LAPORAN AKHIR
KONTROL LAMPU OTOMATIS DENGAN SENSOR *PASSIVE INFRA RED* (PIR) DAN BLYNK

Dibuat Sebagai Penyelesaian Tugas Magang MBKM PT. Krakatau Tirta Industri

YUSUF BUDI KUSUMA – 3332210005

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS SULTAN AGENG TIRTAYASA
CILEGON, BANTEN
2024

DAFTAR ISI

HALAMAN JUDUL 1
BAB I PENDAHULUAN 3
1.1. Latar Belakang 3
1.2. Rumusan Masalah 3
1.3. Tujuan 3
BAB II TINJAUAN PUSTAKA 5
2.1. Relay 5
2.2. Saklar 7
2.3. Lampu Bohlam 8
2.4. ESP-32 8
2.5. PIR Sensor 11
2.6. Kabel 12
BAB III METODOLOGI PEMBUATAN 13
3.1. Diagram Alir 13
3.2. Alat dan Bahan 13
3.3. Rancangan Anggaran Biaya 14
BAB IV HASIL DAN PEMBAHASAN 16
4.1. Rangkaian Skematik 16
4.2. Source Code 19
4.3. Tampilan Antarmuka Blynk 25
4.4. Hasil Simulasi 26
BAB V PENUTUP 27
5.1. Kesimpulan 27

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dengan perkembangan teknologi *Internet of Things* (IoT) yang semakin pesat, banyak solusi cerdas telah dikembangkan untuk meningkatkan kenyamanan dan efisiensi dalam kehidupan sehari-hari. Salah satu aplikasi yang populer dalam ranah IoT adalah kontrol lampu otomatis. Dalam konteks ini, penggunaan sensor PIR (*Passive Infrared*) menjadi sangat relevan karena kemampuannya untuk mendeteksi gerakan dan perubahan suhu, khususnya dari tubuh manusia. Sensor PIR secara efektif memungkinkan sistem untuk merespons secara otomatis ketika ada kehadiran orang di sekitarnya. Dengan memanfaatkan informasi yang diberikan oleh sensor ini, sistem dapat mengatur pencahayaan secara otomatis, mengurangi konsumsi energi yang tidak perlu dan memberikan kenyamanan bagi pengguna dengan menyediakan pencahayaan yang sesuai dengan kebutuhan.

Salah satu platform yang populer untuk mengembangkan proyek IoT adalah Blynk. Blynk menyediakan infrastruktur yang mudah digunakan untuk menghubungkan perangkat fisik dengan aplikasi berbasis smartphone atau web. Dengan menggunakan Blynk, pengguna dapat dengan mudah mengontrol dan memantau perangkat IoT mereka dari jarak jauh melalui internet. Dengan menggabungkan sensor PIR dan platform Blynk, proyek kontrol lampu otomatis dapat diimplementasikan dengan mudah. Sensor PIR digunakan untuk mendeteksi gerakan manusia, sementara platform Blynk digunakan untuk mengatur dan mengontrol lampu secara otomatis dari jarak jauh. Kombinasi ini tidak hanya meningkatkan kenyamanan pengguna, tetapi juga membantu mengurangi konsumsi energi yang tidak perlu, menjadikannya solusi yang ramah lingkungan dan efisien.

1.2. Rumusan Masalah

Berikut merupakan rumusan masalah dari pembuatan sistem kontrol lampu otomatis dengan sensor PIR dan kontrol IoT dengan Blynk:

1. Bagaimana membuat sistem kontrol lampu otomatis untuk suatu ruangan?
2. Bagaimana mengontrol lampu dengan berbasis *Internet of Things* (IoT)?

1.3. Tujuan

Berikut merupakan tujuan dari pembuatan sistem kontrol lampu otomatis dengan sensor PIR dan kontrol IoT dengan Blynk:

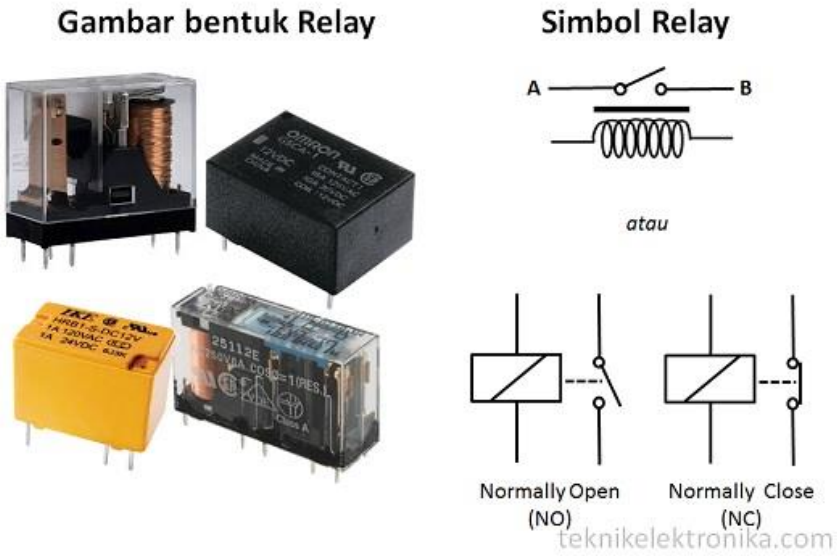
1. Membuat simulasi sistem kontrol lampu otomatis pada suatu ruangan tertentu.

