STUDIES AND DEVELOPMENT OF AN IOT BASED TUNABLE LUMINAIRE FOR STREET LIGHTING APPLICATIONS

A Thesis Submitted Towards Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
IN
ILLUMINATION TECHNOLOGY & DESIGN

Submitted By

Mr. SHAHEEN WASIF

Examination Roll No. – M6ILT23006

Registration No. – 154541 2020-2021

Under the Supervision of

PARTHASRATHI SATVAYA
&
PROF. ARABINDA DAS

School of Illumination Science, Engineering and Design
Faculty of Interdisciplinary, Law & Management

Course affiliated to
Faculty of Engineering and Technology
JADAVPUR UNIVERSITY

KOLKATA – 700032

2023

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled "STUDIES AND DEVELOPMENT OF AN IOT BASED TUNABLE LUMINAIRE FOR STREET LIGHTING APPLICATION" is a bonafide work carried out by **Shaheen Wasif** under my supervision and guidance for partial fulfilment of the requirement of ***MASTER OF TECHNOLOGY IN ILLUMINATION TECHNOLOGY & DESIGN***, during the academic session 2020 - 2021.

--	---------------------------------------
PARTHASARATHI SATVAYA	PROF. ARABINDA DAS
Assistant Professor	Professor
School of Illumination Science, Engineering & Design	Electrical Engineering Department
Jadavpur University	Jadavpur University
Kolkata - 700032	Kolkata - 700032

------------------------------	-------------------------------
PARTHASARATHI SATVAYA	Prof. Tushar Jash
Assistant Professor	Dean
School of Illumination Science, Engineering & Design	Faculty of Interdisciplinary, Law & Management
Jadavpur University	Jadavpur University
Kolkata- 700032	Kolkata- 700032

<u>CERTIFICATE OF APPROVAL</u>

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactorily to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

Committee of final examination ---

for evaluation of Thesis

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC ETHICS

It is hereby declared that this thesis contains literature survey and original research work by the undersigned candidate, as part of his **MASTER OF TECHNOLOGY IN ILLUMINATION TECHNOLOGY & DESIGN**, studies during academic session 2020-2021.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rules and conduct, I have fully cited and referred all material and results that are not original to this work.

NAME	: SHAHEEN WASIF
ROLL NUMBER	: M6ILT23006
REGISTRATION NUMBER	: **154541 of 2020-21**
THESIS TITLE	: **STUDIES AND DEVELOPMENT OF AN IOT BASED TUNABLE LUMINAIRE FOR STREET LIGHTING APPLICATION**

SIGNATURE: DATE:

Acknowledgement

I have immense pleasure in expressing my profound gratitude and sincere thanks to my respected thesis advisor **Parthasarathi Satvaya** Director of SISED, Jadavpur University, & **Prof. Arabinda Das** Professor of Electrical Engineering Department, Jadavpur University for their valuable guidance, constant encouragement, useful suggestions, keen interest and timely help in completing this thesis successfully. It has been an excellent opportunity for me to work under their supervision.

I would like to thank **Prof. (Dr.) BISWANATH ROY,** Head of the Department, Electrical Engineering, Jadavpur University for providing the necessary facilities required to carry out this thesis.

I would like to thank **Prof. (Dr.) Saswati Mazumdar,** Dean of Faculty of Engineering and Technology, Jadavpur University for her constant encouragement, useful suggestion that motivated me to work hard complete my thesis with dedication.

I would also like to thank **Mr. Gobinda Paul**, Senior Lab Attendant of SISED, Jadavpur University & **Mr. Samir Mandi**, Technical Assistant(Group II),Illumination Engineering Section, Electrical Engineering Department, Jadavpur University for letting me have access to necessary laboratories for carrying out the required experimental and developmental works required for completion of the thesis work and also their assistance in performing the experimental procedures required for completing the thesis work.

I would like to acknowledge my sincere thanks and respect to **Mr. Santosh Sahoo**, Visiting Helper of Illumination Engineering Laboratory for his enormous support.

Further, I also take this opportunity to thank all the teachers who taught me and shared their knowledge with me. I must express my heartiest thanks to my juniors, seniors and my alma matter Jadavpur University.

Date: ----------------------------------
Place : Jadavpur University Shaheen Wasif
Kolkata – 700032

TABLE OF CONTENTS

1.INTRODUCTION

1.1 Overview

In this modern era of technology, it is tremendously important for us to implement various smart ways for energy savings and energy consumption. Lighting constitutes a large part of energy consumption.

In a metropolitan city, Street lights account for approximately 30-40% of the total energy consumption [1]. The street lights are switches on from evening till early morning. Hence a large power consumption is observed in case of street lighting due to it's unregulated usage. However, the electrical power consumption can be reduced to a great extent by intelligent and sophisticated control of brightness of street lights.

The energy consumption can be regulated by controlling the dimming levels of the street light in accordance to time of the day. The dimming levels can also be controlled in accordance with one's needs and convenience.

The street lights can also be dimmed when there is no vehicular movement but will get set to a particular brightness level when there is vehicular movement. In this way energy consumption can be drastically reduced.

Another issue is the introduction of automation in street lighting system. The street lighting system can be thought of as a network of nodes where sensors, controllers, etc are installed. The ease of automated operation is introduced in this concept. Here a central control facility is provided and the operation is controlled from this station. The operation can be anything from switching ON and switching OFF of all the luminaires in the network, to controlling and monitoring the dimming levels as well as electrical power consumption, so as to make an efficient, robust system which requires less maintenance.

A pictorial representation of the concept discussed above has been shown below in Figure 1.1.

In this figure all the street lights have Wi-Fi based lamp controller mounted on them and one Gateway which connects to the internet, allowing for the user ,wireless access to control lighting of luminaires and monitoring their conditions.

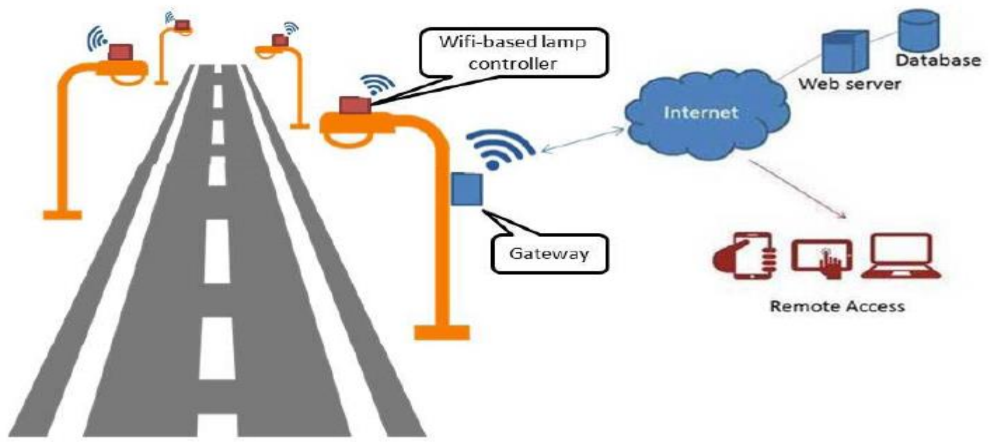


Fig1.1 Street light controlling using Wi-Fi-Mesh network.

Wireless control of luminaires in the street lighting system is being implemented with special emphasis on wireless dimming control. Integrated into this system, are various sensors for sensing the vehicular movement and setting of the brightness level according to various types of condition will also be implemented.

The development of a robust system which is user friendly, easy to maintain and less complicated has been done, without compromising on functionality.

1.2 Light Emitting Diode (LED): [2]

A. What is a LED ?

LED is a short form of Light Emitting Diode. It is a type of diode that emits light when a current pass through it. In other words, LED is a special type of diode that converts electrical energy into light energy. It is a simple PN junction diode that radiates light in forward bias. The junction is covered in transparent epoxy to direct the light emitting from the junction in all directions. When charge carriers electrons and hole combines, they release light energy. Not every material is capable to have such property.

The property of a material to convert electrical energy into light energy is called Electro-Luminance. Instead of Silicon and germanium, gallium arsenide, gallium phosphide, and indium phosphide compounds are used depending on the colour of emitting light.

B. Circuit symbol and terminals of a LED.

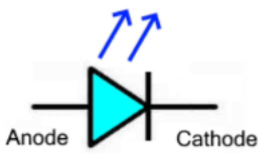


Fig 1.2 circuit symbol of a LED

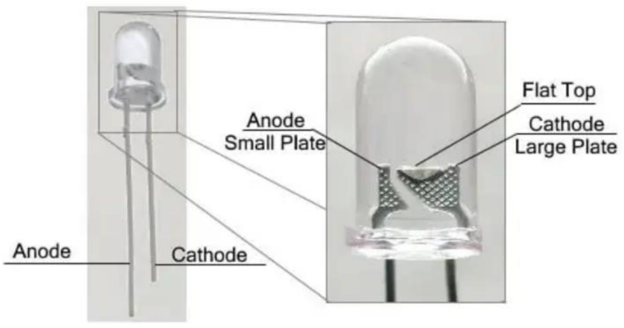


Fig 1.3 Terminals of a LED.

C. Construction of LED

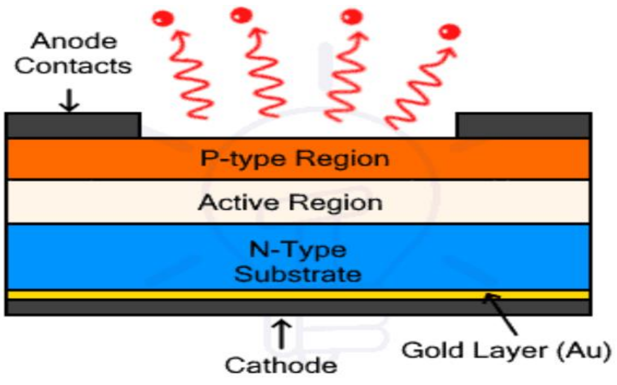


Fig 1.4 Cross sectional view of a LED showing it’s various layers

LED is made of three layers i.e. P-type semiconductor layer, N-type semiconductor layer and active region. The N-layer had the majority of electrons while the P-layer has a majority of holes. The active region has an equal amount of electrons and holes therefore there are no majority charge carriers. The active region is also known as the depletion region. The electrons and holes recombine in this region. Light emits when an electron and a hole combines. The holes are the absence of electrons. They do not move. The electrons move and combine with holes in the p-layer. Therefore, the p-layer is designed to be kept at the top of the LED.

The layer of P-type material and N-type material is combined together on top of each other with an active region between them. As the electron-hole recombination occurs in p region, the p layer is kept at the top and the anode is deposited at the edge of the p layer to have maximum light emission. While for the cathode, a gold film is deposited at the bottom of the N-type layer as shown in the figure.

Physically the LED is designed to have maximum light emission. Therefore, the junction is covered in transparent epoxy with a dome shaped top. It helps in concentrating the light emitted from the junction in the upward direction. The thin gold film at the bottom is especially used to reflect the light back in the upward direction to increase the LED efficiency. Since most of the light emits from the p-region, increasing its area increases the light intensity. The shape of the epoxy resin does not have to be hemispherical. It can be triangular, or rectangular as well depending on the application.

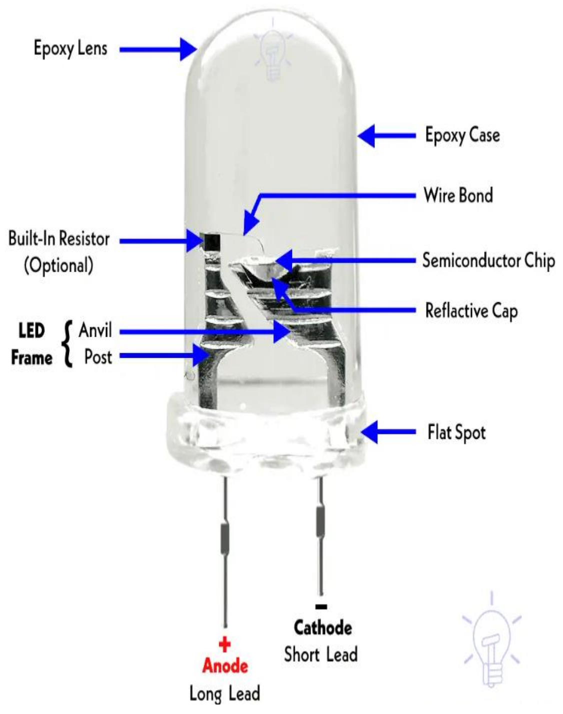


Fig 1.5 Figure showing various parts of a LED.

D. Working of an LED

Just like any normal diode, LED or light Emitting Diode only operates in forward bias i.e. the anode is kept at a higher voltage as compared to the cathode, or the anode is connected with the positive terminal and the cathode is connected with the negative terminal of the battery. The n region has electrons in the majority while the p region has holes in the majority. Apart from that, the n-type layer is heavily doped as compared to p-type layer.

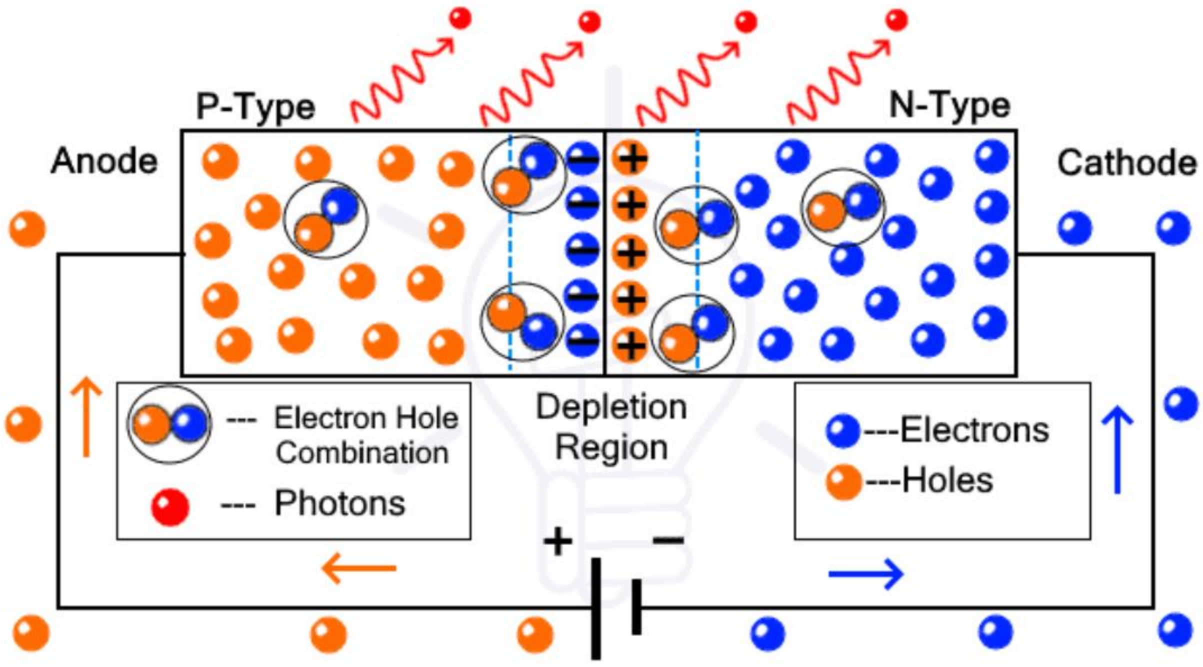


Fig 1.6 Working of a LED

When LED is forward biased, the applied potential starts pushing on the P-layer and the N-layer. As a result, the depletion region or the active layer starts to shrink. Therefore electrons from n region and holes from p region start passing through the junction. It starts to recombine in the active region or depletion region. During its recombination, the electrons from the higher band (conduction band) fall into the lower energy band (valance band) by recombining with the holes (absence of an electron in the valance band) and release the energy in the form of light. After a few recombination, the width of the depletion region further decreases and the intensity of the light is increased.

The property of conversion of electricity into light energy is called Electro-Luminance. Certain semiconductors exhibit such property such as the GaAs, GaAsP, GaP. Silicon and germanium cannot emit light but only heat. Why these materials exhibit such property can be understood by using the energy band theory of solids.

Electron can attain energy in discrete form. The energy of an electron can be determined by its location in the energy bands. When an electron gains energy it jumps to a higher energy band and when it falls back to a lower energy band, it releases energy. The valance band has lower energy than the conduction band. The difference between the conduction band and the valence band is called the energy gap.

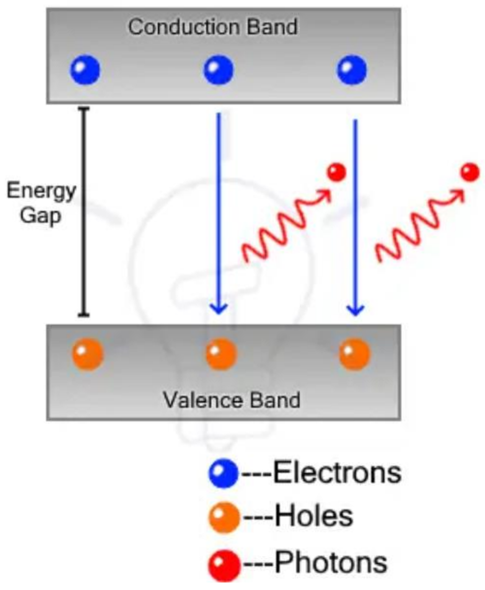


Fig 1.7 Phenomenon of Electro luminance explained through Energy Band Theory.

When an electron and hole combine it emits the energy equal to the energy gap between the conduction band and the valence band.

Direct Band Gap

The direct band gap semiconductor material emits photons or light energy when it releases energy. The two energy bands i.e. conduction band and valence band are directly above each other along with a momentum ‘k’ and energy graph. When an electron and hole combine it emits the energy equal to the energy gap between the conduction band and the valence band. Therefore semiconductors having a large energy gap emits high-intensity light. Different materials emit different wavelength therefore the color of the light depends on the type of material.

Such semiconductor materials are used in construction is LED. Example of direct band semiconductors is Aluminium Gallium Arsenide(AlGaAs), Gallium Arsenic Phosphide(GaAsP), Aluminium Gallium Phosphide(AlGaP), Indium gallium nitride (InGaN), and Zinc Selenide(ZnSe), etc.

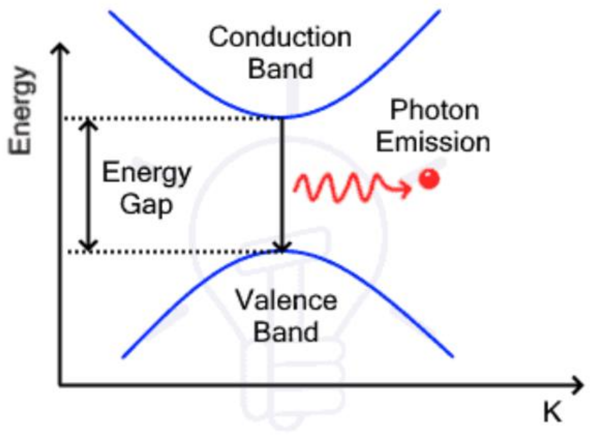


Fig 1.8 Energy-Momentum diagram of direct band gap semiconductors.

Indirect Band Gap

The indirect band gap semiconductor materials do not emit photons when the electron releases energy but rather releases it in the form of heat. In such materials, the conduction band does not directly align with the valence band as shown in the figure. Due to the difference in the momentum ‘k’, the electron-hole combination only releases energy in the form of heat. Examples of such materials are silicon, germanium, etc.

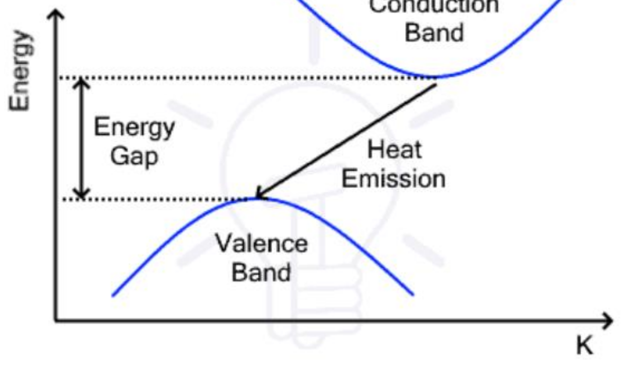


Fig 1.9 Energy-Momentum diagram of indirect band gap semiconductors.

E. Output characteristics of a LED

The output characteristic of LED shows the relationship between the output light intensity and the input forward current I_F of the LED. The light intensity is represented by the vertical axis and the forward current is represented by the horizontal axis.

The light intensity is directly proportional to the forward current going through the LED. The intensity or radiant power varies linearly with the forward current. The light intensity also depends on the temperature. An increase in temperature decreases the light intensity of the LED as shown in the graph.

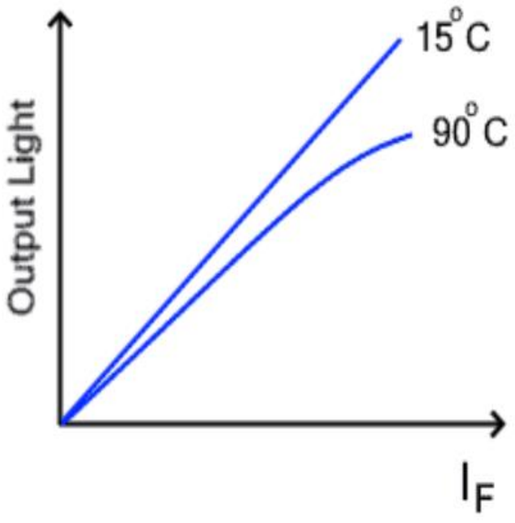


Fig 1.10 Graph showing output characteristics of a LED

F. Electrical characteristics of LED

LED basically operates as a normal diode i.e. It allows current in forward direction, therefore, the graph only shows forward voltage and forward current. The forward voltage of LED is greater than the normal diode due to the presence of an active layer. At first, LED does not pass current and does not produce light until the forward voltage exceeds knee voltage. Different color of LED has different knee voltage. Once the LED starts conduction, the current starts increasing exponentially which is directly proportional to the intensity of the light emitted.

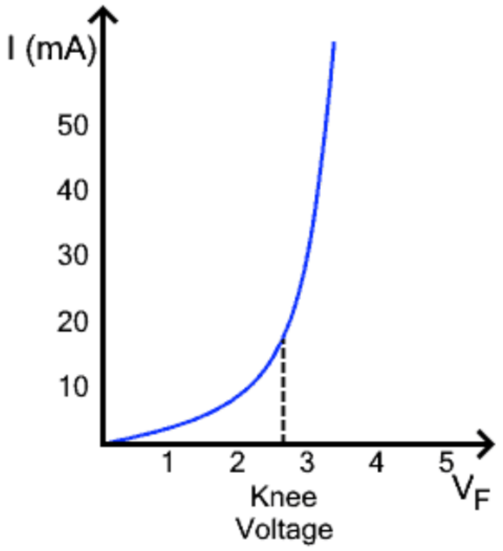


Fig 1.11 Graph showing electrical characteristics of a LED

G Advantages of LEDs

- LEDs are cheaper, cost-effective and very reliable.
- Low temperature does not affect its performance.
- The switching time of LED is very fast in the range of 1ns. Therefore they are suitable for dynamic operation.
- It doesn't need to warm up. It instantly lights up.
- It can be designed to emit various colours of light such as red, green, yellow, orange, white, etc.
- The intensity or brightness of LED can be easily controlled by varying the current flowing through it.
- As compared to an incandescent bulb, LEDs are 10-50 times more efficient.

- It doesn't require toxic gasses that are harmful to the environment.
- It has a longer lifespan.

H. Disadvantages of LEDs.

- It has a higher operating voltage and consumes more power as compared to conventional PN junction diode.
- It cannot tolerate spikes in current and voltage. It can permanently damage the LED.
- It can also get overheated due to radiant power. It reduces the LED life.
- Its efficiency drops due to a large current or heat. This phenomenon is also known as efficiency droop.
- Its light color may shift over its life span.
- It has a higher initial cost as compared to an incandescent light bulb

1.4 Dimming control of LED luminaires

Luminaires can be dimmed so as to save energy. Most of the luminaires used today are LEDs due to their superior performance and energy efficiency over the other lamps. The LED. According to the paper on "Hybrid Series-Parallel PWM dimming technique for Integrated-Converter-Based HPF LED Drivers" [3], dimming can be done by mostly two methods:

A. Amplitude Modulation (AM): In this type of dimming, the amplitude of the DC current passing into the LEDs is changed leading to a change in the output luminous flux. This is the easiest one to implement. The disadvantage with this type of modulation is that at high injected currents a lack in linearity may occur and also a shift in chromaticity is observed.
B. Pulse Width Modulation (PWM): In this type of modulation, the width of the pulse is varied which causes the change in output luminous flux of the led. It has been proved that LEDs could work with pulsating currents, which indicates that the dimming of the LEDs could be made perfectly through PWM dimming techniques. The reason behind this is the fact that the LEDs are semiconductors devices, so they could be turned on and off quite fast.

DUTY CYCLE : It is the percentage measurement of how long the signal stays ON.

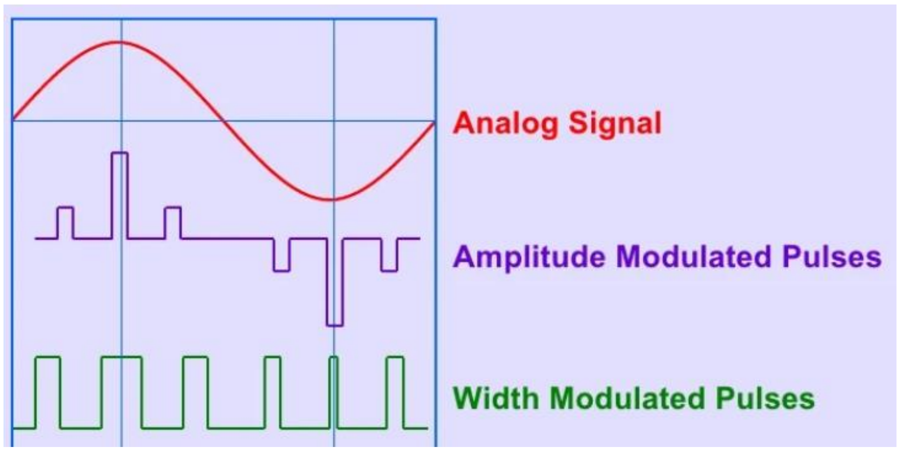
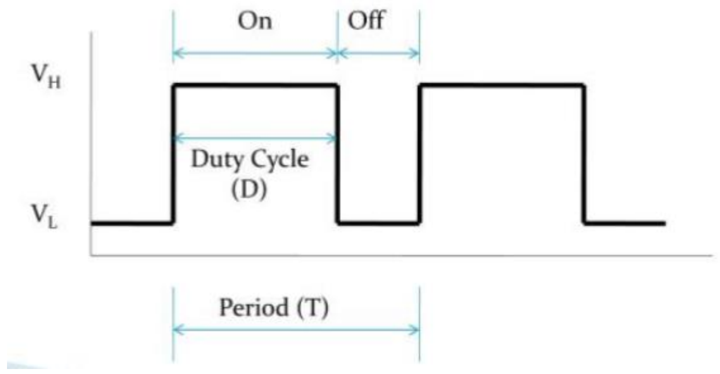


Fig 1.12 Explanation of AM and PWM dimming techniques



Duty Cycle:

$$\text{Duty Cycle} = \frac{\text{On Time}}{\text{Period}} \times 100\%$$

Fig 1.13 Concept of Duty cycle

1.5 Wireless connectivity protocols

Some of the most commonly used wireless connectivity protocols are discussed below:

- Bluetooth: This technology is more commonly used on portable devices such as mobile phones, headphones and laptops and it allows small amounts of data (up to and including music) to be transmitted from one Bluetooth device to another – hence it commonly being used for streaming music from a phone to headphones. It's limited in terms of how much data can be transmitted, meaning that although video data can theoretically be streamed over Bluetooth, in reality this doesn't work too well and so WiFi is often used for anything beyond music streaming.
- ZigBee: This is a low-power, low-bandwidth protocol which was designed for home automation and similar uses**.** It supports encryption as standard, and it also has a 'mesh network' as standard which means that ZigBee devices can link together and improve the range (and reliability) of the overall ZigBee network. Zigbee devices can find each other without the need for extra passwords being entered, and the whole process is encrypted.
- WiFi : Wireless Fidelity or Wi-Fi is the go-to standard for smart home setups and applications involving large amounts of data movement across a wireless network. The Wi-Fi technology is based on the family of wireless networking standards which define many specifications for Wi-Fi LANs. Wi-Fi has an RF range of up to 100 meters (328 feet). Wi-Fi is implemented in all laptops, tablets, smartphones and TVs, making it logical to leverage this ubiquitous infrastructure. However, Wi-Fi is too resource-intensive. It star topology, which means that all its nodes connect directly to a wireless router, may cause single point of failures if the router goes down. [3],[5]

Each of the above techniques of dimming control has it's own disadvantages :

I. ZigBee:
   - Cost of the ZigBee devices are most than the comparative Wi-Fi devices.
   - ZigBee uses 2.4GHz radio band which is same as that of Wi-Fi, so sometimes it can interfere with Wi-Fi.

II. Wi-Fi:
   - New password has to be inserted for every new device that we connect to the Wi-Fi network.

- If we change the router, we again have to change the Wi-Fi-password for every device or change the router SSID to be the same as old one.

III. Bluetooth:

- Range of connectivity is not good.
- Bandwidth is poor
- Bluetooth also uses 2.4GHz radio band which is same as that of Wi-Fi, so sometimes it can interfere with Wi-Fi.

Table 1.1 Table showing the comparison between the parameters of the three different protocols as of July, 2023 [3].

PARAMETERS	**ZigBee**	**Wifi**	**Bluetooth**
RANGE	GOOD	GOOD	NOT GREAT
POWER USE	LOW	HIGH	MEDIUM
BANDWIDTH	POOR	EXCELLENT	POOR
RF BAND	2.4GHz	2.4GHz/5GHz	2.4GHz
NEEDS HUB?	YES	NO	NO
PRICE OF SMART DEVICES	HIGH	LOW	MEDIUM

1.6 Objectives

The Objectives of carrying out the thesis work are :

- Wireless tunable lighting control of luminaires.
- Integration of IoT in wireless data transfer for lighting control of luminaire.
- Observation and analysis of the electrical and illumination parameters output of wirelessly tunable street lighting system
- Observing different fixed illumination levels for different digitally controlled lighting levels.
- Development of wireless communication methods for receiving and transmitting data with and without special conditions .
- Development of a microcontroller based controller circuit for controlling the LED drivers.

2. LITERATURE REVIEW

2.1 INTRODUCTION

A huge amount of work has been done in the field of smart lighting. Various wireless connectivity protocols have been used for connecting to internet as well as wirelessly controlling the dimming level of luminaires. Developmental works have been done in fields like digital control of luminaires, connection of microcontrollers to internet, transferring data packets via wireless communication, integration of sensors with smart lighting as well as object detection, etc. Some of these works have been reviewed.

2.2 Wireless control of luminaires

Wireless connectivity is an important part of the Internet of Things (IoT). Wireless connection is the conjunction of wireless networking and automatic lighting which makes lighting smart and intelligent. There are various protocols for wireless connectivity. Wireless lighting control refers to over-the-air management of lamps or light fixtures using wireless communication technologies. Wireless control not only simplifies design, installation, commissioning, and operation of lighting systems, but it also provides a way for users to make a single environment flexible enough to accommodate various visual needs.

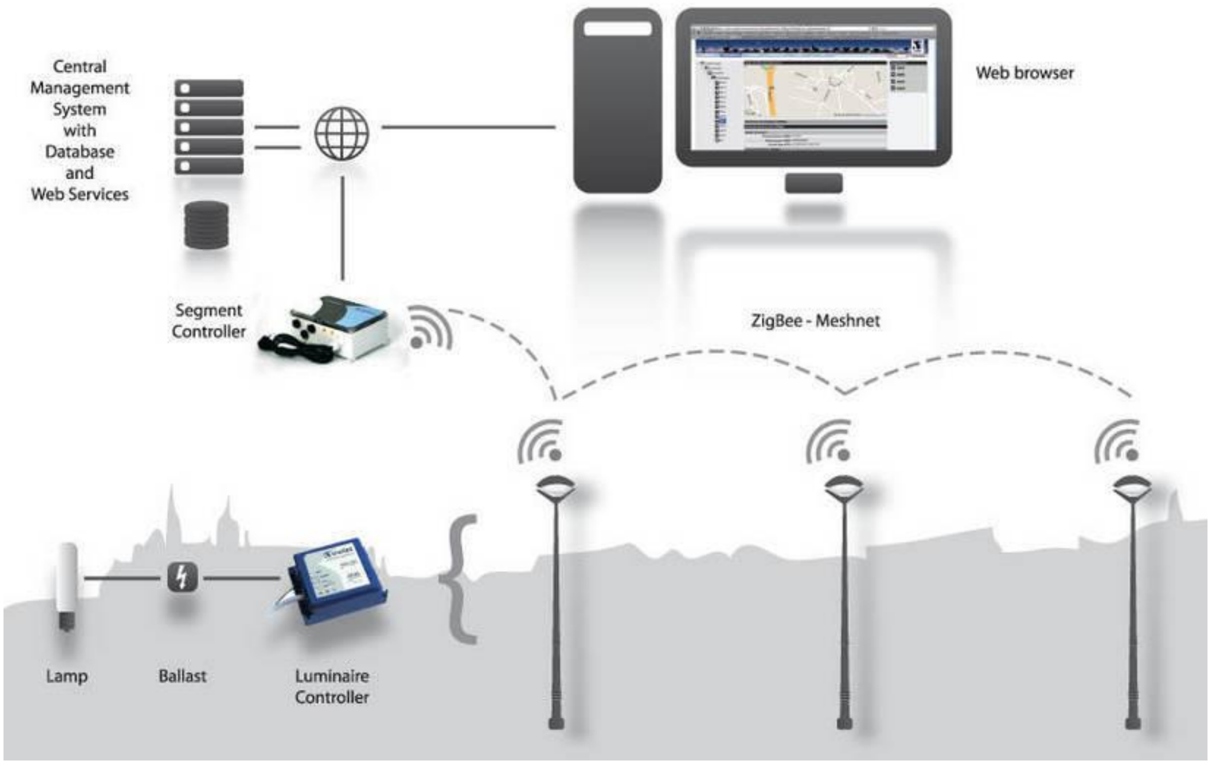


Fig 2.1 System showing wireless lighting control of street luminaire

Wireless connectivity are the blood vessels of smart lighting systems. The digital nature of led technology has enabled the usage of lighting products to become digital nodes and allows them to participate in IoT environment.

The figure 2.1 shows a basic outline of the concept of the operation and monitoring of smart street lighting. The control signal for wireless dimming control of street luminaires is being transmitted via ZigBee protocol.

2.2.1. Bluetooth lighting control of luminaires

Bluetooth is a wireless communication protocol which is used for transmission of short data packets between a fixed and mobile devices. The Bluetooth range is small which enables to create own Personal Area Network (PAN). A feature of smart lighting requires luminaires to be controlled digitally. The lighting control can be simple switching ON or switching OFF, as well as dimming control, thereby saving electrical energy as well as catering to the visual comfort of consumers. Bluetooth can be a simple but powerful tool for the above mentioned application.

In the work done by Yi Shuangshuang and others [6], an introduction to a

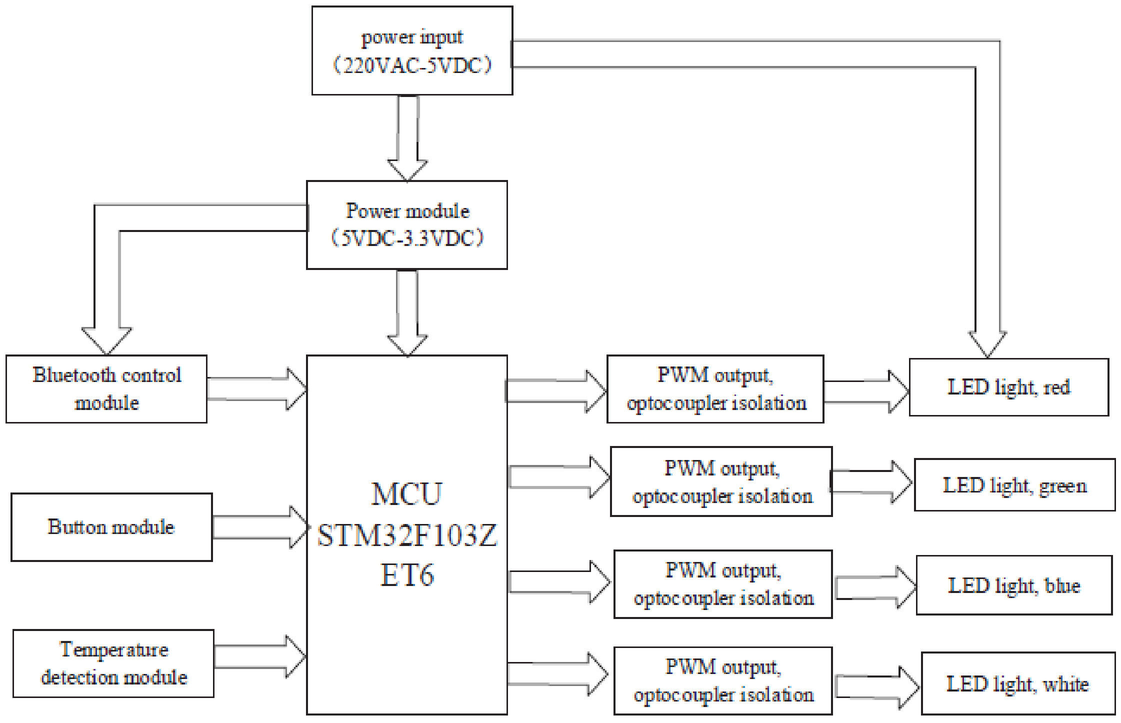

power input (220VAC-5VDC)
Power module (5VDC-3.3VDC)
Bluetooth control module
Button module
Temperature detection module
MCU STM32F103ZET6
PWM output, optocoupler isolation
PWM output, optocoupler isolation
PWM output, optocoupler isolation
PWM output, optocoupler isolation
LED light, red
LED light, green
LED light, blue
LED light, white

Fig 2.2 Block diagram of the system architecture

system which has used Bluetooth connection for the dimming control as well as push buttons for turning ON and turning OFF of luminaires has been done. The Bluetooth connects to mobile and through a user friendly mobile Application (App), which provides machine – human interface, thereby enabling lighting control of the luminaire.

