AUTOMATED STREET LIGHT CONTROL SYSTEM WITH SERVER CONTROL

By

Tajwarul Islam
14221020
Sanjida Mazid Trisha
15321010
Tanjilur Rahman Abid
16121002
Md. Hasibul Hassan Siam
16321045

A thesis submitted to the Department of Electrical and Electronic Engineering of Brac University, in partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Electronic Engineering

Department of Electrical and Electronic Engineering
Brac University
June 2021

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Tajwarul	Trisha
Tajwarul Islam 14221020	**Sanjida Mazid Trisha** 15321010
Abid	Siam
Tanjilur Rahman Abid 16121002	**Md. Hasibul Hassan Siam** 16321045

Approval

The thesis titled “Automated Street Light Controlling Prototype System with Server Control” submitted by

1. Name: Tajwarul Islam
2. Name: Sanjida Mazid Trisha
3. Name: Tanjilur Rahman Abid
4. Name: Md. Hasibul Hassan Siam

of Spring, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of [Bachelor of Science in Electrical and Electronic Engineering] on 9th June, 2021

Examining Committee:

Supervisor:
(Member)

_____

Dr. A. S. Nazmul Huda
Assistant Professor, Dept. of EEE
Brac University

Thesis Coordinator:
(Member)

_____

Dr. Abu S.M. Mohsin
Assistant Professor, Dept. of EEE
Brac University

Departmental Head:
(Chair)

_____

Dr. Md. Mosaddequr Rahman
Professor and Chairperson, Dept. of EEE
Brac University

Abstract

Our manuscripts aim to design an automatic street light system controlled by a microcontroller to be synchronized with web portal to monitor. Street light plays a vital world in urbanized area or even in highways connected to city. While having dedicated manpower to operate street light is overwhelming and also illogical in this modern area. Our thesis project aimed to design a solar powered automatic street light controlling system. An ESP32 based design connected to the server portal allows to get live status of the light connected to the system. On the other hand, this design allows to turn ON/OFF the light connected to the system remotely. This design prototype functions based on the value of light depending resistor attached to the system and the light control and live status of the system can be monitored through the web portal from anywhere.

Keywords: Street light, Automated system, IoT, LDR

Acknowledgement

I would like to express sincere thanks to my thesis supervisor, Dr. A.S. Nazmul Huda, Assistant Professor, Dept. of Electrical & Electronic Engineering (EEE), Brac University, for his supervision to make a successful completion of the thesis. I am also grateful to Brac University for providing me the necessary help for the successful completion of this thesis.

Contents

List of Figures

Chapter 1

Introduction

Chapter 1: Introduction

1.1. Introduction

Increasing rate of road and transportation in any urban areas or as highways are increasing every day due to the urbanization. The fundamental requirements to drive safely on any road is a proper system of street light. Street lightweight may be a raised supply of sunshine that's unremarkably used on walkways and streets once the encompassing turns dark. associate degree economical street lightweight system is that the major demand in today's lifetime of transportation for safety functions and avoiding accidents throughout the night. The development of Bangladesh Road and transport has evolved in so many ways during last decades. The government of Bangladesh has requested the Asian Development bank and the Islamic Development bank for co-financing of up to 300 million for the implementation of the Bangladesh Power Systems efficiency Improvement Project [1]. Among the other project goals, LED street Lighting over 1000 km with solar enabling is the sub project mentioned in the goals of the projects. Despite the modernization of the streets with better solar system, in today's busy life no one is exactly aware about switching street lights off/on when not required. The project introduced here gives a solution to this by eliminating manpower and reducing power consumption. This requires three basic components i.e., LDR, Sensors, microcontroller, and an RTC.

Automatic system is becoming more usual to cover up the monotonous manual system that requires manpower and sometimes costly. Automated system eliminates the necessity of manpowered system while saving more energy than the usual. These automation system makes our daily life easier. On that same note, internet of things (IoT) has become just another part of our life to enable a smart life connected to the internet. IoT enables us connect to the internet and smartly handle any operation that might eliminate the necessity of manpower. This research is based on the server connection established by the microcontroller that updates the current status of the design prototype.

This paper aims at coming up with and capital punishment the advanced development in embedded systems for energy saving of street lights with observance services to alter quicker fault detection. Nowadays, using manpower just to keep the lights on beside a highway is not a suitable decision as humans have become too busy, and are unable to find time even to switch the lights wherever not necessary. The present system illustrates, the street lights will be switched on in the evening before the sun sets by using the light depending resistor that measures the light upon its surface and then the following day, the lights associated to the system will atomically switched off once there is sufficient light on the roads. This thesis gives the best solution for electrical power wastage to enable an automated street light controlling system. Also, the manual operation of the lighting system is completely eliminated. In this paper the two sensors are used which are Light Dependent Resistor LDR sensors to indicate a day/night time. The microcontroller is used as a brain to control the street light system, where the programming language used for developing the software for the microcontroller is C-language. Finally, the system has been successfully designed and implemented as a prototype system.

1.2. Background of the study

The idea of designing a new system for the streetlight that does not consume huge amounts of electricity is suitable to harness power from photovoltaic solar cell panels, and illuminate large areas with the best intensity of sunshine is regarding every engineer operating during this field. Providing street lighting is one among the foremost vital and costly responsibilities of a town. Lighting will account for 10–38% of the full energy bill in typical cities worldwide. Street lighting could be a notably important concern for public authorities in developing countries as a result of its strategic importance for economic and social stability. Inefficient lighting wastes important monetary resources once a year, and poor lighting creates unsafe conditions. Energy-efficient technologies and design mechanisms can reduce the cost of street lighting drastically.

The internet of things, also known as IoT, is a network of interconnected computing devices, mechanical and digital machinery, goods, livestock, and people with unique IDs. The IoT also

stands for the data interaction between human to human or human to computer or computer to computer without any physical interaction [2]. A system consisting of web-based smart devices that has access to the processors or sensors to comply with the server for data exchange is known as an IoT ecosystem. As this system works without human intervention makes it a suitable choice to build an automated system. Connecting the design prototype to the internet makes the system much more suitable to monitor remotely while getting notified about the technical fault in the system. An automated street light system enables the street light alongside the road to switch on automatically onset of the dark weather and turning it off during the light hours.

Nowadays, most of the existing street light systems are wired which are not only difficult to construct but also have poor flexibility. While manual management is vulnerable to errors and ends up in energy wastages and manually dimming throughout mid-night is infeasible, an automatic street light-weight system will certainly work as an appropriate answer to avoid energy waste. Also, dynamically following the sunshine level is manually infeasible. the present trend is that the introduction of automation and remote management solutions to regulate street lighting. There are a unit numerous numbers of management methods and ways in dominant the road light-weight system like style and implementation of CPLD based mostly star power-based street light-weight system, street intensity level management, and road safety module mistreatment embedded system.

Currently, as per the BRTA, there is 21000km of national highway consisting of regionals and Zilla or district roads [3]. The road network capable to carry vehicles has increased significantly and is increasing every year. Only a few years back, there were no proper planning and system for road maintenance and death by accident was a common phenomenon in Bangladesh. This much area is surely in need of proper street light controlling system that should avoid human intervention. In most cases, dark roads are responsible for heinous crimes as road accidents that cost us the most valuable lives of our dearest ones. Several studies proved that ambient light has an impact on human behavior and the crime rate at a darker area is much larger than the area where the streets are covered in light [4]. On the same note, another study continued by (Painter, 1994) illustrated that half of total recorded crime occurs after the dark that explains the necessity of proper lighting around the roads and highways [5].

1.3. Objectives of this Research

As per the necessity of the proper lighting facility around the streets in urban areas and highways, we require a modernized system which eliminates the problems related to manpower, energy waste and expensive material. Our thesis project is prepared to provide a full coverage of street lights on a particular area that can be monitored from a server portal, eliminating the necessity of human resource. Designing an automatic street light system with server control will effectively enables us to monitor the live status of the lights associated to the system. This research projects are also can be powered through a solar source to avoid disruption of electricity at night. So, this project demonstration and design implementation represents a developed way of implementing automatic street light system with server control to access remote monitoring.

1.4. Research Methodology

This research project is implemented on a prototype design with the help of desired electric components as LDR, ESP32, 12V solar panel, and other necessary components. A proper schematic diagram is drawn at first to find the best way of implementing the project. Later the hardware implementation and synchronized server control operation define the actual process of this research project. During the research of design prototype of automatic solar-powered street light system, several pieces of literature based on this criterion are properly analyzed and utilized to build the current form of design implementation. The following research is based on the embedded language C or wiring language and a private server website adafruit.io has been used to design the server control portal.

