Energy Monitoring System Over LoRaWan Network

Bachelor Thesis

German University in Cairo

Information Engineering and Technology Faculty

Author:

Omar Emam

Supervisors:

Assoc. Prof. Tallal Elshabrawy

TABLE OF CONTENTS

LIST OF FIGURES

SYMBOLS

LoRaWan : Long Range Wide area network

Twu : LoRaWan transceiver wake up period

Iwu : LoRaWan transceiver wake up state current

Tpre : LoRaWan transceiver radio preparation period

Ipre : LoRaWan transceiver radio preparation state current

Ttx : LoRaWan transceiver transmission period

Itx : LoRaWan transceiver transmission state current

Tw1w : LoRaWan transceiver wait first window period

Iw1w : LoRaWan transceiver wait first window state current

Trx1w : LoRaWan transceiver receiving first window period

Irx1w : LoRaWan transceiver receiving first window state current

Tw2w : LoRaWan transceiver second window period

Iw2w : LoRaWan transceiver second window state current

Trx2w : LoRaWan transceiver second receive window period

Irx2w : LoRaWan transceiver second receive window state current

Toff : LoRaWan transceiver radio off period

Ioff : LoRaWan transceiver radio off state current

Tpost : LoRaWan transceiver post-processing period

Ipost : LoRaWan transceiver post-processing state current

Tseq : LoRaWan transceiver turnoff sequence period

Iseq : LoRaWan transceiver turnoff sequence state current

Tsleep : LoRaWan transceiver sleep period

Isleep : LoRaWan transceiver sleep state current

SFD : Start frame delimiter

CRC : Cyclic redundancy check

JSON : Javascript Object Notation

RESTApi : Representational state transfer application programming interface

NPM : Node package manager

ABSTRACT

Internet of Things can be used for automating processes which might seem inconvenient for humans to carry out. A growing number of corporates and entities are utilizing the Internet of Things technologies. In addition, Internet of Things can be used with the LoRaWan technology expanding the capabilities and removing the limitations which some Internet of Things applications might face since LoRaWan provides a reliable Low power wide area network for the system.

In this thesis, an electrical energy monitoring system is created utilizing Internet of Things along with LoRaWan technology. This thesis is targeting entities wanting to monitor geographically isolated devices energy consumption and checking for abnormalities in the consumption and taking action upon such devices. This application can decrease the overall power consumption and allow longevity of the measured devices allowing firms decrease costs of operation.

1. INTRODUCTION

1.1 Motivation

Nowadays several businesses and entities working in different sectors tend to look for different ways to decrease their costs of operation, avoid possible asset breakdowns and increase workplace safety measures. This thesis will discuss a system that can make such tasks easier without having to worry about providing internet connection to the IoT chips and sensors monitoring the device(s) operation or providing any human intervention. When using LoRaWan as a method of communication between two IoT chips a secure and long-range transfer of bits is guaranteed as it can transfer data across [Ref:1] 20 Km's but on average it travels 1-5 Km's according to the SF deployed upon the transmission with interference taken into account. [Ref:2] [Ref:3] The higher the SF the lower the bit transfer rate. However, in this thesis high bit transfer rate is not required since it is not critical that the operation is executed rapidly. Moreover, the bits needed for transfer per unit time does not exceed a few bits. In addition, due to the low power consumption nature of LoRaWan the lifespan of such system is much longer than any other system relying on different information transfer approaches. That is why LoRaWan fits the system being discussed throughout this thesis.

1.2 Aim of the project

The aim of this thesis is to discuss whether implementation of such system is possible and reliable and to implement a system which can be used to measure electrical current of devices existing in a geographically isolated area without the need of providing an internet connection or any sort of a wired connection.

2. BACKGROUND

2.1 Low power wide area Network (LoRaWan)

2.1.1 Definition and history

LoRaWan is a high capacity, long range, open, low power wide area network operating on the ISM bandwidth which started in 2010 by a company called Cycleo which was later acquired by Semtech in 2012. The aim of introducing such technology is to facilitate network connections which might seem geographically out of reach connecting different nodes of the network without consuming too much power to transmit data.

2.1.2 Operation

LoRaWan uses a CSS, there are wide range of SF's and bandwidths which can be used to best optimize the LoRaWan operation according to the factors which could affect the system application such as distance, interference, desired bit transfer rate, and desired bit error rate.[Ref:3] ISM bands used are 433MHz – 868MHz or 915 MHz The SF and frequency could be fine-tuned to match the application specifications. Circumstances where longer range is required SF is increased although it trades off the bit transfer rates. On the other hand, applications requiring a certain degree of real-time transfer will look to decrease SF as much as possible. Increasing SF also increases the bit error rate.

2.1.3 Performance

LoRaWan excels in providing long range signals over an open area with no or few obstacles. However, given an obstacle [Ref:3] LoRaWan failed to bypass a mountain due to less wave diffraction wave abilities. However, in some cases LoRaWan delivered a signal where GPS sometimes failed to deliver.

However, As SF increase the BER increase as shown in Figure (1) [Ref:4]. In addition, due to the nature of the LoRaWan being a pure ALOHA network it shows a correlation between number of devices increasing and packet loss [Ref:4].