The overall system architecture has been shown in Fig 2.2. From the block diagram it can be seen that additional Bluetooth module has been required to be interfaced with the microcontroller so as to provide connection with the Bluetooth. The microcontroller has been powered by a power module which has also powered the Bluetooth control module. Attached to this microcontroller unit are Button and Temperature detection modules. The output of the microcontroller provide the required pwm outputs for various types of LEDs through optical isolation.

The system has used the information sent by the user via the Android App , has processed it and based upon that it controls the switching or the dimming of the luminaire. The signals generated by the microcontroller have been used for switching / dimming of the respective led strips of different colours. The strength of the received packets have also been noted at the receiver end so that there is no loss of packets during transmission.

The dimming settings are such that in the android App, if the respective buttons are pressed, the LEDs will be dimmed to a fixed duty cycle.

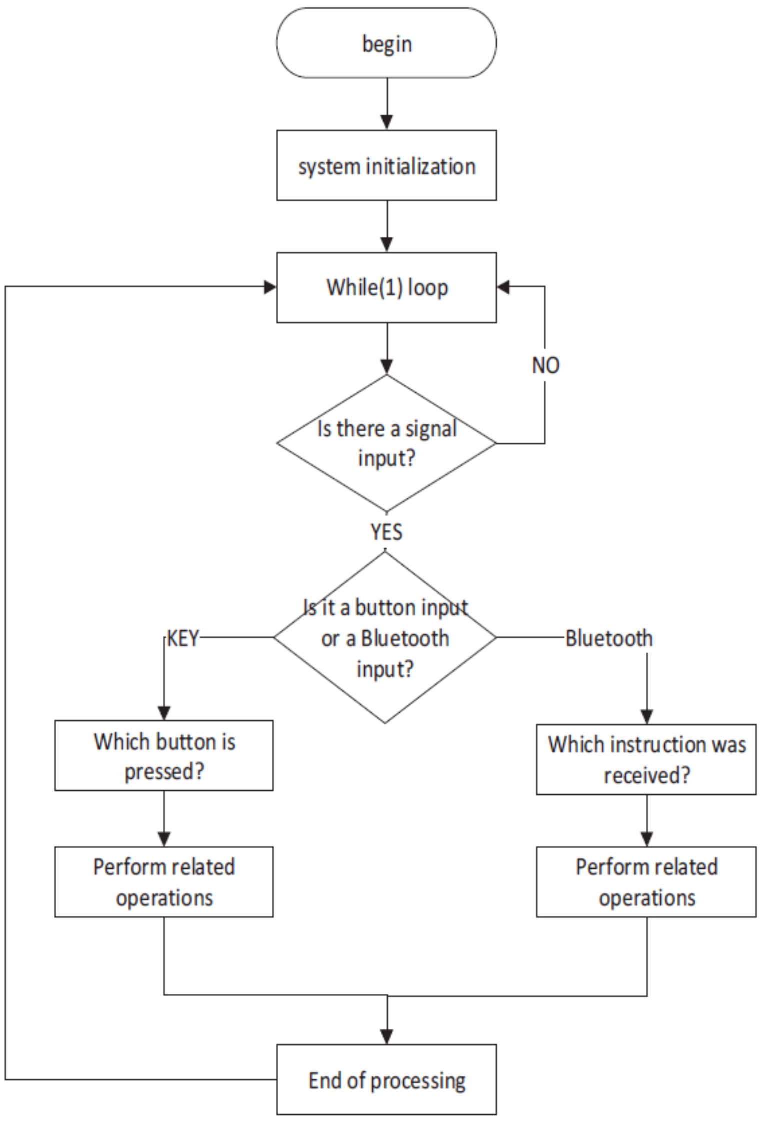

begin
system initialization
While(1) loop
NO
Is there a signal input?
YES
Is it a button input or a Bluetooth input?
KEY
Bluetooth
Which button is pressed?
Which instruction was received?
Perform related operations
Perform related operations
End of processing

Fig 2.3 Flowchart showing working of the entire system

The figure shown above represents a flowchart showing the working of the entire system. The flowchart shows after system initialization has been done, the microcontroller unit which has been connected to the LEDs detect weather there is an input signal or not. If there is an input signal, what kind of input signal is, i.e. signal from Bluetooth or button. The microcontroller then takes the decision for executing the further steps accordingly. If the input is from Bluetooth, the code has been set in a way that the microcontroller searches for which instruction has been received and then performs the related operations. If the input is from a button, it searches for which button has been pressed and then performs the related operations accordingly.

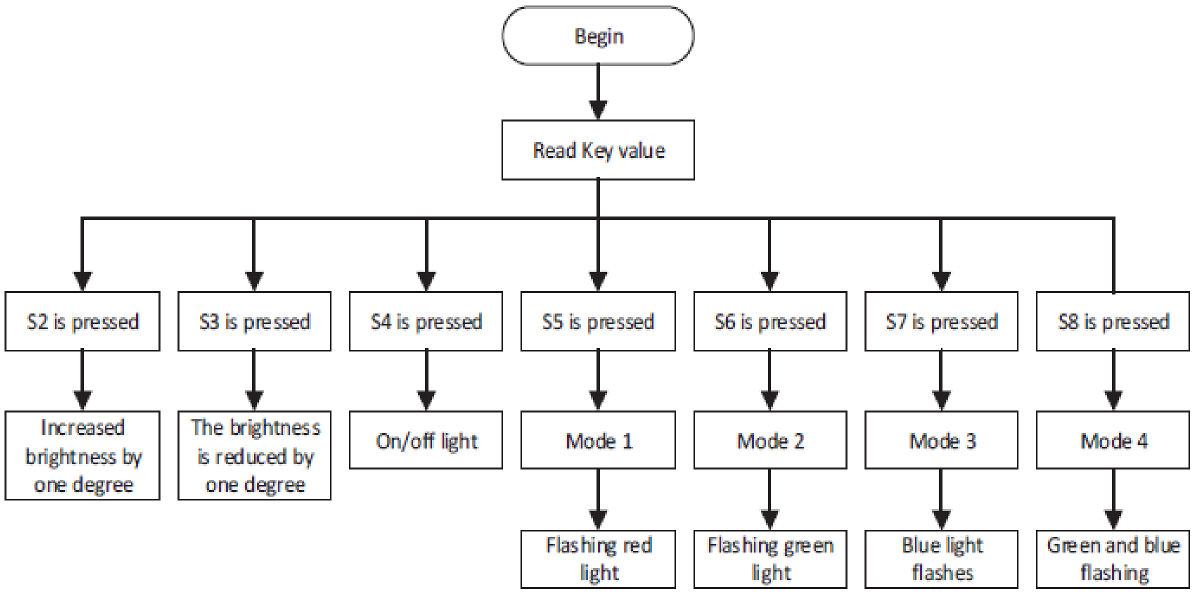


Fig 2.4 Key module block diagram

Figure 2.4 shows the block diagram for the logic of the Key module. The logic of the button module is very simple. When a Key has been pressed, the system detects which key has been pressed, and then it compares with the key value to make corresponding operation.

In the work done by Yu-Shan Cheng1 and others [7], Bluetooth control has been used for varying the dimming of RGB led strips.

The system also has used a Bluetooth module, connected with a microcontroller for dimming control of the respective led strips. The soft dimming requires that the pwm signals applied to the led strips are at least 2kHz so that the flickering of the LEDs are not noted. Here, three different sliders have also been introduced in the Android App interface so that the brightness of the three different led strips can be controlled. Here we have the feature of varying the duty cycles continuously for the dimming control of the led strips , according to one's needs.

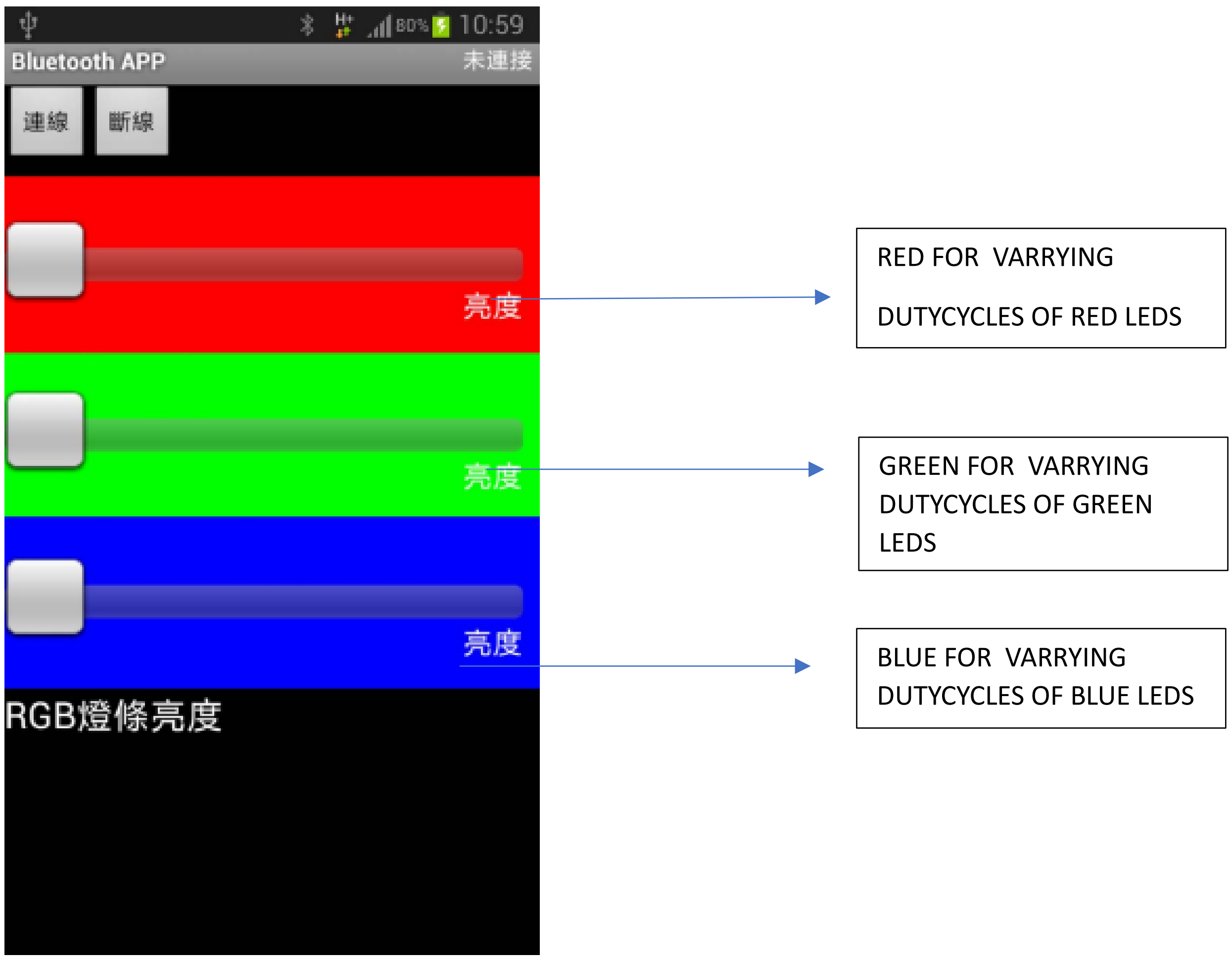

10:59
Bluetooth APP
未連接
連線
斷線
亮度
亮度
亮度
RGB燈條亮度
RED FOR VARRYING DUTYCYCLES OF RED LEDS
GREEN FOR VARRYING DUTYCYCLES OF GREEN LEDS
BLUE FOR VARRYING DUTYCYCLES OF BLUE LEDS

Fig 2.5 Display interface for controlling of the system

The figure shown above is the Android App with which is used for controlling the dimming levels of LEDs. The red slider has been used for varying the duty cycle of the red LEDs, the green slider has been used for varying the duty cycle of green LEDs and the blue slider has been used for varying the duty cycle of green LEDs.

The information of variation of the duty cycle of the has been sent to the microcontroller via the Bluetooth module. The microcontroller will have received the information about variation of duty cycles and accordingly will have provided the respective pwm signals to the respective LEDs for dimming control through optical isolation.

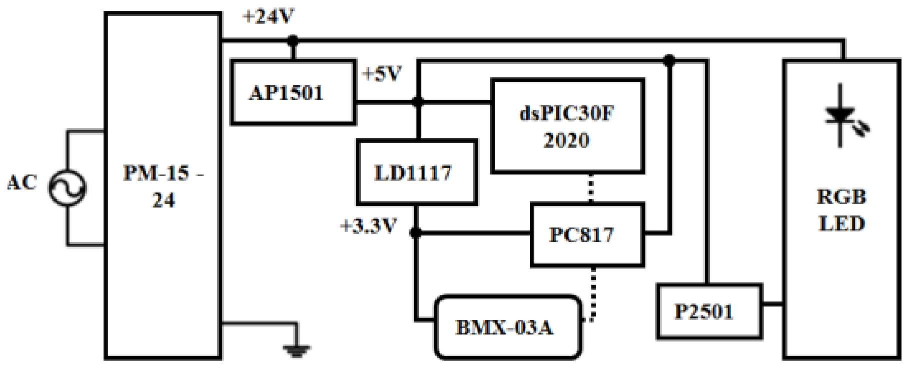


Fig 2.6 The block diagram of the proposed system architecture has been shown.

Since the input voltage is 110VAC, a power supply module PM-15-24 has been used to rectify and step down to 24VDC to provide driving voltage to LED. Then, IC AP1501 has been utilized utilized to further step down the voltage to 5V for supplying microcontroller and IC P2501. Also, IC LD1117 has been used to provide 3.3V to Bluetooth module BMX-03A. Photo-coupler PC817 has been employed to adjust the voltage level, which therefore can avoid the error during data transmission since the operating voltage of microcontroller and Bluetooth module is different. In this study, a low cost digital signal controller (DSC) dsPIC30F2020 from Microchip Corp. has been used to provide digital control for the LED dimming.

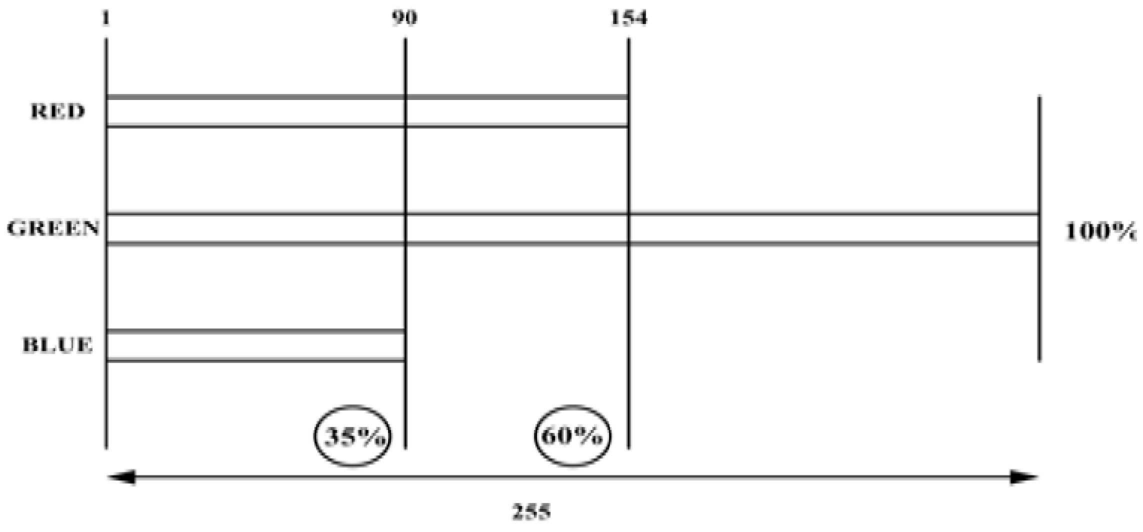


Fig 2.7 Dimming control method used

As shown in Fig. 2.7, timeline comparison method has been utilized to realize the dimming control. The dimming command has been compared with timeline and drives the LEDs at a high frequency. The entire bit range is 255 and has been shown in the figure 2.7, 90 has represented 35%, 154 has represented 60%, etc. for the RED, GREEN and BLUE LEDs.

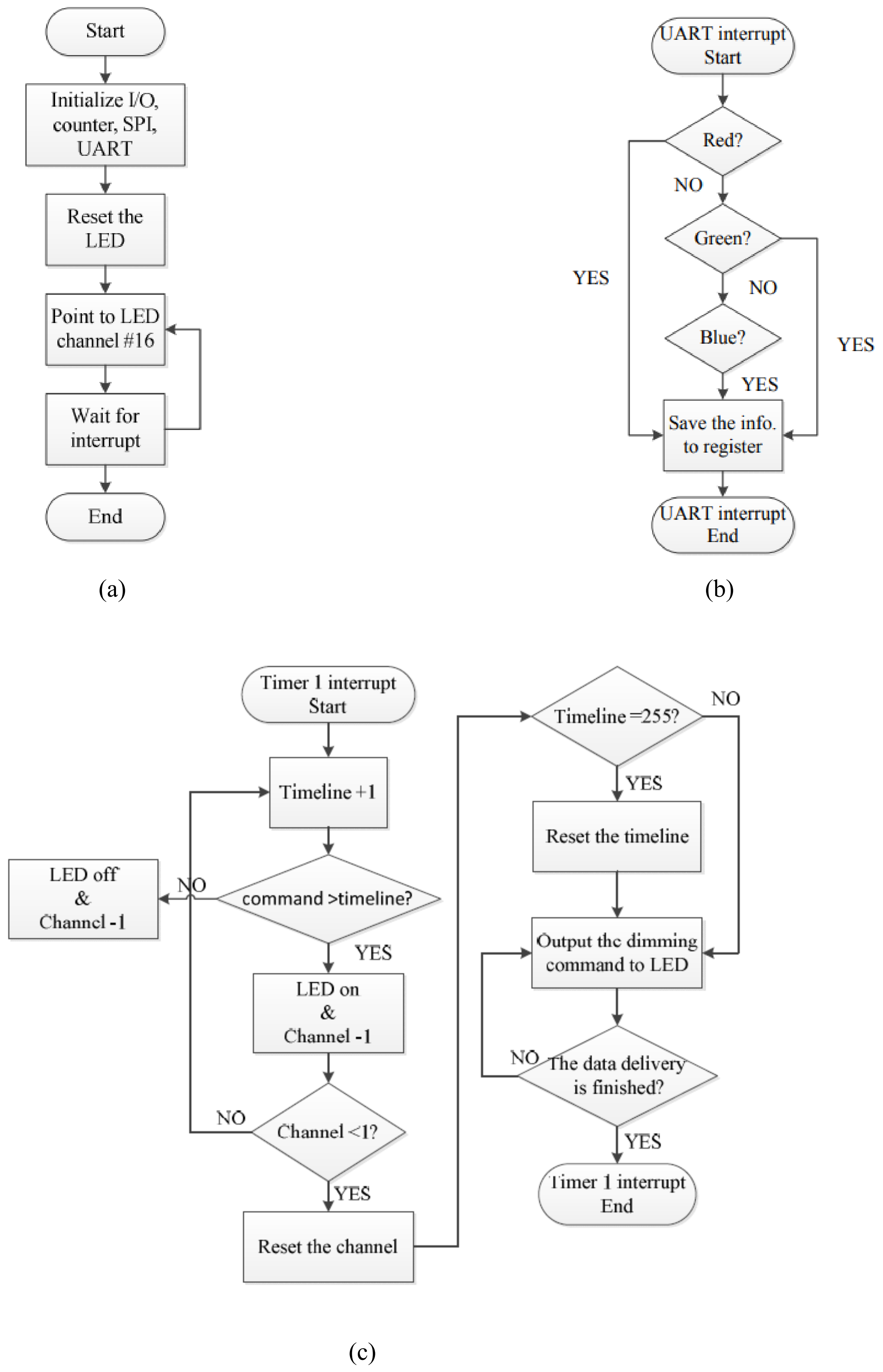


Fig 2.8 Flowcharts (a) , (b) and (c) showing the operation of the entire system

In Fig 2.8 a, the flowchart shows that initialization of the system has started by initialization of input,output devices, UART communication protocols, etc. In the second step , the resetting of the LEDs has been done. In the third step, the LED channel has been pointed. In the fourth step, the microcontroller has been waiting for the interrupt.

In Fig 2.8 b, the flowchart shows the process of functioning of interrupt. In the first step, the interrupt has been started. Then there is a verification of the colour, which has then been saved in the register.

In fig 2.8 c,the flowchart shows the process of functioning of the entire system.In the first step, Timer 1 has been started.In the second step, the timeline has been incremented by 1.In the third step, there has been comparison whether the command is greater than the timeline.If the result is NO, then the LED is switched OFF and channel is set to 1 , but if the output is YES, then the LED is switched ON and the channel is set to 1. In the next step, the channel is compared. If the channel is less than 1, then the channel is again incremented and the process iterates ,if not, then in the next step,channel is reset.In the next step, the timeline is compared to see weather it is equal to 255 or not.If YES, then the resetting of the timeline takes place.If NO,the dimming command will have been outputed to the LEDs. In the next step, the system has been checking, weather the data delivery isfinished or not.If the data delivery is finished, Timer 1 interrupt will have ended or else again the command will have been sent as an output to the LED.

2.2.2 Zigbee control of luminaires

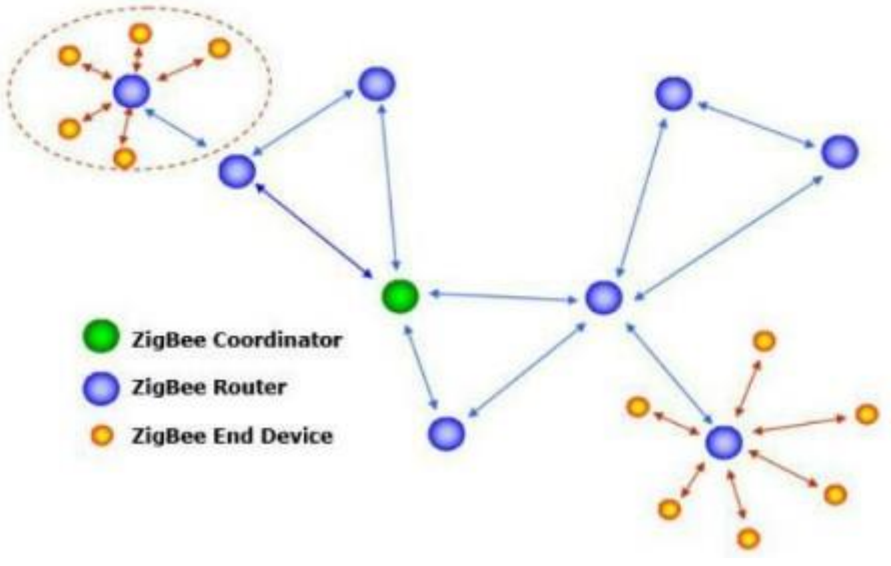


Fig 2.9 Figure showing the concept of ZigBee mesh networking

ZigBee is a mode of wireless communication allowing the central node to communicate with all other nodes for exchange of data. In this type of communication there is a central gateway, which actually coordinates the whole network. The gateway communicates with all the other nodes in the network which are identified by the same PAN ID. In the work done by Kuthsav Thattai and others , titled “ZigBee and ATmega32 Based Wireless Digital Control and Monitoring System For LED Lighting ”[8], a system has been proposed in which a microcontroller, ATMEGA32, has been connected to a ZigBee module and is used to receive data , sent from the transmission end and is used for controlling the lighting of luminaire.

On the transmission side, a PC has been connected to a ZigBee Gateway device and GSM Modem via its serial port. Also the led node sends a system failure notification on the PC.

The system has been designed to work in two modes

- Manual Mode
- Time Based Automatic Control (TBAC) Mode.

In the basic Manual Mode the user can use the buttons and track bar provided in Graphical User Interface (GUI). Part of the GUI Basic Manual Mode has been shown in Fig.2.10. When a button or track bar is used corresponding command is sent to the Control Device from the Operator Station. When the Operator Station has received failure signal from the Control Device, the 'Failure Indicator' (Red Dot) on the GUI will have been highlighted.
Control Command corresponding to the control of LIGHT1 and PWM duty cycle has been shown in TABLE 2.1. For instance after switching on the LED Light by pressing the ON button, if track bar is moved to position four, "LIGHT1$4" command is transmitted to the Control Device. On receiving this command microcontroller will update its register so as to generate a PWM with 40% duty cycle.

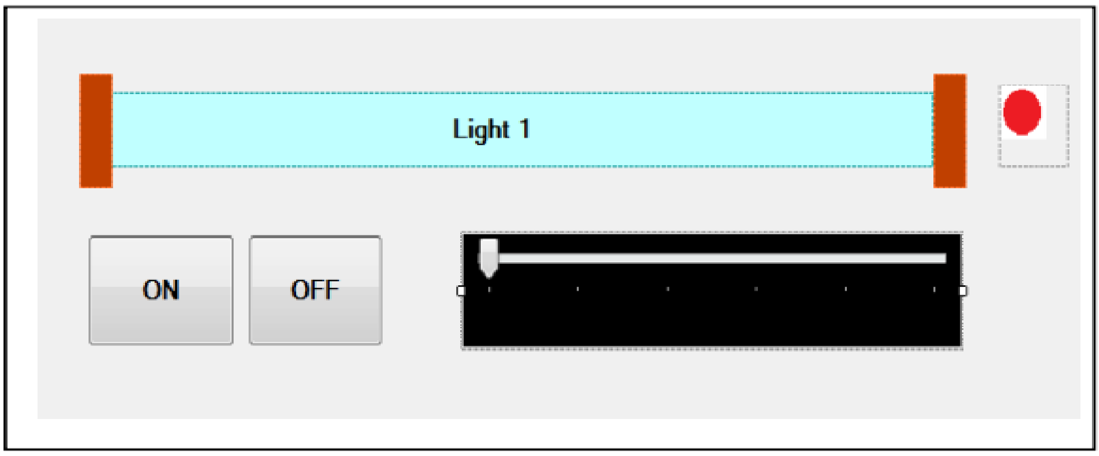
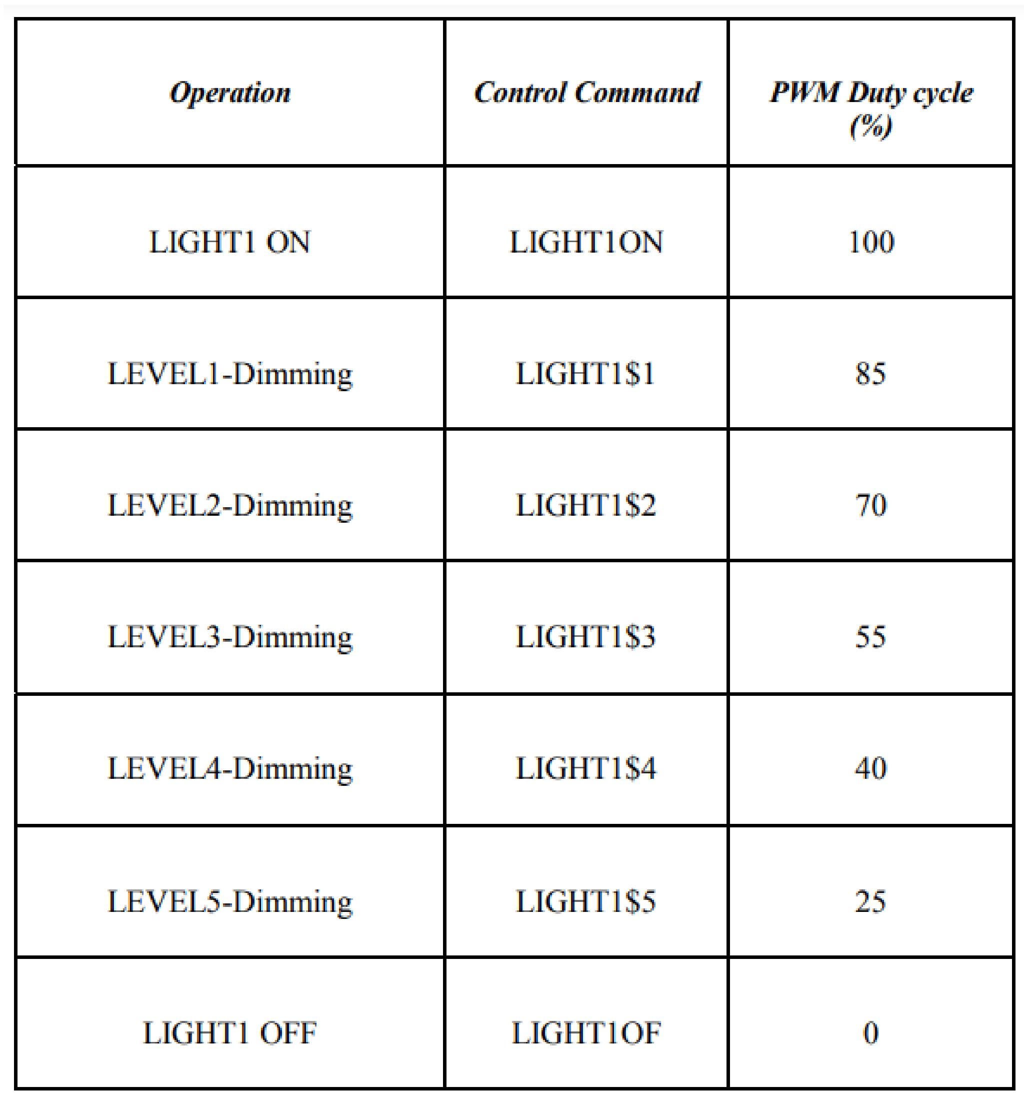
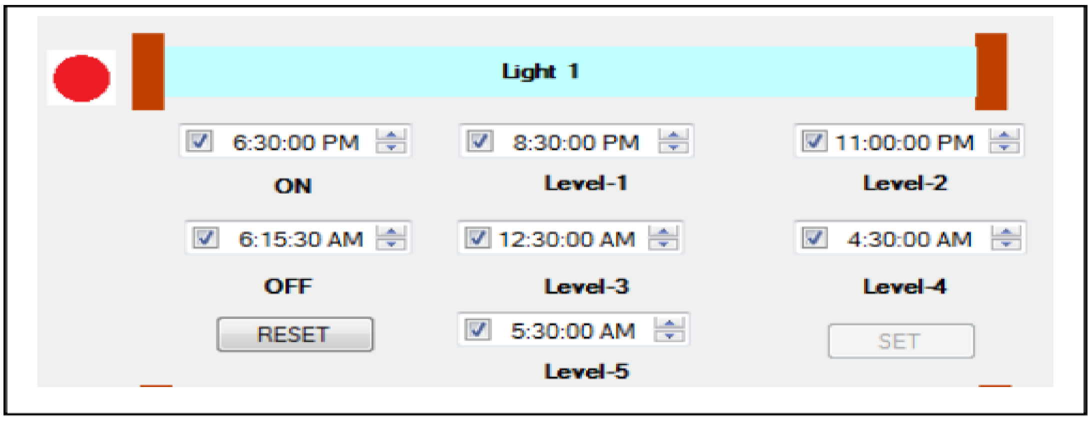


Fig 2.10 Figure showing the display interface for lighting control for manual mode

Table2.1 Table showing the duty cycle setting for wireless dimming controlling of the luminaire



Operation	*Control Command*	*PWM Duty cycle (%)*
LIGHT1 ON	LIGHT1ON	100
LEVEL1-Dimming	LIGHT1$1	85
LEVEL2-Dimming	LIGHT1$2	70
LEVEL3-Dimming	LIGHT1$3	55
LEVEL4-Dimming	LIGHT1$4	40
LEVEL5-Dimming	LIGHT1$5	25
LIGHT1 OFF	LIGHT1OF	0



Light 1

☑ 6:30:00 PM	☑ 8:30:00 PM	☑ 11:00:00 PM
ON	Level-1	Level-2
☑ 6:15:30 AM	☑ 12:30:00 AM	☑ 4:30:00 AM
OFF	Level-3	Level-4
RESET	☑ 5:30:00 AM	SET
	Level-5	

Fig 2.11 Figure showing the status of dimming level of luminaire at various instances of the day for TBAC.

In TBAC type of control one can set the time for each operation. Once the time is set, the Operator Station will automatically send the corresponding control command at that time. At any instance if the Operator Station receives any failure signal, it passes on the information to the user via SMS. Fig. 2 11 shows the GUI Control used in TBAC. Time for different operations can be entered in their respective fields. The preset time can be reset by pressing the ‘RESET’ button given in GUI.

In the work done by Báez C. Rolando and others, [9], a personal area network has been created by the ZigBee protocol as well as running the luminaire control app on the mobile phone. Here the duty cycle percentages are communicated through the mobile app to the Bluetooth module, connected to the microcontroller. The microcontroller receives the information via UART protocol and transmits the same via ZigBee module which is also connected to it. The ZigBee module searches the devices in the network connected and transfers the information to these nodes. The nodes are then used for dimming of luminaires connected to them. The functioning of the entire system has been shown in Fig 2.12

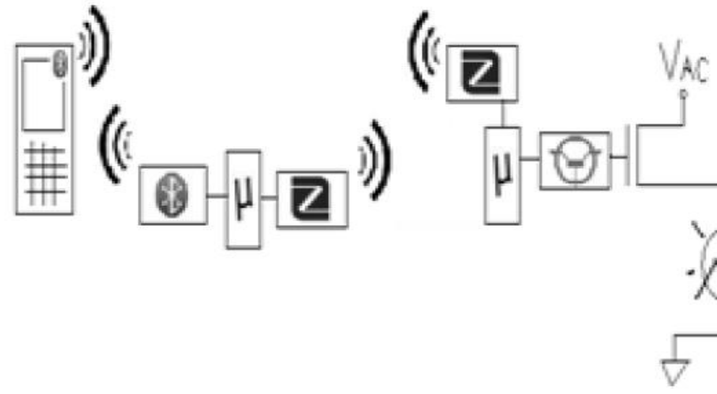


Fig 2.12 Figure showing the operation of the entire system

2.2.3. Wifi control of luminaires

In the work by Kok-Hua Teng and others [10], provides an interface for android mobile application based controlling the dimming levels using wifi protocol. The block diagram of the system has been shown in fig. 2.13

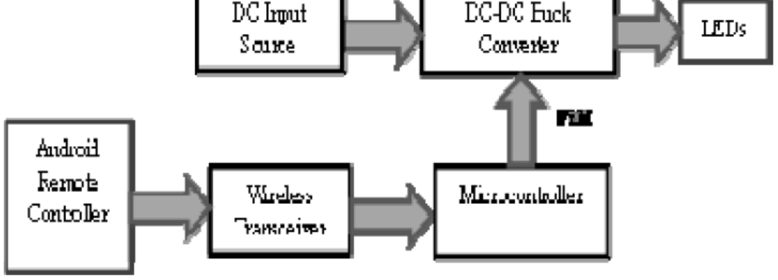


Fig2.13 Block diagram showing the working of the entire system.

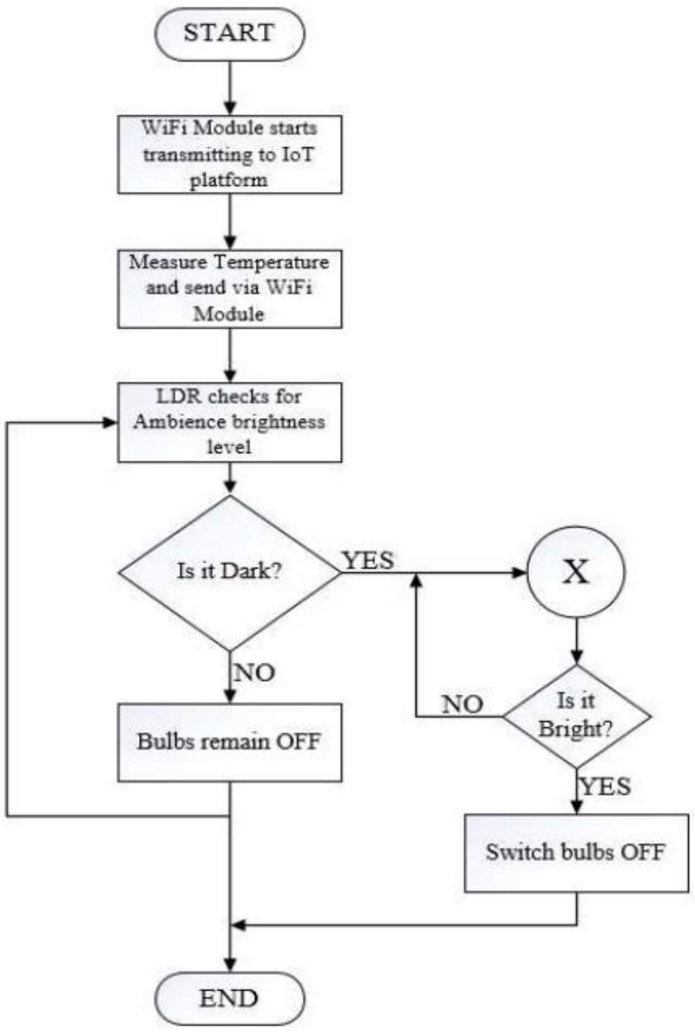
In the block diagram, the signal ,generated by the remote controller has been sent wirelessly to the microcontroller. The signal may be a simple switch ON or OFF, or it may be pwm signal, controlling the variation of the duty cycle. The wireless transfer protocol which has been used is the WiFi. The microcontroller receives the signals and with the help of a DC-DC buck converter powers and controls the lighting of the LEDs. The DC-DC buck converter has been powered by a dc input source.

