

**IOT-BASED INTELLIGENT STREETLIGHTS SYSTEM WITH FAULT-
TOLERANT MECHANISM USING SENSOR FUSION**

By
Saw Wei Chin

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

May 2023

REPORT STATUS DECLARATION FORM

Title: IoT-Based Intelligent Streetlights System with Fault-Tolerant Mechanism using Sensor Fusion

Academic Session: May 2023

I SAW WEI CHIN
(CAPITAL LETTER)

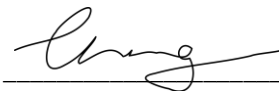
declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

163B-05-03 Centrio
Avenue, 11700 Gelugor,
Penang

Ts Dr Chang Jing Jing

Supervisor's name

Date: 15/9/2023

Date: 15/9/2023

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 15/9/2023

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that Saw Wei Chin (ID No: 20ACB05406) has completed this final year project entitled "IOT-BASED INTELLIGENT STREETLIGHTS SYSTEM WITH FAULT-TOLERANT MECHANISM USING SENSOR FUSION" under the supervision of Ts Dr Chang Jing Jing (Supervisor) from the Department of Computer and Communication Technology, Faculty of Information and Communication Technology .

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,



Saw Wei Chin

DECLARATION OF ORIGINALITY

I declare that this report entitled “**IOT-BASED INTELLIGENT STREETLIGHTS SYSTEM WITH FAULT-TOLERANT MECHANISM USING SENSOR FUSION**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :  _____

Name : Saw Wei Chin

Date : 15/9/2023

ACKNOWLEDGEMENTS

I want to convey my heartfelt gratitude to my supervisor, Ts Dr Chang Jing Jing, and my moderator, Ts Dr Tong Dong Ling, for granting me this invaluable opportunity to participate in a project centered around embedded systems and the Internet of Things (IoT). This venture marks my initial stride towards building a career in the field of embedded systems. I extend my deepest appreciation to both of you for making this possible.

Lastly, I cannot fail to acknowledge the unwavering support, love, and unceasing encouragement that my parents and family have bestowed upon me throughout this journey. Their belief in my pursuits has been an immense source of strength and motivation. Thank you from the bottom of my heart.

ABSTRACT

This project introduces an Intelligent Streetlight System that seamlessly integrates Cloud and Internet of Things technology to address the inherent limitations of traditional streetlights. Conventional streetlights are renowned for their excessive energy consumption and the absence of automated fault detection. To optimize energy utilization while upholding safety standards, the system incorporates sensor fusion technology, which aggregates data from multiple sensors, thereby ensuring more dependable outcomes. This fusion of sensor technology with the Intelligent Streetlight System significantly elevates the system's reliability, availability, and safety. Furthermore, the project implements a meticulously designed fault tolerance mechanism, underpinned by fuzzy rules, enhancing the system's dependability by substantially reducing the likelihood of potential failures. The culmination of this project yields a fully operational prototype of the Intelligent Streetlight System, featuring an embedded fault tolerance architecture based on fuzzy logic. These innovations collectively represent a sustainable, efficient, and highly reliable alternative to conventional streetlighting, offering benefits including reductions in energy consumption, streamlined maintenance processes, and an elevated standard of safety. Significantly, this project seamlessly integrates Cloud Technology and IoT capabilities, facilitating data collection, real-time monitoring, and collaborative data sharing among streetlights, thus enhancing the system's capabilities for urban infrastructure management. This initiative presents an innovative response to pressing urban and rural infrastructure challenges and holds the potential to make substantial contributions to the advancement of sustainable development goals.

TABLE OF CONTENTS

DECLARATION OF ORIGINALITY	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	x
LIST OF TABLES	xii
LIST OF ABBREVIATIONS	xiii
CHAPTER 1 – PROJECT BACKGROUND	1
1.1 Introduction	1
1.2 Problem Statement and Motivation	3
1.3 Objectives	4
1.4 Project Scope	4
1.5 Contributions	6
1.6 Report Organization	6
CHAPTER 2 – LITERATURE REVIEW	8
2.1 Previous Studies	8
2.1.1 Sensor fusion	8
2.1.2 Internet of Things	9
2.1.3 Fuzzy Logic	9
2.1.4 Fault-tolerant	10
2.1.5 Intelligent Streetlight System	10
2.2 Limitations of Previous Studies	11
2.3 Proposed Solutions	12
CHAPTER 3 – SYSTEM METHODOLOGY/APPROACH	14
3.1 System Requirement	14
3.1.1 Hardware	14
3.1.2 Software	14
3.2 System Overview	15
3.3 System Architecture	16
3.4 Cloud Architecture in Arduino Cloud	17
3.5 Fuzzy Logic	18

3.6	Policy for fault-tolerant	20
3.7	Verification Plan.....	22
CHAPTER 4 – SYSTEM DESIGN		26
4.1	Design Specification	26
4.2	System Block Diagram.....	26
4.3	System Data Flow Diagram.....	27
4.4	System Schematic Diagram	28
4.5	Fault-Tolerant Design	30
4.6	User Interface Design (UI Design)	33
CHAPTER 5 – SYSTEM IMPLEMENTATION		37
5.1	Component and Technology Involved.....	37
5.1.1	Hardware	37
5.1.2	Software	41
5.1.3	Circuit	42
5.2	Setup	43
5.2.1	Setup Arduino IoT Cloud.....	43
5.3	Fault Tolerance Policy Diagram	44
5.4	Fuzzy Logic Algorithm	46
5.4.1	Fuzzy Rules.....	47
5.5	Implementation Issue and Challenges.....	51
CHAPTER 6 – SYSTEM DELIVERABLE AND TESTING		53
6.1	System Deliverable	53
6.1.1	Fault-Tolerant Mechanism	53
6.1.2	System Prototype	53
6.1.3	Dashboard Interface	56
6.2	System Testing	58
6.2.1	Automated Brightness Adjustment (TC1).....	58
6.2.2	Fault Tolerance (TC2)	61
6.2.3	Fault Detection and Notification (TC3)	62
6.2.4	Internet of Things (IoT) (TC4)	63
6.2.5	Test Result	64
CHAPTER 7 – CONCLUSION		66
7.1	Conclusion.....	66

7.2	Novelties and Contributions	67
7.3	Future Works	67
	REFERENCES	69
	APPENDIX	A-1
	Poster	A-1
	Steps to Setup Arduino Cloud	A-2
	Source Code	A-16
	Plagiarism Check Result	A-32
	Weekly Report	A-33

LIST OF FIGURES

Figure Number	Title	Page
Figure 3.1	System Architecture Diagram	17
Figure 3.2	Cloud Architecture Diagram	18
Figure 4.1	System Block Diagram	27
Figure 4.2	Data Flow Diagram	28
Figure 4.3	System Schematic Diagram	29
Figure 4.4	Fault-Tolerant Architecture Diagram	31
Figure 4.5	Dashboard's UI (PC view)	34
Figure 4.6	Dashboard's UI (Smartphone view)	35
Figure 5.1	Arduino Nano 33 IoT	37
Figure 5.2	NodeMCU ESP8266	37
Figure 5.3	Passive Infrared Sensor (PIR)	38
Figure 5.4	Light Dependent Resistor (LDR)	38
Figure 5.5	ADS1115 Module	39
Figure 5.6	Light-emitting Diode (LED)	40
Figure 5.7	Resistor	40
Figure 5.8	Jumper Wires and Breadboard	41
Figure 5.9	System Schematic Diagram	43
Figure 5.10	Policy diagram for LDR Sensor	45
Figure 5.11	Policy diagram for PIR	45
Figure 5.12	Policy diagram for Cloud data	46
Figure 6.1	Top View of System Prototype	54
Figure 6.2	Front View of System Prototype	55

Figure 6.3	Side View of System Prototype	55
Figure 6.4	System Prototype used to Test System Fault Detection and Notification Features	56
Figure 6.5	Screenshot of Dashboard	57
Figure 6.6	Screenshot of Serial Monitor 1	59
Figure 6.7	Screenshot of Serial Monitor 2	60
Figure 6.8	Screenshot of Serial Monitor 3	61
Figure 6.9	Screenshot of Dashboard of Streetlight 1	62
Figure 6.10	E-mail Notification	63
Figure 6.11	Screenshot of Dashboards during Testing at time 08:24:19	64
Figure 6.12	Screenshot of Dashboards during Testing at time 08:24:20	64

LIST OF TABLES

Table Number	Title	Page
Table 3.1	Specifications of laptop	14
Table 3.2	Levels of input of LDR parameter	19
Table 3.3	Levels of input of PIR parameter	19
Table 3.4	Levels of output	19
Table 3.5	Verification Plan 1	22
Table 3.6	Verification Plan 2	23
Table 3.7	Verification Plan 3	24
Table 3.8	Verification Plan 4	25
Table 4.1	Dashboard Component Description	35
Table 5.1	Decision Table summarise RULE 1 and RULE 2	47
Table 5.2	Decision Table summarise RULE 3 and RULE 4	48
Table 5.3	Decision Table summarise RULE 5 and RULE 6	48
Table 5.4	Decision Table summarise RULE 7, RULE 8, and RULE 9	49
Table 5.5	Decision Table summarise RULE 10 and RULE 11	50
Table 5.6	Decision Table summarise RULE 12, RULE 13, RULE 14, RULE 15, and RULE 16	51
Table 6.1	Test Result table	63

LIST OF ABBREVIATIONS

<i>ADDR</i>	Address
<i>FIS</i>	Fuzzy Inference System
<i>GSM</i>	Global System for Mobile Communication
<i>G pin</i>	Ground Pin
<i>GND</i>	Ground
<i>IDE</i>	Integrated Development Environment
<i>IoT</i>	Internet of Things
<i>IR</i>	Infrared
<i>LDR</i>	Light Dependent Resistor
<i>LDR-e</i>	Light Dependent Resistors sensor use to collect environmental brightness
<i>LDR-e (others)</i>	Light Dependent Resistors sensor use to collect environmental brightness from the other lamps
<i>LDR-e (self)</i>	Light Dependent Resistors sensor use to collect environmental brightness build on the lamp
<i>LED</i>	Light-Emitting Diode
<i>MATLAB</i>	Matrix Laboratory
<i>MUX</i>	Multiplexer
<i>NoSQL</i>	Not only SQL
<i>NREL</i>	National Renewable Energy Laboratory
<i>OUT pin</i>	Output Pin
<i>PC</i>	Personal Computer
<i>PIR</i>	Passive Infrared
<i>PIR1</i>	Passive Infrared Sensor 1
<i>PIR2</i>	Passive Infrared Sensor 2
<i>PRE</i>	Pre-lighting – indicator use to indicate whether the streetlight needs to perform advance lighting
<i>PWM</i>	Pulse Width Modulation
<i>SCL</i>	Serial Clock
<i>SDA</i>	Serial Data
<i>SQL</i>	Structured Query Language

<i>UI</i>	User Interface
<i>VCC</i>	Voltage Common Collector
<i>VDD</i>	Voltage Drain to Drain
<i>Wi-Fi</i>	Wireless Fidelity

CHAPTER 1 – PROJECT BACKGROUND

1.1 Introduction

Streetlight is an infrastructure that plays a significant role in illuminating the street at night for both rural and urban area. This infrastructure not only helps in enhancing the visibility of streets, but also builds a safer environment for the citizens. However, the streetlight consumes the highest electrical energy, a total electricity consumption of 40% of the cities [1].

An intelligent streetlight system is a type of smart city technology that aims to improve energy efficiency, reduce costs, and enhance safety in urban and rural areas. These systems use sensors and advanced analytics to automatically adjust lighting levels based on the time of day, traffic volume, and other environmental factors.

The intelligent streetlight systems help to reduce the energy consumption and costs of a city. For example, a study by the National Renewable Energy Laboratory (NREL) found that smart street lighting systems can reduce energy use by up to 80% compared to traditional street lighting. Other than that, according to the study of Energy Savings Trust, they found that upgrading street lighting systems to smart LED technology can result in significant cost savings for local governments over the long term. These systems reduce the energy consumption by adjusting the light intensity based on the traffic flow or turning off lights while the environment is bright enough. In term of enhancing public safety, intelligent streetlight system can detect and report outages, which allow for faster repairs and reduce the risk of accidents happened in the area that implement the intelligent streetlight system.

Sensor fusion is a technique that combines data from multiple sensors to provide a more reliable and complete understanding of the environment. The goal of sensor fusion is to provide a more reliable and comprehensive interpretation of input data compared to the data obtained only from one sensor. Therefore, this technique is suitable to be used in the intelligent streetlight system to combine data from various sensors that are implemented in the system such as motion sensor and light sensor to determine the appropriate light intensity for a given time and location. For example, a motion sensor can detect the level of activity on road, while a light sensor can measure the ambient light levels in that area. By combining the data sources from these sensors, the system can determine the most appropriate lighting levels to ensure optimal visibility and safety for the road users.

Sensor fusion can also help in reducing the impact of individual sensor limitations or errors. For example, a light sensor may not accurately detect the light levels in a shaded area but when combine with data from other sensors, the system can still make accurate decisions about the appropriate lighting levels.

However, in reality the system is unable to make a decision by using the data collected from these sensors. It is because the data collected from these sensors is imprecise and uncertain. Therefore, fuzzy logic plays a significant role in the intelligent streetlight system to analyse the data collected from the sensors. Unlike the traditional Boolean logic, which operates on binary values, fuzzy logic allows for a range of values between true and false. Which makes it more suitable for dealing with real-world problems that involve ambiguity and vagueness.

Fuzzy Logic is based on the concept of fuzzy sets, which represent a group of objects that have degrees of membership to the set. In other words, a fuzzy set can have elements that are not completely in the set, but rather partially belong to it. This allows fuzzy logic to oversee situations where there do not have a clear-cut boundary between objects or where there is uncertainty in the data.

In addition, the Internet of Things (IoT) has been incorporated and it has significantly bolstered the capabilities of this system. IoT, an acronym for the Internet of Things, represents a revolutionary technological paradigm that interconnects a wide array of physical objects, devices, and machinery through the internet. This integration of IoT within our system empowers these "things" to seamlessly collect and exchange data not only among themselves but also with centralized systems. Furthermore, it facilitates the real-time transmission of data from the system to cloud-based repositories, thus enhancing its data processing and analytics capabilities.

Overall, intelligent streetlight systems could make the process of lighting up the street more cost and energy efficient, more sustainable, and safer. Where sensor fusion and fuzzy logic is an important technique for intelligent streetlights system with fault detection and fault tolerance, as it allows systems to make a more accurate decision based on a wide range of environmental data sources. These systems are becoming more popular around the world, especially in those cities that looking to reduce the cost and energy consumption without sacrificing public safety and quality of life at the same time.

1.2 Problem Statement and Motivation

There are several limitations and problems with the traditional streetlight systems. These limitations and problems can affect their effectiveness, efficiency, and safety in the community.

First, the **high energy consumption** issue is well known by the public. The streetlights were typically implemented using high-pressure sodium lamps that consume a lot of energy when operating. These streetlights lead to a high operating cost and environmental impact due to those streetlights being light up based on the predetermined schedule.

Secondly, the **flexibility of the traditional streetlight systems is low**. Traditionally the streetlight systems are often designed to operate at a fixed level of brightness which the light intensity of the streetlight is unable to adjust based on the needs or the conditions on the road.

To reduce the electricity consumption, replacement of traditional High-Pressure Sodium streetlight to LED lamp has been implemented along with intelligent streetlight systems. The intelligent streetlight system controls the intensity of the streetlights depending on the traffic. For example, the intensity of the streetlight decreases automatically when the flow of traffic reduces based on the information gathered from the sensor until the dawn [2-3].

Other than that, the **maintenance of the traditional streetlight systems is inefficient**. It is because the traditional streetlight system does not have the sensor implemented in it, thus the technician might need to perform **manual inspection and maintenance**. Manual inspection and maintenance in a wide area can be time-consuming and costly.

In addressing the challenges associated with conventional streetlights, the concept of intelligent street lighting has emerged as a subject of extensive research and development. Nonetheless, it is evident that many of the proposed solutions in this domain **lack in establishing communication among the streetlights** themselves. The incorporation of intercommunication between streetlights enhances the system's overall intelligence, leading to superior performance when compared to solutions that lack such connectivity. This interconnectedness empowers the system to gather a wealth of information, subsequently facilitating more optimal decision-making processes.

Lastly, the **limited control** of the traditional streetlight systems. Traditional streetlight systems do not provide real-time monitoring of the road. This causes the difficulty to respond to changing the light intensity needs or emergencies.

To address these limitations and problems, intelligent streetlight systems based on sensor fusion using fuzzy logic is proposed to provide a more effective, efficient, and flexible lighting solution that reduces energy consumption and enhances public safety. By implementing sensors and an advanced control algorithm, the intelligent streetlight system can adjust the light intensity of the streetlights based on real-time conditions automatically and optimize the energy consumption. Moreover, the system can provide real-time monitoring and control that make it easier to respond to changing lighting needs or emergencies and make the maintenance works more efficient, by automating the inspection of the streetlight condition.

1.3 Objectives

The objectives of this project are as follows:

- To develop an intelligent streetlight system that adjusts the intensity of light based on the light intensity of the environment and traffic flow using sensor fusion.
- To design a fault-tolerant architecture for an intelligent streetlight system.
- To implement fault-tolerant mechanism and Internet of Things (IoT) in the intelligent streetlight system with cloud technology.

1.4 Project Scope

The main project scope is to develop an intelligent streetlight system with the implementation of sensor fusion technology that can reduce the energy consumption of streetlights based on the light intensity of its surroundings and the flow of traffic. This will make sure that the streetlights will have appropriate intensity when needed. This system will be developed by using the light dependent resistor sensor and motion sensors to help in adjusting the light intensity of the streetlight based on the information gathered from the sensors. Internet of Things (IoT) has been integrated into this project to enhance the system's intelligence and provide it with a more comprehensive understanding of the environment. To enable the realization of the IoT concept in this project, cloud services will be employed. This facilitates information sharing and communication among microcontrollers. Additionally, it enables the system to notify administrators about the status of the streetlights. The integration of cloud services also allows for real-time monitoring and data collection, which streamlines maintenance processes and grants improved remote control. The system proposed is to enable minimization of the energy consumption of streetlights to save energy.

Fault detection and fault tolerance are critical components of any system that operates in the real world. In this project, the system is designed to be fault-tolerant by utilizing sensor fusion technology. Fault detection is used to detect any malfunction or fault in the sensors or the system itself. By detecting these faults, the system can automatically trigger a fault tolerance mechanism to mitigate the impact of the faulty sensor.

The implementation of fault detection and fault tolerance can improve the reliability and availability of the system. With fault detection, the system can detect faults before they become a problem, thus reducing downtime and maintenance work duration. This is accomplished by automating the inspection process, which reduces the need for manual inspections.

In addition, the fault tolerance function ensures that the system can still provide reliable output even when multiple sensors are down. By incorporating this function into the system, the decision on the light intensity of streetlights is more dependable and accurate. Overall, the implementation of fault detection and fault tolerance is essential in enhancing the system's safety, reliability, and availability.

In this project, several hardware and software tools are required to implement the proposed intelligent streetlight system. Firstly, the Arduino Cloud will be used as the software tool to develop the logic and code for the system, which will be uploaded to the microcontroller board. The microcontroller board serves as the main microcontroller board, which will host and integrate the sensors and LED components onto it.

Two types of sensors will be used to gather the necessary data for the system to function properly. The first type is the Passive Infrared (PIR) sensor, which will be used to detect any motion or movement in the environment. This data will help the system in determining whether there are pedestrians or vehicles on the street. The second type is the Light Dependent Resistor (LDR) sensor, which will be used to determine the light intensity of the environment.

Cloud integration is incorporated to facilitate seamless communication and information sharing among microcontrollers, ensuring the dependability of the gathered data. The inclusion of cloud services in this initiative empowers the system to promptly alert the responsible individual in case of malfunctions. Furthermore, the incorporation of cloud services elevates the system's sophistication. The utilization of cloud services within this framework is aimed at enabling the realization of the IoT concept. This entails microcontrollers exchanging information, engaging in communication, and notifying administrators about the streetlight

status. Through the integration of cloud services, the system also enables real-time monitoring, thereby expediting maintenance procedures and enhancing remote control capabilities.

Finally, fuzzy logic will be implemented to provide a decision-making mechanism for the system. The fuzzy logic will be integrated between the hardware components to make the system more intelligent and effective in reducing energy consumption.

1.5 Contributions

The integration of cloud services brings substantial contributions to the project. It enhances data reliability by enabling seamless microcontroller communication and information exchange, ensuring dependable data collection. Swift fault detection and alerts are achieved, minimizing downtime. Additionally, cloud integration advances system capabilities, aligning with IoT principles. Real-time monitoring empowers remote oversight and maintenance, optimizing efficiency and control. In essence, cloud services enhance data quality, fault management, system capabilities, and remote control, culminating in an optimized intelligent streetlight system.

1.6 Report Organization

The report is organized into seven well-structured chapters, each following a specific outline. Chapter one serves as an introductory cornerstone, encompassing critical elements such as the problem statement, background, motivation, scope, objectives, and the project's contributions. Moving on to chapter two, we delve into the realm of literature reviews, where we identify the shortcomings of previous research and propose potential solutions.

In the third chapter, we provide a comprehensive overview of the system, covering its architecture, policies, and test plan. Chapter four delves into the intricate details of the system design, presenting a system block diagram, data flow diagram, system schematic diagram, fault-tolerance concepts, and an explanation of the UI design.

Chapter five then shifts focus to the system's components and technologies, offering insights into system setup, the logic implemented within it, and the challenges faced during implementation. In the subsequent chapter, chapter six, we explore the system deliverables and report the results of rigorous system testing.

CHAPTER 1

Finally, chapter seven serves as the conclusion, summarizing the project's essential details, highlighting its novelties and contributions, and providing a roadmap for the next steps in its implementation.

CHAPTER 2 – LITERATURE REVIEW

Streetlights bring a lot of benefits to the cities at night. These includes reducing the risk of accidents, reducing the chance of crime, and light up the way for pedestrians. However, traditional streetlights will still be turning on in their maximum light intensity even during the late night when the streets are empty. This has caused unnecessary waste of energy, which is not environmentally friendly and costly. Therefore, the intelligent streetlight system has been introduced by many people to reduce the energy consumption of streetlights.

This chapter summarises what was the previous related work, and which features and functionalities that we may adapt into our new proposed system.

2.1 Previous Studies

2.1.1 Sensor fusion

Sensor fusion, also referred to as sensor data fusion or multi-sensor fusion, is a process that involves combining or integrating data from multiple sensors to achieve a more accurate, comprehensive, and reliable understanding of the environment or the object under surveillance and control [4]-[6]. This technique has proven invaluable in compensating for the inherent limitations of individual sensors, resulting in reduced errors, increased redundancy, and enhanced performance across various conditions. As a result, sensor fusion has become a critical component of modern sensing and perception systems [11].

In [11], it is highlighted that ensuring data reliability often necessitates the integration of new, more accurate sensors. However, replacing older sensors may not always be economically justifiable. Instead, a recommended approach is to have new sensors operate in parallel with the existing ones. While the older sensors may not match the accuracy of the new ones, they still contribute to improving the quality of measurements obtained from the new sensors.

The potential of integrating sensor fusion into intelligent streetlight systems has been well-documented in [5], [7], [8], and [12]. Furthermore, the practical application of sensor fusion technology has been validated for enhancing both the intelligence and dependability of intelligent streetlight systems, as demonstrated in [12]. This underscores the significant impact of sensor fusion in improving the performance and reliability of the intelligent streetlight system.

2.1.2 Internet of Things

The Internet of Things (IoT) refers to a network of interconnected physical objects or "things" embedded with sensors, software, and other technologies. These objects, which can encompass a wide range of items, including household appliances, vehicles, industrial machines, wearable devices, and even entire buildings or cities, are designed to collect and exchange data with other devices and systems via the internet. The core concept of IoT is to enable these objects to communicate and share information, thereby facilitating improved automation, monitoring, and control of various processes and environments. IoT implementations can be categorized into centralized and decentralized approaches. In centralized systems, such as the one described in [5], data is collected from sensors and transmitted to a central server for processing, after which the results are sent back to the streetlights. In contrast, decentralized systems, as exemplified in [15], involve data collection and processing directly by embedded microcontrollers within the streetlights.

2.1.3 Fuzzy Logic

Fuzzy logic is a mathematical framework designed to address uncertainty and imprecision in decision-making and problem-solving. Unlike classical binary logic, which operates with strict true or false values, fuzzy logic permits the representation of degrees of truth or membership in a set using values between 0 and 1. This flexibility proves especially beneficial in situations where boundaries between categories or states lack clear definition [10]. Fuzzy logic relies on membership functions to specify how an input value pertains to a particular fuzzy set, fuzzy rules to articulate relationships between input and output variables using linguistic terms, and a fuzzy inference system to process these rules and generate output values.

The integration of sensor fusion and fuzzy logic has resulted in the development of a more intelligent streetlight system, as evidenced in [7], [8], and [12]. A method for fusing data from multiple sensors, encompassing two components—Fuzzy Aggregator and Fuzzy Predictor—has been introduced in [11]. While both components have their merits, the Fuzzy Aggregator may be sufficient for the intelligent streetlight system, given its lower complexity and reduced computational demands compared to the Fuzzy Predictor.

2.1.4 Fault-tolerance

In the realm of digital systems, faults can arise from a multitude of sources, including hardware failures, software glitches, data corruption, or computational errors. Ensuring the reliability and robustness of the system is of paramount importance, and this is where the concept of fault tolerance comes into play. Fault tolerance represents a crucial technique employed to avert system failures stemming from such faults.

At its core, fault tolerance relies on the principle of redundancy. This strategy entails duplicating critical components, data, or computational processes within the system. By adopting redundancy, the system can continue to operate seamlessly, even if certain components or data exhibit faults or defects. Victor P. Nelson's work, as documented in this paper, underscores the significance of specific elements that should be integrated into fault tolerance strategies to enhance the system's reliability and availability. These elements encompass masking, detection, containment, diagnosis, repair/reconfiguration, and recovery [9].

It is noteworthy that fault tolerance not only enhances system reliability but also significantly reduces the likelihood of system failures. This is achieved through automated mask, check, detect, diagnose and recovery mechanisms, as elucidated in references [16] - [18]. These mechanisms play a pivotal role in swiftly restoring the system to a functional state, further fortifying its resilience against potential disruptions.

2.1.5 Intelligent Streetlight System

Intelligent streetlight systems, often referred to as smart streetlight systems, are innovative urban infrastructures that seamlessly integrate advanced technologies and automation to revolutionize the management and operation of street lighting. Their primary objectives encompass a wide range of benefits, including heightened energy efficiency, reduced maintenance costs, enhanced road safety, and the introduction of intelligent functionalities to infrastructures.

For instance, in [2], an intelligent streetlight system leverages raw data from Light Dependent Resistors (LDR) and Infrared (IR) sensors, enabling real-time monitoring through the Bolt IoT platform. Similarly, [5] introduces an Internet of Things (IoT)-based streetlight system that utilizes Zigbee technology for wireless communication with lamp modules. This system employs dual LDR sensors to monitor day-night variations and lamp health, transmitting

processed data via Zigbee to a control center for comprehensive monitoring and management. Both systems intelligently adjust light intensity based on the presence of nearby objects and ambient light levels. However, they rely on straightforward if-else rules rather than adopting the more sophisticated fuzzy logic-based decision-making.

In contrast, sensor fusion integrated with fuzzy logic has been demonstrated in [7], [8], [12], [14], and [15], highlighting the potential for more nuanced and adaptable control strategies. Moreover, these implementations vary in architecture and control methods, with [5] employing a centralized server for system control while others opt for decentralized approaches. Common components across these systems include LDR sensors, microcontrollers, PIR sensors, and servers, all playing crucial roles in orchestrating efficient streetlight management.

For example, [14] proposes an IoT-based smart streetlight management system featuring LDR sensors, relays, microcontrollers, ESP-12 Wi-Fi modules, and Temperature-Humidity sensors. On the other hand, [15] introduces an automated streetlight monitoring system utilizing Xen and GSM technology.

It is worth noting that while systems like those in [2], [8], and [14] consider light intensity for intelligent luminous control as either ON or OFF, [5], [7] and [12] introduce an additional option—DIM—when the environment is dark but devoid of traffic. Moreover, [15] considers the colour of the light to ensure visibility in high level of PM2.5 environments.

One critical aspect often overlooked by most systems is fault detection. However, [12] stands out by incorporating fault detection mechanisms that promptly notify operators of issues, enabling swift resolution.

In summary, intelligent streetlight systems represent a significant leap in urban infrastructure, offering a plethora of advantages. These systems leverage diverse technologies and control methodologies, with the potential for further refinement through sensor fusion and advanced decision-making, such as fuzzy logic-based approaches. They play a pivotal role in fostering more efficient, safe, and intelligent urban environments.

2.2 Limitations of Previous Studies

Based on the review of previous studies, it is evident that most intelligent streetlight systems have made significant strides in enhancing energy efficiency when compared to traditional streetlights. However, there remains a critical gap in ensuring safety and security within these

systems. Notably, studies such as [2], [5], [7], [8], [14], and [15] have not implemented fault detection mechanisms to identify potential streetlight failures. Even in the case of [12], where a fault detection mechanism was employed, it lacked a robust fault-tolerance strategy to address and rectify identified issues, thus ensuring uninterrupted system functionality.

It is important to underscore that while these studies have achieved commendable outcomes, they have not adequately addressed fault detection and fault tolerance concerning the hardware components integral to intelligent streetlight systems. The absence of such mechanisms can jeopardize the reliability and availability of the system. Hardware faults can result in streetlight failures, leading to potentially severe consequences such as accidents, reduced safety, and increased energy consumption.

Hence, a critical consideration in the hardware design of intelligent streetlight systems is the incorporation of robust fault detection and fault tolerance mechanisms. These elements play a pivotal role in identifying and rectifying errors or faults within the system's hardware, software, or data, ensuring its continued reliability and availability.

Furthermore, while sensor fusion has been employed in these projects to enhance data collection and decision-making, it is important to acknowledge that sensor data accuracy can be compromised due to inherent noise, as discussed in the previous chapter. Therefore, the integration of redundant sensors becomes imperative to further refine the accuracy of sensed data, thereby bolstering the overall effectiveness of intelligent streetlight systems.

2.3 Proposed Solutions

The primary aim of this project is to create a cutting-edge intelligent streetlight system that utilizes multiple sensors to detect environmental conditions and intelligently adjust the intensity of streetlights accordingly. By employing sensor fusion, which combines data from various sensors to provide a more accurate environmental assessment, this system will be equipped to make informed decisions about optimizing light levels. Additionally, we will design and implement a robust fault-tolerance architecture based on fuzzy logic. This architecture will enable the system to swiftly identify any anomalies or malfunctions in sensors or streetlights and take necessary corrective actions. To further strengthen the fault-tolerant mechanism and enhance overall system reliability and resilience, we will integrate IoT techniques using cloud architecture into the project. Our ultimate goal with this endeavor is to

CHAPTER 2

significantly boost energy efficiency and cost-effectiveness by optimizing energy usage for street lighting, thus minimizing waste and reducing expenses.

CHAPTER 3 – SYSTEM METHODOLOGY/APPROACH

3.1 System Requirement

3.1.1 Hardware

This project utilizes several hardware components to function, including a computer, Microcontroller boards with Wi-Fi Module, Passive Infrared (PIR) sensors, Light Dependent Resistor (LDR) sensors, multiplexers (MUX), resistors, and LEDs. The computer serves a crucial role in this system, facilitating tasks such as code implementation, compilation, and code uploading to the microcontroller. The microcontroller board, equipped with programmed fuzzy logic, plays a pivotal role in processing inputs from various sensors and determining the LED's brightness level. Key sensors in this process include PIR sensors and LDR sensors, which collect environmental data and transform it into usable values fed into the microcontroller as inputs. To bridge the analog-to-digital gap, a multiplexer (MUX) is employed. Ultimately, the LED functions as a streetlight, delivering the necessary illumination.

Description	Specifications
Model	Acer Swift 5 (SF514-55T)
Processor	Intel Core i5-1135G7
Operating System	Windows 11
Graphic	Intel Iris Xe Graphics
Memory	8GB LPDDR4 RAM
Storage	512GB PCIe SSD

Table 3.1: Specifications of laptop

3.1.2 Software

The project utilizes the Arduino Web Editor and Arduino IoT Cloud as its software tool, which enables the implementation and compilation of code and fuzzy logic for uploading to the Microcontrollers. Arduino Agent is needed to enable the used of Arduino Web Editor and Arduino IoT Cloud. The specific version of Arduino Agent utilized is 1.3.2.

3.2 System Overview

The system prototype serves as a fundamental component of the project, mimicking the functionality of a streetlight. To demonstrate the system's capability to communicate with one another, this prototype is replicated into three instances within this project. Each prototype consists of a comprehensive set of components, including three Light Dependent Resistors (LDRs), two Passive Infrared (PIR) sensors, one Multiplexer (MUX), one microcontroller, one Light Emitting Diode (LED) and three 10k Ω resistors.

This project encompasses both hardware and software components, with the microcontroller assuming a central role. It processes all incoming data, adjusts the LED output, and facilitates communication between the prototypes through the cloud. Multiple PIR sensors are integrated to detect movement within the vicinity. The system also incorporates LEDs, serving as virtual streetlights to indicate the operational status of the system.

