ASHESI

ASHESI UNIVERSITY

DESIGN OF A LOW-COST STREETLIGHT MONITORING SYSTEM USING LORA

CAPSTONE PROJECT

Bsc. Computer Engineering

By

Genesis Tangong Nchopereu

2019

ASHESI UNIVERSITY

DESIGN OF LOW-COST STREETLIGHT MONITORING SYSTEM USING LORA

CAPSTONE PROJECT

Capstone Project submitted to the Department of Engineering, Ashesi University in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Engineering.

By Genesis Tangong Nchopereu

2019

DECLARATION

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

……

Candidate's Name:

……

Date:

……

I hereby declare that the preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University.

Supervisor's Signature:

……

Supervisor's Name:

……

Date:

……

Acknowledgment

To all the people whose encouragement and academic advice helped me undertake this project, here is to acknowledge your contribution to this work.

I am grateful to my supervisor for his sleepless nights in helping me revise the ideas in this project, design and implement it into a capstone project and also for his kind academic and emotional support throughout the year. Mr. Gatsi, without you, I wouldn't have done this project. So, THANK YOU very much.

To my friends and family – thank you for your emotional support and love throughout this year and for standing with me through the tough times and the rough moments. You are the actual brains behind this achievement.

To my classmates and Lecturers – although we all had our various projects to work on, I am grateful for those who always took time to help me solve one or two issues with my project. I am thankful for your selfless sacrifices.

Abstract

Streetlights are public goods that beautify cities, ensure safety for road users and security for city neighborhoods. A delay in streetlight maintenance jeopardizes the safety and security of many. Also, when streetlights stay on throughout the day, it wastes power and energy and increases cost. This project presents the design of a low-cost streetlight monitoring system based on LoRa technology. In the design, a Dragino LoraWAN 868MHz (LoRa/GSM) gateway module was used alongside one LoRa shied (transceiver) and an Arduino Uno. The transceiver, Arduino, sensors, and relay formed a LoRa node which was mounted on a streetlight. There is a web application at a remote-control unit with an interface that monitors every node (streetlight) connected to the system. Data about each node was sent to the gateway which uploaded it to the LoRa App server. The LoRa App server decoded the data and made it available through the RESTful API. The data on the LoRa server was accessed via the API using an HTTP integration and was displayed on the web application at the remote monitoring centre. The test LoRa node was able to send accurate information about the state of the streetlight to the LoRa server, and it was accurately displayed at the monitoring unit web interface or dashboard. The node was also able to switch on/off or dim the streetlight at the right times.

Key Words: Streetlight monitoring, LoRa, MQTT, remote-control, gateway, IoT

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Background

Streetlights are one of the most critical infrastructures of a city. They not only make the city beautiful but ensures safety and security as they prevent road accidents at night and keep people safe in dangerous neighborhoods. One cannot imagine a city with streets without lighting. Therefore, we expect at all times to ensure that streets are well lighted and decorated using streetlights.

In the olden days, streetlights used to be turned off and on manually- using a switch. Today, most of our streetlights are turned on automatically by controllers at various stations depending on the time of day or night. When the controller senses a threshold change in light intensity, it switches the streetlights on (intensity below threshold) or off (intensity above threshold).

A major problem with the current street lighting system is that the controllers we have at major transformer stations, that are responsible for switching streetlights on and off are mostly faulty [1]. This is because the controllers depend on time and their timer may malfunction causing them to be out of synch with the time of the day. Hence the reason why we may find some streetlights still on during the day or some streetlights still off when it is already dark. Another issue with using timers is that day times differ during some periods of the year – sometimes it gets very dark by 6 pm and other times it only gets dark by 7 pm and so on. Also, current controllers are not closely monitored to know when they develop faults or not. In other words, they are not smart.

When a streetlight is not on, the controllers would not know and neither would the company monitoring the system be able to identify the fault on the controller or the streetlights without sending a team of technicians to check. Some of these maintenance work

takes days and even weeks to be given attention. This delay is definitely at the expense of the safety and security of road users. A delay in fixing a streetlight in a dangerous neighborhood could also increase crimes. On the other hand, if a streetlight stays on throughout (day and night), a delay in the resolution of this anomaly results in the waste of energy and power.

1.2 Problem Definition

Roadside crimes and insecurity are significant threats in many neighborhoods in our cities, and a city should not afford to have street lights off for just a night in dangerous neighborhoods. Also, driving on the street without street lights at night is not the safest thing. For instance, in Ghana, the lack of streetlights is one of the major causes of many roadside robberies and accidents at night [2]. On the other hand, we also cannot afford to have streetlights on during the day as that brings unnecessary power consumption and cost. Therefore, cities need a street lighting system that is intelligent, cheaper, low power and has a low cost of maintenance.

1.3 Objectives of the Project Work

The main objective of this project work is, therefore to design a low-cost streetlight monitoring system for implementation in a large city such as Accra using LoRa technology. Also, in designing this low-cost system, key components such as remote controlling and fault detection must be achieved. The entire system has to make current streetlights smart and be able to save power when not in use at certain times of the night.

1.4 Expected Outcomes of Project Work

The expected outcome from this project is the implemented design prototype of a streetlight monitoring system that can be used to monitor streetlights in a large city. This project is also expected to have a working model that proves the following about the system.

1. Low cost
2. Intelligence (Smart)
3. Reduced power consumption
4. Improved maintenance – increased maintenance speed due to feedback on the state of streetlights.
5. Reduced road accidents, crime, and insecurity due to non-functioning streetlights

1.5 Justification/Motivation for Project Topic

Making our cities safe and at the same time beautiful and smart is the kind of contribution that a capstone project like this should offer. A community without functioning streetlight has a lot to lose, and we all have to do something about it. Intelligent streetlight monitoring systems have been designed and even deployed in other countries and continents, aside Africa. However, these systems have high costs that we cannot afford. Therefore, there is a need to design a system with a lower cost that suits the situation of the countries in Africa.

1.6 Scope of Work

This project produces a design for a bigger system and monitors it is functioning. However, implementation and testing are done on a smaller scale - using two streetlights (end-nodes) and a remote monitoring unit. This project covers turning street lights on/off remotely, dimming to reduce power consumption and the development of a remote control unit with a web application interface to independently monitor each streetlight. A modified

streetlighting circuitry would also be implemented to enable retrofitting of the streetlight monitoring unit on every streetlight.

Chapter 2: Literature Review

2.1 Streetlight Monitoring – What Has Been Done?

Before diving into what has been done so far in the field of streetlight monitoring, it is critical to revisit what streetlight monitoring means for a big city like Accra. Streetlight monitoring has one single objective of making sure that every streetlight is on or off or dimmed at the right time. As mentioned in Chapter 1, monitoring streetlights is necessary for three reasons:

(a) To ensure that faulty lights are identified and repaired immediately they experience a malfunction

(b) To ensure safety for drivers and security for inhabitants of dangerous neighborhoods

(c) and to reduce power consumption.

To implement robust streetlight monitoring, different researchers and engineers have turned to various technologies and approaches. These approaches either use different techniques to implement full streetlight monitoring systems or are focused on a particular aspect of streetlight monitoring. A good example of the later is the research done by H. Lee and H. Huang that focused primarily on developing a system for detecting faulty streetlights [3]. Others have focused on developing a remote-control system to help the monitoring of streetlight for a big city from one central location to save power [4]. On the other hand, a majority of work done in this domain [1], [4] is focused on building a full streetlight monitoring system that incorporates remote controlling system, fault detection system, and for some, a power-saving technique, e.g., a dimming technology [5]. These full system designs are similar to that which this capstone seeks to produce.

The backbone of any smart streetlight monitoring technique is communication technology. For streetlights to be controlled, there must be a way to communicate with them or for them to communicate with each other. Now, let us take a look at some of the

communication media or technologies that engineers have implemented so far. Most of the systems developed so far have used GSM/GPRS [1], [4], in combination with wireless sensor networks and different microcontrollers [6], [7].