2. Melakukan pembuatan sistem kontrol berbasis IoT dari simulasi yang telah dibuat.

BAB II
TINJAUAN PUSTAKA

2.1. Relay

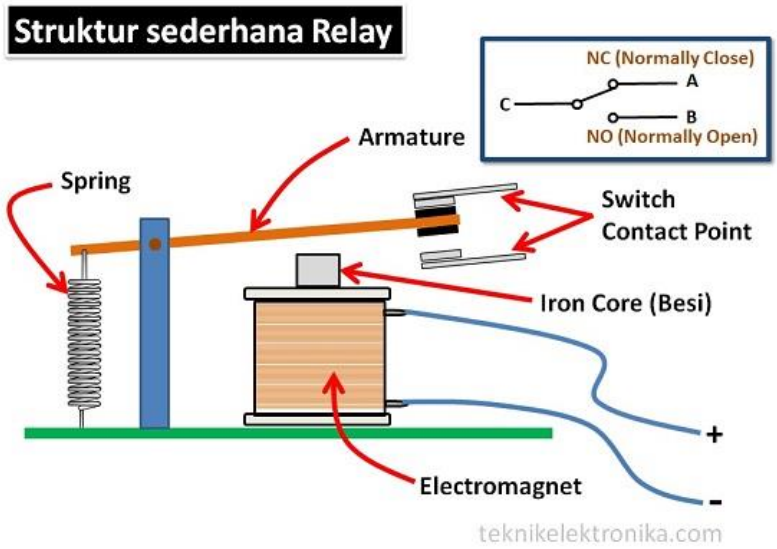
Relay adalah Saklar (*Switch*) yang dioperasikan secara listrik dan merupakan komponen *Electromechanical* (Elektromekanikal) yang terdiri dari 2 bagian utama yakni Elektromagnet (*Coil*) dan Mekanikal (*seperangkat Kontak Saklar/Switch*). Relay bekerja berdasarkan prinsip Elektromagnetik untuk menggerakkan Kontak Saklar sehingga dengan arus listrik yang kecil (low power) dapat menghantarkan listrik yang bertegangan lebih tinggi. Sebagai contoh, dengan Relay yang menggunakan Elektromagnet 5V dan 50 mA mampu menggerakan Armature Relay (yang berfungsi sebagai saklarnya) untuk menghantarkan listrik 220V 2A.



Pada dasarnya, Relay terdiri dari 4 komponen dasar yaitu:

1. Electromagnet (Coil)
2. Armature
3. Switch Contact Point (Saklar)
4. Spring

Berikut ini merupakan gambar dari bagian-bagian Relay :



Kontak Poin (*Contact Point*) Relay terdiri dari 2 jenis yaitu:

- *Normally Close* (NC) yaitu kondisi awal sebelum diaktifkan akan selalu berada di posisi *CLOSE* (tertutup)
- *Normally Open* (NO) yaitu kondisi awal sebelum diaktifkan akan selalu berada di posisi *OPEN* (terbuka)

Berdasarkan gambar diatas, sebuah Besi (*Iron Core*) yang dililit oleh sebuah kumparan *Coil* yang berfungsi untuk mengendalikan besi tersebut. Apabila Kumparan *Coil* diberikan arus listrik, maka akan timbul gaya Elektromagnet yang kemudian menarik Armature untuk berpindah dari Posisi sebelumnya (NC) ke posisi baru (NO) sehingga menjadi Saklar yang dapat menghantarkan arus listrik di posisi barunya (NO). Posisi dimana Armature tersebut berada sebelumnya (NC) akan menjadi OPEN atau tidak terhubung. Pada saat tidak dialiri arus listrik, Armature akan kembali lagi ke posisi Awal (NC). Coil yang digunakan oleh Relay untuk menarik Contact Poin ke Posisi Close pada umumnya hanya membutuhkan arus listrik yang relatif kecil.

Karena Relay merupakan salah satu jenis dari Saklar, maka istilah Pole dan Throw yang dipakai dalam Saklar juga berlaku pada Relay. Berikut ini adalah penjelasan singkat mengenai Istilah Pole and Throw :

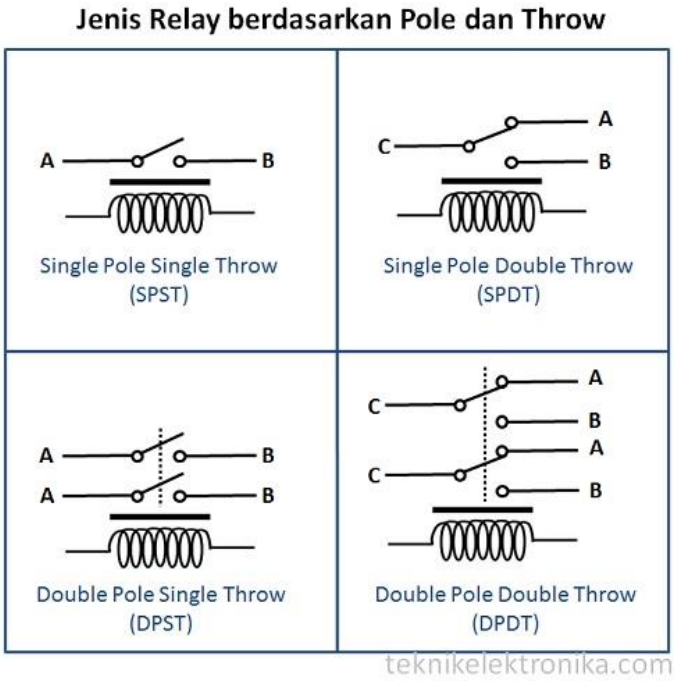
Pole : Banyaknya Kontak (*Contact*) yang dimiliki oleh sebuah relay

Throw : Banyaknya kondisi yang dimiliki oleh sebuah Kontak (*Contact*)

Berdasarkan penggolongan jumlah Pole dan Throw-nya sebuah relay, maka relay dapat digolongkan menjadi :

- Single Pole Single Throw (SPST): Relay golongan ini memiliki 4 Terminal, 2 Terminal untuk Saklar dan 2 Terminalnya lagi untuk Coil.
- Single Pole Double Throw (SPDT): Relay golongan ini memiliki 5 Terminal, 3 Terminal untuk Saklar dan 2 Terminalnya lagi untuk Coil.
- Double Pole Single Throw (DPST): Relay golongan ini memiliki 6 Terminal, diantaranya 4 Terminal yang terdiri dari 2 Pasang Terminal Saklar sedangkan 2 Terminal lainnya untuk Coil. Relay DPST dapat dijadikan 2 Saklar yang dikendalikan oleh 1 Coil.
- Double Pole Double Throw (DPDT): Relay golongan ini memiliki Terminal sebanyak 8 Terminal, diantaranya 6 Terminal yang merupakan 2 pasang Relay SPDT yang dikendalikan oleh 1 (single) Coil. Sedangkan 2 Terminal lainnya untuk Coil.

