BAB IV
PEMBUATAN ALAT

4.1 Pembuatan Prototipe

Pembuatan prototipe merupakan tahap penting dalam merealisasikan rancangan sistem menjadi bentuk fisik yang dapat diuji dan dianalisis. Prototipe dibuat untuk merepresentasikan dan menguji konsep sistem secara nyata, baik dari sisi fungsionalitas maupun efektivitasnya dalam kondisi lapangan. Dalam tugas akhir ini, pembuatan prototipe mencakup perakitan perangkat keras serta pengembangan perangkat lunak yang mendukung fungsi sistem secara menyeluruh. Tujuan dari pembuatan prototipe adalah untuk memastikan bahwa seluruh komponen dapat bekerja secara terintegrasi sesuai dengan desain, serta mampu memenuhi kebutuhan dan tujuan sistem yang telah dirancang. Terdapat dua tahapan utama dalam pembuatan prototipe pada tugas akhir ini, yaitu pembuatan perangkat keras dan pembuatan perangkat lunak.

a. Pembuatan Perangkat Keras

Tahapan ini mencakup proses perakitan seluruh komponen fisik yang membentuk sistem, seperti sensor, mikrokontroler, dan komponen pendukung lainnya. Pembuatan perangkat keras dimulai dari pemilihan komponen sesuai kebutuhan, perancangan rangkaian elektronik, penyusunan wiring diagram, hingga proses penyolderan dan perakitan komponen ke dalam wadah atau casing yang telah dirancang. Selain itu, dilakukan pula perancangan mekanik dan pembuatan PCB guna mendukung tata letak komponen agar lebih rapi, efisien, dan mudah dalam pemeliharaan. Tahapan ini penting untuk memastikan bahwa seluruh perangkat dapat bekerja dengan baik dan terhubung satu sama lain dengan baik.

b. Pembuatan Perangkat Lunak

Tahapan ini berfokus pada penyusunan program atau kode yang ditanamkan pada mikrokontroler ESP32. Perangkat lunak dirancang untuk membaca data dari sensor tegangan (PZEM-004T V3.0) dan sensor gerak (PIR HC-SR501), lalu memproses data tersebut berdasarkan logika sistem, seperti mendeteksi kehilangan tegangan, keberadaan gerakan, dan mengirimkan notifikasi melalui aplikasi

Telegram. Proses ini meliputi penyusunan flowchart sistem, penulisan kode program menggunakan Arduino IDE, pengujian fungsi sistem, serta debugging apabila terjadi kesalahan atau ketidaksesuaian. Keberhasilan sistem secara keseluruhan sangat ditentukan oleh ketepatan logika dan stabilitas perangkat lunak yang digunakan.

4.2 Alat dan Bahan

Pada subbab ini disajikan daftar alat dan bahan yang digunakan dalam proses pembuatan prototipe sistem deteksi pencurian kabel output tranformator menuju PHB-TR berbasis IoT. Seluruh alat dan bahan dipilih berdasarkan fungsi dan kebutuhan sistem, serta mendukung proses perakitan perangkat keras dan pengembangan perangkat lunak. Rincian lengkap alat dan bahan yang digunakan dapat dilihat pada Tabel 4.1 dan Tabel 4.2.

Tabel 4. 1 Alat yang digunakan

No.	Komponen	Spesifikasi	Jumlah
1.	Multimeter Digital	Dekko	1 buah
2.	Obeng Plus	-	1 buah
3.	Obeng Minus	-	1 buah
4.	Bor Baterai	ZHSEN	1 Set
5.	Bor Mini	Kitani	1 Set
6.	Tespen	Sefanly	1 buah
7.	Tang potong	-	1 buah
8.	Gunting	-	1 buah
9.	Spidol Hitam Permanent	Snowman	1 buah
10.	Bubuk FeCl3 (Pelarut)	FerriChlorida	50 gram
11.	Solasi hitam	Nasional	1 buah
12.	Solder	Taffware	1 buah
13.	Amplas	Amplas 120	1 lembar
14.	Cutter	Kenko	1 buah

Tabel 4. 2 Bahan yang digunakan

No.	Komponen	Spesifikasi	Jumlah
1.	Mikrokontroler	ESP32 Devkit V1 Type-C	1 buah
2.	Sensor Tegangan	PZEM-004T-V3.0 (100A)	3 buah
3.	Sensor Gerak	PIR HC-SR501	1 buah
4.	Modul Charging	XH-M604	1 buah
5.	Buck Converter	LM2596	1 buah
6.	Baterai Li-Ion	18650	2 buah
7.	Holder baterai	2S	1 buah
8.	Terminal blok 2x6 pin	-	4 buah
9.	Terminal blok 2 pin	-	2 buah
10.	Kabel JST	4 pin	3 set
11.	Kabel JST	3 pin	1 set
12.	PCB board	Ukuran 20x10	2 buah
13.	Spacer	4cm	4 buah
14.	Spacer	1 cm	4 buah
15.	Pin header female	38 pin	38 pin
16.	Pin header male	4 pin	4 pin
17.	Kabel NYAF	0,75 mm	20 meter
18.	Socket DC Power	-	1 buah
19.	Switch ON/OFF	-	1 buah
20.	Box Project	X7 (22x15) cm	1 buah
21.	Pilot Lamp Merah	INS	3 buah
22.	Adaptor	9V 2A	1 buah
23.	Kabel Charging	Type-C	1 set
24.	Modem Wifi	Hi-Net	1 buah
25.	Stopkontak	2 lubang	1 buah
26.	MCB 3 fasa	Chint 10 A	2 buah
27.	Rel MCB	-	2,5 meter