1.5. Organization of the Thesis

This thesis work is divided into four sections titled Chapter 1: Introduction, Chapter 2: Literature review, Chapter 3: Hardware Components, Chapter 4: Working Principles, and Chapter 5: Results and Discussion. The first chapter introduces us to the common phenomenon related to the street light system and its urgency. It also describes the background of this study and the objectives of this research. The second chapter is a literature review that was based on the initial study material to gather the existing knowledge in this sector. Consequently, the third

chapter is containing information about the required hardware components related to the project and the fourth chapter describes the working principles with a proper circuit diagram and the block diagram of the whole system. The last chapter concluded the study with proper results and discussion that we have founds throughout this research. The section titled Appendix A holds the code, we have written for the microcontroller to interact data from the server.

Chapter 2

Literature Review

Chapter 2: Literature Review

2.1. Introduction

The current automated era that we are living in is surely eliminating the necessity of human resources to make our life easier and cost-effective. IoT-based structures are being implemented everywhere to enable a smart and modernized lifestyle. Implementation of IoT-based projects has made it easier to make any physical implementation be monitored from the web version of it. Internet of things describes the function of the hardware system working principle while supplying the performance of hardware implementation to supply chain and logistics operations.

Due to the combination of IoT and a smart monitoring system for controlling street lights, it is quite remarkable to implement in the highways and city roads. Street lights are a major component of city roads to avoid crimes and accidents during the night. Despite that, no one in society bothers to turn on a light on the street at the right moment or to turn it off. To develop such a structure IoT is a must to implement successfully. Building a cost-effective, practical, ecofriendly, and safest way of controlling the street light system is to connect the hardware implementation to the IoT servers to take back the program online.

Implementation of IoT based street light system requires several components to operate perfectly in terms of accepting commands manually or by the server. Server implementation requires proper designing and pre-declared value of each component to control the hardware operation.

2.2. Arduino Based Streetlights Controlling System

An automated system to control streetlight surely requires a microcontroller. Using Arduino as a microcontroller allows to use several additional external features to attach to the system. A

streetlight controlling system with object detection will find its true purpose on the road. (Z. Mumtaz, S. Ullah, S. Ilyas, S. Liu, N. Aslam, J. Arshad, H.A. Madni, 2018) developed a system which will lead to energy conversion by object detection and a counter associated to the project will count the objects passing through the road. The main object of the study is to reduce energy waste along with increasing the lifetime of street lights through a microcontroller controlling system. The proposed system by (Z.Mumtaz,S.Ullah, S.Ilyas, S. Liu, N. Aslam, J. Arshad, H.A.Madni, 2018) demonstrated an Arduino controlled project synchronized to the LDR that automatically control street light along with the object counter that counts the car or other vehicles passing through the road. Object dependent automation system of their project is another addition to the automatic street light system that will turn off the light automatically when there is no object passing through the road. This proposed design of street light controlling system is cost effective and a safest possible way of reducing power consumption [6**].** As per the author this system can be upgraded by replacing conventional LED modules with solar based LED modules.

2.3. GSM Technology to Control Street Light

(R. Zambare, P. Pawar, P. Jadhav, P. Patil, S. Mule, 2020) described about a street light controller system with GSM technology that aims to send a coded SMS to the light relay system to base statin controller to conduct required automation operations. The main objective of this study to build a cost-effective process of street light controlling system with automated short message system. As per the author, the developed system is cost-effective with lower power consumption, low operating cost and environment friendly. (R. Zambare, P. Pawar, P. Jadhav, P. Patil, S. Mule, 2020) mentioned that the industry of street lighting systems is growing rapidly and to control and maintain complex street light system should not involve direct intervention of human being. The proposed system by (R. Zambare, P. Pawar, P. Jadhav, P. Patil, S. Mule, 2020) is designed in such a way that the individual street light can be controlled separately by encoded text messages. This study proposed several conditions as the solar powered sources will only be used when there is the absence of the supply from the direct gridline. They developed the system with the help of light depending resistor that measures the light falling on its surface. The author addressed the GSM technology as the main hub of their developed

system. On this project, the GSM module get the information and send the same to the main server [7]. Recording or intimidating of the information is controlled by the microcontroller. As per the author, the system is able to identify faulty situation if there is any damage (short circuit) or if any bulb/tube does not work, it could point out the location and could provide an alarm.

2.4. Saving Energy through Automatic Street Light Controlling

(B. K. Subramanyam1 et al, 2013) worked on intelligent wireless street light control and monitoring system, which ease of human-based maintenance system and operates on low energy. In this system, the author has used a solar panel at the lamp post which enables the system to be powered by the solar system and the LDR enables the controlling operation of the system. Through this method, it is possible to save some more power and energy, and also monitoring and controlling the street lights using GUI application, enables to get real-time status of the lights in street or highway lighting systems [8].

Saad, Mustafa &Farij, Abdalhalim& Salah, Ahamed &Abdaljalil, Abdalroof. (2013). Automatic Street Light Control System Using Microcontroller. (M.F.Saad, S. Abdalhalim, Abdalroof A., 2013) proposed an advanced system using embedded systems for energy-saving street lights. This system developed an intelligent way of switching on and off the light automatically at the noon and later in the morning. In this system, the (M.F. Saad, S. Abdalhalim, Abdalroof A., 2013) demonstrates that the LDR value varies according to the amount of light falling on its surface and that is being used as the switch for the system to work into [9]. On the other hand, the author has also described the photoelectric cell that is activated only at night which eventually turns ON a particular light of the system. This system has its own drawback as the system at night only remains ON when there is a movement on the road but the best feature of this study is to save energy.

Chapter 3

Hardware and Software Description

Chapter 3: Hardware and Software Description

3.1. Required Hardware

This chapter is organized with the associated components of the designed system. On this following research we used several components to build the design prototype. ESP 32 which is the main board of the system to send command and receive data output. Information related to light depending resistor and the purpose of relay is described in this section to understand the proper implementation of the system. An RTC as DS3231 and its features to choose it for the hardware implementation is also described in this chapter. Each component with its features and actual outlook is described here to understand the necessity of these components.

3.1.1. ESP32

ESP32 is a low-cost, low-power system on a (SoC) series chip developed by "Espressif Systems". This microprocessor has built in Wi-Fi & dual-mode Bluetooth capabilities. Among the other variations of the ESP family, ESP32 has a dual-core or single-core Tensilica Extensa LX6 microprocessor with a clock rate of up to 240 MHz at its heart. ESP32 is highly integrated with built-in antenna switches, RF balun and power amplifier. Advantages like low-noise receive amplifier implication in a tiny chip strikes it farther to become a suitable choice, filters, and power management modules. Engineered for mobile devices, wearable electronics, and IoT applications, ESP32 achieves ultra-low power consumption through power saving features including fine resolution clock gating, multiple power modes, and dynamic power scaling.

3.1.1.1. Memory of ESP32

ESP32 has a Tensilica Xtensa 32-bit LX6 microprocessor with 2 cores capable of generating 240MHz clock frequency. This chip has an ultra-low power co process which allows analog to digital conversion computation. Wireless connectivity of these chip offers up to 150 Mbit/s through WIFI. Built in Bluetooth operation makes it even better as a standalone system. ESP 32 has 448 KiB of ROM and SRAM of 520 KiB which operates for booting and core functions.

ESP32 is superior due to its all-in-one data processing, interfacing sensors and digital input output. The official software development framework for this chip is ESP-IDF. The chip supports 4 x 16 MBytes of external QSPI flash and SRAM with hardware encryption based on AES ("Overview of ESP32 features [10].

.

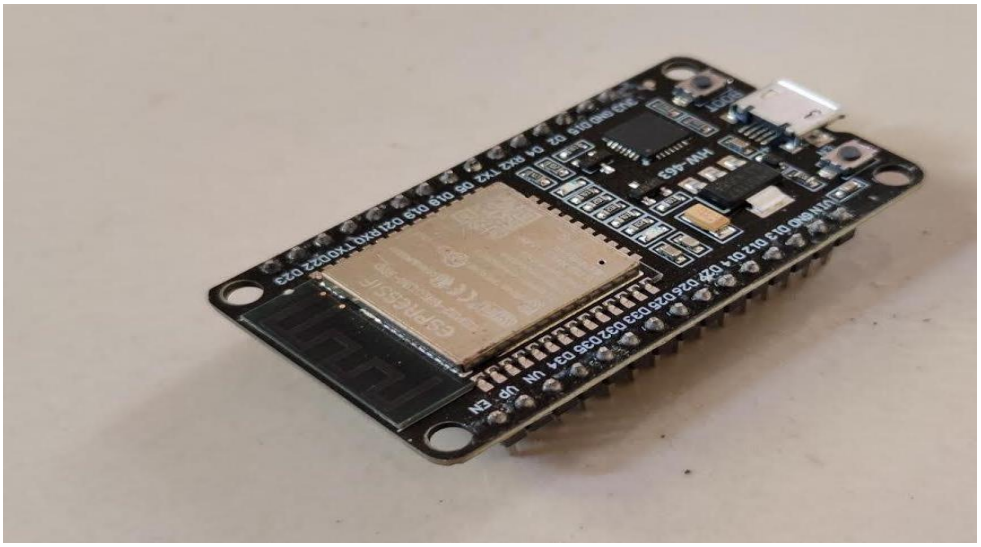


Figure 1:ESP32 chip outlook

Peripheral input/output: Rich peripheral interface like time, real time clock, touch sensor, SPI (Serial Peripheral Interface), I²C (Inter-Integrated Circuit) interface, I²S interface, Infrared remote controller pulse width modulation, hardware accelerator has combinedly make this chip a suitable one for any IoT based project.