Selain Golongan Relay diatas, terdapat juga Relay-relay yang Pole dan Throw-nya melebihi dari 2 (dua). Misalnya 3PDT (*Triple Pole Double Throw*) ataupun 4PDT (*Four Pole Double Throw*) dan lain sebagainya. Untuk lebih jelas mengenai Penggolongan Relay berdasarkan Jumlah Pole dan Throw, silakan lihat gambar dibawah ini:

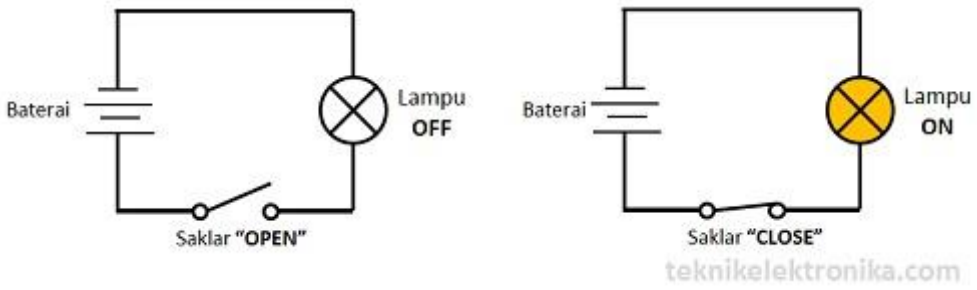


2.2. Saklar

Saklar atau lebih tepatnya adalah Saklar listrik adalah suatu komponen atau perangkat yang digunakan untuk memutuskan atau menghubungkan aliran listrik. Saklar yang dalam bahasa

Inggris disebut dengan Switch ini merupakan salah satu komponen atau alat listrik yang paling sering digunakan. Hampir semua peralatan Elektronika dan Listrik memerlukan Saklar untuk menghidupkan atau mematikan alat listrik yang digunakan. Pada dasarnya, sebuah Saklar sederhana terdiri dari dua bilah konduktor (biasanya adalah logam) yang terhubung ke rangkaian eksternal, Saat kedua bilah konduktor tersebut terhubung maka akan terjadi hubungan arus listrik dalam rangkaian. Sebaliknya, saat kedua konduktor tersebut dipisahkan maka hubungan arus listrik akan ikut terputus.

Saklar yang paling sering ditemukan adalah Saklar yang dioperasikan oleh tangan manusia dengan satu atau lebih pasang kontak listrik. Setiap pasangan kontak umumnya terdiri dari 2 keadaan atau disebut dengan "*State*". Kedua keadaan tersebut diantaranya adalah Keadaan "Close" atau "Tutup" dan Keadaan "Open" atau "Buka". Close artinya terjadi sambungan aliran listrik sedangkan Open adalah terjadinya pemutusan aliran listrik sumber.



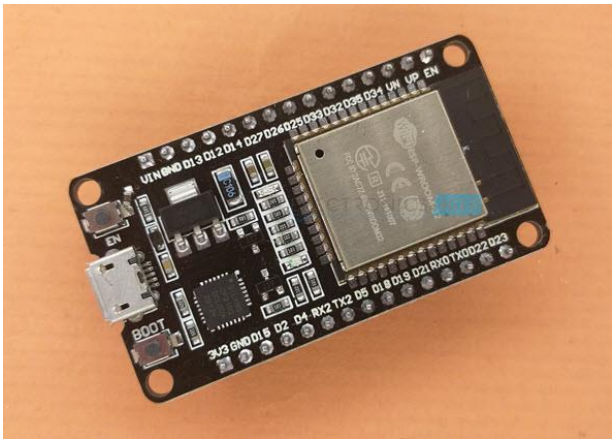
2.3. Lampu Bohlam

Sama seperti keluarga dioda yang lainnya, LED juga memiliki 2 buah kutub yakni kutub positif (P) dan negatif (N). Untuk dapat memancarkan cahaya, LED terlebih dahulu perlu dialiri oleh arus listrik. Pada rangkaian, arus listrik nantinya akan dialirkan dengan sistim bias maju yakni dari anoda ke katoda. Ketika hal tersebut terjadi, kelebihan elektron yang terdapat pada kutub negatif (N) akan berpindah pada tempat lain. Misalnya saja pada area yang memiliki muatan positif (P material). Setelah itu elektron akan bertemu dengan hole lalu melepaskan proton sehingga LED akan memancarkan cahaya dengan bentuk satu warna (monokromatik) sumber.

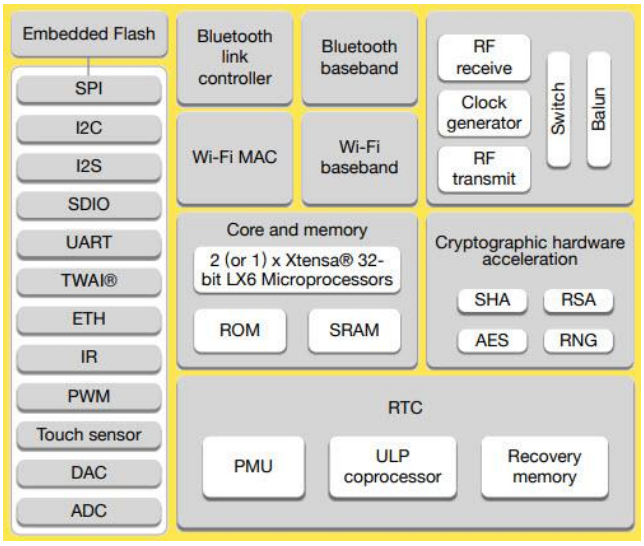
2.4. ESP-32

ESP32 adalah *System on Chip (SoC) Microcontroller* berbiaya rendah dari Espressif Systems, pengembang SoC ESP8266 yang terkenal. Ini adalah penerus ESP8266 SoC dan hadir dalam variasi single-core dan dual-core dari Mikroprosesor Xtensa LX6 32-bit Tensilica dengan Wi-Fi dan Bluetooth terintegrasi. Hal yang baik tentang ESP32 sama seperti ESP8266

adalah komponen RF terintegrasi seperti *Power Amplifier, Low-Noise Receive Amplifier, Antenna Switch, Filter* dan *RF Balun*. Ini membuat merancang perangkat keras di sekitar ESP32 sangat mudah karena Anda memerlukan sangat sedikit komponen eksternal.



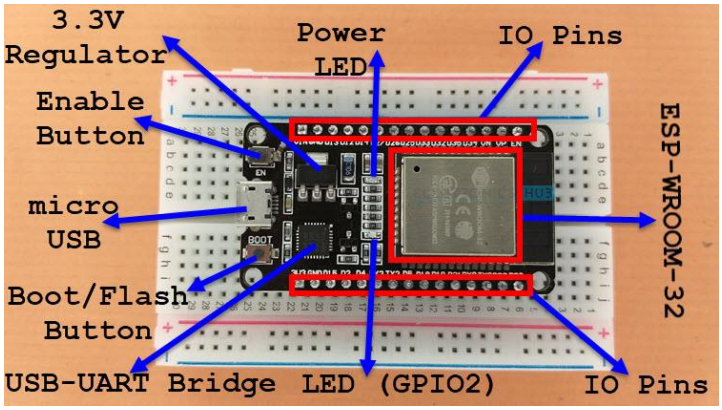
Hal penting lainnya yang perlu diketahui tentang ESP32 adalah diproduksi menggunakan teknologi 40 nm berdaya sangat rendah TSMC. Jadi, merancang aplikasi yang dioperasikan dengan baterai seperti perangkat yang dapat dikenakan, peralatan audio, monitor bayi, jam tangan pintar, dll., Menggunakan ESP32 seharusnya sangat mudah.