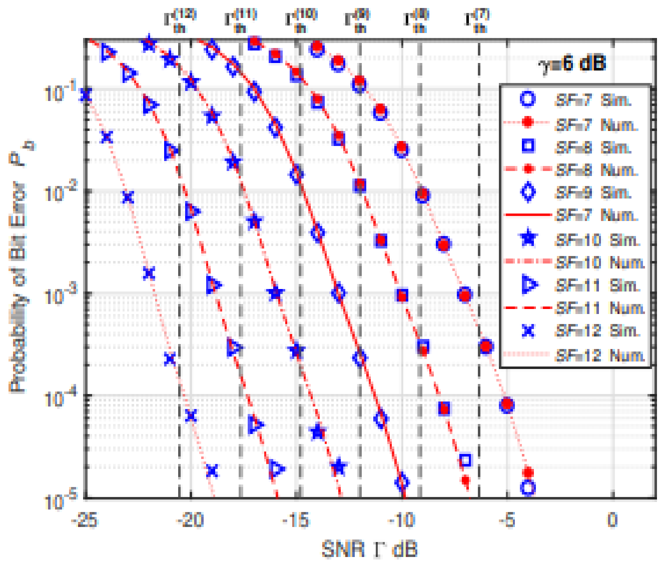


Fig. 2.1. BER and SF

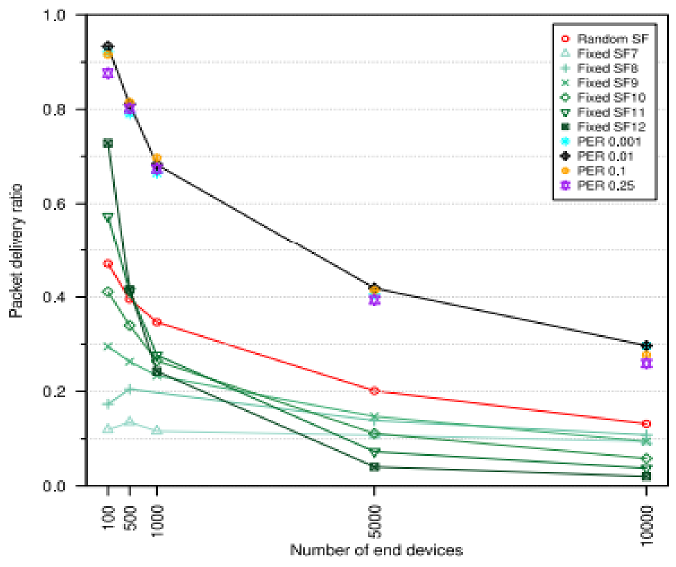


Fig. 2.2. BER and no. of end devices

2.1.4 LoRaWan Energy Consumption

This section will discuss the energy consumption parameters of the LoRaWan transceivers such as the transceiver current consumption, lifespan and energy efficiency of information delivery.

Unacknowledged transmission energy consumption

There is a BER on the transmission of any data across the LoRaWan network. In unacknowledged transmission energy consumption parameters is independent of the BER since the system is not aware of faulty transmissions therefore not retransmitting any further data. [Ref:7]

Acknowledged transmission energy consumption

If the system is an acknowledged transmission system the energy consumption parameters will often be different because of the re-transmissions after any failure of data transfer.

How are energy parameters measured?

A periodic behaviour is assumed for the LoRaWan transceivers where energy parameters are measured on each period. The transceivers usually exist in 11 states, the difference between energy parameters of the 11 states are found to be negligible to some extent of accuracy. [Ref:7]

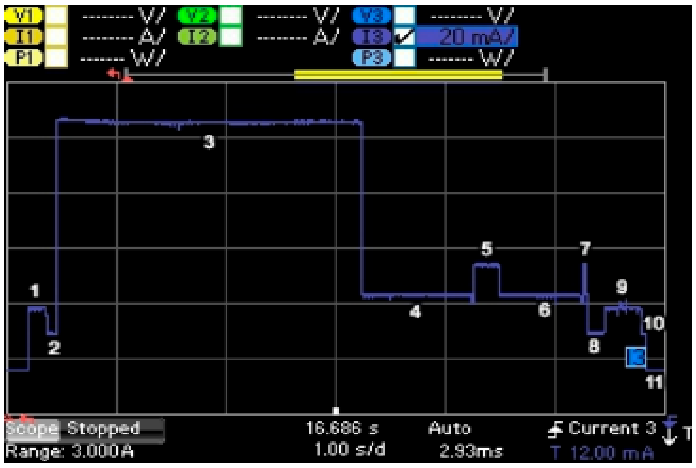


Fig. 2.3. Current reading of a LoRaWan transceiver

This thesis discusses a system which uses no re-transmissions due to the fact that there is no sensitive data to be lost. The system transmitter only uses 6 states which are wake up state, radio preparation state, transmission state, wait first window state, radio off state and sleep state and the receiver uses 6 states which are wake up state, radio preparation state, receiver first window, radio off state, post-processing state and sleep state

Table 2.1

Electrical currents by states

Period	**Value (ms)**	**Current parameter (mA)**	**Value**
Twu	168.2	Iwu	22.1
Tpre	83.8	Ipre	13.3
Ttx	SF dependent	Itx	83.0
Tw1w	983.3	Iw1w	27.0
Trx1w	dependent	Irx1w	38.1
Tw2w	dependent	Iw2w	27.1
Trx2w	33.0	Irx2w	35.0
Toff	147.4	Ioff	13.2
Tpost	268.0	Ipost	21.0
Tseq	38.6	Iseq	13.3
Tsleep	dependent	Isleep	45x10^{-}3

Summation of transmitter state periods

The wake up period will not be added because it only occurs once so it can be neglected

Ttx $= 2^{SF}/BW * SF$

$Ttxtot = Tpre + Ttx + Tw1w + Toff + Tsleep$

Summation of transmitter state currents per period

Itot = Ipre + Itx + Iw1w + Ioff + Isleep

Therefore the electrical current consumed by the LoRaWan transceiver functioning as a transmitter is about 136.5 mA per period of transmission according to Table(2.1) [Ref:7] values.

Summation of receiver state periods

The wake up period will not be added because it only occurs once so it can be neglected
Trxtot = Tpre + Trx1w + Toff + Tpost + Tsleep

Summation of receiver state currents per period

Itot = Ipre + Irx1w + Ioff + Ipost + Isleep
Therefore the electrical current consumed by the LoRaWan transceiver functioning as a receiver is about 85.6 mA per period of receiving according to Table(2.1) [Ref:7] values.