3.1.2. LDR

LDR stands for light depending resistors also known as photoresistors often used in electric circuits to measure the level of light. This type of photoresistor is conveniently used in circuit designs to indicate the resistance for changes in light level. A light depending resistor is an electrical component that is sensitive to light. Based on the emittance of light on its surface, the LDR measures the level of it and changes the resistance value accordingly. The sensitivity of the light depending resistor varies with the wavelength of the incident light. This

photoresistors are made of semiconductor materials that enables their light sensitive properties. In most cases, cadmium sulphide (CdS) is the most popular material to make a light depending resistor though several materials can be used to make such light depending resistor. The basic difference between a phototransistor and a light dependent resistor is that an LDR is a passive device without any PN junction. LDR response to a light within a milliseconds once total darkness is applied but during the absence of light it takes up to a second or more to reach up to the final resistance level.

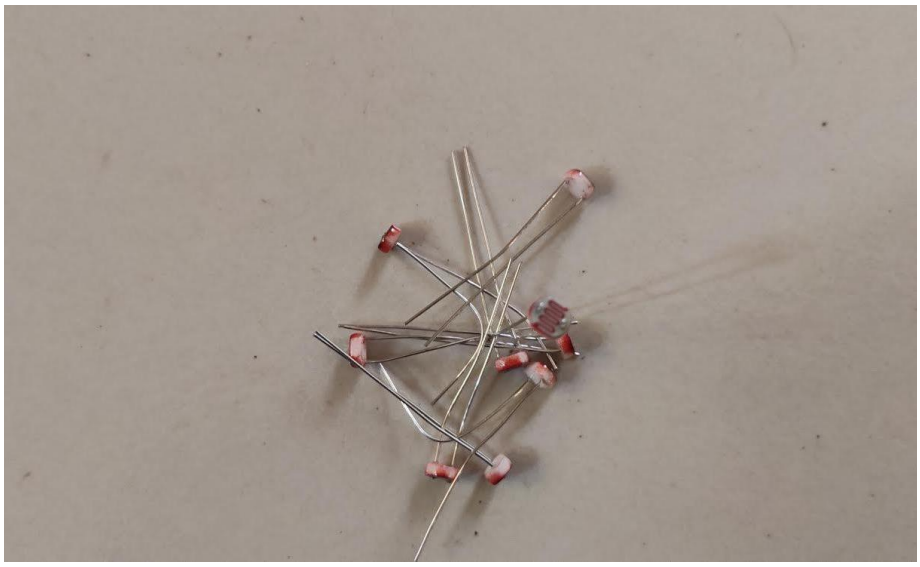


Figure 2: Light Depending resistor

3.1.2.1. Buck Converter

Buck converter is used as a DC-DC converted that converts the high voltage to lower voltage. Sometimes the electrical component requires lower voltage to operate perfectly. To operate in such situation a buck converter is required. Buck converter extends the battery life with controlled power supply. For example, if we need a power supply of 5 volts to power a microcontroller or a system that requires 5 volts to operate. But the available supply rail is of 12 volts. Then in such cases we can use a bulk converter to power our system which will supply the regulated voltage to the system.

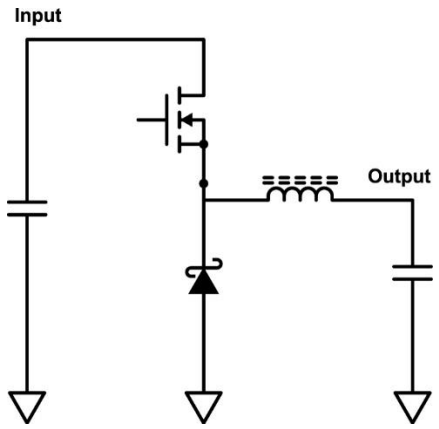


Figure 3: Circuit Diagram of Buck Converter

3.1.3. RTC

A real-time clock (RTC) is an integrated circuit that keeps an updated track of the current time. To be precise, RTC counts hours, minutes, seconds, months, days and even years with facilities like automatically adjusted for months with fewer than 31 days or even a leap year. To keep tracking the time even when the system power fails, RTCs are designed to operate and consuming ultra-lower power to operate is one of its best features. RTCs are used in personal computers, embedded systems and servers, electronic devices requiring accurate timekeeping to operate. This enables them to maintain current time against an absolute time reference, usually set by the microprocessor directly. They are present in everything from the instrument clusters and infotainment systems in automotive applications to house metering. RTCs frequently integrate into other devices—for example, the broadband communications ICs used in car radios

Maintaining the clock of an RTC is done by counting its clock by counting the cycles of an oscillator – usually an external 32.768kHz crystal oscillator circuit, capacitor-based oscillator or an embedded quartz crystal. Different kind of several RTCs can identify transitions and measure the periodic time of any input signal to measure the time.

The connected processor obtains an updated ‘system time’ in some way and writes this new value to the RTC for it to start counting from. This system time could come from manual input from a user interface, reading a GPS unit or from a cloud connection.

Such a DS1302 is a real-time clock having 31 bytes of static memory operated through the tickle-change timekeeping chip. The simple serial interface helps it to communicate simultaneously with a microcontroller. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. Data can be transferred by 1 byte or 31 bytes in a burst mode to and from the clock/ram: CE, input-output data, serial clock.

3.1.3.1. DS3231

DS3231 is a real-time clock having 31 bytes of static memory operated through the tickle-change timekeeping chip. The simple serial interface helps it to communicate simultaneously with a microcontroller. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. Data can be transferred by 1 byte or 31 bytes in a burst mode to and from the clock/ram: CE, input-output data, serial clock.

3.1.3.2. Features of DS3231:

- Simple 3-wire interface.
- Low power operation extends battery backup.
- 8 pin DIP and * pin SO minimizes required space.
- 2v to 5.5 v full operations [11].

Power consumption: RTCs need continuous power and must have extremely low power consumption. Most RTCs use the digital circuits supply when the device is on and active, but switch over to a continuously connected power source when the circuit is powered down. This power source could be a dedicated battery, a charged supercapacitor or a separate power supply from mains [12].

