in *Region 10 Conference, TENCON 2017 - 2017 IEEE*, 2017, pp. 863–868.

[63] Y.-S. Tsai, C.-Y. Chu, M.-C. Li, Y.-H. Lin, and P. Chen, “Intelligent DC power monitoring system and sensor network based on ZigBee-equipped smart sockets,” in *2016 5th International Symposium on Next-Generation Electronics, ISNE 2016*, 2016.

[64] Aa. Shahzad, Y. G. Kim, and A. Elgamoudi, “Secure IoT Platform for Industrial Control Systems,” in *2017 International Conference on Platform Technology and Service, PlatCon 2017 - Proceedings*, 2017.

[65] A. V. D. M. Kayem, H. Strauss, S. D. Wolthusen, and C. Meinel, “Key management for secure demand data communication in constrained micro-grids,” in *Proceedings - IEEE 30th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2016*, 2016, pp. 585–590.

[66] J. L. Tsai and N. W. Lo, “Secure Anonymous Key Distribution Scheme for Smart Grid,” *IEEE Trans. Smart Grid*, 2016.

[67] X. Miao and X. Chen, “Cyber security infrastructure of smart grid communication system,” *China Int. Conf. Electr. Distrib. CICED*, no. Ciced, pp. 5–6, 2012.

[68] W. Stallings, *Cryptography and Network Security*, vol. 139, no. 3. 2011, pp. 312-328.

[69] C.-S. Choi, , Jin-Doo Jeong, Il-Woo Lee, Wan-Ki Park, “LoRa based Renewable Energy Monitoring System with Open IoT Platform,” in *International Conference on Electronics, Information, and Communication (ICEIC)*, 2018, pp. 1–2.

[70] H.-R. Lee,Won-Jong Kim,Ki-Hyuk Park,Han-Jin Cho, Chi-Ho Lin, “Development of an easy payment system based on IoT gateway,” in *International Conference on Electronics, Information, and Communication (ICEIC)*, 2018, pp. 1–3

[71] R. G. Anvekar, Rajeshwari M Banakar, Rajat R Bhat, “Design alternatives for end user communication in IoT based system model,” in *IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR)*, 2017.

[72] W. Stallings, *Cryptography and Network Security*, vol. 139, no. 3. 2011, pp. 312-328.

[73] P. Patil, P. Narayankar, D. G. Narayan, and S. M. Meena, “A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish,” in *Procedia Computer Science*, 2016, vol. 78, pp. 617–624.

[74] Dragino, “LG01 LoRa Gateway User Manual.” Dragino, pp. 22–30, 2018.

Appendixes Containing C++/Arduino Code

Appendix A *C++ code to Breach the Hill Cipher*

```
#include <iostream>
#include <string>
using namespace std;
int main ()
{                   /* Coding Block-1: splitting string into pairs*/
          while (1) { string input;
cout<<"Please input ciphertext string without SPACES and with even number of characters
"<<endl<<endl;
cin>>input;
int counter=0, string_length=input.length(),half_length=string_length/2, counter1=0,
frequency_index[half_length], dummy_counter=0,freq=1, freq_index=0;
string  half_string_o="", half_string_e="",odd,even,pair[half_length];
char array_of_character[string_length];
if ((input.length()%2)==0)
{                   for (counter;counter<input.length();counter++)
          {                   if (counter%2==0)
                    {         half_string_e=half_string_e+input[counter]; }
                                        else if (counter%2!=0)
                    {         half_string_o=half_string_o+input[counter];}
                    array_of_character[counter]=input[counter];
                    cout<<"     "<<array_of_character[counter];}
          cout<<endl<<"Here are the pairs of characters"<<endl;
for (counter1;counter1<half_length;counter1++)
{ odd=half_string_o[counter1]; even=half_string_e[counter1];
pair[counter1]=even+odd;
cout<<pair[counter1]<<" ";}
cout<<endl;
{         /* Block-2 Frequency: Counting repetition of pairs*/
int dummy3=0;
for (dummy_counter;dummy_counter<=half_length;dummy_counter++)
{         int nested_dummy=dummy_counter, frequency=0;
          for(nested_dummy;nested_dummy<half_length;nested_dummy++)
          {         if (pair[nested_dummy]==pair[dummy_counter])
```

```
                frequency=frequency+1;}
        frequency_index[dummy_counter]=frequency;}
cout<<"cipher block"<<"  frequency"<<endl;
for (dummy3;dummy3<half_length;dummy3++)
{       cout<<"    "<<pair[dummy3]<<"          "<<frequency_index[dummy3]<<endl;}
                        /*Block-3: Decryption*/
cout<<endl<<endl<<"                                    decryption    "<<endl<<endl;
int         dummy7=0,         dummy8=0,         p0,p1,counter6=0,plain_text_number[2],
cipher_text_number[2],key[4]={5,19,24,3};
        char plain_text[2],cipher_text[2], array_of_plain_text[string_length];
for (dummy7=0;dummy7<string_length;)
{       for (dummy8=0;dummy8<2;dummy8++)
        {       cipher_text[dummy8]=array_of_character[dummy7];
                if (cipher_text[dummy8]=='A'||cipher_text[dummy8]=='a')
                        cipher_text_number[dummy8]=0;
                else if (cipher_text[dummy8]=='B'||cipher_text[dummy8]=='b')
                        cipher_text_number[dummy8]=1;
                else if (cipher_text[dummy8]=='C'||cipher_text[dummy8]=='c')
                        cipher_text_number[dummy8]=2;
                else if (cipher_text[dummy8]=='D'||cipher_text[dummy8]=='d')
                        cipher_text_number[dummy8]=3;
                else if (cipher_text[dummy8]=='E'||cipher_text[dummy8]=='e')
                        cipher_text_number[dummy8]=4;
                else if (cipher_text[dummy8]=='F'||cipher_text[dummy8]=='f')
                        cipher_text_number[dummy8]=5;
                else if (cipher_text[dummy8]=='G'||cipher_text[dummy8]=='g')
                        cipher_text_number[dummy8]=6;
                else if (cipher_text[dummy8]=='H'||cipher_text[dummy8]=='h')
                        cipher_text_number[dummy8]=7;
                else if (cipher_text[dummy8]=='I'||cipher_text[dummy8]=='i')
                        cipher_text_number[dummy8]=8;
                else if (cipher_text[dummy8]=='J'||cipher_text[dummy8]=='j')
                        cipher_text_number[dummy8]=9;
                else if (cipher_text[dummy8]=='K'||cipher_text[dummy8]=='k')
                        cipher_text_number[dummy8]=10;
                else if (cipher_text[dummy8]=='L'||cipher_text[dummy8]=='l')
                        cipher_text_number[dummy8]=11;
                else if (cipher_text[dummy8]=='M'||cipher_text[dummy8]=='m')
                        cipher_text_number[dummy8]=12;
                else if (cipher_text[dummy8]=='N'||cipher_text[dummy8]=='n')
                        cipher_text_number[dummy8]=13;
```

```
        else if (cipher_text[dummy8]=='O'||cipher_text[dummy8]=='o')
                cipher_text_number[dummy8]=14;
        else if (cipher_text[dummy8]=='P'||cipher_text[dummy8]=='p')
                cipher_text_number[dummy8]=15;
        else if (cipher_text[dummy8]=='Q'||cipher_text[dummy8]=='q')
                cipher_text_number[dummy8]=16;
        else if (cipher_text[dummy8]=='R'||cipher_text[dummy8]=='r')
                cipher_text_number[dummy8]=17;
        else if (cipher_text[dummy8]=='S'||cipher_text[dummy8]=='s')
                cipher_text_number[dummy8]=18;
        else if (cipher_text[dummy8]=='T'||cipher_text[dummy8]=='t')
                cipher_text_number[dummy8]=19;
        else if (cipher_text[dummy8]=='U'||cipher_text[dummy8]=='u')
                cipher_text_number[dummy8]=20;
        else if (cipher_text[dummy8]=='V'||cipher_text[dummy8]=='v')
                cipher_text_number[dummy8]=21;
        else if (cipher_text[dummy8]=='W'||cipher_text[dummy8]=='w')
                cipher_text_number[dummy8]=22;
        else if (cipher_text[dummy8]=='X'||cipher_text[dummy8]=='x')
                cipher_text_number[dummy8]=23;
        else if (cipher_text[dummy8]=='Y'||cipher_text[dummy8]=='y')
                cipher_text_number[dummy8]=24;
        else if (cipher_text[dummy8]=='Z'||cipher_text[dummy8]=='z')
                cipher_text_number[dummy8]=25;
                dummy7=dummy7+1; }
p0=(cipher_text_number[0]*key[0]+cipher_text_number[1]*key[2]);
p1=(cipher_text_number[0]*key[1]+cipher_text_number[1]*key[3]);
p0=p0%26;
p1=p1%26;
plain_text_number[0]=p0;
plain_text_number[1]=p1;
for (counter6=0;counter6<2;counter6++)
{       if (plain_text_number[counter6]==0)
        {plain_text[counter6]='a';}
        else if (plain_text_number[counter6]==1)
                {plain_text[counter6]='b';}
        else if (plain_text_number[counter6]==2)
                {plain_text[counter6]='c';}
        else if (plain_text_number[counter6]==3)
                {plain_text[counter6]='d';}
        else if (plain_text_number[counter6]==4)
```

```
        {plain_text[counter6]='e';}
else if (plain_text_number[counter6]==5)
        {plain_text[counter6]='f';}
else if (plain_text_number[counter6]==6)
        {plain_text[counter6]='g';}
else if (plain_text_number[counter6]==7)
        {plain_text[counter6]='h';}
else if (plain_text_number[counter6]==8)
        {plain_text[counter6]='i';}
else if (plain_text_number[counter6]==9)
        {plain_text[counter6]='j';}
else if (plain_text_number[counter6]==10)
        {plain_text[counter6]='k';}
else if (plain_text_number[counter6]==11)
        {plain_text[counter6]='l';}
else if (plain_text_number[counter6]==12)
        {plain_text[counter6]='m';}
else if (plain_text_number[counter6]==13)
        {plain_text[counter6]='n';}
else if (plain_text_number[counter6]==14)
        {plain_text[counter6]='o';}
else if (plain_text_number[counter6]==15)
        {plain_text[counter6]='p';}
else if (plain_text_number[counter6]==16)
        {plain_text[counter6]='q';}
else if (plain_text_number[counter6]==17)
        {plain_text[counter6]='r';}
else if (plain_text_number[counter6]==18)
        {plain_text[counter6]='s';}
else if (plain_text_number[counter6]==19)
        {plain_text[counter6]='t';}
else if (plain_text_number[counter6]==20)
        {plain_text[counter6]='u';}
else if (plain_text_number[counter6]==21)
        {plain_text[counter6]='v';}
else if (plain_text_number[counter6]==22)
        {plain_text[counter6]='w';}
else if (plain_text_number[counter6]==23)
        {plain_text[counter6]='x';}
else if (plain_text_number[counter6]==24)
        {plain_text[counter6]='y';}
```

```
                    else if (plain_text_number[counter6]==25)
                                {plain_text[counter6]='z';}}
          cout<<plain_text[0]<<plain_text[1];}}}
cout<<endl;}
return 0;}
```