Despite the successes in implementing streetlight monitoring systems using the various technologies described above, there are still limitations. Most of the deficiencies seen in [1], [4], [6] are the fact that the cost of the system is high and therefore, future work needs to help in reducing the cost of those monitoring systems. The second limitation to most of the previous works is power consumption. WIFI, for example, consumes a lot of power, and therefore, future works on streetlight monitoring need to reduce the power consumption of the system.

2.2 About LoRa Technology

Described by its producer, Semtech, as the DNA of IoT [8], LoRa is a low-power device used for long range communication using radio frequency transmission. LoRa is the short form for Long Range. LoRa technologies have many different features that enable IoT projects to achieve excellent results in solving challenging problems. Some of the features of LoRa that are particularly interesting to this project are; long range, low power, and low cost.

2.2.1 Long Range

LoRa technologies can connect devices up to 30 miles apart in rural areas [8]. Indoor and mall IoT options can go about 5km as well making it possible to monitor streetlights a few kilometers apart by using just one gateway. This feature is what makes LoRa superior to Bluetooth, Wi-Fi, GSM, and other wireless technologies in a project like streetlight monitoring.

2.2.2 Low Power

LoRa is a low power device with an extended battery life of about ten years. This is extremely important given the energy crisis that our world currently faces. The small energy consumption aspect of LoRa technology makes it the best fit to build and monitor systems that are isolated and may not have access to a power supply which is key to IoT. Again, this sets LoRa ahead of Wi-Fi, GSM, and other wireless technologies.

2.2.3 Low Cost

The cost of installing Wi-Fi infrastructures or reloading GSM module data are incredibly high and can halt an IoT project in the long run. With long battery life and its use of radio frequency for data transmission, LoRa technology reduces the cost of maintenance and power supply. Cost is essential to this project and has to be kept as low as possible. This makes LoRa the best option for a project like this.

Besides, there are other features of LoRa technology that makes it attractive to IoT projects. Some of these features include high capacity, geolocation, and security. These features are useful to projects on a case by case basis [8].

2.3 How LoRa Works

Figure 2.1 shows a generic LoRa network topology with all sections and applicable protocols shown. LoRa uses a LoRaWAN protocol which is a wide area networking protocol which connects devices wirelessly to the internet. This protocol is standardized by the LoRa Alliance, a non-profit organization driving the technology and its use around the world [8].

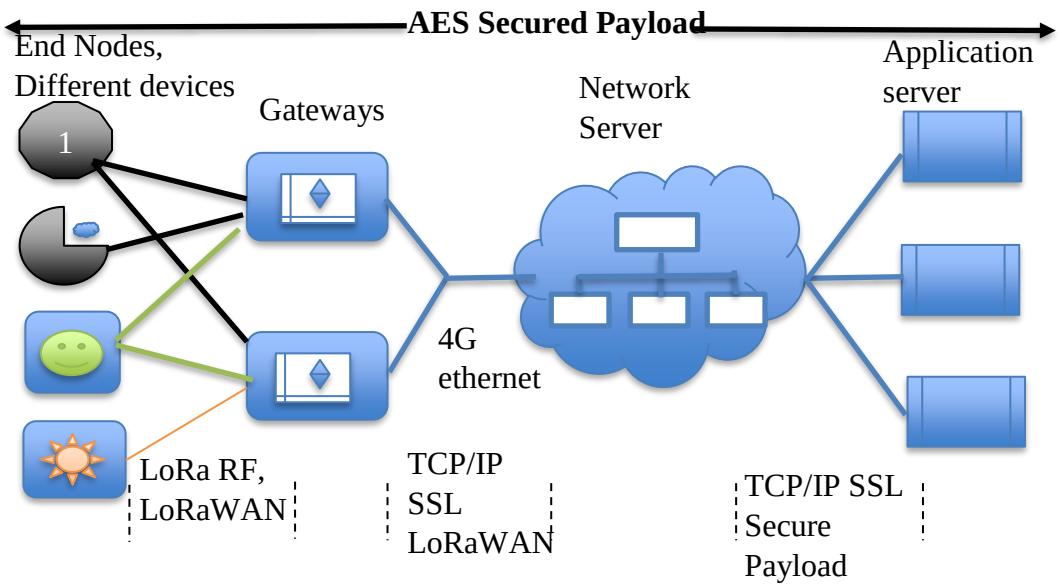

Figure 2. 1: LoRa Network topology

A LoRa end-node or end-device with sensors send data to specified LoRa gateway several kilometers away using radio frequency and the LoRaWAN protocol. The gateway which must have an internet connection via Wi-Fi/cellular or ethernet sends the sensors' data to the LoRa Network server via TCP with the help of OpenSSL. The network server then does the decoding of the data and other network management functions to make the data available to the application server.

The last part of the system is the application server which makes information available to its owner after interpreting. This is where the user or owner of the data can decide what to do with the information – whether harvesting it to use for other purposes on other platforms or just visualizing it on the cloud server.

Note should be taken that even though the above explanation is straight forward, the work is done and codes are written to send data, from end-nodes to gateways and from gateways to the LoRa network server are a bit challenging. One has to install the LoRa server, the gateway bridge and the LoRa app server on Ubuntu, then program them. After that, the gateway must be configured to send the data through the installed server making

use of both IP and MAC addressing. To implement all the steps, strong knowledge of programming in JavaScript and C is needed, and one has to be very comfortable with LINUX command line programming - see [9]. Details of this is found in Chapter 5.

Chapter 3: Design

Looking at the current solutions to streetlight monitoring, it is worth expanding further to reveal the advantages and disadvantages of their technical specifications. This section will dissect the approach using GSM/GPRS and a wireless sensor network to show the cost and power consumption constraints and the need for the design that this project proposes.

3.1 Review of Existing Designs

As seen from the literature review in Chapter 2, the various streetlight monitoring approaches all have aspects that need to be improved upon - mostly in terms of power consumption especially with using GSM or WIFI, and in terms of reducing the high cost of the communication technology used. With the growing popularity of LoRa technology, this capstone seeks to use LoRa to implement a streetlight monitoring approach that solves the problems in the systems reviewed above. LoRa is low-cost and power-saving at the same time and can do a more robust job compared to other technologies as will be discussed in the next chapter. So far, most of the streetlight monitoring systems and approaches have been applied to countries like China, Japan, Germany, etc. but not in Africa as of yet [1], [3], [4], [5], [6]. This project may be one of the few that will try to apply the LoRa streetlight monitoring approach to a local scenario in Africa.

3.2 Design Objective

This project seeks to design and implement an intelligent, low-cost power-saving streetlight monitoring system that could be used for a big city to control street lights remotely.

3.3 Design Decisions

The first major design decision for this project is choosing which communication technology to use as seen in Table 3.1. The communication technology needs to meet our design requirements for low cost, power-saving, long range, reliability and ease of maintenance (how frequently we maintain the system). Following these requirements as shown by the Pugh Matrix, LoRa is the best option for this project. Wi-Fi does not meet the systems power consumption and cost requirements, and neither does GSM/GPRS. Also, Wi-Fi maintenance would be higher than most of the options because of its use of additional infrastructures like routers, cables, etc and many units to cover long distances [10]. Bluetooth is the next best option for LoRa. However, Bluetooth does not meet the long-range requirement for this system. Even though Bluetooth could still be used with point-to-point communication, the complexity of using such an approach becomes too much especially when a faulty point (streetlight) is encountered.

Therefore, LoRa dominates all other technologies for this project because it has long range, low comparative cost, and low power consumption. Most importantly, LoRa fits this project because even though it has a low data rate, the data involved here is minimal and can be handled by LoRa. The data also does not need a lot of processing; neither does it need a small real-time interval when sending.