28.	Kabel Duct	-	1 meter
29.	Kabel Spiral	-	15 cm
30.	Busbar Netral	-	1 buah
31.	Lampu pijar	5 watt	1 buah
32.	Lampu pijar	15 watt	1 buah
33.	Lampu pijar	25 watt	1 buah
34.	Pipa PVC	Tipe D 3 inch	4 meter
35.	DOP pipa	Tipe D 3 inch	6 buah
36.	T pipa	Tipe D 3 inch	2 buah
37.	Klem pipa	Ukuran 3 inch	4 buah

4.3 Pembuatan Perangkat Keras

Pembuatan perangkat keras merupakan tahap penting dalam proses realisasi prototipe sistem, yang melibatkan penyusunan dan perakitan seluruh komponen fisik yang digunakan. Tahapan ini mencakup perancangan mekanik untuk wadah atau casing alat, penyusunan dan pengkabelan rangkaian elektronik, serta perancangan PCB untuk menata komponen agar lebih terstruktur dan efisien. Semua tahapan ini bertujuan agar perangkat dapat bekerja dengan baik, memiliki bentuk yang rapi, dan dapat dioperasikan.

4.3.1 Pembuatan Perangkat Mekanik

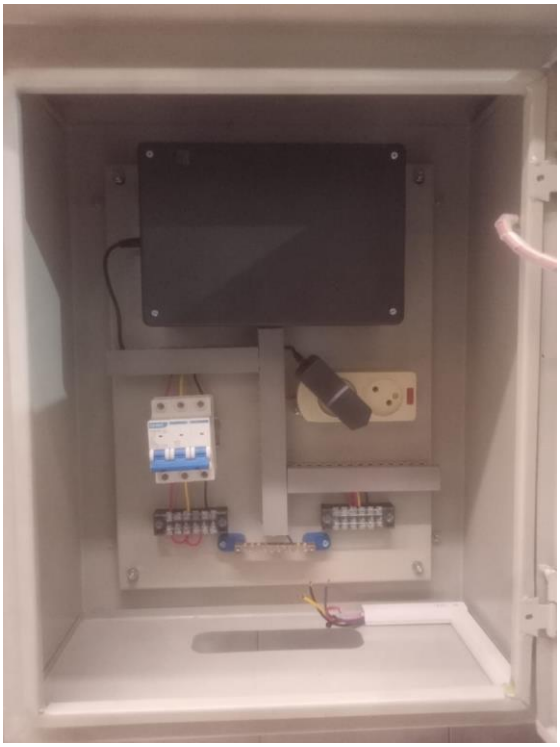
Perancangan mekanik pada sistem deteksi pencurian kabel output transformator menuju PHB-TR berbasis IoT ini difokuskan pada pembuatan wadah atau casing yang dapat menampung seluruh komponen elektronik secara rapi, aman, dan mudah dipasang di lingkungan gardu distribusi tipe portal khusunya di dalam panel PHB-TR. Desain casing harus mempertimbangkan tata letak komponen seperti mikrokontroler ESP32, sensor PZEM-004T V3.0, sensor gerak PIR HC-SR501, modul charging, baterai, serta kabel-kabel penghubungnya.

Box yang digunakan bersifat tertutup untuk melindungi perangkat dari debu dan kelembapan, namun tetap menyediakan lubang untuk memudahkan akses pada port socket dc power atau konektor lainnya. Selain itu, lubang juga disiapkan untuk

pemasangan kabel sensor gerak dan kabel input pada listrik AC dari sensor tegangan ke fasa R, S, dan T. Box alat ditunjukan pada Gambar 4.1.

Gambar 4. 1 Box Alat yang digunakan

Dalam perancangannya, dimensi casing disesuaikan agar seluruh komponen dapat ditata dengan efisien yaitu berukuran 22cm x 15 cm x 9 cm, dan alat dapat dipasang dengan mudah di dinding atau rak panel distribusi. Penempatan sensor dan modul diatur agar tidak saling mengganggu, serta memberikan ruang untuk pengujian, pemeliharaan, atau penggantian komponen jika diperlukan. Tatanan komponen dalam panel ditunjukkan pada Gambar 4.2.



Gambar 4. 2 Tatanan Komponen di Dalam Panel

Sebagai bentuk representasi nyata dari kondisi di lapangan, pada tahap ini juga dibuat sebuah diorama gardu portal yang berfungsi untuk mensimulasikan situasi gardu distribusi secara visual dan fisik. Diorama ini menggambarkan susunan rak panel hubung bagi (PHB-TR), kabel-kabel fasa dan netral, serta lokasi pemasangan alat pendeteksi. Pembuatan diorama bertujuan untuk memberikan gambaran menyeluruh bagaimana sistem ini akan dipasang dan bekerja di lokasi sesungguhnya, serta mempermudah proses demonstrasi dan pengujian sistem dalam skala kecil sebelum diterapkan di lapangan. Proses pembuatan diorama Gardu Portal ditujukan pada Gambar 4.3.