Appendix B *Arduino code for LoRa duplex communication*

```
#include <SPI.h>
#include <LoRa.h>
int counter = 0, unread_bytes=0;
char data[]="Hello world";
void setup() {
  LoRa.setSPIFrequency(8E6);/*it will set controller at 8MHz*/
  LoRa.begin(915E6);/*it will return 1 on success and 0 on failure*/
  LoRa.end();
  Serial.begin(9600);
  /*while (!Serial);
  Serial.println("LoRa Sender");*/
    if (!LoRa.begin(915E6)) //Checking whether the LoRa is receiving 915GHz freq or not
  { Serial.println("Starting LoRa failed!");
    while (1);
 } }
void loop() {
                /*Receive packets*/
    ////////////////////////////////
    unread_bytes=LoRa.available();
   // try to parse packet
  int packetSize = LoRa.parsePacket();
```

```
 if (unread_bytes!=0/*packetSize*/) {
   // received a packet
   Serial.print("Received packet '");
   // read packet
  while (LoRa.available()){
 // LoRa.onReceive(packetSize);
     Serial.print((char)LoRa.read());
  }
  // print RSSI of packet
   Serial.print("' with RSSI ");
   Serial.println(LoRa.packetRssi());
 }
 delay(500);
 unread_bytes=0;
              /*Write message*/
 ////////////////////////////////
 if(LoRa.available()/*unread_bytes==0 /*&& message_to_be_sent!=0*/) {
 Serial.print("Message packet sent: ");
 Serial.println(counter);
  // send packet
 LoRa.beginPacket();
 LoRa.print(data);
 LoRa.print(counter);
   LoRa.endPacket();
 counter++;
 delay(500);
} }
```