2.1.5 LoRa packet structure

The LoRa packet structure consists of 4 parts each part has a distinct functionality the parts are preamble, header, payload and payload CRC. [Ref:8]

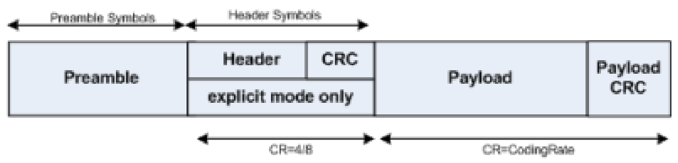


Fig. 2.4. LoRa packet structure

Preamble

A LoRa packet starts with a preamble part which is a 7-octet (56 bits) of alternating 1's and 0's plus one-octet of SFD the preamble is used for synchronization of the receiver clock.

Header

A LoRa packet header contains information about the packet such as length of packet, synchronization bits and advertisement which helps in synchronization of transmitter and receiver.

Payload

The actual data which is sent in a packet is the payload, this is the only data which should be processed by the receiver since other data is concerned with the network setup and synchronization.

Payload CRC

CRC is a code used for error detection which is used in digital networks to find and correct accidental changes in data.

2.2 Current sensor ACS712

2.2.1 Setting up the sensor

In order for the current sensor to operate a 5V DC supply should be given to the sensor VCC pin and GND (Ground) pin. And then the device which is being measured should be wired and passed through the measuring nodes of the sensor.

2.2.2 Sensor readings analogy

The sensor output pin outputs the value of the current of the connected device by a proportional voltage to the current measured. The sensor current measurement range is –5A to 5A and outputs a voltage respective to the current range of 1.5 to

3.4 V meaning if current is 0 output voltage is 2.45V, if current is 5 output voltage is 3.4V and if current is –5 output voltage is 1.5V.

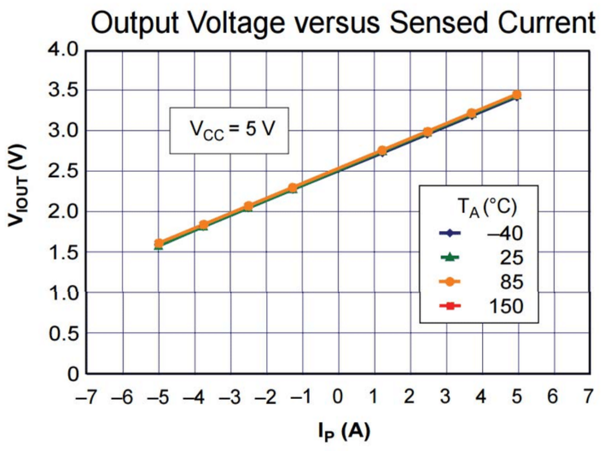


Fig. 2.5. Current vs Voltage ACS712

2.3 NoSQL Database

2.3.1 Definition

NoSQL database provides a mean of storing and retrieving data designed for unstructured data. The database operations are communicated to the database as a form of a query which is understood by the database.

2.3.2 NoSQL vs SQL

SQL is a relational database which stores values in a tabular fashion and different tables can be connect by indexes of the tables. SQL databases are more secure and better structured than NoSQL databases. However, the nature of NoSQL allows easier horizontal scaling of the database which is more convenient to certain applications.

2.3.3 MongoDB

How is MongoDB structured?

Like most NoSQL databases, MongoDB consists of documents and collections where documents are a collection of key-value pair variables similar to JSON and these variables as a convention are somewhat related to be included in the same document each document is indexed by a unique key which is commonly denoted as "id" this key is usually used throughout development to find certain documents when needed.
On the other hand, collections are entities that store documents having same attributes, when a search for a document is carried out, the collection in which the document exists must be specified. The database is a collection of collections each must have its distinct schema.
A collection of databases can be created on the same cluster and different servers allowing easier horizontal scaling of data since it takes care automatically of load balancing and redistributing documents [Ref:10].

Atlas and MongoDB

Atlas is a cloud database service specifically designated for MongoDB. Most of the time deploying a database or a server on a cloud needs management of database which can be inconvenient for some developers and teams. Atlas provides a fully automated deployment and management of MongoDB.

2.4 Thingsboard

2.4.1 Definition

Thingsboard is a designated dashboard which can be structured to simulate any entity such as a building or departments and assets can be placed into these entities these assets can be devices. Each device can be given a set of graphs of measurement of the device readings and they are stored and displayed according to the display of choice.

2.4.2 Recording a reading on Thingsboard

Given a set of devices registered on Thingsboard each device is given an access token which can be used in an HTTP request to record reading to the corresponding device access token. An example of the HTTP request would be curl -v -X POST -d ”ẗemperature ̈: 25” https://demo.thingsboard.io/api/v1/ACCESSTOKEN/telemetry –header ”Content-Type:application/json”

2.4.3 Rule chains Thingsboard

Rule chains on Thingsboard is a framework in which certain actions can be specified on the occurrence of events recorded in Thingsboard. Figure(2.6) shows an example of a rule chain in which a notification is sent to a specified email upon occurrence of an undesired event.

