TABLE OF CONTENTS

ABSTRACT

ABSTRACT

In today's modern era, the operational integrity of streetlights during nighttime is of paramount importance. However, the reliance on traditional methods such as public reporting or periodic technician surveys to identify faults is proving to be inefficient and time-consuming. To address this challenge, this project proposes an innovative IoT solution for real-time fault detection. By leveraging Light Dependent Resistor (LDR) sensors integrated with cost-effective microcontroller boards like Arduino UNO and Modular Network Modules, this system can swiftly detect changes in light intensity or the absence of light. The system captures essential information, including unique identification numbers assigned to each streetlight pole, facilitating precise location tracking and immediate fault reporting. Beyond its primary function of fault detection, this scalable IoT solution serves as a foundational framework for expanding urban infrastructure capabilities. By integrating with existing systems, it offers a versatile platform for future enhancements and innovations, supporting the development of smarter and more efficient urban lighting networks. Expected outcomes of this project encompass significantly improved response times, reduced operational costs, and overall enhancements to urban lighting infrastructure.

LIST OF FIGURES

LIST OF ABBREVATIONS

ABBREVATION	EXPANSION
LDR	LIGHT DEPENDENT RESISTOR
MQTT	MESSAGE QUEUING TELEMETRY TRANSPORT
GSM	GLOBAL SYSTEM OF MOBILE COMMUNICATION
HTTP	HYPERTEXT TRANSFER PROTOCOL
IOT	INTERNET OF THINGS
LoRa	LONG RANGE

INTRODUCTION

CHAPTER–1

INTRODUCTION

1.1 PROJECT OVERVIEW

In modern urban environments, the efficient operation and maintenance of street lighting infrastructure are critical for ensuring public safety and enhancing overall urban liveability. However, the traditional methods of detecting and addressing faults in streetlights present significant challenges, often resulting in delays, inefficiencies, and increased costs. Typically, the information regarding damaged or faulty streetlights is relayed to the Electricity Board (EB) office and subsequently to technicians through two primary channels: public reports or periodic surveys conducted by technicians.

Unfortunately, both approaches suffer from inherent drawbacks. Public reports, while potentially numerous, are often unreliable and difficult to prioritize among the vast number of reports received. On the other hand, periodic surveys conducted by technicians are time-consuming and inefficient, often taking days or even months to complete due to the sheer scale of urban environments. This inefficiency is exacerbated by the absence of viable alternatives, leaving municipalities with few options for improving the maintenance pipeline.

Recognizing the need for optimization, this project proposes a novel solution leveraging wide-scale Internet of Things (IoT) technologies to revolutionize the detection and reporting of faults in street lighting infrastructure. The proposed solution centres around the utilization of Light Dependent Resistor (LDR) sensors, which detect reductions in light intensity or the absence of light. These sensors are seamlessly integrated

into an inexpensive board, which transmits real-time information through a Modular Network Module utilizing GSM or other appropriate networking technologies to a regional hub covering a radius of several kilometres.

Critical to the effectiveness of the proposed solution is the transmission of comprehensive information, including unique identification numbers assigned to each streetlight and relevant telemetry data from the pole. The regional hub acts as a central relay point, forwarding this information to the nearest EB Office mainframe. Here, the unique identification numbers are resolved to determine the precise location of the faulty streetlight, enabling expedited dispatch or alerting of technicians for prompt repair.

By combining advanced sensor technologies with robust communication networks, this project aims to streamline fault detection and reporting processes, significantly reducing response times and improving overall urban lighting infrastructure. The integration of IoT technologies not only addresses the shortcomings of existing maintenance practices but also lays the groundwork for future enhancements and innovations in urban lighting management.

1.2 PURPOSE

The purpose of this project is to address the inefficiencies and challenges associated with traditional methods of detecting and addressing faults in street lighting infrastructure. By leveraging modern Internet of Things (IoT) technologies, the project aims to streamline the fault detection and reporting process, ultimately improving the efficiency and effectiveness of urban lighting management. Specifically, the project seeks to utilize Light Dependent Resistor (LDR) sensors integrated with inexpensive boards and Modular Network Modules to detect changes in light intensity or the

absence of light in real-time. Through the transmission of comprehensive information, including unique identification numbers and telemetry data, to regional hubs and Electricity Board (EB) Office mainframes, the project aims to expedite the dispatch of technicians for prompt repair of faulty streetlights. The overarching purpose is to optimize maintenance processes, reduce response times, and enhance the overall quality of urban lighting infrastructure, thereby contributing to safer and more sustainable cities.

LITERATURE SURVEY

CHAPTER - 2

LITERATURE SURVEY

2.1 SMART STREETLIGHTS IN SMART CITY: A CASE STUDY OF SHEFFIELD [DIZON, E., & PRANGGONO, B. 2022]

Smart streetlights can be used to enhance public safety and well-being. However, not only it is one of the most draining structures in terms of electricity, but it is also economically straining to local government. Typically, many councils adopt a static or conventional approach to street lighting, this presents many inefficiencies as it does not take into account environmental factors such as light levels and traffic flows. This paper will present the utilities of a streetlights in Sheffield and how different councils tackle the issue by using different lighting schemes. Investigation of current implementations of information and communication technologies (ICT) such as Internet of Things (IoT) in streetlights will be necessary to understand different proposed models that are used in 'smart' street lighting infrastructure. Case studies from Doncaster and Edinburgh are explored as they are using similar technology and having a similar sized topology as Sheffield. To analyze different models, StreetlightSim, an open-source streetlight simulator, is used to present different lighting schemes. The study does present an IoT embedded solution for streetlights to reduce energy consumption, it has shown different lighting schemes resulting in different reduction in energy consumption, whilst the adaptive approach remains inconclusive, it does present an obvious reduction in power consumption based on the relationship between traffic flow and total energy expenditure. The solution can be applicable to other cities in reducing energy-consumption.

2.2 SMART STREETLIGHT SYSTEM USING MOBILE APPLICATIONS: SECURED FAULT DETECTION AND DIAGNOSIS WITH OPTIMAL POWERS [M. KANTHI., & RAVILLA DILLI 2023]

The proposed work is mainly focused on the minimization of power consumption in the implementation of a smart street lighting system. Also, use a mobile application for setting up the brightness levels of the lamps in an encrypted form so that an unauthorized person will not be able to modify the settings. In the existing streetlight system, wireless sensors are installed to control and monitor the streetlamps. In the proposed system, using an nRF24L01 radio transceiver module, a secured communication link is established to operate the streetlights depending on the ambient weather conditions, movement of humans, vehicles and any other objects. A failsafe mechanism is implemented in the modules for conventional lamp operation in the case of module failures. Light-dependent resistor (LDR) is used to determine the ambient brightness levels to automatically turn on/off the streetlights based on weather conditions and lighting on roads. Using smartphones, we access and control the brightness information from the master node at which the nRF24L01 radio transceiver module is installed, and the same information is relayed to all the slave nodes. The results show that we could effectively monitor and control the brightness of streetlights in a secure way and there is a significant amount of power savings. The proposed system saves the average powers of 53.45%, 44.76%, 39.39%, and 32.25% respectively for 10%, 20%, 30%, and 50% idle mode brightness compared to the state-of-the-art techniques present in the literature.