Appendix C *Flexible Arduino-code for all level nodes of mesh-network to display/encrypt/decrypt/forward message*

```
#include <SPI.h>
#include <LoRa.h>
#include<Arduino.h>
#include "SSD1306.h"
#define SS      18
#define RST     14
#define DI0     26
#define BAND    915E6  //915E6 --
SSD1306 display (0x3c, 4, 15);
int counter = 0;
String outgoing;              // outgoing message
int msgCount = 0;             // count of outgoing messages
int received_msg_count = 0;           // count of received messages
String received_message_status="";
long lastSendTime = 0;        // last send time
int interval = 500;          // interval between sends
String deviceAddress="Node1", receiverAddress="SCADA";// Device address is the address of this device and receiver address is the address of target device
void setup() {
//*************************8
  pinMode (16, OUTPUT);
```

```
  pinMode (2, OUTPUT);
  digitalWrite (16, LOW); // set GPIO16 low to reset OLED
  delay (50);
  digitalWrite (16, HIGH); // while OLED is running, GPIO16 must go high
  //******************
    Serial.begin(9600);
  while (!Serial); //If just the the basic function, must connect to a computer
  SPI.begin(5,19,27,18);
  LoRa.setPins(SS,RST,DI0);
  Serial.println("SCADA side LoRa Sender/Receiver");
  if (!LoRa.begin(BAND)) {
    Serial.println("Error: LoRa configuring failed!");
    while (1);
  }
  Serial.println("LoRa setup configured successfully!");
//****************
display.init ();
  display.flipScreenVertically ();
  display.setFont (ArialMT_Plain_10);
  delay (1500);
  //*******************888
LoRa.setSpreadingFactor(7);
}
 //*********************************************************
//*********************************************************
//****************************Encryption_Block***********
//*********************************************************
```

```
//********************************************** Working fine
//**********************************************
//**********************************************it takes HEX string and returns after row-shift operation regardless of length of string
String shift_rows(String xy)
{
  int i=xy.length(),j=i/32,l=0,m=0,q=0;
  String st="",tu="";
  for(l=0;l<i;l++)
  {
    m=l%32;q=0;
    if(m==0)
    { st="";
      for(q=0;q<8;q++)
      {
      tu+=xy[l];
      l=l+1;
      }
    }
       m=l%32;
    if(m==8)
    {
      tu+=xy[l+2];   tu+=xy[l+3];      tu+=xy[l+4];     tu+=xy[l+5];
      tu+=xy[l+6];   tu+=xy[l+7];      tu+=xy[l];     tu+=xy[l+1];
      l=l+8;
       }
    m=l%32;
```

```
   if(m==16)
   {
    tu+=xy[l+4];  tu+=xy[l+5];    tu+=xy[l+6];    tu+=xy[l+7];
    tu+=xy[l];  tu+=xy[l+1];    tu+=xy[l+2];    tu+=xy[l+3];
    l=l+8;
     }
   m=l%32;
   if(m==24)
   {
    tu+=xy[l+6];  tu+=xy[l+7];    tu+=xy[l];    tu+=xy[l+1];
    tu+=xy[l+2];  tu+=xy[l+3];    tu+=xy[l+4];    tu+=xy[l+5];
    l=l+7;
   }
 }
   return tu;
}
//***********************************************
//************************************************************
//********************************************** Working nicely
//**********************It takes single HEX character as input and returns binary value
String hex2bin_(char xy)
{
  String bi="";
    if (xy=='0') {  bi+="0000"; }
   else if (xy=='1') { bi+="0001"; }
   else if (xy=='2') { bi+="0010"; }
```

```
   else if (xy=='3') { bi+="0011"; }
   else if (xy=='4') { bi+="0100"; }
   else if (xy=='5') { bi+="0101"; }
   else if (xy=='6') { bi+="0110"; }
   else if (xy=='7') { bi+="0111"; }
   else if (xy=='8') { bi+="1000"; }
   else if (xy=='9') { bi+="1001";}
   else if (xy=='a'||xy=='A') {  bi+="1010"; }
   else if (xy=='b'||xy=='B') {  bi+="1011"; }
   else if (xy=='c'||xy=='C') {  bi+="1100"; }
   else if (xy=='d'||xy=='D') {  bi+="1101"; }
   else if (xy=='e'||xy=='E') {  bi+="1110"; }
   else if (xy=='f'||xy=='F') {  bi+="1111"; }
     return bi;
}
//**************************************************************************
//**************************************************************************
//********************HEX to Binary conversion of a string
String hex2bin_string(String xy)
{
  String bi="";
  int j=xy.length();
  for(int i=0;i<j;i++)
  {
    bi+=hex2bin_(xy[i]);
  }
  return bi;
```

```
}
//**************************************************************************
//**************************************************************************
//**********************************Binary to HEX converter
String bin2hex_(String xy)
{
  String bi="", hex="";
int k=0, l=0,m=0,j=xy.length();
  for (int i=0;i<j;i++)
  {
    bi+=xy[i];
    l+=1;
    //k=bi.length();
    if (l==4)
    {
      if (bi=="0000") { hex+="0";  }
    else if (bi=="0001") { hex+="1"; }
    else if (bi=="0010") { hex+="2"; }
    else if (bi=="0011") { hex+="3"; }
    else if (bi=="0100") { hex+="4"; }
    else if (bi=="0101") { hex+="5"; }
    else if (bi=="0110") { hex+="6"; }
    else if (bi=="0111") { hex+="7"; }
    else if (bi=="1000") { hex+="8"; }
    else if (bi=="1001") { hex+="9";}
    else if (bi=="1010") {  hex+="A"; }
    else if (bi=="1011") {  hex+="B"; }
```

```
   else if (bi=="1100") {  hex+="C"; }
   else if (bi=="1101") {  hex+="D"; }
   else if (bi=="1110") {  hex+="E"; }
   else if (bi=="1111") {  hex+="F"; }
     bi="";
     l=0;
   }
  }
  return hex;
}
//**************************************************************************
//**************************************************************************
//*******************************************Done
//*****************************single character HEX to integer converter
int hex2dec_(char xy)
{
 int bi=0;
  if (xy=='0') {  bi=0; }
   else if (xy=='1') { bi=1; }
   else if (xy=='2') { bi=2; }
   else if (xy=='3') { bi=3; }
   else if (xy=='4') { bi=4; }
   else if (xy=='5') { bi=5; }
   else if (xy=='6') { bi=6; }
   else if (xy=='7') { bi=7; }
   else if (xy=='8') { bi=8; }
   else if (xy=='9') { bi=9;}
```

```
   else if (xy=='a'||xy=='A') {  bi=10; }
   else if (xy=='b'||xy=='B') {  bi=11; }
   else if (xy=='c'||xy=='C') {  bi=12; }
   else if (xy=='d'||xy=='D') {  bi=13; }
   else if (xy=='e'||xy=='E') {  bi=14; }
   else if (xy=='f'||xy=='F') {  bi=15; }
    return bi;
}
//*****************************************
//*****************************************Dec (<256) to Hex (two charcter)*****Done
String dec2hex_string(int bi)
{
  String xy="";
  int p=0,q=0;
  p=bi/16; q=bi%16;
     if (p==0){xy="0"; }
   else if (p==1){xy="1"; }
   else if (p==2){xy="2"; }
   else if (p==3){xy="3"; }
   else if (p==4){xy="4"; }
   else if (p==5){xy="5"; }
   else if (p==6){xy="6"; }
   else if (p==7){xy="7"; }
   else if (p==8){xy="8"; }
   else if (p==9){xy="9"; }
   else if (p==10){xy="A"; }
```

```
   else if (p==11){xy="B"; }
   else if (p==12){xy="C"; }
   else if (p==13){xy="D"; }
   else if (p==14){xy="E"; }
   else if (p==15){xy="F"; }
   if (q==0){xy+="0"; }
   else if (q==1){xy+="1"; }
   else if (q==2){xy+="2"; }
   else if (q==3){xy+="3"; }
   else if (q==4){xy+="4"; }
   else if (q==5){xy+="5"; }
   else if (q==6){xy+="6"; }
   else if (q==7){xy+="7"; }
   else if (q==8){xy+="8"; }
   else if (q==9){xy+="9"; }
   else if (q==10){xy+="A"; }
   else if (q==11){xy+="B"; }
   else if (q==12){xy+="C"; }
   else if (q==13){xy+="D"; }
   else if (q==14){xy+="E"; }
   else if (q==15){xy+="F"; }
    return xy;
}
//*************************************************************************
*********************
//***Multiplies by 2 with input string of
HEX************************************************
```

```
//*************************************************************************
***Completed
String multiply_by_2(String g)
{
  String yz="", zb="";
   yz=hex2bin_(g[0]);
   yz+=hex2bin_(g[1]);
    if(yz[0]=='0')
    {for(int j=0;j<8;j++){ if(j<7){yz[j]=yz[j+1];}if(j==7){yz[j]='0';}}}
      else if(yz[0]=='1')
    {
     {for(int j=0;j<8;j++){if(j<7){yz[j]=yz[j+1];}if(j==7){yz[j]='0';} }}
    if(yz[3]=='1'){yz[3]='0';}
      else if(yz[3]=='0'){yz[3]='1';}
      if(yz[4]=='1'){yz[4]='0';}
      else if(yz[4]=='0'){yz[4]='1';}
      if(yz[6]=='1'){yz[6]='0';}
      else if(yz[6]=='0'){yz[6]='1';}
      if(yz[7]=='1'){yz[7]='0';}
      else if(yz[7]=='0'){yz[7]='1';}
     }
int j=yz.length();
for (int k=0; k<j;k++)
{
  zb+=yz[k];
}
return zb;
```

```
}
//**************************************************************************
**********
//**************************************************************************
**********
//**************************************************************************
**********
//**Multiplies by 3 with input string of
HEX**************************************************
//**************************************************************************
****completed
String multiply_by_3(String g)
{
  String yz="",zy="", zb="";
   yz=hex2bin_(g[0]);
   zy=hex2bin_(g[0]);
   yz+=hex2bin_(g[1]);
   zy+=hex2bin_(g[1]);
   Serial.println(zy);
    if(yz[0]=='0')
    {for(int j=0;j<8;j++){ if(j<7){yz[j]=yz[j+1];}if(j==7){yz[j]='0';}}}
      else if(yz[0]=='1')
    {
      {for(int j=0;j<8;j++){if(j<7){yz[j]=yz[j+1];}if(j==7){yz[j]='0';} }}
    if(yz[3]=='1'){yz[3]='0';}
      else if(yz[3]=='0'){yz[3]='1';}
      if(yz[4]=='1'){yz[4]='0';}
      else if(yz[4]=='0'){yz[4]='1';}
      if(yz[6]=='1'){yz[6]='0';}
```

```
     else if(yz[6]=='0'){yz[6]='1';}
     if(yz[7]=='1'){yz[7]='0';}
     else if(yz[7]=='0'){yz[7]='1';}
    }
  for (int k=0; k<8;k++)
{
   if (yz[k]==zy[k]){yz[k]='0';}
  else if ((yz[k]!=zy[k])&&(yz[k]=='0')) {yz[k]='1';}
  zb+=yz[k];
  }
return zb;
}
//*************************************************************************************
//*************************************************************************************
//*******************input is HEX string of 32 characters (16 bytes)*****Checked for 0nly S00, S10
//******It requires to be checked
*************************************************************
String mix_column(String xy)
{
   int i=0,l=0,m=0,n=0,p=0,x[16]={2,3,1,1,1,2,3,1,1,1,2,3,1,1,2};
  String ab="",bc="",ca="";
  int j=xy.length();
  //String ab="",bc="";
    for (m;m<j;m=m+2)
   {
```

```
    ab="";
    ab=xy[m];
    ab+=xy[m+1];
 if (m==0||m==10||m==20||m==30) //
**************************************Multiply by 2
 {
  ab=multiply_by_2(ab);
 }
 else if (i==6||i==8||i==18||i==28)  //********************* 3s
multiplier***************
 {
  ab=multiply_by_3(ab);
 }
 else {ab=hex2bin_(xy[m]);ab+=hex2bin_(xy[m+1]);}
  bc+=ab;
   }
 // zb+=yz;}
bc=bin2hex_(bc);
return bc;
}
//*************************************************************************
************
//*************************************************************************
************
//*****************************Sbox*****************************************
************
//*************************************************************************
*******Done
int s_box_value(int num)
```

```
{
int sbox[256] =   {
0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7,
0xab, 0x76,
0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4,
0x72, 0xc0,
0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8,
0x31, 0x15,
0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27,
0xb2, 0x75,
0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3,
0x2f, 0x84,
0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c,
0x58, 0xcf,
0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c,
0x9f, 0xa8,
0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff,
0xf3, 0xd2,
0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d,
0x19, 0x73,
0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e,
0x0b, 0xdb,
0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95,
0xe4, 0x79,
0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a,
0xae, 0x08,
0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd,
0x8b, 0x8a,
0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1,
0x1d, 0x9e,
0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55,
0x28, 0xdf,
```

```
0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54,
0xbb, 0x16 };

return sbox[num];

}

//******************
************************************************************

//************************
************************************************************

//***** ************************Inverse of S-box*********
****************************

//*************************************************************************
*******Done


int s_box_inverse(int num)

{

int rsbox[256] =

{ 0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3,
0xd7, 0xfb

, 0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde,
0xe9, 0xcb

, 0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa,
0xc3, 0x4e

, 0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b,
0xd1, 0x25

, 0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65,
0xb6, 0x92

, 0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d,
0x9d, 0x84

, 0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3,
0x45, 0x06

, 0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13,
0x8a, 0x6b
```

```
, 0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4,
0xe6, 0x73

, 0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75,
0xdf, 0x6e

, 0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18,
0xbe, 0x1b

, 0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd,
0x5a, 0xf4

, 0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80,
0xec, 0x5f

, 0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9,
0x9c, 0xef

, 0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53,
0x99, 0x61

, 0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21,
0x0c, 0x7d };

return rsbox[num];

}

//**************************************************************************
**********

//*********************Substitute
Bytes************************************************Done

String swap_with_sbox_value(String y)  // input is two character hex

                                // output is HEX

{

  char s,t;

  int u=0,v=0,uv=0;

  String yz="";

s=y[0];

u=hex2dec_(s);

//Serial.println("u:   "+u);
```

```
t=y[1];
v=hex2dec_(t);
//Serial.println("v:   "+v);
uv=u*16+v;
uv=s_box_value(uv);
yz=dec2hex_string(uv);
return yz;
}
//*************************************************************************
**********
//**************** ********** ************************************Done
String swap_with_inverse_sbox_value(String y)
 // input is two character hex*******************
                                    // Output is HEX
{
  char s,t;
  int u=0,v=0,uv=0;
  String yz="";
s=y[0];
u=hex2dec_(s);
t=y[1];
v=hex2dec_(t);
uv=u*16+v;
uv=s_box_inverse(uv);
yz=dec2hex_string(uv);

return yz;
```

```
}
//*********
**********************************************************************
//***********
**XOR**********************************************************************Done
String xor_(String xx,String yy) // it takes binary string inputs and returns binary string output
{
  int kk=0;
  String zz="";
kk=xx.length();
  for(int j=0;j<kk;j++)
  {
    if(xx[j]==yy[j]){zz+="0";}
    else if(xx[j]!=yy[j]) {zz+="1";}
  }
  return zz;
}
//**********************************************************************************
*********
//**************Generate
key*****************************************************Done
String generate_key(String previous_key, String dummy) // it takes HEX input and returns HEX output
{
  int index4=0,index2=0,index3=0,k=0,l=0;
  String
b1="",b2="",b3="",b4="",b5="",b6="",b7="",b8="",bb="",previous_key_bin=hex2bin_st
ring(previous_key),dummy_fifth=hex2bin_string(dummy);
```

```
 k=previous_key_bin.length();
 l=k/4;
 for(int j=0;j<l;j++)
 {
  b1+=previous_key_bin[j];
  index2=j+l;
  b2+=previous_key_bin[index2];
  index3=j+2*l;
  b3+=previous_key_bin[index3];
  index4=j+3*l;
  b4+=previous_key_bin[index4];
    }
 b5=xor_(b1,dummy_fifth);// it is actually function (b4) but, we have last 5 bytes=0000.
that is why I am taking dummy  only
 bb+=bin2hex_(b5);
 b6=xor_(b5,b2);
 bb+=bin2hex_(b6);
 b7=xor_(b6,b3);
 bb+=bin2hex_(b7);
 b8=xor_(b7,b4);
 bb+=bin2hex_(b8);
 return bb;
}
//*************************************************************************
**********
//*************************************************************************
********Done
```

```
String add_round_key(String x, String y)   //************bit key xor with modified data bits
                                        // it takes HEX strings and returns HEX string
{
  int gg=x.length(),hh=y.length(), kk=0;;
  String xx="", yy="",zz="";
  xx=hex2bin_string(x);
  yy=hex2bin_string(y);
  kk=xx.length();
  for(int j=0;j<kk;j++)
  {
    if(xx[j]==yy[j]){zz+="0";}
    else if(xx[j]!=yy[j]) {zz+="1";}
  }
  zz=bin2hex_(zz);
  return zz;
}
//*************************************************************************************
//*************************************************************************************
//************substitutes bytes with Rajindals S-box. Input is HEX string and returns HEX string
String substitute_bytes(String x)
{
  String y="",dummy="";
  int j=x.length();
  for(int i=0;i<j;i++)
```

```
 {
  dummy+=x[i];
  dummy+=x[i+1];
  i=i+1;
  y+=swap_with_sbox_value(dummy);
  dummy="";
 }
 return y;
}
//*************************************************************************************
//*************************************************************************************
//*****************************It calculates the counter input value for given number of rounds
String calculate_counter_value(int rounds)

{
 String initial_key="FEDCBA9876543210FEDCBA9876543210",x="0123456789ABCDEF0123456789ABCDEF";
  String dummy="ABCDEF01", new_key="",dummy1="0123456789ABCDEF0123456789ABCDEF";
   for(int i=0;i<rounds;i++)
 {
  new_key=generate_key(initial_key,dummy);// it takes HEX strings as input and returns HEX string as output
  dummy1=add_round_key(x,new_key);
 dummy1=substitute_bytes(dummy1);
 dummy1=shift_rows(dummy1);
```

```
 dummy1=mix_column(dummy1); //************* Not verified yet

//**********************************************************Serial.print("Ciphet
-text after ");
 //**********************************************************Serial.print(i+1);
 //**********************************************************Serial.println("
rounds:    "+ dummy1);
  initial_key=new_key;
  x=dummy1;
  }
  return x;
}
//*************************************************************************
************
//**********************************************************Increase binary string
size upto 128bits
String increase_size_upto_128(String x) // It checks the size of message and if, message
is less than 128bits it adds pre-zeroes
{
 int m=x.length();
 String y="";
 if (m==128){}
 else
 {
       m=128-m;
    for(int i=0;i<m;i++)
     {y+="0";}
     }
     y+=x;
```

```
    return y;
}
//*************************************************************************
**********
//*************************************************************************
***Encryption
String encrypt(String message, int rounds)
{
 String ciphertext="", counter_value="";
 counter_value=calculate_counter_value(rounds);
  message=hex2bin_string(message);
 message=increase_size_upto_128(message);
  counter_value=hex2bin_string(counter_value);
 ciphertext=xor_(counter_value,message);
 ciphertext=bin2hex_(ciphertext);
  return ciphertext;
 }
//*************************************************************************
**********
//*************************************************************************
***Decryption
String decrypt(String message, int rounds)
{
 String plaintext="", counter_value="";
 counter_value=calculate_counter_value(rounds);
   message=hex2bin_string(message);
  counter_value=hex2bin_string(counter_value);
 plaintext=xor_(counter_value,message);
```

```
 plaintext=bin2hex_(plaintext);
 return plaintext;
}
//*************************************************************************
***********
//*************************************************************************
**********
//*************************************************************************
***********
//***************Verification of message and
MAC*******************************Incomplete
//*********************************************************************Calcu
late 64bit MAC
String encrypt_with_MAC(String message1,int rounds)// It takes HEX string of <=32
characters, calculates MAC from plaintext and rounds+1
                                        //rounds of counter and returns concatenation of
ciphertext and MAC
{
 double j=0;
 int k=0;
 String message=message1, MAC="",counter_value_MAC="",message_value_MAC="",
ciphertext="", counter_value="";
 message=hex2bin_string(message);
 message=increase_size_upto_128(message);
 for (int i=0;i<128;i++)
 {
   j=i%2;
   if(j==0)
  {
    message_value_MAC+=message[i];
```

```
   }
  }
   counter_value=calculate_counter_value(rounds+1);
  counter_value=hex2bin_string(counter_value);
   for (int i=0;i<128;i++)
  {
    j=i%2;
    if(j==0)
   {
     counter_value_MAC+=counter_value[i];
   }
  }
   MAC=xor_(counter_value_MAC,message_value_MAC);
 MAC=bin2hex_(MAC);
  ciphertext=encrypt(message1,rounds);
  ciphertext+=MAC;
  return ciphertext;
}
//**************************************************************************
************
//**************************************************************************
************
String verify_MAC(String message1,int rounds)// It takes48 character HEX string,
extracts the MAC, then determines MAC from message decryption
                                    //and checks message authenticity
{
  String message="",
given_MAC="",calculated_MAC="",counter_value_MAC="",message_value_MAC="";
```

```
 String ciphertext="", counter_value="", verification_status="";
//**************************************Separate message and MAC
 for (int i=0;i<48;i++)
 {
  if (i<32)
  {
    message+=message1[i];
  }
  else if(i>31 && i<48)
  {
    given_MAC+=message1[i];
  }
 }
  //Serial.println("Given_MAC:                  "+given_MAC);
 Serial.println("Received_MAC:               "+given_MAC);
 //*********************************************
 message=decrypt(message,rounds);
 message=hex2bin_string(message);
 for (int i=0;i<128;i++)
 {
    if(i%2==0)
  {
    message_value_MAC+=message[i];
  }
 }
  counter_value=calculate_counter_value(rounds+1);
 counter_value=hex2bin_string(counter_value);
```

```
 for (int i=0;i<128;i++)
{
      if(i%2==0)
  {
    counter_value_MAC+=counter_value[i];
  }
}
 calculated_MAC=xor_(counter_value_MAC,message_value_MAC);

if (given_MAC==""){calculated_MAC="";}
else {calculated_MAC=bin2hex_(calculated_MAC);}
Serial.println("Calculated_MAC:              "+calculated_MAC);
if(calculated_MAC==given_MAC)
{
verification_status="Message is authentic.";
}
else if(calculated_MAC!=given_MAC)
{
verification_status="Message is suspicious";
}
 return verification_status;
}
//**************************************************************************
***********
//**************************************************************************
************
String decrypt_verified_message(String message1, int rounds) //It takes HEX string of 48
characters, extracts the message and decrypts
```

```
{
  String plaintext="", counter_value="", message="";
    for (int i=0;i<32;i++)
  {
    message+=message1[i];
 }
     counter_value=calculate_counter_value(rounds);
      message=hex2bin_string(message);
  counter_value=hex2bin_string(counter_value);
  plaintext=xor_(counter_value,message);
  plaintext=bin2hex_(plaintext);
  return plaintext;
}
//*************************************************************************
****
//*************************************************************************
****
//*****************Two-way communication
block*********************************
//*************************************************************************
****
void justBridgeMeshNetwork(String message)
{
  LoRa.beginPacket();                 // start packet
  LoRa.print(message);                // add payload
  LoRa.endPacket();                   // finish packet and send it
Serial.println("Mesage forwarded:  " + message);
}
```

```
void sendMessage(String message) {
  int rounds=10;
    Serial.println("pre-encrypt(plain_text):       "+ message);
    message=encrypt_with_MAC(message,rounds);
    Serial.println("ciphertext with MAC:           "+ message);
    ////////////////////////
    //Serial.println("Sending message:               " + message);
    String x=deviceAddress + receiverAddress, y=x + message;
    message=y;//deviceAddress+receiverAddress+message;
    Serial.println("Sending message:               " + message);
  LoRa.beginPacket();                   // start packet
  LoRa.print(message);                  // add payload
  LoRa.endPacket();                     // finish packet and send it
  msgCount++;                           // increment message ID
//***************************
 /* display.drawString (20, 20, "Sending packet:");
  display.drawString (110, 20, String (msgCount));
  display.display ();*/
  //***************************
}
void onReceive(int packetSize) {
  if (packetSize == 0) return;          // if there's no packet, return
  if(LoRa.available()){
  String complete_incoming_message="", incoming = "", sender_device_address="",
receiver_device_address="";
  int i=(deviceAddress.length());
  int dummy_address_counter=0,sender_address_length=i,receiver_address_length=i,
rdal=0,sdal=0;
```

```
int sr_address_length=i*2;
  while (LoRa.available()) {
    if (dummy_address_counter<sr_address_length)
    {
      if (dummy_address_counter<sender_address_length)
      {
        sender_device_address+= (char)LoRa.read();
        sdal=sender_device_address.length();
        dummy_address_counter+=1;
      }
  else
      {
        receiver_device_address+= (char)LoRa.read();
        rdal=receiver_device_address.length();
        dummy_address_counter+=1;
      }
    }
    if (dummy_address_counter>=sr_address_length)
    {
    incoming += (char)LoRa.read();
    }
  }
  ////////////////////////////////////
  if (receiver_device_address==deviceAddress)
  {
  int rounds=10;
  String verification_status="";
```

```
  Serial.println("ReceivedCiphertext with MAC is:"+ incoming);
  verification_status=verify_MAC(incoming,rounds);
  received_message_status=verification_status;
  if (verification_status=="Message is authentic.")
  {
  Serial.println("MAC verification status:       "+ verification_status);
  incoming=decrypt_verified_message(incoming,rounds);
  /*if((incoming[0]='0') && (incoming[1]='2'))
  {
    Serial.println("Message is from 0, for 2, and has been forwarded");
  sendMessage(incoming);
  }
  else*/
  Serial.println("Verified decrypted message is: "+ incoming);
  }
  else if(verification_status=="Message is suspicious")
  {
     //justBridgeMeshNetwork(incoming); //***************************** Added for MESH network**************
     Serial.println("Message verification status:   "+ verification_status);
     Serial.println("Message is suspicious so, no need to process:"+ incoming);
  }
  ////////////////////////////////
  //Serial.println("Received Message:              " + incoming);
 Serial.println("with RSSI: " + String(LoRa.packetRssi()));
 //Serial.println("Snr: " + String(LoRa.packetSnr()));
 Serial.println();
```

```
  //received_msg_count++;
}
//*********************
  /*display.drawString (20, 30, "Received packet:");
  display.drawString (110, 30, String (received_msg_count));
  display.display ();*/
//*********************
else if((receiver_device_address!=deviceAddress) &&
(rdal==(deviceAddress.length()))&&(sdal==(deviceAddress.length()))){
Serial.println("Message is not concerned to me");
String a=sender_device_address + receiver_device_address;
complete_incoming_message=a + incoming;
justBridgeMeshNetwork(complete_incoming_message);
}
//received_msg_count++;
}
received_msg_count++;
}
//*************************************************************************
************
//*************************************************************************
************
//***************Program_Main_Body*****************************************
**************
//*************************************************************************
************

void loop() {
  //*********************
```

```
    display.clear ();
    display.setTextAlignment (TEXT_ALIGN_LEFT);
    display.setFont (ArialMT_Plain_10);
    display.drawString (20, 0, "Inverter side LoRa kit");
    display.drawString (0, 10, "Sen id");
    display.drawString (30, 10, String(deviceAddress));
    display.drawString (62, 10, "/Rec id");
    display.drawString (94, 10, String(receiverAddress));
    //display.setFont (ArialMT_Plain_10);
    display.drawString (20, 20, "Sending packet:");
    display.drawString (110, 20, String (msgCount));
    display.drawString (20, 30, "Received packet:");
    display.drawString (110, 30, String (received_msg_count));
    display.drawString (10, 40, "Received message status:-");
    display.drawString (20, 50, String (received_message_status));
    display.display ();
  //**********************
 String incoming_message="";
if (millis()-lastSendTime >interval ) {
  //String message = "Hello world! I am Inverter side sender";   // send a message
  String message="0123456789ABCDEF0123456789ABCDEF", verification_status="";
     sendMessage(message);
  ////////////////////////
        lastSendTime = millis();            // timestamp the message
  interval = random(2000) + 1000;    // 2-3 seconds
  //if (LoRa.available())
}
```

```
 //if (LoRa.available())
    onReceive(LoRa.parsePacket());
     digitalWrite (2, HIGH); // turn the LED on (HIGH is the voltage level)
      //delay(interval);
  //onReceive(LoRa.parsePacket());
}
```