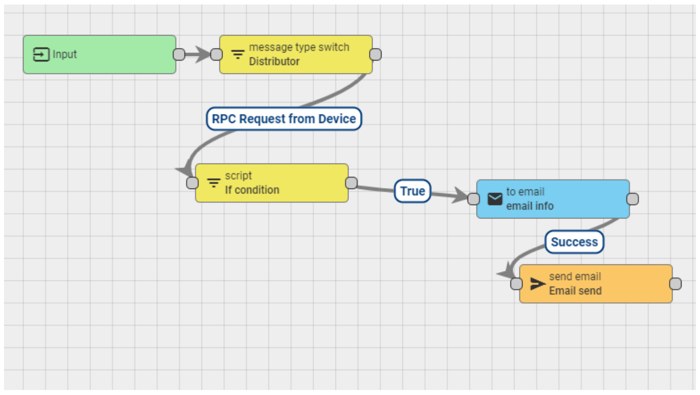


Fig. 2.6. Thingsboard rule chains

2.5 Network edge server

2.5.1 Definition

An edge server is a device (usually a computer) connected to an offline network from one end and the internet from the other end. A server is an entity behind the logic in a network, it specifies certain functions which gets called according to the need of the network.

2.5.2 RESTapi (Representational State Transfer)

RESTapi is a method of communication between clients and servers by using HTTP protocol. In order for the client to specify the type of service required from the server. The client and server communicate using RESTapi. First the server deploys a set of functions then each function is being related to a certain HTTP request with a certain URL and method (GET, POST, PUT, DELETE) and HTTP-Request body and HTTP-Request headers.

2.5.3 Serial communication ports

Serial communication ports in a computer is an interface of data transfer through wired connections which transfers data bit by bit. This form of communication is efficient when there is a limited number of devices communicating to the network edge server or when there is one type of data being transferred.

2.5.4 Parallel communication ports

On the other hand, parallel communication ports in a computer transfers multiple bits at a time through wired connections this is an effective way of handling multiple readings at one time instance, some readings and applications require two or more readings to be reported simultaneously.

2.5.5 Arduino

Arduino is an open source software which is used to program programmable chips and electronic circuits. Since Arduino is an open source software, there are so many libraries which can be imported to facilitate the work of the programmer. In addition, the Arduino community is helpful, rich and full of useful documentations and guides which is a huge plus for Arduino in general.

Programming language of Arduino

Arduino uses C/C++ functions to be coded and compiled. Arduino is based on two functions start and loop functions, start function gets called at the beginning of operation, then after the start function is executed, loop function executes after each period or delay specified and according to the code written in it.

2.5.6 NodeJS

NodeJS is a run-time environment based on JavaScript which you can write a multi-purpose code. When used with ExpressJS its mainly used for backend development. The nature of ExpressJS code structure allows the developer to invest more efficient time on the logic part of the code allowing this library to be a favorite for many developers.

How does ExpressJS allow easier development?

ExpressJS allows easier and cleaner code since it does all the tough work regarding analyzing the HTTP packet and carrying out functions concerning it. Moreover, it allows refactoring of handler functions in much smaller understandable and write-able pieces [Ref:9].

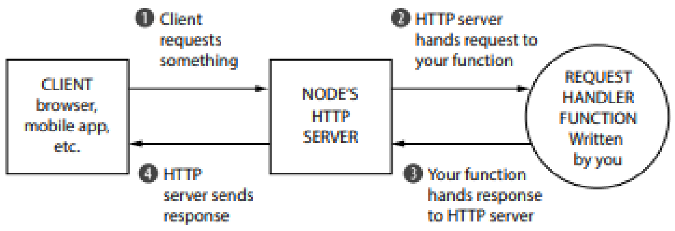

❶ Client requests something
❷ HTTP server hands request to your function
CLIENT browser, mobile app, etc.
NODE'S HTTP SERVER
REQUEST HANDLER FUNCTION Written by you
❹ HTTP server sends response
❸ Your function hands response to HTTP server

Fig. 2.7. NodeJS as a server flow

Notice how an extra hop is added to the diagram this extra hop facilitates the programmers work by allowing the middlewares to exist.

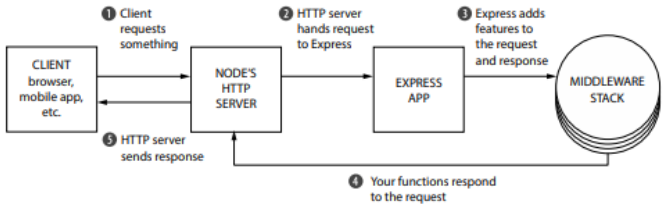
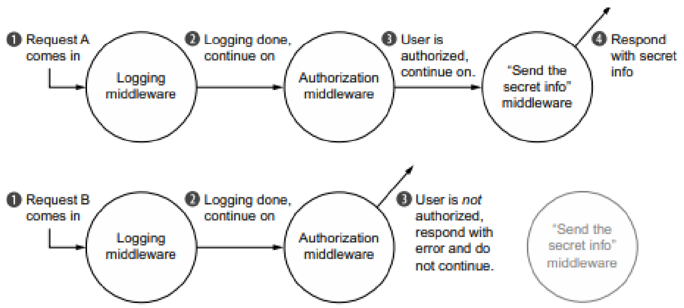

❶ Client requests something
❷ HTTP server hands request to Express
❸ Express adds features to the request and response
CLIENT browser, mobile app, etc.
NODE'S HTTP SERVER
EXPRESS APP
MIDDLEWARE STACK
❺ HTTP server sends response
❹ Your functions respond to the request

Fig. 2.8. NodeJS as a server flow with ExpressJS

Middlewares are functions which can be used as a better substitute to a single HTTP-request handler function. Middlewares allow clean and neat structuring of the code required to run after a certain HTTP-request rather than calling one handler function to do all the work, several middlewares are called to do different jobs this allows re-usability of the middlewares and before sending back the response to the client-side.



1 Request A comes in
Logging middleware
2 Logging done, continue on
Authorization middleware
3 User is authorized, continue on.
"Send the secret info" middleware
4 Respond with secret info
1 Request B comes in
Logging middleware
2 Logging done, continue on
Authorization middleware
3 User is not authorized, respond with error and do not continue.
"Send the secret info" middleware

Fig. 2.9. Middlewares example

2.5.7 Serial port in NodeJS

Serial port library in NodeJS allows the run-time environment of NodeJS to access the serial port on the computer so that it can be used to access the data sent over the serial port.