3.3.1 Pugh Matrix

Table 3. 1: Pugh Matrix

			Options			
			Wi-Fi	**Bluetooth**	**GSM/GPRS**	**LoRa**
Criteria	**Concept**	**Baseline**				
1	System cost	0	-1	0	-1	0
2	Reliability	0	+1	+1	+1	+1
3	Power-saving	0	-1	+1	-1	+1
4	Range (Point to point communication)	0	+1	-1	+1	+1
5	Maintenance (ease and time interval)	0	-1	+1	0	0
Total		**0**	**-1**	**2**	**0**	**3**

3.4 Design Iterations

Figure 3.1 shows an architecture of the low-cost, power-saving system design with various components and how they connect.

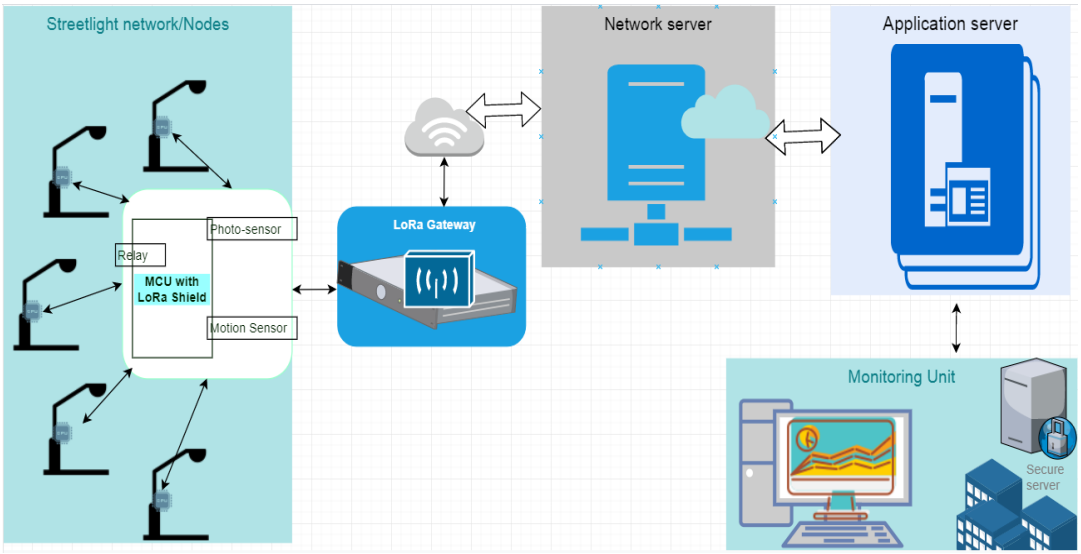

Figure 3. 1: LoRa streetlight monitoring design architecture

3.4.1 System Block Diagram

Figure 3.2 shows a more precise block diagram of the entire system design with more details of each block. The motion sensor is used for late night controls. When many cars are not using the road, the streetlights are dimmed. Once the sensor senses motion towards a streetlight, it tells the system to brighten the streetlight and vice versa. The choice of using an ultrasonic sensor is because it has a shorter response time, is simple, cheap and easy to use. It is noted that although a motion sensor with a range of least 100m is practical in this project, the one used for the prototype can only measure up to 10m. On the other hand, the photosensor is used to sense light intensity to determine when to switch the streetlight off or on. When it reads an intensity below a daytime threshold, it relays the information for the streetlight to be turned on and vice versa. An LDR is used here because it is readily available in our kits and the response time for this project should not be a problem since it does not just get dark within seconds.

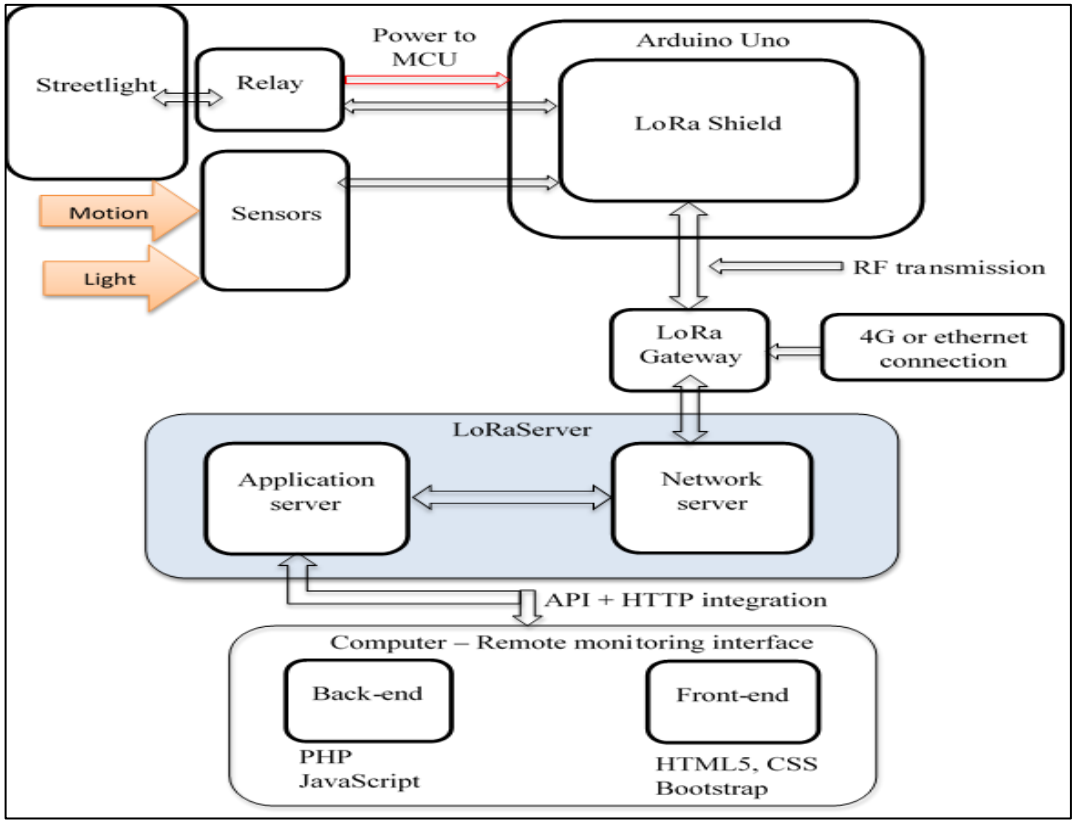

Figure 3. 2: Block diagram of system design

The second section of the block diagram that is important to discuss is the use of an Arduino Uno. An Arduino Uno is used as the microcontroller unit (MCU) because it is readily available, easier to program and it is light-weight for this project. The LoRa shield is designed to sit comfortably on an Arduino Uno. The MCU does the sensor management and processes data from the sensor to make decisions, but it is the shield that uses the LoRaWAN protocol to send the data via RF transmission to the gateway.

The last part of the design is the remote-control web interface where streetlight management see in real time what is happening with every streetlight in each streetlight. The web application needs to be interactive with good semantics. That is why JavaScript and CSS are incorporated with HTML5 to build the interface. Bootstrap also gives additional help in designing the web application. The back-end server side uses PHP to help with data displace when data from the LoRaServer is gotten through HTTP integration API.

3.4.2 Node Circuitry (designed for PCB)

Figure 3.3 shows the circuit design for each node to be printed into a printed circuit board (PCB). The design was done in EAGLE software.