ESP32 memiliki lebih banyak fitur daripada ESP8266 dan sulit untuk menyertakan semua spesifikasi dalam panduan Memulai dengan ESP32 ini. Berikut beberapa spesifikasi penting ESP32.

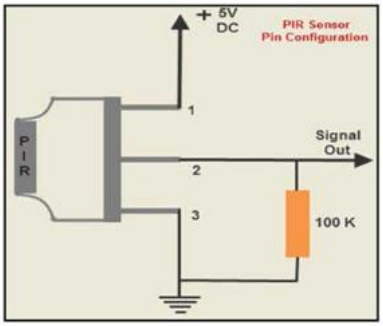
- Mikroprosesor LX6 Single atau Dual-Core 32-bit dengan frekuensi clock hingga 240 MHz.
- 520 KB SRAM, 448 KB ROM dan 16 KB RTC SRAM.
- Mendukung konektivitas Wi-Fi 802.11 b/g/n dengan kecepatan hingga 150 Mbps.
- Dukungan untuk spesifikasi Bluetooth v4.2 dan BLE Klasik.
- 34 GPIO yang Dapat Diprogram.
- Hingga 18 saluran SAR ADC 12-bit dan 2 saluran DAC 8-bit
- Konektivitas Serial termasuk 4 x SPI, 2 x I2C, 2 x I2S, 3 x UART.
- Ethernet MAC untuk Komunikasi LAN fisik (memerlukan PHY eksternal).
- 1 Host controller untuk SD/SDIO/MMC dan 1 Slave controller untuk SDIO/SPI.
- Motor PWM dan hingga 16-saluran LED PWM.
- Boot Aman dan Enkripsi Flash.
- Akselerasi perangkat keras kriptografi untuk AES, hash (SHA-2), RSA, ECC, dan RNG.

Gambar berikut menunjukkan tata letak Papan ESP32 yang saya miliki. Ada banyak Papan ESP32 berdasarkan Modul ESP-WROOM-32 yang tersedia di pasaran. Tata letak, pinout, dan fitur bervariasi dari papan ke papan. Papan yang saya miliki memiliki 30 Pin (15 pin di setiap sisi). Ada beberapa papan dengan 36 Pin dan beberapa dengan Pin yang sedikit lebih sedikit. Jadi, periksa kembali pin sebelum membuat koneksi atau bahkan menyalakan papan. sumber



2.5. PIR Sensor

Sensor PIR atau *Passive Infra Red* adalah sensor yang digunakan untuk mendeteksi pergerakan manusia dalam jarak tertentu. Rentang jarak obyek yang dapat dideteksi oleh sensor ini adalah antara 5m sampai 12m. Umumnya jenis sensor ini dibuat dari bahan utama sensor pieroelektrik yang berguna untuk mendeteksi tingkat radiasi inframerah suatu obyek. Sebenarnya ada banyak jenis sensor PIR yang umum tersedia di pasaran, salah satunya adalah yang berbentuk lensa fresnel kubah. Penggunaan sensor ini biasanya pada proyek rangkaian untuk mendeteksi keberadaan manusia pada suatu area tertentu. Dimana sensor ini akan terhubung ke modul rangkaian elektronika untuk dibaca dan diambil keputusan, misalnya mengaktifkan sistem peringatan.



Setiap kali ada obyek yang memiliki suhu panas, misalnya manusia melewati bidang jangkauan sensor PIR, maka sensor akan mendeteksi kemunculan obyek tersebut. Sinar inframerah yang dideteksi oleh sensor akan diubah oleh sensor menjadi sinyal listrik yang dapat digunakan untuk mengaktifkan alarm atau sistem peringatan lainnya.

Sensor PIR secara internal terdiri dari dua bagian : satu bagian positif dan bagian yang lainnya dianggap negatif. Jadi, setengah bagian menghasilkan satu sinyal ketika mendeteksi gerakan benda panas dan setengah lainnya menghasilkan jenis sinyal lain. Kedua sinyal yang berbeda ini dihasilkan sebagai sinyal output. Sensor ini terdiri dari lensa Fresnel yang memiliki dua cabang untuk mendeteksi radiasi inframerah yang dihasilkan oleh gerakan benda panas pada rentang yang luas atau area tertentu sumber.