3. SMART CITY LORAWAN SCALING

In this chapter the scaling of the LoRaWan system being implemented in this thesis is discussed. In order for a city to be considered a smart city connectivity between all the components in the city which needs to be smart must be stable and reliable and not energy consuming and that is what LoRaWan provides.

3.0.1 What is a smart city?

A smart city is a city which uses IoT and sensors to collect data from its designated spots and automate certain operations done. The modernizing of roads, transportation systems, industries and agriculture, the data collected can be used to optimize resources and handle them efficiently as well as automating certain processes upon the data.

3.0.2 How can the system scale?

The system implemented and discussed in this thesis is a prototype which consists of one LoRa end device and one LoRa gateway connected to one device from one end and the network edge server from the other end. The system is very scalable since the next nodes can always serve many previous nodes. There are several ways the system can scale in.

Many to One, End devices to LoRa gateways

Since one LoRa gateway can support many end devices to report and send data the network can scale from the end devices part given there is only one LoRa gateway the end devices can be thousands [Ref:4]. However, as shown figure(2.2) the increase of end devices above a certain limit can lead to an exceptionally low packet delivery ratio [Ref:4], this can lead to a need of increasing the number of LoRa gateways.

3.0.3 Many to Many, End devices to LoRa gateways

According to the sensitivity of data being transmitted from the end devices and the need for high success rate of packet transmission the ratio between end devices and LoRa gateways is adjusted as in figure(2.2).

Increasing the number of network edge servers

If the network architecture scaled up to hold an entire city there needs to be multiple servers running in the network. Understanding the limitations and concurrency loads of the server being deployed is the key for knowing the ratio required to handle the LoRa gateways numbers. In addition to the need of using load balancers or reverse proxies.

Increasing the number of databases

The amount of data stored can be enormous for such scale so a wise decision is increasing the number of databases. With MongoDB scaling is relatively easy to do. A collection of databases can be created and connected to different servers allowing easier horizontal scaling of data since it takes care automatically of load balancing and redistributing documents [Ref:10].

4. PROJECT COMPONENTS AND IMPLEMENTATION

4.1 Introduction

In this section the thesis will walk through the technical aspects of the system. Discussing the complete network structure, discussing each node in the network in detail and discussing technologies and frameworks used throughout the implementation.

4.2 Network Implementation

4.2.1 Network Structure

The network is structured in a pipeline which has two sides the transmitter side and the receiver side. The transmitter side is usually not connected to the internet. On the other hand, the receiver side is connected to the internet in order to report data and record the data in the database for future reference. The pipeline goes as following, first a sensor reads the device numbers then the sensor readings are then processed by the LoRa transmitter chip. Once the chip processes the data, data is sent to the LoRa receiver over a long distance. As soon as the receiver receives the data it is forwarded to the network edge server which has an internet access. The server uses the database to store the data being forwarded in order for the server later to use the database as a reference for normal behavior numbers. Data is reported simultaneously to Thingsboard to be displayed while it gets stored in the database.

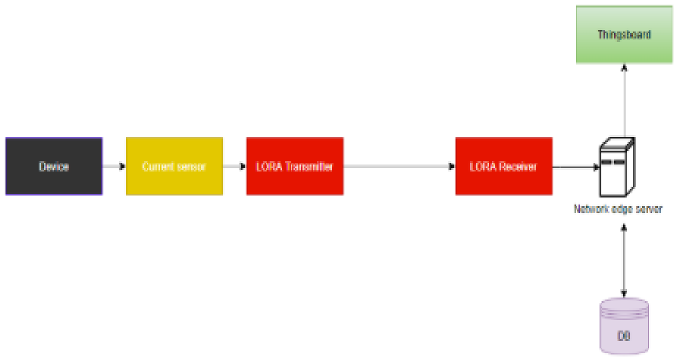


Fig. 4.1. Network structure

4.2.2 Network Scalability

The network is scalable from the transmitter part as many devices can be placed and a single receiver can receive packets from all the transmitters. However, scaling the network this way can come with the possibility of packet loss or packet drop. Increasing the number of end devices increases packet loss [Ref:4] due to the fact of it being based on ALOHA protocol.

If the network scaled from the receivers part so that there is more receivers than there is serial ports in the server, parallel ports or RESTApi can be used, the server is coded to handle both serial port transfer and RESTApi transfer. However, the embedded code in the chips must be changed if this change is needed to be made instead of writing the values to the serial port of the server, an HTTP request is needed to be made by the chip.

4.3 Current sensor and device

4.3.1 Why measure current?

Most electrical power issues come from excessive supply of current to devices due to either a faulty device or electric panels, excessive current can severely damage the device being exposed to the current and cause increased electrical power consumption.

4.3.2 Setup and wiring

In order to measure electrical current the device wire supplying electrical power must be connected in series with the current sensor the ACS712 comes in many variants, the variant in this thesis is the 5A variant which can only read up to 5A.

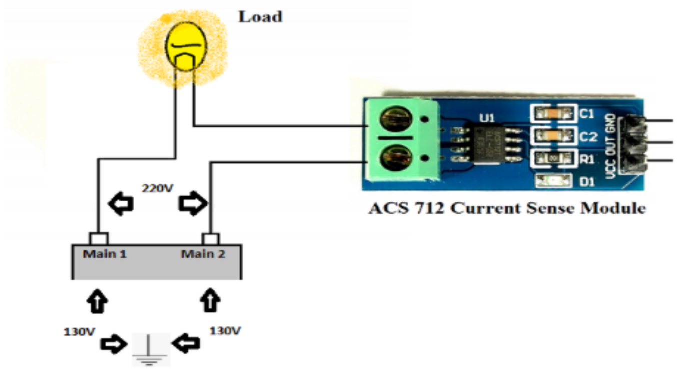


Fig. 4.2. Current sensor load wiring

4.4 Current sensor and LoRa Transmitter

4.4.1 Wiring

First, current sensor and the LoRa chip are connected to a common ground then the OUT pin in the current sensor is directly connected to the SCL analog input pin in the LoRa chip, a 5V DC supply is also needed for the VCC of the current sensor in order for it to operate.