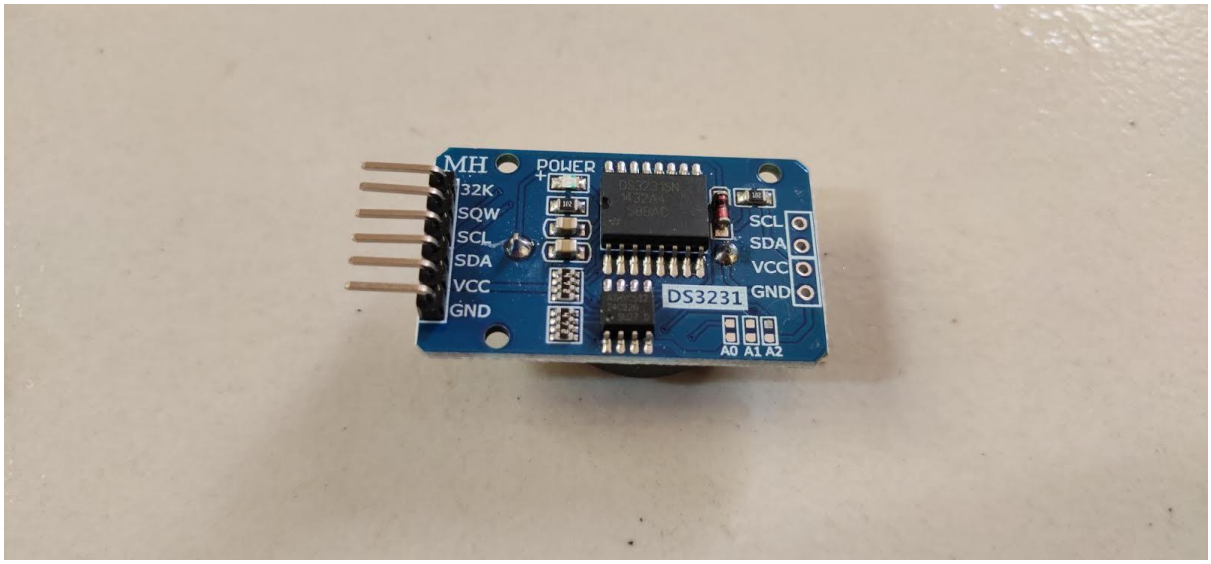


Figure 4: DS3231outlook

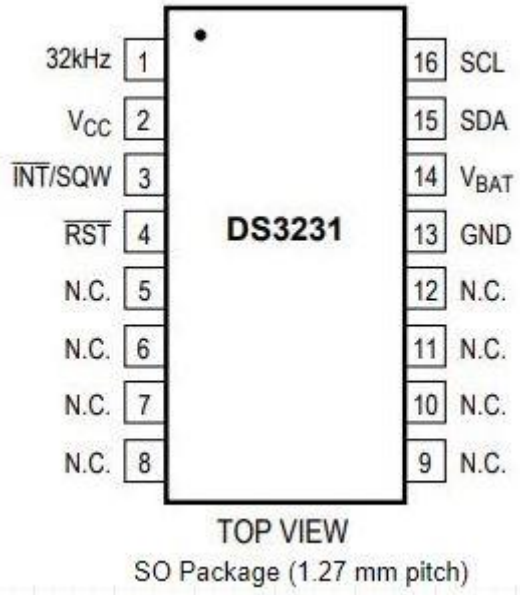
3.1.3.3. Key Features of DS3231

- Highly Accurate RTC Completely Manages All Timekeeping Functions
- Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year, with Leap-Year Compensation Valid Up to 2100
- Pushbutton Reset of DS3231 is active low that debounce input
- Programmable Square-Wave Output Signal extends the implementation sectors of DS3231.
- It enables the easy connection procedures with most of the microcontrollers
- I2C interface is capable of up to 400kHz
- To enable timekeeping continuously, DS3231 has battery supported input
- Operating with low power enables the associated power source to last longer
- Operatable with 3.3 V makes it a lower power consumable chip

3.1.3.4. Applications/Uses of DS3231

- Global Positioning Systems (GPS)
- Servers
- Telematics
- Utility Power Meters [13]

3.1.3.5. Pin out Diagram of DS3231



3.1.4. Relay

The relay is the device that open or closes the contacts to cause the operation of the other electric control. It detects the intolerable or undesirable condition with an assigned area and gives the commands to the circuit breaker to disconnect the affected area. Thus, protects the system from damage.

3.1.4.1. Selected relay module and its features

The eight-channel relay module contains eight 5V relays and the associated switching and isolating components, which makes interfacing with a microcontroller or sensor easy with minimum components and connections. Each relay on the board has the same circuit, and the

input ground is common to all eight channels.

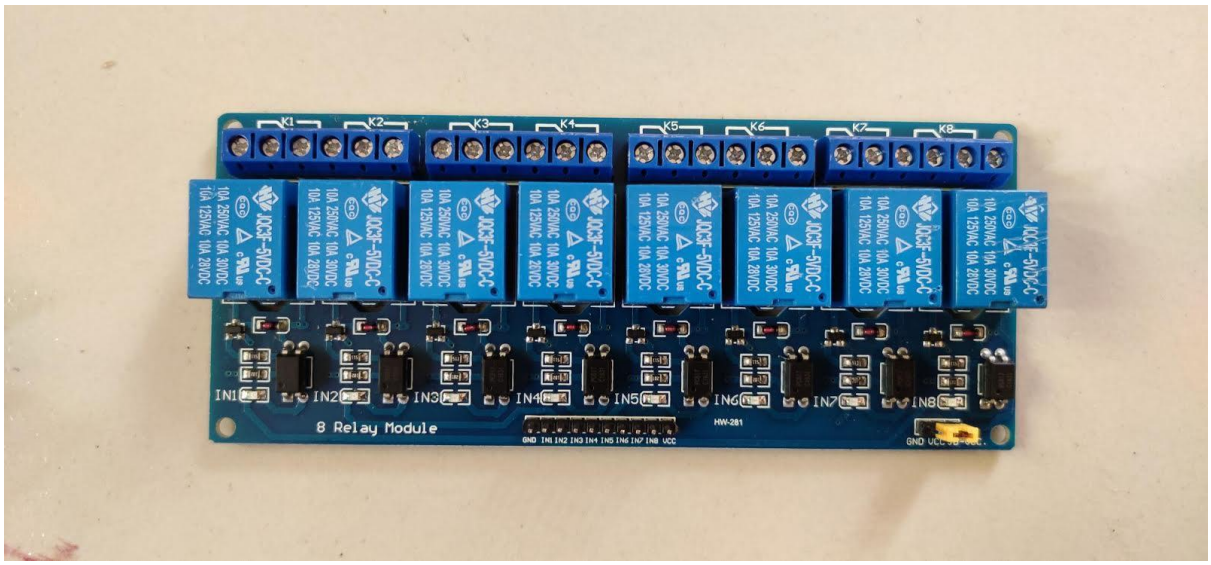


Figure 5: Relay outlook

3.1.4.2. Pin diagram of relay

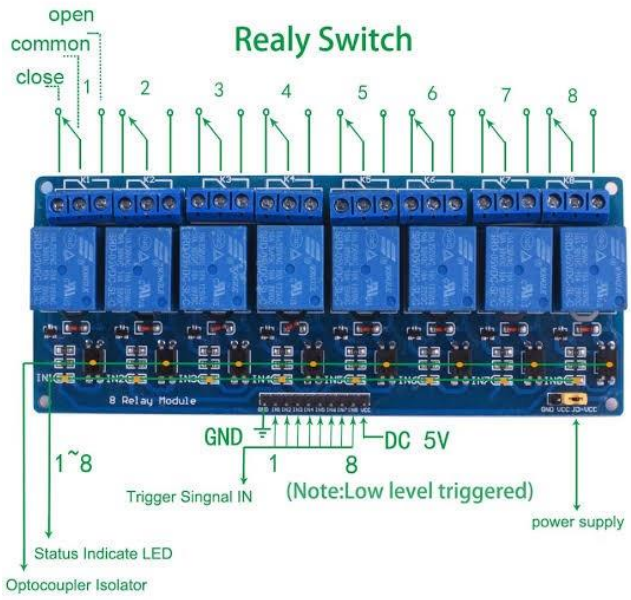


Figure 6: Pin Diagram of Relay

3.2. Software Analysis

3.2.1. Arduino IDE

The Arduino IDE is a cross-platform software written in C and C++ functions. It's used to build and upload applications to Arduino-compatible boards and other vendor development boards with third-party core support. The source code of the IDE is licensed under the GNU General Public License, version 2. The Wiring project comprises a software library that comes with the Arduino IDE and offers a variety of common input and output processes. The Arduino IDE platform is programmable in C & C++ which is easier to implement and basic two function to start the sketch and main program loop makes it easier to deal with. The associated library that comes with the Arduino Integrated Development Environment provides sketches with extra functionality. Sketchbook folder holds the installed library and the library folder is created automatically to create a user-friendly interface.

Figure 7: Arduino IDE Interface

3.2.2. Adafruit Server

Adafruit is an online service that allows a cloud service to connect devices to the cloud. Operating hardware devices online requires a server to store data and design the controlling method. Adafruit is one of the easiest ways to connect IoT devices to the cloud. Storing data and receiving data from microcontrollers or kind o hardware implementation are the basic working procedures of adafruit. Though adafruit is an open-source hardware industry but adafruit.io acts as a server for the devices connected to IoT.

The opening page of the server allows individuals to sign up to work as a registered user. A registered user can act as a designer to design his or her own screen of controlling the hardware from anywhere around the world. Adafruit server mostly works as a free server for the researcher and student to check the prototype. A predeclared bound rate of data is the only drawback of this server. An exceeding value of the preregistered data on a short period will temporarily cause the server to hang which will be resulting in an aid ban for any particular user.

Adafruit server acts as an MQTT broker that is used to feed to store data along with metadata from a device to the cloud. Any hardware connected to the adafruit server can be controlled through its designed value

. 

Figure 8: Server logo of Adafruit.io

3.2.3. MQTT Protocol

MQTT protocol maintains bidirectional communication between the device and cloud or vice versa. To broadcast data interaction between the cloud and device MQTT protocol is an essential tool to maintain communication. To connect more than one device up to millions of IoT devices MQTT protocol comes with the support that establishes connection between the

cloud server and the hardware system. With minimal network bandwidth and the small code connects the devices which acts as a lightweight messaging transportation system to connecting devices remotely. Several industries as automotive technology, manufacturing production house, telecommunication industries are using this protocol widely to build a secure and protected system of data interaction. MQTT is an OASIS standard for IoT connectivity that allows to minimize bandwidth and avoid unreliable networks whilst also maintain the messaging protocol of data interaction.