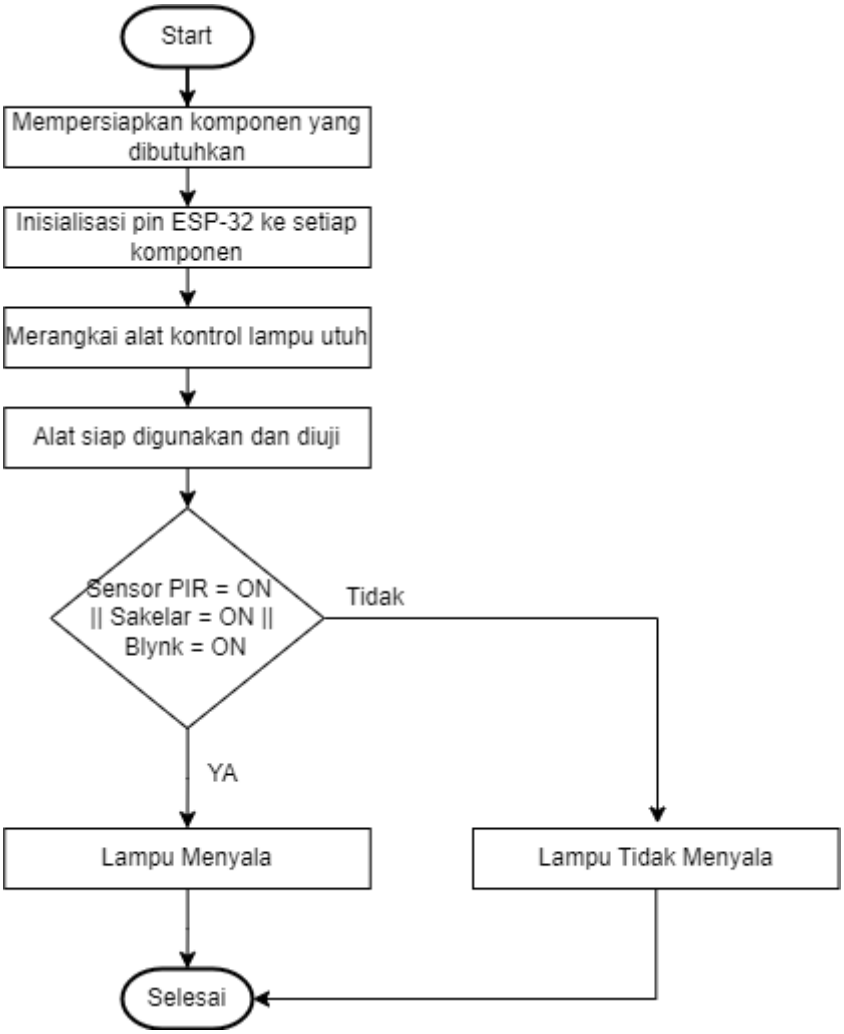
2.6. Kabel

Kabel Listrik yang dalam bahasa Inggris disebut dengan *Electrical Cable* adalah media untuk menghantarkan arus listrik yang terdiri dari Konduktor dan Isolator. Konduktor atau bahan penghantar listrik yang biasanya digunakan oleh Kabel Listrik adalah bahan Tembaga dan juga yang berbahan Aluminium meskipun ada juga yang menggunakan Silver (perak) dan emas sebagai bahan konduktornya namun bahan-bahan tersebut jarang digunakan karena harganya yang sangat mahal. Sedangkan Isolator atau bahan yang tidak/sulit menghantarkan arus listrik yang digunakan oleh Kabel Listrik adalah bahan Thermoplastik dan Thermosetting yaitu polymer (plastik dan rubber/karet) yang dibentuk dengan satu kali atau beberapa kali pemanasan dan pendinginan. sumber

BAB III
METODOLOGI PEMBUATAN

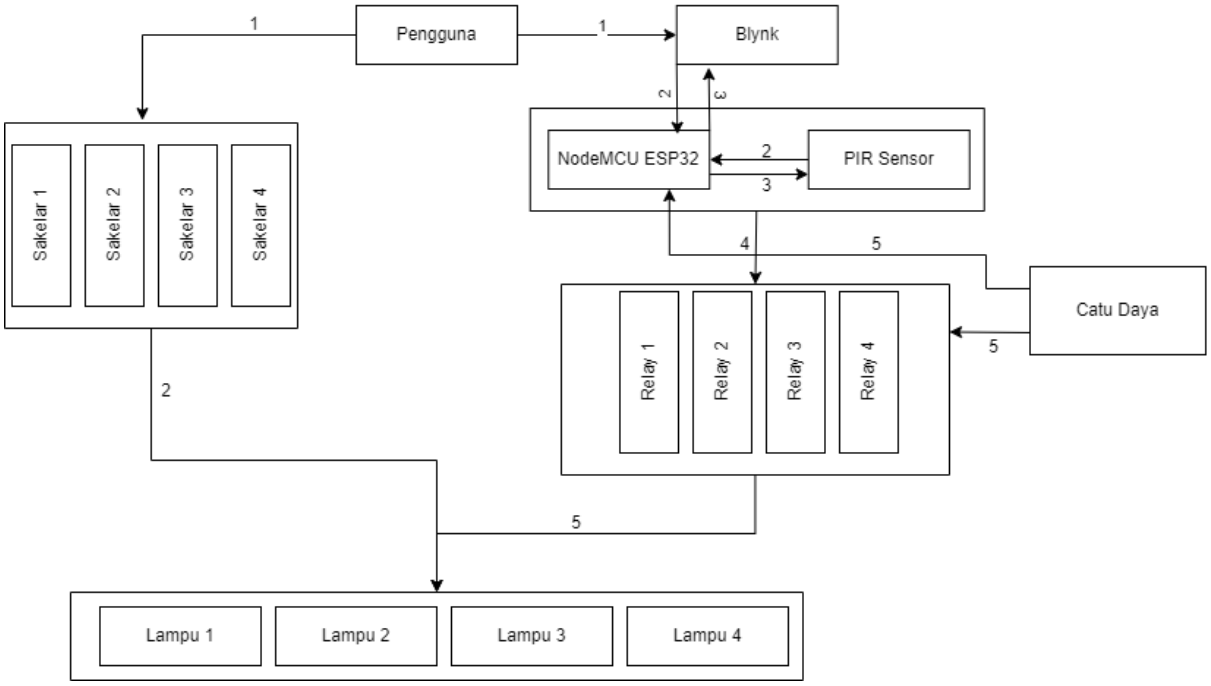
3.1. Diagram Alir

Berikut ini merupakan diagram alir dari alur pembuatan serta sistem kerja alat kontrol lampu otomatis dengan menggunakan sensor PIR dan kontrol IoT dengan Blynk.



3.2. Diagram Blok

Diagram blok merupakan skema dari rangkaian yang akan dibuat, karena dalam diagram blok ini hanya terdapat jalur antara blok – blok. Dimana masing-masing blok bisa mewakili komponen penopang yang berhubungan dengan rangkaian yang sesungguhnya. Diagram blok kontrol lampu otomatis dengan menggunakan sensor PIR dan kontrol IoT dengan Blynk, terdiri dari catu daya, Node MCU ESP32, modul Relay 1 Channel, Aplikasi Blynk, Sensor PIR, Sakelar dan Lampu. Blok diagram sistem secara keseluruhan ditunjukkan pada gambar berikut.



3.3. Alat dan Bahan

Berikut merupakan beberapa peralatan dan bahan bahan yang digunakan dalam pembuatan alat sistem kontrol lampu otomatis dengan sensor PIR dan kontrol IoT menggunakan Blynk.

a) **Relay**
b) **Saklar**
c) **Lampu Bohlam**
d) **ESP-32**
e) **PIR Sensor**
f) **Kabel**

3.4. Rancangan Anggaran Biaya

Berikut ini merupakan rancangan anggaran biaya yang diperkirakan dalam pembuatan alat sistem kontrol lampu otomatis dengan sensor PIR dan kontrol IoT menggunakan Blynk.