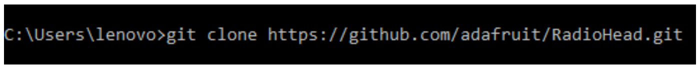
4.4.2 How does the LoRa transmitter chip handle ACS712 readings?

An Arduino code was uploaded on the LoRa chip to handle all of its operations in this part the LoRa chip receives the readings from the current sensor through the SCL pin which is retrieved using the analogRead(SCL) predefined function.

4.5 LoRa chips communication process

4.5.1 Radiohead library setup

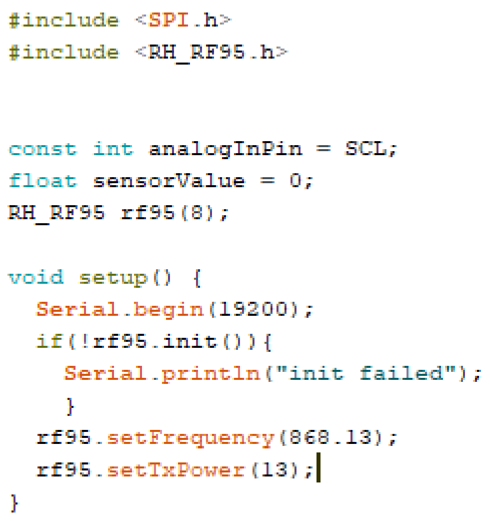
After importing the Radiohead library into the embedded code an instance of the library must be created in the code in order to be used in all the upcoming transmission operations, first the the transmission is initialized then transmission frequency is set in both the transmitter and the receiver, and then at the transmitter the transmission power is set.
First radiohead library is cloned. Then after it is cloned it is imported in arduino as a library.



```
C:\Users\lenovo>git clone https://github.com/adafruit/RadioHead.git
```

Fig. 4.3. Cloning radiohead library

Then the transmitter is setup importing the library and setting the frequency and transmission power.

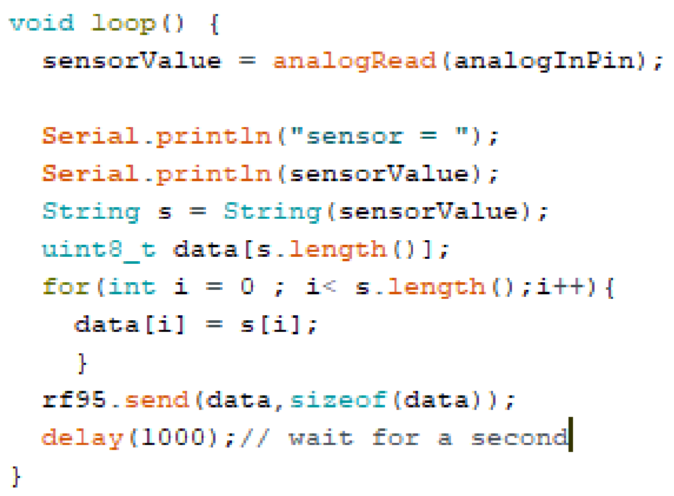


```
#include <SPI.h>
#include <RH_RF95.h>


const int analogInPin = SCL;
float sensorValue = 0;
RH_RF95 rf95(8);

void setup() {
  Serial.begin(19200);
  if(!rf95.init()){
    Serial.println("init failed");
    }
  rf95.setFrequency(868.13);
  rf95.setTxPower(13);
}
```

Fig. 4.4. Transmitter setup snippet

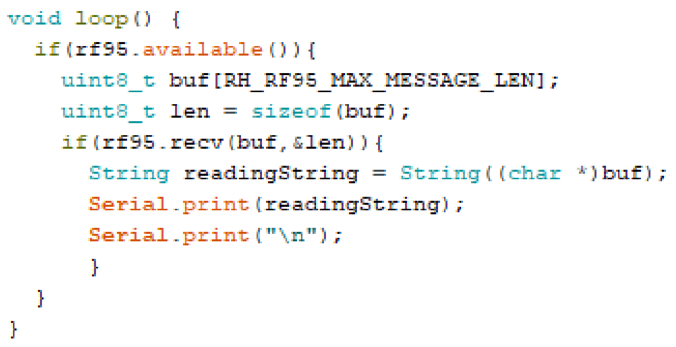
4.5.2 Sending and receiving

Communication is done by the transfer of ascii encoded bits from the transmitter to the receiver, after the sensor value is retrieved the transmitter encodes the bits one by one to ascii and then pushes the encoded bits to an array of ascii bits. The transmitter then sends the bits to the receiver, before each sending phase the transmitter must make sure the previous bits are already sent. On the other hand, the receiver uses a function to receive the bits from the transmitter, this function returns a Boolean specifying whether packets are sent successfully or not, if packets are transmitted successfully the system pipeline continues, if packets are lost in the way no re-transmission request is sent because losing a packet is not critical in the system being discussed.



```
void loop() {
  sensorValue = analogRead(analogInPin);

  Serial.println("sensor = ");
  Serial.println(sensorValue);
  String s = String(sensorValue);
  uint8_t data[s.length()];
  for(int i = 0 ; i< s.length();i++){
    data[i] = s[i];
    }
  rf95.send(data,sizeof(data));
  delay(1000);// wait for a second
}
```

Fig. 4.5. Transmitter sending



```
void loop() {
  if(rf95.available()){
    uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    if(rf95.recv(buf,&len)){
      String readingString = String((char *)buf);
      Serial.print(readingString);
      Serial.print("\n");
      }
  }
}
```

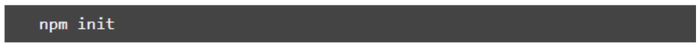
Fig. 4.6. Receiver receiving

4.6 LoRa Receiver to Network edge server

4.6.1 Setting up the server

To setup the NodeJS server being discussed a set of commands must be written after installing the npm.