2.3 IOT BASED AUTOMATIC DAMAGED STREET LIGHT FAULT DETECTION MANAGEMENT SYSTEM [ASHOK KUMAR NANDURI, SIVA KUMAR KOTAMRAJU, G L SRAVANTHI, SADHU RATNA BABU 2020]

The primary goal of the project is to provide control and identification of the damaged streetlight automatically. The lighting system which targets the energy and automatic operation on economical affordable for the streets and immediate information response about the street light fault. In general, the damage of the streetlight is observed by getting the complaints from the colony (street) people. Whereas in this proposed work using sensors these lights working status is easily captured without any manual interaction. So that it reduces manual efforts and the delay to fix problems. So, to reduce such problem we come with the solution wherein automatic detection of street light issues i.e., whether the streetlight is working or not will be found at nighttime, and it should send the notification to the authorized person if there is a problem in particular streetlight and also the location of the place where the streetlight is damaged. The streetlights are automatically ON/OFF using IoT. Moreover, the system is designed to conserve energy by automating the operation of streetlights based on environmental conditions. This research contributes to the smart city initiative by offering a solution for efficient street light management and energy conservation. The system's design incorporates the use of Light Dependent Resistor (LDR) sensors for light condition detection, GPS for location tracking, and cloud storage for data management. The paper underscores the significance of energy conservation and the role of IoT in automating and optimizing street light usage.

2.4 STREET LIGHT CONTROLLING AND MONITORING OF FAULT DETECTION USING LORA [N. SRAVANI, Y. LATHA, G. NIRMALA 2021]

The study presents a dual-model system for energy efficiency and fault detection in street lighting. One model utilizes IEEE 802.11 wireless technology within confined premises, while the other employs a wired setup for extended ranges, suitable for streetlamps. It focuses on reducing energy loss in street lighting by automating the lights based on environmental changes and monitoring their working status through IoT as such the system aims to conserve energy by automating street light operation based on environmental conditions and promptly detecting faults. It leverages a cloud server for data analysis and mobile-based surveillance, enabling remote monitoring and control. It utilizes Long Range (LoRa) technology for communication between streetlights and a central system, enabling remote monitoring and control of lighting systems. The integration of field sensors, LoRa gateways, and cloud platforms facilitates the communication of sensor data to master control stations. This setup allows for the dynamic control of streetlights and sends maintenance alerts for issues like burnt-out bulbs. This research contributes to the ongoing efforts in reducing energy loss in street lighting systems. By automating the streetlights' operation and enhancing fault detection, the study demonstrates the potential of IoT and LoRa in creating more efficient and responsive urban lighting systems.

2.5 SIMULATORS, EMULATORS, AND TEST-BEDS FOR INTERNET OF THINGS: A COMPARISON [N D PATEL, B M MEHTRE, RAJEEV WANKAR 2019]

The paper compares various simulators, emulators, and testbeds for IoT, focusing on their scope, type, programming language, IoT layers, scale of operation, and security measures. It highlights the strengths and weaknesses of each category, with simulators offering high-level abstraction, emulators delivering real-world execution, and testbeds providing the most accurate real-environment testing. It highlights the strengths and weaknesses of each category, with simulators offering high-level abstraction, emulators delivering real-world execution, and testbeds providing the most accurate real-environment testing. The authors discuss vertical and horizontal analysis of these tools to assist in pre-testing stages of IoT development, emphasizing the importance of choosing the right tool for different testing scenarios. The paper concludes with recommendations for specific tools in each category, such as NS-3, Bevywise-IoT, Ansys-IoT, and NCTUns 6.0 for simulators; Cooja for emulators; and FIESTA-IoT and FIT IoT-LAB for testbeds. This review should help researchers and developers in selecting the appropriate tools for their IoT projects and experiments.

SYSTEM ANALYSIS

CHAPTER – 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The current management of street lighting infrastructure relies predominantly on manual methods for fault detection and reporting. Instances of malfunctioning streetlights are primarily identified through either public reports or periodic surveys conducted by technicians. Public reports are often sporadic and unreliable, while surveys are time-consuming and may not comprehensively cover all areas. Once identified, information about faulty streetlights is forwarded to relevant authorities for resolution, and technicians are dispatched for repair or replacement. However, this manual system lacks real-time monitoring capabilities, leading to inefficiencies, delays, and increased maintenance costs.

3.2 PROBLEM IDENTIFICATION

The reliance on manual processes for fault detection and reporting in street lighting infrastructure poses several challenges. These include sporadic and unreliable public reports, time-consuming surveys, lack of real-time monitoring capabilities, and resultant inefficiencies and delays in fault detection and resolution. These issues contribute to increased downtime, maintenance costs, and a diminished quality of urban lighting infrastructure.

3.3 PROPOSED SYSTEM

To address these challenges, this project proposes the implementation of an innovative Internet of Things (IoT) solution for street lighting management. The proposed solution leverages modern sensor technologies

and real-time communication networks to automate fault detection and reporting processes. Specifically, Light Dependent Resistor (LDR) sensors integrated into inexpensive boards are utilized to detect changes in light intensity or absence of light in real-time. This information is transmitted through Modular Network Modules to regional hubs, enabling precise location determination and swift technician dispatch. By combining advanced sensor technologies with robust communication networks, the proposed solution aims to streamline fault detection, reduce response times, and improve overall urban lighting infrastructure efficiency and effectiveness.

SYSTEM SPECIFICATION

CHAPTER - 4

SYSTEM SPECIFICATION

4.1 HARDWARE REQUIREMENTS

Board	:	Arduino UNO R3 / ESP32 Devkit
Sensor	:	Light Dependent Resistor
Networking Module	:	SIM800L GSM / Wi-Fi Module in ESP32
Computer	:	PC / Laptop with USB Port

4.2 SOFTWARE REQUIREMENTS

Programming Languages	:	Python, Micropython, Arduino C++
GUI Framework	:	CustomTKinter
MQTT Broker	:	HiveMQ
DBMS	:	MySQL
Editor	:	VS Code, Arduino IDE
Simulation Tool	:	WokWi
Schematics Creation Tool	:	Fritzing
Operating System	:	Windows / UNIX Based OS

4.3 DEVICES/COMPONENTS REQUIREMENTS

Arduino UNO	:	1
Bread Board Large	:	1
SIM800L	:	1
Basic LED	:	2
LDR Sensor	:	2
Resistors 1K Ohms	:	10
Resistors 10K Ohms	:	10
Jumper Cables	:	20
LM2596 Convertor	:	1
LCD RG1602A Display	:	1

4.4 REQUIREMENTS DESCRIPTION

4.4.1 Arduino UNO R3

Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery. [6], [7]

This Project uses this MCU for its Physical implementation, However the base been tested mainly against ESP32 board. So, Replication can be done on that board instead, with the added benefit of utilizing the Wi-Fi module integrated into it, forgoing the requirement of GSM800L when prototyping.

4.4.2 SIM800L

SIM800L is a GSM module manufactured by SIMCOM, this module was chosen specifically because of its ability to work in rural environments and reduced network infrastructure usage. However, the network module is mostly

decoupled from rest of the system and hence can be changed to various other technologies like NB-IoT, LoRa Wan etc.

4.4.3 Micropython

MicroPython is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library and is optimized to run on microcontrollers and in constrained environments.

However, this project uses Micropython during the simulation testing phase. The physical implementation uses the Arduino C++. If the physical board is ESP32 then the MicroPython code can be used directly. [7]

4.4.4 CustomTKinter

CustomTkinter is a python desktop UI-library based on Tkinter, which provides modern looking and fully customizable widgets. This library has been used in this project for developing the desktop application.

4.4.5 HiveMQ

HiveMQ is an MQTT broker that facilitates message queuing and telemetry transport. It acts as the intermediary for communication between the Arduino board and the regional hub, ensuring reliable data transmission.

4.4.6 WokWi

Wokwi is an online electronics simulator that allows one to experiment with various hardware components and microcontrollers. One can use it to simulate popular boards like Arduino, ESP32, STM32, and even the Raspberry Pi Pico. This project mainly used this for testing and prototyping before the physical components have been procured.

