INTERNSHIP REPORT

MOTION SENSING BASED STREET LIGHT

Submitted by

Prarambha Barman (Roll No-2400441231479
Reg.no-2410041231483)

Sayan Debnath (Roll No-2400441231498
Reg.no-2410041231493)

Sudip Bhowmik (Roll No-2400441231471
Reg.no-2410041231504)

Anurag Biswas (Roll No-2400441231467
Reg.no-24100412311459)

Under the guidance of

Project supervisor- Dr. Sourav Chakraborty

Department of Electronics and Communication Engineering

Cooch Behar Government Engineering College

Cooch Behar, West Bengal

2025

Certificate of Recommendation

It is hereby recommended to consider the project report entitled "Motion sensing-based street light" submitted by Prarambha Barman, Sayan Debnath, Anurag Biswas, Sudip Bhowmik for partial fulfilment of the requirements for the award of the INTERNSHIP CERTIFICATE from the Electronics and Communication Engineering dept. from Cooch Behar Govt. Engineering College affiliated to Maulana Abul Kalam Azad University of Technology (formerly known as West Bengal University of Technology).

Project Supervisor

Contents

CHPATER 1:

INTROCUTION

Street lights are essential part of daily transportation life. While going from one place to another place we need a light source to see and that is where street come in play role to makes see at night on a road.

The street light currently we are using it remains a constant brightness for so long and it makes less energy efficient and power consumption more, with current global affairs energy conservation is a big part, starting with smart home automation, like solar based humidity, soil moisture, smart led street light, smoke and obstacle detecting vehicle etc. is a step to smart innovation and eco-friendly choice.

OBJECTIVE

Our project "Motion Sensing Based Street Light" aims to achieve a smart street light, where the led is at low brightness and when a motion is checked, the LED is increase to High brightness which makes the road more visible for coming vehicle. In this system if it is day time the LED is turned off making it energy efficient.

CHAPTER 2:

OUR WORK

HARDWARE REQUIREMENTS:

ESP32

PASSIVE INFRARED SENSOR

LIGHT DEPENDENT RESISTANCE SENSOR

LED DRIVER (FOR CONSTANT CURRENT WITH PWM PIN)

3 WATT LEDS

12VOLTS AND 1 AMPERE ADAPTER FOR EXTERNAL POWER SOURCE TO LED DRIVER

BREADBOARD, RESISTOR, SMALL LED, JUMPER WIRE, LIVE WIRE.

SOFTWARE REQUIREMENT:

ARDUINO IDE (FOR COMPILING THE LOGIC IN C)