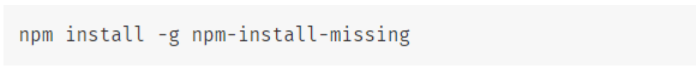
This command is used to initialize the server creating the package.json file which holds all the libraries being used.



```
npm init
```

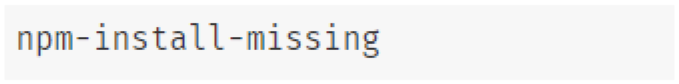
Fig. 4.7. Receiving sending

Then libraries are required then after requiring all libraries as an easier alternative to installing each library on its own.



```
npm install -g npm-install-missing
```

Fig. 4.8. Installing the package which installs missing packages



```
npm-install-missing
```

Fig. 4.9. Installing the missing packages

4.6.2 Serial port

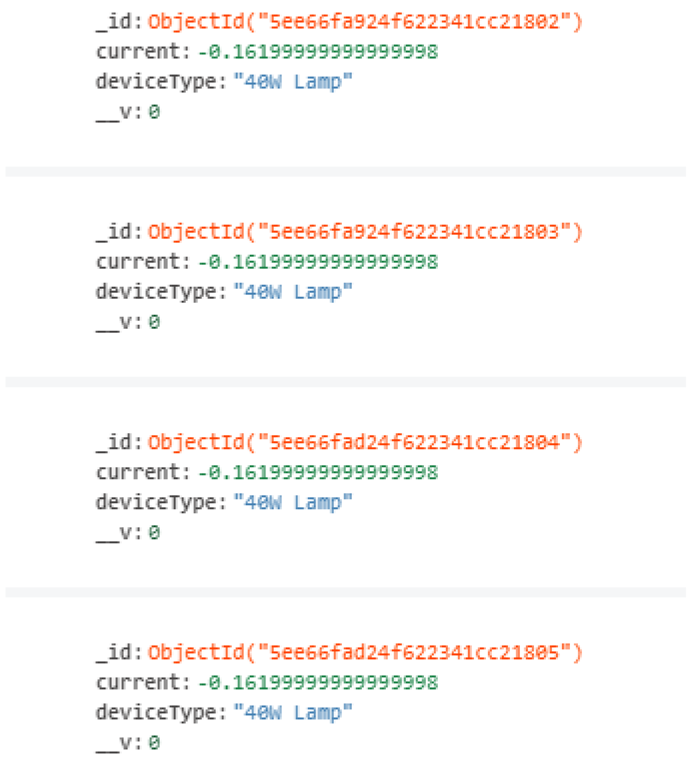
Baud rate is the rate of sending data in serial channels. The baud rate at both ends of the serial port are recommended to be the same otherwise some data might be lost during serial communication. Serial.write() or Serial.print() is used to write from the chip into the specified serial port. On the other end the server using NodeJS receives data by importing the serial port library then initializing a serial port then waiting for the data with the serial port predefined methods.

4.6.3 Server data processing

After data is received by the server it is received in either of two serial ports:

Trusted data port

Data received through the serial port declared as a “Trusted data port” gets stored directly into the MongoDB database connected to the server along with the device type to be distinguished later.



```
_id: ObjectId("5ee66fa924f622341cc21802")
current: -0.16199999999999998
deviceType: "40W Lamp"
__v: 0

_id: ObjectId("5ee66fa924f622341cc21803")
current: -0.16199999999999998
deviceType: "40W Lamp"
__v: 0

_id: ObjectId("5ee66fad24f622341cc21804")
current: -0.16199999999999998
deviceType: "40W Lamp"
__v: 0

_id: ObjectId("5ee66fad24f622341cc21805")
current: -0.16199999999999998
deviceType: "40W Lamp"
__v: 0
```

Fig. 4.10. MongoDB trusted data snippet

Default port

Data received through the serial port declared as a “Default port” waits until it is validated as being a healthy measurement of the device electrical current. That is done by comparing it to the maximum and minimum values of the current of the corresponding device which is stored in the database, if it exceeds any of them it is declared as an unhealthy reading if it is inside the standard deviation then it is declared as a healthy reading.

Healthy readings are sent directly to Thingsboard to be displayed on its corresponding charts while unhealthy readings are also sent to Thingsboard. However, a notification is also sent as an email to alert the system user of the device reporting unusual readings using the Nodemailer library in NodeJS.

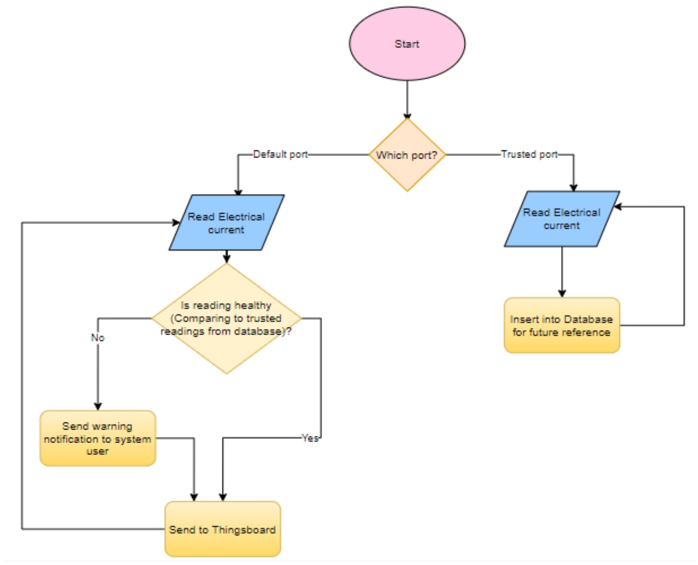

Start
Which port?
Default port
Trusted port
Read Electrical current
Read Electrical current
Is reading healthy (Comparing to trusted readings from database)?
No
Yes
Send warning notification to system user
Send to Thingsboard
Insert into Database for future reference

Fig. 4.11. Current sensor load wiring

4.6.4 Thingsboard

After all the processing and analyzing of data is completed, data is sent to Thingsboard to be displayed.