Gambar 4. 3 Proses Pembuatan Diorama Gardu Portal

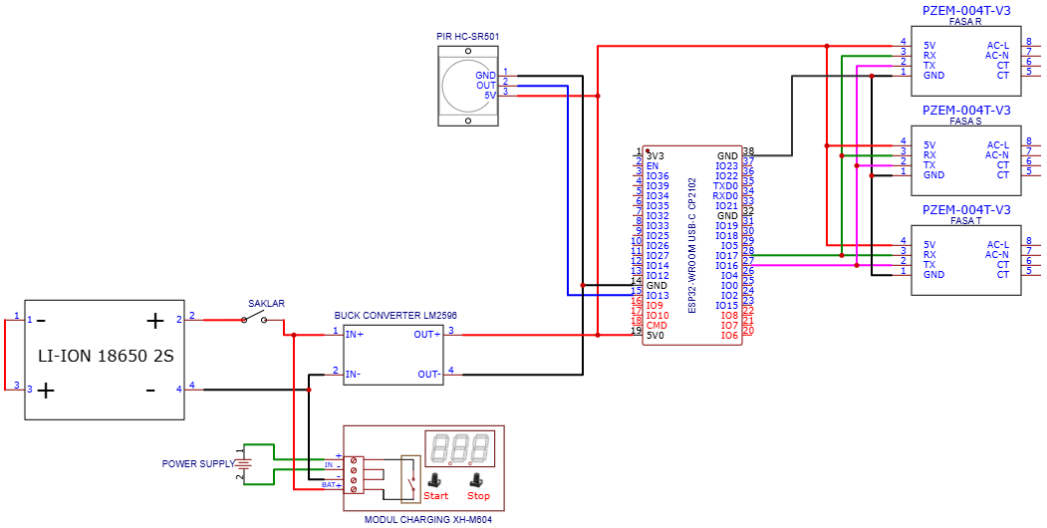
4.3.2 Pembuatan Perangkat Elektrik

Pembuatan perangkat elektrik merupakan tahapan penting dalam membangun sistem deteksi pencurian kabel berbasis IoT secara menyeluruh. Tahapan ini mencakup proses desain sirkuit elektronik, pembuatan papan sirkuit tercetak (PCB),

serta integrasi seluruh komponen elektronik agar dapat berfungsi secara terkoordinasi. Adapun rincian prosesnya adalah sebagai berikut:

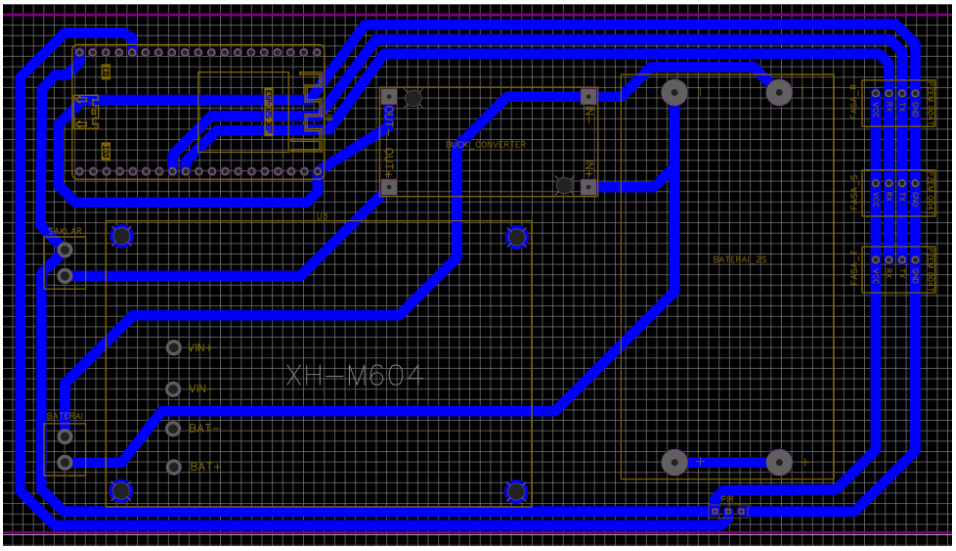
1. Perancangan Skematik dan Layout PCB

Langkah awal dalam pembuatan perangkat elektrik adalah merancang skematik rangkaian elektronik menggunakan software EasyEDA (Electronic Design Automation). Skematik ini mencakup hubungan antara ESP32 sebagai pusat kendali, sensor tegangan PZEM-004T V3.0, sensor PIR HC-SR501, baterai lithium-ion, modul charging baterai, regulator tegangan (LM2596), serta konektor-konektor pendukung lainnya. Skematik komponen ditunjukan pada Gambar 4.4.



Gambar 4. 4 Rancangan Wiring Diagram

Setelah skematik selesai, proses dilanjutkan dengan membuat layout PCB berdasarkan skema tersebut. Penempatan komponen dalam layout disesuaikan agar efisien secara ruang dan memudahkan proses soldering. Jalur-jalur koneksi dirancang sedemikian rupa untuk meminimalkan interferensi serta memaksimalkan keamanan sistem. Desain Layout PCB ditunjukkan pada Gambar 4.5.



Gambar 4. 5 Desain Layout PCB

2. Proses Pencetakan dan Penyolderan PCB

Setelah layout selesai, desain PCB kemudian dicetak menggunakan metode autan. Metode Autan merupakan teknik transfer toner untuk mencetak jalur rangkaian pada papan PCB tembaga dengan menggunakan cairan autan. Teknik ini memanfaatkan cetakan hasil printer laser pada kertas HVS, yang kemudian dipindahkan ke papan PCB menggunakan cairan Autan.

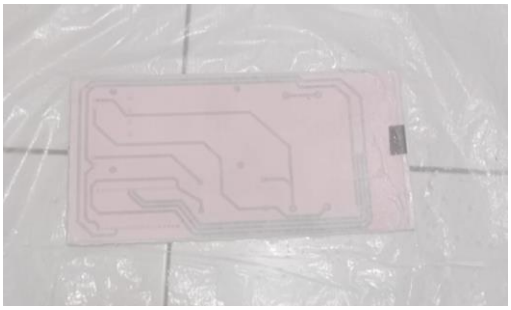
- Siapkan Papan PCB

  Papan PCB polos (lapisan tembaga) dipotong sesuai ukuran layout (20cmx10cm), lalu dibersihkan menggunakan amplas halus atau cairan pembersih agar permukaan tembaga bersih dan tidak berminyak. Bagian ini penting agar toner dapat menempel dengan baik.