Two types of LDRs are employed: one functions as a light-controlled variable resistor, continuously monitoring the ambient light intensity and modulating the brightness of the LED streetlights accordingly. The second type of LDR serves as a fault-detection component, reading the LDR value of the streetlight. If the LDR value remains high when the LED should be lit, it signifies a fault within one of the streetlights. The inclusion of various sensors is pivotal in achieving fault tolerance within the system. For example, if one sensor were to fail, a backup sensor would seamlessly take over to ensure uninterrupted system availability. However, due to the constraints imposed by the finite number of sensors, achieving a consensus through a majority vote may not always be feasible. In such cases, fault tolerance mechanisms come into play to ensure the system's uninterrupted operation. The fault tolerance framework within the system places paramount importance on environmental safety, aiming to maximize LED output even in the presence of sensor errors, all the while promptly alerting the responsible party to address the underlying issue.

Furthermore, in this project, the software tool employed for programming and code upload to the microcontroller is the Arduino Web Editor. This tool offers cloud-based storage for projects, ensuring secure online storage of code and related files. This not only guarantees data backup but also enables seamless synchronization across different devices. Additionally, the project leverages the Arduino IoT Cloud to facilitate communication between microcontrollers and provide real-time monitoring and control capabilities to the user.

This comprehensive approach combines both hardware and software elements to create a robust and fault-tolerant intelligent streetlight system, demonstrating its effectiveness in enhancing energy efficiency and safety while ensuring seamless communication and monitoring through cloud technology.

3.3 System Architecture

The system architecture diagram is depicted in Figure 3.3. This project involves a system prototype comprising three streetlights, interconnected through the cloud via Wi-Fi. Each streetlight is equipped with LED lighting, an LDR sensor, and a PIR sensor. Additionally, a user-friendly dashboard has been developed to facilitate real-time monitoring and control.

The system operates as follows: Initially, the LDR sensors on each streetlight collect environmental light intensity data, which is then uploaded to the cloud via the internet. Among the streetlights, one is designated as the master, which processes and uploads this data back to the cloud. The other streetlights utilize the processed data to make decisions. Importantly, the system has fault tolerance mechanisms in place. If the master streetlight experiences a failure during this process, another streetlight is promptly selected as the master to handle data processing. Similarly, if a slave streetlight loses its internet connection, it processes its data based on its locally collected information.

Subsequently, motion is detected through the motion sensors of the streetlights, and this data is combined with the previously processed data. The microcontroller within each streetlight leverages this information to determine the appropriate output for the LED lighting. Even in cases where conflicting data is received from the motion sensors, the microcontroller can still make suitable lighting decisions, because of the implementation of fault tolerance mechanisms. The motion data collected is also uploaded to the cloud, enabling the next streetlight in line to anticipate and provide the appropriate lighting level. This implementation exemplifies the power of IoT, rendering the entire system more intelligent. For instance, when a car passes a streetlight's motion sensor, the current and next streetlights can increase their light intensity, enhancing road visibility for the driver.

All real-time data collected allows responsible personnel to monitor the system through the dashboard. This dashboard provides an interface for viewing the environmental situation via computers or smartphones. Furthermore, users can download collected data from the dashboard for future analysis or machine learning purposes. Additionally, the dashboard grants control

over the streetlights, featuring an emergency switch that activates maximum lighting output when needed during emergencies.

The system also boasts fault detection and notification features. These functionalities enable the system to identify faults, such as master or slave disconnections from the internet, sensor malfunctions, LED failures, and more. When a fault is detected, the system triggers a cloud alarm, which, in turn, sends an email notification to the responsible personnel, specifying which streetlight encountered the error and the time of occurrence.

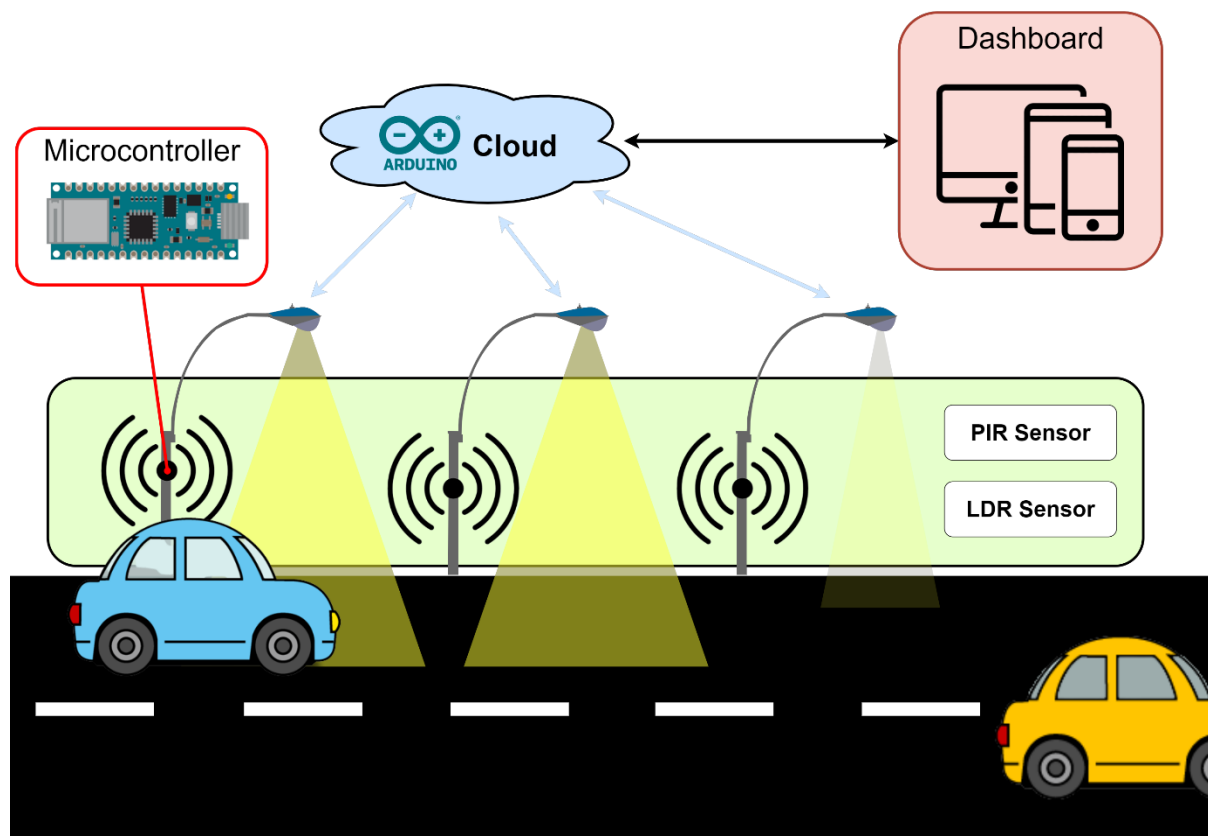


Figure 3.1: System Architecture Diagram

3.4 Cloud Architecture in Arduino Cloud

Figure 3.2 provides an illuminating glimpse into the cloud architecture diagram of our system. In order for the microcontroller to seamlessly upload the sensed data to the Arduino Cloud, it's imperative to initiate the registration process within the Arduino IoT Cloud, thereby establishing a secure connection with the Arduino Cloud infrastructure.

Facilitating data sharing necessitates the declaration of variables within the Arduino IoT Cloud, serving as a temporary repository for the collected data. An integral component of this

architecture is the Cloud Trigger function, intricately connected to the Arduino Cloud, as it relies on the functionalities of the Arduino IoT Cloud.

When the Cloud Trigger is activated, it initiates a notification mechanism, promptly dispatching messages to specified email addresses. This feature ensures that pertinent individuals are promptly informed of critical events or conditions within the system.

Furthermore, the user is afforded a dynamic interface to engage in real-time monitoring and control of the system through our meticulously designed dashboard. This user-friendly dashboard not only facilitates immediate system oversight but also provides the option to retrieve historical data stored within the Cloud. This historical data can serve as a valuable resource for in-depth analysis and decision-making processes.

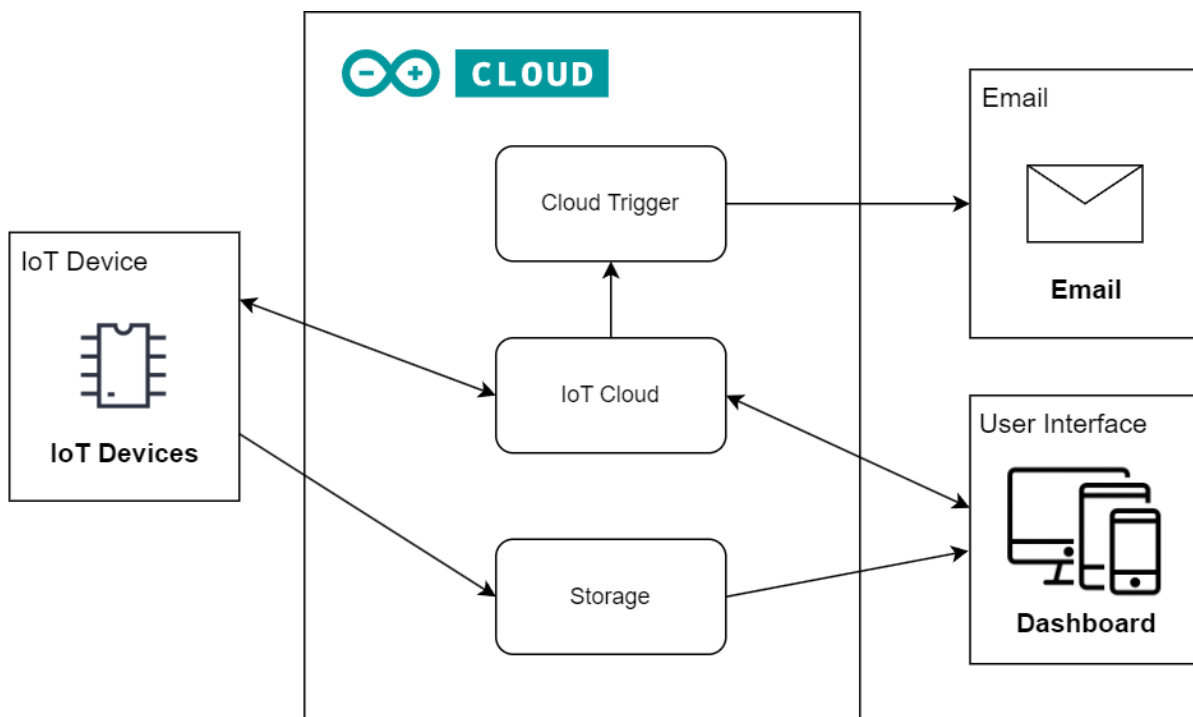


Figure 3.2: Cloud Architecture Diagram

3.5 Fuzzy Logic

This system utilizes the embedded fuzzy logic library, a predefined software resource developed by the Robotic Research Group (RRG) at the State University of Piauí (UESPI-Teresina), which is readily accessible through the Arduino Library. This library employs Mamdani Fuzzy Inference Systems. Prior to implementing fuzzy logic rules, it is essential to categorize the inputs and outputs into predefined levels. These categorizations facilitate the

establishment of fuzzy rules for processing inputs and determining the appropriate output. For a detailed overview of these input and output levels, please refer to Tables 3.2, 3.3, and 3.4.

Input of LDR	
Condition	Description and Range
Dark	Low light observed in the area (0 – 450)
Average	Average light observed in the area (350 – 550)
Bright	High light observed in the area (450 – 1023)

Table 3.2: Levels of input of LDR parameter

Table 3.2 presents the LDR input categorized into three distinct levels. The "Dark" level spans from 0 to 450, defining conditions when the ambient light intensity is low. The "Average" level encompasses a range of 350 to 550, signifying situations where the light intensity in the environment is moderate. Lastly, the "Bright" level extends from 450 to 1023, indicating instances when the light intensity within the environment is high. These delineated ranges provide clear thresholds for assessing and classifying light conditions, facilitating precise environmental monitoring and control.

Input of PIR	
Condition	Description and Range
High	Moving object detected (1)
Low	No moving object detected (0)

Table 3.3: Levels of input of PIR parameter

Table 3.3 illustrates the PIR input, categorized into two distinct levels. The "High" level is represented by the value 1, indicating the detection of a moving object. Conversely, the "Low" level is denoted by the value 0, signifying the absence of any detected moving objects. These binary states provide a clear and straightforward means of identifying the presence or absence of motion within the monitored area, enabling precise motion detection and control.

Output to LED	
Condition	Description and Range
High	Maximum brightness of LED (100 – 255)
Medium	Average brightness of LED (50 – 150)
Low	Low brightness of LED (0 – 75)

Table 3.4: Levels of output

Table 3.4 provides an overview of the LDR output, categorized into three distinct levels. The "High" level spans from 100 to 255, signifying when the LED output is at its maximum brightness. The "Medium" level encompasses a range of 50 to 150, indicating conditions where the LED output is maintained at an intermediate, or average, level of brightness. Lastly, the

"Low" level extends from 0 to 75, defining situations where the LED output is set to a low level of brightness. These delineated ranges serve as clear guidelines for adjusting the LED output to match the desired lighting conditions, offering precise control over the illumination intensity.

3.6 Policy for fault-tolerant

Ensuring ample street lighting is a critical element in safeguarding the safety of system users. Effective street illumination not only mitigates accidents and deters criminal activities but also enhances visibility, instilling users with a profound sense of security and comfort. Thus, it is imperative to ensure the system's functionality even in the presence of faults, a mechanism often referred to as fault tolerance. To implement such a mechanism, we must first design a fault-tolerant policy, which necessitates an understanding of the potential fault-prone areas within the system.

The initial focal point concerns hardware faults, particularly sensor-related issues and noise interference. To address these concerns, our proposed system incorporates redundant sensors and leverages sensor fusion techniques to enhance the reliability of collected data. However, achieving this reliability often entails the use of a majority technique, which, in turn, requires additional sensors to filter out noise and biased data, potentially impacting the system's cost-effectiveness. Striking a harmonious balance between safety and cost, our system integrates fuzzy logic, enabling the system to accommodate partial truths and degrees of uncertainty. Nonetheless, given the presence of only one redundant sensor, the system may encounter scenarios where multiple sensors yield conflicting data—such as one light sensor indicating a "dark" environment while another perceives it as "bright." In such instances, our policy is designed to assume both sets of data are erroneous, opting for the safest course by considering the environment dark and delivering maximum LED output to ensure well-lit streets. A similar approach is employed with motion sensors; if one sensor detects motion while another remains passive, the system prudently assumes motion is detected and maximizes LED output.

The second facet of concern pertains to connectivity faults, specifically regarding the reliability of the link between streetlights and the cloud. Given the master-slave architecture of our system, wherein all streetlights rely on data processed by the master streetlight, we must address potential internet connectivity issues. To avert a collapse of the streetlight system due to a fault in the master streetlight's internet connection, our system is designed to designate another

streetlight as the master when a fault is detected. This proactive measure ensures system availability during such connectivity disruptions. Nevertheless, this primarily addresses instances of the master's disconnection from the internet. But what about when a slave streetlight loses connectivity? In our system, we've also contemplated this scenario. By enabling streetlights to make decisions based on their locally collected data when they recognize an inability to retrieve data from neighbouring streetlights, we ensure continued functionality. The system constantly monitors data from other streetlights, flagging a disconnection if the received data remains static for an extended period. Moreover, if data from all other streetlights ceases to update, the system identifies itself as disconnected and transitions to decision-making based on its own data.

Our system goes beyond fault tolerance by empowering active supervision through the integration of monitoring and notification functions. These functions provide responsible personnel with real-time insights into the status of hardware components and connectivity, granting them the ability to proactively detect anomalies or potential issues. By continuously evaluating the system's health and performance, the monitoring functions not only contribute to maintaining seamless operations but also enable swift intervention when necessary.

Furthermore, the alerting function serves a critical role in scenarios where the system encounters challenge it cannot autonomously resolve. Take, for instance, the case of an LED within a streetlight burning out. In such instances, the system may lack the capability to restore the LED's functionality, particularly if a backup LED is unavailable, which is often the case. It is precisely during these situations that the alerting function becomes a vital component of our fault-tolerant mechanism.

By promptly notifying external personnel or administrators about the issue, the system ensures a timely response and resolution. This external input becomes instrumental in addressing complex or non-self-recoverable problems, thereby minimizing downtime and service disruptions. Through the seamless coordination of these fault tolerance mechanisms, our system achieves not just fault tolerance but an elevated level of robustness and resilience. It not only safeguards against potential failures but also actively manages and addresses them, ultimately enhancing its overall dependability and effectiveness.

3.7 Verification Plan

The intelligent streetlight system is designed to reduce the power consumption of streetlight when there is low traffic flow with fault tolerance control. Nevertheless, the system will still be affected by some situations according to different circumstances. These situations will also lead to some problem that occurs in the system. Thus, before any of the problems occur or happen in running the system after implementing it, verification plans are required for further enhancement. The verification plans are provided below:

1. The intelligent streetlight system able to adjust the light intensity of the streetlight based on the traffic flow.
2. The intelligent streetlight system can still work as expected if there is a faulty motion sensor.
3. The intelligent streetlight system can notify the responsible personnel when LED fault is detected.
4. The streetlights can exchange information among themselves.

Verification Plans

Automated Brightness Adjustments

Plan ID	P1
Method	Black box testing
Requirement	Lower the intensity of the LED when no motion detected, higher the intensity of the LED when motion detected.
Aim	To ensure the system able to adjust the light intensity based on the traffic flow.
Item Under Test	Microcontroller, PIR sensor, LED
Precaution	The testing should be done in dark environment.
Limitation	The system must be connected to a computer to access the serial monitor.
Equipment Required	- An object act as car or human
Data Recorded	Light intensity value observed from the serial monitor.
Acceptance Criteria	Observe dimmer LED light when no object passing by and brighter LED light when object passing by.

Procedures	<ol style="list-style-type: none"> 1. Prepare an object to act as a car or human. 2. Place the prototype in a dark environment. 3. Observe the LED light intensity from the system serial monitor. 4. Record down the value observe from the serial monitor. 5. Make the object pass by the PIR sensors. 6. Observe the LED light intensity from the system serial monitor or using eye observation. 7. Record down the value observe from the serial monitor. 8. Compare the value observe.
Troubleshooting	<ul style="list-style-type: none"> - Make sure the PIR sensors are facing to an empty space. - Make sure the serial monitor is connected.
Post-Test Activities	None.

Table 3.5: Verification Plan 1

Fault Tolerance

Plan ID	P2
Method	Black box testing
Requirement	Higher light intensity of LED when object passing by while one PIR sensor is down.
Aim	To ensure the system can still work as expected if there is a faulty motion sensor
Item Under Test	Microcontroller, PIR sensors, LED
Precaution	The testing should be done in dark environment, block one PIR sensor using a cupboard.
Limitation	The system must be connected to a computer to access the serial monitor.
Equipment Required	<ul style="list-style-type: none"> - An object act as car or human. - A cupboard.
Data Recorded	Light intensity value observed from the serial monitor.
Acceptance Criteria	The LED has higher light intensity when object pass through
Procedures	<ol style="list-style-type: none"> 1. Prepare an object to act as a car or human and a cupboard. 2. Place the prototype in a dark environment. 3. Block a PIR sensor using a cupboard. 4. Observe the LED light intensity from the system serial monitor. 5. Record down the value observe from the serial monitor. 6. Make the object pass by another PIR sensor. 7. Observe the LED light intensity from the system serial monitor or using eye observation. 8. Record down the value observe from the serial monitor.

	9. Compare the value observe.
Troubleshooting	<ul style="list-style-type: none"> - Make sure a PIR sensor is facing to an empty space, while another is blocked by a cupboard. - Make sure the serial monitor is connected.
Post-Test Activities	None.

Table 3.6: Verification Plan 2

Fault Detection and Notification.

Plan ID	P3
Method	Black box testing
Requirement	Notify the responsible personnel when LED is not working
Aim	To ensure the system can notify the responsible personnel when LED fault is detected.
Item Under Test	Microcontroller, LDR sensors, LED
Precaution	<ul style="list-style-type: none"> - The testing should be done in dark environment. - Make sure the prototype has been connected to the Wi-Fi. - Make sure the LDR sensors use to detect the light intensity of LED does not irradiate by the LED.
Limitation	The prototype must be connected to the Wi-Fi.
Equipment Required	<ul style="list-style-type: none"> - A cupboard.
Data Recorded	None.
Acceptance Criteria	A notify message is received.
Procedures	<ol style="list-style-type: none"> 1. Prepare a cupboard. 2. Place the prototype in a dark environment. 3. Block the LED using cupboard to make sure the LDR sensors use to detect the light intensity of LED does not irradiate by the LED. 4. Check E-mail.
Troubleshooting	<ul style="list-style-type: none"> - Check the internet connection.
Post-Test Activities	None

Table 3.7: Verification Plan 3

Internet of Things (IoT).

Plan ID	P4
Method	Black box testing
Requirement	The next streetlight become brighter when an object passes by the current streetlight.
Aim	To ensure the streetlights can exchange information among themselves.
Item Under Test	Microcontroller, motion sensors, LED
Precaution	<ul style="list-style-type: none"> - The testing should be done in dark environment. - Make sure the prototypes have been connected to the Wi-Fi.
Limitation	The prototypes must be connected to the Wi-Fi.
Equipment Required	<ul style="list-style-type: none"> - An object act as car or human.
Data Recorded	None.
Acceptance Criteria	The next streetlight become brighter when an object passes by the current streetlight.
Procedures	<ol style="list-style-type: none"> 1. Prepare an object act as car or human. 2. Prepare two prototypes. 3. Place the prototypes in a dark environment. 4. Make sure the distance between two prototypes at least 25cm. 5. Observe the LED light intensity of the prototypes from the dashboard. 6. Make the object pass by the PIR sensors of the first prototype. 7. Observe the LED light intensity of other prototypes from the dashboard.
Troubleshooting	<ul style="list-style-type: none"> - Check the internet connection.
Post-Test Activities	None

Table 3.8: Verification Plan 4

CHAPTER 4 – SYSTEM DESIGN

4.1 Design Specification

The proposed intelligent streetlight system aims to enhance energy efficiency and safety by consolidating data from various sensors. This data is then processed using sensor fusion techniques to assess both light intensity in the environment and traffic flow, allowing for automated adjustments to the streetlights. Key to this process is the multiplexer, which acts as a crucial converter by translating analogue data from the LDR sensor into a digital format. The microcontroller will utilize predefined fuzzy logic algorithms to make informed decisions based on this data. To further improve data accuracy and reliability, supplementary PIR and LDR sensors will also be deployed.

In addition to this, the microcontroller will establish cloud-based connectivity, enabling data exchange with other microcontrollers to provide a comprehensive environmental perspective. Once the sensor data is collected, the microcontroller will execute a sequence of steps, including fuzzification, processing via the Fuzzy Rule Base, and defuzzification to determine the optimal light intensity. Subsequently, the microcontroller will regulate the power supply to the LEDs to achieve the desired luminance level.

In the event of an error, the cloud system will promptly trigger an alarm and notify designated personnel via email. Moreover, the microcontroller will incorporate fault tolerance mechanisms which enables system to function even when a fault event occurred, to ensure system reliability and actively facilitate the sharing of sensed data, further strengthening fault tolerance control.

The responsible authority will possess the capability to oversee the system via a user-friendly dashboard interface and exercise manual control over streetlight functions in case of emergencies. Furthermore, the dashboard provides convenient access to all the data accumulated from the streetlights, enabling seamless downloads for analysis or utilization in machine learning applications.

4.2 System Block Diagram

Figure 4.1 illustrates the system block diagram, showcasing its operational flow. Initially, a power supply is essential to activate the system, providing the necessary energy to the microcontroller. Subsequently, data acquisition commences with the utilization of LDR sensors

and PIR sensors. The PIR sensors' output is directed to the microcontroller, while the LDR sensor's analogue data is routed through a MUX for conversion into digital format before reaching the microcontroller. Microcontroller shares the power supply to the MUX to enable it works. To facilitate seamless data transfer, the microcontroller synchronizes communication with MUX via the SCL and SDA channels. Concurrently, data from other streetlights is collected and sent to the microcontroller through cloud integration. The cloud processes the amassed data and conveys both the measured values and control commands back to the microcontroller, which regulates the LED accordingly. In case of an emergency signal, it is also relayed to the microcontroller via the cloud, prompting the LED to activate as needed.

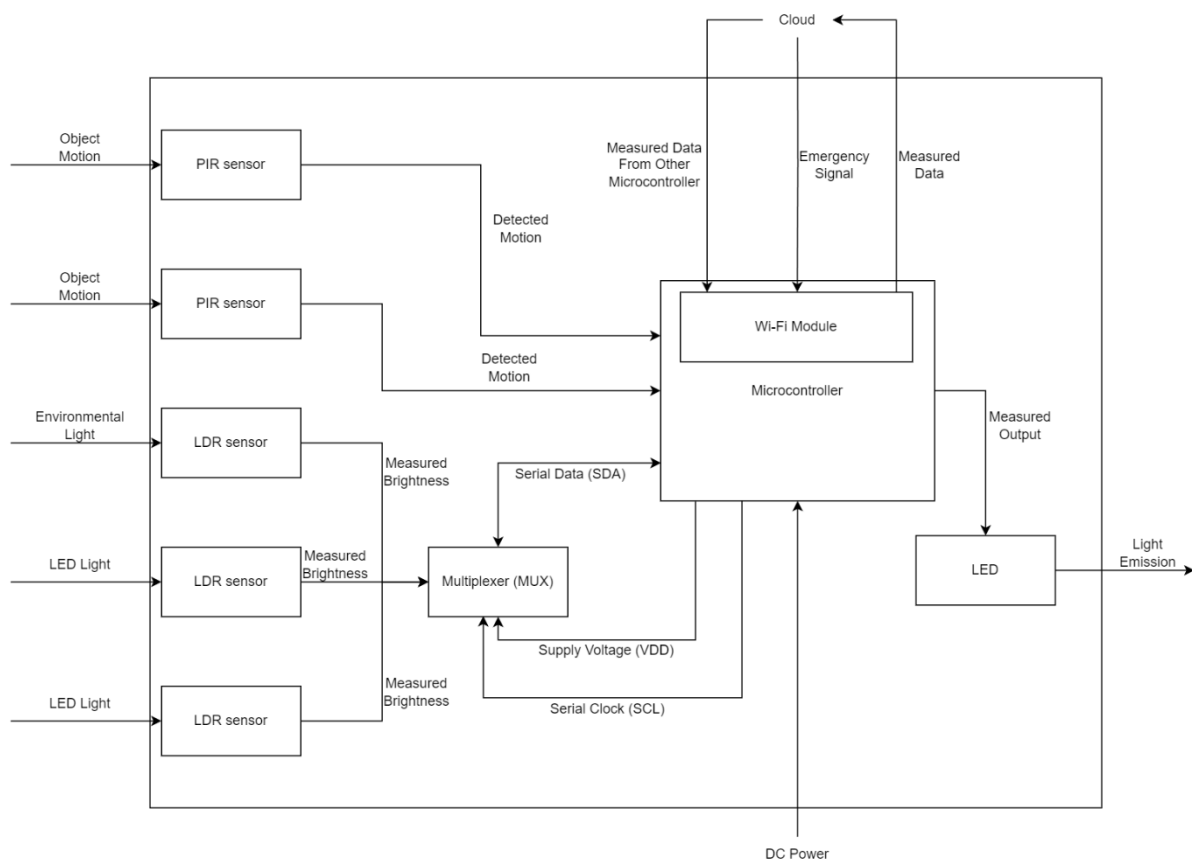


Figure 4.1: System Block Diagram

4.3 System Data Flow Diagram

Figure 4.2 illustrates the system's data flow via the cloud, providing users with real-time data monitoring and the capability to activate emergency lighting via user-friendly dashboards. When a user initiates the emergency lighting function, an emergency signal is promptly transmitted to the microcontroller through the cloud. The microcontrollers exchange measured

data among themselves using cloud integration. Simultaneously, the measured data is archived for record-keeping and also made accessible on the dashboard for ongoing real-time monitoring. In the event of an error or malfunction, the cloud is programmed to trigger an alarm, promptly notifying the responsible personnel via email.

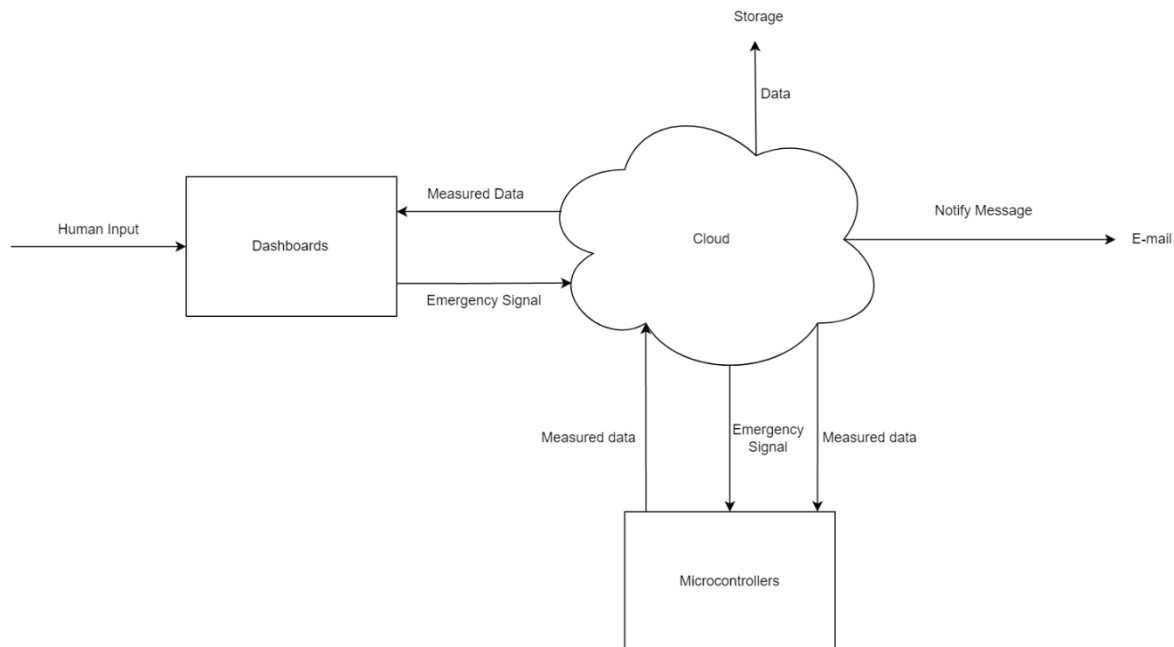


Figure 4.2: Data Flow Diagram

4.4 System Schematic Diagram

Figure 4.3 depicts the comprehensive wiring configuration of our project, encompassing the Arduino Nano 33 IoT microcontroller, ADS1115 multiplexer (MUX), LED, PIR sensors, LDR sensors, and a 10kOhm resistor, all intricately interconnected to ensure seamless functionality. Beginning with the LED component, its anode, symbolizing the positive pin, is thoughtfully linked to microcontroller pin D2. This strategic connection empowers the microcontroller to regulate the LED's power supply, thereby finely adjusting its luminosity. Simultaneously, the LED's cathode, denoting the negative pin, finds its grounding counterpart in the microcontroller's G pin.

Next, we attend to the PIR sensors' power requirements. The 'VCC' input pin of these sensors receives its vital energy supply from the microcontroller's 3.3V pin. It is important to note that certain PIR sensors may necessitate different voltage levels, necessitating a compatible sensor or modification to align with our project's requirements. The PIR sensors communicate their

readings through serial communication, so their 'OUT' pins are thoughtfully connected to microcontroller pins D5 and D7, equipped for seamless serial data exchange. The 'GND' pin of the sensors is appropriately linked to the microcontroller's 'G' pin for grounding.

To streamline the system's wiring complexity, optimize resource utilization, and enhance scalability, we incorporate the multiplexer to gather data from the LDR sensors. Firstly, the 'VDD' pin of the MUX taps into the microcontroller's 3.3V power source. The 'ADDR' and 'GND' pins of the multiplexer find their grounding connections in the microcontroller's 'G' pin. The 'SDA' pin, serving as the conduit for actual data transfer between devices, finds its home in microcontroller pin A4, which doubles as the SDA pin. Meanwhile, the 'SCL' pin, responsible for transmitting clock signals to synchronize data exchange, harmoniously connects with pin A5 of the microcontroller.

Lastly, to glean readings from the LDR sensors, the A0, A1, and A2 pins are interposed between each LDR sensor and a 10kOhm resistor. This ingenious design introduces a measure of control over the sensitivity of the LDR sensors to ambient light. The inclusion of the 10kOhm resistor serves to fine-tune the sensitivity, offering a degree of flexibility that would otherwise be constrained by the fixed sensitivity set during manufacturing.