Chapter 4

System Working Principles

CHAPTER 4: Working principles

4.1. System Block Diagram

To ensure a constant power supply on our design prototype, we have used a 12V lithium polymer battery that is fed through the solar cell. The design prototype is connected to the power supply that enables the microcontroller to operate. ESP32 microcontroller is the main development board of this design prototype, which feds data to the server. The microcontroller directs the LDR value to the server and the server represents the current status of the light associated to the system LDR values changes based on the light falling on its surface. The LDR sensors value compared to the initial set value on a particular time of a day will turn ON or OFF the associated light within the system. On the other hand, the microcontroller will continuously update the LDR value to the server to monitor.

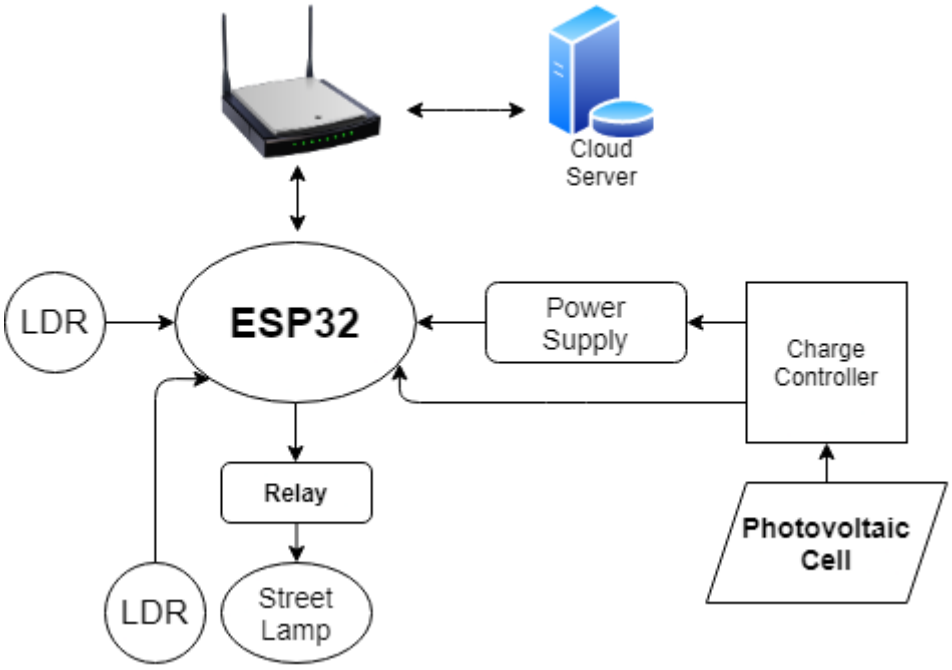


Figure 9: System Block Diagram

4.1.1. Control Circuit Diagram

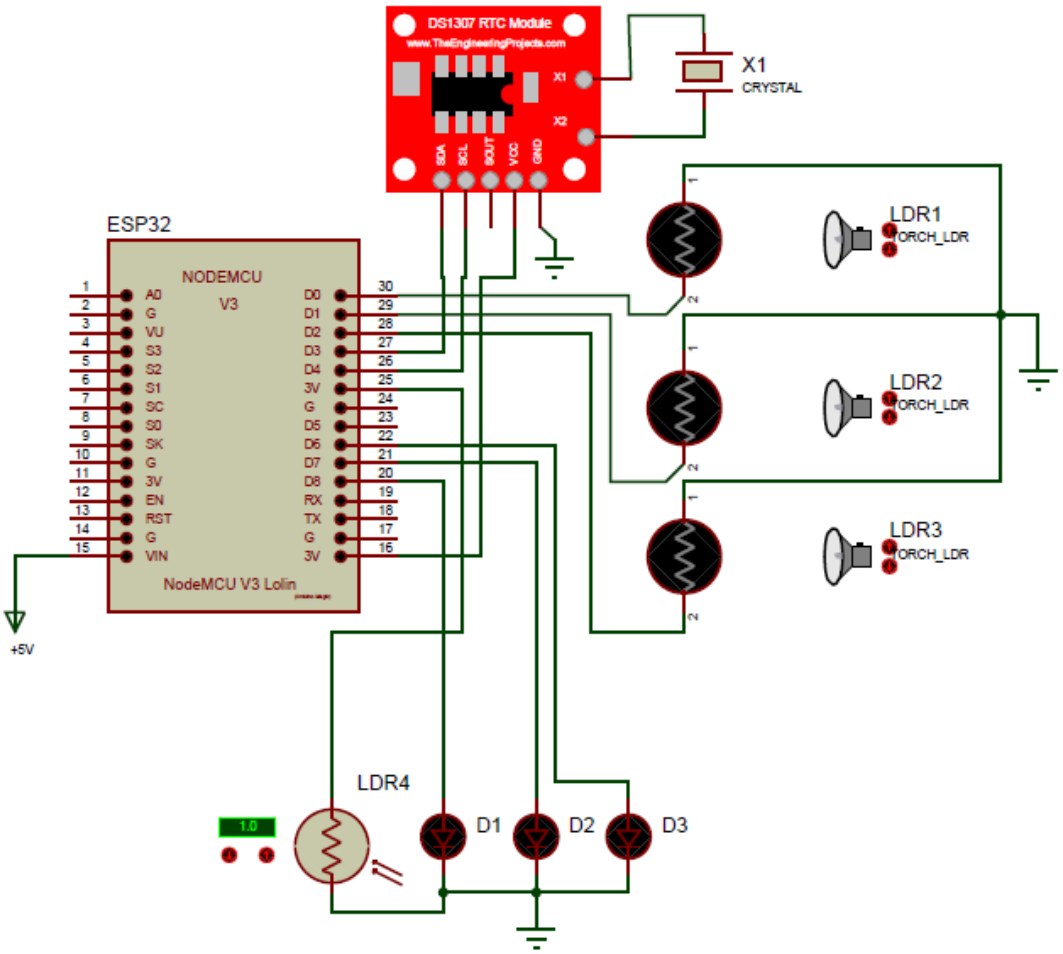


Figure 10: Circuit Diagram

The circuit given above consists of the development board ESP32, or in another name Node MCU which is the central controlling unit for the entire smart system. The instruction set via the programming language (basic C/wiring) has been installed on the chip so that it can perform accordingly.

Here, three of the LDR sensors denoted as LDR1, LDR2 and LDR3 have been used as the input for each individual street lamp denoted as D1, D2 and D3. When the calibrated amount of light

will be read by the sensor then the bulbs (D1, D2 and D3) will turn On and OFF depending on the lighting condition.

For the slave bypassing system, an RTC (Realtime Clock) which is DS3231 has been installed. This will help the entire system to perform more accurately. In case of bad weather, the LDR might not work well because of the low visible light condition but this RTC can solve the issue immediately (if needed).

Custom control: The ESP32 development kit is mutually connected with the custom server via WLAN connection. So, it can be controlled from anywhere, anytime. A basic HTML page has been designed so that the central override can be activated at any time in case of emergency.

Fault detection: Highly calibrated LDR sensor has been placed beside every load bulb so that when it is turned on then the status of those bulbs can be generated and fed back to the central monitoring system. From the reading of that LDR it can be detected whether the lamp is damaged or not. Here, in the circuit above, LDR4 is the sensor which has been used for that purpose only.

4.2. Server portal

A famous server hosting website adafruit.io has enabled function like hosting a private portal at free of cost is used to design the server portal. The server portal allows to monitor several statuses designed to understand the current situation of our designed system. The prototype design functions by enabling a server control that allows to turn on or off the bulb associated to the system remotely. Along with that, once the bulb is turned on or off, sudden changes can be also monitored from the server portal. The server portal has 3 different layouts that can be organized for the best user experience. Bulb indicator, status of the bulb, LDR value and status indicator of the technical fault is placed in two different columns of the server portal. By using the proper user id and password of the Adafruit system, one can easily have access into the system to monitor the current status of the designed system. Though the server has its own limitation. The server has a fixed rate of data interaction for a project. If the data interaction rate is increased by a maximum rate, then a particular account gets blocked for a fixed period of time.