4.4.7 Fritzing

Fritzing is an open-source hardware initiative software that makes

electronics accessible as a creative material for anyone. It is mainly used to create Schematics diagram and plan out the breadboard circuit to give a better idea of layout and structure of the circuit. Schematics present in this documentation are created using Fritzing.

SYSTEM DESIGN

CHAPTER - 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

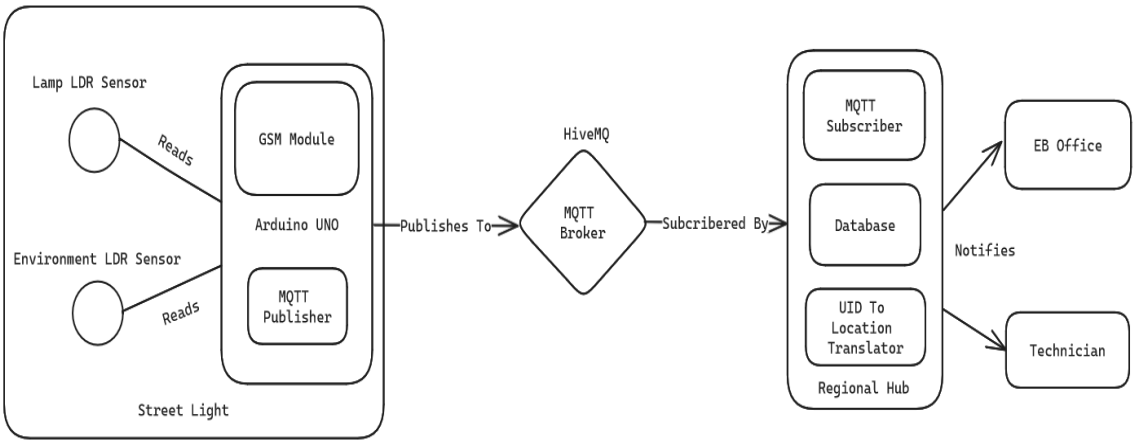

Fig - 5.1.1 System Architecture

The system incorporates two key sensors: the Lamp LDR Sensor and the Environment LDR Sensor, aimed at determining the operational status of streetlights. These sensors are seamlessly integrated with the Arduino UNO R3 Microcontroller Board. Utilizing data from both sensors, the board calculates the streetlight's operational state. Subsequently, the board generates compressed data encapsulating its unique identifier, lamp sensor value, environment sensor value, and the operational state into a 32-bit integer.

To facilitate event-based communication, the board operates an MQTT Publisher client connected to an MQTT broker. Through the SIM800L GSM Module, the board publishes the compressed data to the MQTT broker. The regional hub, subscribed to the MQTT broker, actively monitors incoming messages. Upon reception of sensor data, the regional hub decompresses the information and stores it within its local database, updating relevant tables accordingly.

In the event of a reported fault, the regional hub performs a calculation to determine the technician with the least workload and assigns them the repair task. Additionally, the hub hosts a server interface, allowing users to access telemetry data from each sensor conveniently.

This integrated approach ensures efficient fault detection, real-time data monitoring and streamlined repair assignment processes, ultimately enhancing the reliability and effectiveness of street lighting infrastructure management.

5.2 SYSTEM COMPONENTS

5.2.1 Lamp LDR Sensor and Environment LDR Sensor

These sensors are pivotal components designed to meticulously monitor light conditions in the vicinity of streetlights. The Lamp LDR Sensor focuses on detecting variations in light intensity emitted by the streetlight itself, while the Environment LDR Sensor is attuned to ambient light levels surrounding the streetlight, providing comprehensive environmental context.

5.2.2 Arduino UNO R3 Microcontroller Board

Serving as the central nervous system of the system, the Arduino UNO R3 Microcontroller Board orchestrates the intricate operations within the streetlight monitoring framework. This powerful board assimilates data streams from the LDR sensors, intelligently computes the streetlight's operational status by analyzing sensor readings and executes critical decision-making processes.

5.2.3 SIM800L GSM Module

This module plays a pivotal role in establishing wireless

connectivity, facilitating communication between the Arduino UNO board and external networks. Leveraging GSM technology, the SIM800L GSM Module empowers the system to transmit compressed data packets encapsulating vital sensor readings and streetlight status updates to designated endpoints.

5.2.4 MQTT Publisher Client

At the heart of the data transmission mechanism lies the MQTT Publisher Client, a software component integrated into the Arduino UNO board. This dynamic client application orchestrates the publication of compressed data packets to an MQTT broker, ensuring swift and efficient dissemination of critical sensor data to downstream recipients.

5.2.5 MQTT Broker

Functioning as a communication intermediary, the MQTT Broker serves as a centralized hub for message exchange between the sensor module and downstream recipients. This middleware component receives published messages from the sensor module and facilitates seamless dissemination to subscribed clients which in this case: the regional hub.

5.2.6 Regional Hub

The Regional Hub stands as a formidable bastion of data processing and analysis within the system architecture. Tasked with receiving, processing, and storing sensor data, this sophisticated component serves as the nerve center for fault detection and management. By subscribing to the MQTT Broker, the Regional Hub actively monitors incoming sensor data, analyzing and interpreting received information to derive actionable insights.

5.2.7 Server Interface

The Regional Hub boasts a server interface, crafted to provide users with intuitive access to critical telemetry data from each sensor. This user-friendly interface serves as a comprehensive dashboard, empowering users to monitor streetlight operational status, conduct historical data analysis, and derive actionable insights to optimize infrastructure management strategies.

.

MODULE DESCRIPTION

CHAPTER - 6

MODULE DESCRIPTION

6.1 MODULES

Modules can either be workflow, source file, a component, software etc., the following modules are main driving force behind the project.

1) Sensor Data Collection
2) Data Masking
3) Data Transmission
4) Data Storage
5) Data Monitoring and Visualization

6.1.1 Sensor Data Collection

There are two LDR Sensors, Lamp LDR which is placed such that it faces the bulb of the light pole, and the environment LDR which is exposed to the sky and placed such that it is isolated from the Lamp LDR. The two LDR Sensors are simultaneously read with minimal delay such as 5ms. This process is repeated for atleast 20 times for each main loop and the average of the readings is considered for further processing. This type of smoothening is called as rolling average and is done to eliminate fluctuations in the sensors, this smoothening algorithm is very simple and is good enough for this application [8]. The environment LDR gives the context like night, dusk, dawn and day, and lamp LDR must be greater than environment during night and dusk.

Since the LDR are connected in voltage divider circuit the resistance of the second resistor must be one magnitude higher than the LDR ohm range to get a readable voltage that makes sense during night and daytime [9].

6.1.2 Data Masking

Data Masking is done to pack as much data as possible within a unit of storage, therefore this step is more akin to data packing or data compression, the masking part is the one that achieves it, hence the name. The MCU would publish four pieces of data to the broker server: ID of the light pole, Lamp LDR value, Environment LDR value and finally the state of the lamp. The regional hub wouldn't manage more than 1000 light poles within its territory; therefore, the ID of the light pole can be represented in a 10-bit number instead of 4 bytes. The ADC in Arduino board would only resolve up to 10-bit (0-1023) so the values of both LDR is essentially capped at 0-1023 and finally the state/condition of the lamp would only have three states: working, sleeping and fault, which can be represented in 2 -bit number. Adding all the space of the data, we get 10+10+10+2 = 32 bits, which is an unsigned long. The data are bit-shifted and masked before payload publishing. The figure below will show the structure of the masked long in binary form.