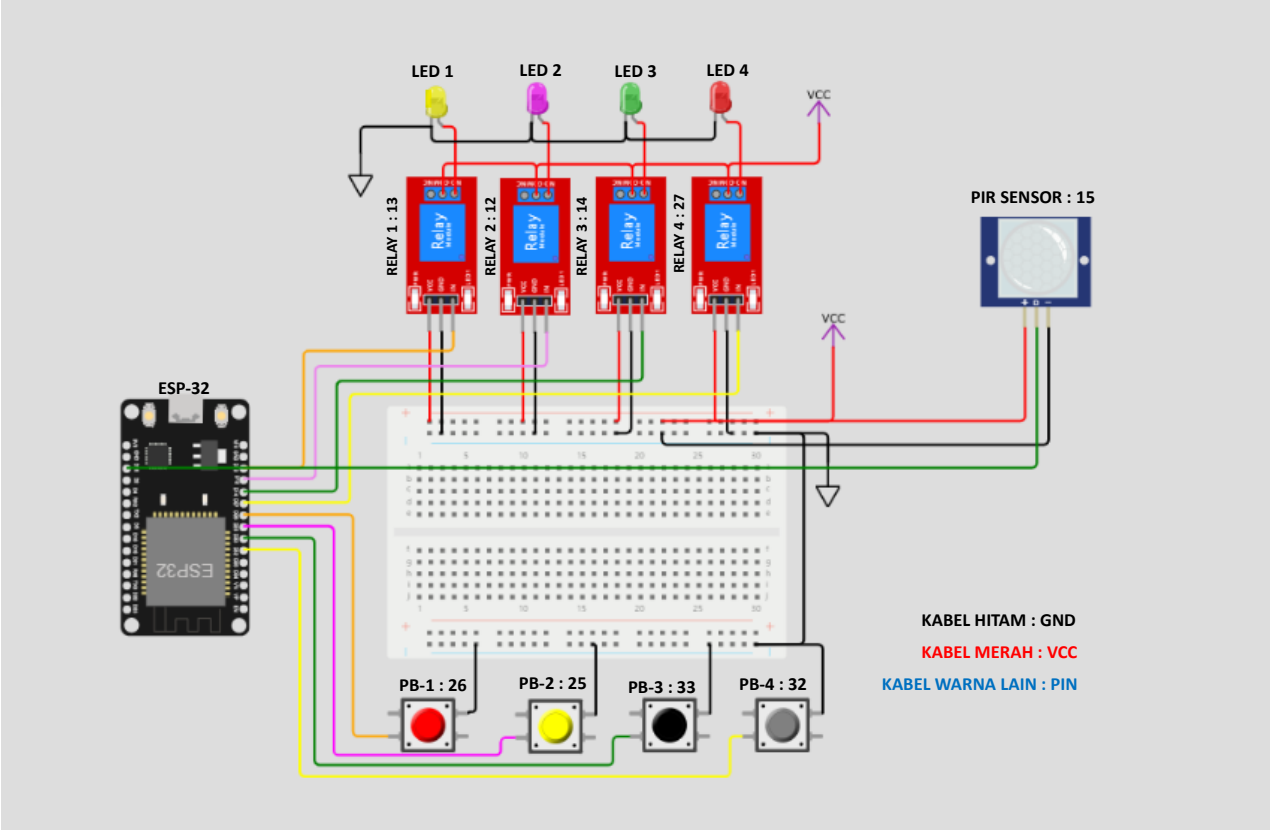
RAB Kontrol Lampu Otomatis						
No	**Bahan**	**Merk**	**Volume**	**Satuan**	**Harga Satuan**	**Harga Total**
1	Modul Relay 1 Channel	Relay Module	4	pcs	12,000.00	48,000.00
2	Sakelar	IB SC-151	4	pcs	5,000.00	20,000.00
3	PIR Sensor	HC-SR501	2	pcs	14,000.00	28,000.00
4	Lampu Bohlam 60 watt	Pancaran Bohlam Jumbo	4	pcs	20,000.00	80,000.00
5	Kabel Data Arduino	Due Leonardo	1	pcs	15,000.00	15,000.00
6	Breadboard	Easyware	1	pcs	10,000.00	10,000.00
6	ESP32	ESP-32S	1	pcs	75,000.00	75,000.00

7	Kabel (Serabut) 17 M	Hakamitsu	1	Gulung	40,000.00	40,000.00
8	Kabel (Tunggal) 45 M	Hakamitsu	1	Gulung	150,000.00	150,000.00
9	Kabel Jumper	Jevon Elektronik	3	Pack	15,000.00	45,000.00
10	Langganan Aplikasi Blynk	Blynk	12	Bulan	150,000.00	1,800,000.00
11	Black box	-	1	pcs	20,000.00	20,000.00
12	Cetak cover	-	1000	gram	1,000.00	1,000,000.00
13	Adaptor AC to DC	-	1	pcs	50,000.00	50,000.00
	Total					**3,381,000.00**

BAB IV
HASIL DAN PEMBAHASAN

4.1. Rangkaian Skematik

Berikut ini merupakan rangkaian skematik dari sistem kontrol lampu otomatis dengan sensor PIR dan kontrol IoT menggunakan Blynk yang telah dibuat.



Berikut merupakan gambar dari setiap komponen yang digunakan:

a. Relay Module 1 Channel

b. Saklar

c. PIR Sensor

d. ESP-32

e. Lampu Bohlam 60 Watt

f. Kabel Serat Optik / Serabut

g. Kabel Singular / Tunggal

h. Kabel Jumper

i. Adaptor

4.2. Source Code

Berikut ini merupakan kode yang digunakan untuk mengoperasikan rangkaian simulasi yang telah dibuat.

```
#define BLYNK_TEMPLATE_ID "TMPL6T7VQuyH4"
#define BLYNK_TEMPLATE_NAME "Automation Room"
#define BLYNK_AUTH_TOKEN "LKweClhPJ58hhhGKrM26VXqw7mnh_Ukn"

// Comment this out to disable prints and save space
#define BLYNK_PRINT Serial

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

char auth[] = BLYNK_AUTH_TOKEN;

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "Wokwi-GUEST";
char pass[] = "";

BlynkTimer timer;

#define button1_pin 26
#define button2_pin 25
#define button3_pin 33
#define button4_pin 32
#define pir_sensor_pin 15 // PIR sensor pin

#define relay1_pin 13
#define relay2_pin 12
#define relay3_pin 14
#define relay4_pin 27

int relay1_state = 0;
int relay2_state = 0;
int relay3_state = 0;
int relay4_state = 0;

// Change the virtual pins according to the rooms
```

```
#define button1_vpin    V1
#define button2_vpin    V2
#define button3_vpin    V3
#define button4_vpin    V4

bool pir_triggered = false; // Flag to track PIR sensor state

// This function is called every time the device is connected to the Blynk.Cloud
// Request the latest state from the server
BLYNK_CONNECTED() {
  Blynk.syncVirtual(button1_vpin);
  Blynk.syncVirtual(button2_vpin);
  Blynk.syncVirtual(button3_vpin);
  Blynk.syncVirtual(button4_vpin);
}

// This function is called every time the Virtual Pin state changes
// i.e., when a web push switch from Blynk App or Web Dashboard is activated
BLYNK_WRITE(button1_vpin) {
  relay1_state = param.asInt();
  digitalWrite(relay1_pin, relay1_state);
}

BLYNK_WRITE(button2_vpin) {
  relay2_state = param.asInt();
  digitalWrite(relay2_pin, relay2_state);
}

BLYNK_WRITE(button3_vpin) {
  relay3_state = param.asInt();
  digitalWrite(relay3_pin, relay3_state);
}

BLYNK_WRITE(button4_vpin) {
  relay4_state = param.asInt();
  digitalWrite(relay4_pin, relay4_state);
}

void setup() {
  // Debug console
```

```
  Serial.begin(115200);

  pinMode(button1_pin, INPUT_PULLUP);
  pinMode(button2_pin, INPUT_PULLUP);
  pinMode(button3_pin, INPUT_PULLUP);
  pinMode(button4_pin, INPUT_PULLUP);
  pinMode(pir_sensor_pin, INPUT_PULLUP); // Set PIR sensor pin as input

  pinMode(relay1_pin, OUTPUT);
  pinMode(relay2_pin, OUTPUT);
  pinMode(relay3_pin, OUTPUT);
  pinMode(relay4_pin, OUTPUT);

  // During starting, all relays should be turned off
  digitalWrite(relay1_pin, HIGH);
  digitalWrite(relay2_pin, HIGH);
  digitalWrite(relay3_pin, HIGH);
  digitalWrite(relay4_pin, HIGH);

  Blynk.begin(auth, ssid, pass);
}

void loop() {
  Blynk.run();
  timer.run();
  listen_push_buttons();

  // Check PIR sensor
  check_pir_sensor();
}

void listen_push_buttons() {
  if (digitalRead(button1_pin) == LOW) {
    delay(200);
    control_relay(1);
    Blynk.virtualWrite(button1_vpin, relay1_state); // Update button state
  } else if (digitalRead(button2_pin) == LOW) {
    delay(200);
    control_relay(2);
    Blynk.virtualWrite(button2_vpin, relay2_state); // Update button state
  } else if (digitalRead(button3_pin) == LOW) {
```

```
    delay(200);
    control_relay(3);
    Blynk.virtualWrite(button3_vpin, relay3_state); // Update button state
  } else if (digitalRead(button4_pin) == LOW) {
    delay(200);
    control_relay(4);
    Blynk.virtualWrite(button4_vpin, relay4_state); // Update button state
  }
}

void control_relay(int relay) {
  if (relay == 1) {
    relay1_state = !relay1_state;
    digitalWrite(relay1_pin, relay1_state);
    Serial.print("Relay1 State = ");
    Serial.println(relay1_state);
    delay(50);
  } else if (relay == 2) {
    relay2_state = !relay2_state;
    digitalWrite(relay2_pin, relay2_state);
    delay(50);
  } else if (relay == 3) {
    relay3_state = !relay3_state;
    digitalWrite(relay3_pin, relay3_state);
    delay(50);
  } else if (relay == 4) {
    relay4_state = !relay4_state;
    digitalWrite(relay4_pin, relay4_state);
    delay(50);
  }
}

void check_pir_sensor() {
  if (digitalRead(pir_sensor_pin) == HIGH) {
    if (!pir_triggered) {
      // PIR sensor triggered
      pir_triggered = true;
      // Turn on all relays
      digitalWrite(relay1_pin, HIGH);
      digitalWrite(relay2_pin, HIGH);
      digitalWrite(relay3_pin, HIGH);
```

```
      digitalWrite(relay4_pin, HIGH);
      Serial.println("Motion detected! All lights turned on.");
    }
  } else {
    if (pir_triggered) {
      // PIR sensor deactivated
      pir_triggered = false;
    }
  }
}
```

Berikut merupakan penjelasan dari kode yang dimasukan diatas beserta dengan fungsi fungsinya

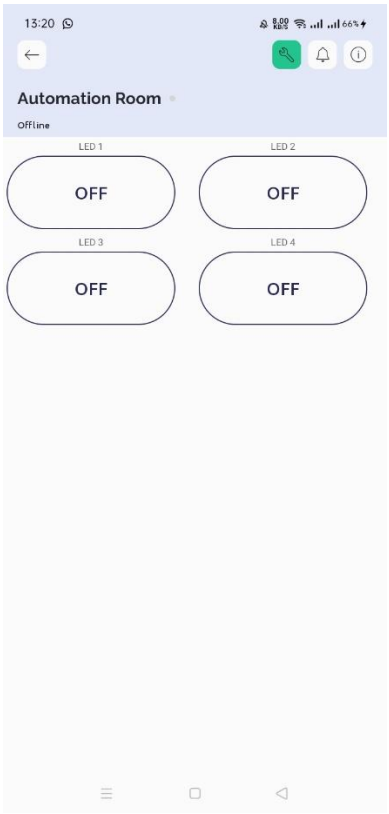
1. **#define BLYNK_TEMPLATE_ID "TMPL6T7VQuyH4":** Mendefinisikan ID template Blynk yang digunakan dalam proyek.
2. **#define BLYNK_TEMPLATE_NAME "Automation Room":** Mendefinisikan nama template Blynk yang digunakan dalam proyek.
3. **#define BLYNK_AUTH_TOKEN "LKweClhPJ58hhhGKrM26VXqw7mnh_Ukn":** Mendefinisikan token otentikasi Blynk yang diperlukan untuk menghubungkan perangkat ke server Blynk.
4. **#define BLYNK_PRINT Serial**: Mengaktifkan pencetakan informasi debug ke serial monitor.
5. **#include <WiFi.h>:** Memasukkan library WiFi yang diperlukan untuk mengakses fitur jaringan WiFi pada ESP32.
6. **#include <WiFiClient.h>:** Memasukkan library WiFiClient yang diperlukan untuk koneksi WiFi.
7. **#include <BlynkSimpleEsp32.h>:** Memasukkan library BlynkSimpleEsp32 yang menyediakan kelas untuk berkomunikasi dengan server Blynk pada platform ESP32.
8. **char auth[] = BLYNK_AUTH_TOKEN**; : Mendeklarasikan variabel array `auth` yang akan menyimpan token otentikasi Blynk.
9. **char ssid[] = "Wokwi-GUEST";** : Mendeklarasikan variabel array `ssid` yang menyimpan nama SSID dari jaringan WiFi yang akan digunakan.
10. **char pass[] = "";** : Mendeklarasikan variabel array `pass` yang menyimpan kata sandi (jika ada) dari jaringan WiFi yang akan digunakan.
11. **BlynkTimer timer;** : Membuat objek timer untuk digunakan dalam program.
12. **#define button1_pin 26** : Mendefinisikan pin GPIO yang digunakan untuk tombol 1.