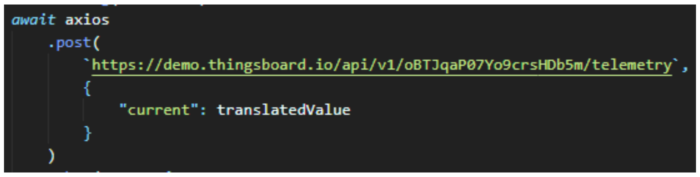
Setting up Thingsboard

Thingsboard have a completely customizable dashboards, collections of devices can be created and integrated easily. After each device is created it is given an access token so that the device dashboard can be targeted to display its corresponding readings.

4.6.5 Sending data from the edge server to Thingsboard

In NodeJS using the library ”axios” which sends HTTP requests, a POST request is sent to Thingsboard URL with the device access token in addition to a body which contains the desired readings to be recorded. The HTTP request consists of a JSON which contains key-value pairs , in Thingsboard the key is the label of the reading being recorded and the value is the value of the reading being recorded.

In figure(3.13) a POST request is sent to Thingsboard containing the value of the electrical current currently being recorded.



```
await axios
    .post(
        `https://demo.thingsboard.io/api/v1/oBTJqaP07Yo9crsHDb5m/telemetry`,
        {
            "current": translatedValue
        }
    )
```

Fig. 4.12. Axios library HTTP request

In figure(3.14) a recording of a resistive load is recorded to Thingsboard displaying the following graph

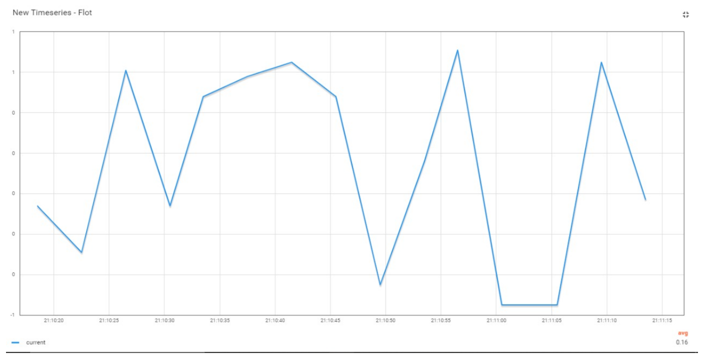


Fig. 4.13. Thingsboard resistive load recording

5. SUMMARY

A highly scalable LoRaWan system was implemented in this thesis which is based on LoRaWan network transmitting data across an area geographically isolated and not connected to the internet to another far away area. The system measures electrical currents of devices and then transmits the values across the network to be analyzed by the server and report useful information for the user. Several frameworks, code bases and hardware implementation were done in this thesis creating a complete network starting from the wiring of the devices to the electrical current sensor and then to the end device, then implementing the embedded chip code on the LoRa devices in order to send and receive reliably then implementing the server code connecting it to the database and allowing it to receive requests both as an HTTP request and through serial ports. Finally reporting the data to be displayed in Thingsboard.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Anupriya, K. Yomas, J. 2015, “A review on IoT protocols for long distance and low power”.

[2] D. Bankov, E. Khorov and A. Lyakhov, ”On the Limits of LoRaWAN Channel Access,” 2016 International Conference on Engineering and Telecommunication (EnT), Moscow, 2016, pp. 10-14, doi: 10.1109/EnT.2016.011.

[3] A. J. Wixted, P. Kinnaird, H. Larijani, A. Tait, A. Ahmadinia and N. Strachan, ”Evaluation of LoRa and LoRaWAN for wireless sensor networks,” 2016 IEEE SENSORS, Orlando, FL, 2016, pp. 1-3, doi: 10.1109/ICSENS.2016.7808712.

[4] F. Van den Abeele, J. Haxhibeqiri, I. Moerman and J. Hoebeke, ”Scalability Analysis of Large-Scale LoRaWAN Networks in ns-3,” in IEEE Internet of Things Journal, vol. 4, no. 6, pp. 2186-2198, Dec. 2017, doi: 10.1109/JIOT.2017.2768498.

[5] U. Noreen, A. Bounceur and L. Clavier, ”A study of LoRa low power and wide area network technology,” 2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Fez, 2017, pp. 1-6, doi: 10.1109/ATSIP.2017.8075570.

[6] Elshabrawy, T. Robert, J. 2018, “Analysis of BER and Coverage Performance of LoRa Modulation under Same Spreading Factor Interference,”.

[7] Casals, L. Mir, B. Vidal, R. Gomez, C. 2017, “Modeling the Energy Performance of LoRaWAN,”

[8] A. Lavric and V. Popa, ”A LoRaWAN: Long range wide area networks study,” 2017 International Conference on Electromechanical and Power Systems (SIELMEN), Iasi, 2017, pp. 417-420, doi: 10.1109/SIELMEN.2017.8123360.

[9] H. Evan, Express In Action, Manning publications, 2016

[10] C. Kristina, MongoDB definitive guide, O'reilly Media, 2013