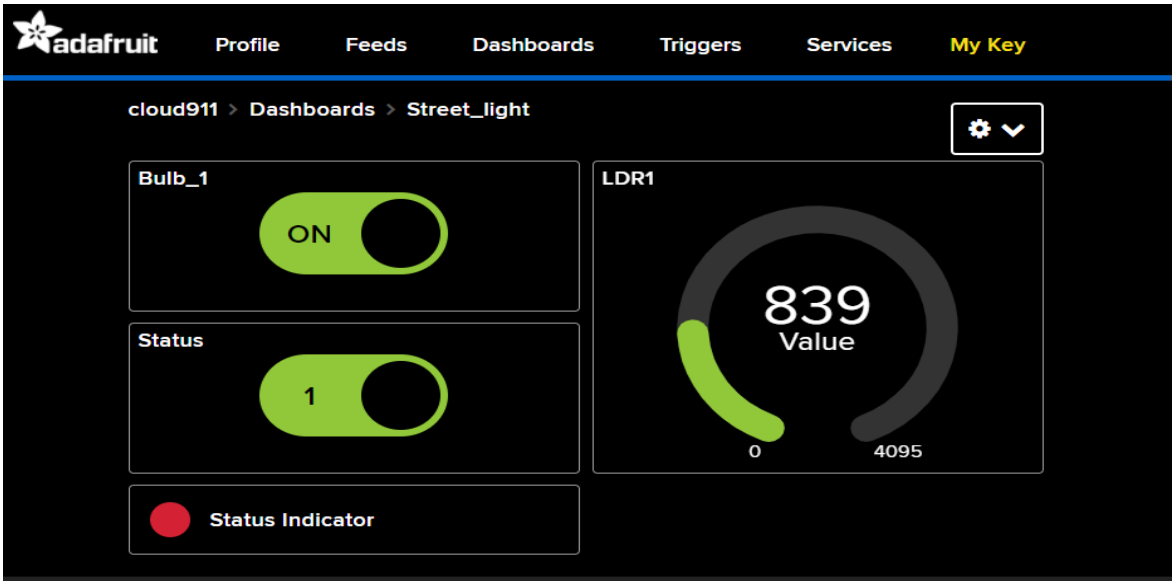


Figure 11: Server Portal

4.3. Working principle of the program:

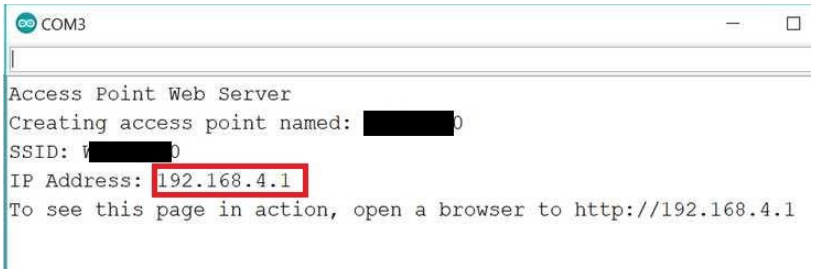
All the required libraries have been called at the top portion.

```
3  #include <WiFi.h>
4  #include "Adafruit_MQTT.h"
5  #include "Adafruit_MQTT_Client.h"
```

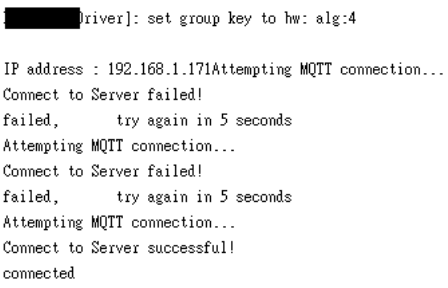
Then the most necessary part after recalling library is to get the system connected with the internet so that it can start engaging the MQTT protocol to keep the communication in between devices and server the perspective WLAN SSID and the PASSKEY are required in this part. If the SSID or the passkey is found wrong then it won't connect to the internet rather the WiFi shield will keep trying to reconnect again and again until it is connected with the internet.

```
9  #define WLAN_SSID       "Sn[redacted]w"
10 #define WLAN_PASS       "M[redacted]"
```

If the WiFi shield is successfully get connected to the internet, then the local device IP address and the message are printed to ensure it gets connected successfully.



When internet connectivity is established then MQTT protocol tries to engage the communication between the server and the device immediately. If it fails in any case then after five (5) seconds of delay it'll try again to reconnect with the server until it gets connected.



After that, there are four very basic parts that needed to introduce inside the code. They are:

server address, server port, the user's name of the portal, and then the most important API Key.

```
14 #define AIO_SERVER      "io.adafruit.com"
15 #define AIO_SERVERPORT  1883                   // use 8883
16 #define AIO_USERNAME    "XYZ"
17 #define AIO_KEY         "aio_XXXXX-0000000000-G8Gjuh0dd"
```

After that part, every GPIO pins need to be declared in the first place to avoid any inconvenience.

```
int s =0;
int d = 0;
int LDR1 = 33;
int LDR2 = 34;
int LDR3 = 35;
int LDR4 = 32;
int LED = 2;
int Load_LED1 = 4;
int Load_LED2 = 5;
```

A server request is sent including all the necessary address and access details.

```
WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
```

Later that part, all the part_value or feeds need to be called so that the correct value get published incorrect path or vice versa.

```
Adafruit_MQTT_Publish pub1 = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/gauge");
Adafruit_MQTT_Publish pub4 = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/stat");
Adafruit_MQTT_Subscribe Light = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME "/feeds/led-control");//
Adafruit_MQTT_Publish Light_status= Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/led-status");
```

It is necessary to declare the pins purposes in the program such as which will be acting as i/p and which will be acting as o/p. A delay is used so that no buffer value is generated.

```
Serial.begin(115200);
pinMode(LDR1, INPUT);
pinMode(LDR2, INPUT);
pinMode(LDR3, INPUT);
pinMode(LDR4, INPUT);
pinMode(LED,OUTPUT);
pinMode(Load_LED1,OUTPUT);
pinMode(Load_LED2,OUTPUT);
delay(10);
```

The main loop starts here with all the necessary incoming signal read provided by the sensors so that these can be utilized inside the code in the first place.

```
82  int LDR_VAL1 = analogRead(LDR1);
83  int LDR_VAL2 = analogRead(LDR2);
84  int LDR_VAL3 = analogRead(LDR3);
85  int LDR_VAL4 = analogRead(LDR4);
```

```
77  void loop() {
78    //main Code Here
79    //dummy block
80  }
```

Simplified condition in wiring language helps the program runs successfully. And the loop continues the reading and writing according to the prior instructions given inside the code unless it got disconnected by the power.

Chapter 5

Results and Conclusion

Chapter 5: Results and Conclusion

5.1. Results

5.1.1. Hardware Implementation

The hardware implementation of the design prototype of automatic street light control consists of ESP 32(as development board), LDR, photovoltaic panel, battery (as primary power source) and LEDs. All components are attached together to find the actual operating procedure of the system. Here all the sensing LDR have been calibrated in such a situation where the actual lighting condition can be portrayed. The threshold value for the calibration is 1000 of analog read in scale of 4095. As per the operational flow of different states, when the LDR value falls under 1000 the led (as load) associated to the system turned itself in an ON state automatically after a delay of 5 seconds. The delay is usually happened due to server operating protocols that actually accepts only a range of data within a fixed time. To use an uninterrupted system, we have to send data with faster flow which can be resulted in a server banned situation. Such situation can be easily removed by an original database & domain dedicated to this particular project. The hardware accepts command through server portal, automatically turned-on light change the bulb_1 indicator to a green sign indicating the bulb is in ON state. When a bulb is in ON state the status icon changes its position to 1 state and if everything is happened perfectly then the status indicator turns green, resulting a fault less situation of automatic street light controlling system.

Here, another calibration for the status LDR was needed and the value is 3000 in the scale of 4095(total bits). That denotes the bulb is mechanically fine, if the bulb is ON and at the same time status LDR got more than the calibrated value but if from the system it is turned on automatically/manually (*system overridden*) but the status LDR is not reading any inputs then that lead to a permanent damage of the LED/bulb/load.

If the conditions are mentioned then it'd look like:

if (Bulb = =1)

if (status_LDR= =1)

then (status_indicator = = 1)

But if in any case the conditions don't match to each other than it'd look like:

if (Bulb = =1)

if (status_LDR= =0)

then (status_indicator = = 0)

That means the bulb is mechanically defected and need to be replaced or repaired.

Now, the calibrated value may vary from place to place, depending on the different lighting condition it may changes that is why a manual system is installed so that any critical conditions can be avoided in such cases.