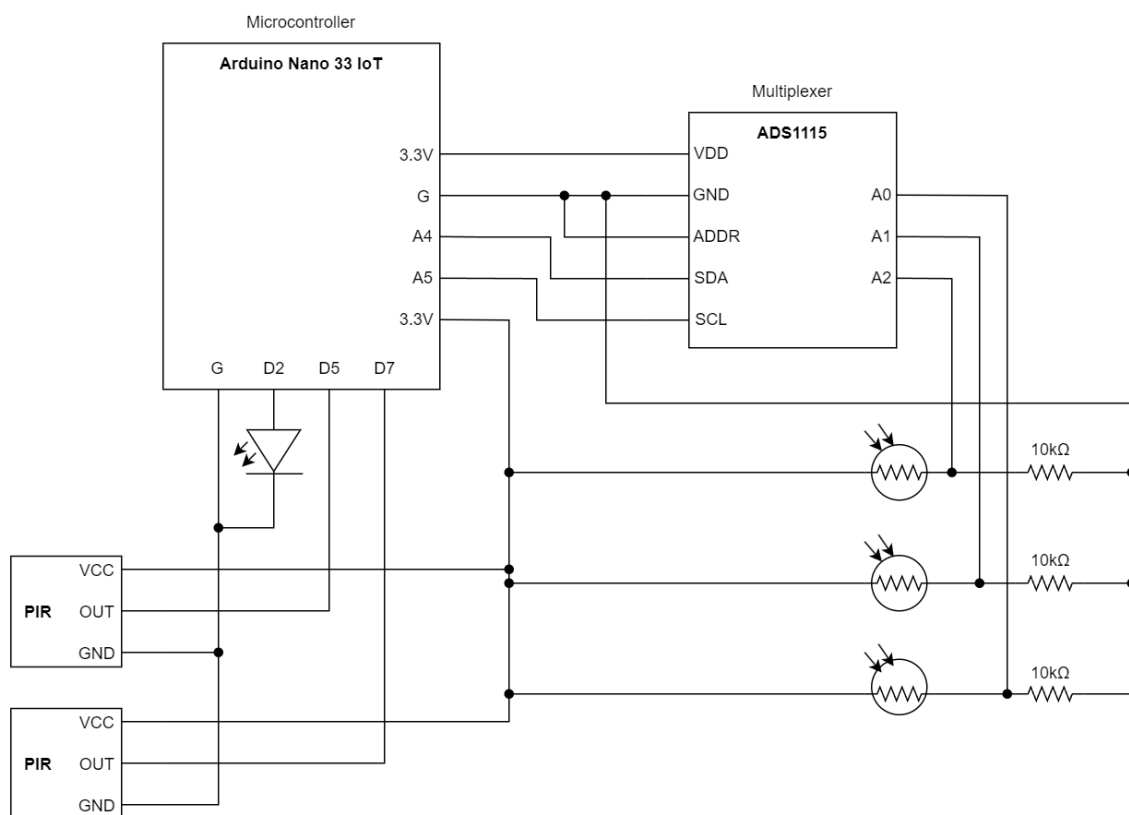


Figure 4.3: System Schematic Diagram

4.5 Fault-Tolerant Design

In Figure 4.4, we present a comprehensive diagram illustrating the fault-tolerant architecture of our system, which has been strategically organized into three layers.

The first layer, known as the control layer, serves as the initial hub of data interaction. Within this layer, sensors and the cloud collaboratively gather data from external sources, funnelling it towards the central microcontroller. The microcontroller, residing in between the first and second layer, then becomes the recipient of this incoming data.

Here, in the fault detection and resolution layer, the system's critical analysis takes place. The fault detection and resolution function diligently examine the incoming data to ascertain the presence of any faults or anomalies. If a fault is detected, the function swiftly initiates a process that involves the third layer, the human supervision layer. In this layer, a fault message is dispatched to the user interface, serving as a notification to responsible personnel about the occurrence of the fault.

Upon receiving this notification, human operators are promptly alerted to take action and address the fault. However, in instances where the fault detection and resolution function possess the capability to autonomously resolve the issue, it proceeds to provide a solution to the microcontroller. This solution guides the microcontroller in processing the incoming data, ensuring the system's continued functionality.

The implementation of this fault-tolerant architecture within our system is instrumental in achieving robust fault tolerance. It not only identifies, and addresses faults effectively but also integrates human supervision when necessary, culminating in an enhanced level of system reliability and responsiveness. This layered approach fortifies the system's ability to adapt and respond to a multitude of potential challenges.

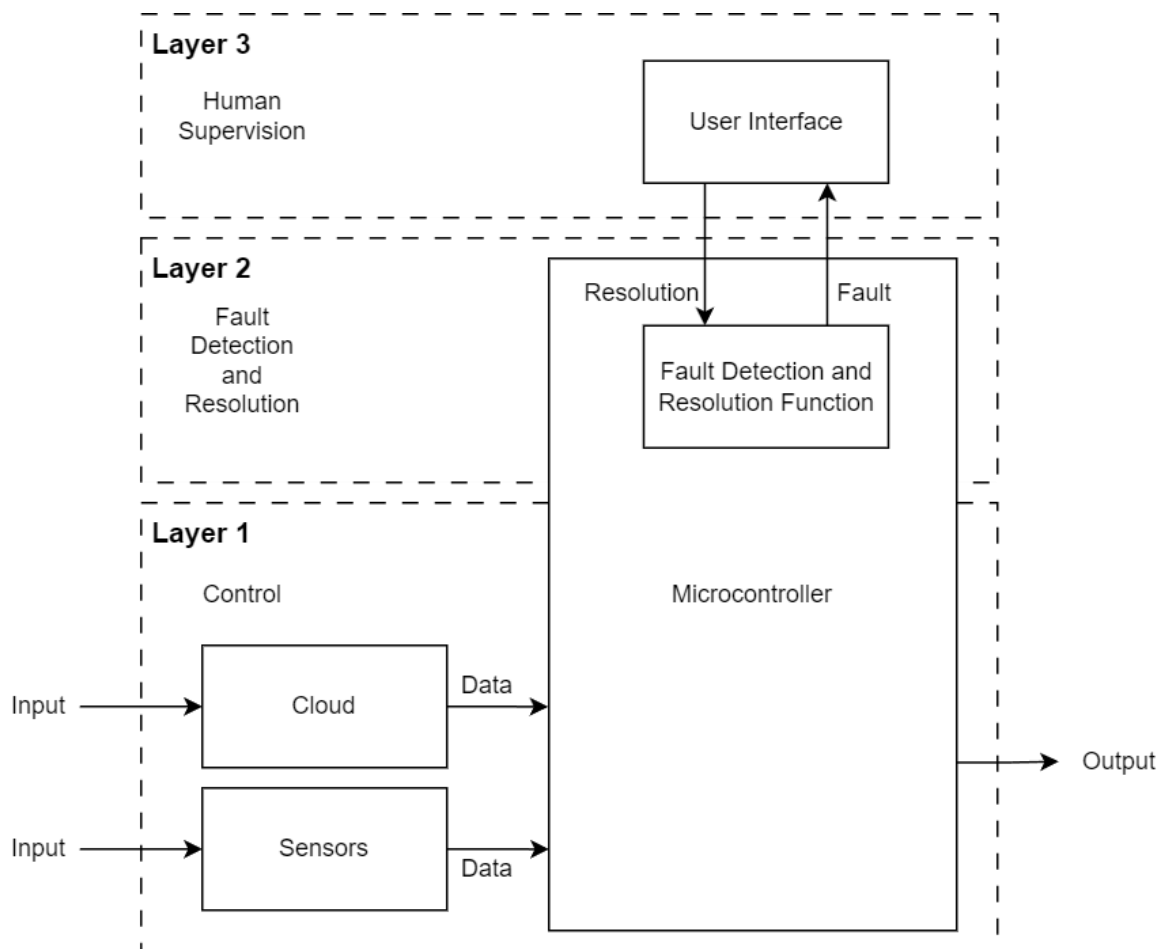


Figure 4.4: Fault-Tolerant Architecture Diagram

In details, to establish robust fault tolerance within this system, redundancy is strategically integrated into the hardware components, notably employing additional PIR (Passive Infrared) sensors and LDR (Light Dependent Resistor) sensors to gather redundant data streams. This redundancy in hardware serves a vital purpose, enabling the system to cross-reference and validate data. By accumulating data from multiple sources, the system employs a multi-faceted approach to validate the accuracy and reliability of data acquired from primary sources. This redundancy-driven approach enhances the system's resilience and guarantees the integrity of its operations, even when confronted with unforeseen challenges or sensor anomalies. In situations where discrepancies or conflicts arise within the data sets, a failover mechanism comes into play as a fail-safe measure, ensuring that the system can derive a precise and well-informed decision by relying on the most reliable and consistent data source available.

A failover mechanism is a pivotal component within fault-tolerant systems and high-availability architectures. It serves the essential function of seamlessly and automatically transitioning operations from a primary component, system, or server to an alternative

secondary or backup component when the primary one encounters a failure or becomes inaccessible. The paramount objective of a failover mechanism is to minimize service disruptions and uphold system availability, even in the face of unforeseen failures.

In our specific system, the failover mechanism has been meticulously engineered to prioritize safety and reliability. Its decision-making process hinges on the evaluation of the most critical data sources. Within this system, two distinct types of failover mechanisms have been thoughtfully devised to address different fault scenarios, bolstering the system's resilience.

The first type of failover mechanism pertains to the crucial connection between the streetlight and the cloud. Some data, such as environmental brightness information, relies on real-time data sharing across the cloud. To safeguard against potential faults like internet disconnections, a meticulous mechanism has been devised. The microcontroller continuously monitors the cloud connection status by scrutinizing changes in cloud variable data. As cloud variable data inherently evolves over time, tracking these variations allows the system to detect instances when the streetlight becomes disconnected from the internet. In such cases, the system intelligently shifts its focus away from the compromised data, instead utilizing alternative data sources or even considering only the data from its sensor. This adaptive approach is applicable both when the master device experiences disconnection or when an individual slave device faces connectivity issues.

The system's second failover mechanism addresses the potential challenge of data conflicts that may arise when gathering information from multiple sensing sources. In this system, it's essential to collect data from at least two distinct sources to make informed decisions. However, there are instances where these data sources might provide conflicting or contradictory information. To effectively manage and resolve such conflicts, a carefully engineered mechanism has been devised.

This mechanism operates by intelligently prioritizing the most critical data source when conflicts occur. It does so by evaluating the reliability, accuracy, and relevance of each data source in real-time. By assigning a higher level of importance to specific data streams during conflicts, the system makes calculated decisions that prioritize road user safety and enhance the overall reliability of its operations.

This meticulous design strategy exemplifies the system's unwavering commitment to maintaining safety and consistency, even when confronted with varying data inputs. It ensures that the system remains steadfast in its mission to deliver dependable and high-performance

results, regardless of the complexities that may arise from diverse data sources. Ultimately, this approach strengthens the system's reliability and underpins its ability to make informed, safety-centric decisions, further enhancing its overall dependability and operational excellence.

Other than that, the system will incorporate a comprehensive monitoring and alerting functionality to enhance its overall reliability and fault tolerance. This includes the implementation of dashboards to continuously monitor the status of hardware components and connections. Additionally, a trigger function will be integrated to promptly notify responsible personnel in the event of a fault or issue.

Monitoring functions play a pivotal role in empowering system supervision. They provide real-time insights into the status of hardware components and connectivity, enabling administrators to proactively detect anomalies or potential problems. By continuously assessing the system's health and performance, monitoring functions contribute to maintaining a proactive stance in ensuring seamless operations.

Complementing this, the alerting function serves a critical purpose when the system encounters issues that it cannot autonomously resolve. For instance, consider the scenario where an LED within a streetlight burns out. In such cases, the system itself may not have the capability to restore the LED's functionality, especially if a backup LED is not available, which is often the case. It is during these situations that the alerting function comes into play within this fault-tolerant mechanism. By promptly notifying external personnel or administrators about the issue, the system ensures timely intervention and resolution. This external input becomes instrumental in addressing complex or non-self-recoverable problems, thus minimizing downtime and service disruptions.

In essence, the integration of monitoring and alerting functions not only bolsters the system's fault tolerance but also elevates its robustness and reliability. It transforms the system into a proactive, adaptive, and responsive entity, capable of addressing challenges swiftly and effectively, ultimately enhancing its dependability and performance.

4.6 User Interface Design (UI Design)

The system is designed to create a dashboard that enables users to perform real-time monitoring and control. This functionality allows users to access the dashboard using two types of devices: personal computers (PCs) and smartphones. Consequently, two distinct user interface (UI)

designs are required to accommodate both device types. Figure 4.5 and Figure 4.6 below illustrates the UI design of the dashboard.

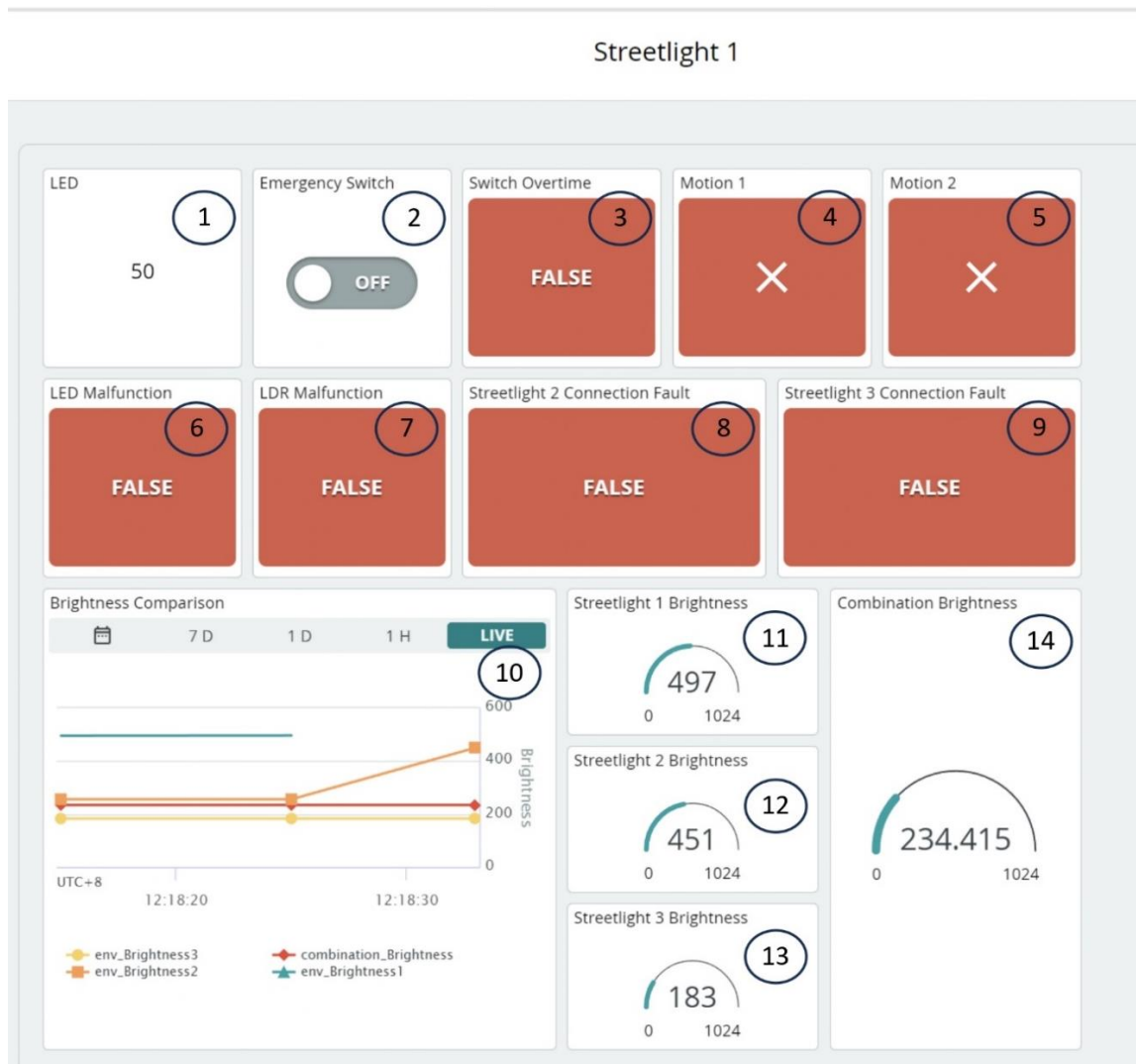


Figure 4.5: Dashboard's UI (PC view)

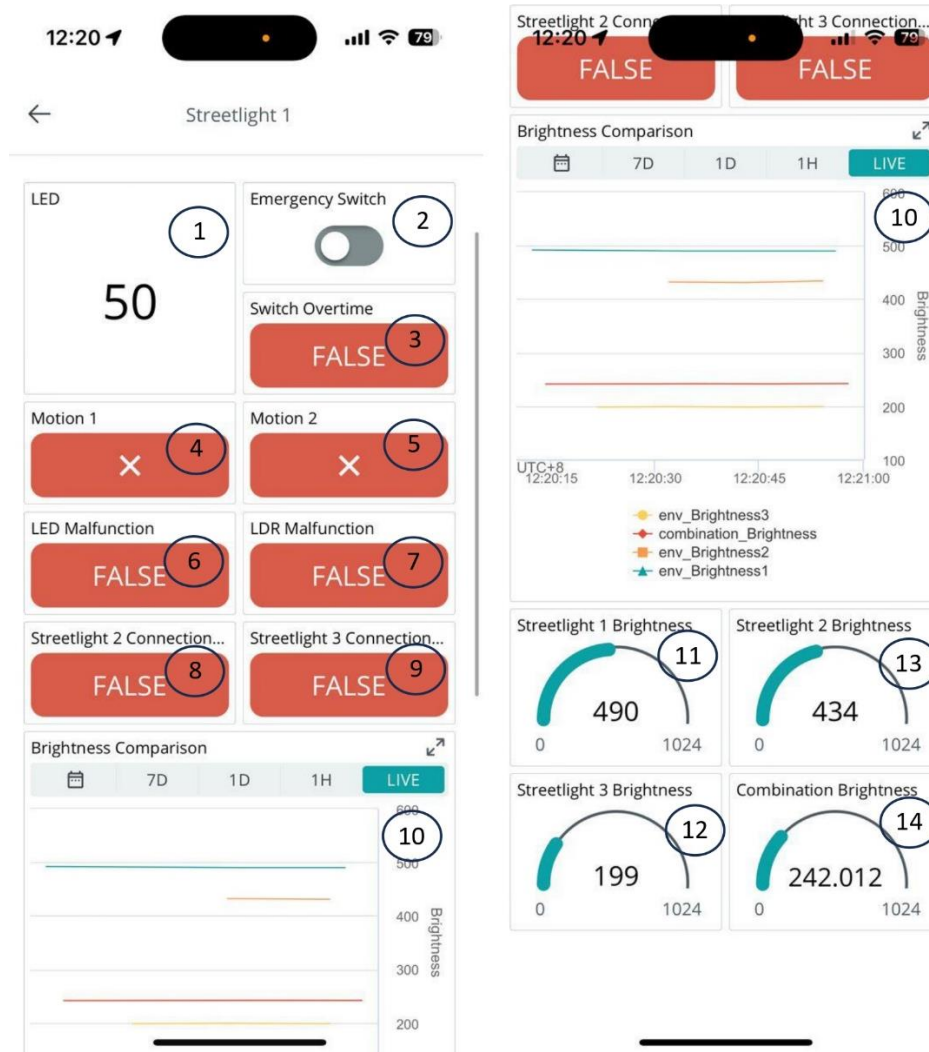


Figure 4.6: Dashboard's UI (Smartphone view)

Figure 4.5 shows the PC view of the dashboard's UI while the Figure 4.6 shows the dashboard's UI from smartphone view, the component of the dashboards include:

No.	Component Name	Description
1	LED	Brightness of LED (min: 0, max 255)
2	Emergency switch	Switch that allows the responsible personnel to make the streetlight give maximum light intensity when needed.
3	Switch Overtime	Indicator that indicates whether the emergency switch has been turned on for a long time.
4	Motion 1	Indicator that indicates whether there is any motion detected by the PIR sensor 1 of the streetlight.

5	Motion 2	Indicator that indicates whether there is any motion detected by the PIR sensor 2 of the streetlight.
6	LED Malfunction	Indicator that indicates whether the LED is malfunctioning.
7	LDR Malfunction	Indicator that indicates whether the LDR sensor is malfunctioning.
8	Streetlight 2 Connection Fault	Indicator that indicates whether the streetlight 2 is disconnected.
9	Streetlight 3 Connection Fault	Indicator that indicates whether the streetlight 3 is disconnected.
10	Brightness comparison	Light Intensity Graph displays historical and current data for environment light intensity collected by each streetlight and the processed value. This graph allows users to track changes in light intensity over time, aiding in monitoring and managing streetlight performance.
11	Streetlight 1 Brightness	Indicator that indicates the environment light intensity collected by streetlight 1.
12	Streetlight 2 Brightness	Indicator that indicates the environment light intensity collected by streetlight 2.
13	Streetlight 3 Brightness	Indicator that indicates the environment light intensity collected by streetlight 3.
14	Combination Brightness	Indicator that indicates the environment light intensity after processed by the microcontroller.

Table 4.1: Dashboard Component description

CHAPTER 5 – SYSTEM IMPLEMENTATION

5.1 Component and Technology Involved

5.1.1 Hardware

Microcontroller

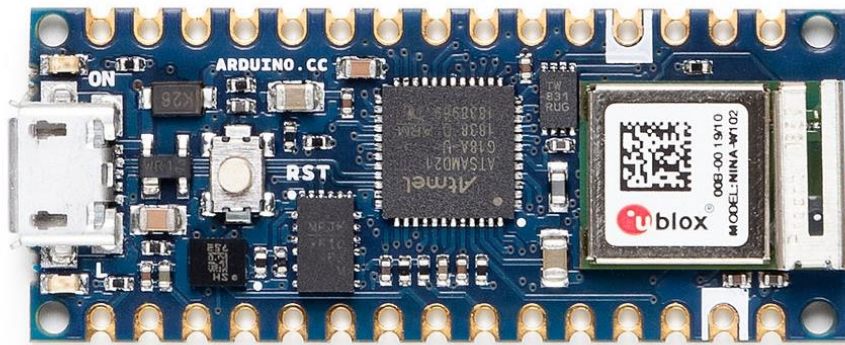


Figure 5.1: Arduino Nano 33 IoT

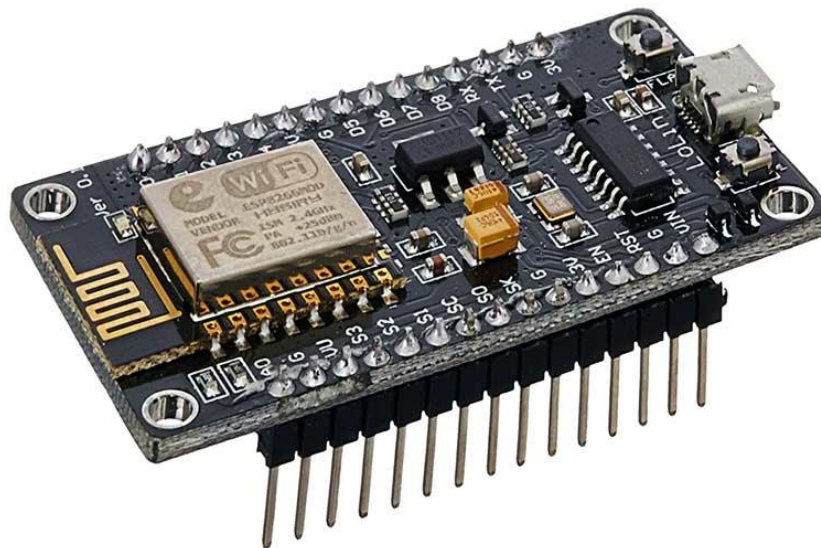


Figure 5.2: NodeMCU ESP8266

Various microcontroller boards are accessible in the market, yet the prime selections for this system encompass the Arduino Nano 33 IoT and the NodeMCU ESP8266. Both microcontrollers stand out as optimal choices for realizing an Internet of Things (IoT)-enabled system. They come equipped with built-in Wi-Fi modules, facilitating seamless internet connectivity. Furthermore, these microcontroller boards are compatible with the Arduino IoT cloud, which is the software utilized to orchestrate the implementation of this system, ensuring a harmonious and efficient synergy between hardware and software components.

Passive Infrared Sensor (PIR)



Figure 5.3: Passive Infrared Sensor (PIR)

A passive infrared sensor (PIR) is an invaluable electronic device that detects infrared (IR) radiation emitted by objects within its field of view, commonly employed in security alarms and automated lighting systems. This sensor captures incoming infrared signals bounced off obstacles and wirelessly relays them to the controller. The controller, equipped with decision-making process, then determines whether to activate or deactivate the streetlight or adjust its brightness, all contingent on the received signal. This inherent automation ensures the streetlight illuminates pre-emptively, offering a practical solution for areas with minimal nighttime traffic. It's worth noting that some PIR sensors may require varied voltage levels, necessitating the use of compatible sensors or project-specific modifications, particularly crucial in our setup where the microcontroller's 3.3V power supply is the standard.

Light Dependent Resistor (LDR)

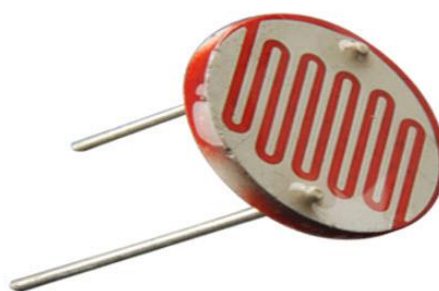


Figure 5.4: Light Dependent Resistor (LDR)

A light dependent resistor (LDR), also known as a photoresistor, is a type of variable resistor that changes its resistance according to the amount of light falling on it. This phenomenon, called photo conductivity, is utilized in this system in two ways. Firstly, the LDR senses the brightness of the environment, with its resistance becoming high in the dark and low in bright conditions. This allows it to automatically switch the lights ON/OFF during night and daytime,

respectively. Secondly, it serves as a fault-detection sensor, with LDR values below a certain threshold indicating a fault. To prevent other light sources, such as streetlights, from interfering with the LDR's readings, it must be positioned carefully. A time delay circuit is necessary to avoid short periods of darkness or brightness from switching the lights on or off, resulting in significant energy savings and preventing the wastage of energy by illuminating unwanted areas. In this project the LDR is comprising with the 10kOhm resistor, to have a more sensitivity to the light.

Multiplexer (MUX)

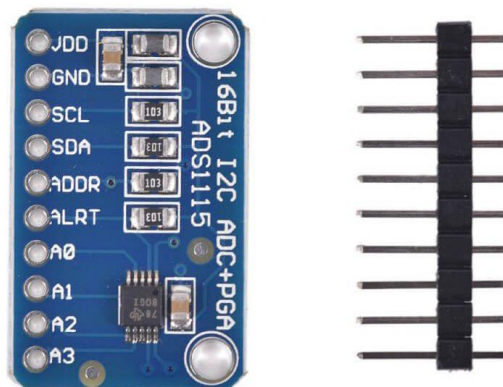


Figure 5.5: ADS1115 Module

The ADS1115 is a 16-bit analogue-to-digital converter (ADC) integrated circuit commonly used in electronic applications to convert analogue voltage signals into digital data. It offers high precision and accuracy, making it well-suited for tasks that require precise analogue measurements. The ADS1115 can handle up to four analogue input channels and communicates with a microcontroller through I2C communication. Its features include programmable gain amplification, a low-drift internal voltage reference, and the ability to operate in single-shot or continuous conversion modes. This versatile ADC plays a pivotal role in converting the analogue data acquired from the LDR sensor into digital format, enabling further processing and analysis in this system.

Light-emitting Diode (LED)



Figure 5.6: Light-emitting Diode (LED)

LED, or light-emitting diode, is a type of diode that emits visible light when activated. LED lights have many advantages such as low energy consumption, long lifespan, and instant full brightness. In this system, an LED circuit is connected to the Arduino board and is controlled by it based on the readings from the PIR and LDR sensors. When the LDR detects a low brightness, it sends a signal to the Arduino which in turn activates the LED light. LED lights are becoming increasingly popular due to their energy efficiency and other benefits compared to traditional lighting options.

Resistor

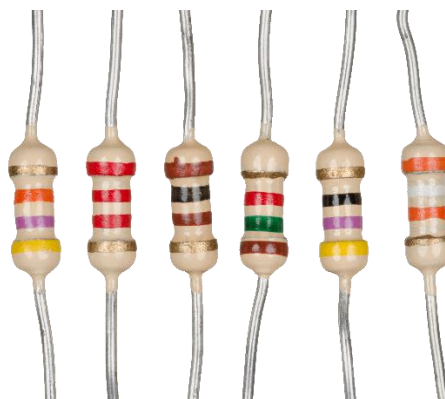


Figure 5.7: Resistor

A resistor is a type of electronic component that is not powered by electricity, and it is commonly paired with other electronic components like LDR sensors to regulate or restrict the

flow of electrons. This is demonstrated in Figure 5.7. Additionally, a resistor can also be used to generate a certain voltage for an active device, like a transistor.

Other Components (Jumper Wire, Breadboard)

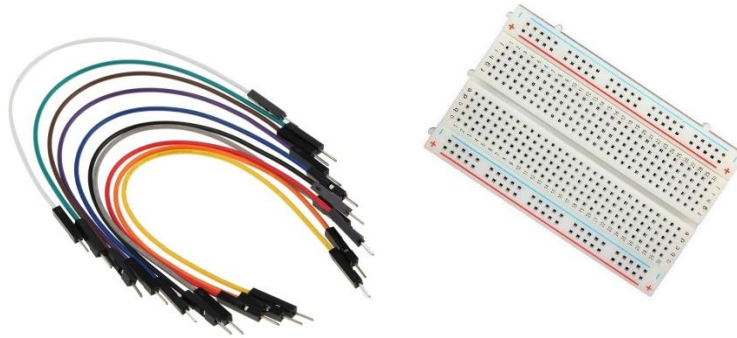


Figure 5.8: Jumper Wires and Breadboard

Jumper wires are wires with connector pins at both ends, and they will be extensively utilized in the proposed project for linking the Arduino Uno with the breadboard and other components. Male-to-male, male-to-female, and female-to-female are the three types of jumper wires. The breadboard is a component that provides speedy electrical connections between components such as LED, PIR, LDR, microcontroller, resistors, and multiplexer.

5.1.2 Software

Arduino IoT Cloud

Arduino IoT Cloud is a cloud-based platform developed by Arduino for building Internet of Things (IoT) applications. It provides an easy and integrated way to connect Arduino devices and any other third-party devices to the cloud, allowing users to remotely monitor and control their IoT projects. Arduino IoT Cloud offers features such as device management, data storage, and visualization tools, making it simpler for makers, developers, and hobbyists to create, manage, and interact with their IoT devices through a user-friendly web interface. It's designed to simplify the process of developing and deploying IoT projects without the need for extensive coding or complex infrastructure setup.

Arduino Web Editor

The Arduino Web Editor is a web-based integrated development environment (IDE) offered by Arduino. It allows users to write, compile, and upload code to microcontroller boards

directly from a web browser without the need to install software on their computer. This cloud-based IDE simplifies the process of programming Arduino microcontrollers, making it accessible to beginners and experienced users alike. It also provides convenient features like code sharing, collaborative editing, and easy access to libraries, making it a versatile tool for the Arduino community.

Arduino Cloud Trigger

Arduino Cloud Trigger is a powerful capability within the Arduino Cloud ecosystem, empowering users to establish customized triggers based on specified conditions that, when met, trigger predefined actions. This versatile tool supports both Boolean and String variables for creating condition logic and facilitates email notifications as part of the responsive actions. In our system, this feature plays a pivotal role in crafting robust fault detection and notification mechanisms, enabling seamless automated responses, such as sending email alerts to the relevant personnel when system faults occur, thus ensuring timely intervention and problem resolution.

5.1.3 Circuit

Before the implementation of fuzzy logic, the prototype system in Figure 5.9 must be constructed first. A system prototype is implemented, and then another 2 prototype is replicated based on the first prototype. Each prototype contains 3 LDR sensors, 2 PIR sensors, 1 LED, and three 10k Ω resistors. The LDR sensors measure the ambient light levels, while the PIR sensors detect the movement of objects. The resistors are used to regulate the current flowing through the circuit and control the sensitivity of the LDR sensors. The LED is used as the streetlight, and the output of the fuzzy logic system will determine its brightness. By setting up this circuit, it becomes possible to apply fuzzy logic and begin creating the system that can intelligently control the streetlights.