Gambar 4. 6 Bahan Pembuatan PCB

- Transfer Toner ke PCB dengan Autan

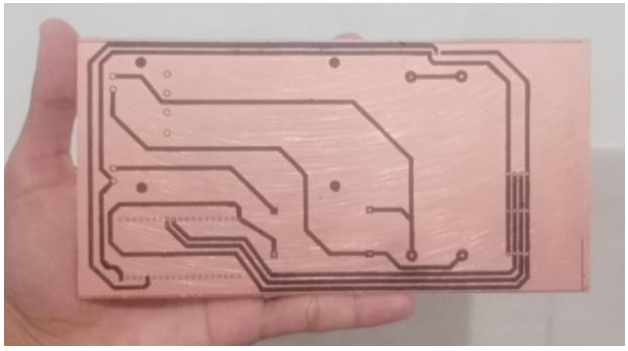
  Kertas hasil cetakan ditempelkan pada permukaan PCB. Setelah itu, cairan Autan disemprotkan secara merata di atas permukaan kertas hingga agak basah. Selanjutnya, kertas digosok perlahan menggunakan benda datar (seperti kartu atau penggaris) agar toner menempel ke permukaan tembaga. Diamkan beberapa menit.



Gambar 4. 7 Proses Transfer Toner ke PCB

- Pelepasan Kertas

  Setelah cairan mengering sebagian, kertas dilepaskan secara perlahan. Toner hitam dari printer akan menempel pada tembaga, membentuk pola jalur rangkaian yang diinginkan. Jika transfer tidak sempurna, dapat diulang atau diperbaiki manual dengan spidol tahan asam.



Gambar 4. 8 Hasil Transfer Toner ke PCB

- Proses Etching

  PCB dengan pola toner dimasukkan ke dalam larutan ferric chloride ($FeCl_3$) untuk melarutkan bagian tembaga yang tidak tertutup toner. Setelah beberapa menit, hanya bagian jalur yang tertutup toner yang akan tersisa.

Gambar 4. 9 Proses Etching PCB

- Pembersihan dan Finishing

  Setelah proses etching selesai, toner dibersihkan sehingga papan PCB siap dilubangi dan disolder.

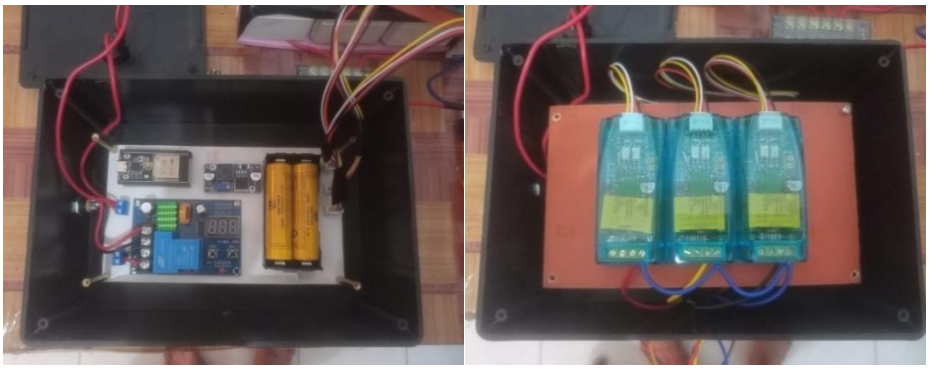
Gambar 4. 10 Proses Pengeboran Lubang PCB

Komponen-komponen seperti ESP32 (melalui pin header), soket sensor, resistor, dan komponen lainnya kemudian disolder secara manual sesuai dengan tata letak pada PCB. Penyolderan dilakukan secara hati-hati untuk menghindari korsleting atau koneksi yang tidak sempurna. Proses ini memastikan bahwa seluruh komponen terhubung dengan baik.

Gambar 4. 11 Proses Penyolderan Komponen ke PCB

3. Integrasi Sistem

Setelah proses penyolderan selesai dan seluruh komponen terpasang pada PCB, tahap selanjutnya adalah mengintegrasikan sistem secara keseluruhan. Modul-modul seperti sensor tegangan, sensor PIR, serta modul charging baterai dihubungkan ke papan utama melalui konektor atau kabel yang telah disiapkan. Proses integrasi mencakup pengujian koneksi, pengecekan kontinuitas, dan pengukuran tegangan untuk memastikan semua jalur bekerja sesuai rancangan.



Gambar 4. 12 Proses Integrasi Sistem

4.4 Pembuatan Perangkat Lunak

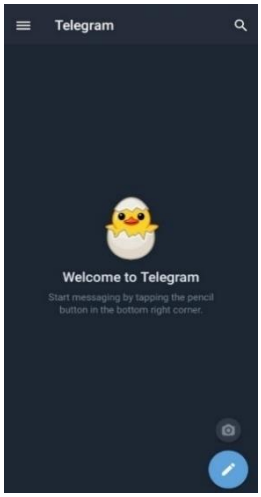
Pembuatan perangkat lunak merupakan tahap penting dalam mengatur logika kerja sistem agar dapat menjalankan fungsi-fungsi yang telah dirancang. Dalam tugas akhir ini, perangkat lunak dikembangkan untuk mikrokontroler ESP32 menggunakan software Arduino IDE**.** Perangkat lunak bertugas untuk membaca data dari sensor, memproses logika deteksi, serta mengirimkan notifikasi melalui platform Telegram.