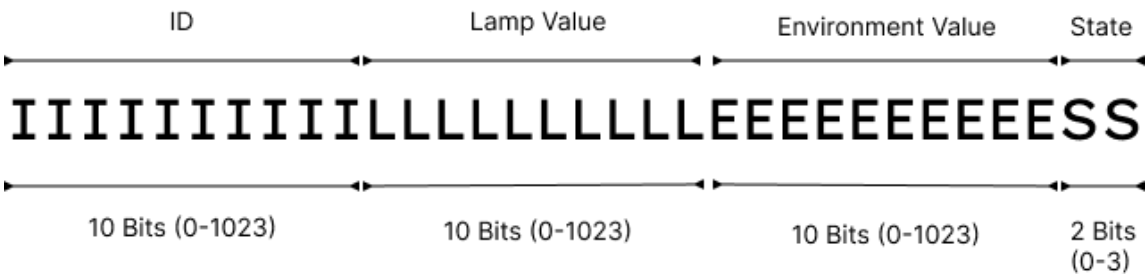

Fig - 6.1.2 Bit Masked Long Format

However, due to how the library PubSubClient is developed, there is no way to publish the integer directly, for that we need to convert the integer to string. Directly converting the data to string would completely negate the purpose of the data masking stage as each character would take a byte of memory and an 32 bit number string would have 10 characters taking up 10 bytes. Hence we convert the integer to a hex number to represent it in shorter amount of memory would be then converted to string. The client must convert the hex to int before processing.

6.1.3 Data Transmission

At the heart of this module is the SIM800L GSM module, which is responsible for establishing a cellular connection to transmit the collected data. The module is programmed to publish data at regular intervals, specifically every 10 seconds, ensuring a consistent flow of information.

During each cycle of the main loop, the Arduino microcontroller performs a check to ensure that the GSM and GPRS connections, as well as the MQTT connection to the broker, are intact and functioning properly. This vigilance is crucial to maintain uninterrupted data transmission and to quickly identify and address any connectivity issues that may arise.

The GPRS functionality of the SIM800L module is particularly utilized for the data transmission process. This allows for the efficient use of cellular data networks to send sensor data to the designated MQTT broker, which then relays the information to the appropriate endpoints for further processing and analysis.

To facilitate the communication between the hardware and the network, the project employs two libraries: TinyGSM and PubSubClient. TinyGSM provides a simplified interface for interacting with GSM modules, while PubSubClient supports the MQTT protocol, enabling the Arduino to publish and subscribe to messages within the MQTT network.

Given the SIM800L module's requirement for a substantial current supply, specifically two amperes, it is imperative to provide a stable and sufficient power source to ensure its optimal operation. This consideration is essential to prevent any power-related disruptions in the module's functionality.

Moreover, the design of the data transmission module is flexible, allowing for the SIM800L to be swapped out for other modules that may be more suitable for specific regional requirements. This adaptability ensures that the project can be tailored to various operational environments, making it a versatile solution for different geographic locations.

6.1.4 Data Storage

The project employs MySQL database for this purpose, which houses data pertaining to the streetlights, including the mapping of their unique identifiers to their respective locations. It also maintains a record of registered technicians and repair tasks, thereby facilitating efficient task assignment and tracking.

In addition to this, the MySQL database also serves as a time-series database, storing sensor data published by the streetlights. This allows for historical data analysis and trend identification, which can be instrumental in predictive maintenance and resource planning.

The choice of MySQL for this project was primarily driven by familiarity rather than efficiency. While MySQL is a robust and widely used database system, it may not be the most efficient choice for time-series data, especially when the system needs to handle multiple write operations concurrently. MySQL also lack robust support for data aggregation, which is often required when dealing with time-series data.

Given these limitations, future improvement would be to considering alternatives that are better suited for handling time-series data. PostgreSQL, when used in conjunction with its time-series database extension, could offer improved performance and more powerful data aggregation capabilities. Other dedicated time-series databases like InfluxDB could also be considered, given their optimized architecture for handling time-series data.

Another suggested approach is to use MongoDB as a collector for sensor data, while periodically updating the MySQL tables. This hybrid approach could leverage the strengths of both systems, with MongoDB handling the high-volume sensor data and MySQL managing the less frequently updated data.

6.1.5 Data Monitoring and Visualization

The Data Monitoring and Visualization module offers users a comprehensive interface to monitor sensor data and visualize the status of streetlights in real-time, ensuring efficient management of urban lighting infrastructure. Leveraging CustomTkinter, a Python GUI framework that wraps the standard Tkinter library, this module provides a user-friendly and visually appealing platform for data visualization and analysis.

At the heart of the interface lies a sensor list displayed on the right-hand side, showcasing the operational state of each sensor – "Working," "Sleeping," or "Fault" – represented by colored icons of green, yellow, and red, respectively. This intuitive representation enables users to quickly assess the status of individual sensors immediately, facilitating timely decision-making and response to detected faults.

Complementing the sensor list is an interactive map on the left-hand side, offering a geographical overview of lamp locations. The map dynamically updates to reflect the status of streetlights, allowing users to visualize sensor faults and their corresponding locations in real-time. This spatial representation enhances situational awareness and facilitates targeted maintenance interventions, optimizing resource allocation and improving overall system reliability.

In addition to sensor status visualization, the GUI incorporates a graph displaying the telemetry data of each sensor. This graph provides a visual representation of sensor readings over time, enabling users to analyze trends and fluctuations in light intensity and environmental conditions. By visualizing telemetry data, users can identify patterns, anomalies, and potential issues, empowering proactive maintenance and optimization strategies.

Moreover, the GUI seamlessly integrates with the system's backend functionality, ensuring timely updates whenever sensor state messages are posted in the MQTT queue. In the event of a detected fault, the software automatically assigns a repair task to the technician with the least workload, streamlining the maintenance process and ensuring prompt resolution of issues. This automation

enhances operational efficiency and minimizes downtime, contributing to improved urban lighting infrastructure and public safety.

Overall, the Data Monitoring and Visualization module serves as a cornerstone of the streetlight monitoring system, providing users with a powerful toolset for monitoring sensor data, managing maintenance tasks, and analyzing telemetry trends. By offering a user-friendly interface and advanced visualization capabilities, this module facilitates informed decision-making, proactive maintenance, and continuous optimization of urban lighting infrastructure.

SYSTEM TESTING

CHAPTER - 7
SYSTEM TESTING

7.1 UNIT TESTING

Completed rigorous unit testing for each module of the system, including the Sensor Data Module, Database Module, MQTT Client, and GUI Module. Utilized comprehensive test suites to verify the functionality and behavior of individual components, ensuring they meet specified requirements. Identified and addressed any defects or discrepancies within each module to enhance reliability and correctness.

7.2 INTEGRATION TESTING

Completed comprehensive integration testing to validate the interoperability and integration of different system components. Tested the interactions and interfaces between modules, such as the Sensor Data Module, Database Module, MQTT Client, and GUI Module, to ensure seamless data flow and communication. Identified and resolved any issues related to data exchange, communication protocols, or interface compatibility, ensuring smooth integration across the system architecture.

7.3 SIMULATION TESTING

The IoT System has been thoroughly tested using simulation tool called WokWi. This is done to ensure that the code behaves correctly without having to worry about damaging any real components. It also enabled us to jumpstart our development without having to wait for delivery of the physical components. The Wokwi tool also had network emulator, so the networking capability of the board is well testing even before proceeding to physical implementation of the system. It also reduced the downtime of having to wait around for recompiling and uploading the program to Arduino. However, the simulation had its drawbacks like lack of GSM module, actual light intensity emulation etc. Despite these

drawbacks, the Arduino code was tested properly before deployment.