Appendix D *Configuring SD card and logging data locally*

```
#include "FS.h"
#include "SD.h"
#include "SPI.h"
#define MISO 02
#define SCK 14
#define MOSI 15
#define CS 13
#define D1 04
void listDir(fs::FS &fs, const char * dirname, uint8_t levels){
    Serial.printf("Listing directory: %s\n", dirname);
    File root = fs.open(dirname);
    if(!root){
        Serial.println("Failed to open directory");
        return;
    }
    if(!root.isDirectory()){
        Serial.println("Not a directory");
        return;
    }
    File file = root.openNextFile();
```

```
    while(file){
      if(file.isDirectory()){
        Serial.print("  DIR : ");
        Serial.println(file.name());
        if(levels){
          listDir(fs, file.name(), levels -1);
        }
      } else {
        Serial.print("  FILE: ");
        Serial.print(file.name());
        Serial.print("  SIZE: ");
        Serial.println(file.size());
      }
      file = root.openNextFile();
    }
}
void createDir(fs::FS &fs, const char * path){
    Serial.printf("Creating Dir: %s\n", path);
    if(fs.mkdir(path)){
      Serial.println("Dir created");
    } else {
      Serial.println("mkdir failed");
    }
}
void removeDir(fs::FS &fs, const char * path){
    Serial.printf("Removing Dir: %s\n", path);
    if(fs.rmdir(path)){
```

```
        Serial.println("Dir removed");
    } else {
        Serial.println("rmdir failed");
    }
}
void readFile(fs::FS &fs, const char * path){
    Serial.printf("Reading file: %s\n", path);
    File file = fs.open(path);
    if(!file){
        Serial.println("Failed to open file for reading");
        return;
    }
    Serial.print("Read from file: ");
    while(file.available()){
        Serial.write(file.read());
    }
    file.close();
}
void writeFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Writing file: %s\n", path);
    File file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }
    if(file.print(message)){
        Serial.println("File written");
```

```
    } else {
        Serial.println("Write failed");
    }
    file.close();
}
void appendFile(fs::FS &fs, const char * path, const char * message){
    Serial.printf("Appending to file: %s\n", path);
    File file = fs.open(path, FILE_APPEND);
    if(!file){
        Serial.println("Failed to open file for appending");
        return;
    }
    if(file.print(message)){
        Serial.println("Message appended");
    } else {
        Serial.println("Append failed");
    }
    file.close();
}
void renameFile(fs::FS &fs, const char * path1, const char * path2){
    Serial.printf("Renaming file %s to %s\n", path1, path2);
    if (fs.rename(path1, path2)) {
        Serial.println("File renamed");
    } else {
        Serial.println("Rename failed");
    }
}
```

```
void deleteFile(fs::FS &fs, const char * path){
   Serial.printf("Deleting file: %s\n", path);
   if(fs.remove(path)){
      Serial.println("File deleted");
   } else {
      Serial.println("Delete failed");
   }
}
void testFileIO(fs::FS &fs, const char * path){
   File file = fs.open(path);
   static uint8_t buf[512];
   size_t len = 0;
   uint32_t start = millis();
   uint32_t end = start;
   if(file){
      len = file.size();
      size_t flen = len;
      start = millis();
      while(len){
         size_t toRead = len;
         if(toRead > 512){
            toRead = 512;
         }
         file.read(buf, toRead);
         len -= toRead;
      }
      end = millis() - start;
```

```
        Serial.printf("%u bytes read for %u ms\n", flen, end);
        file.close();
    } else {
        Serial.println("Failed to open file for reading");
    }
    file = fs.open(path, FILE_WRITE);
    if(!file){
        Serial.println("Failed to open file for writing");
        return;
    }
    size_t i;
    start = millis();
    for(i=0; i<2048; i++){
        file.write(buf, 512);
    }
    end = millis() - start;
    Serial.printf("%u bytes written for %u ms\n", 2048 * 512, end);
    file.close();
}
void setup(){
    Serial.begin(9600);
    //SPI.begin(14,2,15);
    SPI.begin(SCK,MISO,MOSI,CS);
   // digitalWrite(13,LOW);
   // LoRa.setPins(SS,RST,DI0);
    //if(!SD.begin(13)){
      if(!SD.begin(CS)){
```

```
    Serial.println("Card Mount Failed");
    return;
}
uint8_t cardType = SD.cardType();
if(cardType == CARD_NONE){
    Serial.println("No SD card attached");
    return;
}
Serial.print("SD Card Type: ");
if(cardType == CARD_MMC){
    Serial.println("MMC");
} else if(cardType == CARD_SD){
    Serial.println("SDSC");
} else if(cardType == CARD_SDHC){
    Serial.println("SDHC");
} else {
    Serial.println("UNKNOWN");
}
uint64_t cardSize = SD.cardSize() / (1024 * 1024);
Serial.printf("SD Card Size: %lluMB\n", cardSize);
listDir(SD, "/", 0);
createDir(SD, "/mydir");
listDir(SD, "/", 0);
removeDir(SD, "/mydir");
listDir(SD, "/", 2);
writeFile(SD, "/hello.txt", "Hello ");
appendFile(SD, "/hello.txt", "World!\n");
```

```
    readFile(SD, "/hello.txt");
    deleteFile(SD, "/foo.txt");
    renameFile(SD, "/hello.txt", "/foo.txt");
    readFile(SD, "/foo.txt");
    testFileIO(SD, "/test.txt");
    Serial.printf("Total space: %lluMB\n", SD.totalBytes() / (1024 * 1024));
    Serial.printf("Used space: %lluMB\n", SD.usedBytes() / (1024 * 1024));
}
void loop(){
}
```