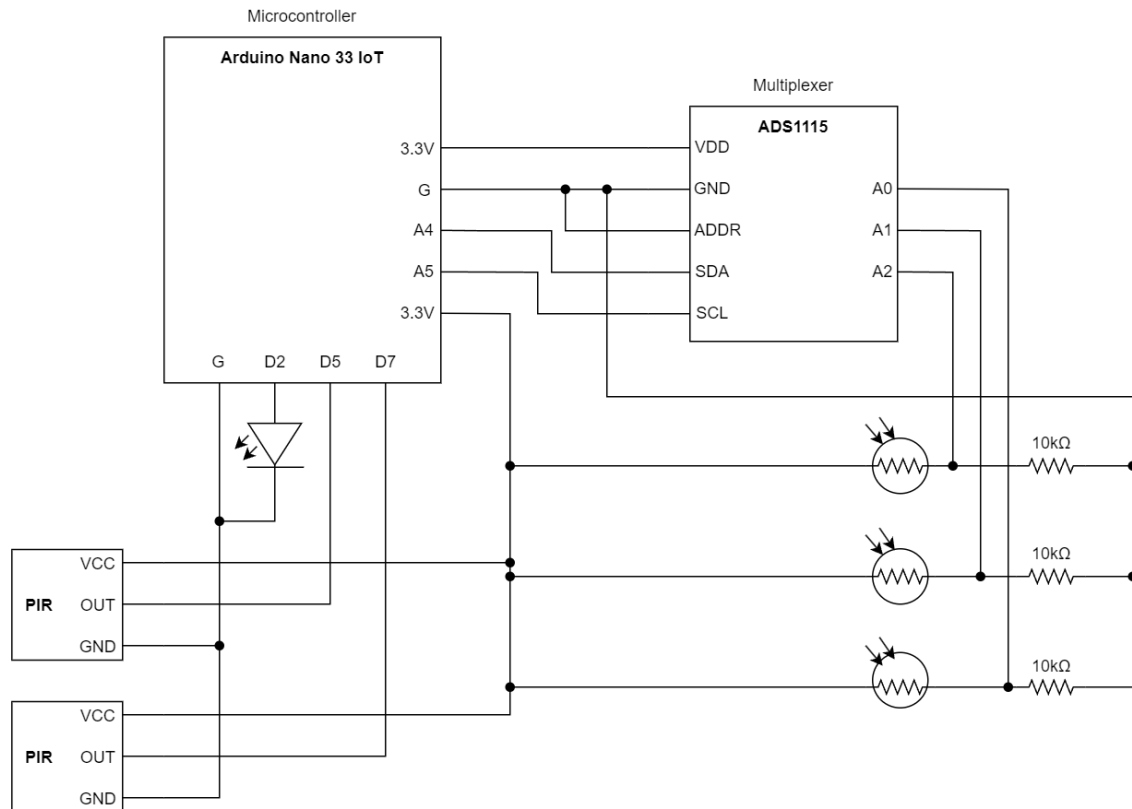


Figure 5.9: System Schematic Diagram

5.2 Setup

5.2.1 Setup Arduino IoT Cloud

Prior to implementing the system, it is imperative to set up the Arduino Cloud to harness its services and seamlessly integrate them into the system. This entails several essential steps. Firstly, one must create an Arduino account, which serves as the gateway to accessing the platform's services. Once this account is established, the next crucial step is to register the microcontroller as a device in the Arduino IoT Cloud, enabling it to communicate with the platform effectively. To ensure the microcontroller's connectivity to the internet, it is imperative to define the Wi-Fi credentials.

The cornerstone of this integration process is the creation of Cloud Variables within the Arduino IoT Cloud. These variables serve a dual purpose by acting as both storage units for specific data and facilitating data synchronization between the local system and the cloud. The data stored within these Cloud Variables resides in the Arduino Cloud's storage and can be conveniently retrieved through the platform's dashboard.

With the Cloud Variables in place, development can commence using the Arduino Web Editor. This integrated development environment facilitates coding, and upon completion, it uploads the code seamlessly into the microcontroller.

Following code implementation, the next step involves crafting a customized dashboard for a specific streetlight within the system. Each component of this dashboard is intricately linked to the previously created Cloud Variables. This interconnectedness empowers users to monitor data changes and access real-time information effortlessly.

Lastly, the system's integrity is fortified through the creation of Cloud Triggers. These cloud-based alarms are designed to notify designated individuals when specific conditions are met. In this system, Cloud Triggers are instrumental in detecting abnormalities and promptly alerting users via email. Users can customize the content of these email notifications using the Cloud Trigger cloud apps.

For more comprehensive instructions and additional details, please consult the "[Appendix: Steps to Setup Arduino Clouds](#)"

5.3 Fault Tolerance Policy Diagram

To ensure the system's reliability in the face of potential faults and to optimize the effective utilization of fuzzy rules for providing appropriate street lighting, it's imperative to establish a comprehensive policy centered on system safety. This policy should encompass strategies for handling sensor malfunctions and other potential faults, as well as clear guidelines for interpreting the data gathered by these sensors. Moreover, the policy should underscore the importance of designing the system with redundancy and fault tolerance as primary considerations, thereby enabling it to continue functioning even when one or more system components encounter issues.

By giving paramount importance to the safety of the system, this policy serves a dual purpose: it ensures that street users receive the necessary illumination for their safety and minimizes the risk of accidents or other incidents.

Let's delve into the policy further. Initially, it's essential to address data conflicts that may arise among the sensors responsible for assessing the current environmental conditions. As depicted in Figures 5.10 and 5.11, the policy diagram outlines the sensor's role in this process. When the sensors collect data regarding the current environmental conditions, they transmit this information to the microcontroller. The microcontroller then applies a predetermined policy to

identify the most critical data, such as a dark environment, while disregarding less pertinent data. Subsequently, it processes the selected critical data accordingly. In cases where no data conflicts exist, the information smoothly proceeds to the next stage for further processing, ultimately culminating in the provision of an output.

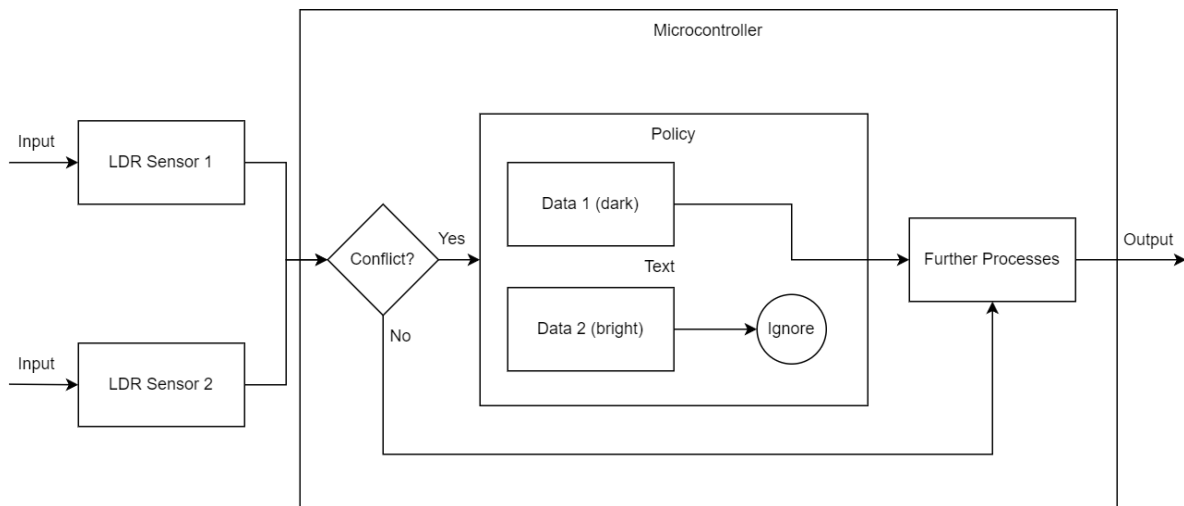


Figure 5.10: Policy diagram for LDR Sensor

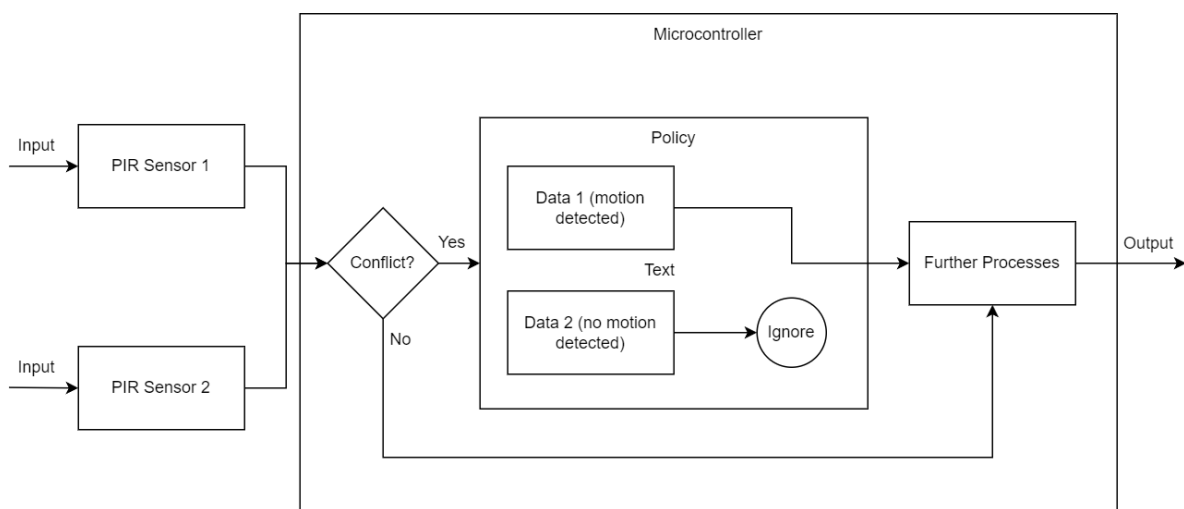


Figure 5.11: Policy diagram for PIR

In Figure 5.12, we can observe the policy framework devised for the microcontroller, aimed at making a crucial decision regarding the source of data: whether to use data from the cloud or rely on data from its own local sensors. This pivotal decision-making process plays a key role in ensuring the system's effectiveness.

The flow of data in this scenario involves information collected from other streetlights passing through the cloud before reaching the microcontroller. Concurrently, the microcontroller is actively collecting data from its own local sensors. When the microcontroller receives data from the cloud, it initiates a series of checks. First and foremost, it assesses whether the cloud

data has become outdated or "expired," which typically occurs when the data has remained static for an extended period. As explained in previous chapters, cloud data is inherently dynamic, evolving over time to reflect real-time conditions. By monitoring these changes, the system can identify instances when a streetlight has lost its internet connection.

If the microcontroller determines that the cloud data is indeed expired, it opts to utilize the locally sensed data gathered by its own sensors. Conversely, if the cloud data remains current and reliable, the microcontroller selects it for use in decision-making processes.

This policy-driven approach ensures that the system makes informed choices regarding data sources, prioritizing real-time and reliable information when available, while seamlessly transitioning to local sensor data when the cloud data becomes outdated. In doing so, the system maintains its responsiveness and adaptability, ultimately enhancing its performance and reliability in providing efficient streetlight management.

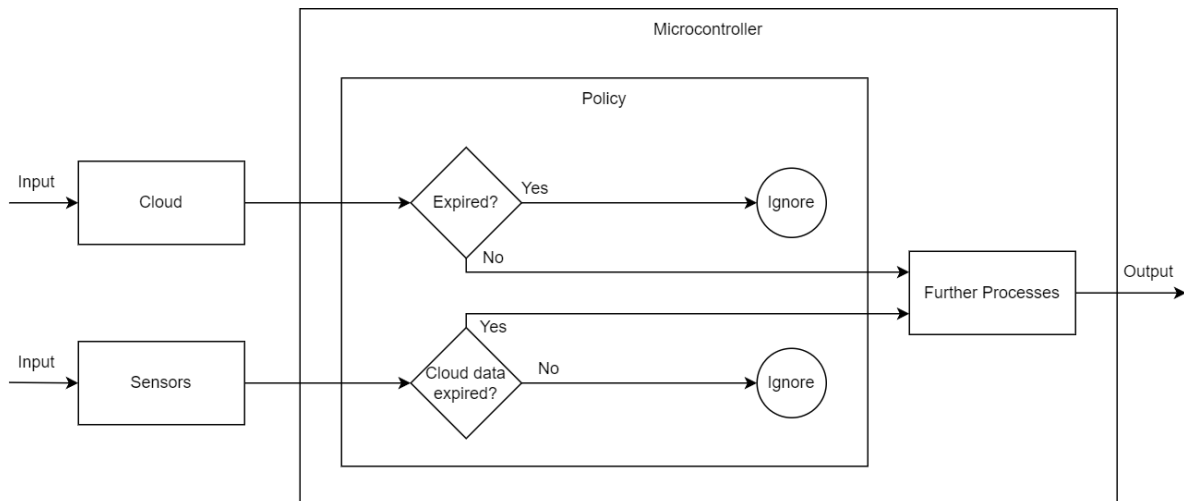


Figure 5.12: Policy diagram for Cloud data

5.4 Fuzzy Logic Algorithm

Fuzzy logic algorithm is a method of reasoning that resembles human reasoning in its ability to handle uncertainty and imprecision. It is used in many applications where decision-making processes require flexibility and tolerance for ambiguity, such as in artificial intelligence, robotics, and control systems.

Fuzzy logic algorithms use sets of rules that assign membership functions to input and output variables, allowing for the representation of vague and ambiguous concepts. The rules are expressed in terms of linguistic variables, such as "high," "low," "medium," and "very high," which are then mapped to numerical values using membership functions.

The output of a fuzzy logic algorithm is a value that represents the degree of membership of a given input to a particular category. The degree of membership is calculated using fuzzy sets and logical operations such as AND, OR, and NOT. Fuzzy logic algorithms can be used to perform tasks such as classification, prediction, optimization, and control.

One of the main advantages of fuzzy logic algorithms is their ability to handle uncertainty and incomplete information, which is often encountered in real-world applications. They can also be used to model nonlinear and complex systems that are difficult to represent using traditional mathematical methods.

Overall, fuzzy logic algorithms provide a powerful tool for handling uncertainty and imprecision, and they are widely used in a variety of applications where flexibility and adaptability are required.

5.4.1 Fuzzy Rules

In order to complete the fuzzy logic algorithm, fuzzy rules are expected to be implemented in this intelligent streetlight system to process the input of the data and provide an accurate output for the microcontroller to make decision on the power supply to the LED.

Below is the list of abbreviations is used in the fuzzy rules:

LDR-e (self) – Light Dependent Resistors sensor use to collect environmental brightness build on the streetlight.

LDR-e (others) – Light Dependent Resistors sensor use to collect environmental brightness from the other streetlights.

PIR1 – Passive Infrared Sensor 1

PIR2 – Passive Infrared Sensor 2

PRE – Indicator use to indicate whether the streetlight needs to perform advance lighting.

RULE 1:

IF LDR-e (self) = bright and LDR-e (others) = dark THEN notify true.

RULE 2:

IF LDR- e (self) = dark and LDR-e (others) = bright THEN notify true.

No.	LDR-e (self)	LDR-e (others)	Notify
-----	--------------	----------------	--------

1	bright	bright	false
2	bright	dark	true
3	dark	bright	true
4	dark	dark	false

Table 5.1: Decision Table summarise RULE 1 and RULE 2

According to Table 5.1, if there is a disparity in the data collected by the LDR sensors used to measure the environmental brightness of the streetlight and other streetlights, the system will send a notify message.

RULE 3:

IF PIR1 = high and PIR2 = low THEN notify true.

RULE 4:

IF PIR1 = low and PIR2 = high THEN notify true.

No.	PIR1	PIR2	Notify
1	high	high	false
2	high	low	true
3	low	high	true
4	low	low	false

Table 5.2: Decision Table summarise RULE 3 and RULE 4

Table 5.2 shows that if there is a conflict in the data collected by the PIR sensors, the system will send a notify message.

RULE 5:

IF LDR-1 1 = bright and LDR-1 2 = dark THEN notify true.

RULE 6:

IF LDR-1 1 = bright and LDR-1 2 = dark THEN notify true.

No.	LDR-1 1	LDR-1 2	Notify
-----	---------	---------	--------

1	bright	bright	false
2	bright	dark	true
3	dark	bright	true
4	dark	dark	false

Table 5.3: Decision Table summarise RULE 5 and RULE 6

Table 5.3 indicates that if there is a discrepancy in the data collected by the LDR sensors used to measure the LED brightness of the streetlight, the system will send a notify message.

RULE 7:

IF LDR – e (self) = dark or LDR – e (others) = dark THEN environment dark.

RULE 8:

IF LDR – e (self) = average or LDR – e (others) = average THEN environment average.

RULE 9:

IF LDR – e (self) = bright and LDR – e (others) = bright THEN environment bright.

No.	LDR-e (self)	LDR-e (others)	environment
1	bright	bright	bright
2	bright	average	average
3	bright	dark	dark
4	average	bright	average
5	average	average	average
6	average	dark	dark
7	dark	bright	dark
8	dark	average	dark
9	dark	dark	dark

Table 5.4: Decision Table summarise RULE 7, RULE 8, and RULE 9

Table 5.4 shows that if any of the LDR sensors used to measure the environmental brightness record as dark, the system will classify the environment as dark. If no sensor used to measure the environmental brightness record as dark and there is sensor that record as average, then the system will classify the environment as average. For the system to classify the environment as

bright, all of the LDR sensors used to measure the environmental brightness must record as bright.

RULE 10:

IF PIR1 = high or PIR2 = high or PRE = high THEN motion detected true.

RULE 11:

IF PIR1 = low and PIR2 = low and PRE = low THEN motion detected false.

No.	PIR1	PIR2	PRE	motion detected
1	high	high	high	true
2	high	high	low	true
3	high	low	high	true
4	high	low	low	true
5	low	high	high	true
6	low	high	low	true
7	low	low	high	true
8	low	low	low	false

Table 5.5: Decision Table summarise RULE 9 and RULE 10

Table 5.5 indicates that if any of the PIR sensors or PRE record a high value, the system will classify motion as detected. In order for the system to classify motion as not detected, all of the PIR sensors must record low values.

RULE 12:

IF environment = dark and motion detected = true THEN power supply to LED is high.

RULE 13:

IF environment = dark and motion detected = false THEN power supply to LED is medium.

RULE 14:

IF environment = bright THEN power supply to LED is low.

RULE 15:

IF environment = average and motion detected = true THEN power supply to LED is medium.

RULE 16:

IF environment = average and motion detected = false THEN power supply to LED is low.

No.	environment	motion detected	power supply to LED
1	dark	true	high
2	dark	false	medium
3	average	true	medium
4	average	false	low
5	bright	true	low
6	bright	false	low

Table 5.6: Decision Table summarise RULE 12, RULE 13, RULE 14, RULE 15, and RULE

16

Table 5.6 shows that when the environment is classified as dark and motion is detected, the system will supply high power to the LED in order to increase its brightness. When the environment is dark, but no motion is detected, the system will supply a medium power to the LED to maintain an average brightness level. When the environment brightness is average, but no motion is detected, the system will supply a low power to the LED. However, if the motion is detected in the average brightness environment, the output to the LED will become medium to maintain an average brightness level. Finally, when the environment is classified as bright, the system will supply a low power to the LED.

5.5 Implementation Issue and Challenges

Implementing an intelligent streetlight system using microcontrollers presents three formidable challenges. Firstly, the foremost challenge pertains to the intricate nature of fuzzy logic algorithms. Fuzzy logic, a mathematical approach used for modelling uncertainty and imprecision, plays a pivotal role in such systems. Its application demands a profound grasp of mathematics, programming, and electronics, making it a formidable task for many researchers and technicians. Regrettably, resources guiding individuals on how to integrate fuzzy logic into microcontrollers remain scarce, comprising a significant barrier.

The second challenge revolves around the intricacy of the system's circuitry. A robust fuzzy logic system necessitates inputs, rules, and outputs, which often involve an array of sensors, actuators, and various components. This complexity results in a web of wiring and connections, substantially increasing the potential for circuit errors. Troubleshooting such a labyrinthine circuit can be a formidable endeavour, especially when the issue lies in incorrect wiring.

The third and no less significant challenge is hardware troubleshooting, an especially pertinent concern in embedded system projects. These projects frequently interface with multiple devices and hardware components through a complex network of wires and connections. Even a single wire's damage or disconnection can lead to erroneous outputs. Identifying the precise fault within this intricate web of connections is a Herculean task.

In summary, implementing a fuzzy logic algorithm in microcontrollers demands a high level of technical acumen and meticulous attention to circuit design and wiring. However, the existence of numerous online resources and supportive communities offers hope to individuals seeking to surmount these hurdles and successfully integrate fuzzy logic algorithms into microcontroller-based systems.

CHAPTER 6 – SYSTEM DELIVERABLE AND TESTING

6.1 System Deliverable

6.1.1 Fault-Tolerant Mechanism

The fault-tolerant architecture of our system is designed with a focus on redundancy and adaptability. Redundant hardware components, such as PIR and LDR sensors, are strategically integrated to collect redundant data streams. This redundancy allows the system to cross-reference and validate data from multiple sources, enhancing its resilience and ensuring the integrity of operations, even in the face of unforeseen challenges or sensor anomalies. A failover mechanism plays a pivotal role in seamlessly transitioning operations from a primary to a secondary component when faults occur, minimizing service disruptions and upholding system availability.

Within our specific system, two distinct types of failover mechanisms have been meticulously engineered to prioritize safety and reliability. The first addresses the connection between streetlights and the cloud, monitoring for issues like internet disconnections. The microcontroller intelligently shifts focus when connectivity problems arise, ensuring the system's continuity. The second failover mechanism deals with potential data conflicts, intelligently prioritizing the most critical data source to resolve conflicts and enhance reliability.

Furthermore, the system incorporates comprehensive monitoring and alerting functions, including dashboards for real-time hardware and connectivity status monitoring and a trigger function for prompt fault notifications. These functions empower system supervision, enabling proactive detection of anomalies or problems. The alerting function becomes instrumental in cases where the system cannot autonomously resolve issues, ensuring timely intervention and minimizing downtime.

In essence, our system's fault-tolerant architecture combines redundancy, failover mechanisms, and proactive monitoring to create a robust and reliable system capable of adapting to diverse data inputs and swiftly addressing challenges. This comprehensive approach enhances the system's overall dependability, performance, and safety.

6.1.2 System Prototype

The system prototype has been developed based on the system block diagram, system data flow diagram and system schematic diagram. After integrating system logic and also the cloud

technology, the system prototype is completed. The system prototype is then fixed on a polystyrene board that has covered with a black paper to act as a road and with a board to block the motion sensor to sensed unwanted object. The view of the prototype has been illustrated in Figure 6.1, 6.2 and 6.3.

To facilitate testing and real-world simulation, the system prototype was securely mounted on a polystyrene board. This board was thoughtfully covered with black paper, serving as a representation of a road surface. Additionally, a board was strategically positioned to obstruct the motion sensor's field of view, simulating scenarios where it needed to ignore unintended objects or movements.

This physical representation of the prototype is vividly captured in Figure 6.1, 6.2 and 6.3. These visual aids provide a comprehensive view of how the prototype is configured, illustrating its layout and components in real-world conditions. This practical approach enhances our understanding of how the system would perform in an operational environment and aids in validating its functionality and adaptability.



Figure 6.1: Top View of System Prototype

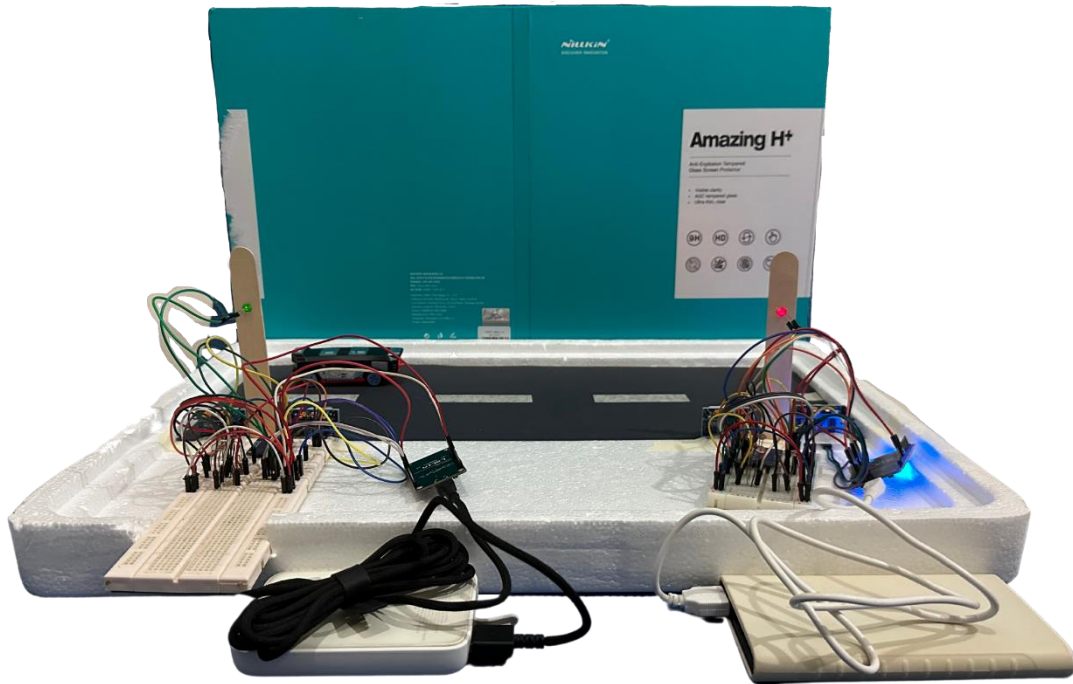


Figure 6.2: Front View of System Prototype

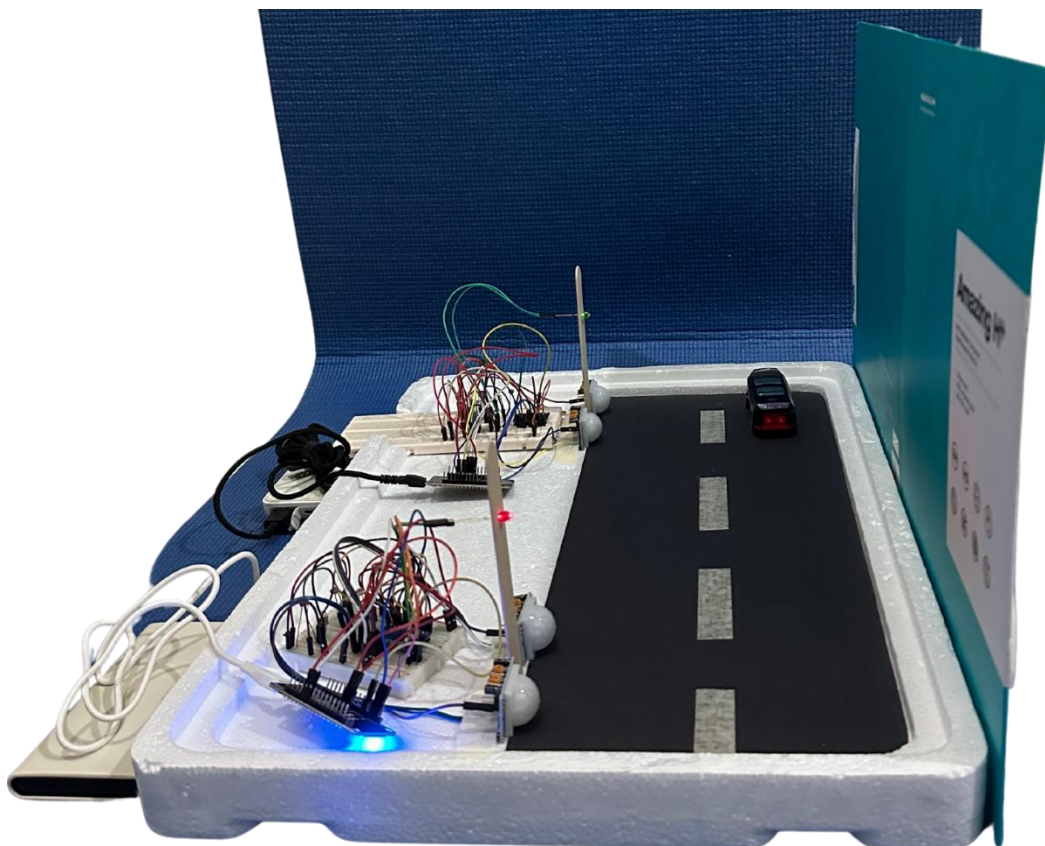


Figure 6.3: Side View of System Prototype

Figure 6.4 shows the prototype create used to test for features fault detection and notification. This prototype used a small box to act as the lamp box. It is used to show that the LDR used to sense the light intensity of the streetlight is implemented inside the lamp box to get a more accurate data.

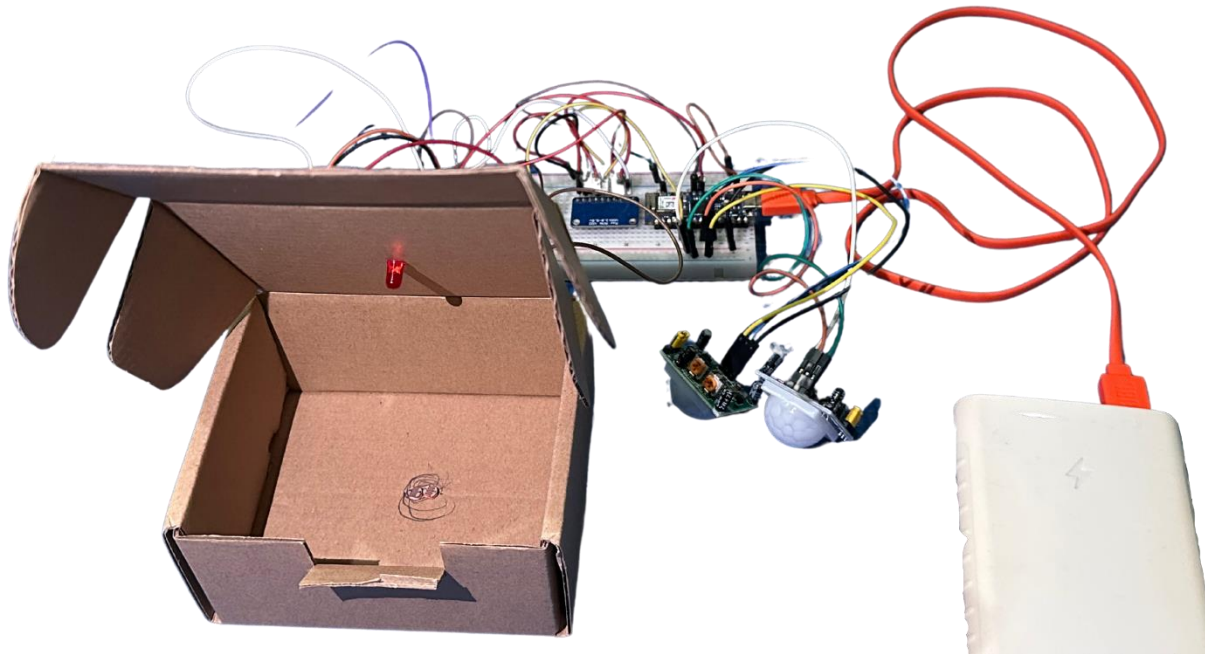


Figure 6.4: System Prototype used to test system fault detection and notification features

6.1.3 Dashboard Interface

We have designed and implemented a user-friendly dashboard interface that offers responsible personnel an intuitive and comprehensive overview of the system. This interface has been meticulously crafted to prioritize ease of use, ease of learning, and ease of comprehension. It empowers users with the capability to monitor system operations in real-time and exercise real-time control over various aspects of the system's functionality.

Figure 6.5 provides a snapshot of one such dashboard interface within the system. This interface serves as a window into the system's operations, presenting essential data and controls in a clear and accessible manner. Responsible personnel can leverage this interface to gain insights into system performance, make informed decisions, and take immediate actions as needed.

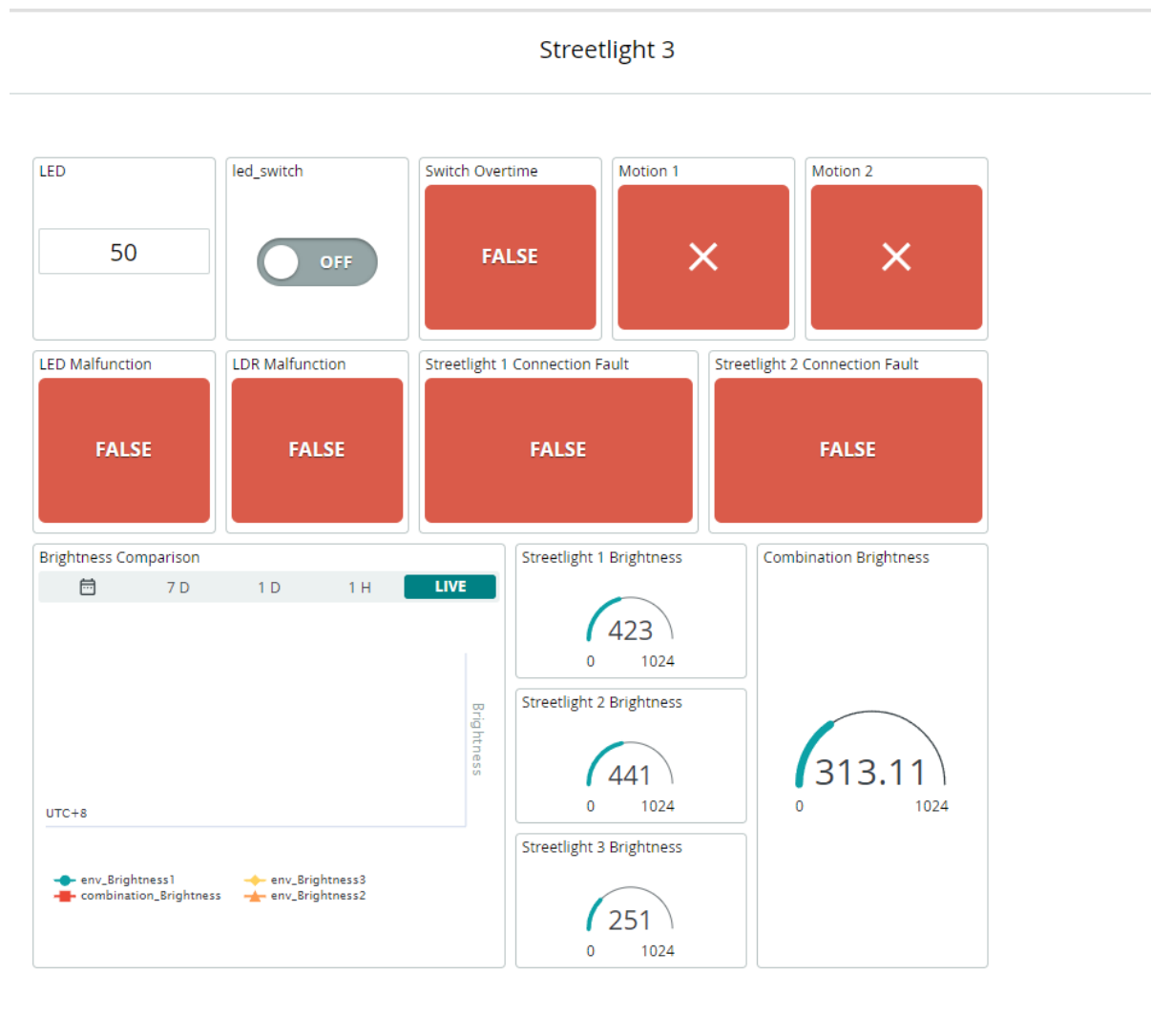


Figure 6.5: Screenshot of Dashboard

The development of this user-friendly dashboard is a critical component of the system, as it enhances usability and empowers personnel to effectively manage and optimize the system's operation in real-world scenarios. It represents a user-centric approach aimed at simplifying system interaction and ensuring efficient monitoring and control.