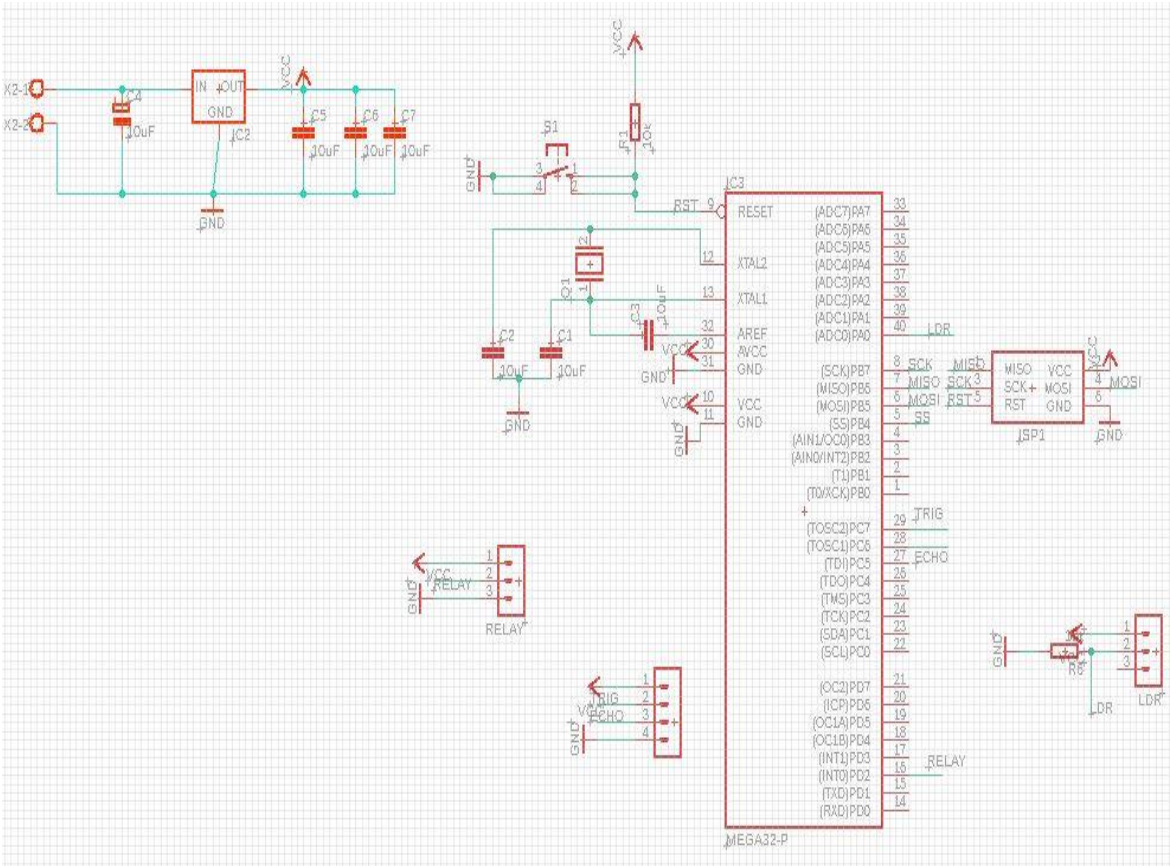

Figure 3. 3: End-node circuit design for PCB

Chapter 4: Implementation

4.1 Experimental Setup

The prototype for the low-cost streetlight monitoring system consists of a LoRa end-node, a LoRa gateway from Dragino, and a web application built using HTML5, CSS, PHP and JavaScript with the help of Bootstrap.

4.1.1 LoRa Setup

The LoRa end-node was implemented using an Arduino Uno, a LoRa shield with 868MHz compatibility on which an ultrasonic motion sensor, an LDR, and a 2-way relay are connected. The shield was mounted on the Arduino and the sensors and relay connected directly onto the shield and programmed to send data to the LoRa gateway through RF transmission.

The LoRa server, the LoRa App server, and the LoRa gateway bridge must be installed on an Ubuntu server before using them. Therefore, a VMware was installed, and a virtual machine for Ubuntu 16.04 server installed and configured for use. LoRa gateway bridge was installed on the Ubuntu server to enable access and use of the LoRa gateway. The loraserver and the lora-app-server were also built with the required PostgreSQL databases and MQTT to allow access to the LoRa App server to which the gateway would later send data. One user for the lora-app-server and another for the loraserver were created and given passwords.

The corresponding `.toml` script files for each of them were edited to specify the users and their credentials. For the lora-app-server, this must be done to use the right configuration variable for PostgreSQL.dsn. The following command was used to open the file; `cd /etc/lora-app-server/` then `nano lora-app-server.toml`. Once the file

opened, the configuration variable for PostgreSQL.dsn was changed to include the password and domain name for this project's Lora App server as seen below and the file saved.

```
dsn="Postgres://loraserver_as:yourpassword@localhost/loraserver_as
?sslmode=disable"
```

Also, for the loraserver `toml` configuration, the same commands, as shown above, were used but this time replacing lora-app-server with lorasrver. However, once the file is opened, we rather edit only the line for JWT secret. In the file, the JWT secret token has to be generated and specified. The JWT secret token is used for API authentication, without which the data from the LoRaServer cannot be accessed through the APIs provided. This JWT token was generated by running the `OpenSSL rand -base64 32` commands and assigned as shown below. Then the file was saved.

```
JWT secret = "OghkHLgvrdW5LcLH2mSrTBoHadgmMnTOGPvvZnX+/Ws="
```

The lora-gateway-bridge, the loraserver, and the lora-app-server were run (using the commands `sudo systemctl start lora-gateway-bridge`, `sudo systemctl start loraserver`, `sudo systemctl start lora-app-server` respectively) and checked to make sure they were running without errors before proceeding to the next step. Errors were rectified by checking their corresponding `.toml` scripts to ensure accurate configurations.

The gateway was programmed to use the 4G cellular network to transmit data to the LoRa network server. The 4G network is configured on a PC with a manually assigned static IPv4 address that would ensure that the gateway, the LoRaServer installed on Ubuntu 16.04 with the LoRa App server run on the same network. When the gateway and the LoRa server are running on the computer, they are assigned unique host IPs through a DHCP (dynamic host configuration protocol) setting. To access the gateway settings, its host IP address or

the default Dragino IP 192.168.42.195 is used. Once accessed, the gateway is set to send data to a particular LoRa App server by giving it the server name, and its IP address.

The host IP assigned to the LoRa server installed on the virtual machine (gotten from running `ifconfig` in the virtual machine terminal) is used to access the server online. Usually, we type the IP, say `192.168.1.130:8080` to access the LoRaServer. The `:8080` is a convention for LoRa. Once the App server is loaded, we can log in and set up the environment.

4.1.2 LoRa App Server

LoRa App Server is an open-source LoRaWAN application-server, part of the LoRaServer project. It is responsible for the device "inventory" as part of a LoRaWAN infrastructure, handling of join-request, handling, and encryption of application payloads. It offers a web-interface where users, organizations, applications, and devices can be managed [9]. To set up this server;

- Information about the gateway and the end-node and sensor devices are needed
- A gateway profile must be created for the gateway to show on the App server
- A service profile must be set up to use the App server as a service
- After that a network server is setup
- Then an Application is created and assigned the service profile and the network server
- Inside the application, devices are created, and in this case, a streetlight monitoring (SLM) end-node called Node1
- Then, the credentials of the device and its gateway are set up as seen in Figure 4.1 and Figure 4.2

Figure 4. 1: LoRa App server showing an application-SLM and a device-Node1 created

For the gateway, the MAC address of the gateway and its host IP address were used. For an end-node, it is MAC address (a non-unique 32-bits DevAddr) must be issued to the server alongside two very essential addresses; the network session key (NWKSKEY), and the application session key (APPSKEY). Both the NWKSKEY and the APPSKEY are 128 bits. The NWKSKEY is used for interaction between the device and the network server to check the validity of messages. It is also used to change the non-unique DevAddr of the device to a unique 64-bit DevEUI (end-device identifier). The APPSKEY is used for encrypting and decrypting particular payloads coming from the device. This information was equally included in the Arduino code for the end-node as shown in Figure 4.2.

```
// LoRaWAN NwkSKey, network session key
static const PROGMEM u1_t NWKSKEY[16] = { 0x00, 0x03, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f };

// LoRaWAN AppSKey, application session key
static const u1_t PROGMEM APPSKEY[16] = { 0x00, 0x03, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f };

// LoRaWAN end-device address (DevAddr)
static const u4_t DEVADDR = { 0x00010205 };
```

Figure 4. 2: Arduino code sample showing LoRaWAN network end-device keys setup

Once the device setup was completed, the PuTTY software was used to login to the Dragino gateway, and the package forwarder was run using the command `/etc/iot/scripts/lg01.pkt.fwd` which initiates packet forwarding to the LoRa server and can be viewed through the LIVE DEVICE DATA log on LoRaServer as uplink payloads.