2.3Obstacle detection:

In modern day, the topic of energy savings is of great concern due to the rapidly increasing energy demand. Street lights alone are accountable for 19% energy consumption in the whole world[9]. In this era of automation, development of energy efficient automated systems is one of our main concerns.

Work done by Tobiloba Somefun & others [11] have been shown the concept of energy savings by dimming of LEDs in smart street lights. This was done by dimming the light levels during different instances of the night, as well as only lighting with full intensity when a vehicle has been detected.
The work has been achieved by a Light Dependent Resistor (LDR) as well as a Passive Infra-Red (PIR) sensor. The LDR senses the ambient brightness and accordingly with the time of the day, given by a Real Time Clock (RTC) module, switches on the luminaire with 100% intensity or less. The PIR sensor detects a vehicle when it is under the street luminaire and only then switches on the luminaire with full intensity at night otherwise it is dimmed at 40% at night.

The entire system operation has been shown. The microcontroller responsible for dimming the leds has been connected to a wifi module so that we are able to monitor the current state of the street luminaire on the internet. The operation with the PIR sensors has been activated activated only during the night.



START
WiFi Module starts transmitting to IoT platform
Measure Temperature and send via WiFi Module
LDR checks for Ambience brightness level
Is it Dark?
YES
X
NO
Bulbs remain OFF
NO
Is it Bright?
YES
Switch bulbs OFF
END

Fig2.14(a)

The Fig 2.14 (a) shows the flowchart for the operation of the system. After starting, the first process box shows that the WiFi Module has started transmitting data to the IoT platform. The second step shows that temperature measured has been sent as transmitted data to the IoT platform. The Light Dependent Resistor (LDR) has been checking for ambience brightness level. In the next step, decision box, based on the readings taken by LDR, darkness is detected. If darkness is not present, the bulbs will have been switched OFF, and the LDR will take further readings. If darkness has been detected, then, X procedure will get executed. After execution of the X procedure has been done, the level of brightness will have been checked by the LDR. Until rightness level has been reached, procedure X will haven been repeated. When brightness will have been detected, bulbs will have been switched OFF.

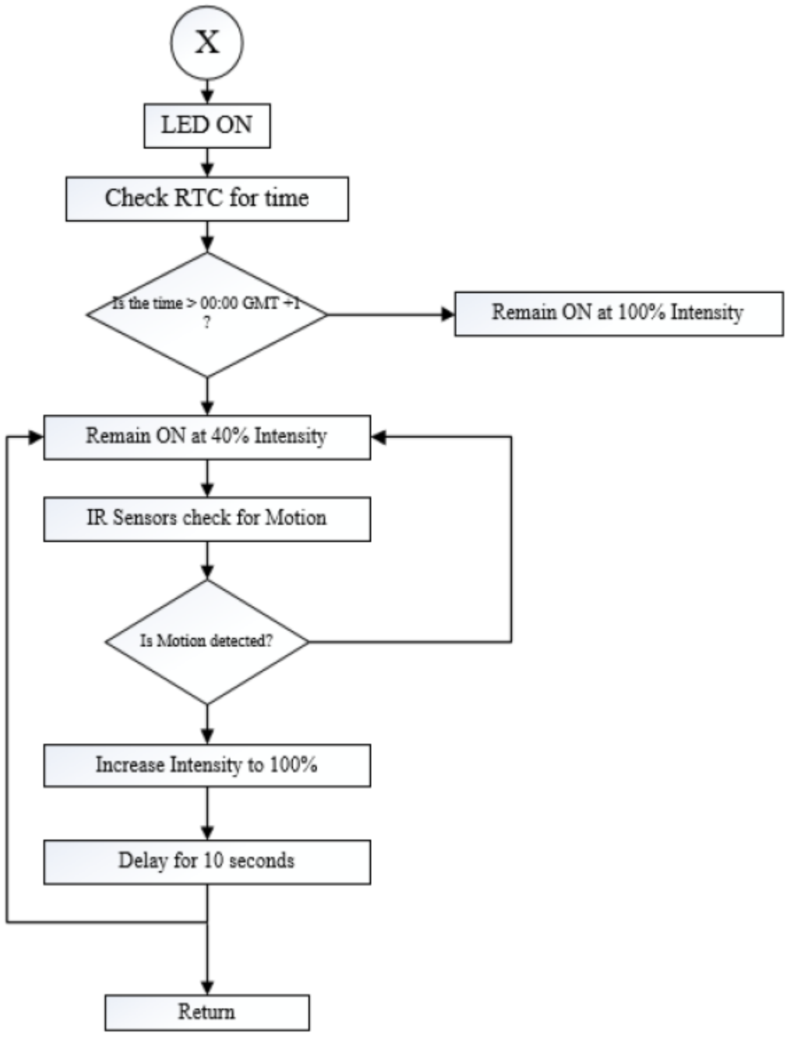

X
LED ON
Check RTC for time
Is the time > 00:00 GMT +1 ?
Remain ON at 100% Intensity
Remain ON at 40% Intensity
IR Sensors check for Motion
Is Motion detected?
Increase Intensity to 100%
Delay for 10 seconds
Return

Fig 2.14(b)

Fig 2.14 Figure showing the working of the entire system

In Fig 2.14(b), procedure X has been elaborately explained. As the LED has been turned ON, time has been checked by the Real Time Clock (RCT) Module. If the time has been midnight, the LEDs have been turned ON at 100% intensity. If the motion has been detected by the Infra Red (IR) sensor, only then, the LEDs will have remained ON at 100% ,but if the motion has not been detected, LEDs will have remained ON at 40%.

In the work done by Bhaavan Sri Sailesh A and others [12], the Ultra Sonic sensors are used to measure the distance of the object in front of it. This is how it is used to detect a vehicle so as to perform the switching action of the luminaire.

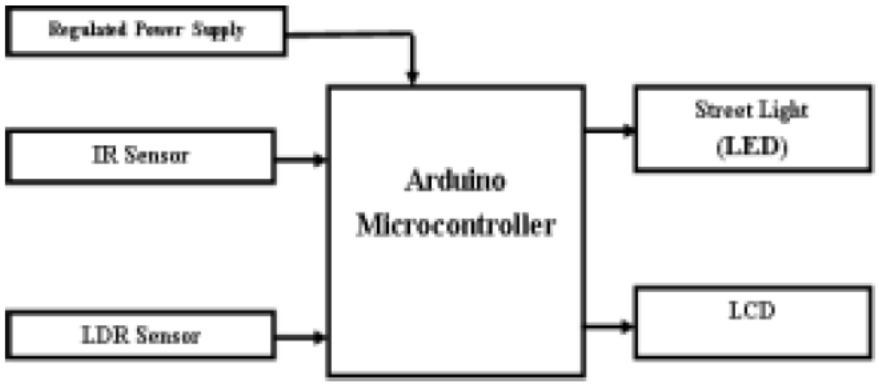


Fig 2.15 block diagram of the above system is shown

The above block diagram shows the working of the Arduino based smart lighting system. The various sensors connected to the microcontroller are the Light Dependent Resistance (LDR) sensor, InfraRed(IR) sensor. A Liquid Crystal Display (LCD) is connected to the microcontroller to show the parameters required. The Arduino microcontroller is powered by a regulated power supply with a DC voltage compatible for powering the microcontroller. The output of the microcontroller is connected to the street light which is LED.

3.METHODOLOGY

3.1 OVERVIEW:

The distance from two remote locations have been observed and collected using sensors and they are being sent wirelessly to the master controller which receives the information and accordingly switches the street lights on wirelessly. The street lights are located at different locations and they will get switched on and off at the same time.

In our work, these street lights will get switched on at some specific duty-cycle when the distance coming from the sensor will be below a threshold value. But when the distance will be above that threshold value, a feature will be enabled which will allow us to wirelessly vary the brightness of the street lights via a webpage, consisting of two sliders to vary the duty cycles via pulse-width-modulation (pwm) technique. The block diagram as well as the flowchart of the working of the entire system has been shown in Fig 3.1 and Fig 3.2 respectively.

The performance of the street luminaire is observed and are recorded.

Hence, our work can be divided into three parts:

1. Obstacle detection Unit for measuring, recording and transmitting the data(distance).
2. Master-Controller Unit which is used to receive the transmitted data and transmit pwm duty-cycle signals to street lights for their dimming purpose, based upon some condition. This unit is the heart of our system.
3. Receiver Unit which is used to receive the pwm data(dutycycle) from the Master controller and use them to dim or brighten street luminaire.

3.2 DESIGN OF SYSTEM:

BLOCK DIAGRAM OF SYSTEM

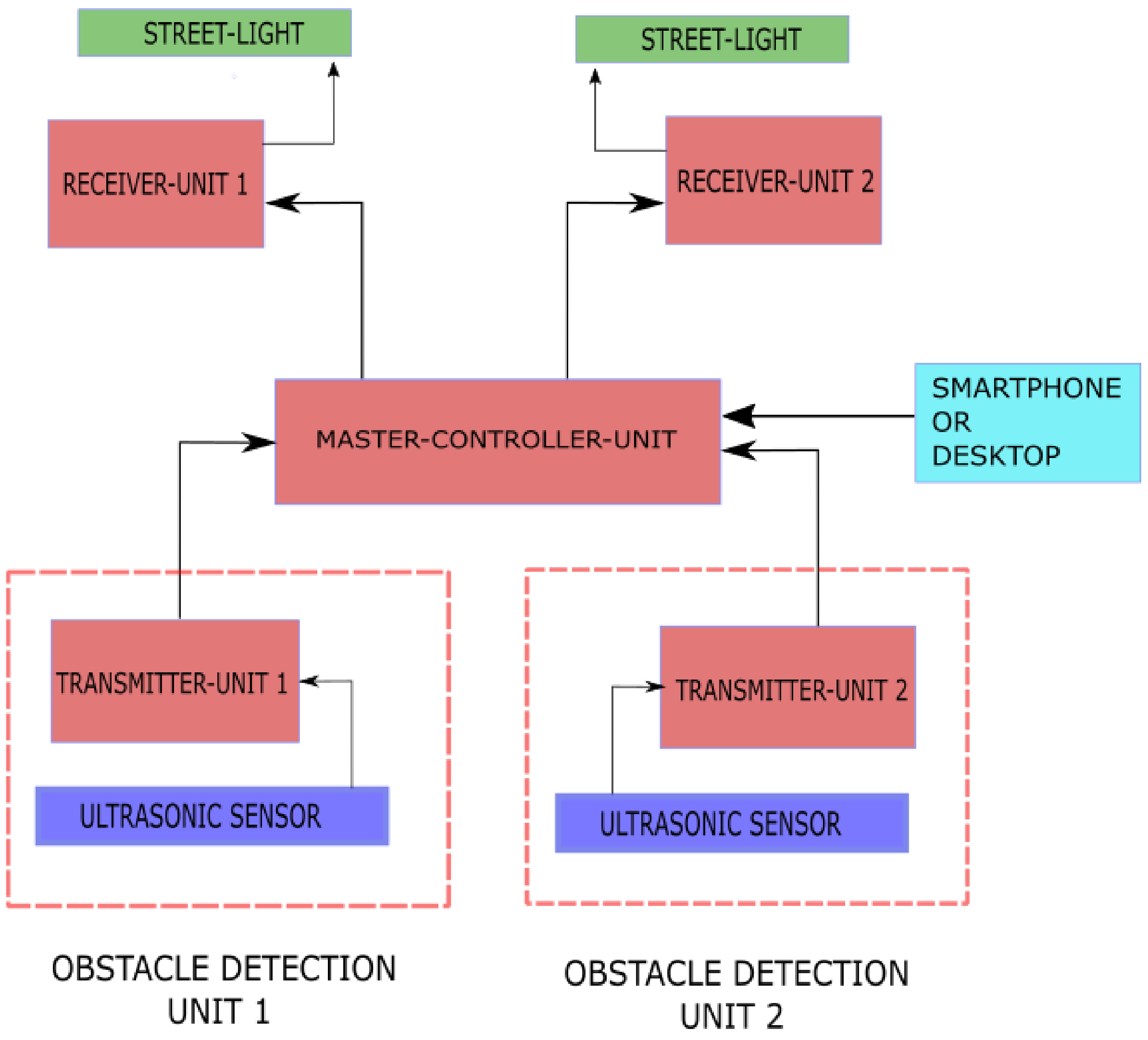


Fig 3.1

Figure showing the operation of the entire system in block diagram

Fig3.1 shows the block diagram of the entire system. The ultrasonic sensors along with the transmitter units make up the Obstacle Detection Unit. The sensors measure the distances sensed and send them wirelessly to the master controller.

The Master Controller Unit is the heart of the entire system which provides access to a webpage consisting of two sliders for changing of duty cycles for pwm brightness control. It also receives the distances from the two Obstacle detection units. The data coming from the Obstacle Detection Units are received and processed here.

The Receiver Units are attached to the street lights and receive the information sent wirelessly by the Master Controller Unit.

3.3FLOWCHART OF THE EXPERIMENT

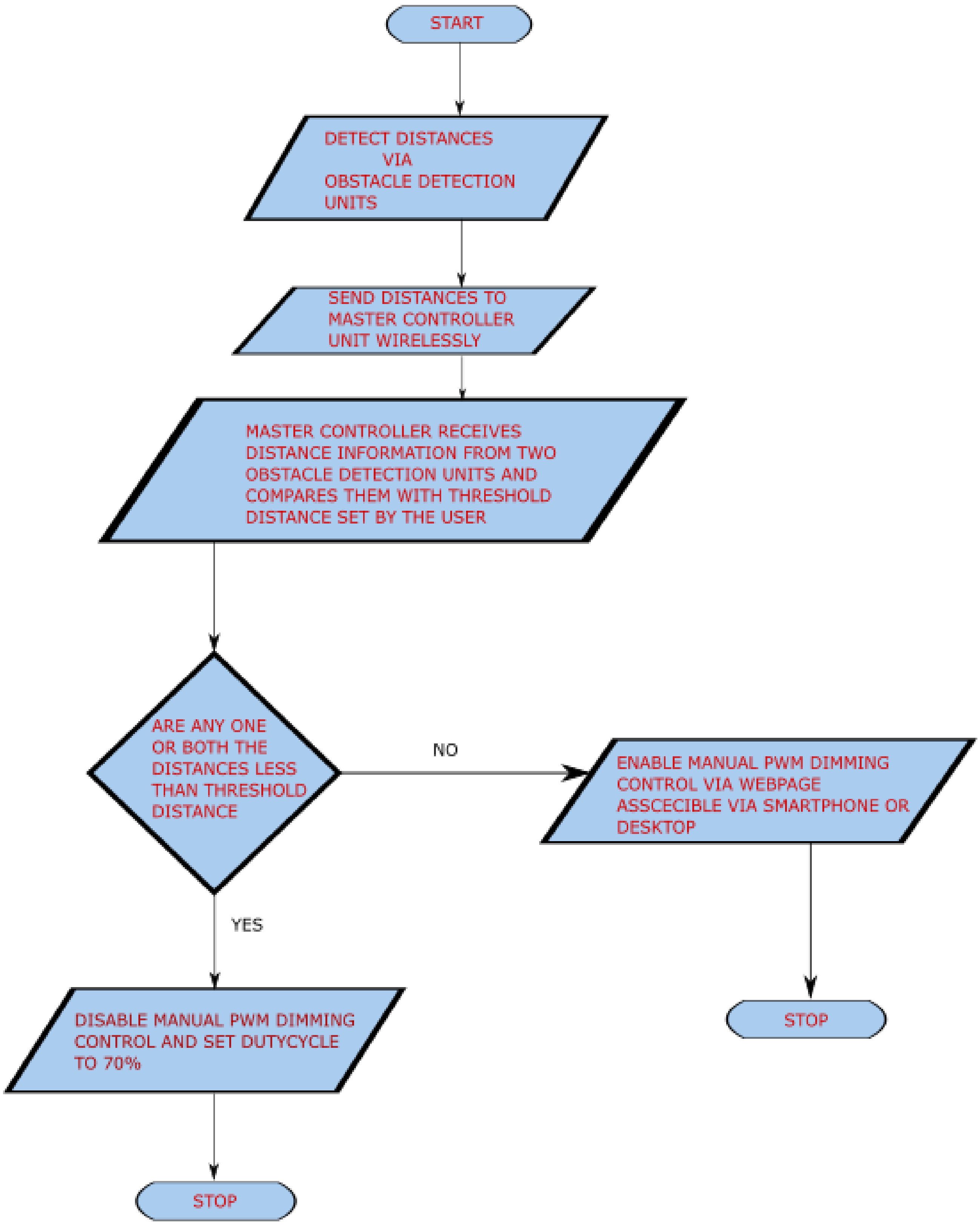

START
DETECT DISTANCES VIA OBSTACLE DETECTION UNITS
SEND DISTANCES TO MASTER CONTROLLER UNIT WIRELESSLY
MASTER CONTROLLER RECEIVES DISTANCE INFORMATION FROM TWO OBSTACLE DETECTION UNITS AND COMPARES THEM WITH THRESHOLD DISTANCE SET BY THE USER
ARE ANY ONE OR BOTH THE DISTANCES LESS THAN THRESHOLD DISTANCE
NO
ENABLE MANUAL PWM DIMMING CONTROL VIA WEBPAGE ASSCECIBLE VIA SMARTPHONE OR DESKTOP
STOP
YES
DISABLE MANUAL PWM DIMMING CONTROL AND SET DUTYCYCLE TO 70%
STOP

Fig 3.2 Figure showing the flowchart for the operation of the entire system developed.

Fig3.2 shows the flowchart of the entire system operation. The first process box (parallelogram box) shows that the distances are being detected and measured by the obstacle detection units. The obstacle detection units consists of ultrasonic sensors which are measuring the distances .

The second process box shows that after the detection have been done by the sensors of the obstacle detection units, the distances are being sent wirelessly to the Master Controller Unit.

The third process box shows that the distance data are being received by the Master Controller Unit and are being compared with the threshold distance which has been set by the user continuously .

In the decision box (diamond box) , the decision is being made , based upon comparison that if the distances coming from any one or both the Obstacle Detection Units are less than the threshold distance.

If the result of the decision box is no then fifth process box showing activation of manual control of pwm sliders which has been accessible through webpage in our laptop or mobile screen is done.

Other wise, the fifth process box showing deactivation of the manual control of pwm dimming is done, thereby setting the duty cycle to a constant value of 70% is done.

3.4 COMMUNICATION PROTOCOLS USED IN SYSTEM

3.4.1UART COMMUNICATION PROTOCOL:

Universal Asynchronous Receive Transmit (UART) or Serial communication is one of the most simple communication protocols between two devices. It transfers data between devices by connecting two wires between the devices, one is the transmission line while the other is the receiving line. The data transfers bit by bit digitally in form of bits from one device to another. The main advantage of this communication protocol is that its not necessary for both the devices to have the same operating frequency. For example, two microcontrollers operating at different clock frequencies can communicate with each other easily via serial communication. However, a predefined bit rate that is referred to as baud rate usually set in the flash memory of both microcontrollers for the instruction to be understood by both the devices.

The transmitting UART takes bytes of data and transmits the bits in a sequential form. The second transmitter which is the receiver reassembles the bits into a complete byte. Serial transmission of data through a single wire is actually more cost-effective than parallel transmission through multiple wires.

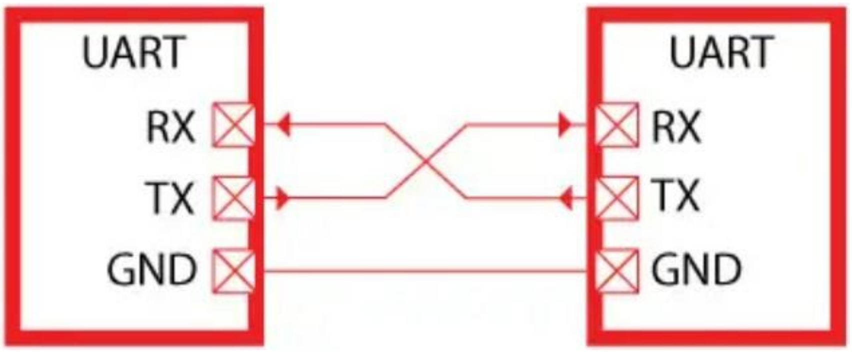


Fig 3.3 Connection diagram for UART communication protocol

It takes two UART's to communicate directly with each other. On one end the transmitting UART converts parallel data from a CPU into serial form then transmits the data in serial form to the second UART which will receive the serial data and convert it back into parallel data. This data can then be accessed from the receiving device.

Instead of cloak signals the transmitting and receiving bit use start and stop bit signals for the data packages. These start and stop bits define the beginning and the end of the data packages. Therefore the receiving UART knows when to start and stop reading the bits.

The Receiving UART will detect the start bit then start reading the bits. The specific frequency used to read the incoming bits is known as the baud rate. The baud rate is a measure used for the speed of data transfer. The unit used for baud rate is bits per second (bps). In order for the data transfer to be a success both the transmitting and receiving UART must operate at almost the same baud rate. However, if the baud rates differ between both UARTs they must only differ by 10%. The receiving and transmitting UART must be configured to receive the same data packages.

3.4.2ESP-NOW COMMUNICATION PROTOCOL

ESO-NOW is a protocol developed by Espressif, which enables multiple devices to communicate with one another without using Wi-Fi. The protocol is similar to the low-power 2.4GHz wireless connectivity. The pairing between devices is needed prior to their communication. After the pairing is done, the connection is safe and peer-to-peer.

This means that after pairing a device with each other, the connection is persistent. In other words, if suddenly one of your boards loses power or resets, when it restarts, it will automatically connect to its peer to continue the communication.

ESP-NOW supports the following features:

- Encrypted and unencrypted unicast communication;
- Mixed encrypted and unencrypted peer devices;
- Up to 250-byte payload can be carried;
- Sending callback function that can be set to inform the application layer of transmission success or failure.

ESP-NOW technology also has the following disadvantages:

- Limited encrypted peers. 10 encrypted peers at the most are supported in Station mode; 6 at the most in SoftAP or SoftAP + Station mode;
- Multiple unencrypted peers are supported, however, their total number should be less than 20, including encrypted peers;
- Payload is limited to 250 bytes.

To communicate via ESP-NOW, knowledge of the MAC Address of the receiver is required , which is unique to each device. That's how one knows to which device data will be sent to.

MAC Address stands for Media Access Control Address and it is a hardware unique identifier that identifies each device on a network.MAC Addresses are made up of six groups of two hexadecimal digits, separated by colons, for example: 30:AE:A4:07:0D:64. MAC Addresses are assigned by manufacturers.

The basic flowchart of the entire system is divided into three:

a) FLOWCHART FOR GETTING MAC ADDRESS
b) FLOWCHART FOR TRANSMITTER
c) FLOWCHART FOR RECEIVER

3.4.2.1 FLOWCHART FOR GETTING MAC ADDRESS :

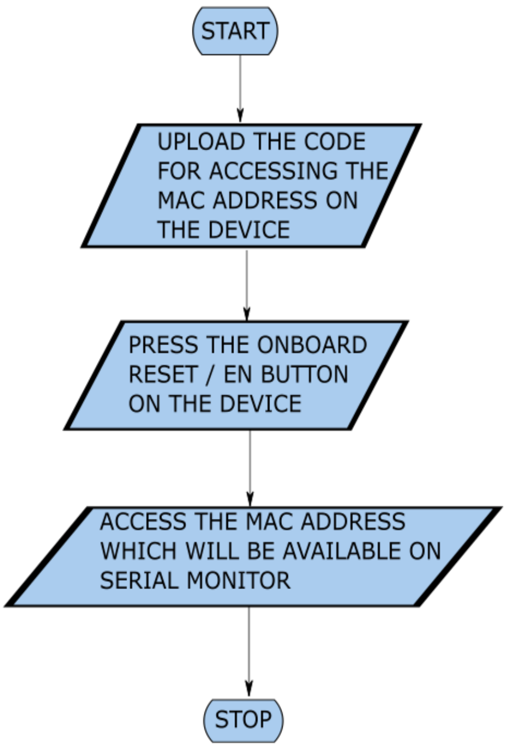

START
UPLOAD THE CODE FOR ACCESSING THE MAC ADDRESS ON THE DEVICE
PRESS THE ONBOARD RESET / EN BUTTON ON THE DEVICE
ACCESS THE MAC ADDRESS WHICH WILL BE AVAILABLE ON SERIAL MONITOR
STOP

Fig3.4 Flowchart for the accessing the MAC Address

Fig 3.4 shows the flowchart for accessing the MAC address of the microcontroller. The first process box is showing the process of uploading the code on the microcontroller, the second process box is showing that after the code has been uploaded on the microcontroller the reset or the enable button has to be pressed. The third process box is showing that after the reset or enable button has been pressed, the MAC Address will be accessible on the serial monitor of the monitor screen.

3.4.2.2 FLOWCHART FOR TRANSMITTER:

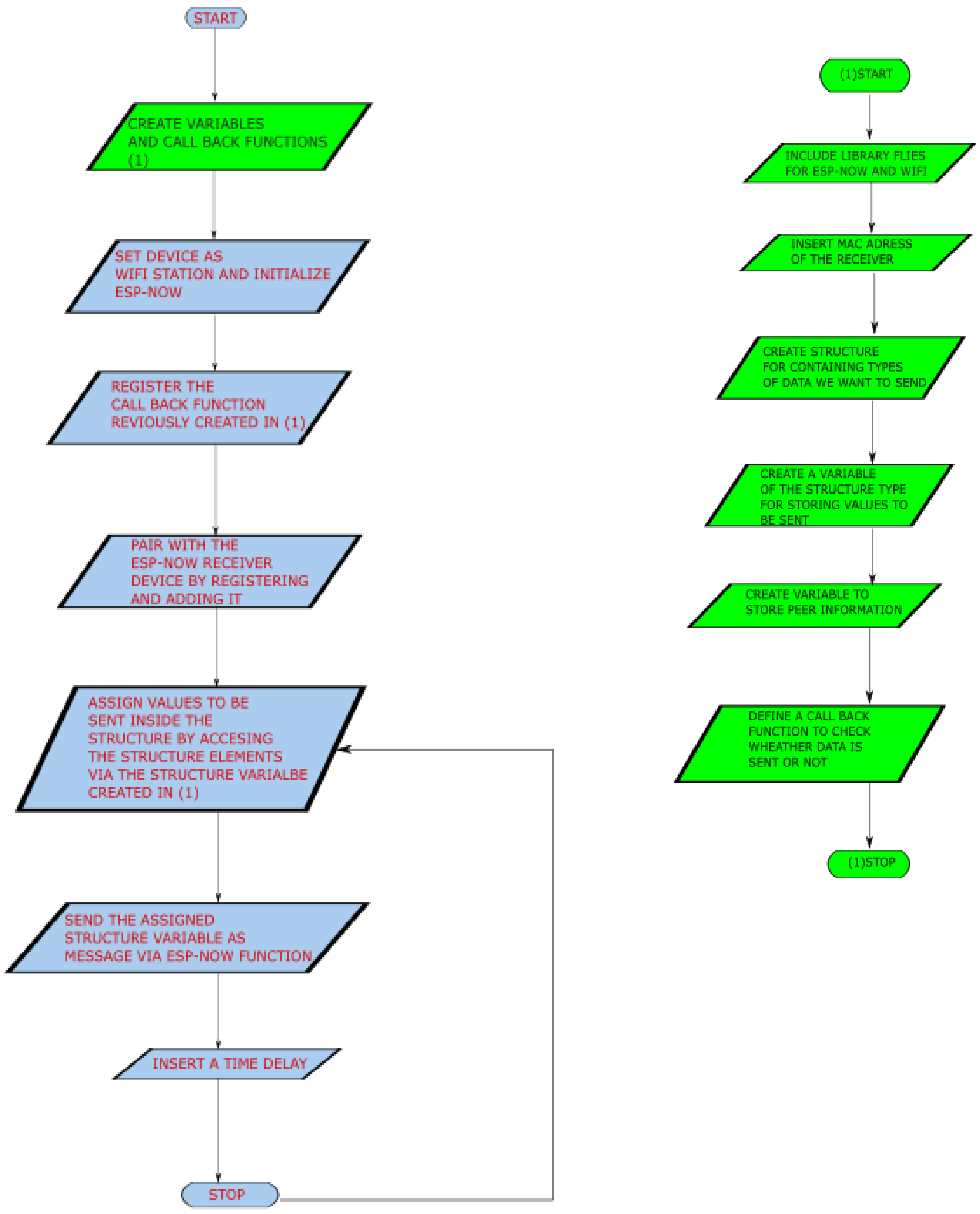

START
CREATE VARIABLES AND CALL BACK FUNCTIONS (1)
SET DEVICE AS WIFI STATION AND INITIALIZE ESP-NOW
REGISTER THE CALL BACK FUNCTION REVIOUSLY CREATED IN (1)
PAIR WITH THE ESP-NOW RECEIVER DEVICE BY REGISTERING AND ADDING IT
ASSIGN VALUES TO BE SENT INSIDE THE STRUCTURE BY ACCESING THE STRUCTURE ELEMENTS VIA THE STRUCTURE VARIALBE CREATED IN (1)
SEND THE ASSIGNED STRUCTURE VARIABLE AS MESSAGE VIA ESP-NOW FUNCTION
INSERT A TIME DELAY
STOP
(1)START
INCLUDE LIBRARY FLIES FOR ESP-NOW AND WIFI
INSERT MAC ADRESS OF THE RECEIVER
CREATE STRUCTURE FOR CONTAINING TYPES OF DATA WE WANT TO SEND
CREATE A VARIABLE OF THE STRUCTURE TYPE FOR STORING VALUES TO BE SENT
CREATE VARIABLE TO STORE PEER INFORMATION
DEFINE A CALL BACK FUNCTION TO CHECK WHEATHER DATA IS SENT OR NOT
(1)STOP

Fig3.5 Flowchart for the working of transmitter

Fig 3.5 shows the flowchart for working of the transmitter .In the first process box, which is green, all the variables and the call back functions ,required for the working of the protocol have been defined or declared.

The inbuilt library files for esp-now and wifi are called, the MAC Address of the receiver is inserted. A structure containing various types of variables is created and a variable of the structure type is created for accessing the structure.A variable for storing the peer information has also been created.

In the second process box, which is blue, the microcontroller has been set as a wifi station esp-now has been initialized.

In the third process box, the call back functions created earlier are registered.

In the fourth blue process box, the transmitter microcontroller has been paired with the receiver microcontroller using MAC Address of the receiver microcontroller.

In the fifth process box, the data to be sent have been assigned inside the structure by accessing the structure elements through the structure variable created initially.

In the sixth process box, the assigned structure containing the data to be sent has been sent as a message via the esp-now protocol.

In the seventh process box. A small time delay has been inserted .

The sixth and seventh steps are repeated continuously.

3.4.2.3 FLOWCHART FOR RECEIVER:

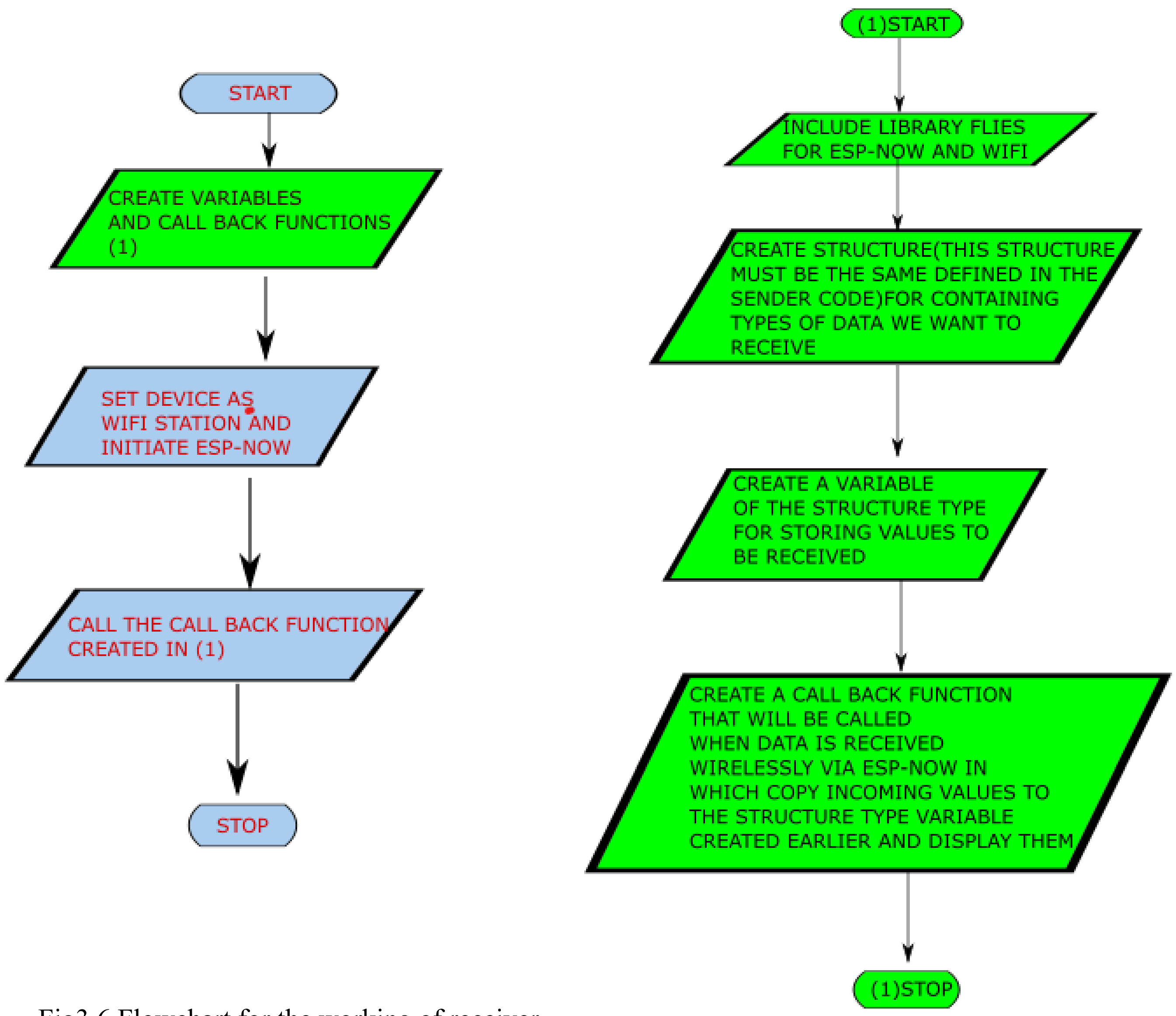

START
CREATE VARIABLES AND CALL BACK FUNCTIONS (1)
SET DEVICE AS WIFI STATION AND INITIATE ESP-NOW
CALL THE CALL BACK FUNCTION CREATED IN (1)
STOP
(1)START
INCLUDE LIBRARY FLIES FOR ESP-NOW AND WIFI
CREATE STRUCTURE(THIS STRUCTURE MUST BE THE SAME DEFINED IN THE SENDER CODE)FOR CONTAINING TYPES OF DATA WE WANT TO RECEIVE
CREATE A VARIABLE OF THE STRUCTURE TYPE FOR STORING VALUES TO BE RECEIVED
CREATE A CALL BACK FUNCTION THAT WILL BE CALLED WHEN DATA IS RECEIVED WIRELESSLY VIA ESP-NOW IN WHICH COPY INCOMING VALUES TO THE STRUCTURE TYPE VARIABLE CREATED EARLIER AND DISPLAY THEM
(1)STOP

Fig3.6 Flowchart for the working of receiver

Fig 3.6 shows the flowchart for working of the receiver. The first process box , which is green in colour is showing the variables and the functions declaration for the receiver. Here, the inbuilt libraries required for the esp- now transfer protocol. A structure has been created for storing the incoming data. This structure must be the same structure, defined at the transmitter end. A call back function has also been created which is called continuously each time when it receives the data to be received. This function also stores the incoming data into the structure created earlier. This function also displays the received values on the display.

In the second process box which is blue, the receiver microcontroller has been set as wifi station and esp-now protocol has been initiated.

In the third blue process box, the call back function created earlier is called.