4.4.1 Pembuatan Bot Telegram

Untuk memungkinkan sistem mengirimkan notifikasi secara otomatis melalui aplikasi Telegram, langkah awal yang harus dilakukan adalah membuat bot Telegram khusus. Bot ini bertindak sebagai pengirim pesan yang akan diakses oleh mikrokontroler (ESP32) dalam sistem. Proses pembuatan bot Telegram dilakukan menggunakan bantuan BotFather, yaitu bot resmi dari Telegram yang digunakan untuk membuat dan mengelola bot Telegram. Berikut adalah tahapan lengkapnya:

1. Buka Aplikasi Telegram

Langkah pertama adalah harus memiliki akun Telegram yang aktif. Setelah itu, buka aplikasi Telegram, baik di smartphone maupun melalui aplikasi desktop/web.



Gambar 4. 13 Tampilan Aplikasi Telegram

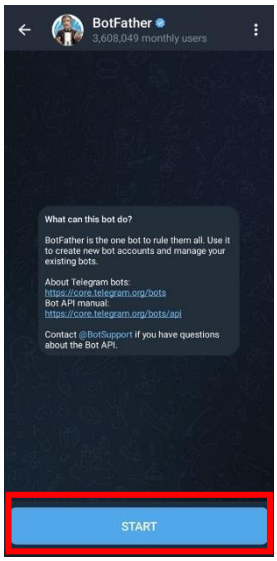
2. Membuat Bot dengan BotFather

Langkah selanjutnya adalah mencari akun BotFather. BotFather adalah bot resmi dari Telegram dengan tanda centang biru sebagai verifikasi. Cara mencarinya:

- Ketik @BotFather pada kolom pencarian. Pilih akun BotFather yang terverifikasi.

Gambar 4. 14 Mencari akun BotFather pada Telegram

- Tekan tombol Start atau ketik /start untuk memulai interaksi dengan BotFather.



Gambar 4. 15 Memulai interaksi dengan BotFather

3. Membuat Bot Baru

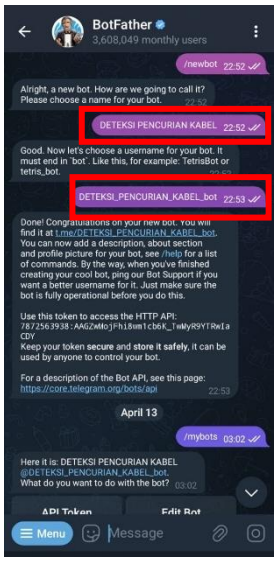
Setelah BotFather aktif, kirimkan perintah /newbot untuk membuat bot baru. BotFather akan meminta dua informasi:

- Nama Bot

  Masukkan nama lengkap bot sesuai keinginan, misalnya: **DETEKSI PENCURIAN KABEL**. Nama ini adalah nama tampilan bot, bebas dan tidak harus unik.

- Username Bot

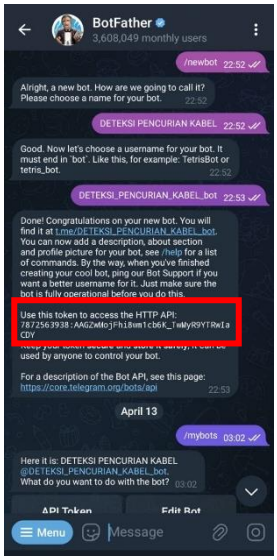
  Masukkan username yang unik dan harus diakhiri dengan kata bot.



Gambar 4. 16 Proses Pembuatan Bot Telegram

4. Mendapatkan Token Bot

Setelah berhasil membuat bot, BotFather akan memberikan token API, berupa teks acak panjang. Token bot adalah sebuah kode autentikasi yang dikeluarkan oleh Telegram melalui BotFather setelah pengguna berhasil membuat bot baru. Token ini berfungsi sebagai kunci akses resmi yang memungkinkan ESP32 berkomunikasi dengan server Telegram API dan mengirimkan perintah berupa pesan notifikasi. Tanpa token ini, sistem tidak akan diizinkan untuk menggunakan layanan bot Telegram secara otomatis.

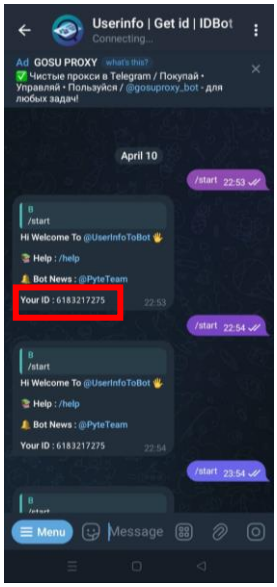


Gambar 4. 17 Mendapatkan Token Bot Telegram

5. Mendapatkan Chat ID Telegram

Untuk mengirimkan notifikasi dari ESP32 ke akun pengguna, sistem juga perlu mengetahui Chat ID, yaitu identitas unik akun Telegram pengguna**.** Chat ID adalah identifikasi unik dari akun Telegram penerima pesan, yang digunakan oleh bot untuk mengetahui ke mana notifikasi harus dikirim. Chat ID ini biasanya merujuk pada ID pengguna individu atau grup Telegram yang berinteraksi dengan bot. Dengan menyertakan chat ID ke dalam program, sistem dapat secara spesifik mengarahkan notifikasi ke pengguna yang dituju, seperti peringatan kehilangan tegangan, indikasi gerakan mencurigakan, atau status pemadaman listrik.

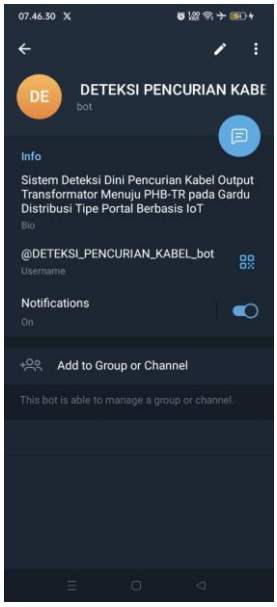
- Buka Telegram, cari bot dengan nama @UserInfoToBot.
- Tekan tombol Start.
- Bot akan membalas pesan berisi informasi pengguna, termasuk “Your ID”.