6.1.4 Notification Functions

In this system, we have implemented a notification function designed to alert the responsible personnel to any anomalies within the system. This function serves a crucial role by eliminating the need for a laborious manual inspection of streetlights, instead offering a direct pinpointing of the problematic streetlight. This not only streamlines the process but also expedites the resolution of streetlight issues, thereby reducing downtime and minimizing service disruptions.

The incorporation of notification functions into the system represents a significant enhancement in fault tolerance, robustness, and reliability. It fundamentally transforms the system into a proactive, adaptive, and highly responsive entity capable of swiftly and effectively addressing various challenges. Consequently, this integration substantially elevates the system's dependability and overall performance, ensuring a more efficient and dependable streetlight management system.

6.2 System Testing

6.2.1 Automated Brightness Adjustment (TC1)

The purpose of this system is to showcase its capability to autonomously adjust streetlight intensity based on environmental conditions and traffic flow. Figures 6.6, 6.7, and 6.8 depict screenshots captured from the serial monitor during the testing phase. In these figures, key data points are presented for analysis.

Specifically, 'Motion detected 1' and 'Motion detected 2' serve as indicators of motion detection by the system's sensors. 'Environment brightness' offers a numerical representation of the ambient light conditions, ranging from 0 to 1023. To simplify interpretation, we have categorized this range into three distinct levels, aligning with the fuzzy rules defined in the previous chapter. A brightness reading falling within the 0 to 450 range signifies a dark environment, while 350 to 550 suggests an average brightness, and 450 to 1023 corresponds to a bright environment. Additionally, the 'LED brightness level' denotes the current light intensity setting of the LED, with levels 1, 2, and 3 indicating low, average, and high intensity, respectively.

In Figure 6.6, we observe that the streetlight effectively provides an average light intensity (LED brightness level: 2) to its surroundings. This adjustment is in response to the dark environmental conditions (Environment brightness: 215.22) and the absence of motion detection (both 'Motion detected 1' and 'Motion detected 2' registering as 0). This successful demonstration underscores the system's ability to adapt to varying conditions and optimize streetlighting accordingly.

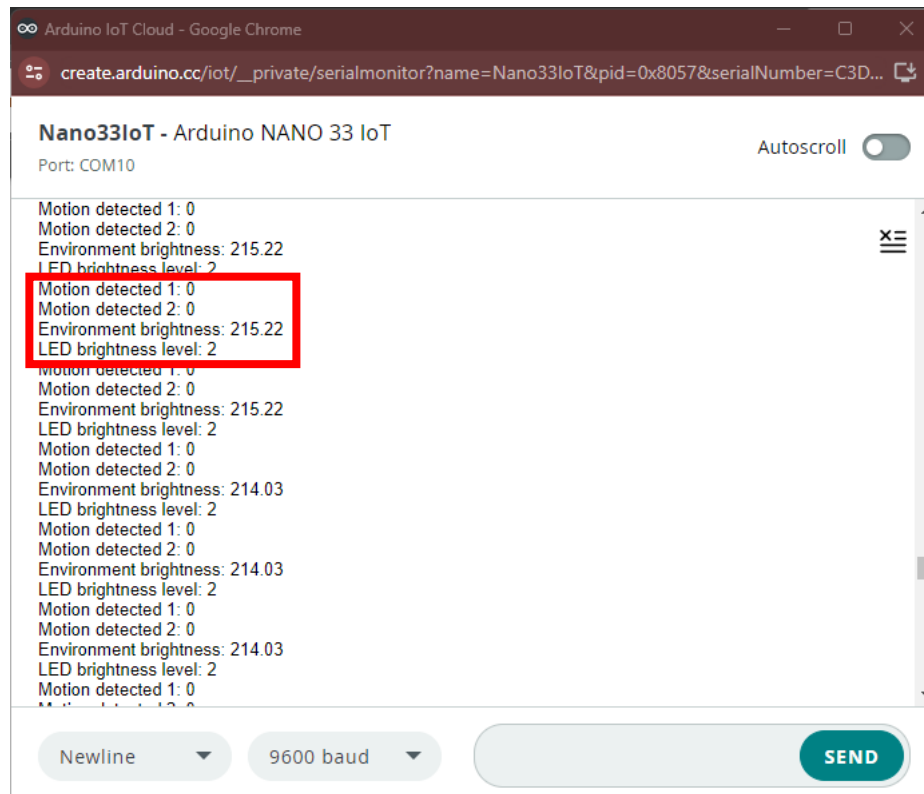


Figure 6.6: Screenshot of Serial Monitor 1

Figure 6.7 aptly demonstrates the system's capacity to deliver maximum illumination (LED brightness level: 3) to its surroundings in response to a dark environment (Environment brightness: 216.41) and the detection of motion (both 'Motion detected 1' and 'Motion detected 2' registering as 1).

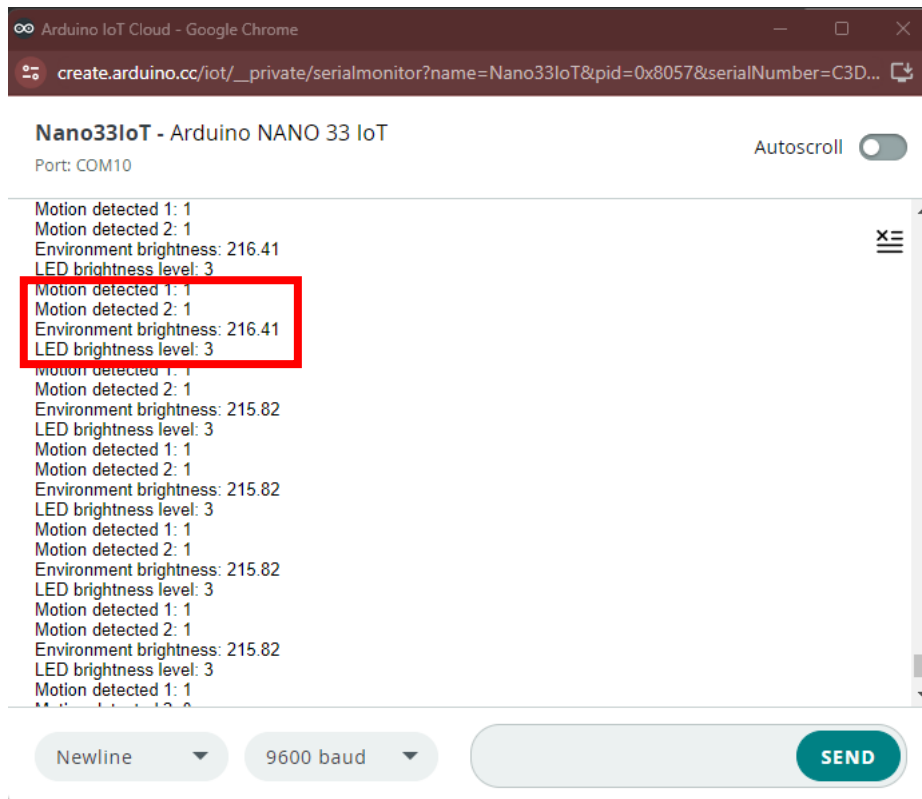


Figure 6.7: Screenshot of Serial Monitor 2

In Figure 6.8, it is evident that the streetlight effectively deactivates its LED (LED brightness level: 1) when the ambient light intensity reaches a sufficiently high level (Environment brightness: 782.98), regardless of whether motion is detected or not.

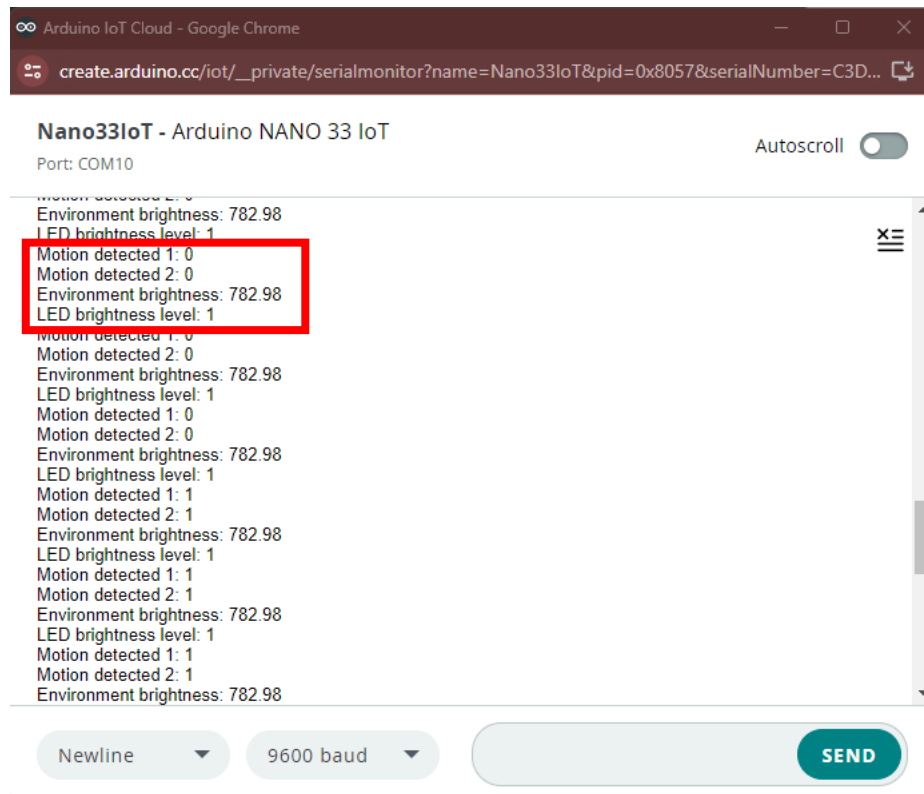


Figure 6.8: Screenshot of Serial Monitor 3

6.2.2 Fault Tolerance (TC2)

The purpose of this testing is to illustrate the system's resilience in the face of faults. Figure 6.9 provides a glimpse of Streetlight 1's dashboard, revealing a conflict between the data generated by motion sensor 1 and motion sensor 2 within Streetlight 1. Despite this discrepancy, the streetlight admirably manages to arrive at the correct decision, maximizing the LED light intensity to level 255.

Referring to the fuzzy rules we defined in the previous chapter, we understand that when the environment is poorly lit and motion is detected, the LED should deliver maximum illumination. Figure 6.1 verifies this scenario, as the combined brightness reading of 201.042 clearly falls within the low environmental light intensity category. Consequently, the system exhibits its fault tolerance by generating the appropriate output even in the presence of conflicting sensor data. This robust performance underscores the system's capacity to handle faults and maintain functionality under adverse conditions.

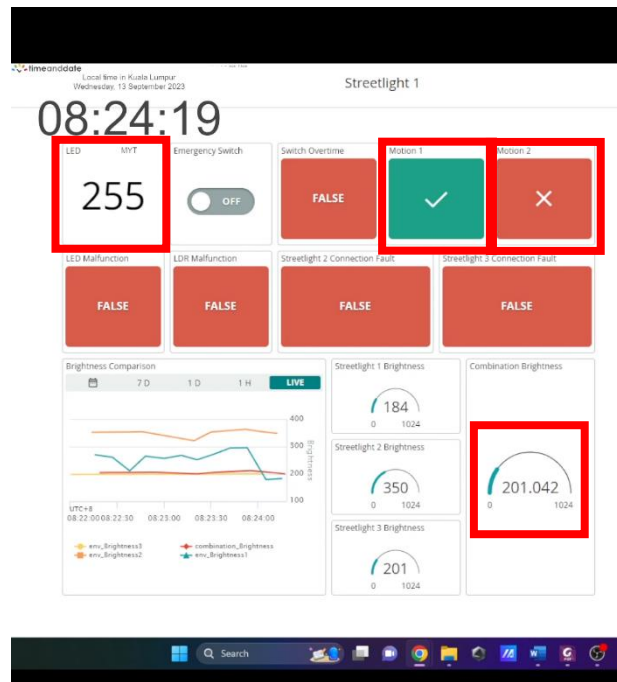


Figure 6.9: Screenshot of Dashboard of Streetlight 1

6.2.3 Fault Detection and Notification (TC3)

The purpose of this testing is to demonstrate the system's proficiency in detecting faults and promptly notifying the relevant parties. In Figure 6.10, we can observe an email notification received from the Arduino Cloud. The system exhibited its capability to recognize when the streetlight's LED was not functioning as anticipated, triggering an automatic email alert to notify the responsible personnel. Figure 6.10 serves as compelling evidence that the fault detection and notification mechanisms within the system are operating seamlessly and effectively.

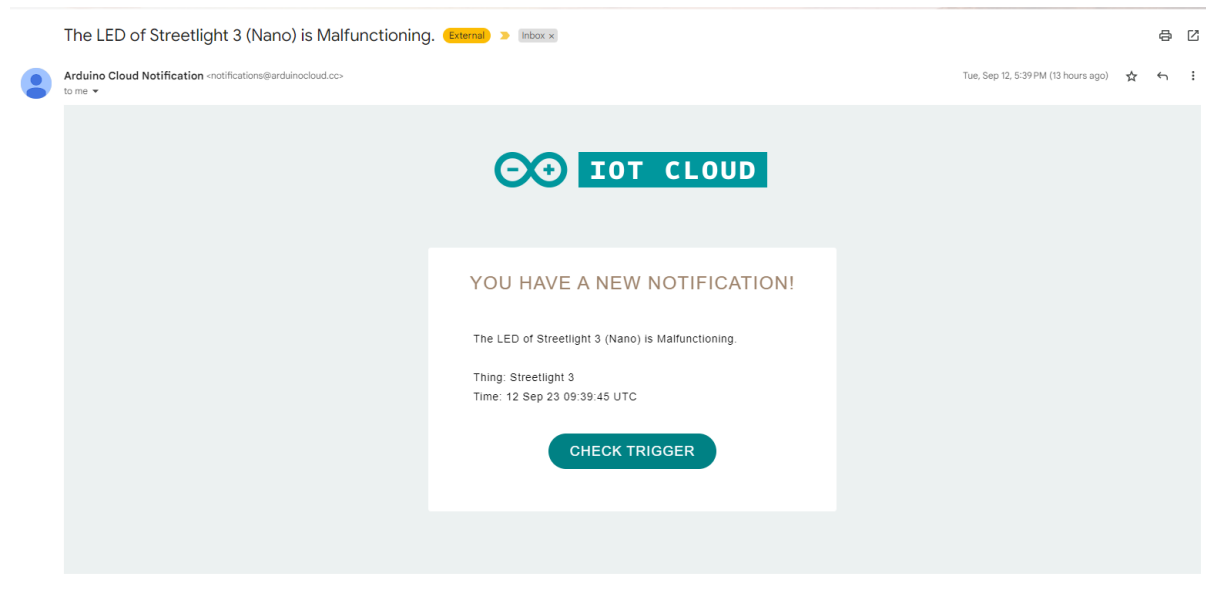


Figure 6.10: E-mail Notification

6.2.4 Internet of Things (IoT) (TC4)

In this testing scenario, Streetlight 1 is positioned ahead of Streetlight 2, implying that Streetlight 2 is expected to provide advanced lighting when Streetlight 1 detects a moving object. Figure 6.11 captures a pivotal moment at 08:24:19 when one of Streetlight 1's motion sensors registered activity in its vicinity. Almost instantaneously, Streetlight 1 transmitted this data to the cloud. Subsequently, as Figure 6.12 illustrates, Streetlight 2 received this data from Streetlight 1 precisely at 08:24:20.

In response to this data transmission, Streetlight 2 swiftly adjusted its LED output to its maximum level, set at 255. It's noteworthy from Figure 6.12 that Streetlight 2's motion sensor did not detect any motion at this time. This observation reaffirms that Streetlight 2's LED output was driven by the data it received from Streetlight 1 rather than its internal motion sensor.

This seamless interaction vividly showcases the streetlights' capability to communicate and collaborate, leveraging cloud-based updates as their medium of exchange. This interconnectedness greatly enhances their collective understanding of the environment, enabling more informed and precise decision-making. In essence, it demonstrates the system's adaptability and its ability to respond effectively to dynamic situations by leveraging data sharing and collaboration among streetlights.

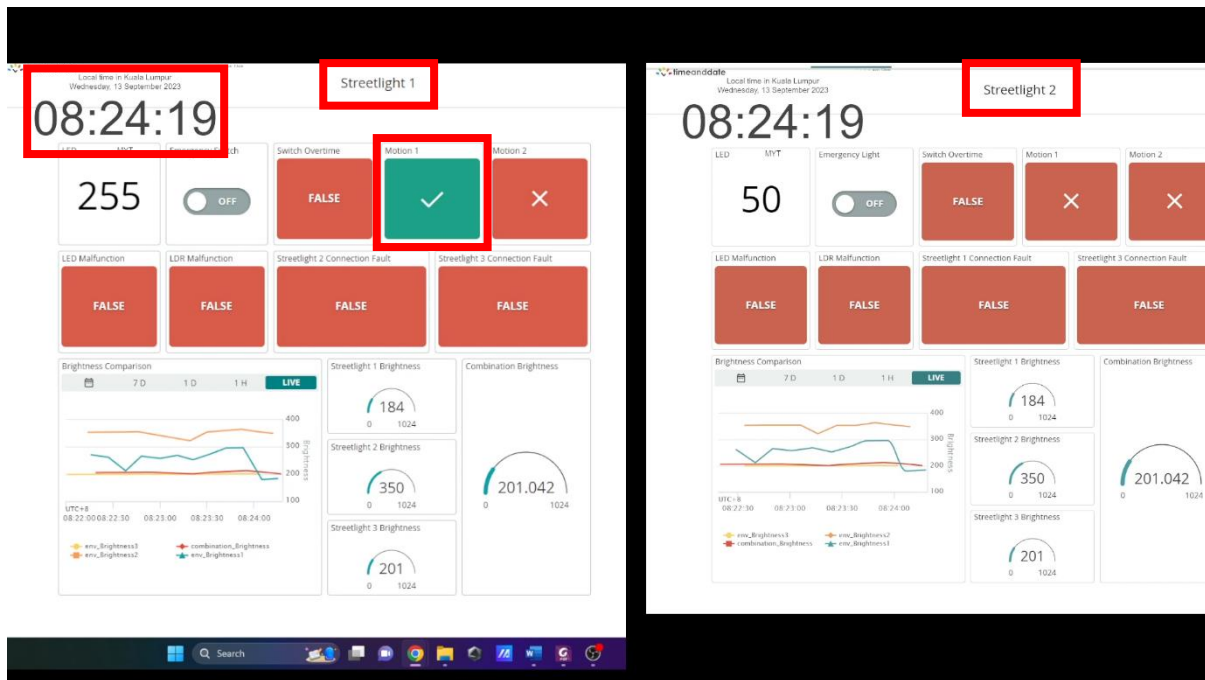


Figure 6.11: Screenshot of Dashboards during Testing at time 08:24:19.

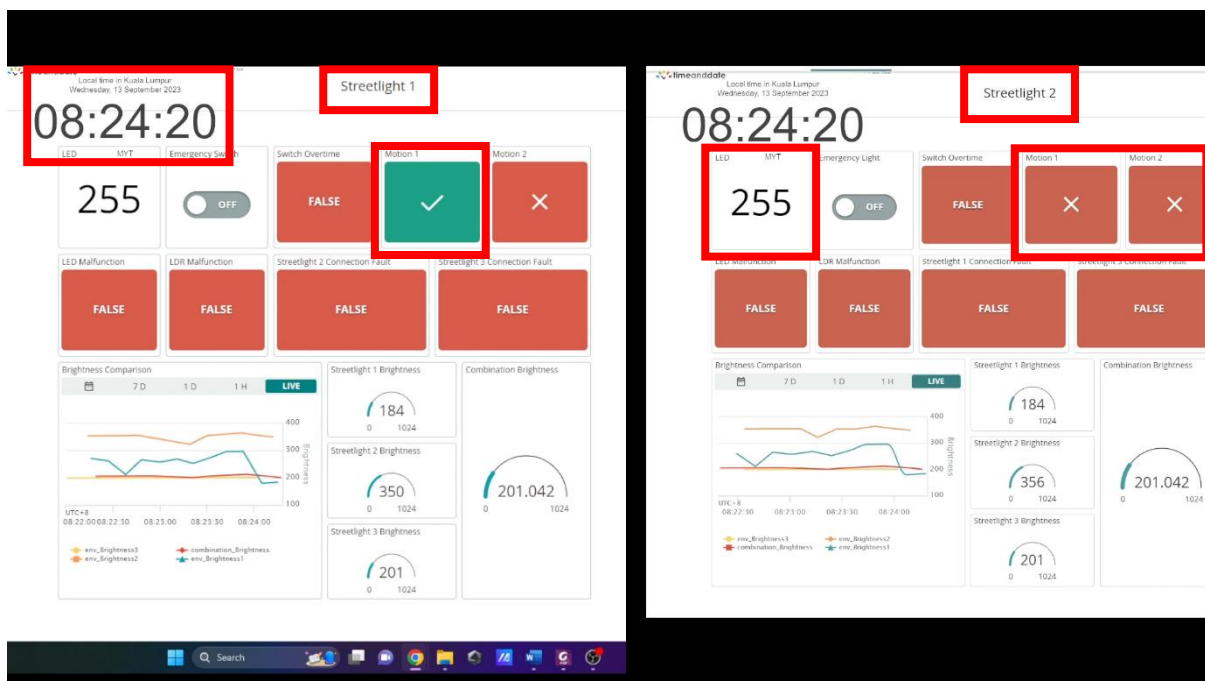


Figure 6.12: Screenshot of Dashboards during Testing at time 08:24:20.

6.2.5 Test Result

Table 6.1 presents the outcomes of our rigorous testing regimen. Within this table, we observe that we meticulously executed four distinct functional tests on the system through manual intervention, and it is gratifying to report that all of these tests have yielded successful results.

The fact that all four functional tests passed without any issues demonstrates the robustness and reliability of our system. This positive outcome reflects the meticulous development and testing processes that were carried out, reinforcing the confidence in the system's ability to perform as intended. Such a level of validation is pivotal when considering the system's readiness for deployment and its potential impact on real-world scenarios.

Test Result				
Plan ID	Type of Testing	Tool	Pass/Fail	Remark
P1	Functional	Manual	Pass	-
P2	Functional	Manual	Pass	-
P3	Functional	Manual	Pass	-
P4	Functional	Manual	Pass	-

Table 6.1: Test Result table.

CHAPTER 7 – CONCLUSION

7.1 Conclusion

This project endeavours to introduce an intelligent streetlight system designed to dynamically adjust its light intensity based on environmental light levels and traffic flow. Adding to its sophistication, this intelligent streetlight system incorporates a robust fault detection and fault tolerance mechanism, employing Internet of Things (IoT) technology and the concept of sensor fusion. By integrating fault tolerance measures into the system, it significantly enhances its reliability and availability, consequently elevating the safety of implementing such a system on roadways.

In the course of this project, three prototypes have been successfully developed, each serving as a functional streetlight unit. These systems utilize Light-Dependent Resistor (LDR) sensors for capturing ambient light intensity data, as well as Light Emitting Diode (LED) and Passive Infrared (PIR) sensors to gather motion data within specific areas. The integration of a multiplexer facilitates the conversion of analogue data received from the LDR sensors into digital format, while a microcontroller is entrusted with processing the collected data. The microcontroller operates in conjunction with a fuzzy logic algorithm, which employs fuzzification and defuzzification techniques to interpret data and make informed decisions regarding LED output. Consequently, the system possesses the capability to dynamically adjust LED brightness in response to environmental lighting conditions and traffic dynamics.

The data generated by these sensors is seamlessly shared among the streetlights via cloud connectivity. This data exchange enhances the reliability and provides a comprehensive analysis of the environmental situation. In summation, this project represents a groundbreaking approach to street lighting, harmoniously combining sensor fusion, fuzzy logic algorithms, IoT technology, cloud connectivity, and microcontrollers to yield a system that is not only highly efficient but also remarkably adaptive to the unique requirements of its environment.

Finally, the test results unequivocally affirm the capabilities of our proposed system, demonstrating its ability to perform key functions such as automated brightness adjustments, fault tolerance, fault detection and notification, and effective utilization of Internet of Things (IoT) technology.

7.2 Novelties and Contributions

The primary innovation in this project revolves around leveraging Internet of Things (IoT) technology to bolster the fault tolerance mechanism within the intelligent streetlight system. This innovation is pivotal because it empowers the intelligent streetlight system to make more reliable decisions while minimizing the number of sensors required to achieve fault tolerance. This achievement is made possible by enabling the streetlights to seamlessly exchange data with one another, thereby providing a more comprehensive understanding of their environment.

As previously discussed in the preceding chapter, the conventional approach to ensuring fault tolerance often necessitates redundant sensors within the system to validate the accuracy of collected data. However, the introduction of this IoT-enabled concept transforms the landscape. Streetlights can now directly gather data from neighbouring streetlights, reducing their reliance solely on the data generated by their own sensors. This collaborative data exchange mechanism not only enhances the system's fault tolerance but also promotes data accuracy and reliability.

Furthermore, the proposed solution simplifies the user experience by eliminating the need for intricate configuration processes typically associated with setting up Cloud architecture for IoT development. This simplification is facilitated through the use of a preconfigured IoT device, which serves as the proposed solution's foundation. This IoT device comes equipped with pre-established Cloud connectivity, alleviating users of the burdensome task of configuring Cloud themselves. This streamlined approach enhances the accessibility and usability of the intelligent streetlight system, making it more user-friendly and efficient.\

7.3 Future Works

In the context of the upcoming project, the utilization of Artificial Intelligence (AI) or machine learning technologies is strongly advised. These cutting-edge technologies can significantly elevate the performance of the fault-tolerant mechanism and enhance overall system reliability. This recommendation is rooted in the current project's reliance on safety constraints, which dictate that the LED should produce the maximum output when conflicts arise in the collected data.

While safety constraints are undoubtedly valuable for ensuring system stability, they lack the capacity to make predictive decisions based on data patterns. This is where AI and machine learning come into play. By integrating these technologies, the system gains the ability to

analyse incoming data, identify patterns and trends, and subsequently make informed decisions by predicting the expected output.

In essence, AI and machine learning empower the system to go beyond simple rule-based responses and enable it to adapt and optimize its behaviour based on evolving data conditions. This not only enhances fault tolerance but also contributes to improved system reliability, as it can proactively address potential issues and conflicts, ultimately leading to a more robust and efficient project outcome.

REFERENCES

- [1] A. Ożadowicz and J. Grela, "Energy saving in the street lighting control system—a new approach based on the EN-15232 standard," *Energy Efficiency*, 10(3), 2017, pp.563-576.
- [2] S. M. Sorif, D. Saha and P. Dutta, "Smart Street Light Management System with Automatic Brightness Adjustment Using Bolt IoT Platform," 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), 2021, pp. 1-6, doi: 10.1109/IEMTRONICS52119.2021.9422668.
- [3] C. Badgaiyan, P. Sehgal, Smart street lighting system, *international journal of science and research*, vol. 4, 2015, pp. 271 – 274
- [4] M. L. Fung, M. Z. Q. Chen and Y. H. Chen, "Sensor fusion: A review of methods and applications," 2017 29th Chinese Control And Decision Conference (CCDC), 2017, pp. 3853-3860, doi: 10.1109/CCDC.2017.7979175.
- [5] N. Ouerhani, N. Pazos, M. Aeberli and M. Muller, "IoT-based dynamic street light control for smart cities use cases," 2016 International Symposium on Networks, Computers and Communications (ISNCC), 2016, pp. 1-5, doi: 10.1109/ISNCC.2016.7746112.
- [6] N. Pazos, M. Müller, M. Aeberli and N. Ouerhani, "ConnectOpen - automatic integration of IoT devices," 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, pp. 640-644, doi: 10.1109/WF-IoT.2015.7389129.
- [7] G. Twesigye, A. Ngenzi and E. Ndashimye, "An Embedded Fuzzy Logic Based Smart Street Lighting System," 2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON), 2022, pp. 1-5, doi: 10.1109/NIGERCON54645.2022.9803089.
- [8] R. B. Caldo et al., "Design and development of fuzzy logic controlled dimming lighting system using Arduino microcontroller," 2015 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), Cebu, Philippines, 2015, pp. 1-6, doi: 10.1109/HNICEM.2015.7393161.
- [9] V. P. Nelson, "Fault-tolerant computing: fundamental concepts," in *Computer*, vol. 23, no. 7, pp. 19-25, July 1990, doi: 10.1109/2.56849.
- [10] R. E. Gibson, D. L. Hall and J. A. Stover, "An autonomous fuzzy logic architecture for multisensor data fusion," *Proceedings of 1994 IEEE International Conference on MFI '94. Multisensor Fusion and Integration for Intelligent Systems*, Las Vegas, NV, USA, 1994, pp. 143-150, doi: 10.1109/MFI.1994.398450.
- [11] M. A. A. Akhoundi and E. Valavi, "Multi-sensor fuzzy data fusion using sensors with different characteristics," *arXiv.org*, 08-Sep-2019. [Online]. Available: <https://arxiv.org/abs/1010.6096>. [Accessed: 18-Apr-2023].

REFERENCES

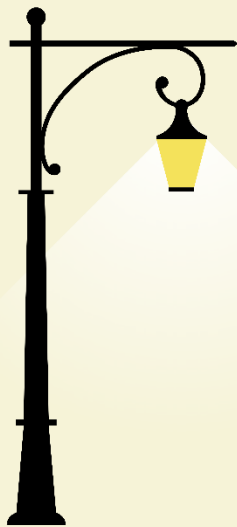
- [12] R. Fan, L. Li and X. Hong, "Research On Application of Data Fusion In Streetlights Control System," 2020 Chinese Automation Congress (CAC), Shanghai, China, 2020, pp. 739-742, doi: 10.1109/CAC51589.2020.9327316.
- [13] D. L. Hall and J. Llinas, "An introduction to multisensor data fusion," in Proceedings of the IEEE, vol. 85, no. 1, pp. 6-23, Jan. 1997, doi: 10.1109/5.554205.
- [14] P. P. F. Dheena, G. S. Raj, G. Dutt and S. V. Jinny, "IOT based smart street light management system," 2017 IEEE International Conference on Circuits and Systems (ICCS), Thiruvananthapuram, India, 2017, pp. 368-371, doi: 10.1109/ICCS1.2017.8326023.
- [15] J. Qian, B. Duan, W. Xie and Y. Zhao, "Edge Computing for Brightness and Color Temperature of Smart Streetlight," 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), Nanchang, China, 2021, pp. 699-703, doi: 10.1109/ICBAIE52039.2021.9390033.
- [16] R. E. Harper, J. H. Lala and J. J. Deyst, "Fault tolerant parallel processor architecture overview," [1988] The Eighteenth International Symposium on Fault-Tolerant Computing. Digest of Papers, Tokyo, Japan, 1988, pp. 252-257, doi: 10.1109/FTCS.1988.5328.
- [17] R. E. Strom, D. F. Bacon and S. A. Yemini, "Volatile logging in n-fault-tolerant distributed systems," [1988] The Eighteenth International Symposium on Fault-Tolerant Computing. Digest of Papers, Tokyo, Japan, 1988, pp. 44-49, doi: 10.1109/FTCS.1988.5295.
- [18] A. Avizienis, M. R. Lyu and W. Schutz, "In search of effective diversity: a six-language study of fault-tolerant flight control software," [1988] The Eighteenth International Symposium on Fault-Tolerant Computing. Digest of Papers, Tokyo, Japan, 1988, pp. 15-22, doi: 10.1109/FTCS.1988.5291.