Appendix E *Send email under abnormal conditions*

```
#include <WiFi.h>

WiFiClient client;

String MakerIFTTT_Key ;

String MakerIFTTT_Event;

char *append_str(char *here, String s) {  int i=0; while (*here++ = s[i]){i++;};return here-1;}

char *append_ul(char *here, unsigned long u) { char buf[20]; return append_str(here, ultoa(u, buf, 10));}

char post_rqst[256];char *p;char *content_length_here;char *json_start;int compi;

void setup()

{

 Serial.begin(9600);

 WiFi.disconnect();

 delay(3000);

 Serial.println("Starting to connect");

  WiFi.begin("BELLALIANT0351","MXPEC57EK31N88GG");

 while ((!(WiFi.status() == WL_CONNECTED))){

   delay(300);

 }

 Serial.println("Connected!");

 if (client.connect("maker.ifttt.com",80)) {

   MakerIFTTT_Key ="dnJOkb5Yd5K7nPcAoo1N4OWx4S-tCpMaazTp3Yun7Z0";

   MakerIFTTT_Event ="Email";
```

```
  p = post_rqst;
  p = append_str(p, "POST /trigger/");
  p = append_str(p, MakerIFTTT_Event);
  p = append_str(p, "/with/key/");
  p = append_str(p, MakerIFTTT_Key);
  p = append_str(p, " HTTP/1.1\r\n");
  p = append_str(p, "Host: maker.ifttt.com\r\n");
  p = append_str(p, "Content-Type: application/json\r\n");
  p = append_str(p, "Content-Length: ");
  content_length_here = p;
  p = append_str(p, "NN\r\n");
  p = append_str(p, "\r\n");
  json_start = p;
  p = append_str(p, "{\"value1\":\"");
  p = append_str(p, "aikhichi7@gmail.com");
  p = append_str(p, "\",\"value2\":\"");
  p = append_str(p, "Sender is ESP32");
  p = append_str(p, "\",\"value3\":\"");
  p = append_str(p, "Hello World: I am ESP32");//"Hello World: I am ESP32 I think we
can");
  p = append_str(p, "\"}");
  compi= strlen(json_start);
  content_length_here[0] = '0' + (compi/10);
  content_length_here[1] = '0' + (compi%10);
  client.print(post_rqst);
  Serial.println("Email sent");
 }
```

```
}
void loop()
{
 }
```