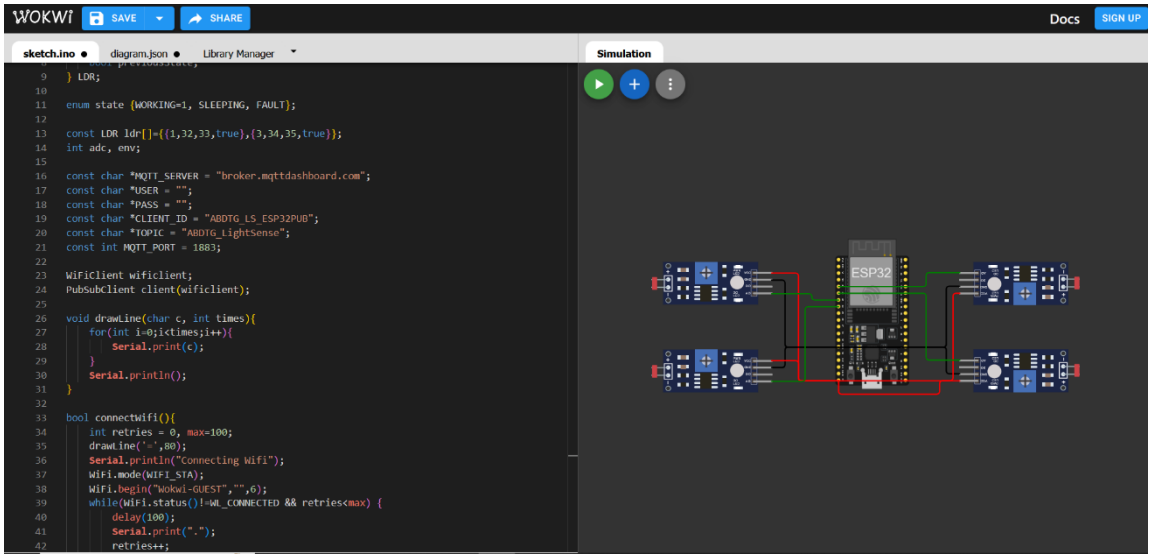

Fig - 7.3.1 Simulation Testing

SYSTEM IMPLEMENTATION

CHAPTER – 8

SYSTEM IMPLEMENTATION

8.1 CONNECTING COMPONENTS

The Components are connected according to the above diagram. The LDR sensors are connected to a Voltage divider to step down the voltage to an readable level. One sensor reads the LED and the other reads the environment. The GSM module can only work on voltages 3.4V - 4.4V both of which Arduino is incapable of providing so an external power source is required. An 2G/3G sim is then inserted into the GSM module. An LCD display also connected to display the information of each LDR sensor's intensity values and transmission information. The Final assembly would look like below implementation.