4 IMPLIMENTATION OF THE SYSTEM

4.1 MASTER CONTROLLER UNIT

4.1.1 Flowchart for Master of Master Controller Unit

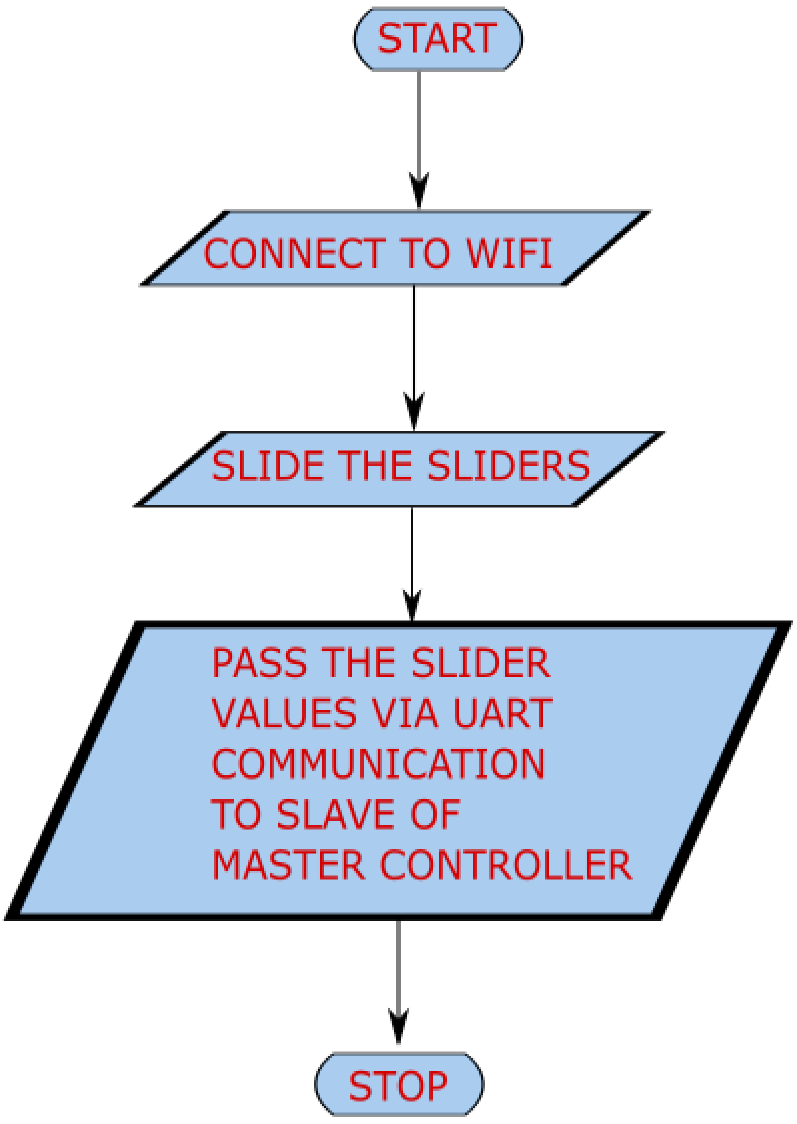

START
CONNECT TO WIFI
SLIDE THE SLIDERS
PASS THE SLIDER VALUES VIA UART COMMUNICATION TO SLAVE OF MASTER CONTROLLER
STOP

Fig4.1 Flowchart for the operation of Master of the Master Controller

Fig 4.1 shows the flowchart for the operation of Master of Master Controller Unit. The first step is to connect to the wifi provided by the microcontroller. The second step once connected to the wifi, an interface consisting of two sliders will appear. In the third step, the Master passed the slider values to the Slave of the Master Controller Unit via UART communication.

4.1.2 Software code for Master of Master Controller:

```
#include <esp_now.h>
#include <esp_wifi.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
// ========================================
#include <HardwareSerial.h>
HardwareSerial SerialPort(2); // use UART2
HardwareSerial SerialPort1(1); // use UART1

// Include the contents of the User Interface Web page, stored in the same folder as the .ino file
#include "PageIndex.h"
#define STATUS_button 5
#define STATUS_led 15
int currentstate;

// Defines the Wi-Fi channel.
// Configured channel range from 1~13 channels (by default).
// "ESP32 Master" and "ESP32 Slaves" must use the same Wi-Fi channel.
#define CHANNEL 1

// Defines the Digital Pin of the "On Board LED".
#define ON_Board_LED 2

// ======================================== Access Point Declaration and
Configuration.
const char* ssid = "ESP32_WS";  //--> access point name
const char* password = "helloesp32WS"; //--> access point password

IPAddress local_ip(192,168,1,1);
IPAddress gateway(192,168,1,1);

IPAddress subnet(255,255,255,0);

// ========================================


// ======================================== REPLACE WITH THE MAC
ADDRESS OF YOUR SLAVES / RECEIVERS / ESP32 RECEIVERS.

uint8_t broadcastAddressESP32Slave1[] = {0x8C, 0xAA, 0xB5, 0x8B, 0x83, 0xCC}; // Slave 1
MAC Address

uint8_t broadcastAddressESP32Slave2[] = {0xC8, 0xF0, 0x9E, 0x9F, 0x3E, 0x3C}; // Slave 2
MAC Address

// ========================================


// ======================================== The variables used to check the
parameters passed in the URL.

// Look in the "PageIndex.h" file.

// "set_LED?board="+board+"&gpio_output="+gpio+"&val="+value

// For example :

// set_LED?board=ESP32Slave1&gpio_output=13&val=1

// PARAM_INPUT_1 = ESP32Slave1

// PARAM_INPUT_2 = 13

// PARAM_INPUT_3 = 1

const char* PARAM_INPUT_1 = "board";

const char* PARAM_INPUT_2 = "gpio_output";

const char* PARAM_INPUT_3 = "val";

// ========================================


// ======================================== Structure example to send data

// Must match the receiver structure

typedef struct struct_message_send {

  int send_GPIO_num;

  int send_Val;

} struct_message_send;
```

```
struct_message_send send_Data; //--> Create a struct_message to send data.
int c = 0;
// ========================================

// Create a variable of type "esp_now_peer_info_t" to store information about the peer.
esp_now_peer_info_t peerInfo;

// Create AsyncWebServer object on port 80
AsyncWebServer server(80);

//
________________________________________________________________________________
__ Callback when data is sent
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  digitalWrite(ON_Board_LED, HIGH); //--> Turn on ON_Board_LED.
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
  Serial.println(">>>>>");
  digitalWrite(ON_Board_LED, LOW); //--> Turn off ON_Board_LED.
}
//
________________________________________________________________________________
__

//
________________________________________________________________________________
__ Subroutine to prepare data and send it to the Slaves.
void send_data_to_slave(int slave_number, int gpio_number, int value) {
  Serial.println();
  Serial.print(">>>>> ");
  Serial.println("Send data");
  Serial.println(slave_number);
  c = slave_number;


  send_Data.send_GPIO_num = gpio_number;
  send_Data.send_Val = value;


  esp_err_t result;


  // :::::::::::::::: Sending data to Slave 1
  if (slave_number == 1) {
    result = esp_now_send(broadcastAddressESP32Slave1, (uint8_t *) &send_Data,
sizeof(send_Data));
  }
  // ::::::::::::::::


  // :::::::::::::::: Sending data to Slave 2
  if (slave_number == 2) {
    result = esp_now_send(broadcastAddressESP32Slave2, (uint8_t *) &send_Data,
sizeof(send_Data));
  }
  // ::::::::::::::::


  if (result == ESP_OK) {
    Serial.println("Sent with success");
  }
  else {
    Serial.println("Error sending the data");
  }
}
//
________________________________________________________________________________
__
```

```
//
________________________________________________________________________________
__ VOID SETUP()
void setup() {
  // put your setup code here, to run once:

  Serial.begin(115200);
  Serial.println();
  SerialPort.begin(15200, SERIAL_8N1, 16, 17);
  SerialPort1.begin(15200, SERIAL_8N1, 18, 19);

  pinMode(ON_Board_LED,OUTPUT); //--> On Board LED port Direction output
  digitalWrite(ON_Board_LED, LOW); //--> Turn off Led On Board
  pinMode(STATUS_button , INPUT_PULLUP);
  // ---------------------------------------- Set Wi-Fi as Access Point and Wifi Station.
  WiFi.mode(WIFI_AP_STA);
  // ----------------------------------------

  // ---------------------------------------- Set the Wi-Fi channel.
  // "ESP32 Master" and "ESP32 Slaves" must use the same Wi-Fi channel.

  int cur_WIFIchannel = WiFi.channel();

  if (cur_WIFIchannel != CHANNEL) {
    //WiFi.printDiag(Serial); // Uncomment to verify channel number before
    esp_wifi_set_promiscuous(true);
    esp_wifi_set_channel(CHANNEL, WIFI_SECOND_CHAN_NONE);
    esp_wifi_set_promiscuous(false);
    //WiFi.printDiag(Serial); // Uncomment to verify channel change after
  }
  // ----------------------------------------
  // ---------------------------------------- Setting up ESP32 to be an Access Point.
  Serial.println();
  Serial.println("-------------");
  Serial.println("Setting up ESP32 to be an Access Point.");
  WiFi.softAP(ssid, password); //--> Creating Access Points
  delay(1000);
  Serial.println("Setting up ESP32 softAPConfig.");
  WiFi.softAPConfig(local_ip, gateway, subnet);
  Serial.println("-------------");
  //----------------------------------------

  // ---------------------------------------- Init ESP-NOW
  Serial.println();
  Serial.println("-------------");
  Serial.println("Start initializing ESP-NOW...");
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    Serial.println("Restart ESP32...");
    Serial.println("-------------");
    delay(1000);
    ESP.restart();
  }
  Serial.println("Initializing ESP-NOW was successful.");
  Serial.println("-------------");
  // ----------------------------------------

  // ---------------------------------------- Once ESPNow is successfully Init, we will register for Send
CB to get the status of Trasnmitted packet
  Serial.println();
  Serial.println("get the status of Trasnmitted packet");
  esp_now_register_send_cb(OnDataSent);
```

```
// ---------------------------------------

// --------------------------------------- Register Peer
Serial.println();
Serial.println("-------------");
Serial.println("Register peer");
peerInfo.encrypt = false;
// :::::::::::::::: register first peer
Serial.println("Register first peer (ESP32 Slave 1)");
memcpy(peerInfo.peer_addr, broadcastAddressESP32Slave1, 6);
if (esp_now_add_peer(&peerInfo) != ESP_OK){
  Serial.println("Failed to add first peer");
  return;
}
// ::::::::::::::::
// :::::::::::::::: register second peer
Serial.println("Register second peer (ESP32 Slave 2)");
memcpy(peerInfo.peer_addr, broadcastAddressESP32Slave2, 6);
if (esp_now_add_peer(&peerInfo) != ESP_OK){
  Serial.println("Failed to add second peer");
  return;
}
// ::::::::::::::::
Serial.println("-------------");
// ---------------------------------------

// --------------------------------------- Handle Web Server
Serial.println();
Serial.println("Setting Up the Main Page on the Server.");
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/html", MAIN_page);
  });
  // ----------------------------------------

  // ---------------------------------------- Send a GET request to
<ESP_IP>/set_LED?board=<inputMessage1>&gpio_output=<inputMessage2>&val=<inputMess
age3>
  server.on("/set_LED", HTTP_GET, [] (AsyncWebServerRequest *request)
  {
    // :::::::::::::::
    // GET input value on
<ESP_IP>/set_LED?board=<inputMessage1>&gpio_output=<inputMessage2>&val=<inputMess
age3>
    // PARAM_INPUT_1 = inputMessage1
    // PARAM_INPUT_2 = inputMessage2
    // PARAM_INPUT_3 = inputMessage3
    // Board = PARAM_INPUT_1
    // LED_gpio_num = PARAM_INPUT_2
    // LED_val = PARAM_INPUT_3
    // :::::::::::::::

    String Board;
    String LED_gpio_num;
    String LED_val;

    if (request->hasParam(PARAM_INPUT_1) && request->hasParam(PARAM_INPUT_2) &&
request->hasParam(PARAM_INPUT_3)) {
      Board = request->getParam(PARAM_INPUT_1)->value();
      LED_gpio_num = request->getParam(PARAM_INPUT_2)->value();
      LED_val = request->getParam(PARAM_INPUT_3)->value();
      //LED_val = digitalRead(STATUS_button);

      String Rslt = "Board : " + Board + " || GPIO : " + LED_gpio_num + " || Set to :" + LED_val;
      Serial.println();
      Serial.println(Rslt);
      // Conditions for sending data to Slave 1.
      if (Board == "ESP32Slave1") send_data_to_slave(1, LED_gpio_num.toInt(),
LED_val.toInt());
      // Conditions for sending data to Slave 2.
      if (Board == "ESP32Slave2") send_data_to_slave(2, LED_gpio_num.toInt(),
LED_val.toInt());
    }
    else {
      Board = "No message sent";
      LED_gpio_num = "No message sent";
      LED_val = "No message sent";
    }
    request->send(200, "text/plain", "OK");
  }
  );
  // ----------------------------------------

  Serial.println();
  Serial.println("Starting the Server.");
  server.begin();

  Serial.println();
  Serial.println("------------");
  Serial.print("SSID name : ");
  Serial.println(ssid);
  Serial.print("IP address : ");
  Serial.println(WiFi.softAPIP());
  Serial.print("Wi-Fi channel : ");
  Serial.println(WiFi.channel());
  Serial.println();
  Serial.println("Connect your computer or mobile Wifi to the SSID above.");
  Serial.println("Visit the IP Address above in your browser to open the main page.");
  Serial.println("------------");
  Serial.println();
}
//
_____________________________________________________________________________
__


//
_____________________________________________________________________________
__ VOID LOOP()
void loop() {
  // put your main code here, to run repeatedly:


//Serial.println(c);
if (c == 1)
{ Serial.print("1:");
  Serial.println(send_Data.send_Val);
  SerialPort.print(send_Data.send_Val);
  }
if (c == 2)
{ Serial.print("2:");
  Serial.println(send_Data.send_Val);
  SerialPort1.print(send_Data.send_Val);}

//Serial.println(send_Data.send_Val);
//SerialPort.print(send_Data.send_Val);
delay(50);
}
```

Page Index Code :

```
const char MAIN_page[] PROGMEM = R"=====(
<!DOCTYPE HTML>
<html>
  <head>
    <title>ESP32 ESP-NOW & WEB SERVER (CONTROLLING)</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>
      html {font-family: Arial; display: inline-block; text-align: center;}
      p {font-size: 1.2rem;}
      body {margin: 0;}
      .topnav {overflow: hidden; background-color: #6C0BA9; color: white; font-size: 1.5rem;}
      .content {padding: 20px; }
      .card {background-color: white; box-shadow: 0px 0px 10px 1px rgba(140,140,140,.5); border: 
1px solid #6C0BA9; border-radius: 15px;}
      .card.header {background-color: #6C0BA9; color: white; border-bottom-right-radius: 0px; 
border-bottom-left-radius: 0px; border-top-right-radius: 12px; border-top-left-radius: 12px;}
      .cards {max-width: 700px; margin: 0 auto; display: grid; grid-gap: 2rem; grid-template-
columns: repeat(auto-fit, minmax(300px, 1fr));}
      .reading {font-size: 2.8rem;}
      .packet {color: #bebebe;}
      .temperatureColor {color: #fd7e14;}
      .humidityColor {color: #1b78e2;}
      .LEDColor {color: #183153;}

      /* ---------------------------------- Toggle Switch */
      .switch {
        position: relative;
        display: inline-block;
        width: 70px;
        height: 34px;
      }
```

```
.switch input {display:none;}

.sliderTS {
  position: absolute;
  cursor: pointer;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: #D3D3D3;
  -webkit-transition: .4s;
  transition: .4s;
  border-radius: 34px;
}

.sliderTS:before {
  position: absolute;
  content: "";
  height: 26px;
  width: 26px;
  left: 4px;
  bottom: 4px;
  background-color: #f7f7f7;
  -webkit-transition: .4s;
  transition: .4s;
  border-radius: 50%;
}

input:checked + .sliderTS {
  background-color: #62c934;
}

input:focus + .sliderTS {
  box-shadow: 0 0 1px #2196F3;
}

input:checked + .sliderTS:before {
  -webkit-transform: translateX(26px);
  -ms-transform: translateX(26px);
  transform: translateX(36px);
}

.sliderTS:after {
  content:'OFF';
  color: white;
  display: block;
  position: absolute;
  transform: translate(-50%,-50%);
  top: 50%;
  left: 70%;
  font-size: 10px;
  font-family: Verdana, sans-serif;
}

input:checked + .sliderTS:after {
  left: 25%;
  content:'ON';
}
/* -------------------------------------------- */

/* -------------------------------------------- Slider */
```

```
.slidecontainer {
  width: 100%;
}

.slider {
  -webkit-appearance: none;
  width: 50%;
  height: 10px;
  border-radius: 5px;
  background: #d3d3d3;
  outline: none;
  opacity: 0.7;
  -webkit-transition: .2s;
  transition: opacity .2s;
}

.slider:hover {
  opacity: 1;
}

.slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 20px;
  height: 20px;
  border-radius: 50%;
  background: #1b78e2;
  cursor: pointer;
}

.slider::-moz-range-thumb {
        width: 20px;
        height: 20px;
        border-radius: 50%;
        background: #1b78e2;
        cursor: pointer;
      }
      /* ----------------------------------- */
    </style>
  </head>

  <body>
    <div class="topnav">
      <h3>ESP32 ESP-NOW & WEB SERVER (CONTROLLING)</h3>
    </div>

    <br>

    <div class="content">
      <div class="cards">

        <div class="card">
          <div class="card header">
            <h2>ESP32 BOARD #1</h2>
          </div>
          <br>

          <h4 class="LEDColor">LED 1 : </h4>
          <label class="switch">
            <input type="checkbox" id="togLED1"
onclick="send_LED_State_Cmd('ESP32Slave1','togLED1','13')">
            <div class="sliderTS"></div>
          </label>
          <br><br>
          <h4 class="LEDColor">LED 2 : </h4>
          <div class="slidecontainer">
            <input type="range" min="0" max="10" value="0" class="slider" id="mySlider1">
          </div>
          <br>
        </div>

        <div class="card">
          <div class="card header">
            <h2>ESP32 BOARD #2</h2>
          </div>
          <br>

          <h4 class="LEDColor">LED 1 : </h4>
          <label class="switch">
            <input type="checkbox" id="togLED2"
onclick="send_LED_State_Cmd('ESP32Slave2','togLED2','13')">
            <div class="sliderTS"></div>
          </label>
          <br><br>
          <h4 class="LEDColor">LED 2 : </h4>
          <div class="slidecontainer">
            <input type="range" min="0" max="10" value="0" class="slider" id="mySlider2">
          </div>
          <br>
        </div>

      </div>
    </div>
```

```
<script>
  //----------------------------------------------------------- The variables for the slider.
  var sliderLED1 = document.getElementById("mySlider1");
  var last_sliderLED1_val = 0;
  var sliderLED2 = document.getElementById("mySlider2");
  var last_sliderLED2_val = 0;
  let sliders_used = "";
  //-----------------------------------------------------------

  //----------------------------------------------------------- The variables for the Timer.
  var myTmr_send;
  var myTmr_send_interval = 250;
  var myTmr_send_start = 1;
  var myTmr_count_to_stop = 0;
  //-----------------------------------------------------------

  //----------------------------------------------------------- The function called by "Toggle Switch" to
control the LED.
  function send_LED_State_Cmd(board,id,gpio) {
    var tgLEDFlash = document.getElementById(id);
    var tgState;

    if (tgLEDFlash.checked == true) tgState = 1;
    if (tgLEDFlash.checked == false) tgState = 0;

    send_cmd(board,gpio,tgState);
  }
  //-----------------------------------------------------------

  //----------------------------------------------------------- Function to detect when the slider is used
and get its value and activate the Timer.
```

```
sliderLED1.oninput = function() {
  sliders_used = "SL1";
  myTmr_count_to_stop = 0;
  if (myTmr_send_start == 1) {
    myTmr_send = setInterval(myTmr_send_LED_PWM_Cmd, myTmr_send_interval);
    myTmr_send_start = 0;
  }
}

sliderLED2.oninput = function() {
  sliders_used = "SL2";
  myTmr_count_to_stop = 0;
  if (myTmr_send_start == 1) {
    myTmr_send = setInterval(myTmr_send_LED_PWM_Cmd, myTmr_send_interval);
    myTmr_send_start = 0;
  }
}
// ------------------------------------------------------------

// ---------------------------------------------------------------------- Timer for sending slider values
and other data from this page to ESP32.
function myTmr_send_LED_PWM_Cmd() {
  if (sliders_used == "SL1") {
    if (last_sliderLED1_val != sliderLED1.value) {
      send_cmd('ESP32Slave1',12,sliderLED1.value);
    }
    last_sliderLED1_val = sliderLED1.value;
  }

  if (sliders_used == "SL2") {
    if (last_sliderLED2_val != sliderLED2.value) {
          send_cmd('ESP32Slave2',12,sliderLED2.value);
         }
         last_sliderLED2_val = sliderLED2.value;
        }

        myTmr_count_to_stop++;
        if (myTmr_count_to_stop > 5) {
          myTmr_send_start = 1;
          clearInterval(myTmr_send);
        }
      }
      // ---------------------------------------------------------------------

      // --------------------------------------------------------------------- XMLHttpRequest to submit data.
      function send_cmd(board,gpio,value) {
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "set_LED?board="+board+"&gpio_output="+gpio+"&val="+value, true);
        xhr.send();
      }
      // ---------------------------------------------------------------------
    </script>
  </body>
</html>
)=====";
```

4.1.3 Code Explanation:

The above code enables the microcontroller to act as Access Point for wifi connection, i.e. the microcontroller acts as a Wi-Fi hotspot to which we can connect our laptop or mobile for accessing the webpage. The Wi-Fi id name and password has been set as well as the IP address has been set, so that we can connect our devices to it. The interface shown below will open which will contain two sliders.

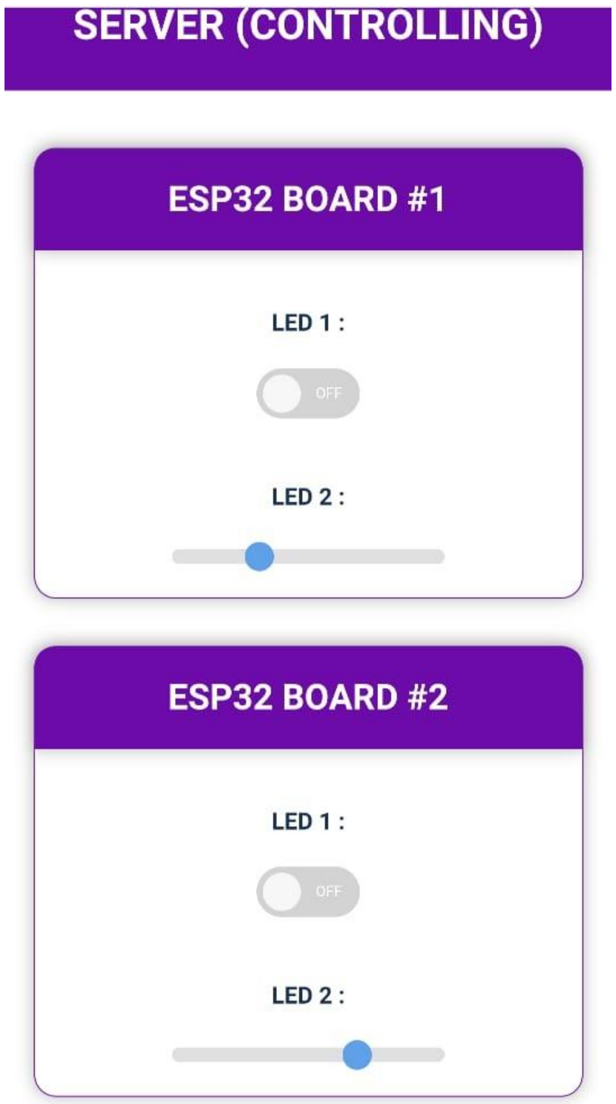


Fig4.2 Figure showing the display interface for the dimming control of the system developed.

This webpage has been written using html code which is in a separate file called PageIndex. This file is called in the parent function. In the parent function, the slider values are detected and is passed through UART communication protocol to the Slave. Two different UART communication channels are used for sending the values of two different sliders.

4.1.4 Flowchart for Slave of Master Controller Unit

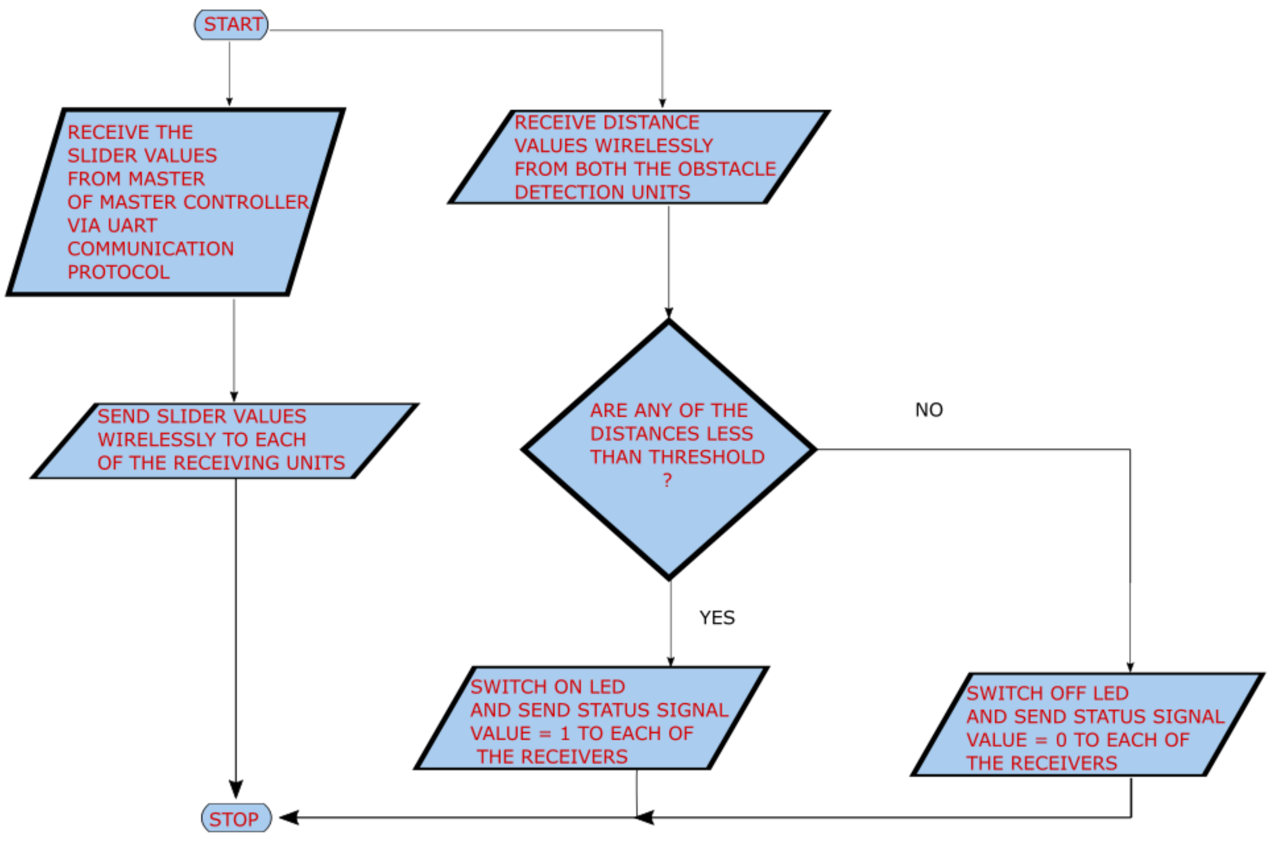

START
RECEIVE THE SLIDER VALUES FROM MASTER OF MASTER CONTROLLER VIA UART COMMUNICATION PROTOCOL
RECEIVE DISTANCE VALUES WIRELESSLY FROM BOTH THE OBSTACLE DETECTION UNITS
SEND SLIDER VALUES WIRELESSLY TO EACH OF THE RECEIVING UNITS
ARE ANY OF THE DISTANCES LESS THAN THRESHOLD ?
NO
YES
SWITCH ON LED AND SEND STATUS SIGNAL VALUE = 1 TO EACH OF THE RECEIVERS
SWITCH OFF LED AND SEND STATUS SIGNAL VALUE = 0 TO EACH OF THE RECEIVERS
STOP

Fig 4.3 Flowchart for the operation of Slave of Master Controller

Fig 4.3 shows the flowchart working of the Slave of the Master Controller Unit. The first process box shoes that the slider values which were passed from Master to Slave via UART communication protocol are being received by the Slave and are being sent wirelessly to each receiver. At the same time, the Slave is also receiving the distances from the obstacle detection units wirelessly and based upon the distance data received, decision is taken weather to switch on or off the status led. The status signal is then sent wirelessly to each receiver.If the LED is switched ON , send status signal as 1 and vice versa.

4.1.5 Software code for Slave of Master Controller Unit

```
#include <HardwareSerial.h>
#define STATUS_button 5
#define STATUS_led 15
//int currentstate;

#include <esp_now.h>
#include <WiFi.h>
uint8_t broadcastAddress1[] = {0x8C, 0xAA, 0xB5, 0x8B, 0x83, 0xCC};
uint8_t broadcastAddress2[] = {0x30, 0xC6, 0xF7, 0x00, 0x39, 0xD0};

typedef struct test_struct {
  int id;
  int x ;
  int z ;
  int c ;
  int p ;
  int y;
} test_struct;
test_struct incomingtest; //this structure is used to receive readings

typedef struct test_struct1 {
  int a;
  int b;
  int c;
} test_struct1;
test_struct1 test; //this structure is used to send readings



test_struct board1; //these structures are created to hold the incoming readings
test_struct board2;
test_struct boardsStruct[2] = {board1, board2};

esp_now_peer_info_t peerInfo;

// callback when data is sent
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  char macStr[18];
  //Serial.print("Packet to: ");
  // Copies the sender mac address to a string
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
           mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4], mac_addr[5]);
  //Serial.print(macStr);
 // Serial.print(" send status:\t");
  //Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
}

// callback function that will be executed when data is received
void OnDataRecv(const uint8_t * mac_addr, const uint8_t *incomingData, int len) {
  char macStr[18];
  //Serial.print("Packet received from: ");
  snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",
           mac_addr[0], mac_addr[1], mac_addr[2], mac_addr[3], mac_addr[4], mac_addr[5]);
  //Serial.println(macStr);
  memcpy(&incomingtest, incomingData, sizeof(incomingtest));
  //Serial.printf("Board ID %u: %u bytes\n", test.id, len);
  // Update the structures with the new incoming data
  boardsStruct[incomingtest.id-1].p = incomingtest.p;
  boardsStruct[incomingtest.id-1].x = incomingtest.x;
  boardsStruct[incomingtest.id-1].y = incomingtest.y;
  boardsStruct[incomingtest.id-1].z = incomingtest.z;
  boardsStruct[incomingtest.id-1].c = incomingtest.c;
  //boardsStruct[incomingtest.id-1].y = incomingtest.y;
  //Serial.printf("p value: %d \n", boardsStruct[incomingtest.id-1].p);
  //Serial.printf("x value: %d \n", boardsStruct[incomingtest.id-1].x);
  //Serial.println();
}

HardwareSerial SerialPort(2); // use UART2
HardwareSerial SerialPort1(1); // use UART1
char number  = ' ';
char number1  = ' ';
int x1;
int z1;
int c1;
void setup()
{
  Serial.begin(115200);
  SerialPort.begin(15200, SERIAL_8N1, 16, 17);
  SerialPort1.begin(15200, SERIAL_8N1, 18,19);
  //pinMode(STATUS_button , INPUT_PULLUP);
  pinMode(STATUS_led , OUTPUT);
   WiFi.mode(WIFI_STA);

  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Trasnmitted packet
  esp_now_register_send_cb(OnDataSent);
  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_register_recv_cb(OnDataRecv);

  // register peer
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
  // register first peer
  memcpy(peerInfo.peer_addr, broadcastAddress1, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    //Serial.println("Failed to add peer");
    return;
  }
  // register second peer
  memcpy(peerInfo.peer_addr, broadcastAddress2, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
   // Serial.println("Failed to add peer");
    return;
  }
}
void loop()
{
  int board1X = boardsStruct[0].x;
  int board1Z = boardsStruct[0].z;
  int board1C = boardsStruct[0].c;  //These are the incoming readings
  int board1Y = boardsStruct[0].y;
  int board1P = boardsStruct[0].p; // Distance coming from blue board
  int board2P = boardsStruct[1].p; // Distance coming from black board

  Serial.print("Distance coming from first:");Serial.print(board1P);
  Serial.print(" Distance coming from second:");Serial.println( board2P);
```

```
 if (SerialPort.available())
 {
   char number = SerialPort.read();
   x1 = number-'0';
   c1 = 1;
   Serial.print(" Data coming from :");Serial.print( c1);
   Serial.print(" Num1= ");Serial.println(x1);
  }

if (SerialPort1.available())
 {
   char number1 = SerialPort1.read();
   z1 = number1-'0'; c1 = 2;
   Serial.print(" Data coming from :");Serial.print( c1);
   Serial.print(" Num2= ");Serial.println(z1);
 }



 if(board1P<30 || board2P<30)
 {   digitalWrite(STATUS_led , HIGH);
         test.a = 0;
         test.b = 0;
         test.c = 1;
   }

 else
 {
    digitalWrite(STATUS_led , LOW);
        test.a =x1;
        test.b =z1;
        test.c = 0;
      }



    esp_err_t result = esp_now_send(0, (uint8_t *) &test, sizeof(test_struct));

  if (result == ESP_OK) {
    Serial.println("Sent with success");
  }
  else {
    Serial.println("Error sending the data");
  }

  delay(10);
}
```

4.1.6 Code Explanation:

In the above code, the Slave is receiving the distance data from the two Distance Observation Units and processing them. It switches ON an indicator led if any or both the distances are below the threshold value of distance, set by the programmer.

The Slave is receiving the values of the sliders from the Master via UART communication protocol through two channels, each for different slider. The Slave is then sending the slider data to each of the two Receiving Units wirelessly.

4.1.7 Schematics for Master Controller Unit

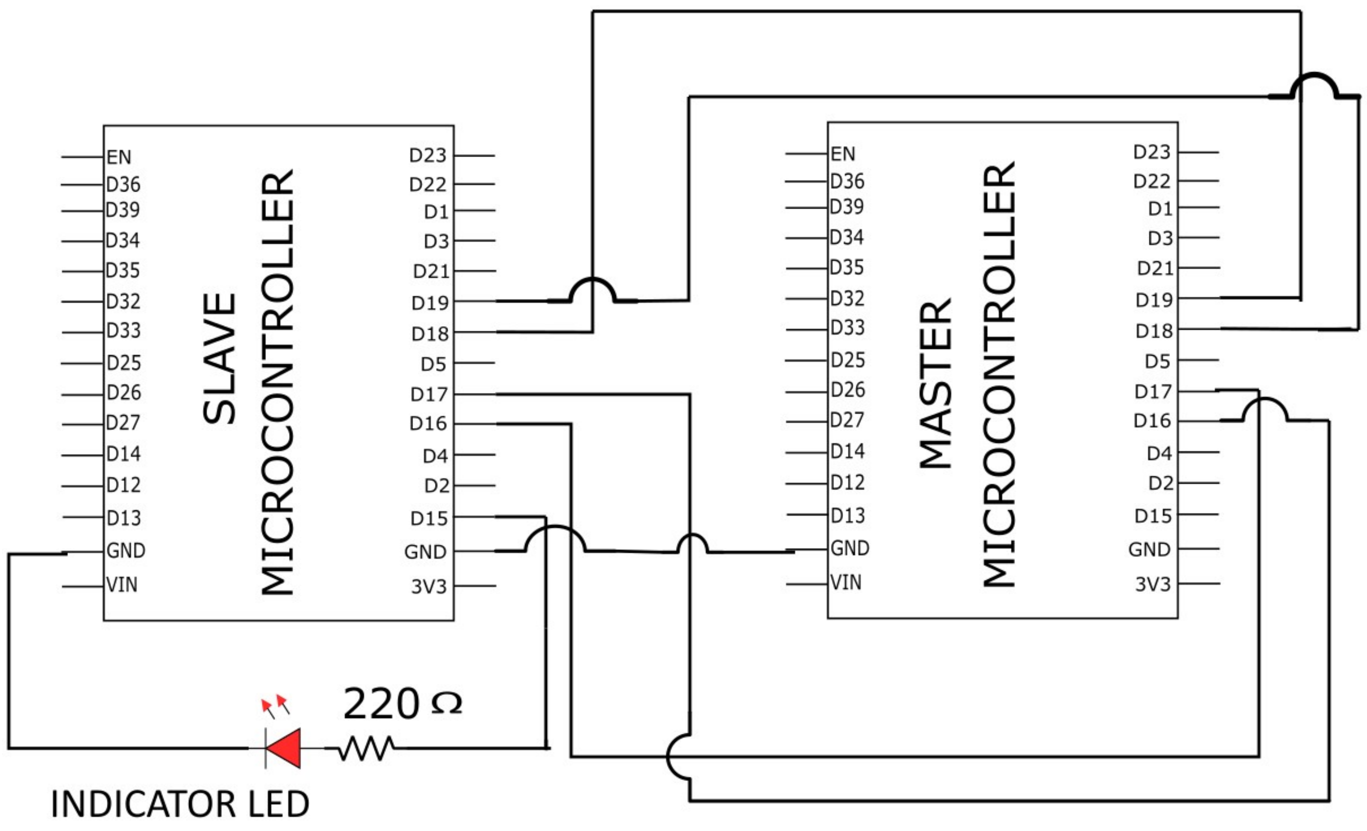


Fig4.4 Figure showing schematics of the Master Controller Unit

The above diagram shows the schematics of the Master Controller Unit This unit is the heart of the entire experiment. The unit has two microcontrollers, one Master another is the Slave, both connected together in such a way that they communicate with each other via serial communication protocol.