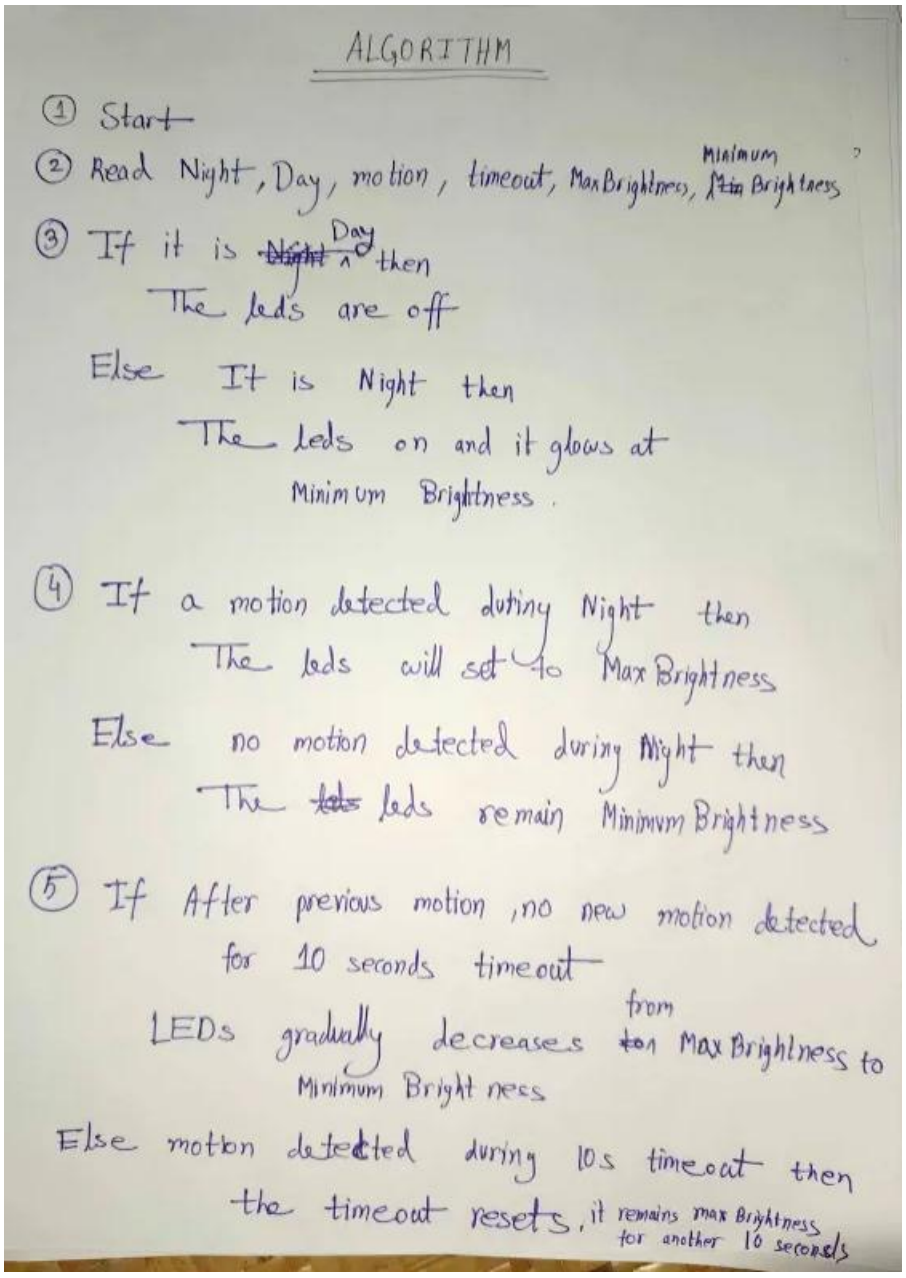
BLYNK IoT

ONE DEVICE FOR WIFI CLIENT.

METHODOLOGY:

We have made a 3 LED system, where LDR which checks for Day/Night, LDR's Pin VCC and GND connected to Master ESP32's 3v3(3 volts) pin and ground (GND) pin, the DO (Digital Output) pin connected to D36(VP) pin of ESP32. The DO pin gives output 1 or 0 to ESP32, the 1 for night and 0 for day and ESP32 takes it as Input. Then we have PIR sensor which checks for motion in our system, both master and slave ESP32 has one PIR sensor to detect any motion. There is LED DRIVER which helpful when we deal with High power LEDS, since we are using 3 Watt LEDS which are in series we have to supply a constat current of 0.7 Ampere, this where LED Driver comes handy. IN+ and IN- of LED DRIVER connected to external source (Adapter) to give power supply to, LED+ pin of LED DRIVER connected to anode of first led and LED- pin is connected to cathode of last led. There is a PWM pin in LED DRIVER, which helps in Diming a LED.

ALGORITHM:



ALGORITHM

① Start

② Read Night, Day, motion, timeout, MaxBrightness, Minimum Brightness

③ If it is ~~Night~~ Day then
The leds are off

Else It is Night then
The leds on and it glows at
Minimum Brightness.

④ If a motion detected during Night then
The leds will set to Max Brightness

Else no motion detected during Night then
The ~~leds~~ leds remain Minimum Brightness

⑤ If After previous motion, no new motion detected
for 10 seconds timeout
LEDs gradually decreases from ~~to~~ Max Brightness to
Minimum Brightness