4.1.3 Encoding and Decoding Payload

Encoding and decoding of payload from each device are handled by the LoRaServer. However, a JavaScript function must be written to specify how the data should be encoded and decoded and where to get the data from. The JavaScript code snippet that was implemented to handle this section of the setup is shown in Appendix A.

4.2 End-node Setup and Programming

The end-node is the portion of the project design that is retrofitted to a streetlight. It is made up of an ultrasonic motion sensor, an LDR, a relay which is all mounted on a LoRa shield which is in turn mounted on an Arduino Uno. The sensor reading from the LDR informs what actions the MCU must take at all times. Therefore, the LDR was set up on the LoRa shield mounted on an Arduino Uno and trained to identify night times and day times based on light intensity reading. Readings for when it is about to get dark, when it is dark, when the streetlight is on when the streetlight is dimmed and when it is dawn were taken and a threshold established.

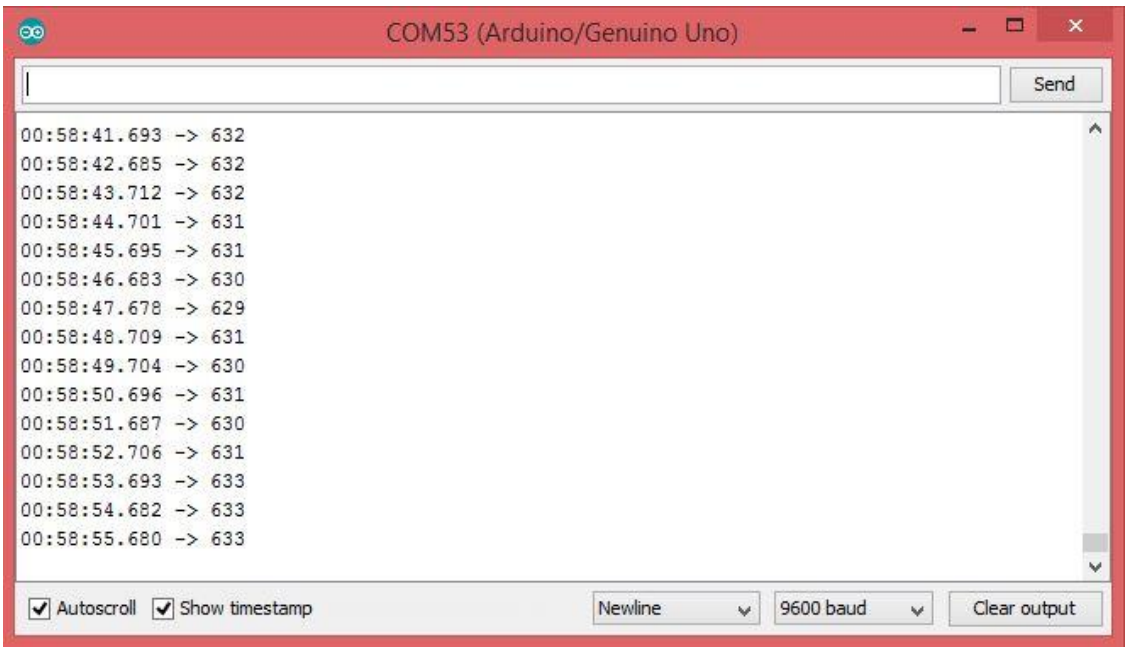

Figure 4. 3: Sample light intensity reading using an LDR when a streetlight is on at midnight

Table 4.1 shows the different light intensity threshold used to switch on/off or dim the streetlights. The action taken also depends on whether motion is sensed around the streetlight or not.

Table 4. 1: Sensors' threshold values for ON, OFF and DIM actions

Action	LDR reading threshold	Ultrasonic sensor reading
Turn ON	< 70	Ignored
Turn OFF	1000	Ignored
DIM	600-700	5m - No motion
Brighten	< 600	< 5m – Motion detected

The motion sensor is used to sense motion at late night to determine what action to perform on the streetlight (to dim or brighten). Although the maximum range for the sensor is 10m, in practice, the sensor could only measure motion up to 5m from the streetlight. The relay is what holds the two wires that would connect to the streetlight to facilitate dimming and switch OFF/ON.

All programming at the end-node were done in the Arduino environment. All sensors were initialised and configured to the various ports. All codes including LoRa MQTT integration are found in the attached project documents folder.

The streetlight (end-node) was tested using a series of 4 LEDs divided into two groups of 2. Dimming was carried out by putting off one group of 2 LEDs and leaving 2 ON and turning the streetlight ON meant switching on all four LEDs, similar to brightening.

Information about the state of the streetlight (whether it is ON/OFF or Dimmed) was sent via the LoRa RF transmission to the LoRa gateway. The state of the streetlight is gotten

from the last action taken by the MCU as shown in Table 4.1. The data ends up on the LoRaServer, and it is retrieved and displayed on the monitoring web application interface. For each state, 35 uplink data were collected and checked to see whether they matched the actual state. The comparison and accuracy as shown in Table 5.1 were what informed the threshold decisions for Table 4.1.

4.3 Web Application Interface

Remote monitoring is achieved with a web application interface from which the state of each streetlight connected to the LoRa system is observed in real time. The web application was built with the help of HTML5, PHP, JavaScript, MySQL database and bootstrap. A secure login was created such that access to the interface is granted only to authorized persons. Their credentials are manually inputted into the database, and no signUp is allowed for any persons as access to the system is very restricted. The user interface (UI) is simple and responsive showing the streetlight number as pole numbers and their corresponding states at each time of the night. Data from the streetlight gotten through the LoRaServer is stored in the MySQL database and displayed in real time for each pole. The UI is as shown in Figure 4.4. On the interface, a streetlight is marked as “faulty” if it cannot be reached. There, someone needs to go and check the issue with the streetlight. If all streetlights in the street are ON, then it shows a green box after the street name on the second nav bar. Otherwise, it shows a red box.

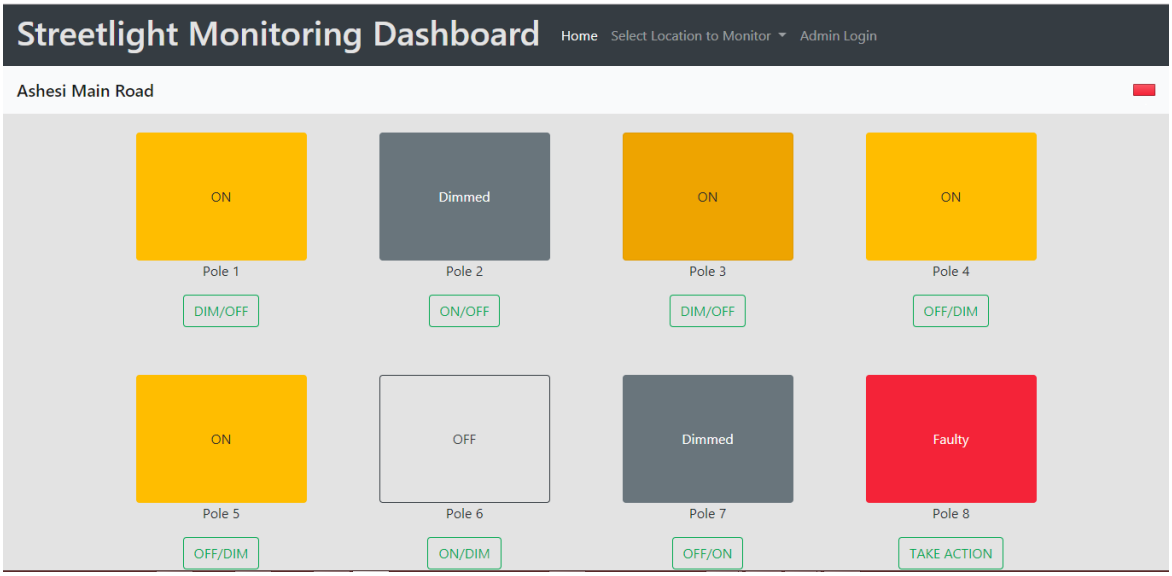

Figure 4. 4: Web application Interface to monitor the states of streetlights remotely