The Master microcontroller is the one which provides the wifi connection so that we may be able to access the webpage. The webpage has two sliders by which the brightness can be controlled by pwm dimming.

The Master controller takes the values corresponding to the sliders and feeds them to the Slave controller .

The Slave is already receiving distance data from the two Distance Observation Units wirelessly.

The Slave transmits the slider data wirelessly to the Receiver units which are connected to the driver circuits of the luminaires. The slave receives the distance data from the Distance Observation Units and compares them so that they are less than a threshold distance set by the programmer.

4.2 RECEIVER UNIT

4.2.1 Flowchart for receiver1

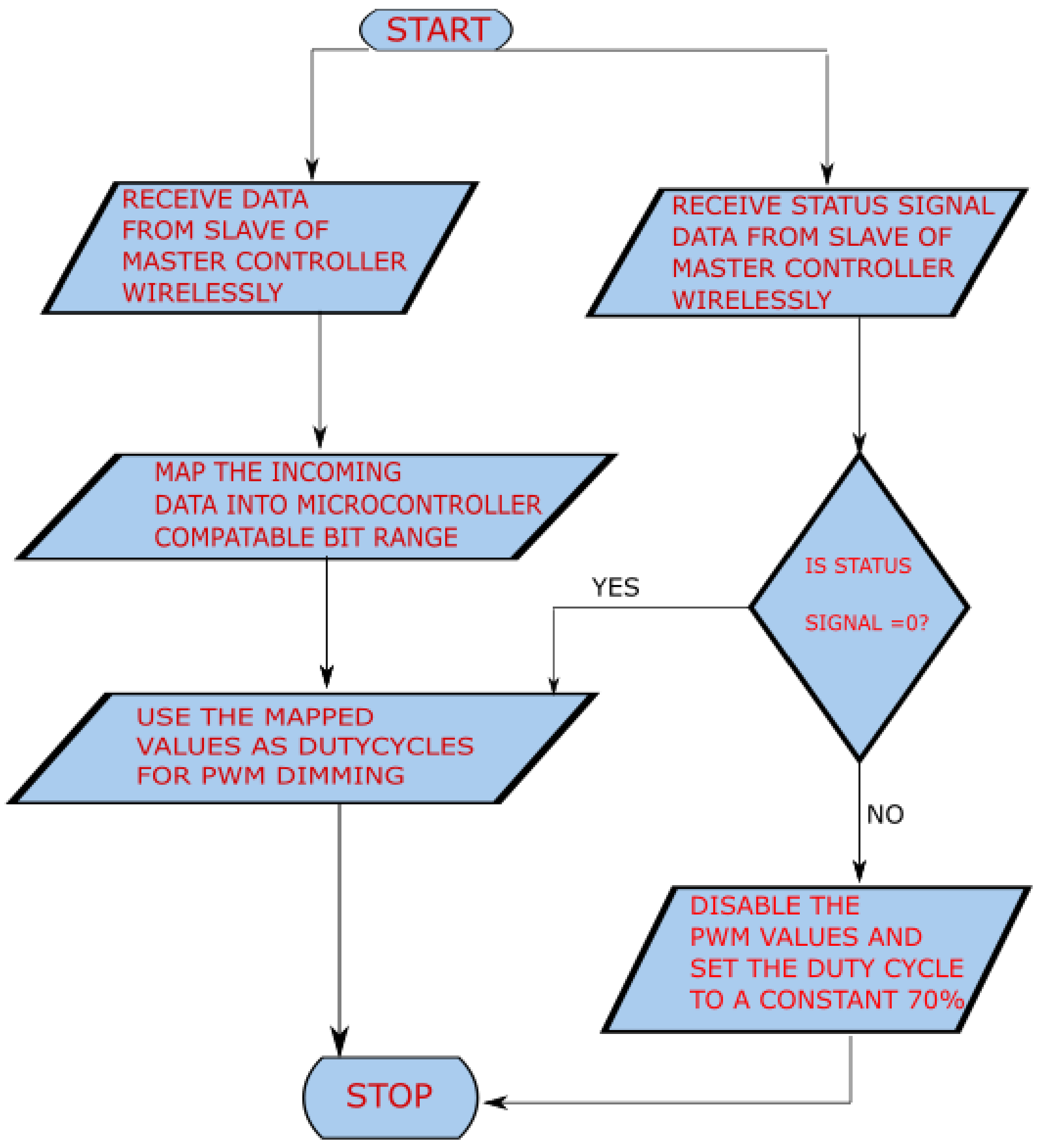

START
RECEIVE DATA FROM SLAVE OF MASTER CONTROLLER WIRELESSLY
RECEIVE STATUS SIGNAL DATA FROM SLAVE OF MASTER CONTROLLER WIRELESSLY
MAP THE INCOMING DATA INTO MICROCONTROLLER COMPATABLE BIT RANGE
IS STATUS SIGNAL =0?
YES
USE THE MAPPED VALUES AS DUTYCYCLES FOR PWM DIMMING
NO
DISABLE THE PWM VALUES AND SET THE DUTY CYCLE TO A CONSTANT 70%
STOP

Fig 4.5 Flowchart showing the operation of the receiver unit 1

Fig 4.5 shows the flowchart explaining the operation of receiver. The receiver has been wirelessly receiving the pwm data from the slave of Master Controller wirelessly and has been mapping these values into microcontroller compatible bit range. These mapped values are being used as duty cycles for pwm dimming. At the same time, The receiving unit has also been receiving the status signal wirelessly from the slave and based upon that decision is taken whether to enable or disable the pwm dimming feature.

4.2.2 Software code for receiver unit1

```
#include <esp_now.h>
#include <WiFi.h>
// setting the Gpio pin nos for pwm dimming
const int ledPin1 = 32;
const int ledPin2 = 25;
// setting PWM properties
const int freq = 1000;
const int ledChannel1 = 0;
const int ledChannel2 = 1;
const int resolution = 8;

//Structure example to receive data
//Must match the sender structure
typedef struct test_struct1 {
  int a;
  int b;
  int c;
} test_struct1;

//Create a struct_message called myData
test_struct1 myData;

//callback function that will be executed when data is received
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.print("Bytes received: ");
  Serial.println(len);
  Serial.print("a: ");
  Serial.println(myData.a);
  Serial.print("b: ");
  Serial.println(myData.b);
  Serial.print("c: ");
  Serial.println(myData.c);


  Serial.println();
}

void setup() {
  //Initialize Serial Monitor
  Serial.begin(115200);
  // configure LED PWM functionalitites
ledcSetup(ledChannel1, freq, resolution);
ledcSetup(ledChannel2, freq, resolution);

// attach the channel to the GPIO to be controlled
ledcAttachPin(ledPin1, ledChannel1);
ledcAttachPin(ledPin2, ledChannel2);

  //Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  //Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_register_recv_cb(OnDataRecv);
```

```
}

void loop() {
int pwm1 = map(myData.a, 0, 10, 0, 255);
int pwm2 = map(myData.b, 0, 10, 0, 255);
if (myData.c==0){
ledcWrite(ledChannel1, pwm1);
ledcWrite(ledChannel2, pwm2);
}
else{
  ledcWrite(ledChannel1, 178);
  ledcWrite(ledChannel2, 178);
  }
}
```

4.2.3 Code Explanation:

The microcontroller wirelessly is receiving the slider values from the two different sliders and mapping them to produce duty cycle percentage . The duty cycle percentages are used to control the pwm dimming on two pins.

4.2.4 Schematics for receiver unit 1

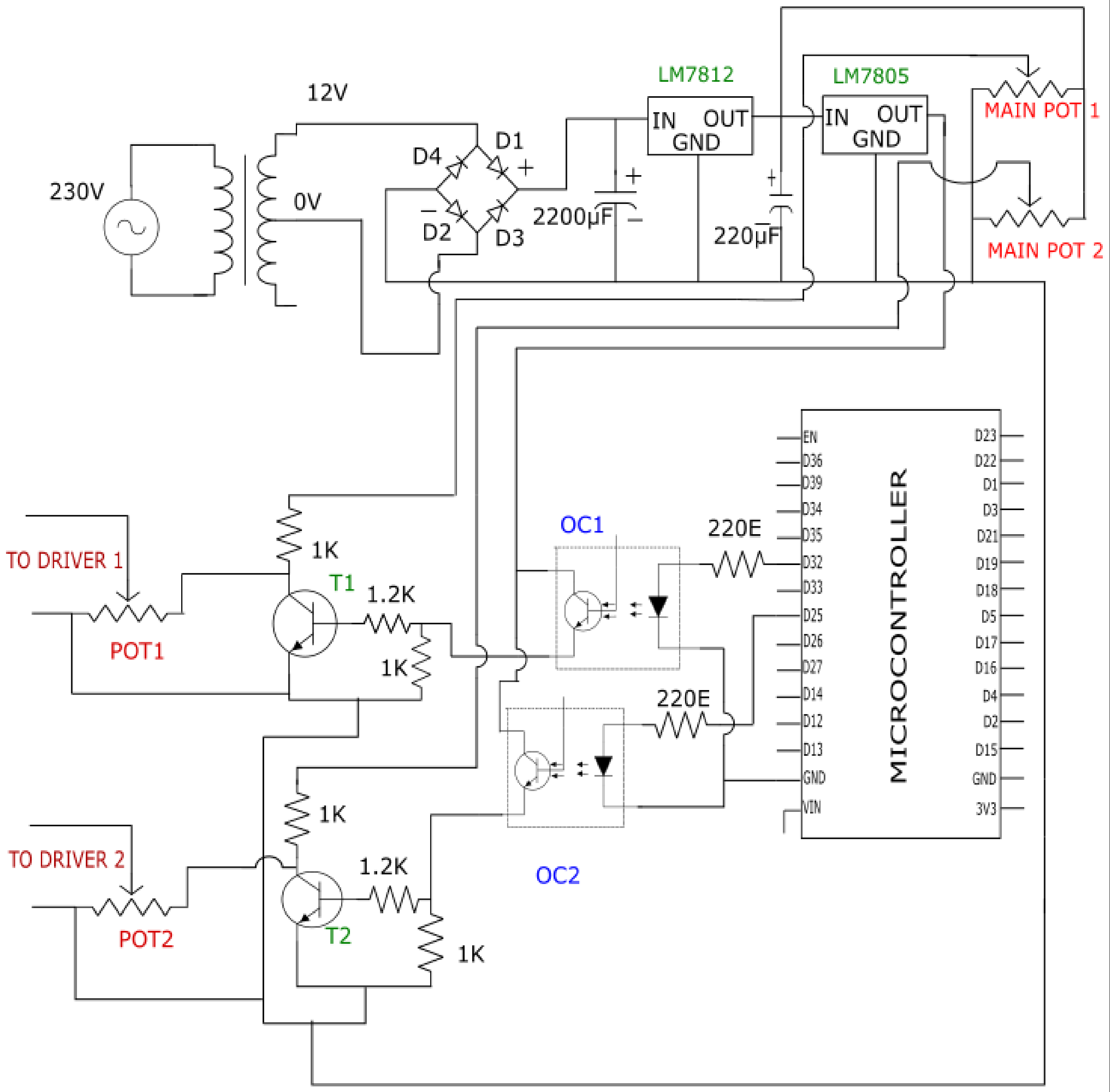


Fig4.6 Schematics of the receiver unit

The above diagram shows the schematics of the receiver unit. The microcontroller is receiving the pwm values wirelessly from the Master Controller Unit. The pwm values are being amplified and fed to the driver.

The amplifications are done by two transistors T1 and T2 , which are biased in the amplification or active region . The phase reversal will be taken care off by the output transistor of the opto-coupler used . The opto-couplers OC1 and OC2 are used to electrically isolate the microcontroller unit from the power circuit and are biased by LM7805.The LM7812 provides biasing for the transistors for their amplification. The potentiometers POT1 and POT2 are used for stepping down the amplitude of the amplified signal output from the transistors so as to meet the input voltage requirement for the drivers. The resulting amplified signals are applied to drivers, DRIVER1 and DRIVER2.

The two main pots MAIN POT 1 and MAIN POT 2 are used for compensating the extra voltage drawn by the two drivers so that there is no threshold.

Usage of Transistors in the experiment

The pwm values generated by the Microcontroller are being amplified and fed to the driver. The Microcontroller generated 3.3V amplitude waveform but the driver circuits are 10V compatable. So an amplification of the signals is required in order to drive the drivers.

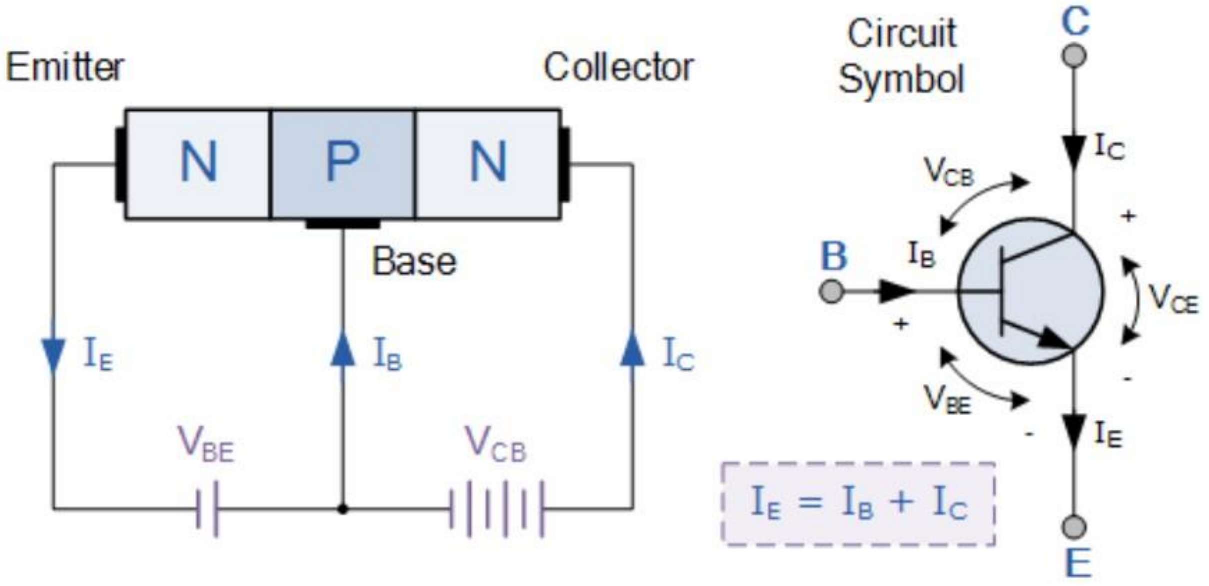


Fig4.7 Working of a Bipolar Junction Transistor

This amplification process is done by the two npn bjt transistors which are working biased in the amplification region. The output taken from the transistors are fed to respective variable resistors so as to step down the voltage levels further so as to achieve the required voltage levels for driving the led driver circuits.

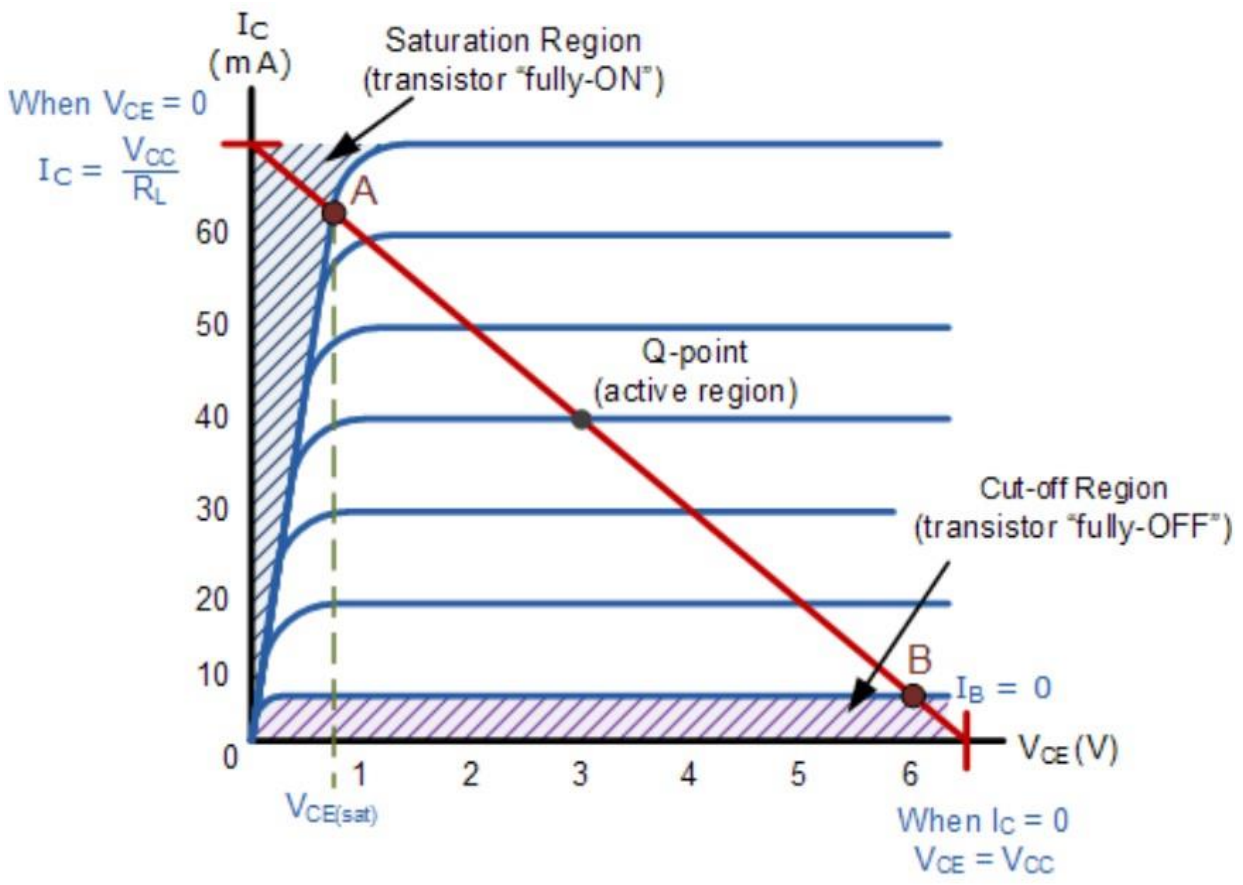


Fig4.8 Graph showing the regions of operation of a Bipolar Junction Transistor

The figures above are showing the n-p-n configuration of bjt transistor along with their characteristics. In the characteristics of the bjt, we see there are three regions.

1. Cut-off Region

Here the operating conditions of the transistor are zero input base current (I_B), zero output collector current (I_C) and maximum collector voltage (V_{CE}) which results in a large depletion layer and no current flowing through the device. Therefore the transistor is switched “Fully-OFF”.

2. Saturation Region

Here the transistor will be biased so that the maximum amount of base current is applied, resulting in maximum collector current resulting in the minimum collector emitter voltage drop which results in the depletion layer being as small as possible and maximum current flowing through the transistor. Therefore the transistor is switched “Fully-ON”.

3. Active Region

Here the transistor will be biased so it acts as an amplifier. The voltage signal to be amplified is applied on the base terminal and the amplified signal is available on the collector terminal, with the emitter terminal grounded. The output will have a phase reversal .We are biasing our transistors in this region to amplify the voltage signal, generated by the microcontroller.

4.2.5 Flowchart for Receiver Unit 2

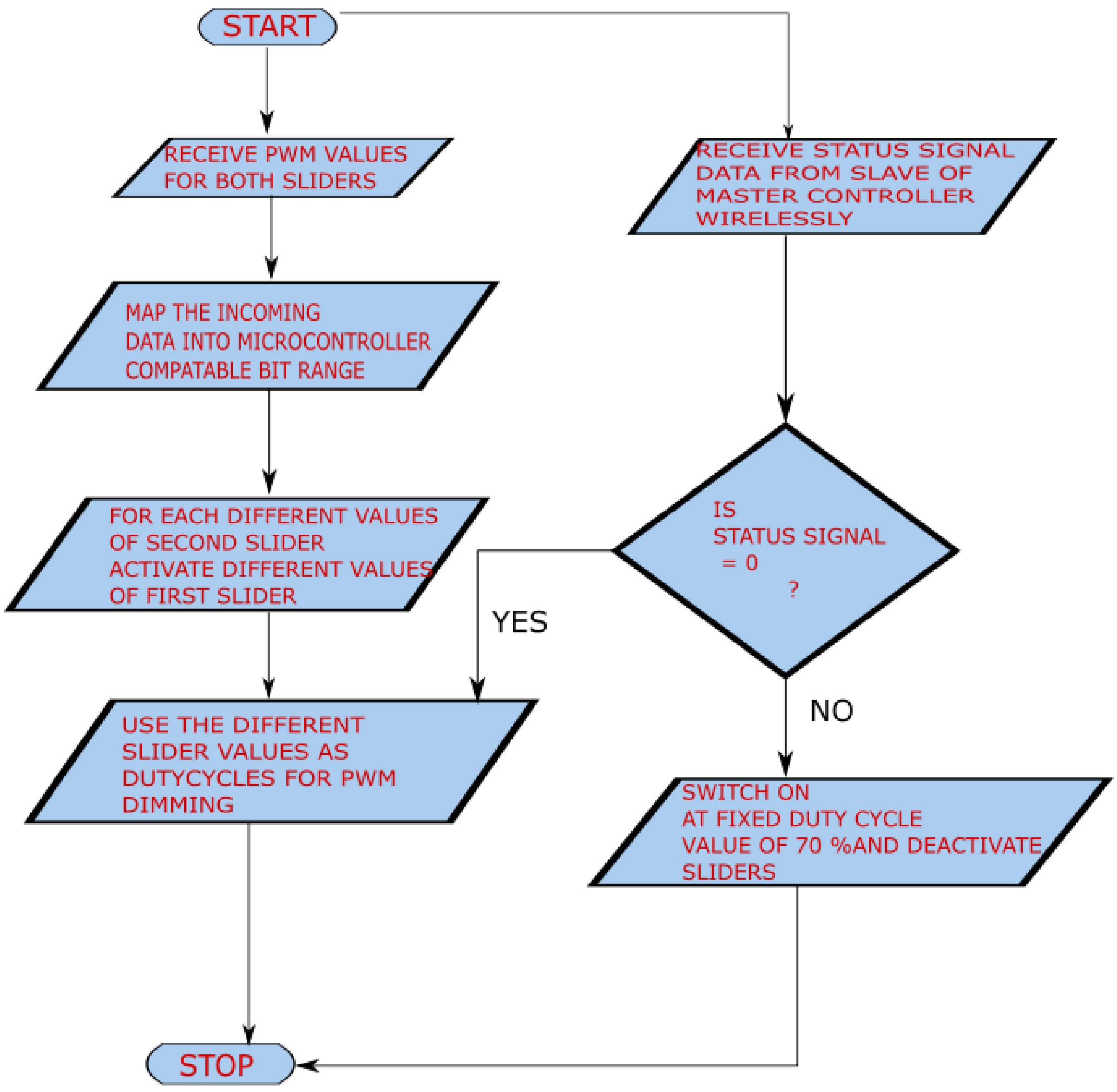

START
RECEIVE PWM VALUES FOR BOTH SLIDERS
MAP THE INCOMING DATA INTO MICROCONTROLLER COMPATABLE BIT RANGE
FOR EACH DIFFERENT VALUES OF SECOND SLIDER ACTIVATE DIFFERENT VALUES OF FIRST SLIDER
USE THE DIFFERENT SLIDER VALUES AS DUTYCYCLES FOR PWM DIMMING
RECEIVE STATUS SIGNAL DATA FROM SLAVE OF MASTER CONTROLLER WIRELESSLY
IS STATUS SIGNAL = 0 ?
YES
NO
SWITCH ON AT FIXED DUTY CYCLE VALUE OF 70 %AND DEACTIVATE SLIDERS
STOP

Fig 4.9 Flowchart showing the operation of the receiving unit.

Fig 4.9 shows the flowchart for the operation of the receiver unit 2. The receiver unit is receiving the pwm slider values wirelessly and are mapped into the microcontroller compatible bit range. For each different values of the second slider different values of the first slider are activated which are set by the programmer. Status signal has also been received wirelessly at the same time .If the status signal is 0, pwm duty cycle cycle variation is activated .If it is 1, duty cycle has been fixed at 70 %.

4.2.6 Software code for Receiver Unit 2

```
#include <esp_now.h>
#include <WiFi.h>
// setting the Gpio pin nos for pwm dimming
const int ledPin1 = 32;
const int ledPin2 = 25;
// setting PWM properties
const int freq = 1000;
const int ledChannel1 = 0;
const int ledChannel2 = 1;
const int resolution = 8;

//Structure example to receive data
//Must match the sender structure
typedef struct test_struct1 {
  int a;
  int b;
  int c;
} test_struct1;

//Create a struct_message called myData
test_struct1 myData;

//callback function that will be executed when data is received
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.print("Bytes received: ");
  Serial.println(len);
  Serial.print("a: ");
  Serial.println(myData.a);
  Serial.print("b: ");
  Serial.println(myData.b);
  Serial.print("c: ");
  Serial.println(myData.c);

  Serial.println();
}

void setup() {
  //Initialize Serial Monitor
  Serial.begin(115200);
  // configure LED PWM functionalitites
ledcSetup(ledChannel1, freq, resolution);
ledcSetup(ledChannel2, freq, resolution);

// attach the channel to the GPIO to be controlled
ledcAttachPin(ledPin1, ledChannel1);
ledcAttachPin(ledPin2, ledChannel2);

  //Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  //Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for recv CB to
  // get recv packer info
  esp_now_register_recv_cb(OnDataRecv);
}
```

```
void loop() {

int pwm1 = map(myData.a, 0, 10, 0, 255);
int pwm2 = map(myData.b, 0, 10, 0, 255);
if (myData.c==0){
/*ledcWrite(ledChannel1, 255-pwm1);    //varying values of duty cycles
ledcWrite(ledChannel2, 255-pwm2);*/
//first matrix
if(myData.b ==1){

  if(myData.a==0)
  {ledcWrite(ledChannel1, 255);
   ledcWrite(ledChannel2, 255);
  }
  if(myData.a==1)
  {ledcWrite(ledChannel1, 26);
   ledcWrite(ledChannel2, 204);
  }
  if(myData.a==2)
  {ledcWrite(ledChannel1, 38);
   ledcWrite(ledChannel2, 190);
  }
  if(myData.a==3)
  {ledcWrite(ledChannel1, 50);
   ledcWrite(ledChannel2, 153);
  }
  if(myData.a==4)
  {ledcWrite(ledChannel1, 90);
   ledcWrite(ledChannel2, 140);
  }
  if(myData.a==5)
  {ledcWrite(ledChannel1, 100);
   ledcWrite(ledChannel2, 90);
  }
  if(myData.a==6)


  {ledcWrite(ledChannel1, 140);
   ledcWrite(ledChannel2, 76);
  }
  if(myData.a==7)
  {ledcWrite(ledChannel1, 153);
   ledcWrite(ledChannel2, 64);
  }
  if(myData.a==8)
  {ledcWrite(ledChannel1, 190);
   ledcWrite(ledChannel2, 50);
   }
  if(myData.a==9)
  {ledcWrite(ledChannel1, 204);
   ledcWrite(ledChannel2, 40);
  }


}


//second matrix
if(myData.b==2){


  if(myData.a==0)
  {ledcWrite(ledChannel1, 255);
   ledcWrite(ledChannel2, 255);
  }
```

```
if(myData.a==1)
{ledcWrite(ledChannel1, 200);
 ledcWrite(ledChannel2, 20);
}
if(myData.a==2)
{ledcWrite(ledChannel1, 150);
 ledcWrite(ledChannel2, 190);
}
if(myData.a==3)
{ledcWrite(ledChannel1, 60);
 ledcWrite(ledChannel2, 156);
}
if(myData.a==4)
{ledcWrite(ledChannel1, 80);
 ledcWrite(ledChannel2, 140);
}
if(myData.a==5)
{ledcWrite(ledChannel1, 85);
 ledcWrite(ledChannel2, 75);
}
if(myData.a==6)

{ledcWrite(ledChannel1, 145);
 ledcWrite(ledChannel2, 80);
}
if(myData.a==7)
{ledcWrite(ledChannel1, 150);
 ledcWrite(ledChannel2, 69);
}
if(myData.a==8)
{ledcWrite(ledChannel1, 190);
```

```
   ledcWrite(ledChannel2, 79);
   }
  if(myData.a==9)
  {ledcWrite(ledChannel1, 200);
   ledcWrite(ledChannel2, 15);
  }


}
//third matrix


if(myData.b==3)
{
  if(myData.a==0)
  {ledcWrite(ledChannel1, 255);
   ledcWrite(ledChannel2, 255);
  }
  if(myData.a==1)
  {ledcWrite(ledChannel1, 55);
   ledcWrite(ledChannel2, 190);
  }
  if(myData.a==2)
  {ledcWrite(ledChannel1, 85);
   ledcWrite(ledChannel2, 190);
  }
  if(myData.a==3)
  {ledcWrite(ledChannel1, 50);
   ledcWrite(ledChannel2, 75);
  }
  if(myData.a==4)
  {ledcWrite(ledChannel1, 90);
   ledcWrite(ledChannel2, 140);
  }
  if(myData.a==5)
  {ledcWrite(ledChannel1, 100);
   ledcWrite(ledChannel2, 15);
  }
  if(myData.a==6)

  {ledcWrite(ledChannel1, 140);
   ledcWrite(ledChannel2, 55);
  }
  if(myData.a==7)
  {ledcWrite(ledChannel1, 153);
   ledcWrite(ledChannel2, 64);
  }
  if(myData.a==8)
  {ledcWrite(ledChannel1, 20);
   ledcWrite(ledChannel2, 155);
   }
  if(myData.a==9)
  {ledcWrite(ledChannel1, 40);
   ledcWrite(ledChannel2, 240);
  }

}

}
else{
  ledcWrite(ledChannel1, 178);      //fixed value of duty cycle
  ledcWrite(ledChannel2, 178);
  }
}
```

4.2.7 Code Explanation:

The Microcontroller receives the two slider values wirelessly and maps them to dutycycle. These dutycycles are used for pwm dimming on two pins of the microcontroller. One pin is used for cool white led array another is for warm white led array. The microcontroller controls the dimming via voltage amplification circuit.

In the above code the each of the second slider pwm values actually sets different dimming ranges for the first slider. Without the activation of the first slider, the dimming level will not change.

Also if the distance coming from any of the Distance Observation Units is less than the threshold, the dutycycle is set to a fixed percentage.

4.2.8 Schematics for Receiver Unit 2

The same schematics for Receiver Unit 1 is used

4.2.9 Set up of the system

The figure shown in Fig 4.10 the set up for the system. The receiver unit is attached with the street luminaire , the master controller unit is placed at a distance and the distance observation unit is placed at a further distance .Wireless control of the luminaire is being done.

4.2.10 Controller Circuit

The figure 4.11 shows the controller circuit used for amplification of the pwm signal from the microcontroller. The power transformer has been used for stepping down the 230 v to 14 v.

The diode bridge rectifier rectifies the ac signal to pulsating dc signal. The diode bridge rectifier is formed by connecting four signal diodes in bridge configuration. The filter capacitors are used give a ripple free output .The voltage regulators provide for constant output voltages. These voltages are for dc biasing the transistors for amplification purpose. Main potentiometers are used for compensation of load voltage requirements. The optocouplers provide for isolation between the microcontroller signals and the power side of the control circuit , thereby ensuring the safety of the microcontroller. The transistors perform the amplification of the signals from the microcontroller. Again at the output of the transistors, there are potentiometers for the final adjustment of the amplitude of the output signal.