Appendix F *LoRa Simple server for Dragino-yun using RF95*

```
#include <Console.h>
#include <SPI.h>
#include <RH_RF95.h>
// Singleton instance of the radio driver
RH_RF95 rf95;
int led = A2;
float frequency = 915.0; //868.0;
void setup()
{
  pinMode(led, OUTPUT);
  Bridge.begin(BAUDRATE);
  Console.begin();
  while (!Console) ; // Wait for console port to be available
  Console.println("Start Sketch");
  if (!rf95.init())
    Console.println("init failed");
  // Setup ISM frequency
  rf95.setFrequency(frequency);
  // Setup Power,dBm
  rf95.setTxPower(13);
```

```
  // Setup Spreading Factor (6 ~ 12)
  rf95.setSpreadingFactor(7);
  // Setup BandWidth, option:
7800,10400,15600,20800,31200,41700,62500,125000,250000,500000
  rf95.setSignalBandwidth(125000);
  // Setup Coding Rate:5(4/5),6(4/6),7(4/7),8(4/8)
  rf95.setCodingRate4(5);
  Console.print("Listening on frequency: ");
  Console.println(frequency);
}
void loop()
{
  if (rf95.available())
  {
    // Should be a message for us now
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    if (rf95.recv(buf, &len))
    {
      digitalWrite(led, HIGH);
      RH_RF95::printBuffer("request: ", buf, len);
      Console.print("got request: ");
      Console.println((char*)buf);
      Console.print("RSSI: ");
      Console.println(rf95.lastRssi(), DEC);
      // Send a reply
      uint8_t data[] = "And hello back to you";
```

```
      rf95.send(data, sizeof(data));
      rf95.waitPacketSent();
      Console.println("Sent a reply");
      digitalWrite(led, LOW);
    }
    else
    {
      Console.println("recv failed");
    }
  }
}
```