4.4 Getting Data From LoRaServer To Web App

When the data goes to the LoRaServer, it is now time to extract the information, process it and display the state of the end-node which is a streetlight on the web application at the remote monitoring unit. To do this, the RESTful API was used with the help of HTTP integration from the LoRaServer application side. A particular POST method was used for logging in as the user of the installed server. The sign-in exercise gave the access token for the admin user which is used to generate the data access token for the desired device using a different GET method shown in Figure 4.5. The data access token also came with various header links that were included in a PHP code used to send HTTP requests for the data to the LoRaServer. The PHP code is included in the project documents folder. The code consists of lines to extract that data and display the state of the streetlight on the web application interface. Figure 4.5 shows the APIs used.

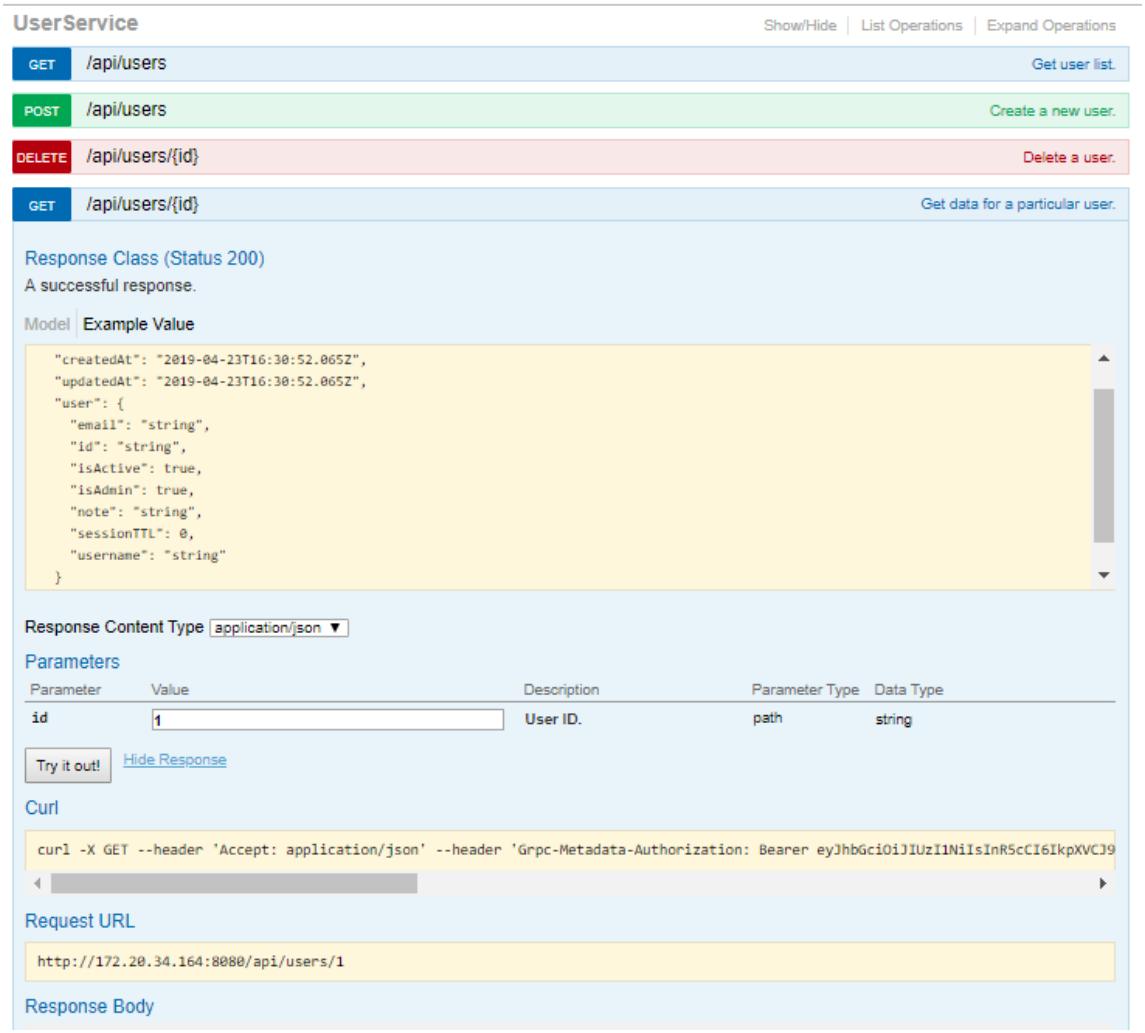

Figure 4. 5: API for getting data from a particular device on a LoRa App server

Chapter 5: Results

5.1 LoRa Setup

The LoRa network and application server were successfully installed, and the LoRa App server configured to receive and display real-time data from an end-node on a streetlight. Figure 5.1 shows a real-time sample data from the motion sensor with decoded payload shown as "Proximity '96'".

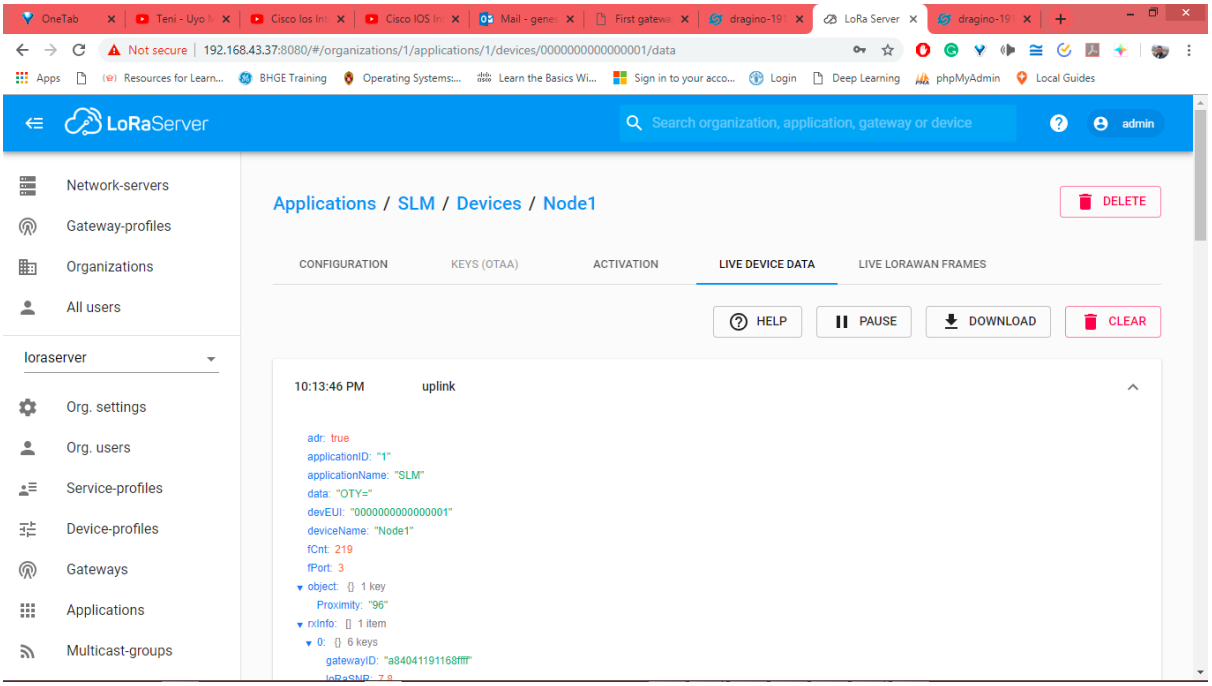

Figure 5. 1: Sample uplink data successfully received and decoded, shown as a downloadable JSON file