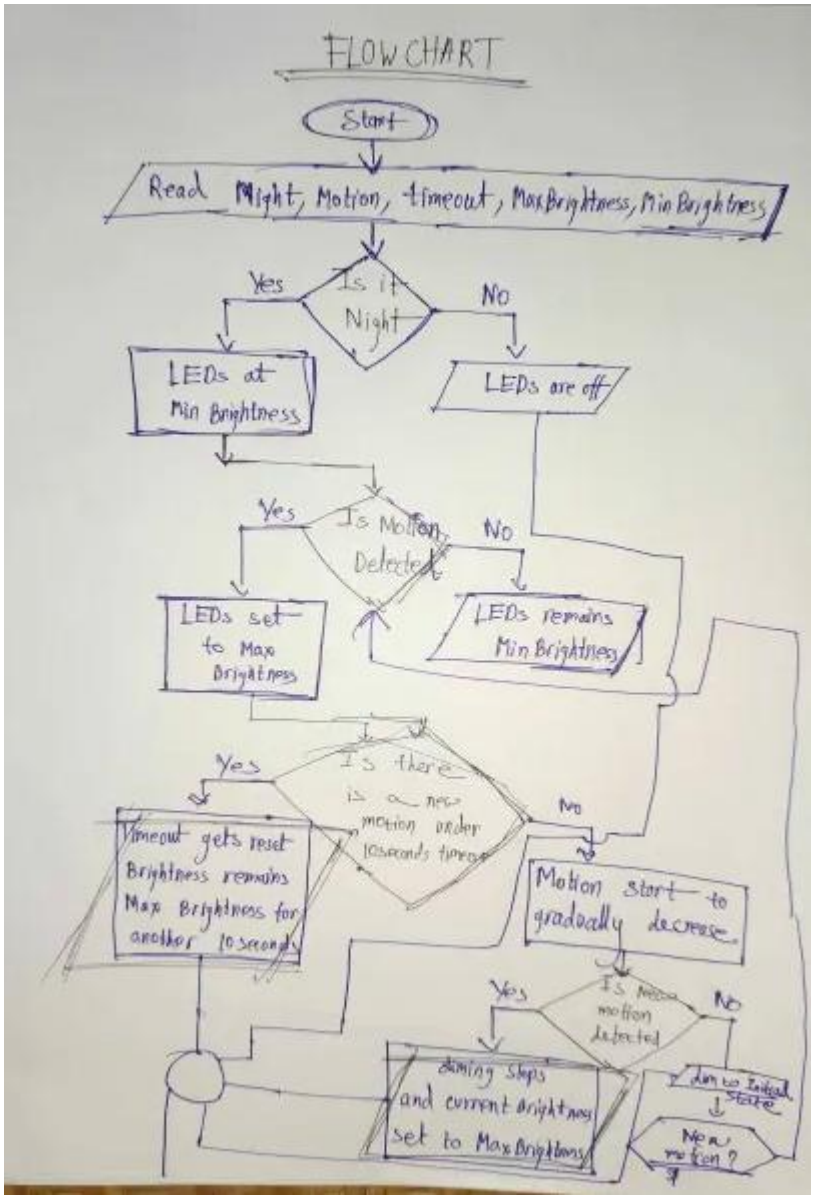
Else motion detected during 10s timeout then
the timeout resets, it remains max Brightness
for another 10 seconds

⑥ If motion detected during gradual dimming then
dimming cancel out and Brightness to ~~Max for~~
Max Brightness for LEDs.