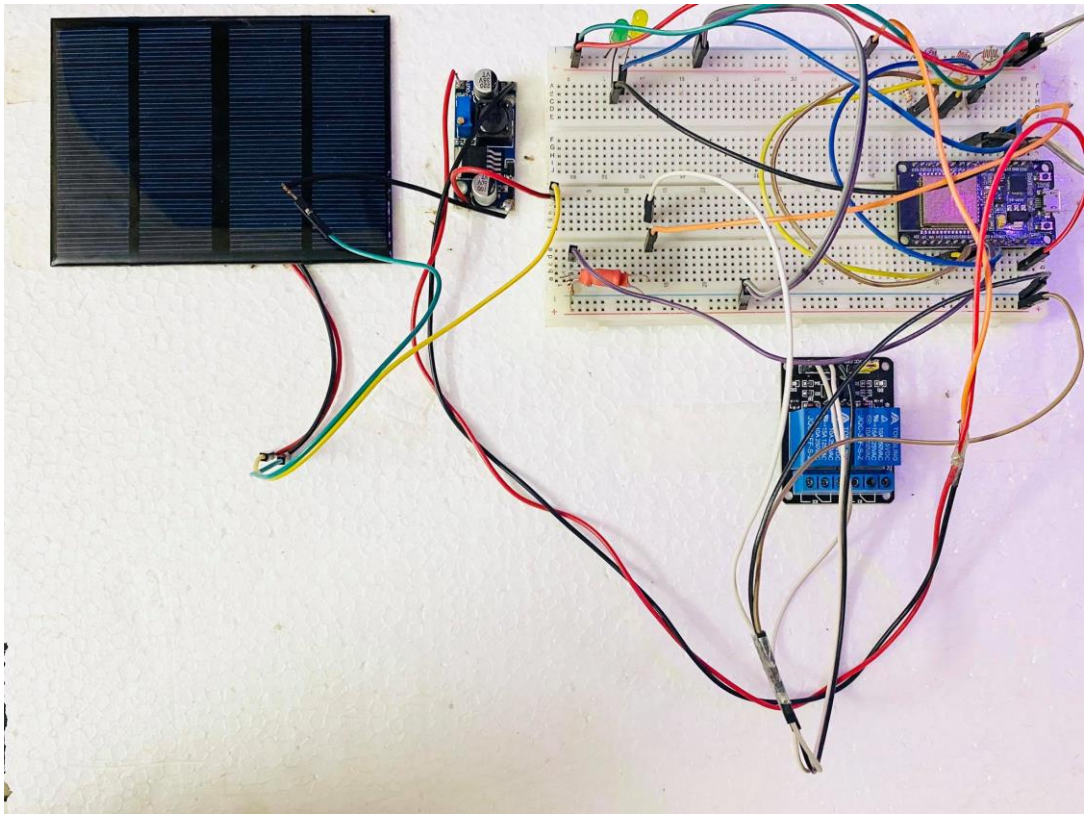


Figure 12: Hardware Implementation of Automatic Street light Controlling system

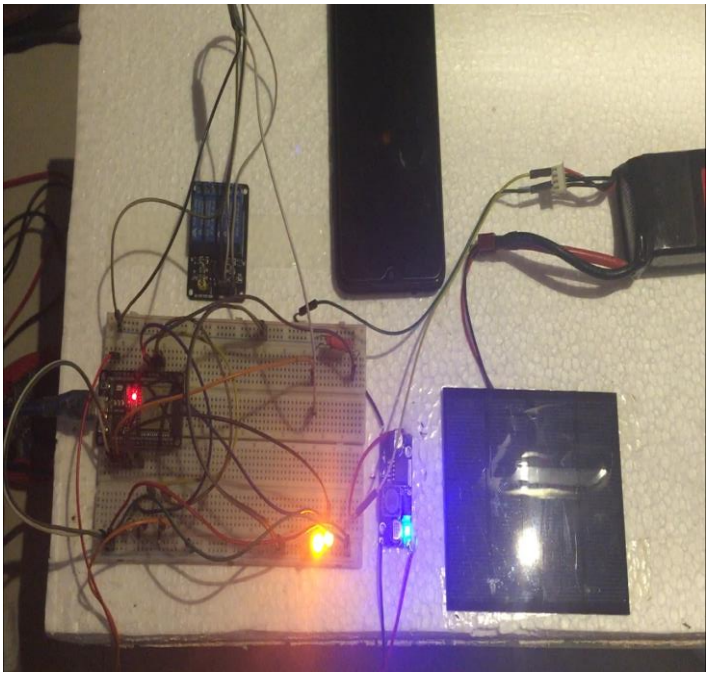


Figure 13: State of Hardware operation while LDR value is less than 1000

5.1.2. Server Control Indicators

Firstly, there is a very unique ***API*** key for every profile in *adafruit* online server. That key has to be in-built inside of the core program. Another important thing regarding code is the code should be according to the portal design. After the designing phase the perspective *part_value* along with proper preference values should be written in proper place inside the program, otherwise the send/get function won't work because of unknown path.

Every component or gauge or switch those have been used to design the portal have their different unique *part_value.* After connecting with the local internet, the MQTT server get started and tries to connect with online server and if it succeeds then first things happens are the *part_values* are get recognized by the server and engage the data send/receive operation. In some scenario it can be seen that MQTT server is trying to connect with the online server

multiple times, this happens only because of the heavy load in the server or if the internet provided somehow blocked that specific IP from the packet tracer protocol.

The design prototype works in three different steps to perform the whole operation of a complete automatic solar powered street light system.

State 1: The system updates the light depending resistor to the server that shows the current value of the resistance of the LDR. A gauge has been used to reflect the current condition of the sensor itself to check whether the sensor is working fine or not and it is synchronized in real-time. Whenever the LDR value falls under 1000 the system will automatically turn on the light associated to the system. As the LDR value indicates the amount of light on a particular environment and eventually the commands on the microcontroller are set in such a way that a particular LDR value will turn on the light. From the figure 9, we can see that the LDR value right at that moment was 839 on a particular environment and that eventually will turn on the light. Which can be monitored from the designed system hardware and the server portal by following the Bulb_1 and Status indicator. Bulb_1 and the status indicator are described in step 2 of this section.

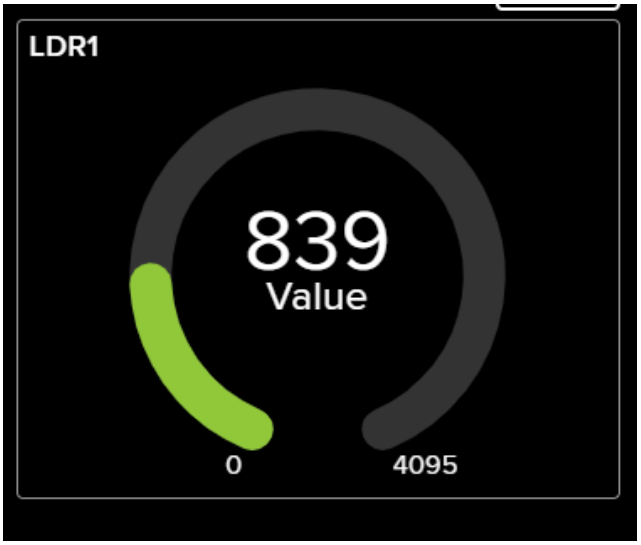


Figure 14: LDR 1 value in server portal

State 2: On the following state the LDR values direct to the server that turns on or off the light associated to the system. There are two indicators designed in the server to address the state of the Bulb and status of the bulb. From the control panel of the Adafruit server, it is possible to turn on the light manually if in any case the light is automatically not turned on because of adverse weather condition. In that case, the status indicator plays its vital part to resolve that particular issue.

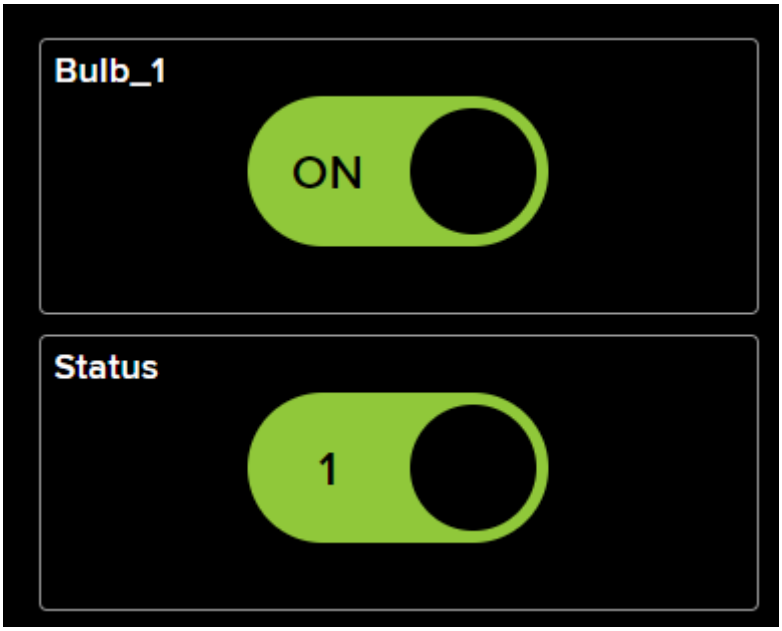


Figure 15: Bulb_1 switch and status of the bulb

Step 3: The last step of the server portal contains status indicator. This portion displays the real time status of the light associated with the system. Status indicator indicates the real time status and technical fault of the hardware. If in case the bulb is not turned on yet the status shows1 then the status indicator of the portal will turn red, which indicates a technical fault in the system. If the status indicator shows green then that will indicate the functionality of the hardware and software system is working just fine.