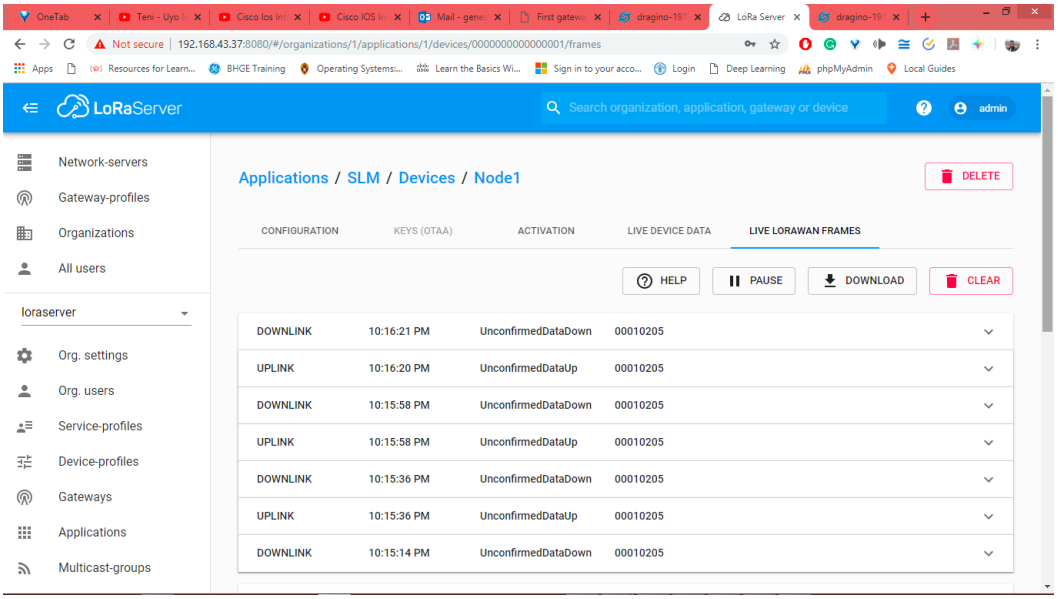

Figure 5. 2: Sample of damaged frames received by LoRaServer

Note should be taken that sometimes packets can get errors as displayed in Figure 5.2. This represents an event published in case of an error related to payload scheduling or handling. E.g., in the case when a payload could not be scheduled as it exceeds the maximum payload size. Figure 5.3 a live log of the gateway sending data to the LoRa server – gotten from PuTTY.

```
INFO (json): [stat update] {"stat":{"time":"2012-01-01 00:26:39 UTC","lati":5.55
220,"long":-0.32520,"alti":0,"rxnb":0,"rxok":0,"rxfw":0,"ackr":0.0,"dwnb":0,"txn
b":0,"pfrm":"LG01/OLG01","mail":"aminou455@yahoo.fr","desc":""}}

INFO (JSON): [up] {"rxpk":[{"tmst":2232336395,"time":"2012-01-01T00:26:40.274342
Z","chan":7,"rfch":0,"freq":868100000,"stat":1,"modu":"LORA","datr":"SF7BW125","
codr":"4/5","lsnr":7.8,"rssi":-82,"size":15,"data":"QAUCAQCAOwADXUfrkqEn"}]}
INFO: [up] PUSH_ACK received in 12 ms

INFO (json): [down] {"txpk":{"imme":false,"tmst":2233336395,"freq":1210.13248,"r
fch":0,"powe":14,"modu":"LORA","datr":"SF7BW125","codr":"4/5","ipol":true,"size"
:24,"data":"YAUCAQCMKwADoAcAAQUA0q2ECAH7VFVH","brd":0,"ant":0}}
INFO: [down]txpk payload in hex(24byte): 60050201008c2b0003a00700010500d2ad84080
1fb545547

INFO (JSON): [up] {"rxpk":[{"tmst":2243152786,"time":"2012-01-01T00:26:51.090759
Z","chan":7,"rfch":0,"freq":868100000,"stat":1,"modu":"LORA","datr":"SF7BW125","
codr":"4/5","lsnr":7.8,"rssi":-82,"size":15,"data":"QAUCAQCAPAADfiaMc1T+"}]}
INFO: [up] PUSH_ACK received in 7 ms

INFO (json): [down] {"txpk":{"imme":false,"tmst":2244152786,"freq":1210.13248,"r
fch":0,"powe":14,"modu":"LORA","datr":"SF7BW125","codr":"4/5","ipol":true,"size"
:24,"data":"YAUCAQCMLAADoAcAAQUA0q2ECAHP2KLV","brd":0,"ant":0}}
INFO: [down]txpk payload in hex(24byte): 60050201008c2c0003a00700010500d2ad84080
1cfd8a2d5
```

Figure 5. 3: Sample end-node packets being sent by LoRa gateway to the server, shown in PuTTY environment

5.2 The Web Application

The MySQL database and PHP code work correctly to provide the expected secure login feature of the web application. Below is the login interface of the web application shown in Figure 5.4 and the UI again in Figure 5.5. The web application was able to display the right information about the state of the streetlight as represented by the data gotten using RESTful API from the LoRa App server. One can see the exact state of every streetlight on the dashboard, and those with faults or unknown states can be given specific attention to resolve their issues.

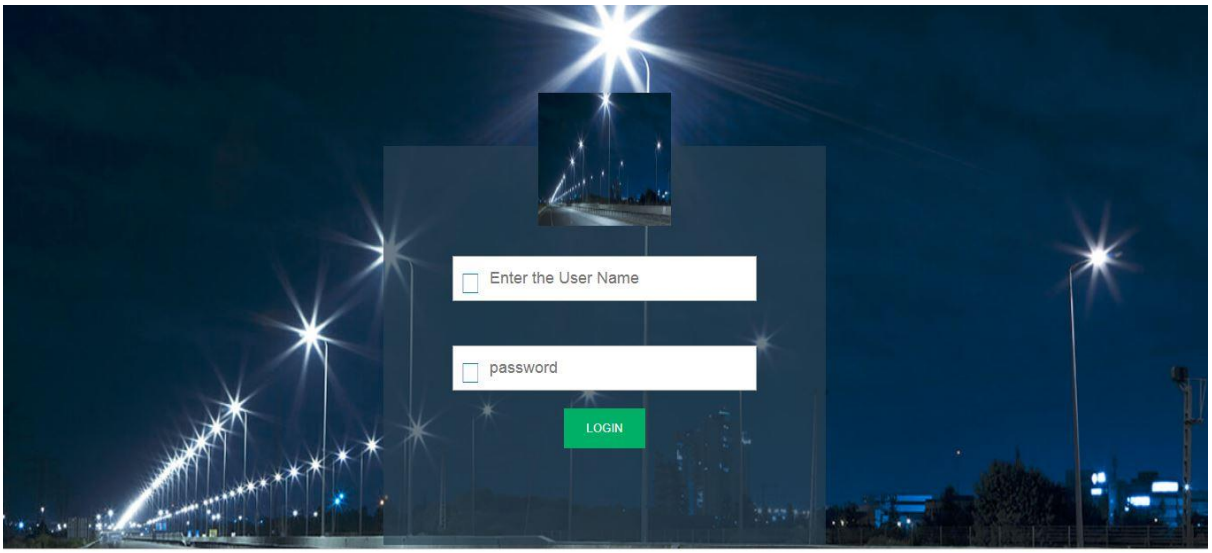

Figure 5. 4: Login Interface for admin

5.3 End-node Results

The end-node was able to send data successfully to the gateway. The data was accurate as the state of the test streetlight matched the actual state about 98% of the time. The end node was also able to turn on and off or dim the test streetlight at the right time as required. Below is a test table showing node reported state and actual state averages of the test streetlight before the threshold was adjusted.