APPENDIX

Poster



IOT-BASED INTELLIGENT STREETLIGHT SYSTEM WITH FAULT-TOLERANT MECHANISM USING SENSOR FUSION



PROJECT DEVELOPER Saw Wei Chin
PROJECT SUPERVISOR Ts Dr Chang Jing Jing

INTRODUCTION

The idea of intelligent streetlights has emerged as a proposed solution to address the challenge of reducing streetlight energy consumption without compromising public safety.

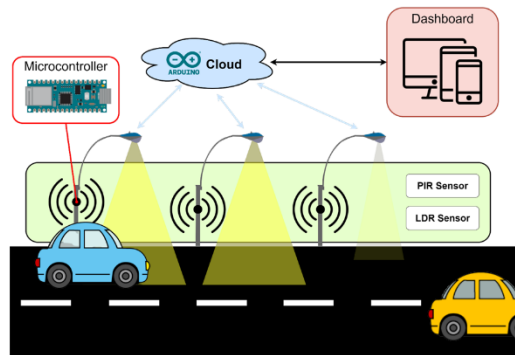
PROBLEM STATEMENT

The conventional street lighting system has high energy consumption and low flexibility, while typical intelligent street lighting systems do not take hardware faults into account, resulting in a lack of fault-tolerance control. Hence, low availability and reliability.

OBJECTIVE

1. To develop an intelligent streetlight system that adjusts the intensity of light based on the light intensity of the environment and traffic flow using sensor fusion.
2. Design a fault-tolerant architecture for an intelligent streetlight system.
3. To implement fault-tolerant mechanism and Internet of Things (IoT) in the intelligent streetlight system with cloud technology.

SYSTEM ARCHITECTURE DIAGRAM



SYSTEM USER INTERFACE (UI)



SYSTEM FEATURES

- Automated Brightness Adjustment
- Fault Tolerance
- Fault Detection and Notification
- Internet of Things (IoT)
- Informative UI for real-time monitoring and control

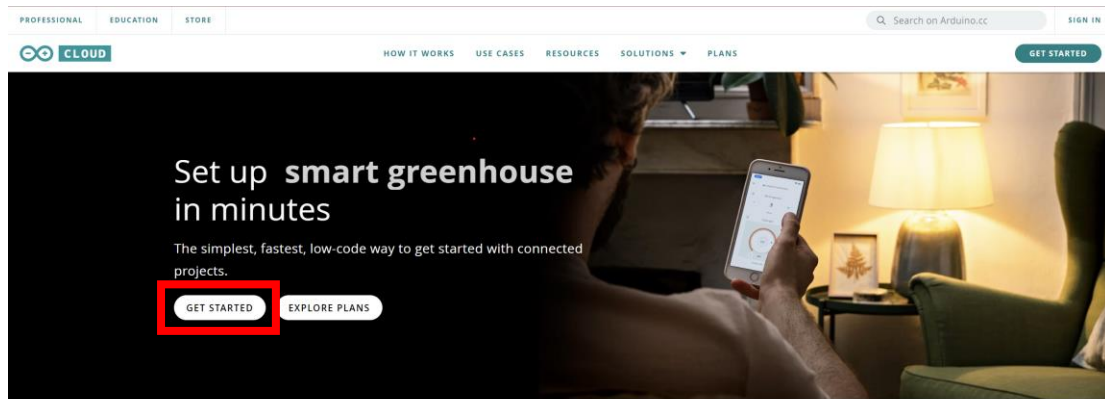
CONCLUSION

The proposed solution has been successfully implemented with automated brightness adjustment.

The system comes with features that are fault-tolerant and Fault detection and Notification with the integration of IoT.

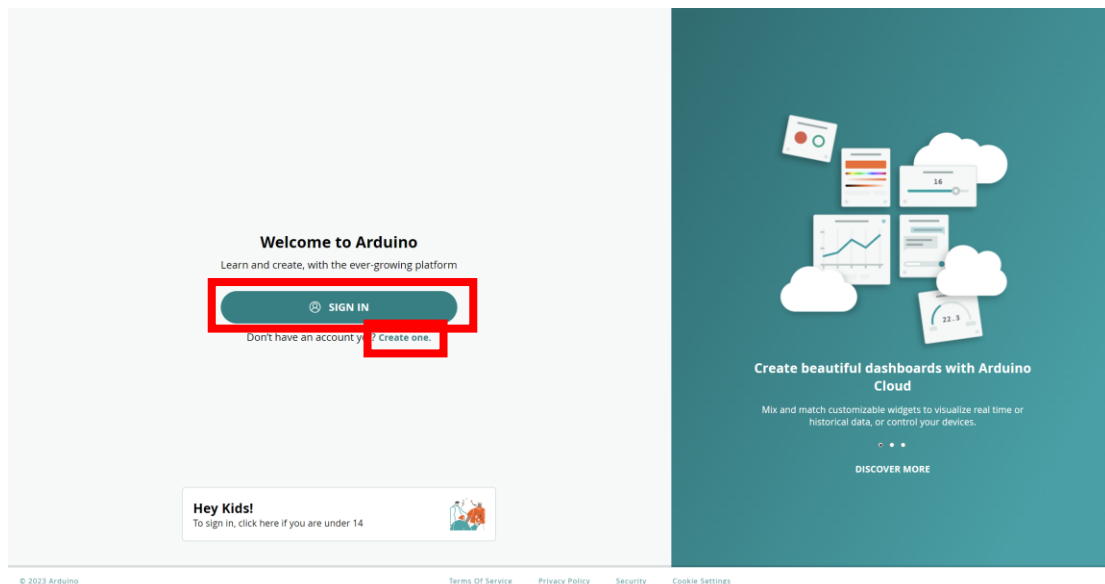
Steps to Setup Arduino Cloud

1. Access the Arduino Cloud webpage by clicking on the following link: <https://cloud.arduino.cc/>.
2. Once on the Arduino Cloud homepage, locate and click on the "GET STARTED" button as shown in Figure 4.9.



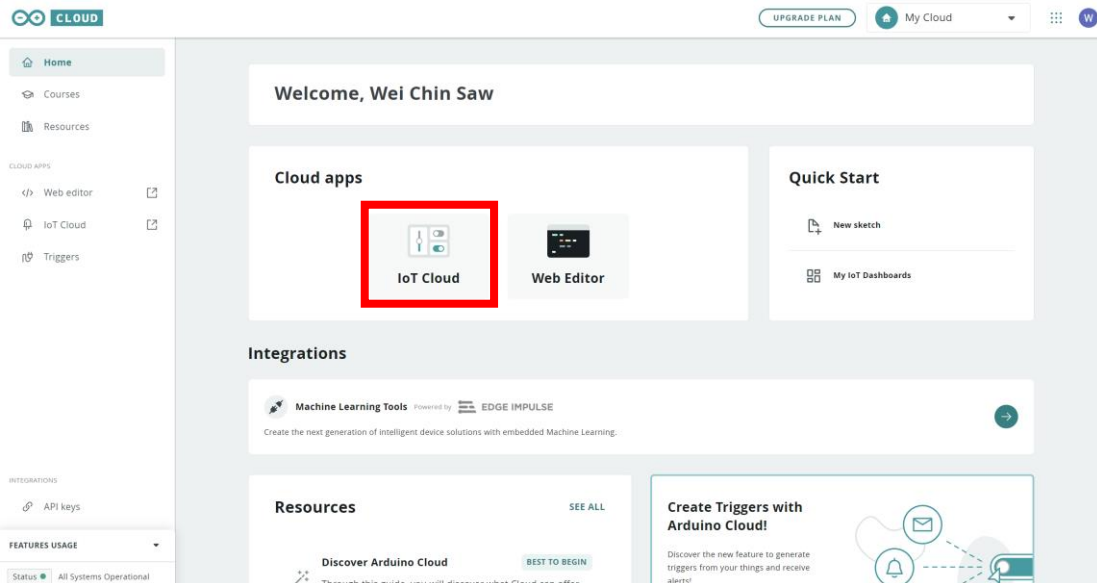
Connected projects, simplified

3. If you already have an Arduino Cloud account, click on the "SIGN IN" button to log in. If you don't have an account, you can create one by clicking on the "Create One" button.

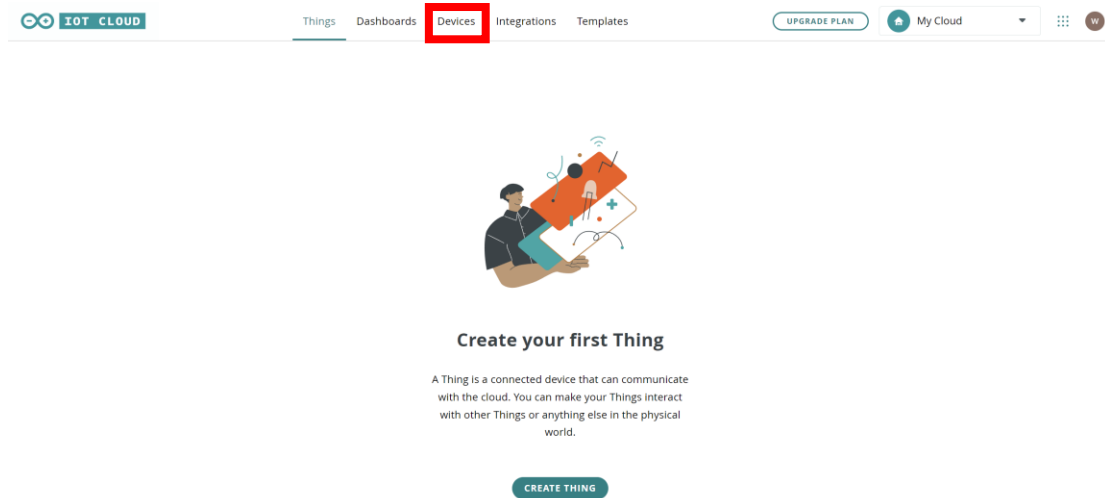


APPENDIX

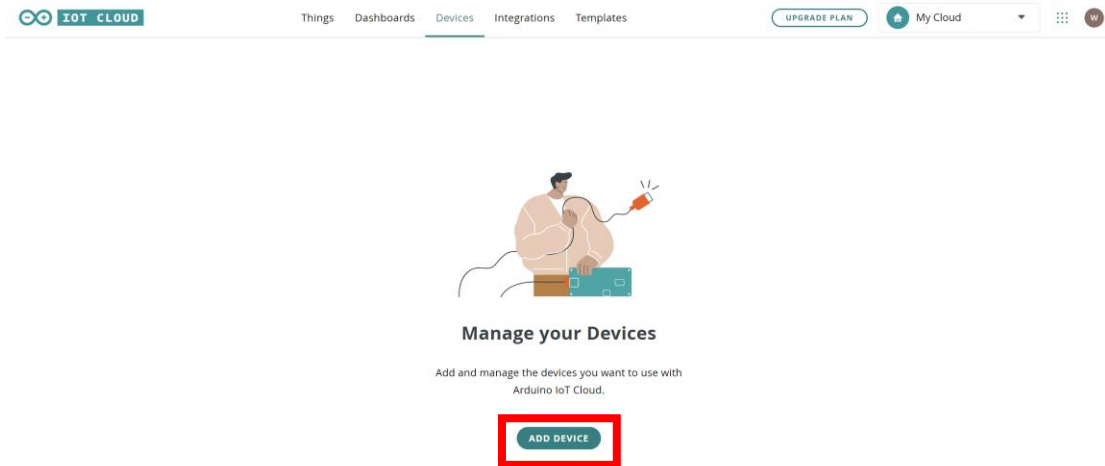
- After successfully logging into your Arduino account, you will be directed to the Arduino Cloud dashboard.



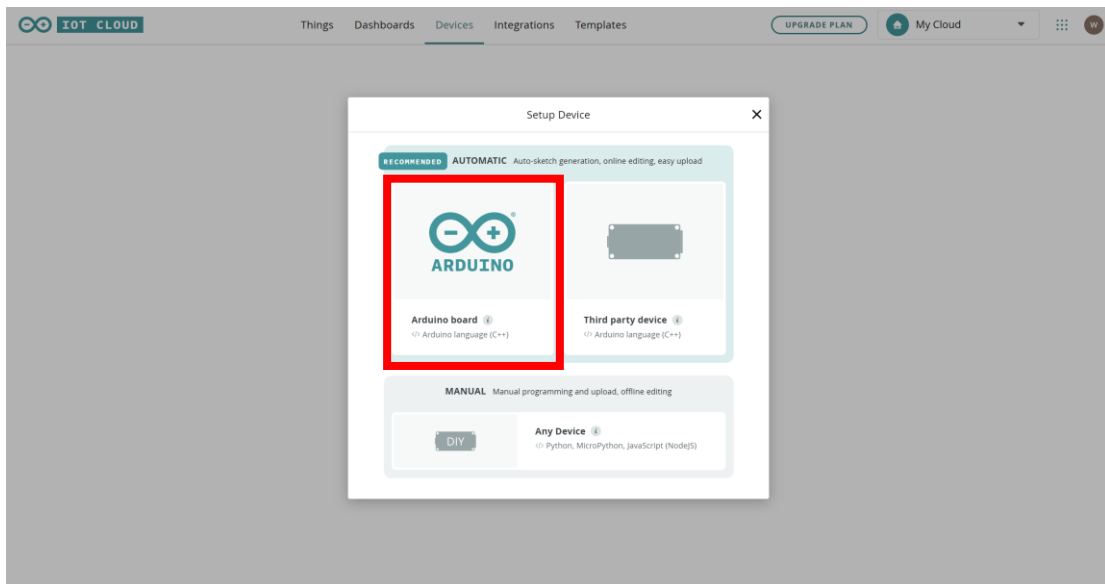
- Click on the "IoT Cloud" button to access the Arduino IoT Cloud apps.



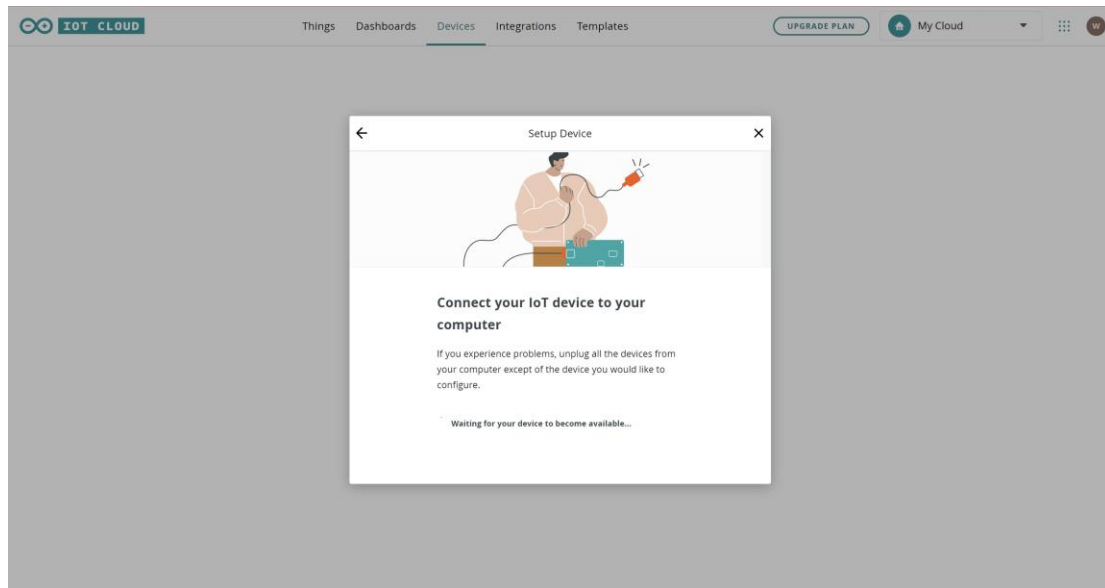
6. Navigate to the "Devices" section to view the devices connected to the cloud.



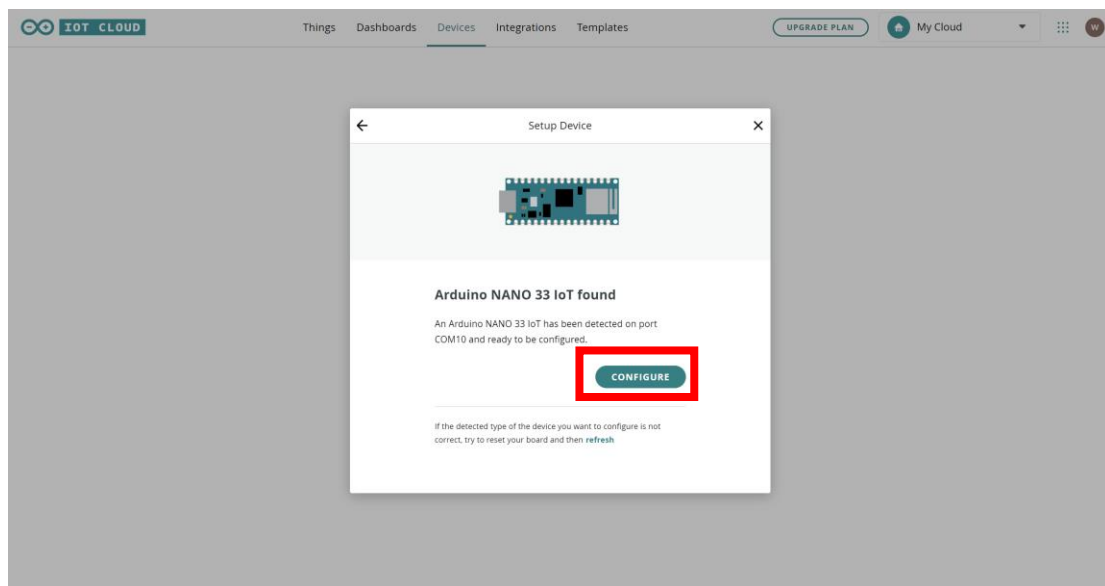
7. To add your microcontroller to the cloud, click on the "ADD DEVICE" button.



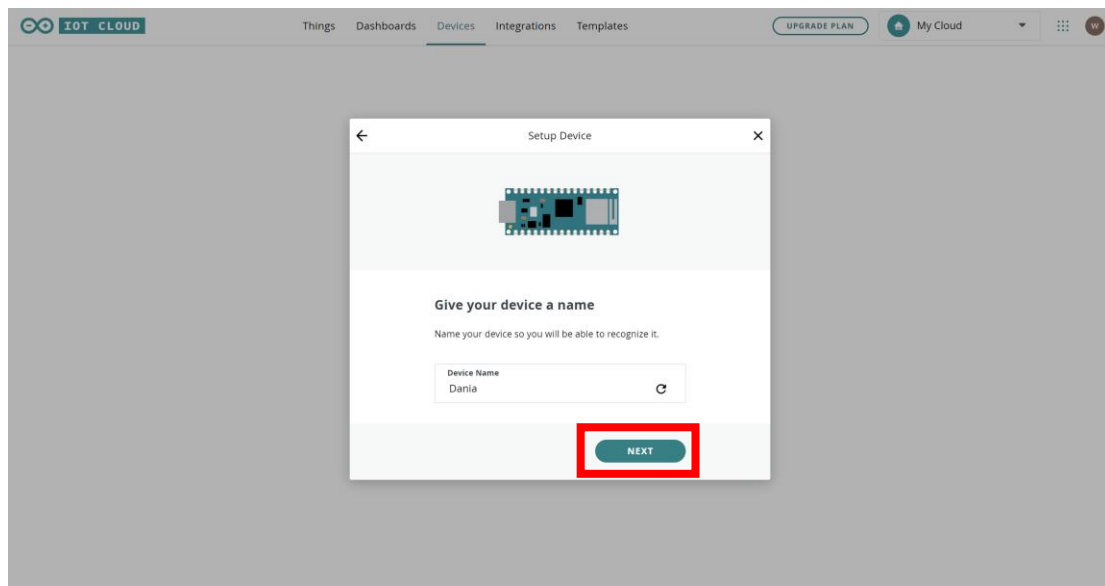
8. Choose "Arduino board" if you're using an Arduino microcontroller. If you're using a third-party device or any other device, select the relevant option. Note that setting up devices requires the installation of the Arduino Create Agent on your PC.



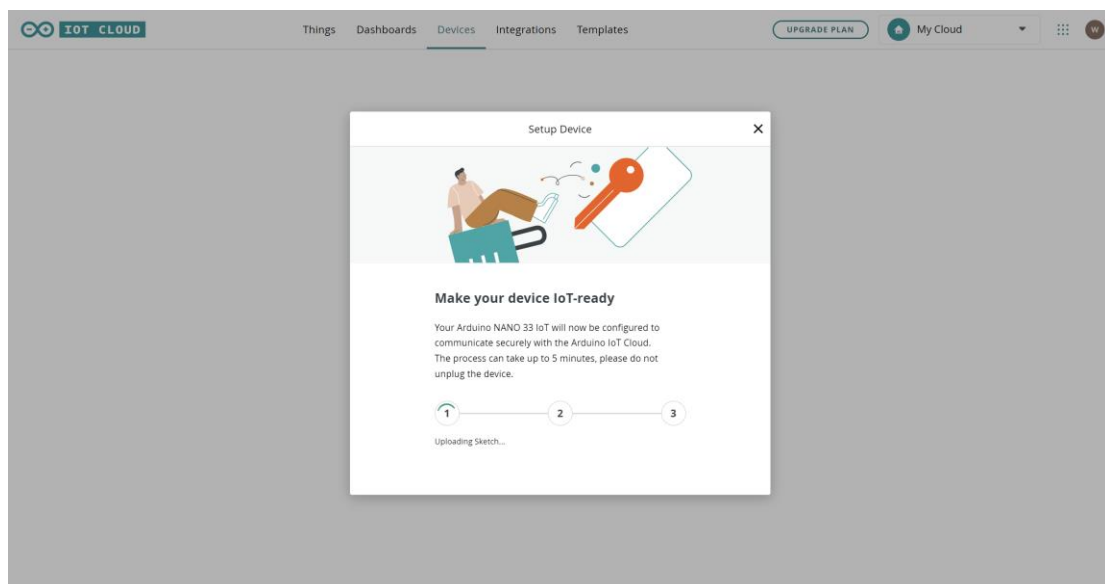
9. Connect your Arduino Board to your PC via USB, and then click the "Arduino board" button. The Arduino Cloud will automatically detect connected devices and display them.



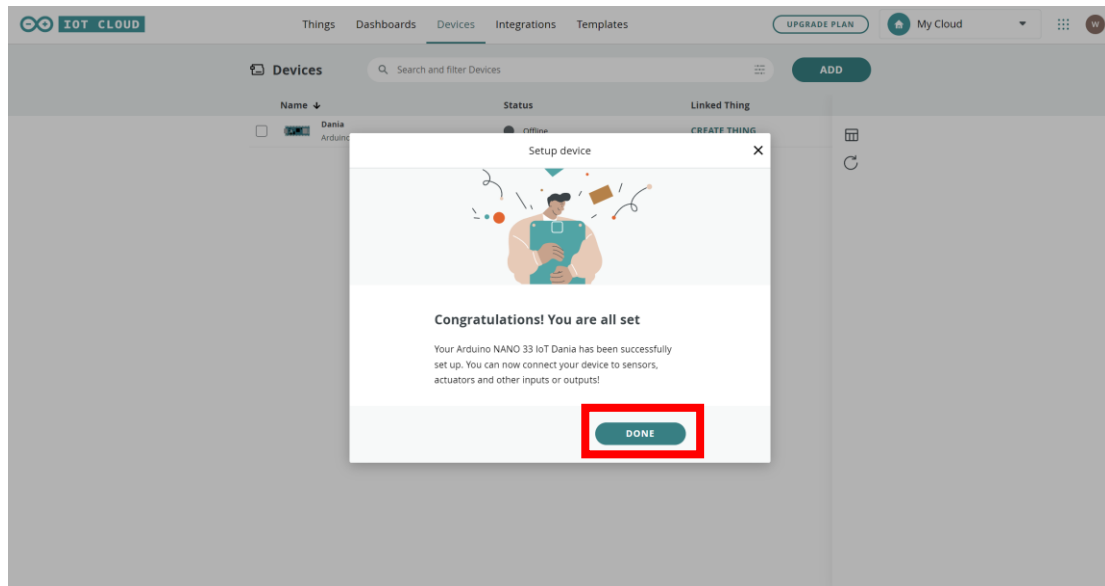
10. Click the "CONFIGURE" button to set up the microcontroller.



11. Provide a name for the device and click "NEXT".

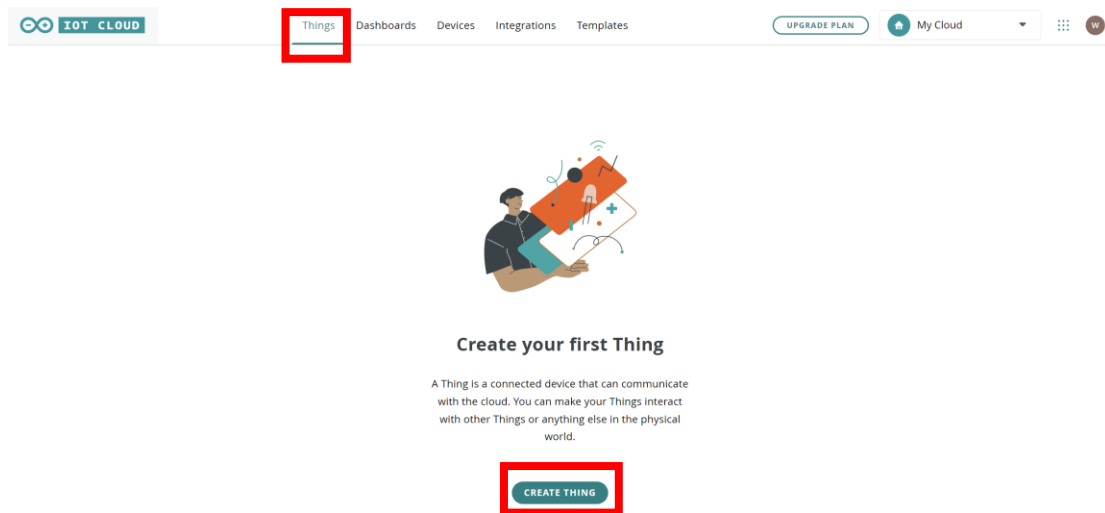


12. The Arduino Cloud will register the device in the cloud.

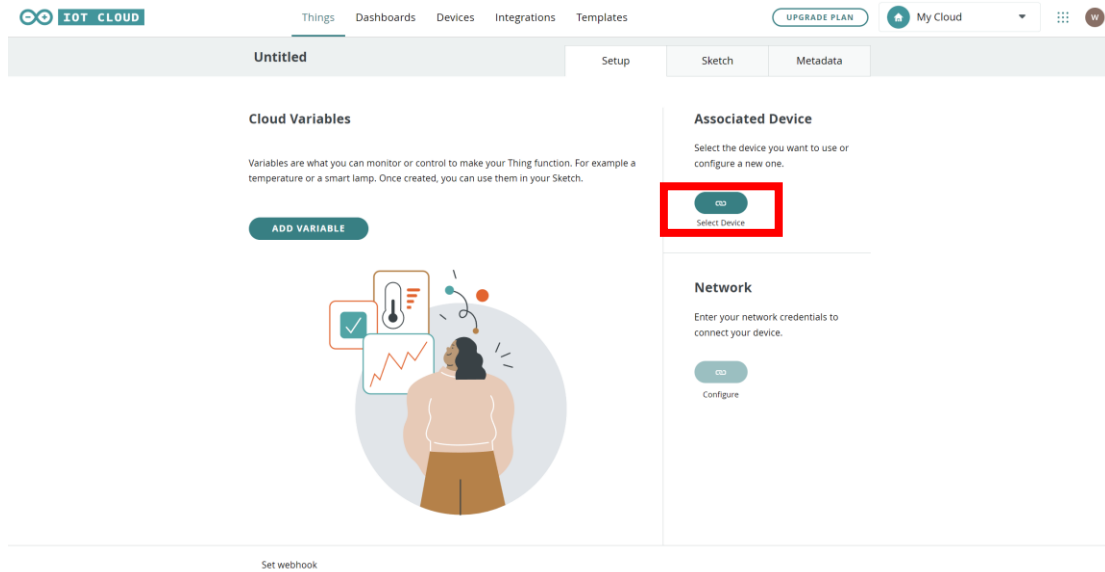



13. After the device is registered press "DONE" button to continue.

14. To create a "Thing" in the cloud, go to the "Things" section.

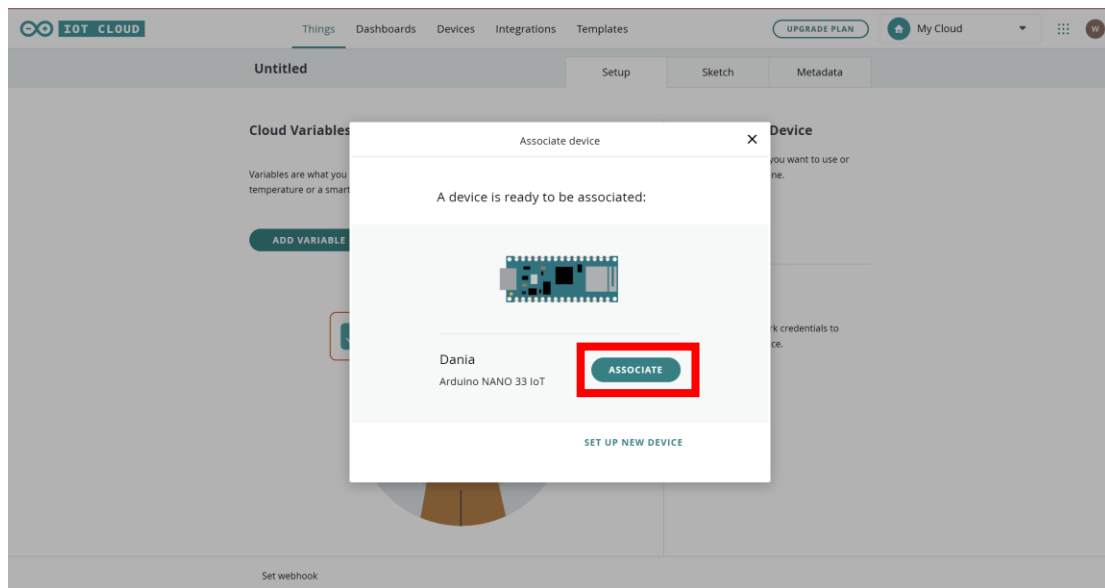


15. Click the "CREATE THING" button to create a new Thing in the cloud.

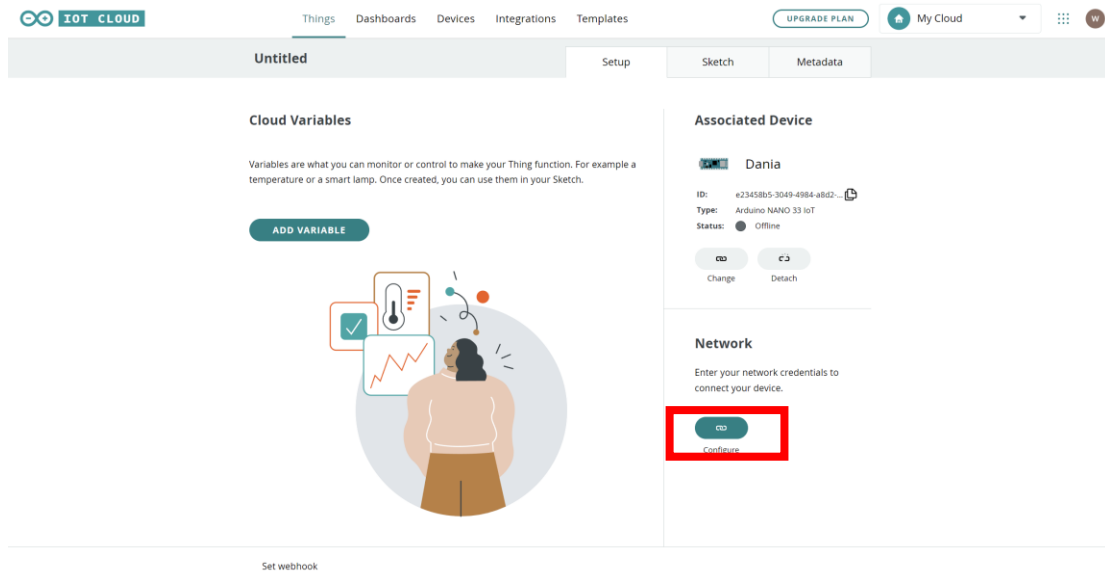


16. In the "Associate Device" section, press the " "  button to connect the device to the thing.

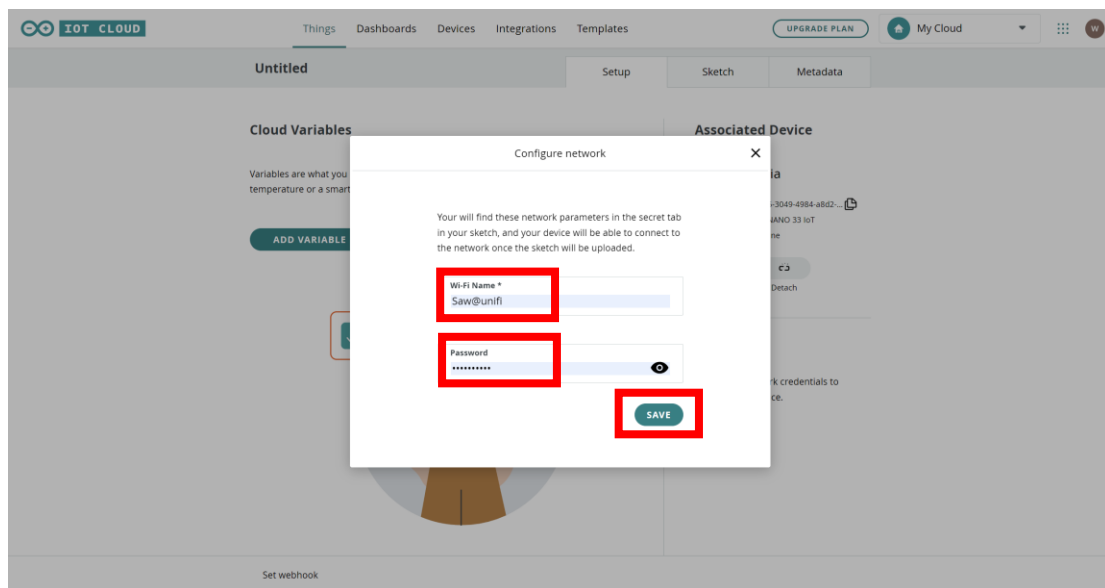
17. Select the device you want to associate with the thing.