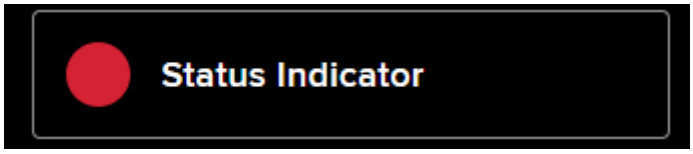


Figure 16: Status Indicator of the server portal

5.2. Conclusion

This research is conducted based on the adafruit.io server system and the common features of a microcontroller enabled an automatic street light control system. This system has work smoothly under the threshold value and a short delay was noticed while operating the design prototype. Such a design enabled an online system to monitor a street light remotely that eventually eliminates the human intervention to operate a street light monitoring system. The status indicator on the server portal is unconditionally the best part of the system that directly indicates the technical fault of the system if there is any. Though the research has its limitation in terms of the faster work process and proper deign monitor but a system consisting designated and dedicated web portal to monitor and automatic switching would really help to eliminate common energy loss related to the street light system.

5.3. Future work recommendation

This research was conducted and developed during a devastating period of the pandemic due to COVID-19 and it has certain limitation as the 3 sec delay to turn on the lights once the LDR

value is below threshold. On the other hand, the server responsible for these issues needs to be resolved studies and a dedicated domain to avoid such issues. This research has further scope to improvised as to implement a new system connecting to the GSM that can sent an automated text message to authorized contact number to inform faulty situations. Moreover, an automatic lighting system with enough precision and pin point accuracy is still can be developed by improving further changes in this criterion.

References

[1].Ther"Resettlement Plan: Solar-LED Streetlights." https://www.adb.org/sites/default/files/linked-documents/37113-013-ban-rpab-03.pdf.

[2]."Internet of Things (IoT) news, blogs and analysis" https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

[3]."Bangladesh Road Network - Logistics Capacity" https://dlca.logcluster.org/display/DLCA/2.3+Bangladesh+Road+Network.

[4]."Lighting, Crime and Safety | International Dark-Sky ...." https://www.darksky.org/light-pollution/lighting-crime-and-safety/

[5]."A Critical Review of Street Lighting, Crime and ... - Springer." https://link.springer.com/content/pdf/10.1057/palgrave.cpcs.8140143.pdf.

[6]."Automatic streetlights that glow on detecting night and ...." 28 Jun. 2018, https://arxiv.org/abs/1806.10968.

[7]."Intelligent Street Light Control Using GSM." http://www.ijiere.com/FinalPaper/FinalPaperIntelligent%20Street%20Light%20Control%20Using%20GSM160263.pdf.

[8]."Intelligent Wireless Street Lighting System." https://www.rroij.com/open-access/intelligent-wireless-street-lighting-system.pdf.

[9]."Automatic Street Light Control System Using Microcontroller." https://www.researchgate.net/publication/321318899_Automatic_Street_Light_Control_System_Using_Microcontroller.

[10]. "Overview of ESP32 features. What do they practically mean ...." https://exploreembedded.com/wiki/Overview_of_ESP32_features._What_do_they_practically_mean%3F.

[11]."DS1302 Trickle-Charge Timekeeping Chip - Maxim Integrated." https://datasheets.maximintegrated.com/en/ds/DS1302.pdf.

[12]."Real-Time Clocks - Basic Electronics Tutorials and Revision." https://www.electronics-tutorials.ws/connectivity/real-time-clocks.html.

[13]."DS3231 Extremely Accurate I²C-Integrated RTC/TCXO/Crystal ...." https://www.maximintegrated.com/en/products/analog/real-time-clocks/DS3231.html.

Appendix A

```
#include <WiFi.h>

#include "Adafruit_MQTT.h"

#include "Adafruit_MQTT_Client.h"

/************************* WiFi Access Point *********************************/

#define WLAN_SSID       "Snowball_Shadow"

#define WLAN_PASS       "Mayday@3down"

/************************* Adafruit.io Setup *********************************/

#define AIO_SERVER      "io.adafruit.com"

#define AIO_SERVERPORT  1883                   // use 8883 for SSL

#define AIO_USERNAME    "cloud911"

#define AIO_KEY         "aio_Nqux378BLYwSvvNYzmzG8Gjuh0dd"

/************************* variable *********************************/

int s =0;

int d = 0;

int LDR1 = 33;

int LDR2 = 34;

int LDR3 = 35;

int LDR4 = 32;
```

```
int LED = 2;

int Load_LED1 = 4;

int Load_LED2 = 5;

/************ Global State (you don't need to change this!) ******************/

WiFiClient client;

Adafruit_MQTT_Clientmqtt(&client, AIO_SERVER, AIO_SERVERPORT,
AIO_USERNAME, AIO_KEY);

/****************************** Feeds ***************************************/

Adafruit_MQTT_Publish pub1 = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/gauge");

Adafruit_MQTT_Publish pub4 = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/stat");

//Adafruit_MQTT_Publish pub2 = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/gauge2");

//Adafruit_MQTT_Publish pub3 = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/gauge-3");

Adafruit_MQTT_Subscribe Light = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME
"/feeds/led-control");//here LED_Control is the feed name that i created in website

Adafruit_MQTT_PublishLight_status= Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/led-status");

/*************************** Sketch Code ************************************/
```

```
void MQTT_connect();

/*------------------------------------setup start-----------------------------------------------*/

void setup() {

Serial.begin(115200);

pinMode(LDR1, INPUT);

pinMode(LDR2, INPUT);

pinMode(LDR3, INPUT);

pinMode(LDR4, INPUT);

pinMode(LED,OUTPUT);

pinMode(Load_LED1,OUTPUT);

pinMode(Load_LED2,OUTPUT);

  delay(10);

Serial.print("Connecting to ");

Serial.println(WLAN_SSID);

WiFi.begin(WLAN_SSID, WLAN_PASS);

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

    delay(500);

Serial.print(".");

  }

Serial.println("WiFi connected");

Serial.println("IP address: "); Serial.println(WiFi.localIP());

mqtt.subscribe(&Light);//here Light is the object of Adafruit_MQTT_Subscribe


}
```

```
/*-------------------------------------setup end-----------------------------------------*/

/*-------------------------------------Loop start----------------------------------------*/

void loop() {

MQTT_connect();

Adafruit_MQTT_Subscribe *subscription;

  int LDR_VAL1 = analogRead(LDR1);

  int LDR_VAL2 = analogRead(LDR2);

  int LDR_VAL3 = analogRead(LDR3);

  int LDR_VAL4 = analogRead(LDR4);

while ((subscription = mqtt.readSubscription(5000))) {

//    if (subscription == &Light) {

//      STED= atoi((char *)Light.lastread);

//      digitalWrite(LED,STED);

//    }

     if (strcmp((char *)Light.lastread, "OFF") == 0) {

digitalWrite(LED, LOW);

       s  = 0;

     }

     if (strcmp((char *)Light.lastread, "ON") == 0) {

digitalWrite(LED, HIGH);

       s = 1;

     }

     if (!Light_status.publish(s))

     {
```

```
Serial.println(F("Failed"));
      }
      else
      {
Serial.println(F("OK!"));
      }
  }
if(LDR_VAL4<3000){
  d = 0;
Serial.println(LDR_VAL4);
}
else{
  d = 1;
Serial.println(LDR_VAL4);
}
    if (!pub4.publish(d))
          {
Serial.println(F("Failed"));
          }
          else
          {
Serial.println(F("OK!"));
          }
  if (!pub1.publish(LDR_VAL1))
```

```
  {
Serial.println(F("Failed"));
  }
  else
  {
Serial.println(F("OK!"));
  }
/*
  if (!pub2.publish(LDR_VAL2))
 {
Serial.println(F("Failed"));
  }
  else
  {
Serial.println(F("OK!"));
  }

  if (!pub3.publish(LDR_VAL3))
  {
Serial.println(F("Failed"));
  }
  else
  {
Serial.println(F("OK!"));
```

```
  }*/
  delay(2500);
if(LDR_VAL2<1000){
digitalWrite(Load_LED1,HIGH);

}
else{
digitalWrite(4,LOW);
}

if(LDR_VAL3<1000){
digitalWrite(Load_LED2,HIGH);
}
else{
digitalWrite(5,LOW);
}
}
/*--------------------------------------Loop END-------------------------------------------*/
void MQTT_connect() {
  int8_t ret;
  if (mqtt.connected()) {
    return;
  }
Serial.print("Connecting to MQTT... ");
```

```
  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
Serial.println(mqtt.connectErrorString(ret));
Serial.println("Retrying MQTT connection in 5 seconds...");
mqtt.disconnect();
    delay(5000);  // wait 5 seconds
    retries--;
    if (retries == 0) {
      while (1);
    }
  }
Serial.println("MQTT Connected!");
}
```