Table 5. 1: Test results

State reported	Actual state	Number of times	Percentage	Explanation
On	Off	2/35	5.7%	False negative for ON, False positive for OFF
Dimmed	On	5/35	14.3%	False positive for Dimmed, False positive for ON
Off	Dimmed	20/35	57%	False negative for OFF, false positive for Dimmed
On	On	28/35	94.3%	True positive for ON
Dimmed	Dimmed	10/35	28.5%	True positive for Dimmed
Off	Off	35/35	100%	True positive for Off

From Table 5.1, it shows that the system reported that the test light was off when it was just dimmed. This meant that the threshold needed to be adjusted for more accurate results. Given the adjusted limits in Table 4.1 above, 40 tests were conducted using each state (ON, OFF, Dimmed) and the reported states were correct 98% of the time as shown in Table 5.2. The test LEDs also turned on or off and dimmed at the right time and when there was need.

Table 5. 2: Final test results

Actual state	Reported state	Number of times	Percentage accuracy
ON	ON	40/40	100%
OFF	OFF	40/40	100%
Dimmed	Dimmed	38/40	95%

Chapter 6: Conclusion

The expected outcome from this project is the implemented design prototype of a streetlight monitoring system that can be used to monitor streetlights in a large city. This project also has a working model that proves the following about the system.

1. Low cost
2. Intelligence (Smart)
3. Reduced power consumption
4. Improved maintenance – increased maintenance speed due to feedback on the state of streetlights
5. Reduced road accidents, crime, and insecurity due to non-functioning streetlights

6.1 Discussion

This project was able to successfully set up a LoRa node to partially monitor a single test streetlight. The LoRa network server, the LoRa App server and the LoRa gateway from Dragino were successfully configured to get data from a streetlight node to the gate and the LoRa server. The encoding and decoding JavaScript codes worked as expected and live data from the end-node were viewed on the LoRa App server.

The web application with remote monitoring interface was successfully implemented and hosted remotely. Data from the LoRa App server was queried using LoRa RESTful APIs with HTTP integration and PHP and displayed on the web application. The state of the test streetlight could be seen at every time.

The above result analysis prove that the streetlights were offered intelligence and they could report their state to the remote monitoring unit. Being able to see the state of a streetlight at all times means better and faster response to streetlight maintenance when they develop faults. Therefore, the safety and security of road users and people in dangerous

neighborhoods would not suffer. Also, the power consumption of the system is reduced by using a low-power device/technology like LoRa. The dimming mechanism included saves power on streetlights when cars are not using the streets at late nights. With streetlights being a public good, the government can save money to do other things for its population.

The cost of the system is also reduced since many end-nodes (streetlights) on the street can send data separately to only one gateway about 3-5km apart. Unlike using Wi-Fi and GSM/GPRS, once the end-node is set, they do not require any frequent maintenance or updates unless they develop a fault. They also do not need airtime or data reload, so the cost of each node is a one-time cost.

6.2 Limitations

The current limitation of this project is that it is not able to send instructions to the end-node from the web application. How can we click a button at the remote monitoring unit to say turn ON/OFF or dim a particular streetlight about 5km away? The second limitation is the fact that the end-node is designed to be powered by a battery/cell; meanwhile, there is also power in the streetlight to which it would be attached.

6.3 Future Work

The current system requires that we have an end-node on every streetlight which can be too much and stressful to fit. Future work would be to look at how to use one end-node for a group of streetlights. Also, more work could be done on implement more robust and faster dimming mechanism for the streetlight such that dimming happens at every time of the night – say maybe in a group of 5 streetlights, one or two can be dimmed even when cars are using the road to save more power. Lastly and most importantly, the streetlighting circuitry could be redesigned so that the end-node can be powered from the power going up the streetlight instead of being powered by batteries.

References

[1] Meihua Xu, Yujie Zhang and Guoqin Wang, "Design of intelligent streetlight monitoring system based on STM32," *2012 IEEE Symposium on Electrical & Electronics Engineering (EEESYM)*, Kuala Lumpur, 2012, pp. 55-58.

[2] A. Opoku, "La: Lack of streetlights threatening security", *citifmonline*, 2018.

[3] H. Lee and H. Huang, "A Low-Cost and Noninvasive System for the Measurement and Detection of Faulty Streetlights," in *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 4, pp. 1019-1031, April 2015.

[4] C. Xiao, S. Hu, and Y. Yuan, "The design of remote monitoring system for energy-saving streetlight," *2010 International Conference on Mechanic Automation and Control Engineering*, Wuhan, 2010, pp. 2750-2753.

[5] J. Hegedüs, G. Hantos and A. Poppe, "Embedded multi-domain LED model for adaptive dimming of streetlighting luminaires," *2016 22nd International Workshop on Thermal Investigations of ICs and Systems (THERMINIC)*, Budapest, 2016, pp. 208-212.

[6] C. Jing, D. Shu and D. Gu, "Design of Streetlight Monitoring and Control System Based on Wireless Sensor Networks," *2007 2nd IEEE Conference on Industrial Electronics and Applications*, Harbin, 2007, pp. 57-62.

[7] V. A. Stan, R. S. Timnea and R. A. Gheorghiu, "Overview of high reliable radio data infrastructures for public automation applications: LoRa networks," *2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, Ploiesti, 2016, pp. 1-4.

[8] "What is LoRa? | Semtech LoRa Technology | Semtech", Semtech.com, 2019. [Online]. Available: https://www.semtech.com/lora/what-is-lora. [Accessed: 13- Mar- 2019].

[9] "LoRa App Server - LoRa App Server documentation", *Loraserver.io*, 2019. [Online]. Available: https://www.loraserver.io/lora-app-server/overview/. [Accessed: 13- Oct- 2018].

[10] "Internet of Things: A Comparison of Communication Technologies", Blog - MONTEM, 2019. [Online]. Available: https://blog.montem.io/2017/03/10/internet-of-things-a-comparison-of-communication-technologies/. [Accessed: 07- Jan- 2019].

Appendix A

Listing 1: JavaScript - script for decoding payload

```
// Decode decodes an array of bytes into an object.
//  - fPort contains the LoRaWAN fPort number
//  - bytes is an array of bytes, e.g. [225, 230, 255, 0]
// The function must return an object, e.g. {"temperature": 22.5}

function Decode(fport, b) {
 /* var result = "";
      for(var i = 0; i < bytes.length; ++i){
           result+= (String.fromCharCode(bytes[i]));
      }
  var str = result ;
  var obj = JSON.parse(result);
  return obj;*/
  if ( fport == 1)
  {
  return {
        SLMstate:String.fromCharCode.apply(null, bytes)
  };}
  else if ( fport == 3)
    {
      return {
        Proximity: String.fromCharCode.apply(null, bytes)
      };
    }
  else
  {
  var lng = (b[0] | b[1]<<8 | b[2]<<16 | (b[2] & 0x80 ? 0xFF<<24 :
0)) / 10000;
var lat = (b[3] | b[4]<<8 | b[5]<<16 | (b[5] & 0x80 ? 0xFF<<24 :
0)) / 10000;
return {
latitude: lat,
longitude: lng
};
  }
}
```

Listing 2: JavaScript- script for encoding payload

```
// Encode encodes the given object into an array of bytes.
//  - fPort contains the LoRaWAN fPort number
//  - obj is an object, e.g. {"temperature": 22.5}
// The function must return an array of bytes, e.g. [225, 230, 255,
0]
function Encode(fPort, obj) {
  var object = obj;
```

```
  var bytes = [];
  var str = JSON.stringify(object);
  for (var i = 0; i < str.length; ++i) {
  var code = str.charCodeAt(i);
  bytes = bytes.concat([code & 0xff, code / 256 >>> 0]);
}
  return bytes;
}
```