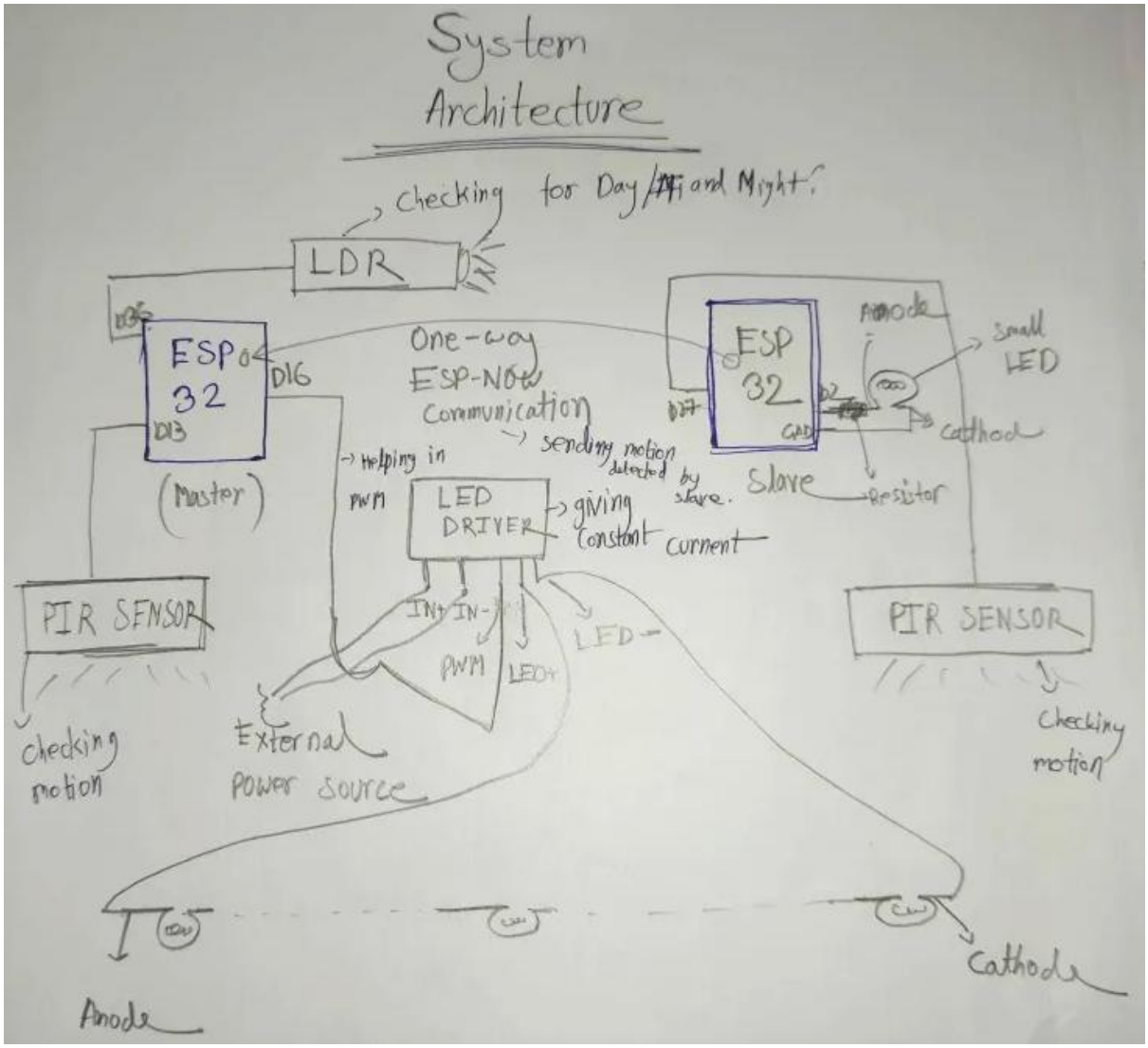
Else
It gradually decreases and set to
minimum Brightness.