Gambar 4. 18 Mendapatkan ID Telegram

6. Pembutan Bot Selesai

Setelah melalui proses pembuatan bot menggunakan layanan BotFather di Telegram, mulai dari pemberian nama bot, pembuatan username unik, hingga memperoleh token bot, serta identifikasi chat ID pengguna yang akan menerima notifikasi, maka bot telah berhasil dikonfigurasi dan siap digunakan. Dengan tersedianya token bot dan chat ID, sistem dapat terhubung secara langsung dengan layanan Telegram API. Bot ini akan digunakan sebagai media komunikasi utama untuk mengirimkan notifikasi secara otomatis dari mikrokontroler ESP32 ke pengguna saat terdeteksi adanya kehilangan tegangan, gerakan mencurigakan, maupun pemadaman listrik. Dengan demikian, proses pembuatan dan konfigurasi bot Telegram telah selesai dan sistem siap menjalankan fungsi untuk mengirimkan notifikasi.



Gambar 4. 19 Berhasil Membuat Bot Telegram

4.4.2 Pembuatan Program Pada Software Arduino IDE

Setelah proses perancangan perangkat keras dan pembuatan bot Telegram selesai, tahap selanjutnya adalah pembuatan program yang ditanamkan ke dalam mikrokontroler ESP32 menggunakan Arduino IDE. Arduino IDE dipilih karena merupakan platform pemrograman yang sederhana, mendukung banyak jenis mikrokontroler termasuk ESP32, serta memiliki banyak pustaka (library) pendukung.

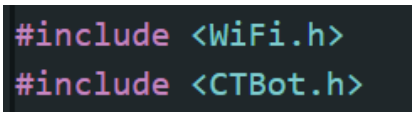
Program dibuat dengan logika utama untuk membaca tegangan 3 fasa (R, S, T) menggunakan sensor PZEM-004T V3.0 (100A) dan mendeteksi gerakan menggunakan sensor PIR HC-SR501. Setiap sensor PZEM diberikan alamat (address) yang berbeda agar dapat dibaca secara bergantian melalui komunikasi Serial UART (HardwareSerial). Selain itu, program juga mengatur waktu aktif sensor PIR. Data tegangan yang diperoleh digunakan untuk mendeteksi beberapa kondisi, yaitu:

- Kehilangan salah satu fasa
- Kehilangan ketiga fasa hampir bersamaan
- Tegangan tidak seimbang atau abnormal

Ketika kondisi-kondisi tersebut terpenuhi, sistem akan mengirimkan notifikasi otomatis ke Telegram melalui bot yang telah dikonfigurasi sebelumnya. Untuk keperluan komunikasi ini, program menggunakan library CTBot yang mendukung koneksi antara ESP32 dan Telegram API melalui jaringan Wi-Fi. Berikut akan dijelaskan bagian-bagian program yang dikembangkan pada pembuatan perangkat lunak.

4.4.2.1 Pembuatan Program Koneksi ESP32 dengan Wifi dan Bot Telegram

Pada bagian ini, dilakukan proses pengembangan program agar ESP32 dapat terhubung ke jaringan WiFi dan berkomunikasi dengan Bot Telegram, sehingga sistem mampu mengirim notifikasi terkait kondisi tegangan dan indikasi pencurian kabel pada gardu distribusi. Proses pembuatan program tersebut diawali dengan membuat inisialisasi untuk wifi dan bot telegram yang ditunjukan pada Gambar 4.20.



Gambar 4. 20 Inisialisai Library Wifi dan Bot Telegram

Pada bagian ini, dilakukan proses inisialisasi koneksi antara ESP32, jaringan WiFi, dan Telegram Bot menggunakan library *WiFi.h* dan *CTBot.h*. Tujuannya adalah agar sistem dapat mengirimkan notifikasi secara otomatis ke pengguna melalui aplikasi Telegram ketika terjadi kondisi tertentu. Setelah melakukan inisialisasi library wifi dan bot telegram yang digunakan, maka dilanjutkan dengan mendefinisikan beberapa parameter penting yang dibutuhkan untuk menghubungkan ESP32 dengan jaringan internet dan Bot Telegram yang ditunjukan pada Gambar 4.21.

```
CTBot myBot;
String ssid = "BY.U";
String pass = "tugasakhiroke";
String token = "7872563938:AAGZwMojFhi8wm1cb6K_TwMyR9YTRwIaCDY";
const int64_t chat_id = 6183217275;
```

Gambar 4. 21 Kode Variabel Wifi dan Bot Telegram

Pada bagian program ini, beberapa variabel penting dideklarasikan untuk mengatur koneksi antara ESP32, jaringan WiFi, dan Bot Telegram. Objek myBot dari kelas CTBot dibuat sebagai penghubung antara ESP32 dan layanan Telegram melalui library CTBot. Selanjutnya, variabel ssid untuk menyimpan nama jaringan WiFi yang akan digunakan, yaitu "BY.U", sedangkan pass untuk menyimpan kata sandi jaringan tersebut, yaitu "tugasakhiroke". Kedua variabel ini diperlukan agar ESP32 dapat terhubung ke internet melalui jaringan lokal.