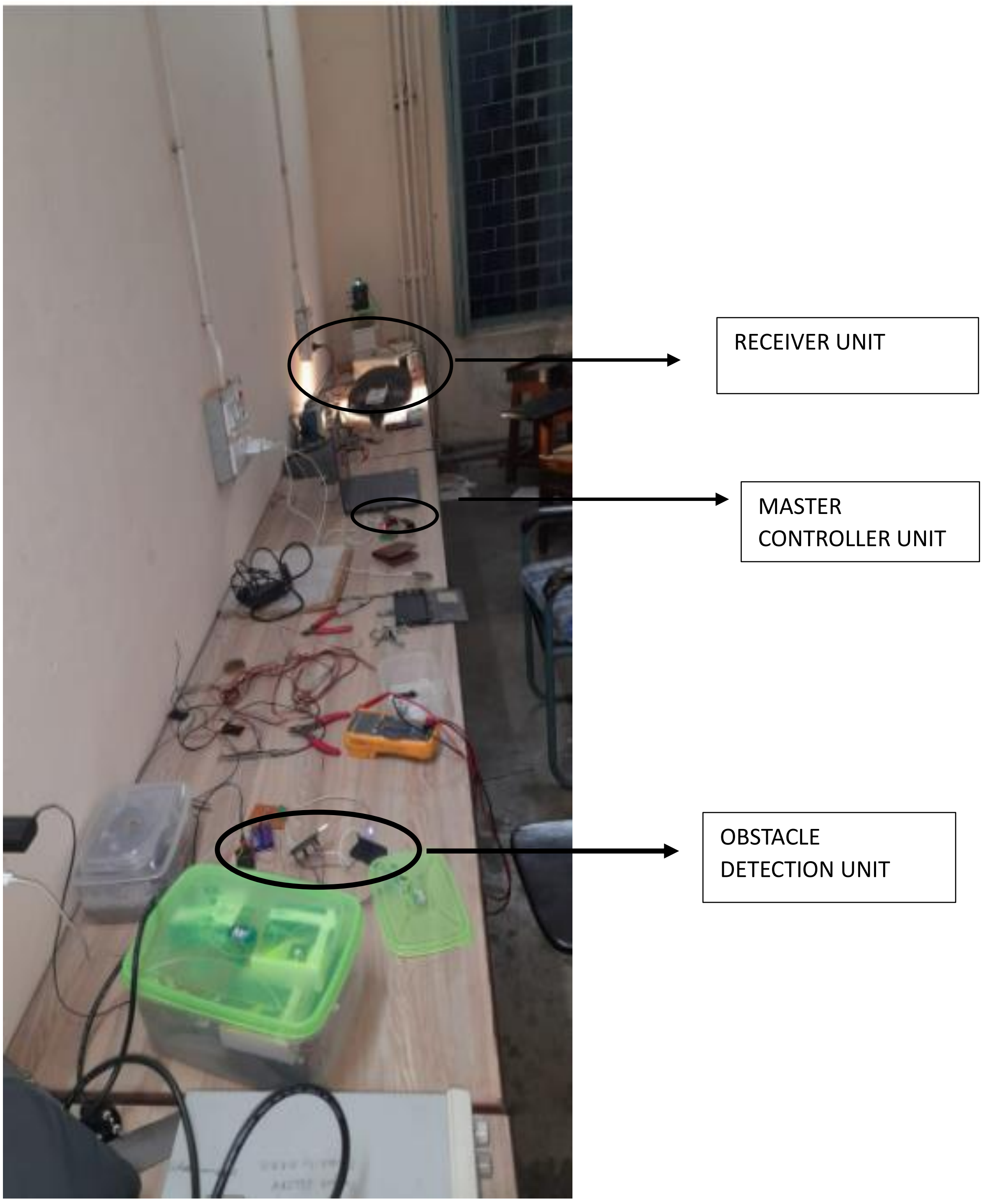


Fig 4.10 Set up for the experiment

4.2.10 Controller Circuit

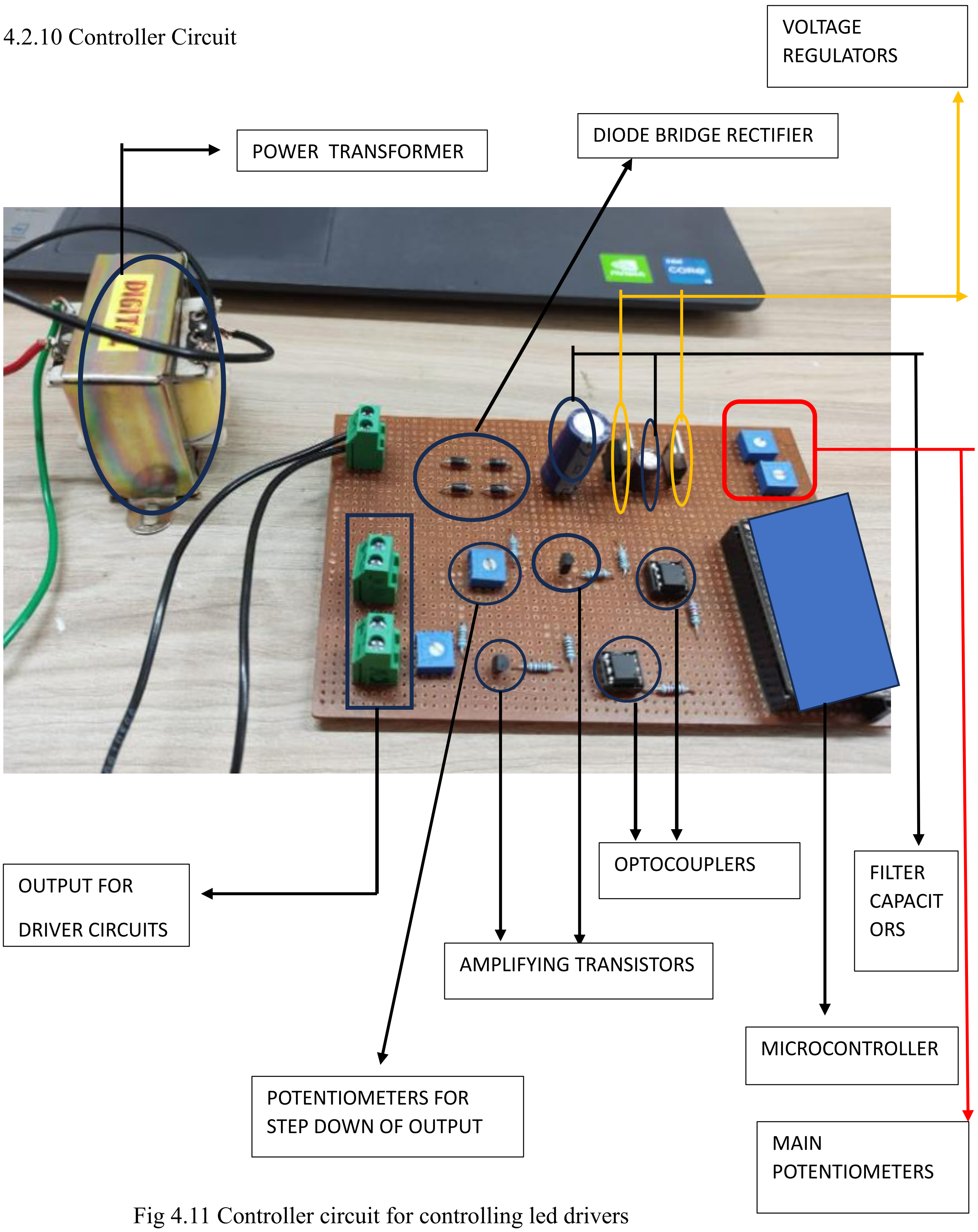


Fig 4.11 Controller circuit for controlling led drivers

4.2.11 Setup for the Receiver Unit

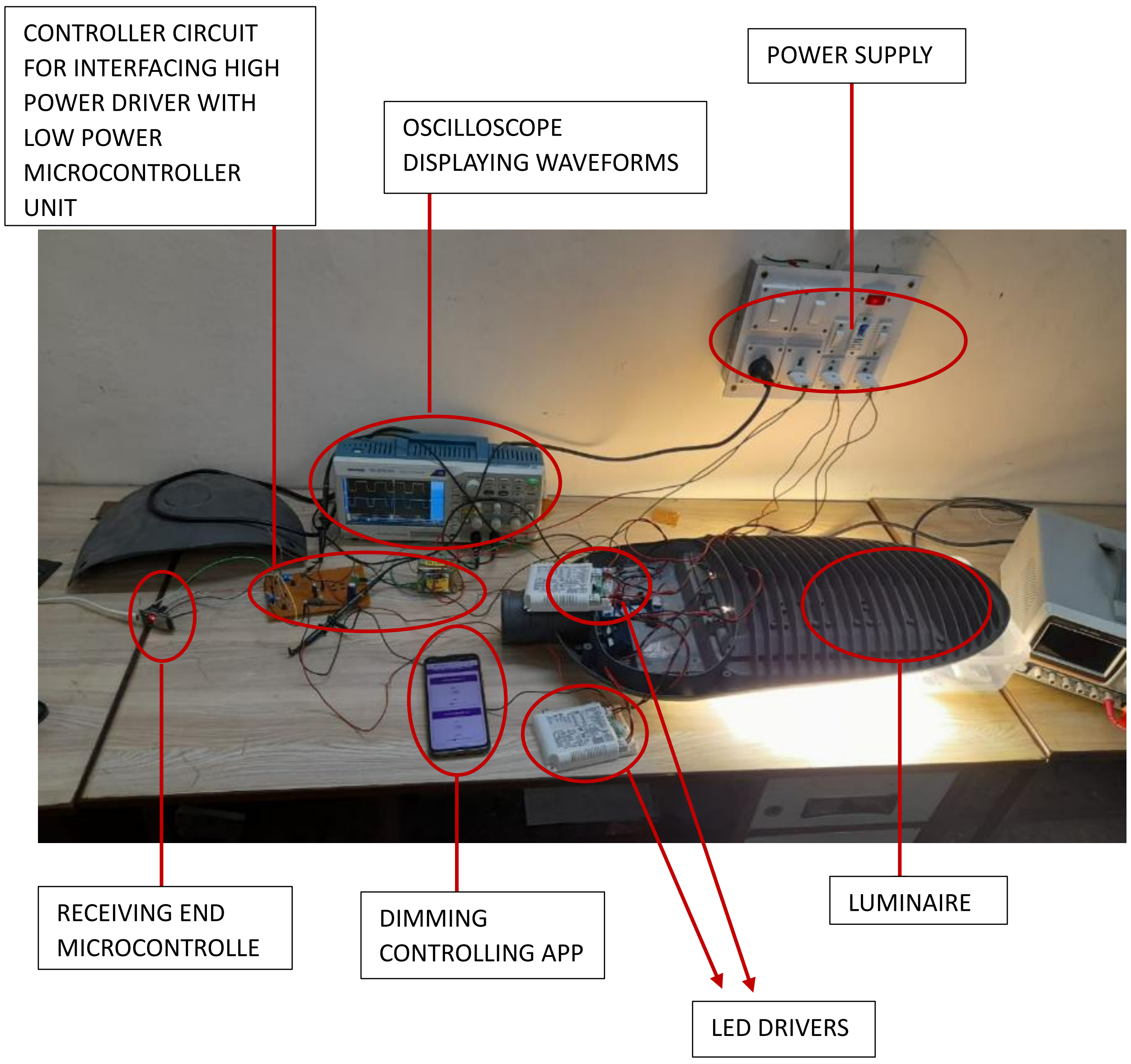

CONTROLLER CIRCUIT FOR INTERFACING HIGH POWER DRIVER WITH LOW POWER MICROCONTROLLER UNIT
OSCILLOSCOPE DISPLAYING WAVEFORMS
POWER SUPPLY
RECEIVING END MICROCONTROLLE
DIMMING CONTROLLING APP
LUMINAIRE
LED DRIVERS

Fig 4.12 Receiving end setup showing various components used during development of the system

The Fig 4.12 shows the setup for the Receiver Unit. The webpage for controlling the dimming level of luminaire is shown along with the waveforms of input to the driver circuits. The microcontroller is shown along with the Controller circuit. The webpage is accessible to the user as the user connects to the microcontroller WiFi. The LED Drivers power the LEDs in the street luminaire. The receiving end microcontroller has been receiving the information, then the signal amplified by the controller circuit is used for the driving of the LED Drivers.

4.3 OBSTACLE DETECTION UNIT

4.3.1 Flowchart for Obstacle Detection Unit 1

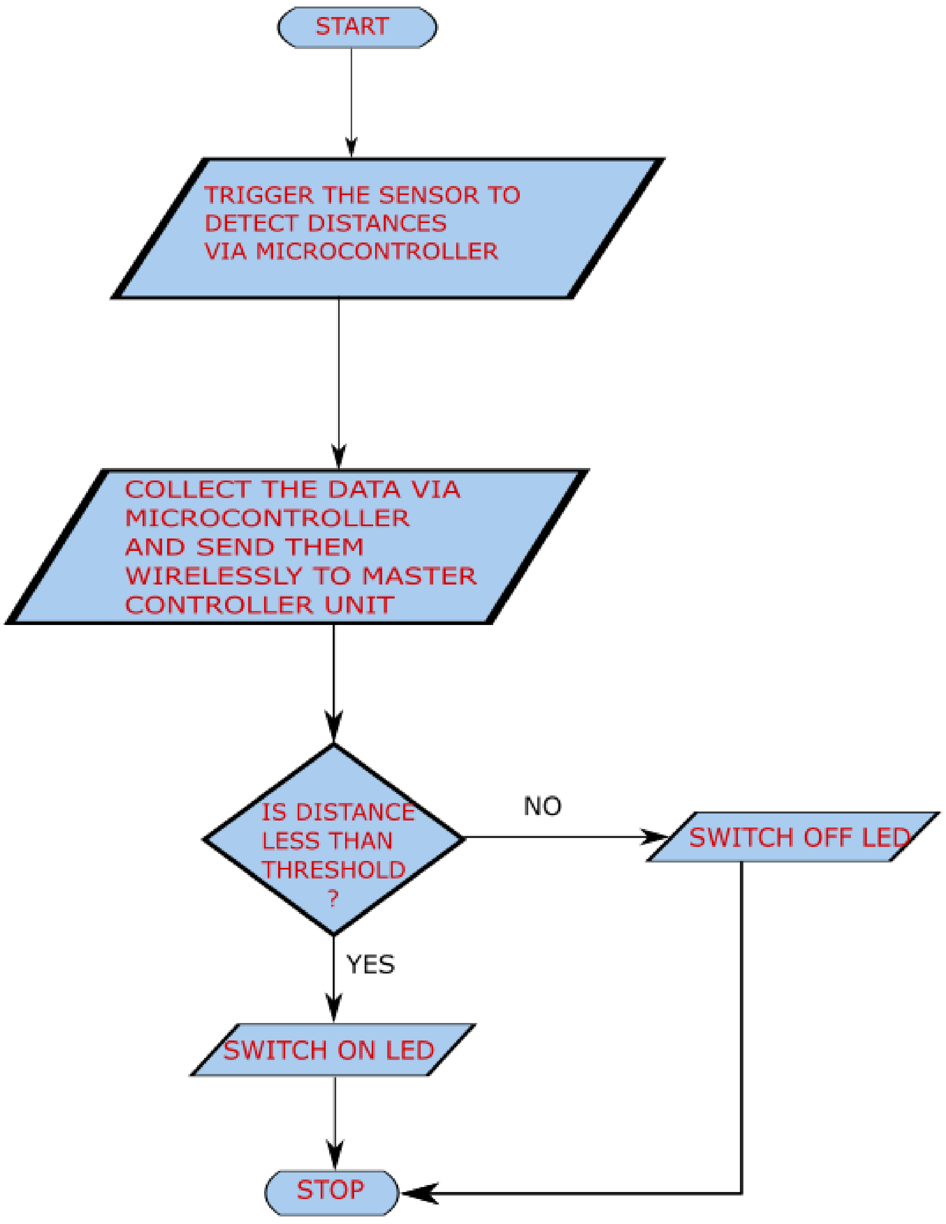

START
TRIGGER THE SENSOR TO DETECT DISTANCES VIA MICROCONTROLLER
COLLECT THE DATA VIA MICROCONTROLLER AND SEND THEM WIRELESSLY TO MASTER CONTROLLER UNIT
IS DISTANCE LESS THAN THRESHOLD ?
NO
SWITCH OFF LED
YES
SWITCH ON LED
STOP

Fig 4.13 Flowchart showing the operation of Obstacle detection Unit1

The above figure shows the flowchart for working of the obstacle detection unit. In the first step, the ultrasonic sensor is triggered via the microcontroller. The distance data are measured and sent to the Master of the Master Controller Unit. The distances measured are compared with the threshold distance set by the programmer. If the distances are less than the threshold, the led has been switched ON else OFF.

4.3.2 Software code for Obstacle Detection Unit-1

```
#include <esp_now.h>
#include <WiFi.h>

const int trigPin = 5;
const int echoPin = 18;
#define LED_PIN    23
int led_state = LOW;

//define sound speed in cm/uS
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

// REPLACE WITH THE RECEIVER'S MAC Address
uint8_t broadcastAddress[] = {0xC8, 0xF0, 0x9E, 0x9F, 0xAE, 0x6C};

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
   int id; // must be unique for each sender board
   int x = 0;
   int z = 0;
   int c = 0;
   int p = 0;
   int y;
} struct_message;
```

```
// Create a struct_message called myData
struct_message myData;

// Create peer interface
esp_now_peer_info_t peerInfo;

// callback when data is sent
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
}

void setup() {
  // Init Serial Monitor
  Serial.begin(115200);

  pinMode(LED_PIN, OUTPUT);
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Trasnmitted packet
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // Add peer
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }
}

void loop() {

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distanceCm = duration * SOUND_SPEED/2;
  if(distanceCm<30)
  {
    digitalWrite(LED_PIN, HIGH);
  }
   else{
     digitalWrite(LED_PIN, LOW);
   }

  // Prints the distance in the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  // Set values to send
  myData.id = 1;
  myData.p = distanceCm;
  myData.x = 0;
  myData.y = 0;
  myData.z = 0;
  myData.c = 0;

  // Send message via ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData, sizeof(myData));

  if (result == ESP_OK) {
    Serial.println("Sent with success");
  }
  else {
    Serial.println("Error sending the data");
  }
  delay(10);
}
```

4.3.3 Code Explanation :

In the above code, the ultrasonic sensor has been interfaced with microcontroller. The sensor will detect the distance of objects and will feed the distance data to the microcontroller. The microcontroller will compare the incoming distances with a threshold distance (here it is 30cm).If the incoming distance is less than 30cm,the microcontroller will switch on a led attached to it.

The microcontroller will send the distance data wirelessly to the Master controller whose address has been set up in the code. Since there are two such units, the id numbers are set accordingly.

4.3.4 Schematics for Obstacle Detection Unit 1

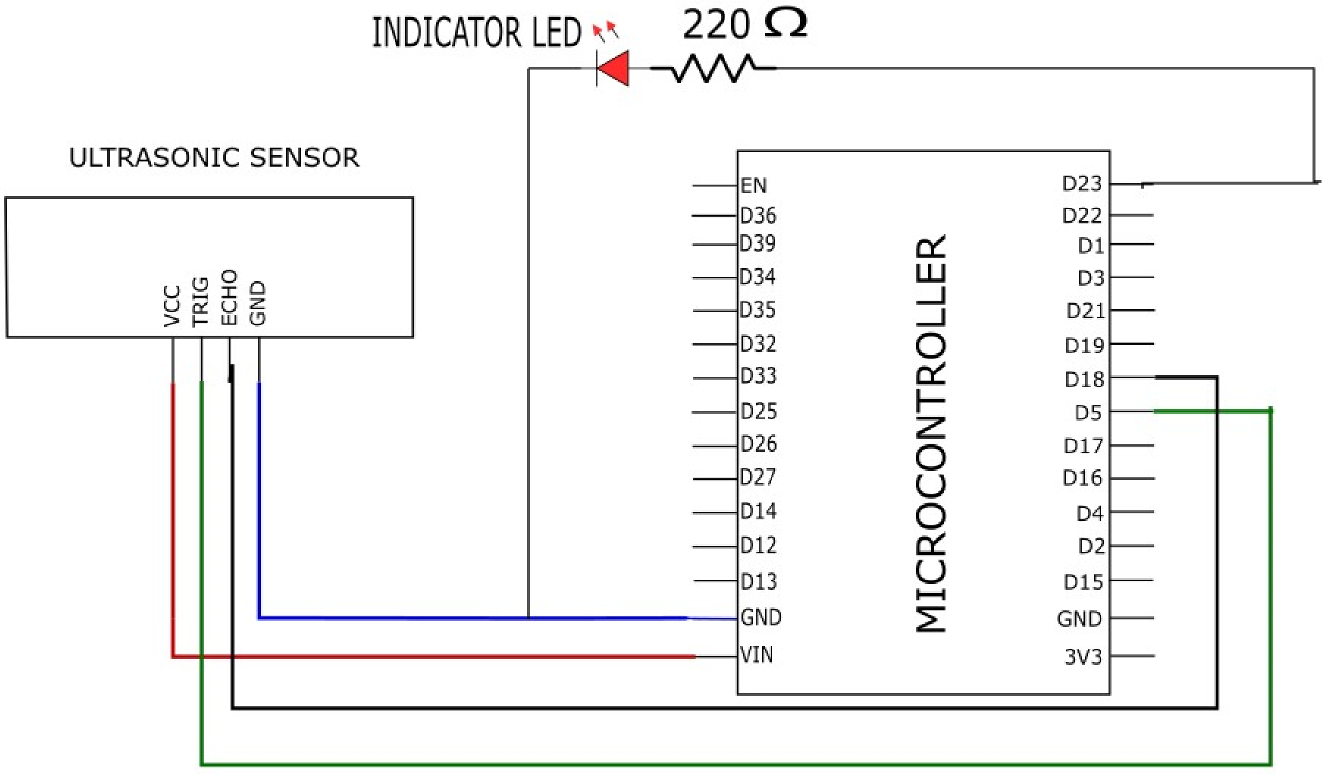


Fig 4.14 Schematics of Obstacle detection unit 1

The schematics shown represents the Obstacle Detection Unit which monitors the distances via the ultrasonic sensor and transmits the data(distances) to the Master Controller wirelessly. There are two such Units located at two different locations and they continuously monitor the distances at the two locations.

4.3.5 Flowchart for Obstacle Detection Unit 2

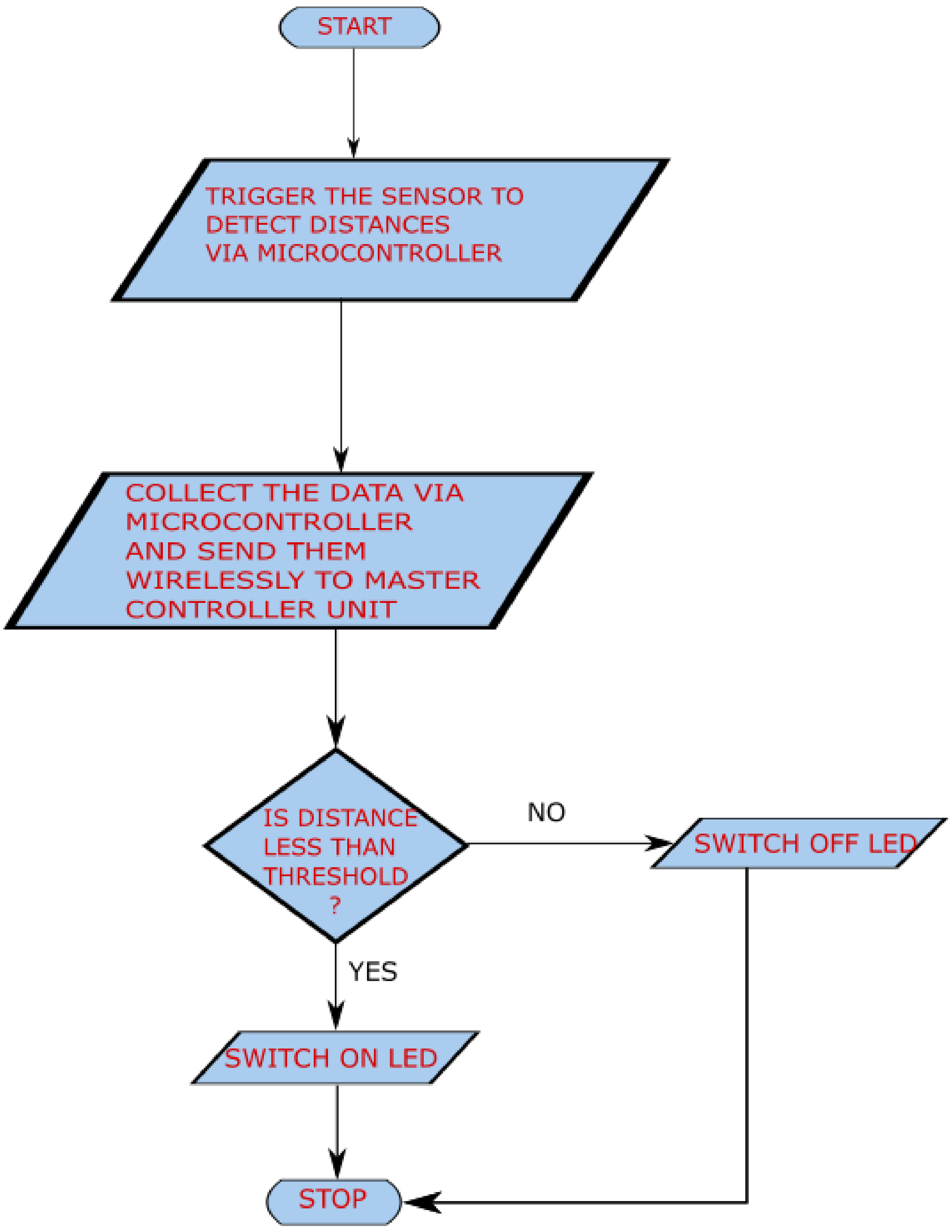

START
TRIGGER THE SENSOR TO DETECT DISTANCES VIA MICROCONTROLLER
COLLECT THE DATA VIA MICROCONTROLLER AND SEND THEM WIRELESSLY TO MASTER CONTROLLER UNIT
IS DISTANCE LESS THAN THRESHOLD ?
NO
SWITCH OFF LED
YES
SWITCH ON LED
STOP

Fig 4.15 Flowchart showing the operation of Obstacle Detection Unit 2

The above figure shows the flowchart for working of the obstacle detection unit. In the first step, the ultrasonic sensor is triggered via the microcontroller. The distance data are measured and sent to the Master of the Master Controller Unit. The distances measured are compared with the threshold distance set by the programmer. If the distances are less than the threshold, the led has been switched ON else OFF.

4.3.6 Software code for Distance Observation Unit-2

```
#include <esp_now.h>
#include <WiFi.h>

const int trigPin = 5;
const int echoPin = 18;
#define LED_PIN    23
int led_state = LOW;

//define sound speed in cm/uS
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

// REPLACE WITH THE RECEIVER'S MAC Address
uint8_t broadcastAddress[] = {0xC8, 0xF0, 0x9E, 0x9F, 0xAE, 0x6C};

// Structure example to send data
// Must match the receiver structure
typedef struct struct_message {
   int id; // must be unique for each sender board
   int x = 0;
   int z = 0;
   int c = 0;
   int p = 0;
   int y;
} struct_message;
```

```
// Create a struct_message called myData
struct_message myData;

// Create peer interface
esp_now_peer_info_t peerInfo;

// callback when data is sent
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail");
}

void setup() {
  // Init Serial Monitor
  Serial.begin(115200);

  pinMode(LED_PIN, OUTPUT);
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Once ESPNow is successfully Init, we will register for Send CB to
  // get the status of Trasnmitted packet
  esp_now_register_send_cb(OnDataSent);

  // Register peer
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  // Add peer
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }
}

void loop() {

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distanceCm = duration * SOUND_SPEED/2;
  if(distanceCm<30)
  {
    digitalWrite(LED_PIN, HIGH);
  }
    else{
      digitalWrite(LED_PIN, LOW);
    }
  // Prints the distance in the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);

  // Set values to send
  myData.id = 2;
  myData.p = distanceCm;
  myData.x = 0;
  myData.y = 0;
  myData.z = 0;
  myData.c = 0;

  // Send message via ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData, sizeof(myData));

  if (result == ESP_OK) {
    Serial.println("Sent with success");
  }
  else {
    Serial.println("Error sending the data");
  }
  delay(10);
}
```

4.3.7 Code Explanation :

In the above code, the ultrasonic sensor has been interfaced with microcontroller. The sensor will detect the distance of objects and will feed the distance data to the microcontroller. The microcontroller will compare the incoming distances with a threshold distance (here it is 30cm).If the incoming distance is less than 30cm,the microcontroller will switch on a led attached to it.

The microcontroller will send the distance data wirelessly to the Master controller whose address has been set up in the code. Since there are two such units, the id numbers are set accordingly.

4.3.8 Schematics for Obstacle Detection Unit 2

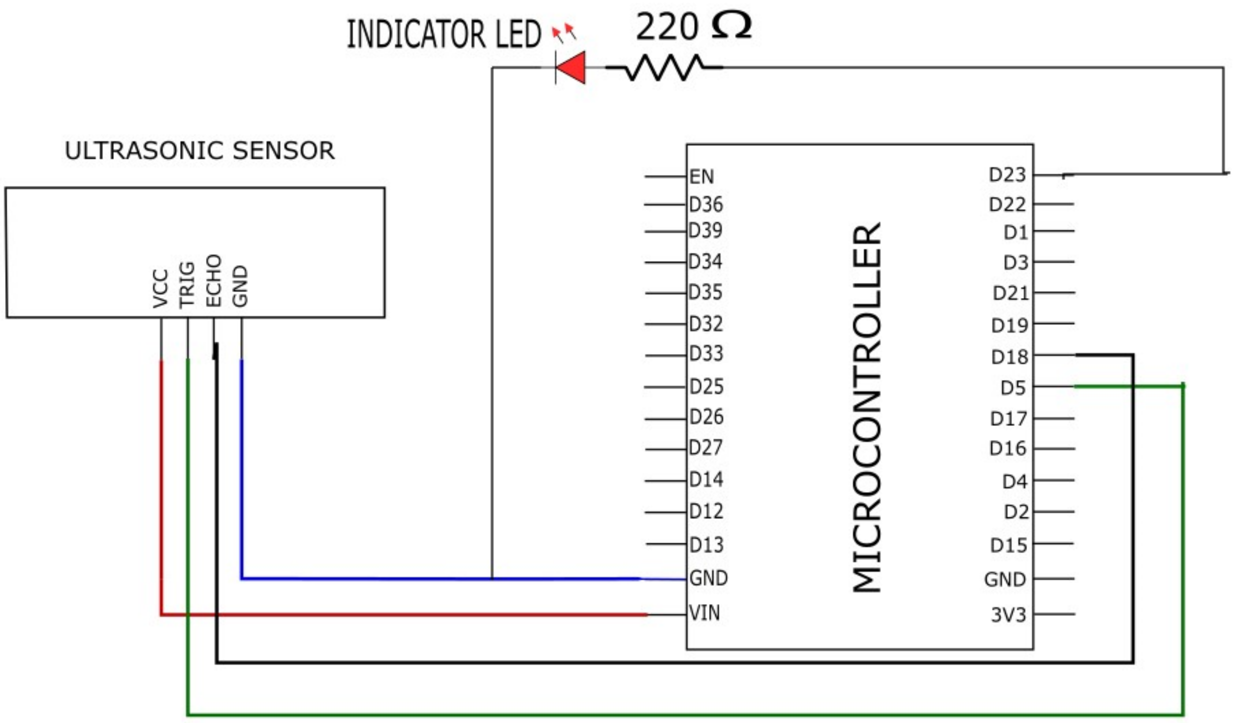


Fig 4.16

Schematics of Obstacle detection Unit 2

The schematics shown in Fig 4.16 represents the Distance Observation Unit which monitors the distances via the ultrasonic sensor and transmits the data(distances) to the Master Controller wirelessly. There are two such Units located at two different locations and they continuously monitor the distances at the two locations.

4.3.9 MAC Address generation code

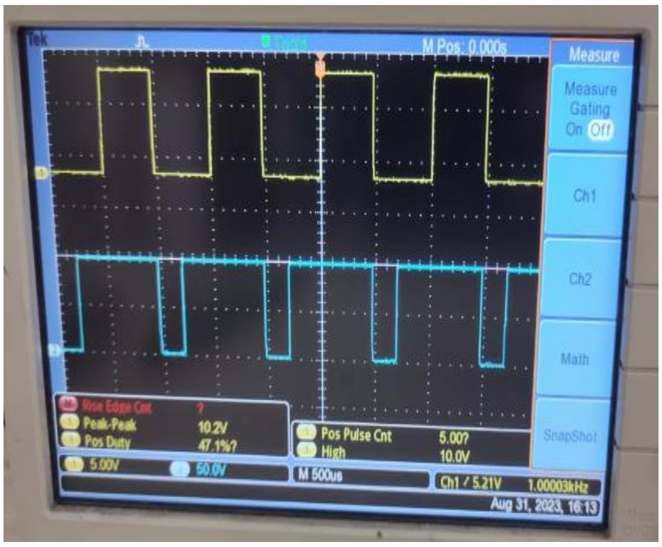
Software code:

```
#include "WiFi.h"

void setup(){
  Serial.begin(115200);
  WiFi.mode(WIFI_MODE_STA);
  Serial.println(WiFi.macAddress());
}

void loop(){

}
```

5.RESULTS OBTAINED

5.1WAVEFORMS OBTAINED FROM THE RECEIVER

Here, we have shown the amplified pwm output signals ontained from the controller circuit , used as input to led drivers for driving them.



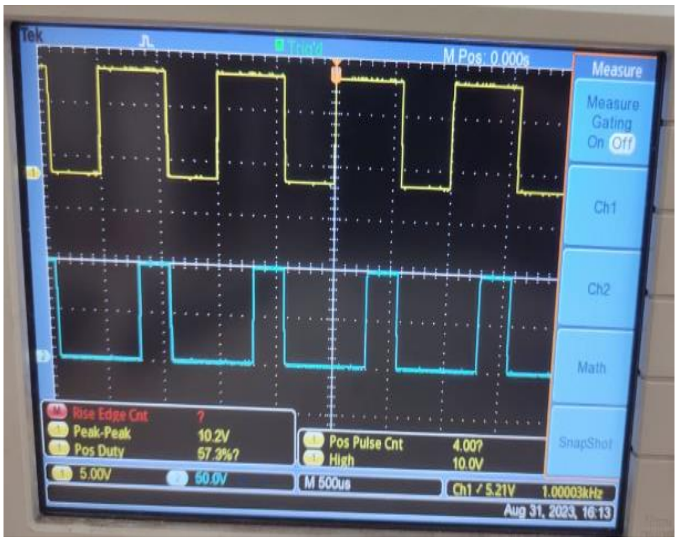


Fig 5.1 (a) Fig 5.1 (b)

Fig 5.1 Waveforms showing dissimilar pwm input signals to the led drivers.

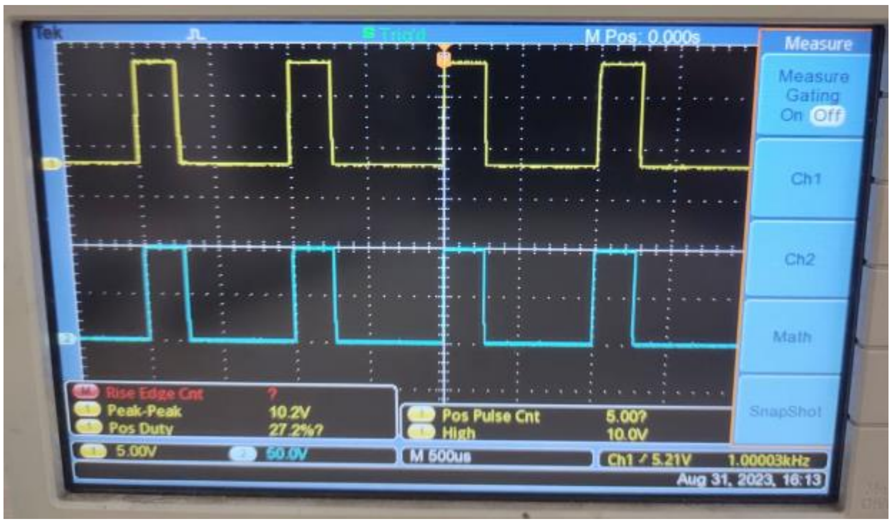


Fig 5.2 Waveforms showing similar input pwm signals to the led drivers

The above figures show the waveforms at the output end of the controller circuit .The controller circuit amplifies the pwm signals of the microcontroller to drive the LED drivers.

5.2 PERFORMANCE ANALYSIS FOR A SINGLE LED DRIVER

Table 5.1 Table showing variations of dc pwm current drawn by the led driver with change in duty cycle at the receiving end, indicated by ammeter.

DUTY CYCLE(%)	DC PWM CURRENT DRAWN BY DRIVER INDICATED BY AMMETER (A)
0	0
10	0.072
20	0.143
30	0.212
40	0.282
50	0.349
60	0.419
70	0.486
80	0.556
90	0.622
100	0.691

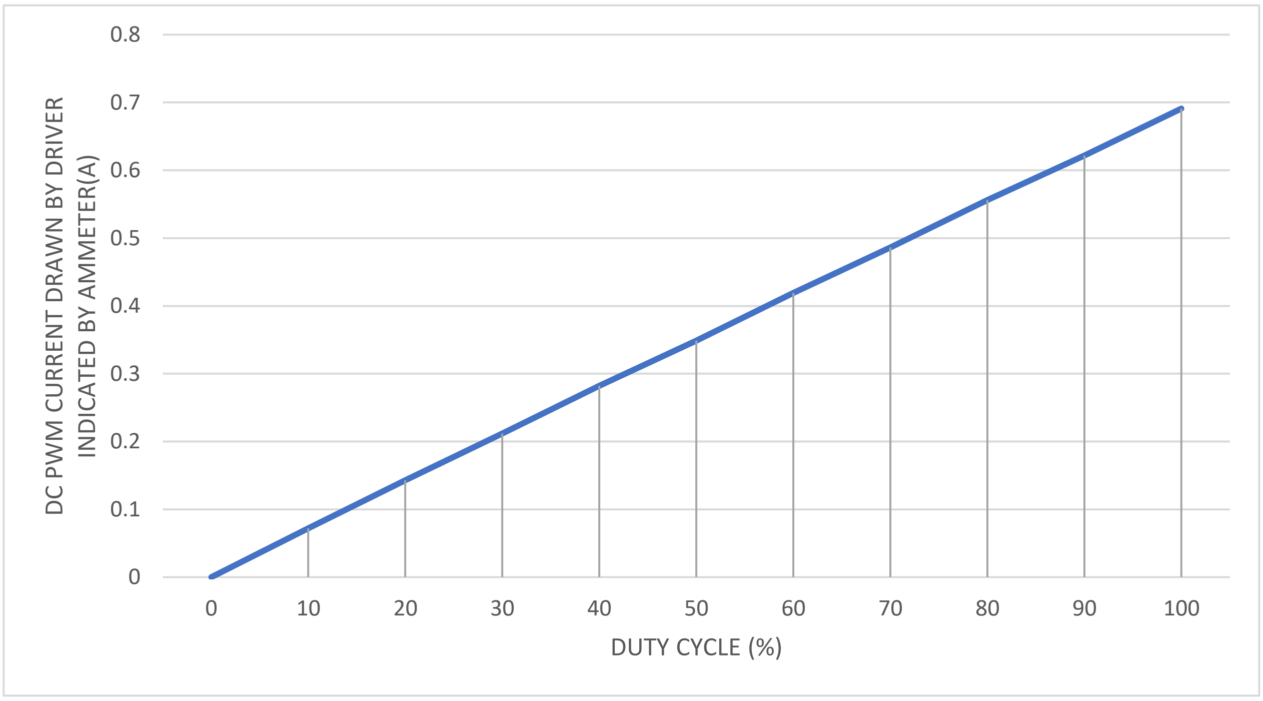


Fig 5.3 Graph showing the variations in current with respect to variations in duty cycle

Description : The led driver is controlled by applying 10 V pwm signal. This signal is fed from the controller circuit. An ammeter is connected in series with the driver circuit as shown. The duty cycle is varied and current readings are noted. From the graph we can see a perfectly linear variation.

As we can see from the block diagram shown above , the ammeter is connected in series with the driver circuit and is placed inbetween the controller circuit and the driver circuit,driving the led array.

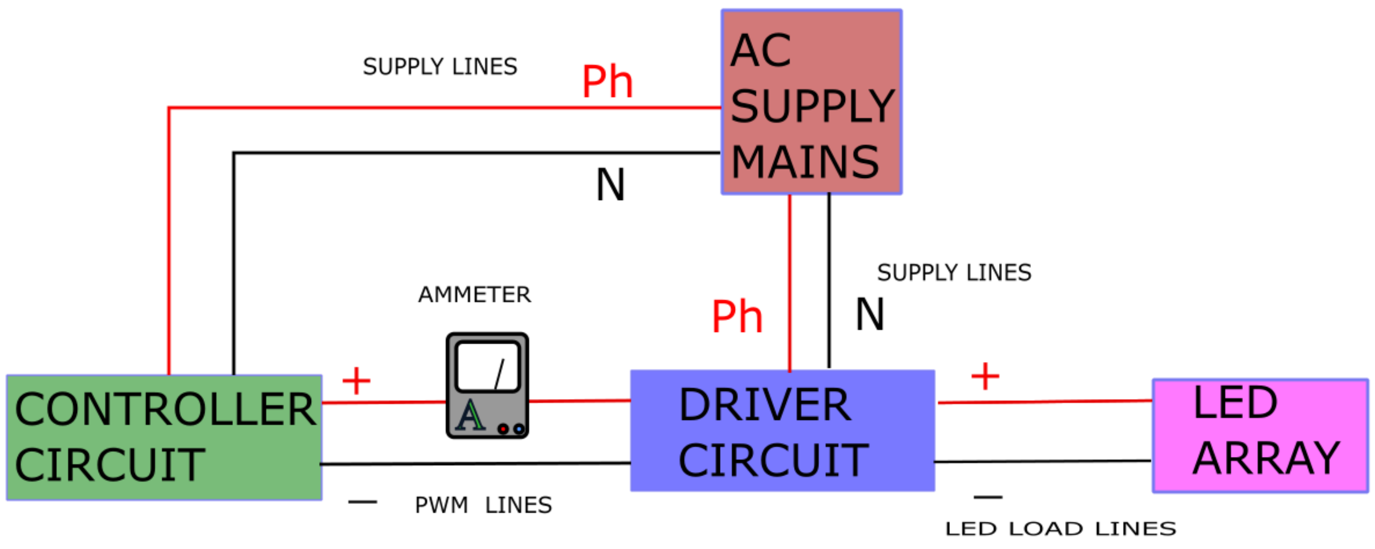


Fig 5.4 Block diagram showing the arrangement used for taking current readings.

Table 5.2: Table showing variations of ac current drawn by the led driver with change in duty cycle at the receiving end, indicated by ammeter.