⑦ End

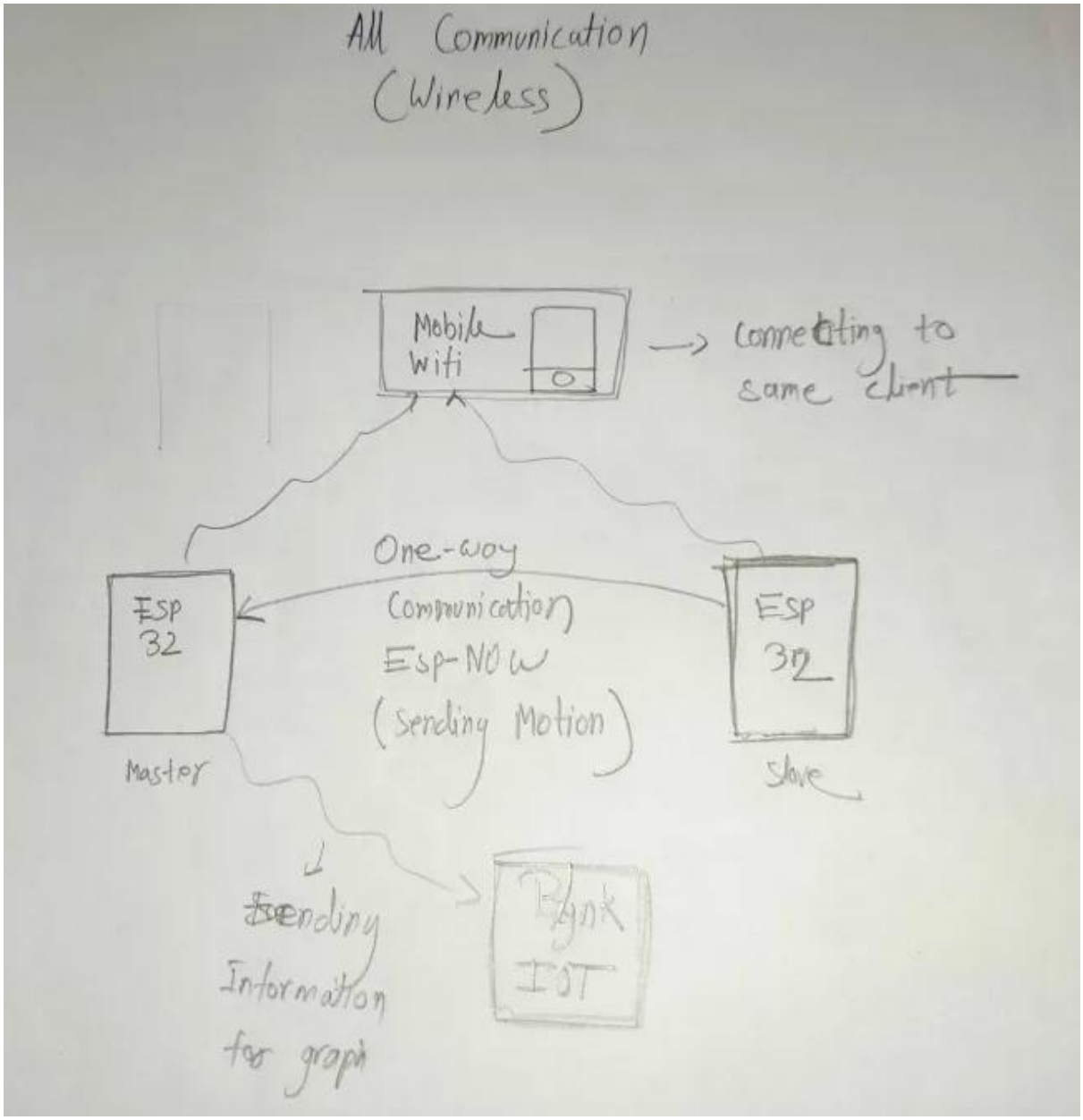
FLOWCHART:


FLOW CHART
Start
Read Night, Motion, timeout, Max Brightness, Min Brightness
Is it Night
Yes
No
LEDs at Min Brightness
LEDs are off
Is Motion Detected
Yes
No
LEDs set to Max Brightness
LEDs remains Min Brightness
Is there is a new motion under 10seconds time
Yes
No
Timeout gets reset Brightness remains Max Brightness for another 10seconds
Motion start to gradually decrease
Yes
No
Stop

SYSTEM ARCHITECTURE:



ALL WIRELESS COMMUNICATION:



ESP-NOW IMPORTANCE IN OUR PROJECT:

In our project there are 2 ESP32, one is master and other is slave. Master mainly taking PIR, LDR outputs with that giving input to LEDS and controlling it, in this way just one way motion completed. We have brought slave PIR sensor so that the system can detect motion if vehicle comes from either side of the road.
ESP-NOW is a Wireless communication which uses peer to peer connection for direction connection.
Here, Master ESP32 takes data as inputs from Slave ESP32 through ESP-NOW, and Slave ESP32 generates the output.

When a motion sensed by the Slave PIR, Slave ESP32 stores a string "MOTION" in the structure variable which is sent to Master ESP32. A function data receive called when the string data arrived in Master ESP32 and it checks if that string is correct. If it is correct the LEDS Will glow to maximum according to the logic, if not then either data long or short or communication failed to due to Wi-Fi channel error.