Kemudian, variabel token berisi token API Bot Telegram yang berupa string unik yang didapatkan dari BotFather yang sudah dijelaskan sebelumnya. Token ini digunakan sebagai identitas autentikasi bot agar ESP32 dapat berkomunikasi dengan server Telegram. Terakhir, chat_id adalah ID numerik yang menunjukkan tujuan pengiriman pesan, yaitu akun Telegram yang ditentukan. Dengan mengatur kelima elemen ini, program mempersiapkan perangkat untuk dapat mengirim notifikasi secara otomatis ke Telegram saat kondisi tertentu terdeteksi. Langkah selanjutnya adalah menginisialisasi koneksi antara ESP32 dengan jaringan WiFi serta Bot Telegram. Proses ini penting agar perangkat dapat mengirim notifikasi secara daring melalui platform Telegram.

```
// Loop menunggu koneksi WiFi berhasil
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");  // Menampilkan titik setiap 500ms sebagai indikasi proses koneksi
}
Serial.println("\nWiFi Terhubung!");  // Tanda koneksi WiFi berhasil

myBot.wifiConnect(ssid, pass);   // Menghubungkan CTBot dengan WiFi
myBot.setTelegramToken(token);   // Mengatur token Telegram Bot
if (myBot.testConnection()) {    // Tes koneksi ke\ Telegram Bot
  Serial.println("Terhubung ke Telegram Bot!");
// Kirim notifikasi bahwa sistem siap digunakan
myBot.sendMessage(chat_id, "✅ SISTEM TERHUBUNG KE WIFI & TELEGRAM BOT \n🟢 ALAT SIAP DIGUNAKAN");
} else {
  Serial.println("Gagal terhubung ke Telegram Bot!");
}
```

Gambar 4. 22 Kode untuk Proses Koneksi Wifi dan Bot Telegram

Bagian program yang ditunjukan pada Gambar 4.22 merupakan proses inisialisasi dan verifikasi koneksi ESP32 ke jaringan WiFi dan Bot Telegram. Pertama, program menggunakan while (WiFi.status() != WL_CONNECTED) untuk menunggu hingga koneksi WiFi berhasil. Selama koneksi belum terhubung, program akan menunda selama 500 milidetik (delay(500)) dan menampilkan titik (.) ke Serial Monitor sebagai indikator proses koneksi yang sedang berlangsung. Setelah koneksi berhasil, ditandai dengan status WL_CONNECTED, program mencetak pesan "WiFi Terhubung!" ke Serial Monitor.

Setelah ESP32 berhasil terhubung ke WiFi, langkah selanjutnya adalah menghubungkan perangkat dengan Bot Telegram. Hal ini dilakukan melalui myBot.wifiConnect(ssid, pass) yang menyambungkan CTBot ke jaringan WiFi menggunakan SSID dan password yang telah ditentukan sebelumnya. Selanjutnya, myBot.setTelegramToken(token) digunakan untuk mengatur token autentikasi dari Bot Telegram agar ESP32 dapat dikenali oleh server Telegram.

Setelah konfigurasi selesai, program melakukan uji koneksi ke Bot Telegram dengan myBot.testConnection(). Jika koneksi berhasil, akan muncul pesan di Serial Monitor: "Terhubung ke Telegram Bot!", dan sistem akan langsung mengirim pesan ke akun Telegram melalui myBot.sendMessage() yang berisi notifikasi "✅ SISTEM TERHUBUNG KE WIFI & TELEGRAM BOT \n🟢 ALAT SIAP DIGUNAKAN" sebagai tanda bahwa alat telah siap digunakan. Sebaliknya, jika koneksi ke bot gagal, maka akan ditampilkan pesan "Gagal terhubung ke Telegram Bot!" ke Serial Monitor. Proses ini penting untuk memastikan bahwa ESP32 siap berkomunikasi secara real-time melalui jaringan Telegram dalam pengoperasian sistem.

4.4.2.2 Pembuatan Program Deklarasi Variabel Global

Pada bagian ini, dilakukan proses deklarasi variabel-variabel global yang digunakan sebagai penyimpan status, flag kontrol, nilai waktu, serta kondisi sistem secara menyeluruh. Variabel global dideklarasikan di bagian awal program agar dapat diakses oleh semua fungsi atau blok kode dalam program ESP32, tanpa perlu mendeklarasikan ulang di dalam setiap fungsi. Dengan adanya deklarasi variabel

global, program dapat berjalan lebih efisien dan terstruktur, karena setiap bagian sistem dapat saling berbagi informasi status tanpa perlu mengulangi pembacaan atau pemrosesan yang sama.

```
// ================================
// ====DEKLARASI VARIABEL GLOBAL=====
// ================================

//PEMBACAAN PERTAMA KALI KETIKA ALAT BARU DI AKTIFKAN
bool sudahTerbacaAwal = false;      //mengecek apakah tegangan sudah terbaca untuk pertama kali
bool sudahTampilPesanAwal = false;  //Tambahan untuk kontrol tampilan pesan awal
```