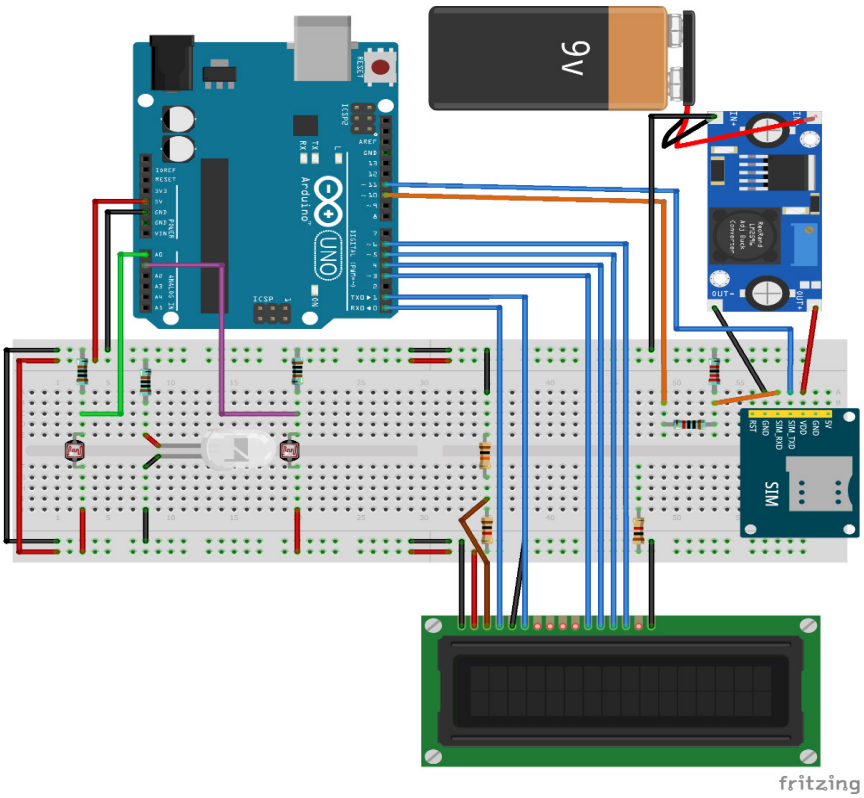

Fig - 8.1.1 Configuring the System

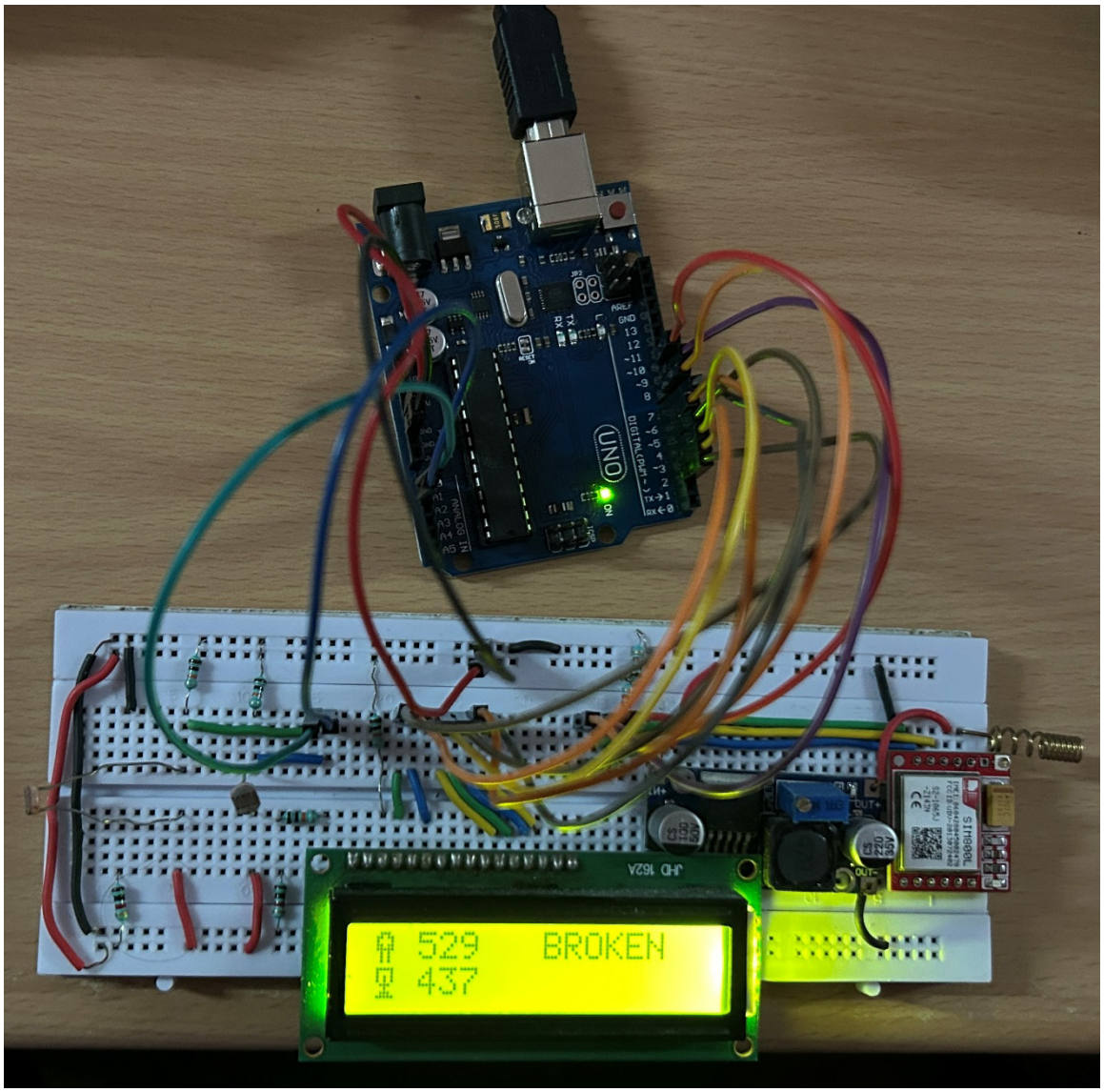

Fig - 8.1.2 Physical Implementation

8.2 CONFIGURING REGIONAL HUB

Start the MQTT Client in the Regional Hub to listen and process the messages published by the Sensors.

```
MQTT Connection Response Received
'userdata=None'
'flags=ConnectFlags(session_present=False)'
"reason_code=ReasonCode(Connack, 'Success')"
'properties=<paho.mqtt.properties.Properties object at 0x000002BD515AB690>'
Broker granted the following QoS: 0
```

Fig - 8.2.1 Configure Regional Hub

8.3 INITIALIZING THE SERVER LISTENING INSTANCE

Start the GUI Instance to view the Sensor Information, the debug is useful to quickly view the primary sensor data, for which the interface is displayed below.

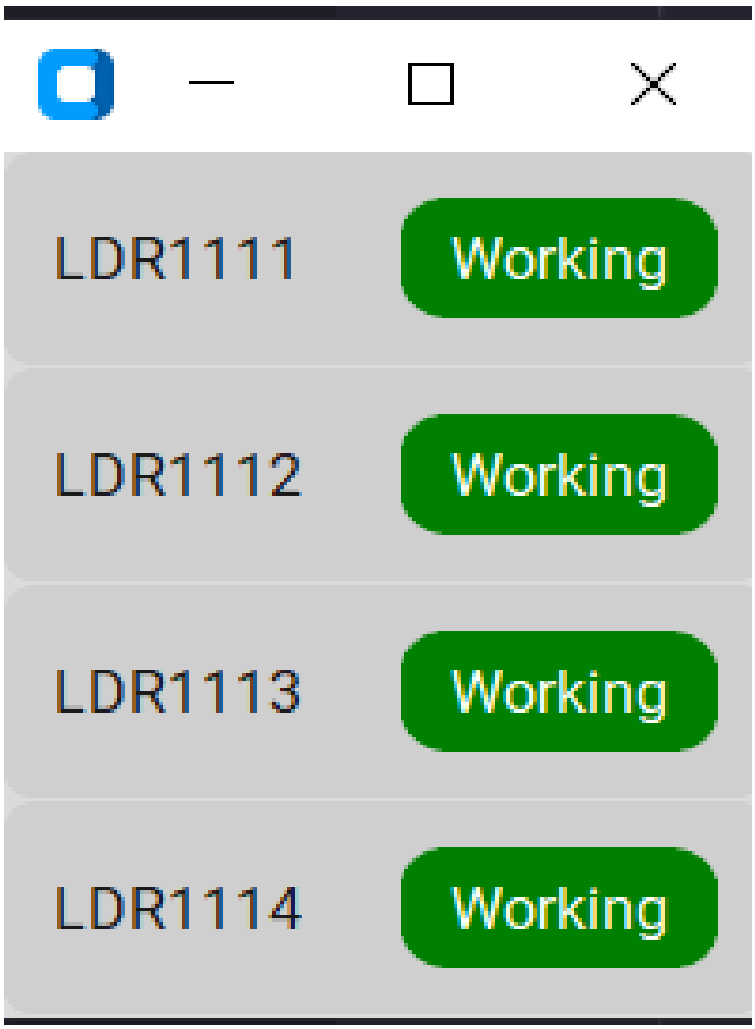

Fig - 8.3.1 Debug GUI

The main interface is much more informational dense and shows the location of each sensor their corresponding state.

Fig - 8.3.2 Desktop UI

CONCLUSION AND FUTURE ENHANCEMENT

CHAPTER–9

CONCLUSION AND FUTURE ENHANCEMENT

9.1 CONCLUSION

In conclusion, the proposed streetlight monitoring system addresses the challenges outlined in the problem statement by leveraging modern IoT technologies to enhance urban lighting infrastructure. Through the implementation of the system described in the proposal, we have provided a solution that enables real-time detection of streetlight faults, precise location tracking, and efficient maintenance processes. By utilizing Light Dependent Resistor (LDR) sensors and modular network modules, we have introduced a cost-effective and scalable approach to streetlight management. This system not only optimizes the process of reporting faults but also lays the foundation for future enhancements and innovations. With its ability to improve response times, reduce costs, and enhance overall urban lighting infrastructure, the proposed solution offers a transformative approach to streetlight maintenance in both urban and rural areas.

9.2 FUTURE ENHANCEMENT

Moving forward, several avenues for future enhancement and optimization present themselves. Firstly, incorporating machine learning algorithms can enable predictive maintenance capabilities, allowing the system to anticipate and address potential faults before they occur. Additionally, expanding the system's sensor capabilities to include environmental factors such as air quality and temperature can provide valuable insights for urban planning and environmental monitoring. Furthermore, integrating advanced analytics and visualization tools into the GUI can empower users with deeper insights into streetlight performance and energy consumption trends. Overall, continued innovation and refinement of the system hold the promise of further enhancing urban lighting infrastructure and promoting sustainable urban development.