Appendix G *Dragino-Yun as a Gateway with partial_decryption*

```
#include <SPI.h>
//#include <RH_RF95.h>
#include <LoRa.h> //**********
#include <String.h>
#include <Console.h>
#include "ThingSpeak.h"
#include "YunClient.h"
#include <math.h>
YunClient client;
//RH_RF95 rf95;
#define BAUDRATE 115200
unsigned long myChannelNumber = 578314;//20xx93;
const char * myWriteAPIKey = "76TZK7YU7PYR4PEY";//"B9ZxxxxxNVEBKIFY";
float frequency = 915.0;//868.0;
void setup()
{
   Bridge.begin(BAUDRATE);
   LoRa.begin(915E6);//**********
   LoRa.end();//**********
   Serial.begin(BAUDRATE);//**********
   //Console.begin();// Don't use Console here, since it is conflict with the ThinkSpeak library.
```

```
   ThingSpeak.begin(client);
   if(!LoRa.begin(915E6)) //***************
   { Serial.println("Starting LoRa failed!");
   while (1);
   }
   Serial.println("LoRa setup configured successfully!");
}
//////////////////////////////////////////
String incoming="";
void onReceive(int packetSize) {
  if (packetSize == 0) return;          // if there's no packet, return
 //x= millis();
 //Serial.println(x);
  //String incoming = "";
  incoming = "";
if(LoRa.available()){
  while (LoRa.available()) {
    incoming += (char)LoRa.read();
  }
}
}
////////////////////////////////////////////////
void loop()
{
  //int h =10;// newData[0];
    //               int t =15;// newData[1];
  incoming = "";
```

```
  onReceive(LoRa.parsePacket());
   long h=calculate_voltage(incoming),i=calculate_current(incoming),
j=calculate_power(incoming);
  //if(incoming[0]="A") {h=5;}
  //if(incoming[1]="B") {t=0;}
     //int h =10;// newData[0];
                 // int t =15;// newData[1];
                  //
                  ThingSpeak.setField(1,h);
                  ThingSpeak.setField(2,i);
                  ThingSpeak.setField(3,j);
                  ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
     delay(3000);
}
int hex2dec_(char xy)
{
  int bi=0;
   if (xy=='0') {  bi=0; }
    else if (xy=='1') { bi=1; }
    else if (xy=='2') { bi=2; }
    else if (xy=='3') { bi=3; }
    else if (xy=='4') { bi=4; }
    else if (xy=='5') { bi=5; }
    else if (xy=='6') { bi=6; }
    else if (xy=='7') { bi=7; }
    else if (xy=='8') { bi=8; }
    else if (xy=='9') { bi=9;}
```

```
   else if (xy=='a'||xy=='A') {  bi=10; }
   else if (xy=='b'||xy=='B') {  bi=11; }
   else if (xy=='c'||xy=='C') {  bi=12; }
   else if (xy=='d'||xy=='D') {  bi=13; }
   else if (xy=='e'||xy=='E') {  bi=14; }
   else if (xy=='f'||xy=='F') {  bi=15; }
 return bi;
}
long calculate_voltage(String incoming_)
{
 int a=0, b=0, c=0, d=0;
  a=hex2dec_(incoming_[8]);
  b=hex2dec_(incoming_[9]);
  c=hex2dec_(incoming_[10]);
  //d=hex2dec_(incoming_[11]);
  long volt=0,a_=(a*256),b_=(b*16),c_=c;//*16^1),d_=d;
volt=a_+b_+c_;//+d_;
return volt;
}
long calculate_current(String incoming_)
{
  int a=0, b=0, c=0, d=0;
  a=hex2dec_(incoming_[12]);
  b=hex2dec_(incoming_[13]);
  c=hex2dec_(incoming_[14]);
  d=hex2dec_(incoming_[15]);
  long ampere=0,a_=(a*256),b_=(b*16),c_=c;//*16),d_=d;
```

```
ampere=a_+b_+c_;//+d_;
return ampere;
}
long calculate_power(String incoming_)
{
int a=0, b=0, c=0, d=0;
  a=hex2dec_(incoming_[16]);
  b=hex2dec_(incoming_[17]);
  c=hex2dec_(incoming_[18]);
  d=hex2dec_(incoming_[19]);
  long power=0,a_=(a*256),b_=(b*16),c_=c;//*16),d_=d;
power=a_+b_+c_;//+d_;
return power;
}
```