DUTY CYCLE (%)	AC CURRENT DRAWN BY LED DRIVER MEASURED BY AMMETER (A)
0	0
10	0.038
20	0.057
30	0.079
40	0.102
50	0.124
60	0.143
70	0.166
80	0.19
90	0.213
100	0.236

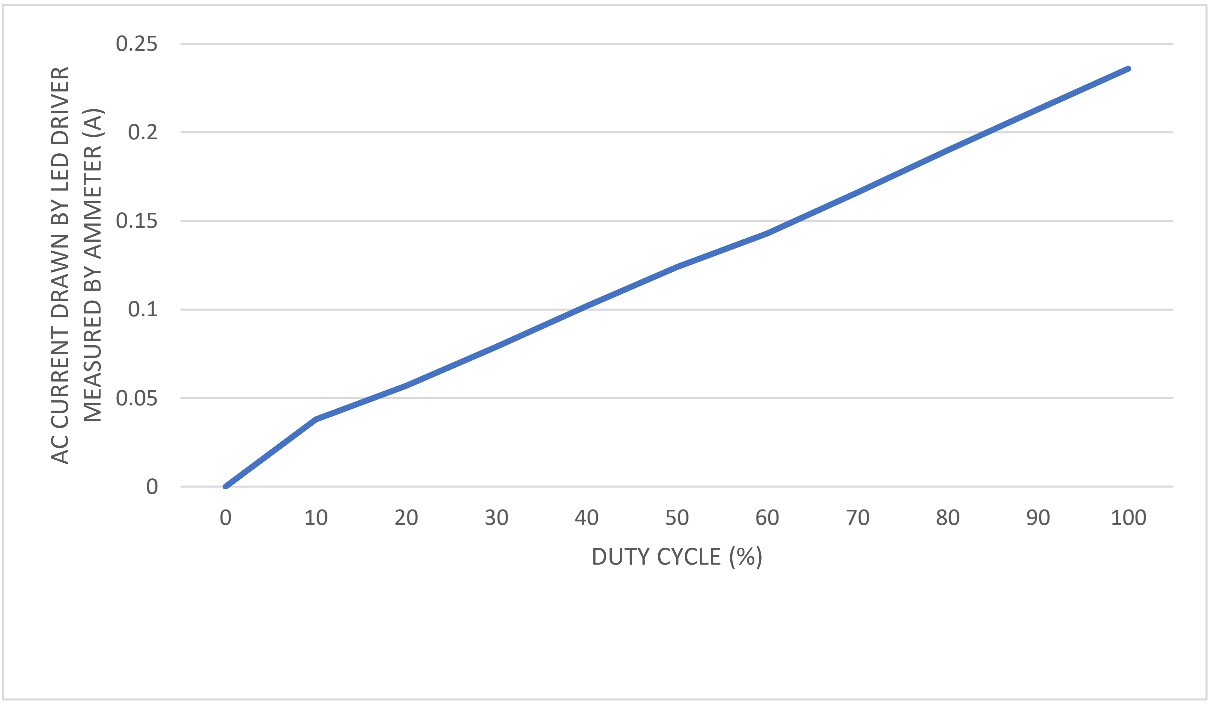


Fig 5.5 Graph showing the variations in current with respect to variations in duty cycle

Description: The block diagram shows the arrangement used for taking the current readings in the ammeter. The ammeter is connected in series with led driver, but now is placed inbetween supply mains and the led driver. The duty cycle is varied and the current readings are noted for each duty cycle .The values are tabulated and the graph is ploted for checking the nature of variation.

We can see from the graph, that there is an almost linear variation , but during the first change of duty cycle, the driver takes a little more current, resulting in the kink in the graph at duty cycle 10%. The graph also shows that at 60% duty cycle a current dip is observed.

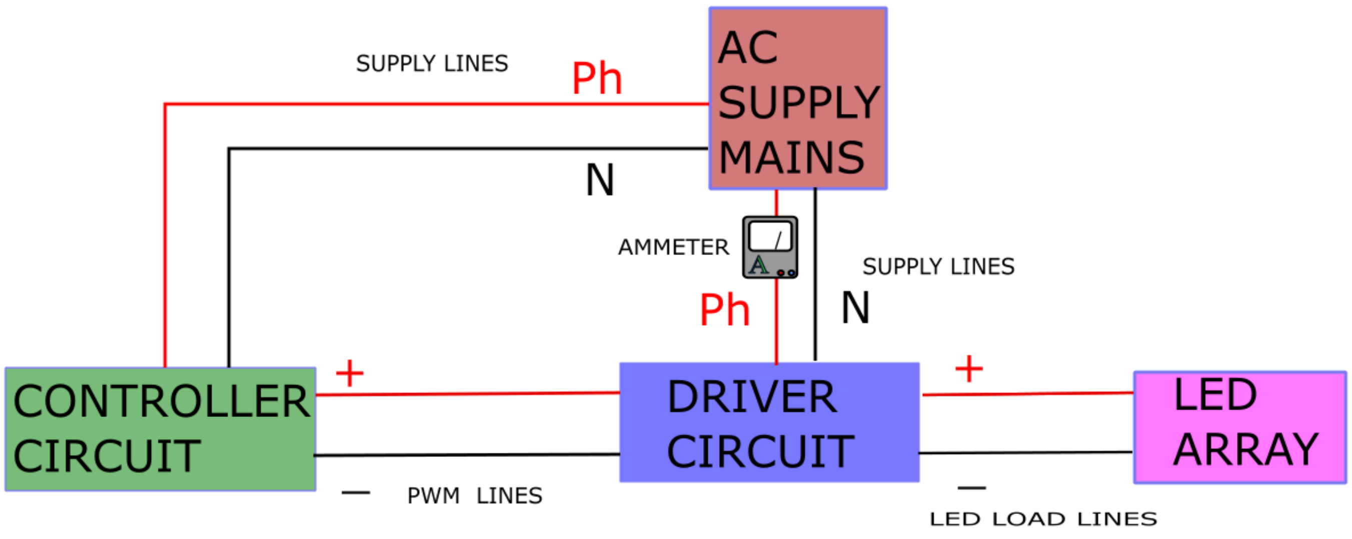


Fig 5.6 Block diagram showing the arrangement used for taking current readings.

Table 5.3: Table showing variations of dc current drawn by the led with change in duty cycle at the receiving end, indicated by ammeter.

DUTY CUCLE (%)	DC CURRENT DRAWN BY LEDS INDICATED BY AMMETER (A)
0	0
10	0.049
20	0.114
30	0.185
40	0.253
50	0.323
60	0.39
70	0.46
80	0.532
90	0.6
100	0.67

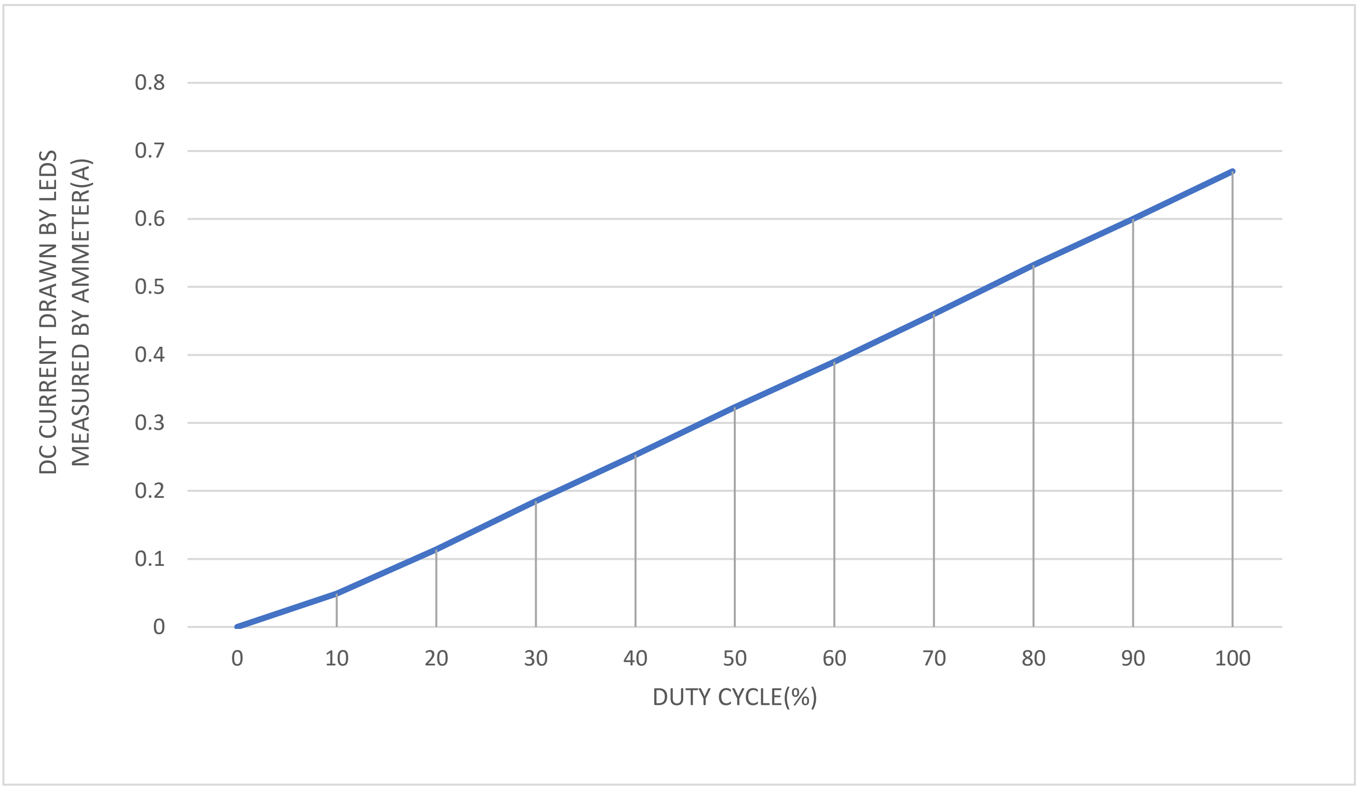


Fig 5.7 Graph showing the variations in current with respect to variations in duty cycle

Description: The block diagram shows the arrangement used for taking the current readings in the ammeter. The ammeter is now connected in series led array and placed in between the led driver and the led array. The duty cycle is varied and the current values are noted and tabulated for each duty cycle.

We can see from the graph that an almost linear variation is obtained but at 10% duty cycle there is an increase in the amount of dc current, thereby giving a slight kink in the graph.

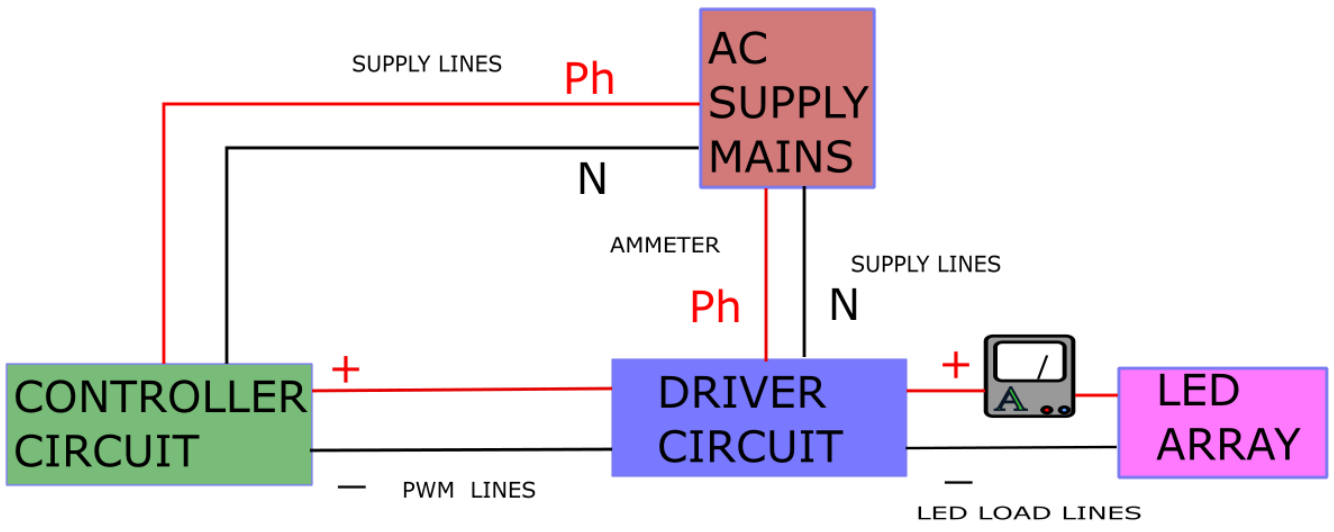


Fig 5.8 Block diagram showing the arrangement used for taking current readings.

CURRENT VARIATIONS SENSED BY CURRENT SENSOR

Table 5.4 Table showing variations of dc pwm current drawn by the led driver with change in duty cycle at the receiving end, indicated by current sensor.

DUTY CYCLE(%)	DC PWM CURRENT READING DRAWN BY LED DRIVER(SENSOR READING)(mA)
0	26.64
10	27.15
20	27.73
30	28.58
40	28.07
50	28.55
60	29.29
70	28.46
80	28.76
90	29.02
100	29.3

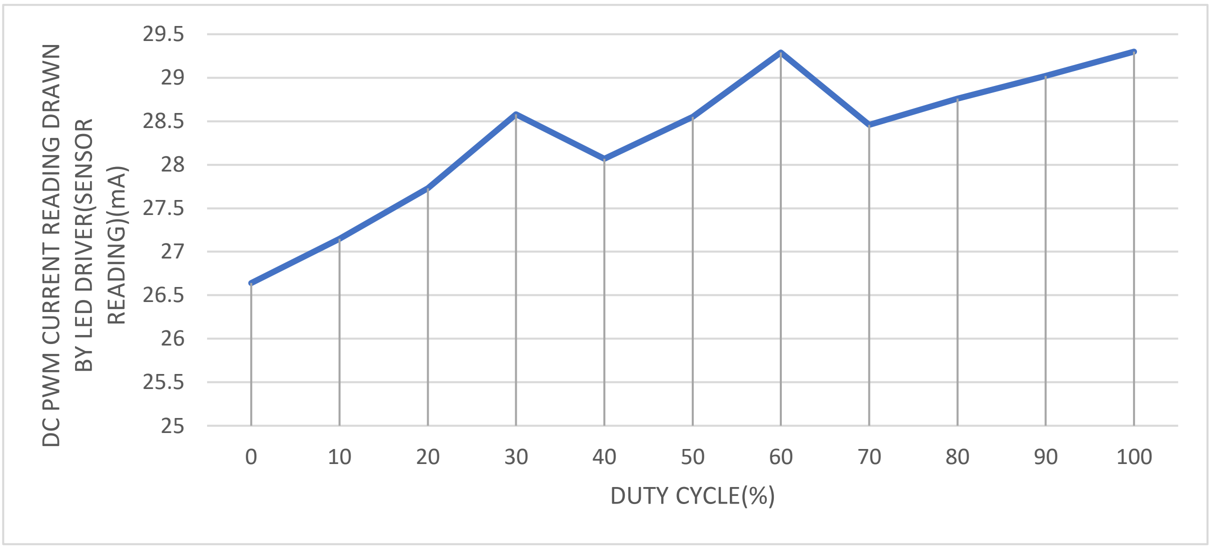


Fig 5.9 Graph showing the variations in current with respect to variations in duty cycle

Description: The block diagram for the experimental setup has been shown. The current sensor is connected in series with the led driver circuit and connected in between the controller circuit and the driver circuit. The duty cycle is varied and the current drawn by the led driver circuit is noted in the current sensor. The values are tabulated and the graph of the variation is shown.

We see from the graph that with increase in duty cycle , there is an increase in the current drawn but at some duty cycles, sharp peaks occur. These fluctuations rise probably because of the sensitivity and the delay in response time of the sensor. This is a major problem in current sensors as their calibration has to be done according to the system which is being tested.

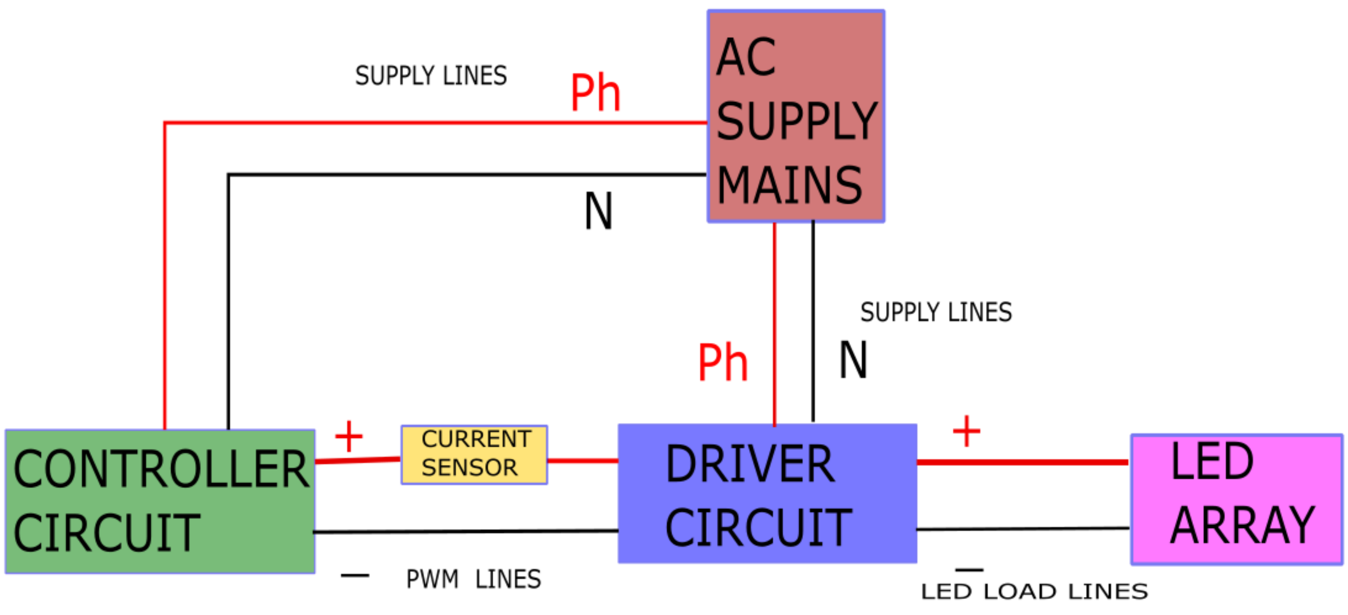


Fig 5.10 Block diagram showing the arrangement used for taking current readings.

Table 5.5: Table showing variations of ac current drawn by the led driver with change in duty cycle at the receiving end, indicated by current sensor

DUTY CYCLE(%)	AC CURRENT READING DRAWN FROM SUPPLY (CURRENT SENSOR READING) (mA)
0	648.47
10	686.33
20	683.055
30	702.08
40	722.08
50	747.29
60	747.50
70	770.815
80	798.87
90	810.91
100	832.72

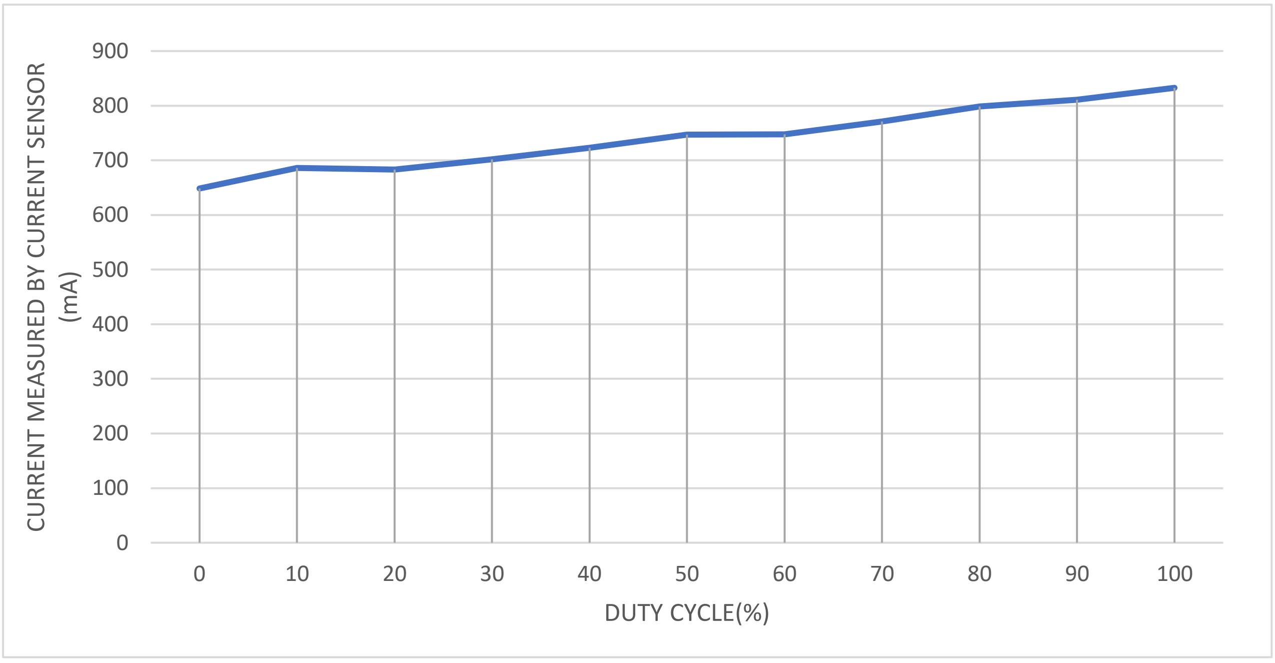


Fig 5.11 Graph showing the variations in current with respect to variations in duty cycle

Description: The block diagram for the experimental setup has been shown. The current sensor is connected in series with the led driver circuit and connected in between the controller circuit and the driver circuit. The duty cycle is varied and the current drawn by the led driver circuit is noted in the current sensor. The values are tabulated and the graph of the variation is shown.

We see from the graph that with increase in duty cycle , there is an increase in the current drawn but there are some irregularities. These fluctuations rise probably because of the sensitivity and the delay in response time of the sensor. This is a major problem in current sensors as their calibration has to be done according to the system which is being tested.

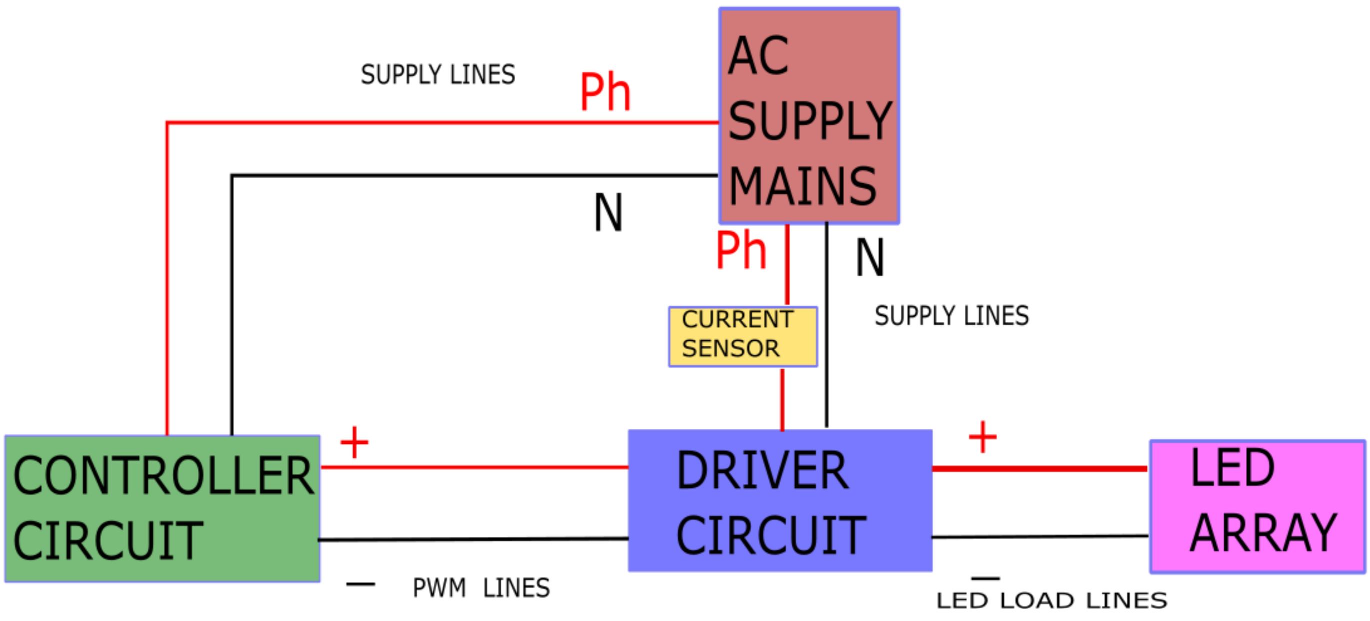


Fig 5.12 Block diagram showing the arrangement used for taking current readings.

Table 5.6: Table showing variations of ac current drawn by the led driver with change in duty cycle at the receiving end, indicated by current sensor

DUTY CYCLE(%)	DC CURRENT READING DRAWN BY LED(SENSOR READING) (mA)
0	123.88
10	125.20
20	128.634
30	130.35
40	133.054
50	135.08
60	137.47
70	139.26
80	142.45
90	145.67
100	148.35

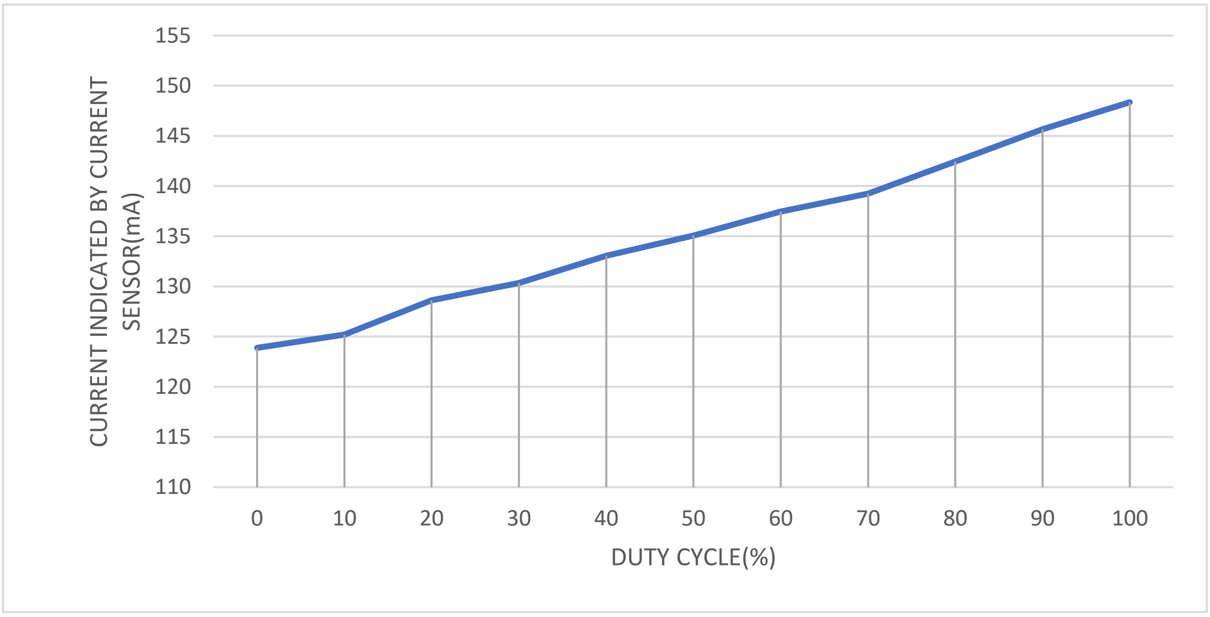


Fig 5.13 Graph showing the variations in current with respect to variations in duty cycle

Description: The block diagram for the experimental setup has been shown . Here, the current sensor is connected in series with the led array. The duty cycle is varied and the readings of the current sensor is noted. These variations are plotted and we can see from the graph that with increase in duty cycle, there is an increase in the current readings.

The plot shows that the variation is almost linear but irregular, suggesting that the current sensor's sensitivity is not good.

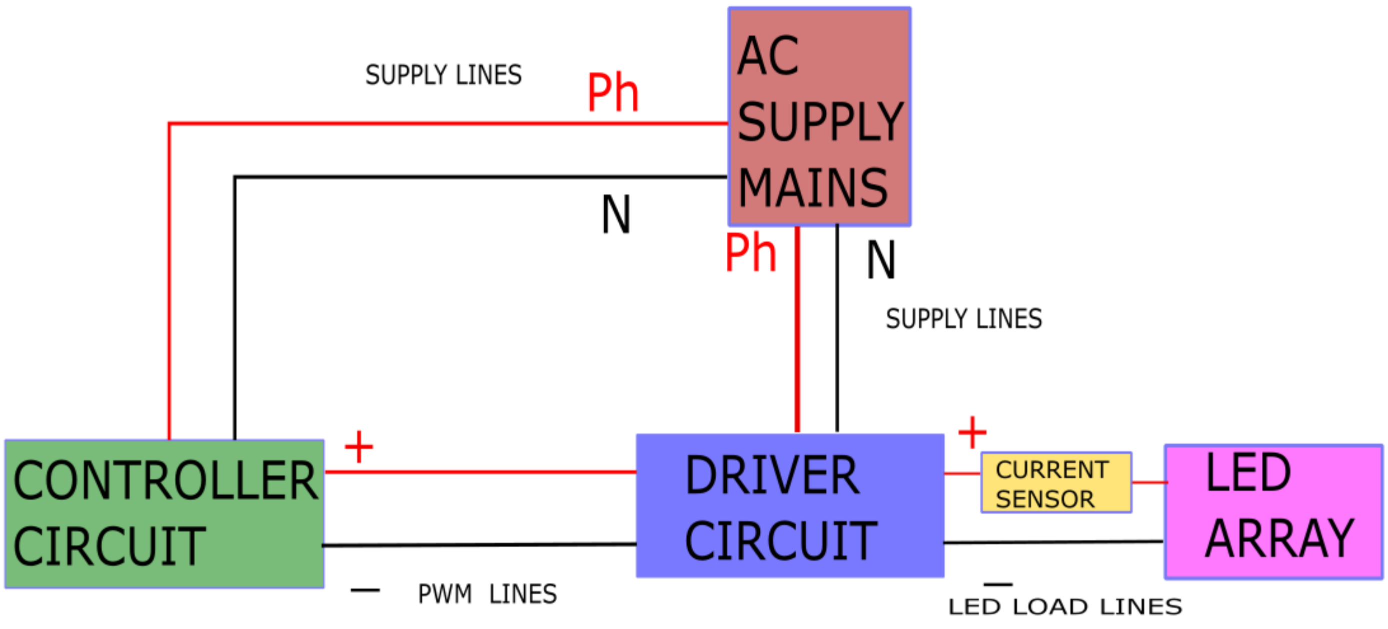


Fig 5.14 Block diagram showing the arrangement used for taking current readings.

5.3PERFORMANCE ANALYSIS OF THE ENTIRE SYSTEM

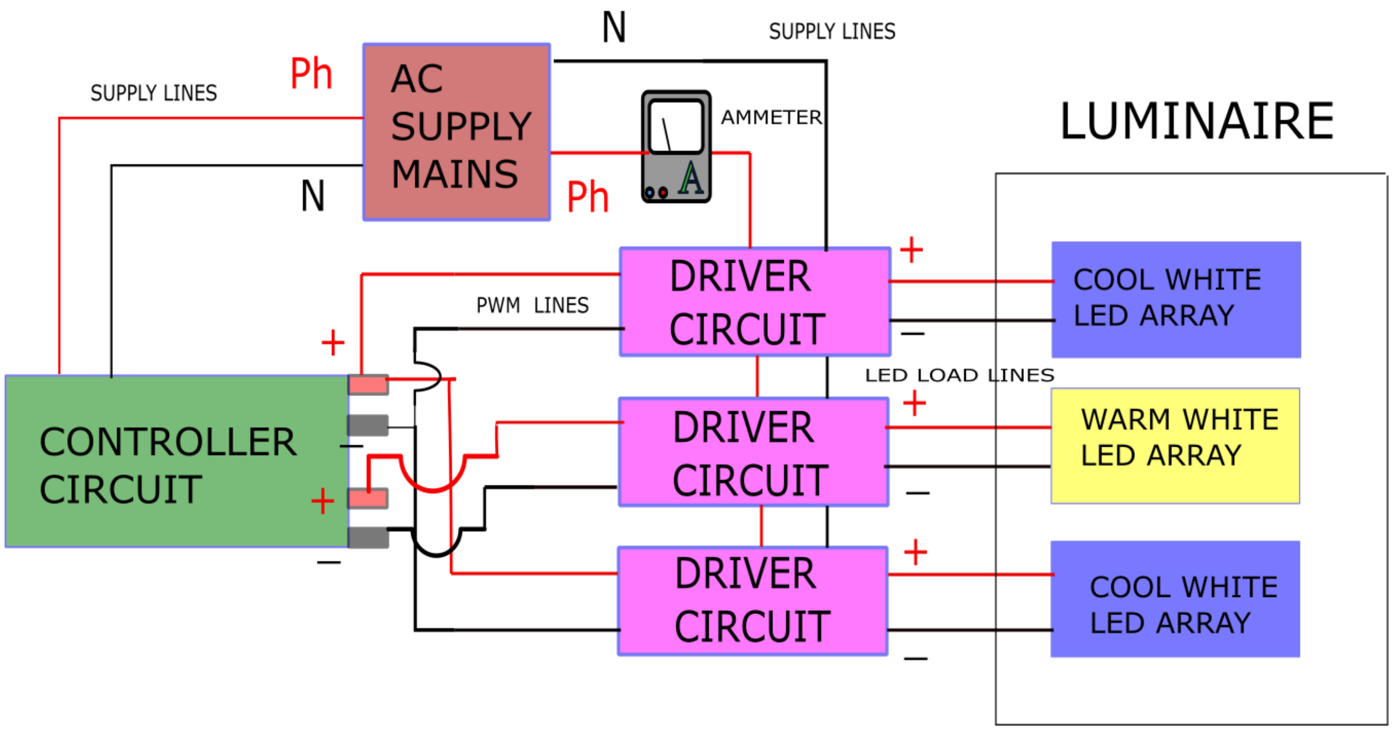


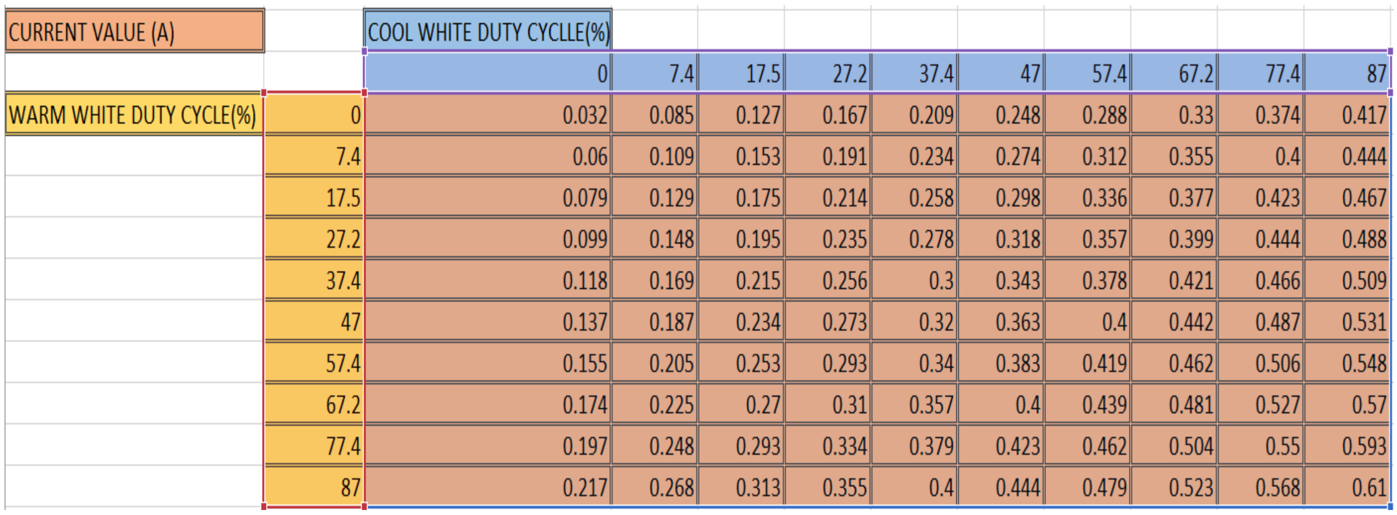
Fig 5.15 Block diagram showing the electrical connection diagram for the evaluation of the performance of the entire system

Description: The led arrays of the luminaire are powered by the respective led drivers, which are controlled by the controller circuit. The two cool white arrays are powered by two drivers which are made common while the warm white array is powered by it's respective driver circuit as shown in the figure. The warm white duty cycle is kept constant while the two cool white duty cycles are varied progressively.

The ammeter is connected as shown in the figure in series with the driver circuits and in between the ac mains and the driver circuits and the current values are noted and plotted in the graph. They all are linearly increasing except one particular reading.

A 3D plot has been plotted where warm white duty cycle is kept constant and cool white duty cycle is progressively varied. The same procedure is repeated for each increasing duty cycle for warm white duty cycle. The current level is noted for each of these variations. Plot for current variation is shown in Fig 5.14 and the plot for variation of illuminance level is shown in Fig 5.15.