APPENDIX

APPENDIX

A.1 SOURCE CODE

A.1.1 Database Configuration on Hub

```
DROP DATABASE IF EXISTS LightSense;

CREATE DATABASE LightSense;

USE LightSense;

CREATE TABLE State (
   id INT PRIMARY KEY,
   value VARCHAR(10)
);

CREATE TABLE StreetLight (
   id INT PRIMARY KEY AUTO_INCREMENT,
   state INT DEFAULT 1,
   coords POINT,
   updated_on TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

   FOREIGN KEY (state) REFERENCES State(id)
);

CREATE TABLE Technician (
   id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(20),
   password VARCHAR(20)
) AUTO_INCREMENT = 1000;

CREATE TABLE Repair_Assignment (
   id INT PRIMARY KEY AUTO_INCREMENT,
```

```
    light_id INT NOT NULL,
    assigned_to INT NOT NULL,
    assigned_on TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    task_state VARCHAR(10) DEFAULT "ASSIGNED",

    FOREIGN KEY (light_id) REFERENCES StreetLight(id),
    FOREIGN KEY (assigned_to) REFERENCES Technician(id),
    INDEX Light_and_taskstate (light_id, task_state),
    INDEX technician_and_taskstate (assigned_to, task_state),
    INDEX task_state (task_state)
);

CREATE TABLE SensorData (
    sensor_id INT,
    state INT,
    lamp_reading INT,
    env_reading INT,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    FOREIGN KEY (sensor_id) REFERENCES StreetLight(id),
    FOREIGN KEY (state) REFERENCES State(id)
);

DELIMITER |
CREATE TRIGGER streetligt_update BEFORE INSERT ON SensorData
FOR EACH ROW
BEGIN
UPDATE StreetLight SET updated_on=NEW.timestamp, state=NEW.state WHERE id
= NEW.sensor_id;
END |

CREATE PROCEDURE Assign_Technician_With_MinWork(IN light_id INT)
BEGIN
    INSERT INTO Repair_Assignment(assigned_to, light_id) VALUE(
```

```
        (
            SELECT t.id FROM Technician t
            LEFT JOIN Repair_Assignment rt ON t.id = rt.assigned_to AND
task_state='ASSIGNED'
            GROUP BY t.id
            ORDER BY COUNT(*)
            LIMIT 1),
        light_id );
END |

DELIMITER ;

INSERT INTO State VALUES
(1, "WORKING"),
(2, "SLEEPING"),
(3, "FAULT");

INSERT INTO StreetLight(coords) VALUES
(POINT(11.55483427,78.01969886)),
(POINT(11.55474462,78.01970609)),
(POINT(11.55465497,78.01971331)),
(POINT(11.55456532,78.01972054)),
(POINT(11.55447566,78.01972777)),
(POINT(11.55438601,78.019735)),
(POINT(11.55429636,78.01974223)),
(POINT(11.5542067,78.01974946)),
(POINT(11.55411705,78.01975668)),
(POINT(11.55403099,78.01973004)),
(POINT(11.55394493,78.01970339)),
(POINT(11.55385887,78.01967675)),
(POINT(11.55377281,78.0196501)),
(POINT(11.55368675,78.01962346)),
(POINT(11.55360069,78.01959681)),
(POINT(11.55351463,78.01957017)),
```

```
(POINT(11.55342857,78.01954352)),
(POINT(11.55334251,78.01951688)),
(POINT(11.55325645,78.01949023)),
(POINT(11.55317039,78.01946358)),
(POINT(11.55308433,78.01943694)),
(POINT(11.55299827,78.01941029)),
(POINT(11.55291221,78.01938365)),
(POINT(11.55282647,78.01935595));
```

A.1.2 GUI Code [debug.py]

```
from turtle import width
from typing import Dict
from paho.mqtt import client as mqttc
import customtkinter
import json


class LDR(customtkinter.CTkFrame):
    def __init__(self, master, id, status = True, **kwargs):
        super().__init__(master, **kwargs)
        self.id = id
        self.status = status
        self.nameLabel: customtkinter.CTkLabel = customtkinter.CTkLabel(self,
text=id)
        self.stateLabel: customtkinter.CTkLabel = customtkinter.CTkLabel(self,
corner_radius=10, text_color="white")
        self.nameLabel.grid(row=0, column=0, padx=10, pady=10)
        self.stateLabel.grid(row=0, column=1, padx=10, pady=10)
        self.refresh_status()

    def set_status(self, status: bool):
```

```
        self.status = status
        self.refresh_status()

    def refresh_status(self):
        if self.status:
            self.stateLabel.configure(text="Working", fg_color="green")
        else:
            self.stateLabel.configure(text="Not Working", fg_color="red")

class StatusFrame(customtkinter.CTkFrame):
    def __init__(self, master, ldr_list= [], **kwargs):
        super().__init__(master, **kwargs)
        self.ldrs:Dict[str, LDR] = {}
        self.ldr_count = 0
        # self.grid_columnconfigure(0, weight=1)

        for i in ldr_list:
            self.add_ldr(i)

    def add_ldr(self, ldr_id, inital_status=True):
        ldr = LDR(self, ldr_id, inital_status)
        ldr.grid(row = self.ldr_count, column=0)
        self.ldr_count+=1
        self.ldrs[ldr_id] = ldr

    def get(self, id):
        return self.ldrs[id]

    def set_ldr_status(self, id, status):
        self.ldrs[id].set_status(status)
```

```
app: customtkinter.CTk = customtkinter.CTk()
app.title("LigthSense Debug Portal")
app.grid_columnconfigure(0, weight=1)
LDR_SENSORS = ["LDR1111", "LDR1112", "LDR1113", "LDR1114"]
ldr_panel: StatusFrame = StatusFrame(app, LDR_SENSORS, width=500)
ldr_panel.grid(row=0, column=0)
# def createFrame(name: str, master: customtkinter.CTk):
#     frame = customtkinter.CTkFrame(master)
#     nameLabel = customtkinter.CTkLabel(frame,text=name)
#     nameLabel.grid(row = 0, column=0, padx=10, pady=10)
#     stateLabel = customtkinter.CTkLabel(frame, text="working",
fg_color="green", corner_radius=10, text_color="white")
#     stateLabel.grid(row = 0, column=1, padx=10, pady=10)
#     return [frame, nameLabel, stateLabel]

# ldr = {}
# for j,i in enumerate(LDR_SENSORS):
#     ldr[i] = LDR(app,i)
#     ldr[i].grid(row=j,column=0)

TOPIC = "ABDTG_LightSense"
PORT = 1883
SERVER = "broker.mqttdashboard.com"

def on_connect(client: mqttc.Client, userdata, flags: mqttc.ConnectFlags,
reason_code: mqttc.ReasonCode, properties ):
    print("MQTT Connection Response Received ")
    print(f"\t{client=}")
    print(f"\t{userdata=}")
    print(f"\t{flags=}")
```

```
    print(f"\t{reason_code=}")
    print(f"\t{properties=}")
    if(reason_code.is_failure):
        print("MQTT Server Failed To Connect")
    else:
        client.subscribe(TOPIC)

def on_subscribe(client, userdata, mid, reason_code_list, properties):
    if reason_code_list[0].is_failure:
        print(f"Broker rejected you subscription: {reason_code_list[0]}")
    else:
        print(f"Broker granted the following QoS: {reason_code_list[0].value}")

def on_message(client, userdata, message):
    payload:dict = json.loads(message.payload)
    for name,status in payload.items():
        ldr_panel.get(name).set_status(status.upper()=="WORKING")
    print(payload)

MQTTC = mqttc.Client(mqttc.CallbackAPIVersion.VERSION2,
protocol=mqttc.MQTTv5)



MQTTC.on_connect = on_connect
MQTTC.on_subscribe = on_subscribe
MQTTC.on_message = on_message



print("Connecting to MQTT Server")

MQTTC.connect(SERVER, PORT)
```

```
MQTTC.loop_start()

app.mainloop()

MQTTC.loop_stop()
```