//MASTER CODE:

```
//Important note: We used esp32 by espressif systems library version 2.0.17

#define BLYNK_TEMPLATE_ID "TMPL3v-7pd2Yp"
#define BLYNK_TEMPLATE_NAME "Msbsl"
#define BLYNK_AUTH_TOKEN "Y3HBA3h0TsKzUVVoVycvxliDmvvoQbwY"

#include <WiFi.h>
#include <esp_now.h>
#include <BlynkSimpleEsp32.h>

#define led     16   // Main LED pin
#define pirpin  13   // PIR sensor pin (master)
#define ldrpin  36   // LDR sensor pin (digital output)

const char* WIFI_NAME = "realme 14 Pro lite 5G (128)";
const char* WIFI_PASSWORD = "z63ajyfu";

uint8_t SlaveAddress[6] = {0x14, 0x33, 0x5C, 0x04, 0x4A, 0xFC}; //Store MAC Address(helps to knows which esp to send/receive data to)
```

```
typedef struct struct_message { //Data container
  char msbsl[16]; //character array here it receives and strores a string
} struct_message;

struct_message mydata; //this holds the string for further

//Varying Voltage as pulse(On and Off)
const int pwmch = 0, pwmfreq = 2000, pwmreso = 8; //16 channel thatt creates pwm,pwmfreq=how often it goes hign and low,here 2000hz per seconds.

int MaxBrightness = 179;    // Full brightness level
int LowBrightness = 26;     // Dim brightness level
const int dimstep = 26;     // Brightness decrement per step

unsigned long previousMillis = 0; // Last motion or brightness change time
const long timeout = 10000;       // After 10s without motion, start dimming
const long interval = 1000;       // Dim every 1 second

bool pirState = LOW;          // Current PIR state
```

```
bool lastPIR = LOW;          // Last PIR state for edge detection
bool lastNight = false;      // Track last night/day state
bool ldrnow;                 // Store LDR reading
int brightness = LowBrightness; // Current LED brightness

bool motionFromSlave = false;
unsigned long slaveMotionTime = 0;
const unsigned long slaveMotionTimeout = 5000;

int masterMotionCount = 0;
int slaveMotionCount = 0;
// Callback when data is receive

void onDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len) {
  memcpy(&mydata, incomingData, sizeof(mydata));
  if (strcmp(mydata.msbsl, "MOTION") == 0) {
    motionFromSlave = true;
    slaveMotionTime = millis();
    slaveMotionCount++;
    Blynk.virtualWrite(V1, slaveMotionCount); // Send to Blynk graph
  }
```

```
}

void setup() {
  Serial.begin(115200);

  // Pin setup
  pinMode(pirpin, INPUT);
  pinMode(ldrpin, INPUT);
  pinMode(led, OUTPUT);

  // PWM setup
  ledcSetup(pwmch, pwmfreq, pwmreso);
  ledcAttachPin(led, pwmch);

  // Connect to WiFi
  WiFi.begin(WIFI_NAME, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);

  }

  // Start Blynk
```

```
 Blynk.begin(BLYNK_AUTH_TOKEN, WIFI_NAME,
WIFI_PASSWORD);

 // Set WiFi to station mode for ESP-NOW
 WiFi.mode(WIFI_STA);

 // Init ESP-NOW
 if (esp_now_init() != ESP_OK) {

   ESP.restart();
 }

 // Register peer (Slave)
 esp_now_peer_info_t peerInfo = {};
 memcpy(peerInfo.peer_addr, SlaveAddress, 6);
 peerInfo.channel = WiFi.channel(); // Same channel as
WiFi
 peerInfo.encrypt = false;
 if (esp_now_add_peer(&peerInfo) != ESP_OK) {
 }

 // Register receive callback
 esp_now_register_recv_cb(onDataRecv);
```

```
  ledcWrite(pwmch, brightness);//set the duty cycle of the
PWM signal.
  previousMillis = millis();
}

void loop() {
  Blynk.run(); // Run Blynk tasks

  unsigned long currentMillis = millis();
  bool pirValue = digitalRead(pirpin);
  bool ldrValue = digitalRead(ldrpin);
  bool night = (ldrValue == 1);



  // PIR change detection
  if (pirValue != lastPIR) {

    lastPIR = pirValue;
  }

  // Nightfall detection
```

```
  if (night && !lastNight) {

    brightness = LowBrightness;
    previousMillis = currentMillis;
  }
  lastNight = night;

  // Reset slave motion after timeout
  if (motionFromSlave && (currentMillis - slaveMotionTime
> slaveMotionTimeout)) {
    motionFromSlave = false;