Table 5.6: Table showing variations of ac current drawn by the led drivers with change in duty cycle at the receiving end, indicated by ammeter.



CURRENT VALUE (A)		COOL WHITE DUTY CYCLLE(%)									
		0	7.4	17.5	27.2	37.4	47	57.4	67.2	77.4	87
WARM WHITE DUTY CYCLE(%)	0	0.032	0.085	0.127	0.167	0.209	0.248	0.288	0.33	0.374	0.417
	7.4	0.06	0.109	0.153	0.191	0.234	0.274	0.312	0.355	0.4	0.444
	17.5	0.079	0.129	0.175	0.214	0.258	0.298	0.336	0.377	0.423	0.467
	27.2	0.099	0.148	0.195	0.235	0.278	0.318	0.357	0.399	0.444	0.488
	37.4	0.118	0.169	0.215	0.256	0.3	0.343	0.378	0.421	0.466	0.509
	47	0.137	0.187	0.234	0.273	0.32	0.363	0.4	0.442	0.487	0.531
	57.4	0.155	0.205	0.253	0.293	0.34	0.383	0.419	0.462	0.506	0.548
	67.2	0.174	0.225	0.27	0.31	0.357	0.4	0.439	0.481	0.527	0.57
	77.4	0.197	0.248	0.293	0.334	0.379	0.423	0.462	0.504	0.55	0.593
	87	0.217	0.268	0.313	0.355	0.4	0.444	0.479	0.523	0.568	0.61

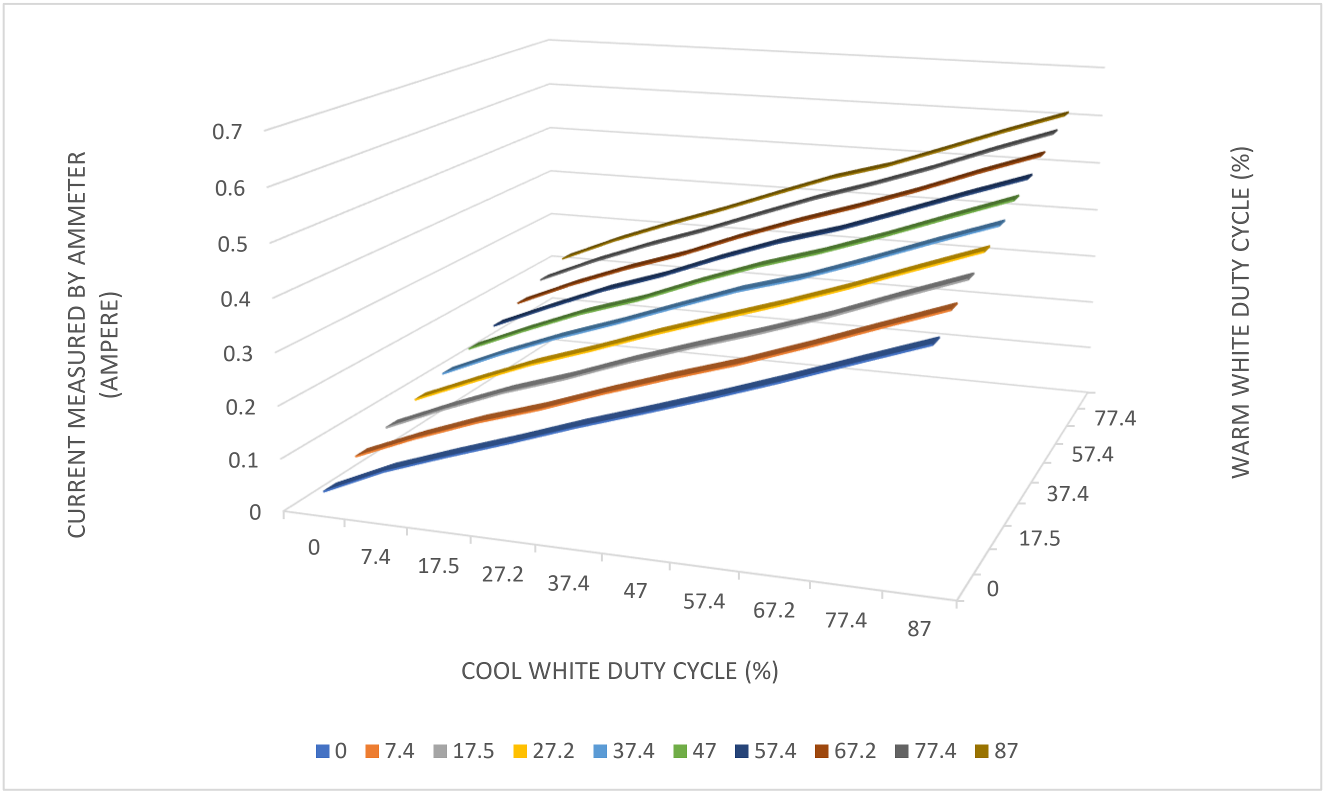


Fig 5.14 3D plot showing variation of current drawn by the system under operation.

Table 5.7: Table showing variations of illuminance level with change in duty cycle at the receiving end, indicated by lux meter

ILLUMINANCE LEVEL (lux)		COOL WHITE DUTY CYCLE(%)									
		0	7.4	17.5	27.2	37.4	47	57.4	67.2	77.4	87
WARM WHITE DUTY CYCLE(%)	0	0	14.4	27.4	38.95	50.5	60.8	71.15	80.8	90.3	99.2
	7.4	6.75	21.65	34.55	46.15	57.6	68	78.45	88	97.35	106.2
	17.5	12.85	27.65	40.6	52.1	63.65	74	84.5	93.95	103.45	112.2
	27.2	18.25	33	45.95	57.6	69.05	79.5	89.9	99.45	108.95	117.8
	37.4	23.65	38.4	51.3	62.9	74.4	84.75	95.15	104.7	114.05	123
	47	28.4	43.15	56.15	67.7	79.25	89.55	99.9	109.45	118.7	127.4
	57.4	33.25	47.95	60.95	72.5	83.95	94.3	104.7	114.1	123.4	132.2
	67.2	37.65	52.35	65.3	76.85	88.3	98.6	108.9	118.35	127.65	136.3
	77.4	41.9	56.7	69.6	81.2	92.7	102.95	113.4	122.8	132.15	140.9
	87	45.8	60.25	73.15	84.7	96.1	106.35	116.6	126.15	135.4	143.7

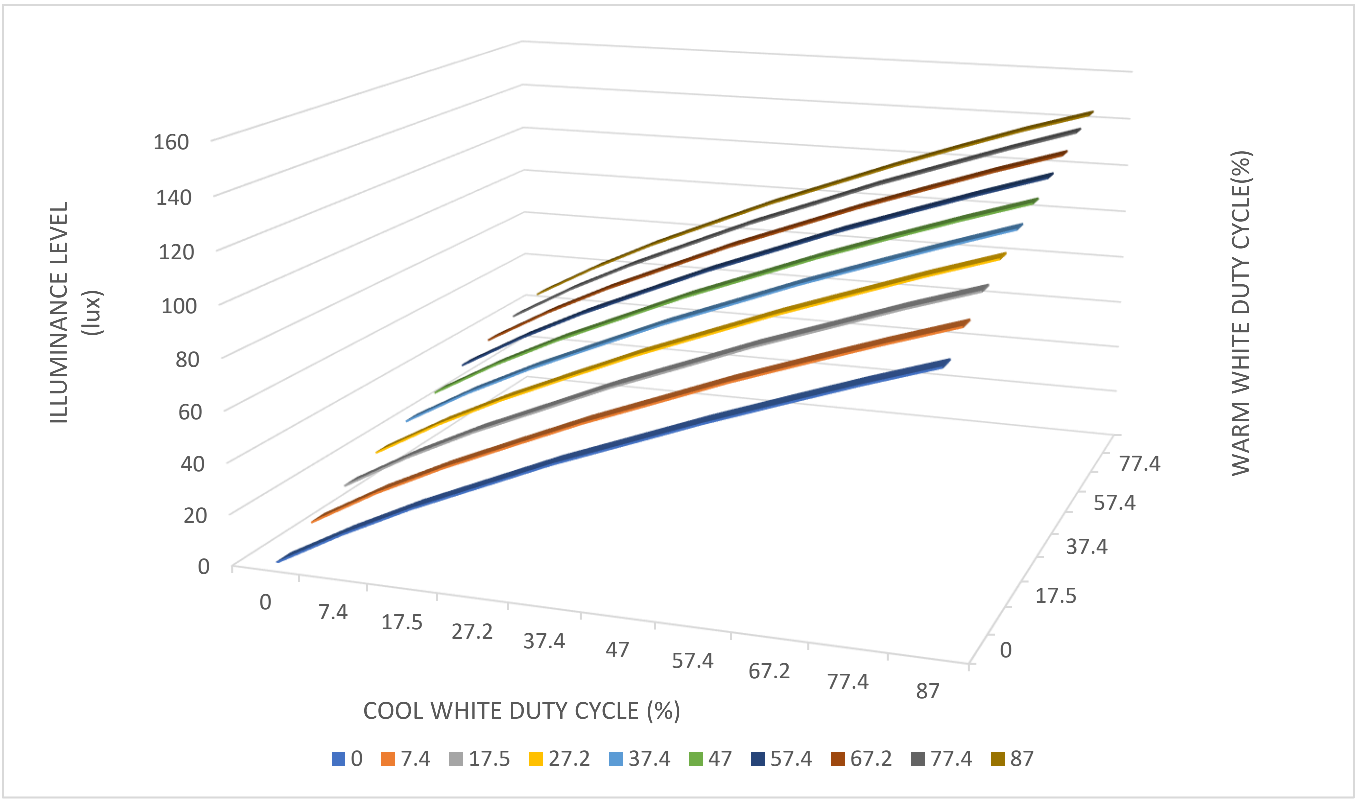


Fig 5.15 3D plot showing variation of illuminance output of the system under operation.

5.4 Fixed illumination level from a fixed height

Table 5.8 Table showing experimental results.

REQUIRED LUX LEVEL	ACHIEVED LUX LEVEL	WARM WHITE DUTY CYCLE(%)	COOL WHITE DUTY CYCLE(%)	ERROR	ERROR(%)
50	50.7	0	37.4	0.7	1.4
75	75.4	37.4	37.4	0.4	0.53
100	100.8	47.2	57.2	0.8	0.8
125	128.5	87	67.2	3.5	2.8

Table 5.8 shows the experimental results of acquiring fixed illumination levels from a fixed height of 3.605m at nadir point. As the height varies, the illumination level varies. Hence, the height of the luminaire has been kept constant.

Some specific lux levels have been taken and the duty cycles have been varied to meet the lux levels. The acquired lux levels have been recorded and the errors have also been calculated.

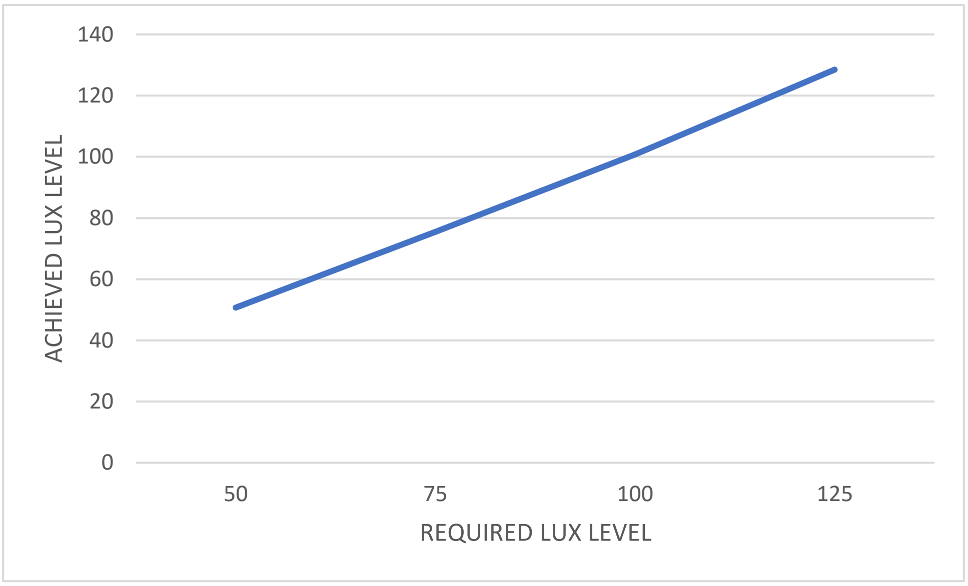


Fig 5.16 Plot showing variation of measured lux level with variation in required lux level.

Fig 5.16 shows that there is almost a linear variation between the required lux levels and the achieved lux levels. Since there is negligible error, the plot has come out to be linear.

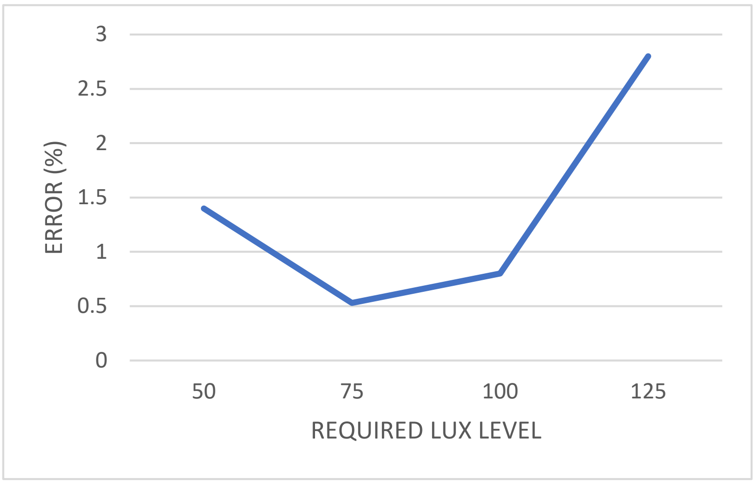
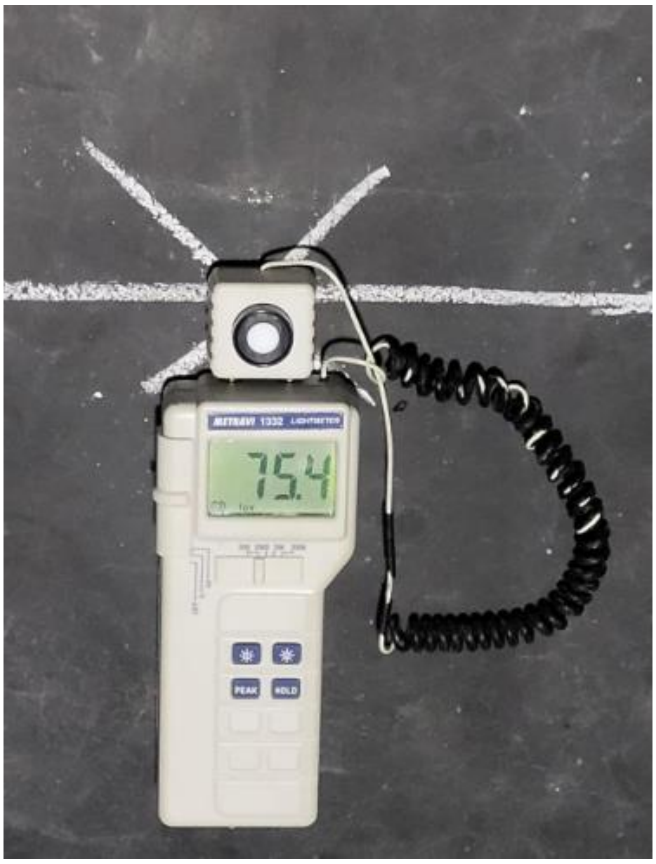


Fig 5.17 Plot showing variation of error % between required lux level and achieved lux levels with variation in required lux level

Fig 5.17 shows the variation of error with variation in required lux levels. The maximum error has been found at 125lux while the minimum has been found at 75lux



.Fig 5.18 Lux meter showing the reading taken for one of the experimental values.

The above figure shows the lux meter readings . The required lux level is 75lux, the achieved lux level is 75.4lux.-

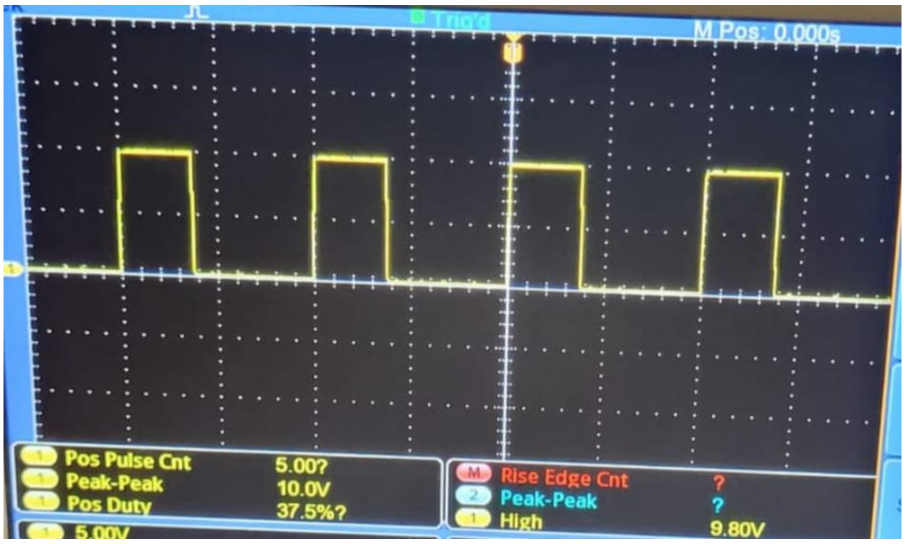


Fig 5.19 (a) Waveform for the input signal to warm white LED driver.

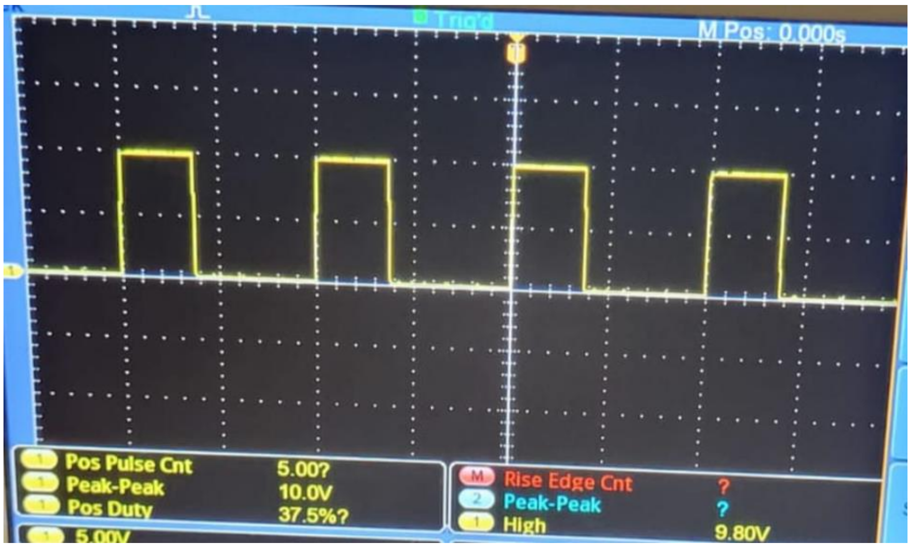


Fig 5.19(b) Waveform for the input signal to cool white LED driver

Fig 5.19 Waveforms for the input pwm signals to LED drivers.

The above waveforms show that the duty cycle % of the warm white and cool white required for achieving the required illumination levels. The amplitude of the signals are 10v as this is required for powering the LED drivers. The duty cycle for both the cool white and warm white LEDs is 37.5% so as to provide roughly 75lux.

6.CONCLUSION:

The system which has been developed has been tested and has proven to be a dependable one. The wireless communication protocol which has been used and developed as well , has proven to be a dependable one.

The wireless communication protocol has proven to be an excellent protocol for the transfer of sensor data from the Obstacle Detection Units to the Master Controller Unit as well as transfer of pwm duty cycles from the Master Controller Unit to each of the Receiving Units.

The Master of the Master Controller has shown to provide WiFi connectivity to which mobile or laptop can connect. The interface consisting of the two sliders have shown their versatility in controlling the duty cycles of the pwm signals for dimming operations. The UART communication has proved to be a dependable hardware serial communication protocol in transferring the pwm values of the sliders to the Slave of the Master Controller.

The Slave of the Master Controller has achieved it's purpose of collecting the sensor data , processing them to produce the status signals as well as receiving the pwm signal data from the Master of the Master Controller via UART communication protocol .The Slave of the Master Controller Unit has also proved to be successful in transferring the pwm data received via UART communication protocol and the status signal generated after receiving the sensor data , to each of the Receiving Units.

The Controller circuit that has been designed has successfully amplified the pwm switching signals from 3.3v to 10v amplitude. The only setback which has been observed is that the duty cycles are not perfect .Thus some further modifications have to be done.

The system which has been developed has been tested analysis of it's performance has been done. The current drawn by the Controller circuit and the receiving end street luminaire to which the Controller circuit is connected has been recorded and it's variation with the variation in duty cycles for the respective drivers for cool white and warm white LEDs have been plotted which has shown a linear variation. The illuminance output of the developed system has also been recorded and it's variation with the variation in duty cycles has been recorded and plotted which has also shown a linear variation.

The output illuminance of the system have been recorded by the variation of duty cycles of both the warm white and the cool white LED drivers. For each duty cycle of the warm white , the cool white duty cycle has been varied progressively. The warm white duty cycle has also been varied progressively and for each value of it's duty cycle, the duty cycle of the cool white has also been varied progressively.

To test the developed system a set of four lux levels have been chosen and they have been achieved by setting suitable duty cycles for the cool white and warm white LED drivers. The achieved lux level values have a small percentage of error. The achieved lux level has been indicated by the lux meter with the necessary waveforms of the electrical signals input to the LED drivers have also been shown.

The only disadvantage of the UART communication protocol is that the the light could not be fully turned ON i.e. duty cycle = 100% could not be achieved. This would require recalibration of the slider scales.The current sensors used for measuring the current have shown to be unreliable.More upgraded and precise current sensors are required.

7.FUTURE SCOPE

The work done has a lot of scope to work on in the future. The integration of solar energy can be done which will enable the usage of renewable energy as well as the reduction in consumption of energy in the street lighting system.

The introduction of solar energy will enable to reduce electrical power consumption as well as enable the introduction of an auxiliary power source for the entire lighting system in case of emergency situation.

The concept of smart grid system can be integrated to this system which will monitor the energy consumption as well as monitor the charging of the battery by, done by the solar panels. The monitoring of the current state of battery charge as well as the current state of electrical power consumption by the LEDs in the street luminaire can be observed as well as stored.

Concepts of IoT based smart street light monitoring system to evaluate the performance of the system can be integrated for energy evaluation.

In the schematics of the receiver unit, a linear power supply system has been developed, which increases the size of the power transformer.

In order to reduce this size and the bulkiness of the transformer, a SMPS using the flyback topology can be used for providing power, thereby reducing the size as well as promoting compactness to the present system.

The system architecture developed above can be extended to indoor lighting, facade lighting, architectural lighting, landscape lighting, etc. The concept of Visual Light Communication with various types of modulation techniques can also be integrated.

8.REFERENCES:

[1] Rifki Muhendra, Yudha Hamdi Arzi, “Development of street light controller using wifi mesh network”.

[2]https://www.electricaltechnology.org/2022/06/led-light-emitting-diode.html#google_vignette

[3]. Guirguis Z Abdelmessih, J. Marcos Alonso, Marina S. Perdigão , “Hybrid Series-Parallel PWM dimming technique for Integrated-Converter-Based HPF LED Drivers”.

[4]https://www.smarthomepoint.com/zigbee-zwave-wifi-bluetooth-comparison/#:~:text=WiFi%20and%20Zigbee%20are%20currently%20the%20most%20popular,protocols%2C%20before%20examining%20each%20one%20in%20more%20detail.

[5]. Fred Greenfeld, Kin Shum, “Control of LED Luminaires - An Overview of Competing Wired and Wireless Standards”.

[6]. Yi Shuangshuang, Li Jianqi, Ou Xiaoqing, Zhou Qingshan, Li Jianying , “Design of intelligent lighting controller based on Bluetooth ’’.

[7]. Yu-Shan Cheng1 , Jing-Hsiao Chen1 , Yi-Hua Liu1 and Shun-Chung Wang, “ Development of Wireless RGB LED Dimming Control Technology Using Smart phone’’.

[8]. Kuthsav Thattai , Kuthsav Thattai , Shiwangi Chhawchharia , Prof.R.Marimuthu, “ZigBee and ATmega32 Based Wireless Digital Control and Monitoring System For LED Lighting’’.

[9]. Báez C. Rolando, Berrelleza O. Julio, Pech A. Esaías, “Controlling Digital Dimmer Through Mobile Phone’’

[10]. Kok-Hua Teng, Zi-Yi Lam, Sew-Kin Wong , “Dimmable WiFi-Connected LED Driver with Android Based Remote Control’’

[11] Tobiloba Somefun, Claudius Awosope, Ademola Abdulkareem, Daniel Adeleye, “DEPLOYMENT OF SMART STREET LIGHTING SYSTEM USING SENSORS ”

[12] Bhaavan Sri Sailesh A, Sudha Madhavi A, Venkata Pavan G, Sravanthi I, Karthik Sai Kiran B, Venkateswara Rao Ch, “ Arduino based Smart Street Light System”.

9.APPENDEX :

9.1 Specifications of Hardwares used

MICROCONTROLLER

- Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, running at 160 or 240 MHz
- Memory: 520 KB SRAM
- Wi-Fi: 802.11 b/g/n
- Bluetooth: v4.2 BR/EDR and BLE
- 12-bit ADC channels
- 8-bit DACs
- touch sensors (capacitive sensing GPIOs)
- SPI
- I^2S interfaces
- I^2C interfaces
- UART
- SD/SDIO/CE-ATA/MMC/eMMC host controller
- SDIO/SPI slave controller
- CAN bus 2.0
- Infrared remote controller (TX/RX, up to 8 channels)
- Motor PWM
- LED PWM (up to 16 channels)
- Hall effect sensor
- Ultra-low-power analog pre-amplifier

ULTRASONIC SENSOR

- Input Voltage: 5V
- Current Draw: 20mA (Max)
- Digital Output: 5V
- Digital Output: 0V (Low)
- Working Temperature: -15°C to 70°C
- Ultrasonic Frequency: 40kHz
- Range: 2cm - 400cm

LED DRIVER

- Additive 10V pwm signal 100Hz ~ 3KHz for dimming operation
- 700mA(default) Voltage Range
- Rated power = 60.3 W
- Input voltage range : 180-295 V (AC)

  254-417 V (DC)

- Input frequency range : (47-63) Hz
- DC output range : 2-86V
- Open circuit voltage (max) :102 V
- Protection :

a. Short circuit : Constant current limiting, recovers automatically after fault condition is removed.
b. Over voltage : 105 ~ 125V, Shutdown o/p voltage, re-power on to recover
c. Over temperature :Shutdown o/p voltage,re-power on to recover

DIODES

- Maximum DC Blocking Voltage : 1000V
- Maximum RMS Voltage : 700V
- Maximum Average Rectified Current : 1A
- Maximum Forward voltage drop : 1.1V at 1A@25°C.
- Maximum DC Reverse current : 5µA at 25°C.

OPTOCOUPLER

- Input Forward Voltage : (1.2-1.5)V
- Reverse Leakage current: 10 µA
- Input Capacitance : 30Pf
- Collector to Emitter Current Transfer ratio: 100%
- Turn-on Time: (10-12)µs
- Turn-off Time:(9-12) µs
- Input-Output Isolation Voltage : 5000V
- Isolation Capacitance : 0.2 Pf.

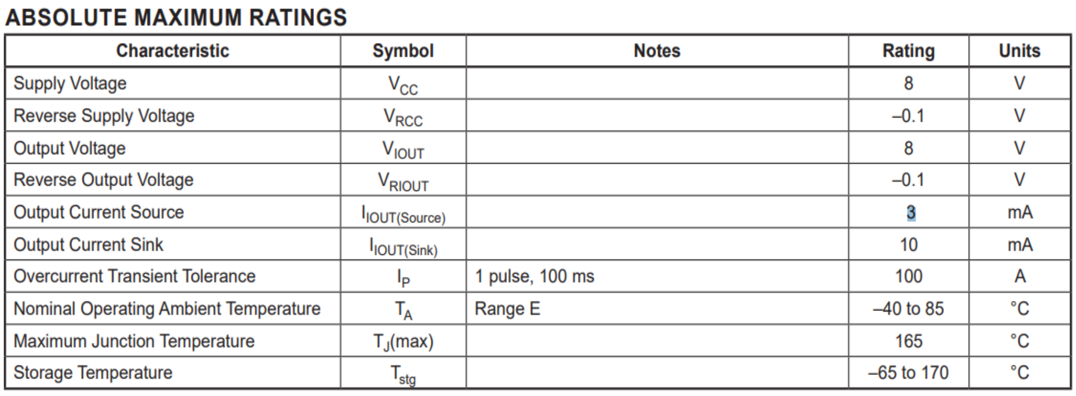
CURRENT SENSOR

a) Current transformer sensor

- Onboard precision micro current transformer.
- Onboard sampling resistor .
- Module can be measured within 5A alternating current, the corresponding analog output 5 A/ 5 mA
- PCB board size: 18.3 (mm) x17 (mm).
- Rated Primary Current at 50/60 Hz: 5 A
- Maximum Primary Current at 50/60 Hz: 20 A
- Turns Ratio: Np:Ns = 1:2,500
- Current Ratio: 5A:2mA
- Winding D.C. Resistance at 20 °C: 155 Ω
- Accuracy •@RL≤10 Ω: 2%

b)Hall Effect based linear current sensor

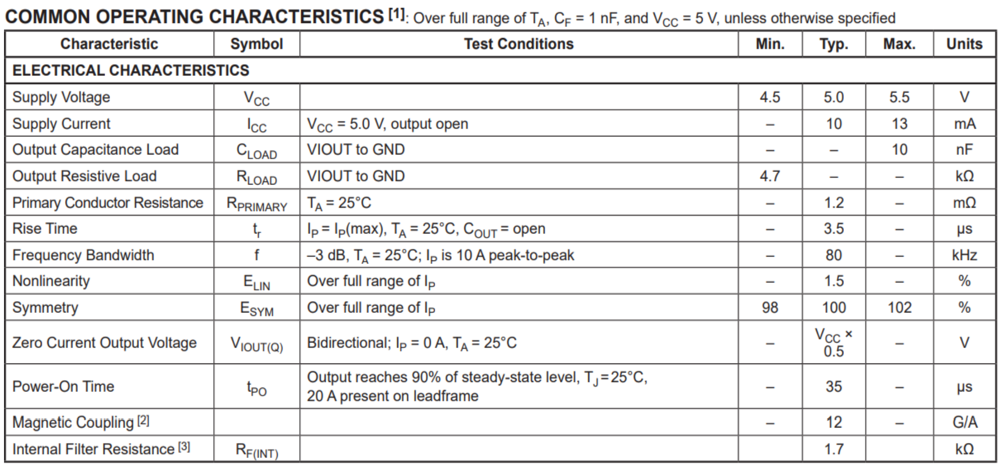
Table 9.1 Table showing the absolute maximum ratings of Hall Effect current sensor.



ABSOLUTE MAXIMUM RATINGS

Characteristic	Symbol	Notes	Rating	Units
Supply Voltage	V_{CC}		8	V
Reverse Supply Voltage	V_{RCC}		–0.1	V
Output Voltage	V_{IOUT}		8	V
Reverse Output Voltage	V_{RIOUT}		–0.1	V
Output Current Source	$I_{IOUT(Source)}$		3	mA
Output Current Sink	$I_{IOUT(Sink)}$		10	mA
Overcurrent Transient Tolerance	I_P	1 pulse, 100 ms	100	A
Nominal Operating Ambient Temperature	T_A	Range E	–40 to 85	°C
Maximum Junction Temperature	T_J(max)		165	°C
Storage Temperature	T_{stg}		–65 to 170	°C

Table 9.2 Table showing the operating characteristics of the Hall Effect current sensor.



COMMON OPERATING CHARACTERISTICS [1]: Over full range of T_A, C_F = 1 nF, and V_{CC} = 5 V, unless otherwise specified

Characteristic	Symbol	Test Conditions	Min.	Typ.	Max.	Units
ELECTRICAL CHARACTERISTICS						
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Supply Current	I_{CC}	V_{CC} = 5.0 V, output open	–	10	13	mA
Output Capacitance Load	C_{LOAD}	VIOUT to GND	–	–	10	nF
Output Resistive Load	R_{LOAD}	VIOUT to GND	4.7	–	–	kΩ
Primary Conductor Resistance	$R_{PRIMARY}$	T_A = 25°C	–	1.2	–	mΩ
Rise Time	t_r	I_P = I_P(max), T_A = 25°C, C_{OUT} = open	–	3.5	–	μs
Frequency Bandwidth	f	–3 dB, T_A = 25°C; I_P is 10 A peak-to-peak	–	80	–	kHz
Nonlinearity	E_{LIN}	Over full range of I_P	–	1.5	–	%
Symmetry	E_{SYM}	Over full range of I_P	98	100	102	%
Zero Current Output Voltage	$V_{IOUT(Q)}$	Bidirectional; I_P = 0 A, T_A = 25°C	–	V_{CC} × 0.5	–	V
Power-On Time	t_{PO}	Output reaches 90% of steady-state level, T_J = 25°C, 20 A present on leadframe	–	35	–	μs
Magnetic Coupling [2]			–	12	–	G/A
Internal Filter Resistance [3]	$R_{F(INT)}$			1.7		kΩ

VOLTAGE REGULATORS

1. LM-7812

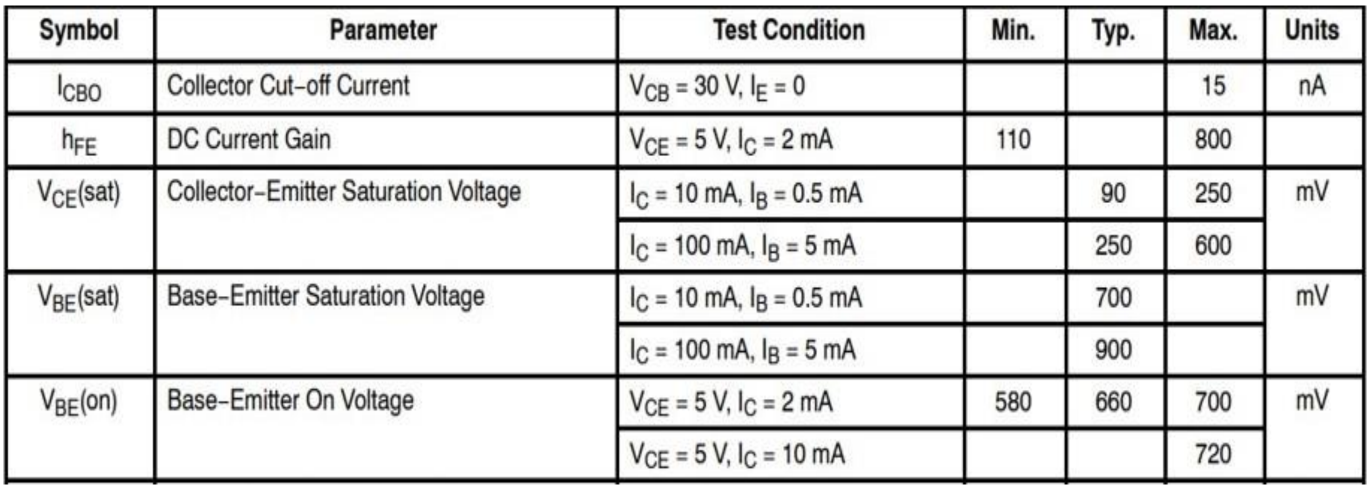
- Input Voltage Range: 14V to 35V
- Output Voltage Range: 11.75V - 12.25V DC
- Output Current (Typical): 1A
- Peak current: 2.2A
- Dropout voltage: 2V
- Available in TO-252, TO-220 & TO-263 Package
- The Output current is 1.5 Ampere
- 12V fixed and acurate output voltage

2. LM-7805

- Max output current : 1A
- Input voltage range : 7V-35V
- Available in TO-252, TO-220 & TO-263 Package
- 5V fixed and acurate output voltage
- Dropout voltage: 2V

TRANSISTOR

Table 9.3 Table showing the operating electrical parameters ratings of the npn BJT transistor used



Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Units
I_{CBO}	Collector Cut-off Current	V_{CB} = 30 V, I_E = 0			15	nA
h_{FE}	DC Current Gain	V_{CE} = 5 V, I_C = 2 mA	110		800	
V_{CE}(sat)	Collector-Emitter Saturation Voltage	I_C = 10 mA, I_B = 0.5 mA		90	250	mV
		I_C = 100 mA, I_B = 5 mA		250	600	
V_{BE}(sat)	Base-Emitter Saturation Voltage	I_C = 10 mA, I_B = 0.5 mA		700		mV
		I_C = 100 mA, I_B = 5 mA		900		
V_{BE}(on)	Base-Emitter On Voltage	V_{CE} = 5 V, I_C = 2 mA	580	660	700	mV
		V_{CE} = 5 V, I_C = 10 mA			720	

9.2 Specification of the Software used

The Arduino integrated development environment (IDE) is a cross-platform application (for Microsoft Windows, macOS, and Linux) that is written in the Java programming language. It originated from the IDE for the languages *Processing* and *Wiring*. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple *one-click* mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.