A.1.3 Arduino Code [sketch.ino]

```
#include <WiFi.h>

#include <PubSubClient.h>


typedef struct {

    int id;

    int lamp;

    int env;

    bool previousState;

} LDR;


enum state {WORKING=1, SLEEPING, FAULT};


const LDR ldr[]={{1,32,33,true},{3,34,35,true}};

int adc, env;
```

```
const char *MQTT_SERVER = "broker.mqttdashboard.com";

const char *USER = "";

const char *PASS = "";

const char *CLIENT_ID = "ABDTG_LS_ESP32PUB";

const char *TOPIC = "ABDTG_LightSense";

const int MQTT_PORT = 1883;


WiFiClient wificlient;

PubSubClient client(wificlient);


void drawLine(char c, int times){

    for(int i=0;i<times;i++){

        Serial.print(c);

    }

    Serial.println();

}


bool connectWifi(){

    int retries = 0, max=100;
```

```
drawLine('=',80);
Serial.println("Connecting Wifi");
WiFi.mode(WIFI_STA);
WiFi.begin("Wokwi-GUEST","",6);
while(WiFi.status()!=WL_CONNECTED && retries<max) {
   delay(100);
   Serial.print(".");
   retries++;
}
Serial.println();

if(retries>max){
   drawLine('=',80);
   Serial.println("Wifi Connection Failed!");
   return false;
} else {
   Serial.println("Wifi Connected");
   drawLine('=',80);
   return true;
}
```

```
}

bool connectMQTT(){
    int retries = 0, max = 5;
    drawLine('=',80);
    Serial.println("Connecting To MQTT Server");
    client.setServer(MQTT_SERVER, MQTT_PORT);
    while(!client.connected() && retries < max) {
        if(client.connect(CLIENT_ID, USER, PASS)){
            Serial.println("MQTT Server Connected Successfully!!");
            drawLine('=',80);
            return true;
        } else {
            Serial.print("Connection Failed With State :");
            Serial.println(client.state());
            Serial.println("Retrying");
            retries++;
            drawLine('-',80);
        }
    }
```

```
    return false;
}

unsigned long mask(int id, int ldr, int global, int state){
    unsigned long data = id;
    data <<= 10;
    data |= ldr;
    data <<= 10;
    data |= global;
    data <<=2;
    data |= state;

    return data;
}

void unmask(unsigned long masked){
    drawLine('-',80);
    Serial.print("State : ");
    Serial.println(masked & 3);
    masked >>= 2;
```

```
    Serial.print("Global : ");
    Serial.println(masked & 1023);
    masked >>= 10;
    Serial.print("LDR : ");
    Serial.println(masked & 1023);
    masked >>= 10;
    Serial.print("ID : ");
    Serial.println(masked & 1023);
}

void setup(){
    Serial.begin(115200);
    if(connectWifi()==false){
        return;
    }
    if(connectMQTT()==false){
        return;
    }
}
```

```
void loop(){
   if(WiFi.status()!=WL_CONNECTED) {
      connectWifi();
   }
   if(!client.connected()){
      connectMQTT();
   }

   for(int i=0;i<2;i++){

      adc = analogRead(ldr[i].lamp);
      env = analogRead(ldr[i].env);
      int state = adc<env?(env>600 ? SLEEPING : FAULT ) : WORKING;
      unsigned long data =  mask(ldr[i].id, adc, env, state);
      unmask(data);
      if(client.publish(TOPIC,String(data).c_str())){
         Serial.println("Published");
      } else {
         Serial.println("Not Published");
      }
```

```
        drawLine('-',80);
    }


    delay(1000);
}
```

A.2 SCREENSHOTS

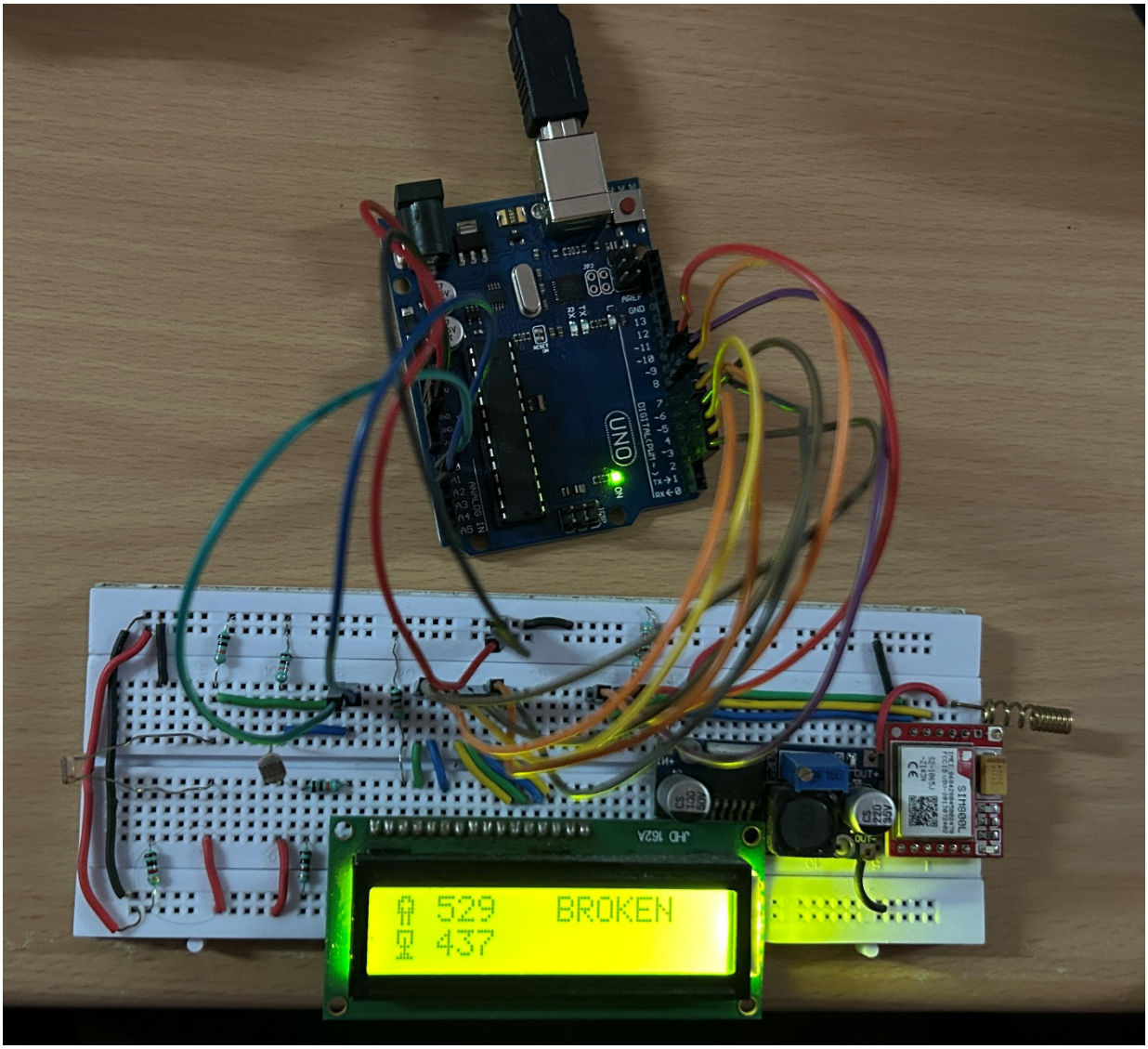

Fig – A.2.1 Physical Implementation

A.2.2 Desktop GUI

REFERENCES

REFERENCES

[1] E. Dizon and B. Pranggono, "Smart streetlights in Smart City: a case study of Sheffield," Journal of Ambient Intelligence and Humanized Computing, 2022.

[2] M. D. R. Kanthi, "Smart streetlight system using mobile applications: secured fault detection and diagnosis with optimal powers," 2023.

[3] B. M. M. a. R. W. N. D. Patel, "Simulators, Emulators, and Test-beds for Internet of Things: A Comparison," in Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2019.

[4] A. K. Nanduri, S. K. Kotamraju, G. L. Sravanthi, S. R. Babu and K. V. K. V. L. P. Kumar, "IoT based Automatic Damaged Street Light Fault Detection Management System," International Journal of Advanced Computer Science and Applications(IJACSA), 2020.

[5] N. Sravani and Y. L. a. G. Nirmala, "Street Light Controlling and Monitoring of Fault Detection using LoRa," International Journal for Modern Trends in Science and Technology, 2021.

[6] "Arduino Uno Rev3," [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno.

[7] "MicroPython - Python for microcontrollers," [Online]. Available: https://micropython.org/.

[8] T. O'Haver, "Smoothing," [Online]. Available: http://terpconnect.umd.edu/~toh/spectrum /Smoothing.html.

[9] A. Meyer, "Adam Meyer," [Online]. Available: https://adam-meyer.com/arduino/LDR.

[10] Intelilight, "Automatic Street Lighting," [Online]. Available: https://intelilight.eu/automatic-street-lighting/.

PUBLICATION