13. **#define button2_pin 25** : Mendefinisikan pin GPIO yang digunakan untuk tombol 2.
14. **#define button3_pin 33**: Mendefinisikan pin GPIO yang digunakan untuk tombol 3.
15. **#define button4_pin 32**: Mendefinisikan pin GPIO yang digunakan untuk tombol 4.
16. **#define pir_sensor_pin 15**: Mendefinisikan pin GPIO yang digunakan untuk sensor PIR.
17. **#define relay1_pin 13**: Mendefinisikan pin GPIO yang digunakan untuk relay 1.
18. **#define relay2_pin 12**: Mendefinisikan pin GPIO yang digunakan untuk relay 2.
19. **#define relay3_pin 14**: Mendefinisikan pin GPIO yang digunakan untuk relay 3.
20. **#define relay4_pin 27**: Mendefinisikan pin GPIO yang digunakan untuk relay 4.
21. **`int relay1_state = 0;`, `int relay2_state = 0;`, `int relay3_state = 0;`, `int relay4_state = 0;`:** Mendeklarasikan variabel untuk menyimpan status relay.
22. **`#define button1_vpin V1`, `#define button2_vpin V2`, `#define button3_vpin V3`, `#define button4_vpin V4`:** Mendefinisikan nomor pin virtual Blynk yang akan digunakan untuk masing-masing tombol.
23. **`bool pir_triggered = false;`:** Mendeklarasikan variabel boolean yang digunakan untuk melacak apakah sensor PIR telah terpicu.
24. **`BLYNK_CONNECTED() {...}`:** Mendefinisikan fungsi yang akan dipanggil setiap kali perangkat terhubung ke server Blynk.
25. **`BLYNK_WRITE(button1_vpin) {...}`, `BLYNK_WRITE(button2_vpin) {...}`, `BLYNK_WRITE(button3_vpin) {...}`, `BLYNK_WRITE(button4_vpin) {...}`:** Mendefinisikan fungsi yang akan dipanggil setiap kali nilai pin virtual Blynk yang sesuai berubah.
26. **`void setup() {...}`:** Mendefinisikan fungsi setup yang akan dijalankan sekali saat perangkat pertama kali dinyalakan.
27. **`void loop() {...}`:** Mendefinisikan fungsi loop yang akan dijalankan secara berulang.
28. **`void listen_push_buttons() {...}`:** Mendefinisikan fungsi yang akan memeriksa tombol yang ditekan dan mengontrol relai.
29. **`void control_relay(int relay) {...}`:** Mendefinisikan fungsi yang akan mengontrol relai berdasarkan nomor relay yang diberikan.
30. **`void check_pir_sensor() {...}`:** Mendefinisikan fungsi yang akan memeriksa status sensor PIR dan mengontrol relai berdasarkan statusnya.

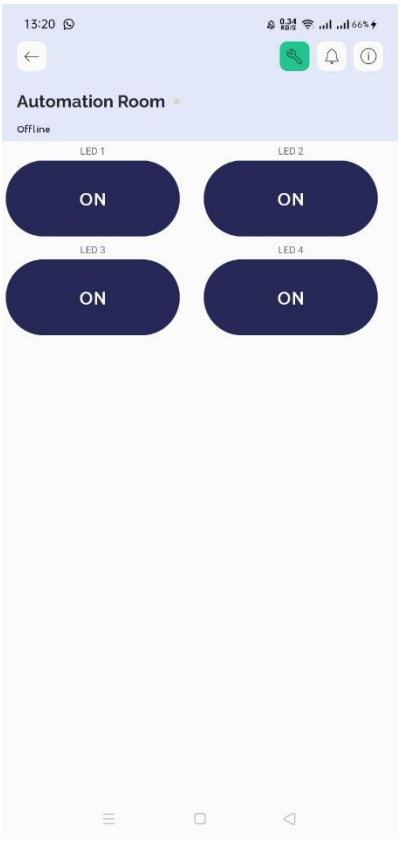
4.3. Tampilan Antarmuka Blynk

Berikut merupakan tampilan antarmuka pada aplikasi Blynk saat pada 2 kondisi seperti berikut.

a. Kondisi ketika semua lampu dalam keadaan tidak menyala / OFF



b. Kondisi ketika semua lampu dalam keadaan menyala / ON



4.4. Hasil Simulasi

Berdasarkan hasil simulasi yang telah dilakukan, ditemukan bahwa dengan menggunakan rangkaian dan kode yang telah disusun, sensor PIR mampu dengan efektif mendeteksi keberadaan objek yang mengeluarkan hawa panas atau inframerah, khususnya manusia. Hal ini memungkinkan sistem untuk memberikan perintah otomatis untuk menyalakan keempat lampu yang terpasang. Selain itu, keempat lampu tersebut juga dapat dikontrol secara manual melalui saklar atau push button yang tersedia pada rangkaian, memberikan fleksibilitas tambahan dalam pengaturan pencahayaan.

Selain aspek keberadaan fisik, proyek ini juga berhasil mengintegrasikan teknologi Internet of Things (IoT) untuk mengontrol lampu dari jarak jauh menggunakan internet. Dengan demikian, pengguna dapat mengatur pencahayaan rumah atau area tertentu dengan mudah dari perangkat pintar mereka, seperti ponsel atau komputer, tanpa harus berada di dekat rangkaian lampu tersebut. Dengan implementasi yang berhasil ini, proyek secara keseluruhan memenuhi dan bahkan melebihi harapan yang telah ditetapkan sebelumnya, menunjukkan kemampuan teknis dan inovatif yang kuat dalam pengembangan solusi berbasis IoT untuk kebutuhan sehari-hari.

BAB V
PENUTUP

5.1. Kesimpulan

Berdasarkan hasil simulasi yang dilakukan, proyek ini berhasil menunjukkan kemampuan efektif sensor PIR dalam mendeteksi keberadaan objek yang mengeluarkan hawa panas, khususnya manusia. Integrasi teknologi IoT melalui platform Blynk memungkinkan pengendalian lampu dari jarak jauh melalui internet, meningkatkan fleksibilitas dan kenyamanan pengguna. Implementasi proyek ini memenuhi dan bahkan melebihi harapan awal, menunjukkan kemampuan teknis dan inovatif dalam pengembangan solusi berbasis IoT untuk kebutuhan sehari-hari.