Appendix H *ESP_Collector Dragino Side*

```
#include <SPI.h>
#include <LoRa.h>
#include<Arduino.h>
#define SS      18
#define RST     14
#define DI0     26
#define BAND    915E6  //915E6 --
String outgoing;              // outgoing message
          // count of outgoing messages
long lastSendTime = 0;        // last send time
int interval = 500;          // interval between sends
void setup() {
  Serial.begin(115200);
  while (!Serial); //It is just the the basic function, must connect to a computer
  SPI.begin(5,19,27,18);
   //Serial.println("ESP_Collector Dragino side");
  if (!LoRa.begin(BAND)) {
    //Serial.println("Error: LoRa configuring failed!");
    while (1);
  }
  //Serial.println("LoRa setup configured successfully!");
}
void sendMessage(String message) {
  //Serial.println("Sending message:               " + message);
```

```
  LoRa.beginPacket();                 // start packet
  LoRa.print(message);                // add payload
  LoRa.endPacket();                   // finish packet and send it
  //msgCount++;                       // increment message ID
}
void onReceive(int packetSize) {
  String incoming = "";
if(LoRa.available()){
  while (LoRa.available()) {
    incoming += (char)LoRa.read();
  }
  }
}
void loop() {
  String received="";
while(Serial.available()>0){
    received=Serial.read();
  }
   sendMessage(received);
      }
```

Appendix I *Configuring ESP32 as LoRa Gateway*

Its code can be downloaded from

https://github.com/things4u/ESP-1ch-Gateway-v5.0

And can be modified to configure ESP32 as gateway after watching four youtube videos available at

[1] https://www.youtube.com/watch?v=v3JIx7ICmnI

[2] https://www.youtube.com/watch?v=OfH80Gmgf-o

[3] https://www.youtube.com/watch?v=eZhDvsJzwwI

[4] https://www.youtube.com/watch?v=sYPL85ViFPE