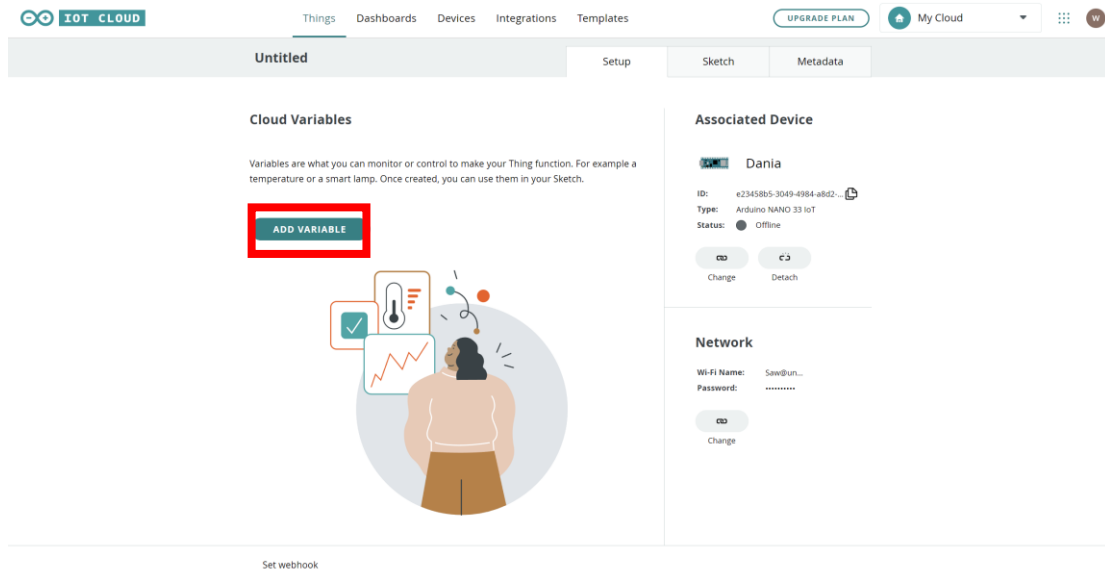
18. Press the “ASSOCIATE” button.



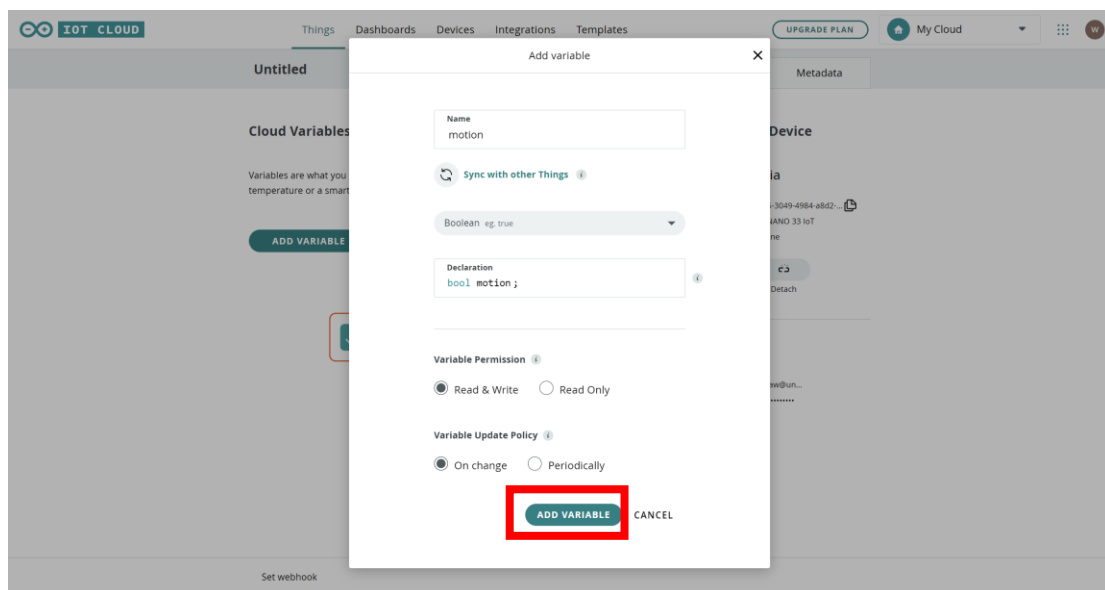
19. Then, in the “Network” section, press  the “ ” button to configure the Wi-Fi network to the device.



20. Enter the Wi-Fi Name and Password. Then, press "SAVE".



21. You can now add cloud variables to the Thing by clicking the "ADD VARIABLE" button. These variables automatically update in the cloud when their values change or based on configuration settings. An example of cloud variable creation is shown as figure below.




22. Press the “ADD VARIABLE” button to add the variable into the thing.

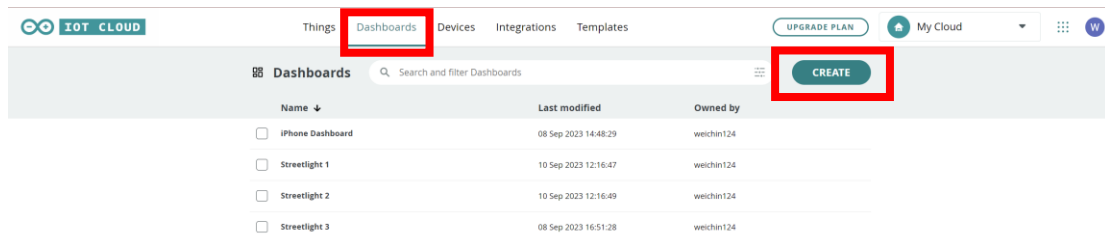
The screenshot shows the Arduino IoT Cloud interface for a device named 'Streetlight 3'. The 'Sketch' tab is highlighted with a red box. Below the tabs, there is a table of 'Cloud Variables' with columns for Name, Last Value, and Last Update. The variables listed include 'advance_Lighting', 'combination_Brightness', 'env_Brightness1', 'env_Brightness2', 'env_Brightness3', 'ldrHalfFunction', 'led', 'led_switch', 'ledHalfFunction', 'motion', and 'motion2'. To the right, the 'Associated Device' section shows 'Nano33IoT' with its ID and status. Below that, the 'Network' section shows Wi-Fi Name and Password.

23. After declaration of all variables, then can proceed to the Arduino Web Editor or press the “Sketch” section to write and upload the code into the microcontroller.

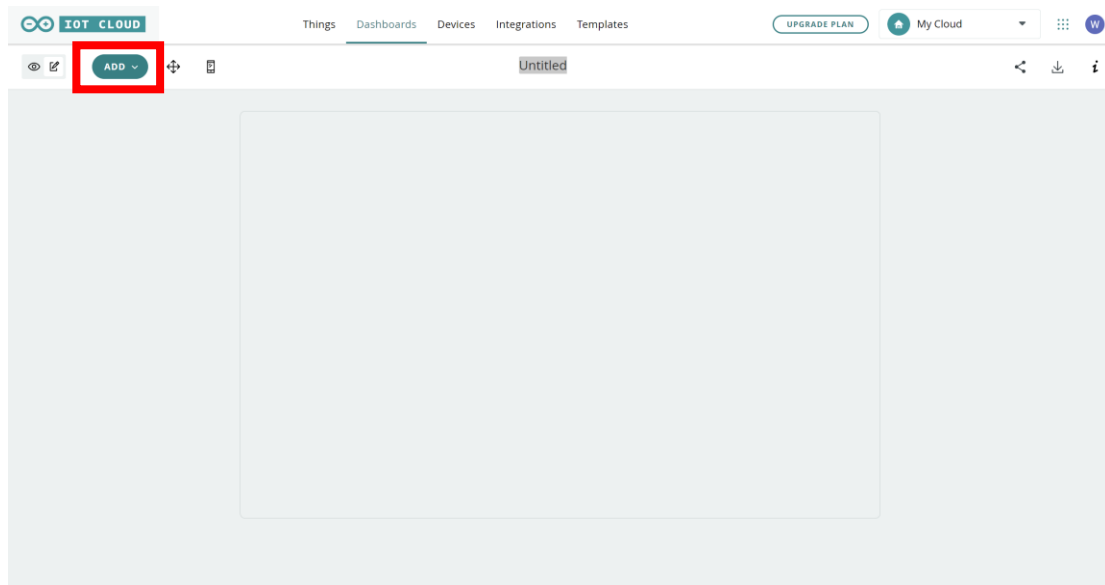
The screenshot shows the Arduino IoT Cloud interface with the 'Sketch' tab selected. The code editor displays C++ code for the Arduino Nano 33 IoT. A red box highlights the 'Upload' button (a right-pointing arrow) in the top right corner of the editor. Below the code editor, a success message reads: 'Success: Done Uploading Streetlight_3_sep08a'. The code includes headers for Adafruit ADS1X15, aFLL, and ThingProperties, and defines variables for combination_Brightness, env_Brightness1-3, led, led_switch, motion, and motion2.

24. After finish coding, press the “” button to upload the code to the microcontroller.

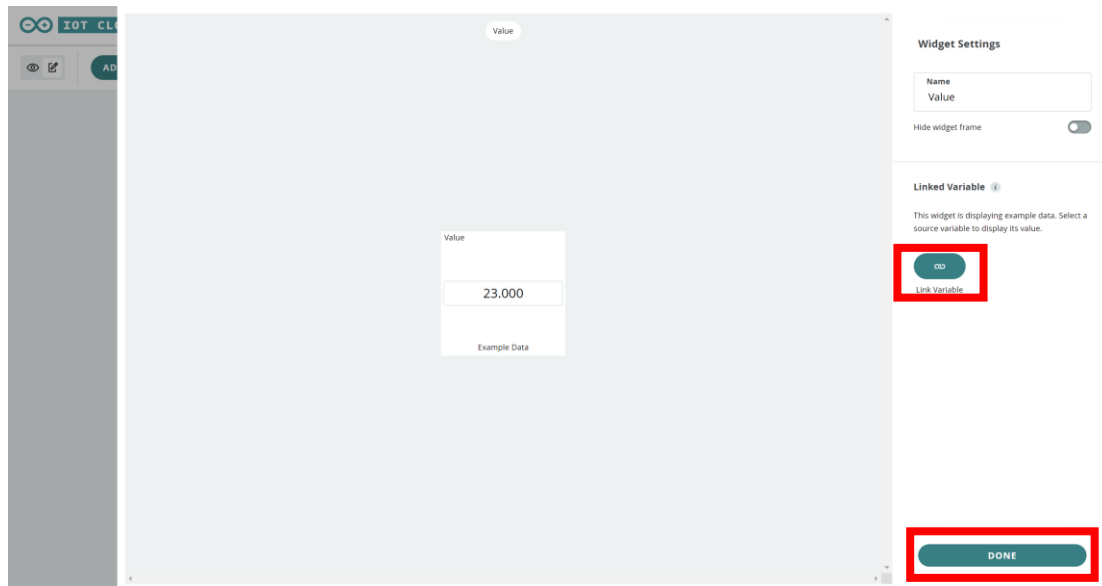
25. After the upload of the code, we can proceed to the create the dashboard. Press the “Dashboard” section to view the list of dashboards.




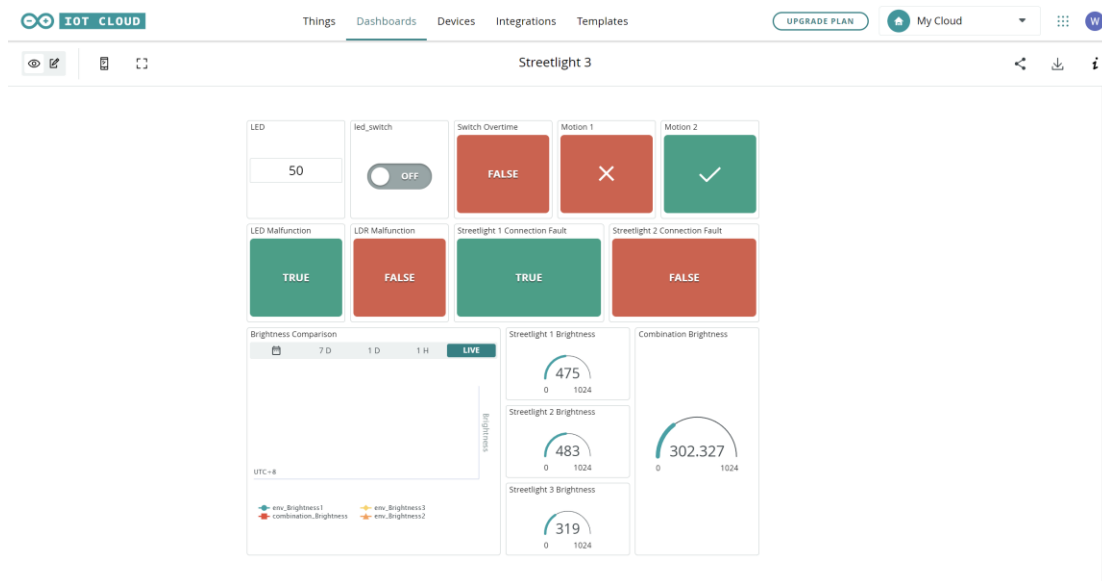
26. Click the "CREATE" button to create a new dashboard.



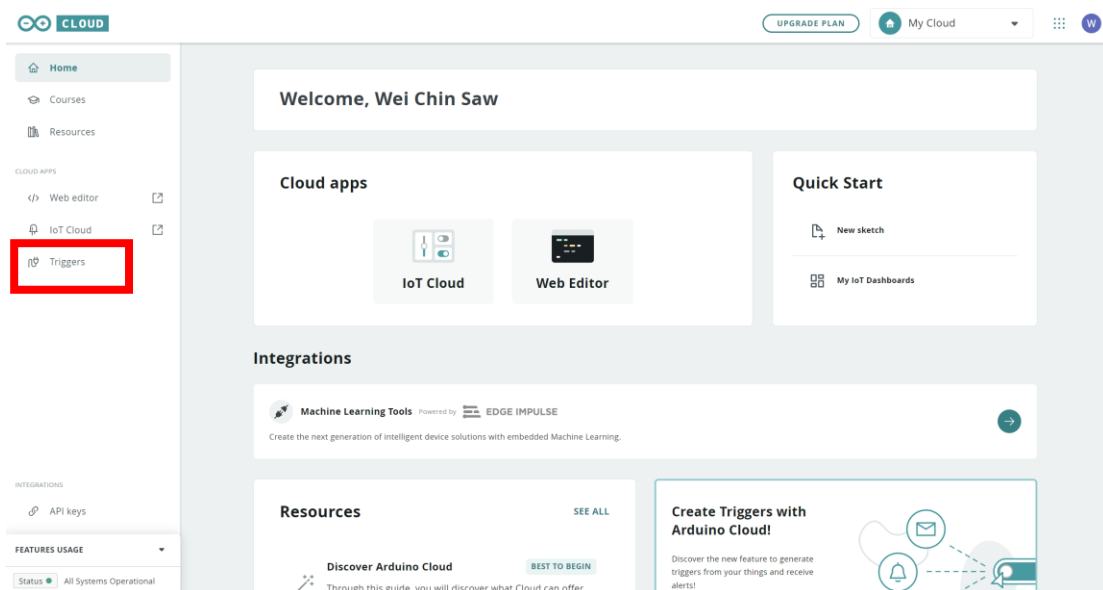
27. To add components to the dashboard, click the "ADD" button.



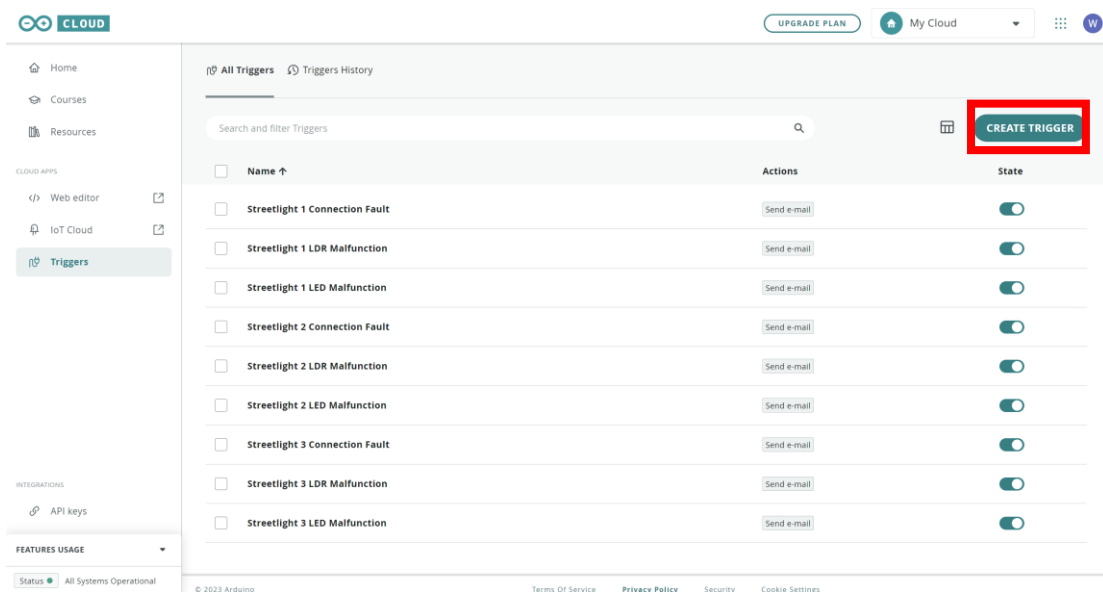
28. Name the component's name in the name section and link the component to the cloud variable by pressing “  ” button. After declaration, press the done button.



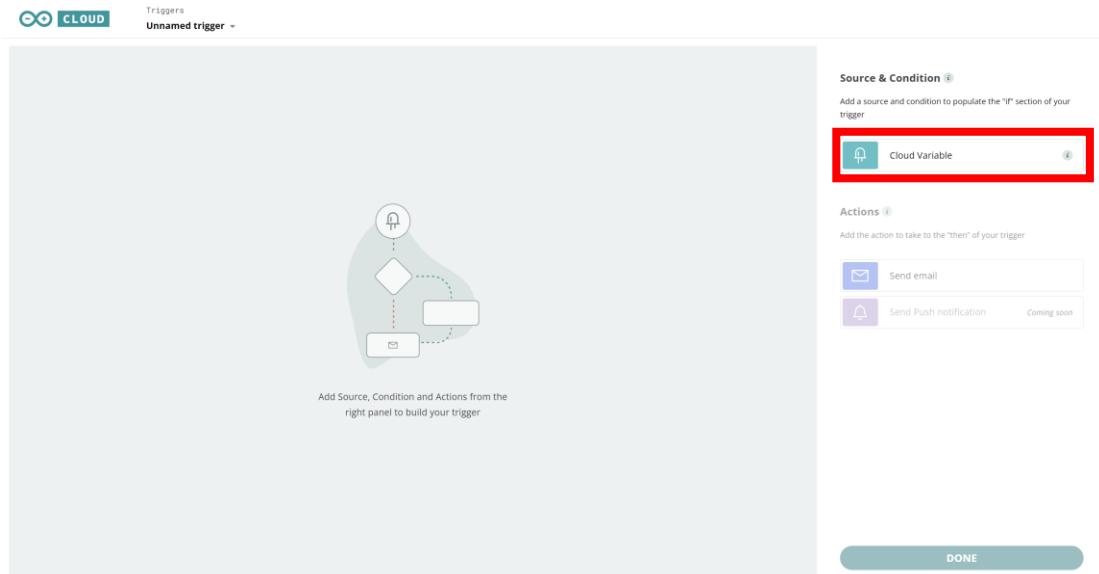
29. After the creation of the dashboard, we need to add the cloud trigger which is an application that act as an alarm to notify the user when incident occurred. Go to the Arduino Cloud dashboard and press the “Triggers” button.



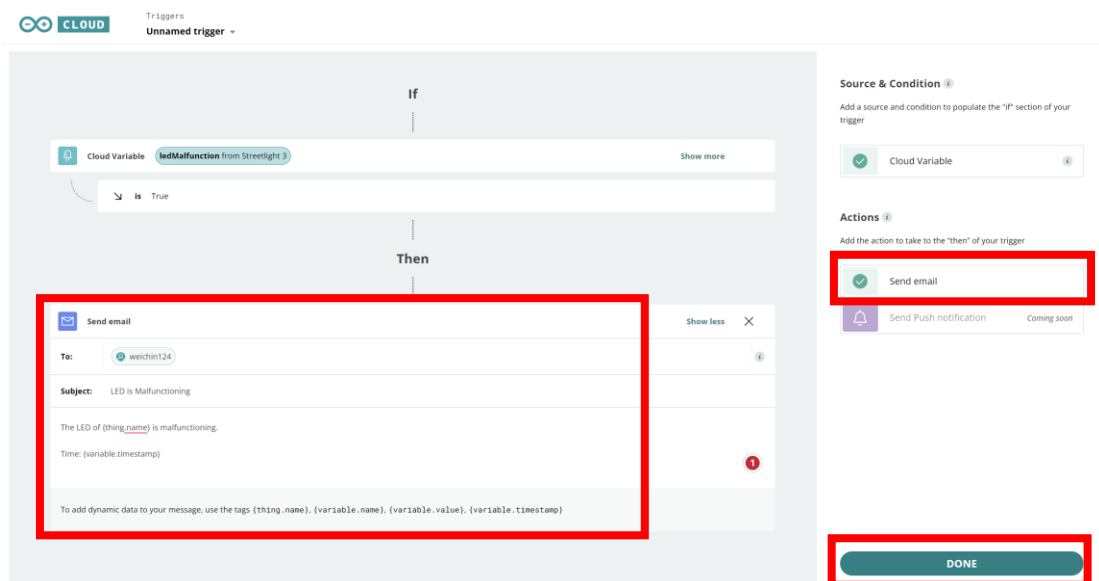
30. Press “CREATE TIGGER” button to create the trigger function.



31. Choose the cloud variable that used to trigger the alarm by pressing the “Cloud Variable” button and choose the action to define the action of the cloud when the alarm is triggered.



32. Write the email template and then press “Done” button. Then, the notify function will be done.



Source Code

```
// Adafruit ADS1X15 - Version: 2.4.0
#include <Adafruit_ADS1X15.h>
// eFLL - Version: Latest
#include <Fuzzy.h>
#include "thingProperties.h"
#include <Wire.h>
/*
  Sketch generated by the Arduino IoT Cloud Thing "Untitled"
  https://create.arduino.cc/cloud/things/04720cb3-5a35-499d-945c-fcbe110044fd

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when changes
  are made to the Thing

  float combination_Brightness;
  int env_Brightness1;
  int env_Brightness2;
  int env_Brightness3;
  int led;
  bool advance_Lighting;
  bool ldrMalfunction;
  bool ledMalfunction;
  bool led_switch;
  bool motion;
  bool motion2;
  bool node1CFNotify;
  bool node1ConnectionFault;
  bool node2CFNotify;
  bool node2ConnectionFault;
  bool switch_overtime;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have
  functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this
  sketch.
*/
Adafruit_ADS1115 ads; /* Use this for the 16-bit version */

// Fuzzy
Fuzzy *fuzzy = new Fuzzy();
// Brightness fuzzy
Fuzzy *bFuzzy = new Fuzzy();

// FuzzyInput: Motion1
```

```

FuzzySet *detected1 = new FuzzySet(1,1,1,1);
FuzzySet *not_detected1 = new FuzzySet(0,0,0,0);

// FuzzyInput: Motion2
FuzzySet *detected2 = new FuzzySet(1,1,1,1);
FuzzySet *not_detected2 = new FuzzySet(0,0,0,0);

// FuzzyInput: pre-lighting
FuzzySet *ready_for_prelighting = new FuzzySet(1,1,1,1);
FuzzySet *no_prelighting = new FuzzySet(0,0,0,0);

// FuzzyInput : Light_E1
FuzzySet *low_lightE1 = new FuzzySet(0, 0, 350, 450);
FuzzySet *mid_lightE1 = new FuzzySet(350, 450, 450, 550);
FuzzySet *high_lightE1 = new FuzzySet(450, 650, 1023, 1023);

// FuzzyInput : Light_E2
FuzzySet *low_lightE2 = new FuzzySet(0, 0, 350, 450);
FuzzySet *mid_lightE2 = new FuzzySet(350, 450, 450, 550);
FuzzySet *high_lightE2 = new FuzzySet(450, 650, 1023, 1023);

// FuzzyInput : Light_E3
FuzzySet *low_lightE3 = new FuzzySet(0, 0, 350, 450);
FuzzySet *mid_lightE3 = new FuzzySet(350, 450, 450, 550);
FuzzySet *high_lightE3 = new FuzzySet(450, 650, 1023, 1023);

// FuzzyInput : Light_Combination
FuzzySet *low_lightC = new FuzzySet(0, 0, 350, 450);
FuzzySet *mid_lightC = new FuzzySet(350, 450, 450, 550);
FuzzySet *high_lightC = new FuzzySet(450, 650, 1023, 1023);

// FuzzyOutput: Light
FuzzySet *minimum_LEDlight = new FuzzySet(0, 0, 35, 75);
FuzzySet *average_LEDlight = new FuzzySet(50, 100, 100, 150);
FuzzySet *maximum_LEDlight = new FuzzySet(100, 175, 255, 255);

// FuzzyOutput: environment
FuzzySet *minimum_environment = new FuzzySet(0, 0, 350, 450);
FuzzySet *average_environment = new FuzzySet(350, 450, 450, 550);
FuzzySet *maximum_environment = new FuzzySet(450, 650, 1023, 1023);

// Variable Declaration
unsigned long currentMillis;
unsigned long prelightingMillis = millis();
unsigned long startUpdateMillis = millis();
unsigned long switchMillis = millis();
unsigned long ledMNotify = millis();
int ledMNTimes = 0;

```

```

unsigned long ldrMNotify = millis();
int ldrMNTimes = 0;
unsigned long envB1Update = millis(); // env_Brightness update Time
unsigned long envB2Update = millis(); // env_Brightness2 update Time
unsigned long node1CFNTime; // NodeMCU1 Connection Fault Notify Time
unsigned long node2CFNTime; // NodeMCU2 Connection Fault Notify Time
const unsigned long period = 2000;
const int ledPin = 2;
const int motion3 = 5;
const int motion4 = 7;
bool pre_Lighting;

void setup() {
    // Initialize serial and wait for port to open:
    Serial.begin(9600);
    // This delay gives the chance to wait for a Serial Monitor without blocking
if none is found
    delay(1500);
    // The ADC input range (or gain) can be changed via the following
    // functions, but be careful never to exceed VDD +0.3V max, or to
    // exceed the upper and lower limits if you adjust the input range!
    // Setting these values incorrectly may destroy your ADC!
    //
DS1115 ADS1015 A
    //
    -----
    // ads.setGain(GAIN_TWOTHIRDS); // 2/3x gain +/- 6.144V 1 bit =
3mV 0.1875mV (default) // activate this if you are using a 5V sensor,
this one should be used with Arduino boards
    ads.setGain(GAIN_ONE); // 1x gain +/- 4.096V 1 bit =
2mV 0.125mV // As the sensor is powered up using 3.3V, this
one should be used with 3.3v controller boards
    // ads.setGain(GAIN_TWO); // 2x gain +/- 2.048V 1 bit =
1mV 0.0625mV
    // ads.setGain(GAIN_FOUR); // 4x gain +/- 1.024V 1 bit =
0.5mV 0.03125mV
    // ads.setGain(GAIN_EIGHT); // 8x gain +/- 0.512V 1 bit =
0.25mV 0.015625mV
    // ads.setGain(GAIN_SIXTEEN); // 16x gain +/- 0.256V 1 bit =
0.125mV 0.0078125mV
    // initialize LED as an output
    pinMode(ledPin, OUTPUT);
    // initialize motion sensor as an input
    pinMode(motion3, INPUT);
    pinMode(motion4, INPUT);

    // Defined in thingProperties.h
    initProperties();
}

```

```

// initialize fuzzy input for lightSensor(e)_1
FuzzyInput *lightSurround1 = new FuzzyInput(5);
lightSurround1->addFuzzySet(low_lightE1);
lightSurround1->addFuzzySet(mid_lightE1);
lightSurround1->addFuzzySet(high_lightE1);
bFuzzy->addFuzzyInput(lightSurround1);

// initialize fuzzy input for lightSensor(e)_2
FuzzyInput *lightSurround2 = new FuzzyInput(6);
lightSurround2->addFuzzySet(low_lightE2);
lightSurround2->addFuzzySet(mid_lightE2);
lightSurround2->addFuzzySet(high_lightE2);
bFuzzy->addFuzzyInput(lightSurround2);

// initialize fuzzy input for lightSensor(e)_3
FuzzyInput *lightSurround3 = new FuzzyInput(7);
lightSurround3->addFuzzySet(low_lightE3);
lightSurround3->addFuzzySet(mid_lightE3);
lightSurround3->addFuzzySet(high_lightE3);
bFuzzy->addFuzzyInput(lightSurround3);

// initialize fuzzy input for motionSensor1
FuzzyInput *motion1 = new FuzzyInput(8);
motion1->addFuzzySet(detected1);
motion1->addFuzzySet(not_detected1);
fuzzy->addFuzzyInput(motion1);

// initialize fuzzy input for motionSensor2
FuzzyInput *motion2 = new FuzzyInput(9);
motion2->addFuzzySet(detected2);
motion2->addFuzzySet(not_detected2);
fuzzy->addFuzzyInput(motion2);

// initialize fuzzy input for advancelighting
FuzzyInput *prelighting = new FuzzyInput(10);
prelighting->addFuzzySet(ready_for_prelighting);
prelighting->addFuzzySet(no_prelighting);
fuzzy->addFuzzyInput(prelighting);

// initialize fuzzy input for lightSensor(combination)
FuzzyInput *lightSurround = new FuzzyInput(11);
lightSurround->addFuzzySet(low_lightC);
lightSurround->addFuzzySet(mid_lightC);
lightSurround->addFuzzySet(high_lightC);
fuzzy->addFuzzyInput(lightSurround);

// initialize fuzzy output for LED1

```

```

FuzzyOutput *LEDoutput1 = new FuzzyOutput(1);
LEDoutput1->addFuzzySet(minimum_LEDlight);
LEDoutput1->addFuzzySet(average_LEDlight);
LEDoutput1->addFuzzySet(maximum_LEDlight);
fuzzy->addFuzzyOutput(LEDoutput1);

// initialize fuzzy output for environmentOut
FuzzyOutput *environmentOut = new FuzzyOutput(2);
environmentOut->addFuzzySet(minimum_environment);
environmentOut->addFuzzySet(average_environment);
environmentOut->addFuzzySet(maximum_environment);
bFuzzy->addFuzzyOutput(environmentOut);

//Fuzzy Rule Antecedent for Environment Brightness
FuzzyRuleAntecedent *ifLS1LOWorLS2LOW = new FuzzyRuleAntecedent();
ifLS1LOWorLS2LOW->joinWithOR(low_lightE1, low_lightE2);

FuzzyRuleAntecedent *ifLS1LOWorLS3LOW = new FuzzyRuleAntecedent();
ifLS1LOWorLS3LOW->joinWithOR(low_lightE1, low_lightE3);

FuzzyRuleAntecedent *ifLS2LOWorLS3LOW = new FuzzyRuleAntecedent();
ifLS2LOWorLS3LOW->joinWithOR(low_lightE2, low_lightE3);

FuzzyRuleAntecedent *ifLS1LOWorLS2LOWorLS3LOW = new FuzzyRuleAntecedent();
ifLS1LOWorLS2LOWorLS3LOW->joinWithOR(ifLS1LOWorLS2LOW, low_lightE3);

FuzzyRuleAntecedent *ifLS1AVERAGEorLS2AVERAGE = new FuzzyRuleAntecedent();
ifLS1AVERAGEorLS2AVERAGE->joinWithOR(mid_lightE1, mid_lightE2);

FuzzyRuleAntecedent *ifLS1AVERAGEorLS3AVERAGE = new FuzzyRuleAntecedent();
ifLS1AVERAGEorLS3AVERAGE->joinWithOR(mid_lightE1, mid_lightE3);

FuzzyRuleAntecedent *ifLS2AVERAGEorLS3AVERAGE = new FuzzyRuleAntecedent();
ifLS2AVERAGEorLS3AVERAGE->joinWithOR(mid_lightE2, mid_lightE3);

FuzzyRuleAntecedent *ifLS1AVERAGEorLS2AVERAGEorLS3AVERAGE = new
FuzzyRuleAntecedent();
ifLS1AVERAGEorLS2AVERAGEorLS3AVERAGE->joinWithOR(ifLS1AVERAGEorLS2AVERAGE,
mid_lightE3);

FuzzyRuleAntecedent *ifLS1HIGHandLS2HIGH = new FuzzyRuleAntecedent();
ifLS1HIGHandLS2HIGH->joinWithAND(high_lightE1, high_lightE2);

FuzzyRuleAntecedent *ifLS1HIGHandLS3HIGH = new FuzzyRuleAntecedent();
ifLS1HIGHandLS3HIGH->joinWithAND(high_lightE1, high_lightE3);

FuzzyRuleAntecedent *ifLS2HIGHandLS3HIGH = new FuzzyRuleAntecedent();
ifLS2HIGHandLS3HIGH->joinWithAND(high_lightE2, high_lightE3);