  }

  // Motion from master or slave
  if ((pirValue || motionFromSlave) && night) {
    ledcWrite(pwmch, MaxBrightness);
    previousMillis = currentMillis;
    motionFromSlave = false; // Reset after use
    if (pirValue) {
      masterMotionCount++;
      Blynk.virtualWrite(V2, masterMotionCount); // Send to
Blynk graph
    }
```

```
  }

  // No motion → set to low brightness after 1s
  if (!pirValue && (currentMillis - previousMillis >= interval)) {
    brightness = LowBrightness;
  }

  // Dimming over time if no motion for 'timeout'
  if ((currentMillis - previousMillis > timeout) && night) {
    if (brightness > LowBrightness) {
      brightness -= dimstep;
      if (brightness < LowBrightness) brightness = LowBrightness;

      delay(1000);
    }
  }

  // Daytime → LED OFF
  if (!night) {
    /*if (brightness != 0) {
```

```
   }*/
   brightness = 0;
  }

  // Apply brightness
  ledcWrite(pwmch, brightness);
  delay(200);
}


//SLAVE CODE:
#include <WiFi.h>
#include <esp_now.h>

#define SMALL_LED 2       // Small indicator LED (on/off)
#define PIR_PIN 27        // PIR sensor pin

const char* WIFI_NAME = "realme 14 Pro lite 5G (128)";
const char* WIFI_PASSWORD = "z63ajyfu";

// Master's MAC address
uint8_t masterAddress[] = { 0x00, 0x4B, 0x12, 0x33, 0x7E,
0x00 };
```

```
// Structure to receive ESP-NOW data
typedef struct struct_message {
  char msbsl[16];  // expects "MOTION"
} struct_message;

struct_message myData;

bool lastPIR = false;
unsigned long ledStart = 0;
bool ledActive = false;

void setup() {
  Serial.begin(115200);
  pinMode(PIR_PIN, INPUT);
  pinMode(SMALL_LED, OUTPUT);

  WiFi.begin(WIFI_NAME, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\n[Slave] WiFi connected");
```

```
  Serial.println("[Slave] Channel: " +
String(WiFi.channel()));

  WiFi.mode(WIFI_STA);
  if (esp_now_init() != ESP_OK) {
    Serial.println("ESP-NOW init failed");
    ESP.restart();
  }
   esp_now_peer_info_t peerInfo = {};
  memcpy(peerInfo.peer_addr, masterAddress, 6);
  peerInfo.channel = WiFi.channel();  // dynamic and
correct
  peerInfo.encrypt = false;

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

  Serial.println("[Slave] ESP-NOW setup complete");
}

void loop() {
  bool pir = digitalRead(PIR_PIN);
```

```
  // Only trigger on rising edge (LOW to HIGH)
  if (pir && !lastPIR) {
    Serial.println("[Slave] PIR motion detected");

    // Turn on status LED for feedback
    digitalWrite(SMALL_LED, HIGH);
    ledStart = millis();
    ledActive = true;

    // Send motion message
    strcpy(myData.msbsl, "MOTION");
    esp_err_t result = esp_now_send(masterAddress, (uint8_t
*)&myData, sizeof(myData));
    if (result == ESP_OK) {
      Serial.println("[Slave] Motion data sent to master");
    } else {
      Serial.println("[Slave] Failed to send motion data");
    }
  }

  lastPIR = pir;
```

```
  // Turn off LED after 5 seconds
  if (ledActive && (millis() - ledStart > 5000)) {
    digitalWrite(SMALL_LED, LOW);
    ledActive = false;
  }

  delay(20); // Light loop
}
```

IMPORTANT CODE:

```
#include <WiFi.h>

void setup() {
  Serial.begin(115200);
  delay(1000);

  // Get the MAC address
  String macAddress = WiFi.macAddress();

  Serial.println("ESP32 MAC Address:");
  Serial.println(macAddress);
}

void loop() {
```

```
  // Nothing here
}
```

CHAPTER 3:

CONCLUSION:

Building this was fun and challenging but we learnt lot from it. Using this Motion Sensing Based Street Light, we can be close to making of smart city. Development in urban areas of India making it easier for them during night, in the big roads using this would help in energy efficiency. In future we are motivated to work more projects related to ESP32 and Arduino.

Bibliography:

Researchgate

Arduino Forum

Github

Youtube

Different Articles on ESP32