```



```

FuzzyRuleAntecedent *ifLS1HIGHandLS2HIGHandLS3HIGH = new
FuzzyRuleAntecedent();
ifLS1HIGHandLS2HIGHandLS3HIGH->joinWithAND(ifLS1HIGHandLS2HIGH,
high_lightE3);

//Fuzzy Rule Antecedent for Motion
FuzzyRuleAntecedent *M1HIGHorM2HIGH = new FuzzyRuleAntecedent();
M1HIGHorM2HIGH->joinWithOR(detected1, detected2);

FuzzyRuleAntecedent *M1LOWandM2LOW = new FuzzyRuleAntecedent();
M1LOWandM2LOW->joinWithAND(not_detected1, not_detected2);

FuzzyRuleAntecedent *M1HIGHorM2HIGHorPreLighting = new
FuzzyRuleAntecedent();
M1HIGHorM2HIGHorPreLighting->joinWithOR(M1HIGHorM2HIGH,
ready_for_prelighting);

FuzzyRuleAntecedent *M1LOWandM2LOWandNoPreLighting = new
FuzzyRuleAntecedent();
M1LOWandM2LOWandNoPreLighting->joinWithAND(M1LOWandM2LOW, no_prelighting);

//Fuzzy Rule Consequent
FuzzyRuleConsequent *thenLEDlightMAX = new FuzzyRuleConsequent();
thenLEDlightMAX->addOutput(maximum_LEDlight);

FuzzyRuleConsequent *thenLEDlightAVERAGE = new FuzzyRuleConsequent();
thenLEDlightAVERAGE->addOutput(average_LEDlight);

FuzzyRuleConsequent *thenLEDlightLOW = new FuzzyRuleConsequent();
thenLEDlightLOW->addOutput(minimum_LEDlight);

FuzzyRuleConsequent *thenEnvironmentMAX = new FuzzyRuleConsequent();
thenEnvironmentMAX->addOutput(maximum_environment);

FuzzyRuleConsequent *thenEnvironmentAVERAGE = new FuzzyRuleConsequent();
thenEnvironmentAVERAGE->addOutput(average_environment);

FuzzyRuleConsequent *thenEnvironmentLOW = new FuzzyRuleConsequent();
thenEnvironmentLOW->addOutput(minimum_environment);

// Building FuzzyRule 1 for low lightSurr and motion detected
FuzzyRuleAntecedent *ifLSCLOWandM1HIGHorM2HIGHorPreLighting = new
FuzzyRuleAntecedent();
ifLSCLOWandM1HIGHorM2HIGHorPreLighting->joinWithAND(low_lightC,
M1HIGHorM2HIGHorPreLighting);

```

```

FuzzyRule *fuzzyRule1 = new FuzzyRule(1,
ifLSCLOWandM1HIGHorM2HIGHorPreLighting, thenLEDlightMAX);
fuzzy->addFuzzyRule(fuzzyRule1);

// Building FuzzyRule 2 for low lightSurr and no motion detected
FuzzyRuleAntecedent *ifLSCLOWandM1LOWandM2LOWandNoPreLighting = new
FuzzyRuleAntecedent();
ifLSCLOWandM1LOWandM2LOWandNoPreLighting->joinWithAND(low_lightC,
M1LOWandM2LOWandNoPreLighting);

FuzzyRule *fuzzyRule2 = new FuzzyRule(2,
ifLSCLOWandM1LOWandM2LOWandNoPreLighting, thenLEDlightAVERAGE);
fuzzy->addFuzzyRule(fuzzyRule2);

// Building FuzzyRule 3 for high lightSurr
FuzzyRuleAntecedent *ifLSCHIGH = new FuzzyRuleAntecedent();
ifLSCHIGH->joinSingle(high_lightC);

FuzzyRule *fuzzyRule3 = new FuzzyRule(3, ifLSCHIGH, thenLEDlightLOW);
fuzzy->addFuzzyRule(fuzzyRule3);

// Building FuzzyRule 4 for average lightSurr and motion detected
FuzzyRuleAntecedent *ifLSCAVERAGEandM1HIGHorM2HIGHorPreLighting = new
FuzzyRuleAntecedent();
ifLSCAVERAGEandM1HIGHorM2HIGHorPreLighting->joinWithAND(mid_lightC,
M1HIGHorM2HIGHorPreLighting);

FuzzyRule *fuzzyRule4 = new FuzzyRule(4,
ifLSCAVERAGEandM1HIGHorM2HIGHorPreLighting, thenLEDlightAVERAGE);
fuzzy->addFuzzyRule(fuzzyRule4);

// Building FuzzyRule 5 for average lightSurr and no motion detected
FuzzyRuleAntecedent *ifLSCAVERAGEandM1LOWandM2LOWandNoPreLighting = new
FuzzyRuleAntecedent();
ifLSCAVERAGEandM1LOWandM2LOWandNoPreLighting->joinWithAND(mid_lightC,
M1LOWandM2LOWandNoPreLighting);

FuzzyRule *fuzzyRule5 = new FuzzyRule(5,
ifLSCAVERAGEandM1LOWandM2LOWandNoPreLighting, thenLEDlightLOW);
fuzzy->addFuzzyRule(fuzzyRule5);

// Building FuzzyRule 6 for environment
FuzzyRule *fuzzyRule6 = new FuzzyRule(6, ifLS1LOWorLS2LOW,
thenEnvironmentLOW);
bFuzzy->addFuzzyRule(fuzzyRule6);

// Building FuzzyRule 7 for environment

```

```
FuzzyRule *fuzzyRule7 = new FuzzyRule(7, ifLS1LOWorLS3LOW,
thenEnvironmentLOW);
bFuzzy->addFuzzyRule(fuzzyRule7);

// Building FuzzyRule 8 for environment
FuzzyRule *fuzzyRule8 = new FuzzyRule(8, ifLS2LOWorLS3LOW,
thenEnvironmentLOW);
bFuzzy->addFuzzyRule(fuzzyRule8);

// Building FuzzyRule 9 for environment
FuzzyRule *fuzzyRule9 = new FuzzyRule(9, ifLS1LOWorLS2LOWorLS3LOW,
thenEnvironmentLOW);
bFuzzy->addFuzzyRule(fuzzyRule9);

// Building FuzzyRule 10 for environment
FuzzyRule *fuzzyRule10 = new FuzzyRule(10, ifLS1AVERAGEorLS2AVERAGE,
thenEnvironmentAVERAGE);
bFuzzy->addFuzzyRule(fuzzyRule10);

// Building FuzzyRule 11 for environment
FuzzyRule *fuzzyRule11 = new FuzzyRule(11, ifLS1AVERAGEorLS3AVERAGE,
thenEnvironmentAVERAGE);
bFuzzy->addFuzzyRule(fuzzyRule11);

// Building FuzzyRule 12 for environment
FuzzyRule *fuzzyRule12 = new FuzzyRule(12, ifLS2AVERAGEorLS3AVERAGE,
thenEnvironmentAVERAGE);
bFuzzy->addFuzzyRule(fuzzyRule12);

// Building FuzzyRule 13 for environment
FuzzyRule *fuzzyRule13 = new FuzzyRule(13,
ifLS1AVERAGEorLS2AVERAGEorLS3AVERAGE, thenEnvironmentAVERAGE);
bFuzzy->addFuzzyRule(fuzzyRule13);

// Building FuzzyRule 14 for environment
FuzzyRule *fuzzyRule14 = new FuzzyRule(14, ifLS1HIGHandLS2HIGH,
thenEnvironmentMAX);
bFuzzy->addFuzzyRule(fuzzyRule14);

// Building FuzzyRule 15 for environment
FuzzyRule *fuzzyRule15 = new FuzzyRule(15, ifLS1HIGHandLS3HIGH,
thenEnvironmentMAX);
bFuzzy->addFuzzyRule(fuzzyRule15);

// Building FuzzyRule 16 for environment
FuzzyRule *fuzzyRule16 = new FuzzyRule(16, ifLS2HIGHandLS3HIGH,
thenEnvironmentMAX);
bFuzzy->addFuzzyRule(fuzzyRule16);
```

```

// Building FuzzyRule 17 for environment
FuzzyRule *fuzzyRule17 = new FuzzyRule(17, ifLS1HIGHandLS2HIGHandLS3HIGH,
thenEnvironmentMAX);
bFuzzy->addFuzzyRule(fuzzyRule17);

// Connect to Arduino IoT Cloud
ArduinoCloud.begin(ArduinoIoTPreferredConnection);

/*
The following function allows you to obtain more information
related to the state of network and IoT Cloud connection and errors
the higher number the more granular information you'll get.
The default is 0 (only errors).
Maximum is 4
*/
setDebugMessageLevel(4);
ArduinoCloud.printDebugInfo();
ads.begin();
}

void loop() {
  ArduinoCloud.update();
  // Your code here
  int16_t adc0, adc1, adc2, adc3; // Declare MUX variable
  int led_brightness1;
  int led_brightness2;
  if(!led_switch){
    currentMillis = millis();
    // Fault Detection
    if(currentMillis - envB1Update >= 1800000){
      node1ConnectionFault = true;
    }
    if(node1ConnectionFault && currentMillis - node1CFNTime >= 1800000){
      node1CFNotify = true;
      node1CFNTime = millis();
    }else{
      node1CFNotify = false;
    }
  }
  if(currentMillis - envB2Update >= 1800000){
    node2ConnectionFault = true;
    node2CFNotify = true;
    node2CFNTime = millis();
  }
  if(node2ConnectionFault && currentMillis - node2CFNTime >= 1800000){
    node2CFNotify = true;
    node2CFNTime = millis();
  }else{

```

```

    node2CFNotify = false;
}
// Read environmental Brightness
adc0 = ads.readADC_SingleEnded(0);
float ldr = ads.computeVolts(adc0);
env_Brightness3 = ldr * 1023 / 3.3;
// Fuzzify environmental brightness
bFuzzy->setInput(5, env_Brightness3);
bFuzzy->setInput(6, env_Brightness1);
bFuzzy->setInput(7, env_Brightness2);
bFuzzy->fuzzify();
float light_combination = bFuzzy->defuzzify(2);
currentMillis = millis();
if(currentMillis - startUpdateMillis >= 5000){ // Avoid data overload due
to frequent change of data
    // Share calculated brightness to others
    combination_Brightness = light_combination;
    startUpdateMillis = millis();
}
// Check advance lighting
if(pre_Lighting == true){
    if(advance_Lighting == true){
        prelightingMillis = millis();
    }
    currentMillis = millis();
    if (currentMillis - prelightingMillis > period){
        pre_Lighting = false;
    }
}
// Check motion
int motionValue3 = digitalRead(motion3);
int motionValue4 = digitalRead(motion4);
delay(50); // Wait for variable to update
motion = motionValue3;
motion2 = motionValue4;
Serial.print("Motion detected 1: ");
Serial.println(motion);
Serial.print("Motion detected 2: ");
Serial.println(motion2);
Serial.print("Environment brightness: ");
Serial.println(combination_Brightness);
// Serial.print("5: ");
// Serial.println(motionValue3);
// Serial.print("7: ");
// Serial.println(motionValue4);

// Fuzzify the light output
fuzzy->setInput(8, motionValue3);

```

```

fuzzy->setInput(9, motionValue4);
fuzzy->setInput(10, pre_Lighting);
fuzzy->setInput(11, light_combination);
fuzzy->fuzzify();
float light_output1 = fuzzy->defuzzify(1);
int brightness_lvl; //brightness level of LED
// Output Refinement
if(light_output1 < 100){
    analogWrite(ledPin,0);
    led = 0;
    brightness_lvl = 1;
} else if(light_output1 >= 100 && light_output1 <= 140){
    analogWrite(ledPin,50);
    led = 50;
    brightness_lvl = 2;
} else{
    analogWrite(ledPin,255);
    led = 255;
    brightness_lvl = 3;
}
Serial.print("LED brightness level: ");
Serial.println(brightness_lvl);
Serial.println(" ");
delay(250);
// Read LED brightness
adc1 = ads.readADC_SingleEnded(1);
adc2 = ads.readADC_SingleEnded(2);
float ldr2 = ads.computeVolts(adc1);
float ldr3 = ads.computeVolts(adc2);
led_brightness1 = ldr2 * 1023 / 3.3;
led_brightness2 = ldr3 * 1023 / 3.3;
int ldr2_b_lvl; // To classify brightness level of collected data from
LDR2
int ldr3_b_lvl; // To classify brightness level of collected data from
LDR3
// Serial.print("LED1: ");
// Serial.println(led_brightness1);
// Serial.print("LED2: ");
// Serial.println(led_brightness2);
// Classify brightness level
if (led_brightness1 <= 250){
    ldr2_b_lvl = 1;
}else if (led_brightness1 <= 450 && led_brightness1 > 250){
    ldr2_b_lvl = 2;
}else{
    ldr2_b_lvl = 3;
}
if (led_brightness2 <= 200){

```

```

    ldr3_b_lv1 = 1;
}else if (led_brightness2 <= 450 && led_brightness2 > 200){
    ldr3_b_lv1 = 2;
}else{
    ldr3_b_lv1 = 3;
}
int LED_b_lv1;
if(ldr2_b_lv1 != ldr3_b_lv1){ // Fault Detection
    // fault occurred
    currentMillis = millis();
    if (currentMillis - ldrMNotify >= 1800000 || ldrMNTimes == 0){
        ldrMalfunction = true;
        ldrMNTimes++;
        ldrMNotify = millis();
    }else{
        ldrMalfunction = false;
    }
    ldr2_b_lv1 < ldr3_b_lv1 ? LED_b_lv1 = ldr2_b_lv1 : LED_b_lv1 =
ldr3_b_lv1; //Fault Tolerance Mechanism
}else{
    LED_b_lv1 = ldr2_b_lv1;
}
if (LED_b_lv1 != brightness_lv1){ // Fault Detection
    // fault occurred
    currentMillis = millis();
    if (currentMillis - ledMNotify >= 1800000 || ledMNTimes == 0){
        ledMalfunction = true;
        ledMNTimes++;
        ledMNotify = millis();
    }
}
else{
    ledMalfunction = false;
}
delay(1000);
}
else{
    analogWrite(ledPin,255);
    led = 255;
    // Read environmental Brightness
    adc0 = ads.readADC_SingleEnded(0);
    float ldr = ads.computeVolts(adc0);
    env_Brightness3 = ldr * 1023 / 3.3;
    // Fuzzify environmental brightness
    bFuzzy->setInput(5, env_Brightness3);
    bFuzzy->setInput(6, env_Brightness1);
    bFuzzy->setInput(7, env_Brightness2);
    bFuzzy->fuzzify();
    float light_combination = bFuzzy->defuzzify(2);

```

```

    currentMillis = millis();
    if(currentMillis - startUpdateMillis >= 5000){ // Avoid data overload due
to frequent change of data
        // Share calculated brightness to others
        combination_Brightness = light_combination;
        startUpdateMillis = millis();
    }
    delay(250);
    //Read LED brightness
    adc1 = ads.readADC_SingleEnded(1);
    adc2 = ads.readADC_SingleEnded(2);
    float ldr2 = ads.computeVolts(adc1);
    float ldr3 = ads.computeVolts(adc2);
    led_brightness1 = ldr2 * 1023 / 3.3;
    led_brightness2 = ldr3 * 1023 / 3.3;
    int ldr2_b_lvl; //to classify brightness level of collected data from LDR2
    int ldr3_b_lvl; //to classify brightness level of collected data from LDR3
    // Classify brightness level
    if (led_brightness1 <= 250){
        ldr2_b_lvl = 1;
    }else if (led_brightness1 <= 450 && led_brightness1 > 250){
        ldr2_b_lvl = 2;
    }else{
        ldr2_b_lvl = 3;
    }
    if (led_brightness2 <= 200){
        ldr3_b_lvl = 1;
    }else if (led_brightness2 <= 450 && led_brightness2 > 200){
        ldr3_b_lvl = 2;
    }else{
        ldr3_b_lvl = 3;
    }
    int LED_b_lvl;
    if(ldr2_b_lvl != ldr3_b_lvl){ // Fault Detection
        // fault occurred
        currentMillis = millis();
        if (currentMillis - ldrMNotify >= 1800000 || ldrMNTimes == 0){
            ldrMalfunction = true;
            ldrMNTimes++;
            ldrMNotify = millis();
        }else{
            ldrMalfunction = false;
        }
        ldr2_b_lvl < ldr3_b_lvl ? LED_b_lvl = ldr2_b_lvl : LED_b_lvl =
ldr3_b_lvl; //Fault Tolerance Mechanism
    }else{
        LED_b_lvl = ldr2_b_lvl;
    }
}

```



```

if (LED_b_lvl != 3){ // Fault Detection
  // fault occurred
  currentMillis = millis();
  if (currentMillis - ledMNotify >= 1800000 || ledMNTimes == 0){
    ledMalfunction = true;
    ledMNTimes++;
    ledMNotify = millis();
  }
}else{
  ledMalfunction = false;
}
currentMillis = millis();
if(currentMillis - switchMillis >= 1800000){
  switch_overtime = true;
  switchMillis = millis();
}
else{
  switch_overtime = false;
}
delay(1000);
}
}

/*
  Since Led is READ_WRITE variable, onLedChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onLedChange() {
  // Add your code here to act upon Led change
}

/*
  Since EnvBrightness1 is READ_WRITE variable, onEnvBrightness1Change() is
  executed every time a new value is received from IoT Cloud.
*/
void onEnvBrightness1Change() {
  // Add your code here to act upon EnvBrightness1 change
  envB2Update = millis();
  node2ConnectionFault = false;
}

/*
  Since EnvBrightness2 is READ_WRITE variable, onEnvBrightness2Change() is
  executed every time a new value is received from IoT Cloud.
*/
void onEnvBrightness2Change() {
  // Add your code here to act upon EnvBrightness2 change
  envB1Update = millis();
}

```

```

    node1ConnectionFault = false;
}

/*
    Since EnvBrightness3 is READ_WRITE variable, onEnvBrightness3Change() is
    executed every time a new value is received from IoT Cloud.
*/
void onEnvBrightness3Change() {
    // Add your code here to act upon EnvBrightness3 change
}

/*
    Since AdvanceLighting is READ_WRITE variable, onAdvanceLightingChange() is
    executed every time a new value is received from IoT Cloud.
*/
void onAdvanceLightingChange() {
    // Add your code here to act upon AdvanceLighting change
    if(advance_lighting == true){
        prelightingMillis = millis(); //save the start time
        pre_lighting = true;
    }
}

/*
    Since CombinationBrightness is READ_WRITE variable,
    onCombinationBrightnessChange() is
    executed every time a new value is received from IoT Cloud.
*/
void onCombinationBrightnessChange() {
    // Add your code here to act upon CombinationBrightness change
}

/*
    Since LedSwitch is READ_WRITE variable, onLedSwitchChange() is
    executed every time a new value is received from IoT Cloud.
*/
void onLedSwitchChange() {
    // Add your code here to act upon LedSwitch change
    switchMillis = millis();
}

/*
    Since SwitchOvertime is READ_WRITE variable, onSwitchOvertimeChange() is
    executed every time a new value is received from IoT Cloud.
*/
void onSwitchOvertimeChange() {
    // Add your code here to act upon SwitchOvertime change
}

```

```
/*
  Since Node1ConnectionFault is READ_WRITE variable,
  onNode1ConnectionFaultChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onNode1ConnectionFaultChange() {
  // Add your code here to act upon Node1ConnectionFault change
}

/*
  Since Node2ConnectionFault is READ_WRITE variable,
  onNode2ConnectionFaultChange() is
  executed every time a new value is received from IoT Cloud.
*/
void onNode2ConnectionFaultChange() {
  // Add your code here to act upon Node2ConnectionFault change
}
```

Plagiarism Check Result

feedback studio | Saw Wei Chin | IoT-Based Intelligent Streetlight System with Fault Tolerant Mechanism using Sensor Fusion

Match Overview
1%

CHAPTER 1 – PROJECT BACKGROUND

1.1 Introduction

Streetlight is an infrastructure that plays a significant role in illuminating the street at night for both rural and urban area. This infrastructure not only helps in enhancing the visibility of streets, but also builds a safer environment for the citizens. However, the streetlight consumes the highest electrical energy, a total electricity consumption of 40% of the cities [1].

An intelligent streetlight system is a type of smart city technology that aims to improve energy efficiency, reduce costs, and enhance safety in urban and rural areas. These systems use sensors and advanced analytics to automatically adjust lighting levels based on the time of day, traffic volume, and other environmental factors.

The intelligent streetlight systems help to reduce the energy consumption and costs of a city. For example, a study by the National Renewable Energy Laboratory (NREL) found that smart street lighting systems can reduce energy use by up to 80% compared to traditional street lighting. Other than that, according to the study of Energy Savines Trust, they found that

1	Lesetja S. Mabunda, Tl. Publication	<1%
2	D. Alvarado. "Wafer con... Publication	<1%
3	R.T. Iqbal, U. Qidwai. "O... Publication	<1%
4	Emmanuel Scorsone, J... Publication	<1%
5	Grace Chinyere Eje, Ish... Publication	<1%
6	A. J. Medland, G. Mulli... Publication	<1%
7	Hitesh Poddar, Rituraj... Publication	<1%
8	P. P. Fathima Dheena, G... Publication	<1%
9	B Prabha. "An IoT Base... Publication	<1%
10	Construction Innovatio... Publication	<1%
11	Fuzzy Hierarchical Mod... Publication	<1%
12	Hong, Sung-Il, Young-S... Publication	<1%

Page: 1 of 64 | Word Count: 15721 | Text-Only Report | High Resolution

Turnitin Originality Report

Document Viewer

Processed on: 13-Sep-2023 23:10 +08
 ID: 2165079878
 Word Count: 15721
 Submitted: 1

Similarity Index	Similarity by Source	
1%	Internet Sources:	N/A
	Publications:	1%
	Student Papers:	N/A

IoT-Based Intelligent Streetlight System with... By Saw Wei Chin

Include quoted | Include bibliography | exclude small matches | mode: quickview (classic) report | print | download

<1% match (Lesetja S. Mabunda, Tiotlollo S. Hlalele. "Design and Construction of a Web-Based Communication Interface for Home Automation", 2023 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (ICABCD), 2023)
[Lesetja S. Mabunda, Tiotlollo S. Hlalele. "Design and Construction of a Web-Based Communication Interface for Home Automation". 2023 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems \(ICABCD\), 2023](#)

<1% match (D. Alvarado. "Wafer contamination and particles resulting from cooling-pad elastomer", 1998 International Conference on Ion Implantation Technology Proceedings (Cat No 98EX144) IIT-98, 1998)
[D. Alvarado. "Wafer contamination and particles resulting from cooling-pad elastomer", 1998 International Conference on Ion Implantation Technology Proceedings \(Cat No 98EX144\) IIT-98, 1998](#)

<1% match (R.T. Iqbal, U. Qidwai. "On the Selection of Fuzzy Classifiers Using AdaBoost", Student Conference On Engineering, Sciences and Technology, 2004)
[R.T. Iqbal, U. Qidwai. "On the Selection of Fuzzy Classifiers Using AdaBoost". Student Conference On Engineering, Sciences and Technology, 2004](#)

<1% match (Emmanuel Scorsone, Jocelyn Boutzen, David Fras, Khasim Cali, Giancarlo Marafioti, Terje Mugaas, Krishna Persaud. "Sniffer: a protein-based advanced VOC sensor for victim search", 2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2022)
[Emmanuel Scorsone, Jocelyn Boutzen, David Fras, Khasim Cali, Giancarlo Marafioti, Terje Mugaas, Krishna Persaud. "Sniffer: a protein-based advanced VOC sensor for victim search". 2022 IEEE International Symposium on Safety, Security, and Rescue Robotics \(SSRR\), 2022](#)

<1% match (Grace Chinyere Eje, Ishaku Prince Abner, Hillary Chijindu Ezeaku. "Green metal price volatility and environmentally sustainable real estate development nexus: A global perspective from post COVID-19 pandemic", Journal of Cleaner Production, 2023)
[Grace Chinyere Eje, Ishaku Prince Abner, Hillary Chijindu Ezeaku. "Green metal price volatility and environmentally sustainable real estate development nexus: A global perspective from post COVID-19 pandemic". Journal of Cleaner Production, 2023](#)

<1% match (A. J. Medland, G. Mullineux. "A decomposition strategy for conceptual design", Journal of Engineering Design, 3/1/2000)
[A. J. Medland, G. Mullineux. "A decomposition strategy for conceptual design". Journal of Engineering Design, 3/1/2000](#)

<1% match (Hitesh Poddar, Rituraj Paul, Sourangsu Mukherjee, Budhaditya Bhattacharyya. "Design of smart bin for smarter cities", 2017 Innovations in Power and Advanced Computing Technologies (I-PACT), 2017)
[Hitesh Poddar, Rituraj Paul, Sourangsu Mukherjee, Budhaditya Bhattacharyya. "Design of smart bin for smarter cities". 2017 Innovations in Power and Advanced Computing Technologies \(I-PACT\), 2017](#)

<1% match (P. P. Fathima Dheena, Greema S. Raj, Gopika Dutt, S. Vinila Jinnu. "IoT based smart street light management system", 2017 IEEE International Conference on Circuits and Systems (ICCS), 2017)
[P. P. Fathima Dheena, Greema S. Raj, Gopika Dutt, S. Vinila Jinnu. "IoT based smart street light management system". 2017 IEEE International Conference on Circuits and Systems \(ICCS\), 2017](#)

<1% match (B Prabha. "An IoT Based Efficient Fire Supervision Monitoring and Alerting System", 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2019)
[B Prabha. "An IoT Based Efficient Fire Supervision Monitoring and Alerting System". 2019 Third International conference on I-SMAC \(IoT in Social, Mobile, Analytics and Cloud\) \(I-SMAC\), 2019](#)

<1% match (publications)
[Construction Innovation: Information, Process, Management, Volume 13, Issue 4 \(2013:09-14\)](#)

<1% match (Fuzzy Hierarchical Model for Risk Assessment, 2013.)
[Fuzzy Hierarchical Model for Risk Assessment, 2013.](#)

Weekly Report

FINAL YEAR PROJECT WEEKLY REPORT*(Project II)*

Trimester, Year: 3,3	Study week no.: 2
Student Name & ID: Saw Wei Chin & 2005406	
Supervisor: Ts Dr Chang Jing Jing	
Project Title: IoT-Based Intelligent Streetlights System with Fault-Tolerant Mechanism using Sensor Fusion	

1. WORK DONE

- Research on what should be done on this FYP 2 and report it to supervisor.
- Identify the content should be discussed in every chapter.

2. WORK TO BE DONE

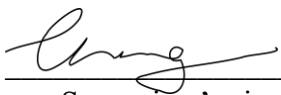
- Create a schedule for this trimester.
- Research on fault tolerant and Internet of Things.
- Research on how to implement sensor fusion and fuzzy logic into the microcontroller.

3. PROBLEMS ENCOUNTERED

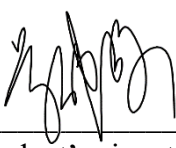
- Limited resources on implementing Fuzzy Logic into microcontroller.

4. SELF EVALUATION OF THE PROGRESS

- The progress is a little bit slow.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3,3	Study week no.: 4
Student Name & ID: Saw Wei Chin & 2005406	
Supervisor: Ts Dr Chang Jing Jing	
Project Title: IoT-Based Intelligent Streetlights System with Fault-Tolerant Mechanism using Sensor Fusion	

1. WORK DONE

- Create a schedule for this trimester.
- Research on fault tolerant and Internet of Things.
- Research on how to implement sensor fusion and fuzzy logic into the microcontroller.

2. WORK TO BE DONE

- Revise chapter 1 to provide a better explanation on the project.
- Research on the Arduino Cloud.
- Draft the system architecture.

3. PROBLEMS ENCOUNTERED

- Problem in figure out how the system architecture looks like.

4. SELF EVALUATION OF THE PROGRESS

- The progress is a little bit slow.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3,3	Study week no.: 6
Student Name & ID: Saw Wei Chin & 2005406	
Supervisor: Ts Dr Chang Jing Jing	
Project Title: IoT-Based Intelligent Streetlights System with Fault-Tolerant Mechanism using Sensor Fusion	

1. WORK DONE

- Revise chapter 1 to provide a better explanation on the project.
- Research on the Arduino Cloud.
- Draft the system architecture.

2. WORK TO BE DONE

- Revise the fuzzy logic algorithm defined in Project 1.
- Revise the membership function of the Fuzzy Rules Algorithm.
- Define the value of the membership function.

3. PROBLEMS ENCOUNTERED

- None

4. SELF EVALUATION OF THE PROGRESS

- The project has caught up with the progress.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3,3	Study week no.: 8
Student Name & ID: Saw Wei Chin & 2005406	
Supervisor: Ts Dr Chang Jing Jing	
Project Title: IoT-Based Intelligent Streetlights System with Fault-Tolerant Mechanism using Sensor Fusion	

1. WORK DONE

- Revise the fuzzy logic algorithm defined in Project 1.
- Revise the membership function of the Fuzzy Rules Algorithm.
- Define the value of the membership function.

2. WORK TO BE DONE

- Additional research on others streetlight system.
- Configure cloud.
- Configure microcontroller.
- Reimplement the prototype.

3. PROBLEMS ENCOUNTERED

- Sensor is insufficient for implement, need to borrow extra sensor from lab.

4. SELF EVALUATION OF THE PROGRESS

- The project is on right path.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3,3	Study week no.: 10
Student Name & ID: Saw Wei Chin & 2005406	
Supervisor: Ts Dr Chang Jing Jing	
Project Title: IoT-Based Intelligent Streetlights System with Fault-Tolerant Mechanism using Sensor Fusion	

1. WORK DONE

- Additional research on others streetlight system.
- Configure cloud.
- Configure microcontroller.
- Reimplement the prototype.

2. WORK TO BE DONE

- Code the system.
- Configure Cloud Alarm.
- Implement another prototype to show the ability of the system in IoT.

3. PROBLEMS ENCOUNTERED

- -

4. SELF EVALUATION OF THE PROGRESS

- Project is on right path.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3,3	Study week no.: 12
Student Name & ID: Saw Wei Chin & 2005406	
Supervisor: Ts Dr Chang Jing Jing	
Project Title: IoT-Based Intelligent Streetlights System with Fault-Tolerant Mechanism using Sensor Fusion	

1. WORK DONE

- Code the system.
- Configure Cloud Alarm.
- Implement another prototype to show the ability of the system in IoT.

2. WORK TO BE DONE

- Finalise FYP 2 report.
- Complete Poster
- FYP 2 submission
- Prepare for presentation.

3. PROBLEMS ENCOUNTERED

- -

4. SELF EVALUATION OF THE PROGRESS

- Project 2 is almost complete.



Supervisor's signature



Student's signature

Form Title: Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



**FACULTY OF INFORMATION AND COMMUNICATION
TECHNOLOGY**

Full Name(s) of Candidate(s)	Saw Wei Chin
ID Number(s)	20ACB05406
Programme / Course	Bachelor of Computer Science (Honours)
Title of Final Year Project	IoT-Based Intelligent Streetlights System with Fault-Tolerant Mechanism using Sensor Fusion

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceed the limits approved by UTAR)
Overall similarity index: <u> 1 </u> % Similarity by source Internet Sources: <u> N/A </u> % Publications: <u> 1 </u> % Student Papers: <u> N/A </u> %	
Number of individual sources listed of more than 3% similarity: <u> 0 </u>	
Parameters of originality required, and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note: Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Signature of Supervisor

Name: Ts Dr Chang Jing Jing

Date: 15/9/2023

Signature of Co-Supervisor

Name: _____

Date: _____

FYP 2 CHECKLIST**UNIVERSITI TUNKU ABDUL RAHMAN****FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)****CHECKLIST FOR FYP2 THESIS SUBMISSION**

Student ID	Saw Wei Chin
Student Name	20ACB05406
Supervisor Name	Ts Dr Chang Jing Jing

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)

Date: 15/9/2023