Gambar 4. 23 Kode untuk Deklarasi Proses Awal

Bagian program yang ditunjukan pada Gambar 4.23 berfungsi untuk mengatur proses saat awal sistem dinyalakan (booting). Variabel *sudahTerbacaAwal* digunakan untuk memastikan bahwa pembacaan tegangan dari sensor (PZEM-004T) dilakukan minimal satu kali ketika alat pertama kali aktif. Nilai awalnya false, dan akan diubah menjadi true setelah pembacaan pertama selesai dilakukan. Tujuannya adalah agar sistem tidak langsung memproses status tegangan atau membuat keputusan sebelum pembacaan pertama selesai. Variabel *sudahTampilPesanAwal* berfungsi untuk menampilkan pesan awal hanya satu kali di Serial Monitor sebagai penanda bahwa sistem telah berhasil aktif dan siap digunakan. Jika tidak dikontrol dengan variabel ini, pesan awal bisa muncul berulang kali.

```
//MODE 1 = DETEKSI PENCURIAN KABEL POWER (FASA R,S,T)
bool hilangR = false, hilangS = false, hilangT = false; //Status kehilangan tegangan untuk masing-masing fasa (true
bool pirAktif = false;                                  //Status apakah sensor PIR sedang aktif untuk deteksi pencur
unsigned long waktuMulaiPIR = 0;                        //Waktu mulai PIR aktif, untuk menghitung durasi aktifnya PI
bool sudahAktifkanPIR_R = false, sudahAktifkanPIR_S = false, sudahAktifkanPIR_T = false;  // Menandakan apakah PIR s
bool sudahKirimNotifikasiPower = false; // Flag untuk menandakan apakah notifikasi kehilangan tegangan sudah dikirim
//fungsi untuk kirim notifikasi ke telegram satu kali ketika hilang fasa satu per satu
bool sudahKirimNotifR = false; // Flag untuk menghindari pengiriman notifikasi kehilangan fasa R berulang-ulang
bool sudahKirimNotifS = false; // Flag untuk menghindari pengiriman notifikasi kehilangan fasa S berulang-ulang
bool sudahKirimNotifT = false; // Flag untuk menghindari pengiriman notifikasi kehilangan fasa T berulang-ulang
bool sudahSiklusPIR = false;   // Status sudah menjalankan siklus PIR mode 1
```

Gambar 4. 24 Kode untuk Deklarasi Variabel Global Mode 1

Bagian program yang ditunjukan pada Gambar 4.24 adalah deklarasi variabel global yang digunakan untuk mendukung Mode 1, yaitu mode deteksi pencurian kabel power (fase R, S, dan T). Berikut penjelasan programnya:

- hilangR, hilangS, hilangT:
  Ketiga variabel ini berfungsi untuk menyimpan status tegangan pada masing-masing fasa. Nilai true berarti fasa tersebut terdeteksi hilang (tidak ada tegangan), sedangkan false berarti fasa dalam kondisi normal.
- pirAktif:
  Menyimpan status apakah sensor gerak PIR sedang aktif atau tidak. Ketika fasa hilang, PIR akan diaktifkan sementara selama 10 detik untuk mendeteksi gerakan di area gardu.
- waktuMulaiPIR:
  Menyimpan waktu (dalam milidetik) saat PIR pertama kali diaktifkan. Data ini digunakan untuk menghitung apakah waktu aktif PIR sudah mencapai batas durasi (misalnya 10 detik) agar bisa dinonaktifkan kembali.
- sudahAktifkanPIR_R, sudahAktifkanPIR_S, sudahAktifkanPIR_T:
  Ketiga variabel ini menandakan apakah PIR sudah pernah diaktifkan untuk masing-masing fasa. Ini mencegah agar PIR tidak terus diaktifkan ulang jika fasa masih hilang.
- sudahKirimNotifikasiPower:
  Menandakan apakah sistem sudah mengirim notifikasi bahwa telah terjadi kehilangan tegangan (pencurian kabel power). Digunakan sebagai kontrol agar tidak mengirim notifikasi berulang kali dalam satu siklus kehilangan.
- sudahKirimNotifR, sudahKirimNotifS, sudahKirimNotifT:
  Masing-masing berfungsi sebagai pengunci notifikasi per fasa. Notifikasi hanya dikirim satu kali untuk masing-masing fasa yang hilang, sehingga tidak muncul berulang selama fasa tersebut belum kembali normal.
- sudahSiklusPIR:
  Menyatakan bahwa siklus aktivasi PIR (aktif selama 10 detik, lalu nonaktif) sudah selesai dijalankan untuk kejadian kehilangan kabel power. Variabel ini digunakan untuk mengatur agar siklus tidak diulang dalam periode yang sama.

```
//MODE 2 = DETEKSI PENCURIAN KABEL NETRAL (TEGANGAN ABNORMAL)
bool sudahKirimInfoAbnormal = false;           // Flag agar notifikasi tegangan abnormal hanya dikirim sekali
const unsigned long waktuTungguAbnormal = 5000; // Waktu tunggu 5 detik untuk pengamatan tegangan abnormal
unsigned long waktuMulaiGabungan = 0;          // Waktu mulai pengamatan gabungan tegangan abnormal
bool dalamPengamatanGabungan = false;          // Status apakah sedang dalam pengamatan gabungan
bool abnormalR = false;                        // Status tegangan fasa R dianggap abnormal
bool abnormalS = false;                        // Status tegangan fasa S dianggap abnormal
bool abnormalT = false;                        // Status tegangan fasa T dianggap abnormal
// Variabel untuk PIR pada mode 2
bool pirAktifNetral = false;            // Status apakah sensor PIR sedang aktif untuk deteksi pencurian kabel netral
unsigned long waktuMulaiPIRNetral = 0;  // Waktu mulai PIR aktif, untuk menghitung durasi aktifnya PIR mode 2
bool sudahKirimNotifikasiNetral = false; // Flag untuk menandakan apakah notifikasi pencurian kabel netral sudah dikirim (untuk mode 2)
bool sudahSiklusPIRMode2 = false;       // Status sudah menjalankan siklus PIR mode 2
```

Gambar 4. 25 Kode untuk Deklarasi Variabel Global Mode 2

Bagian program yang ditunjukan pada Gambar 4.25 adalah deklarasi variabel global yang digunakan untuk Mode 2, yaitu mode deteksi pencurian kabel netral, yang ditandai dengan kondisi tegangan abnormal pada ketiga fasa (R, S, dan T). Berikut penjelasan programnya:

- sudahKirimInfoAbnormal:

  Digunakan sebagai penanda agar notifikasi kondisi tegangan abnormal hanya dikirim satu kali selama satu kejadian. Mencegah pengiriman notifikasi berulang.

- waktuTungguAbnormal:

  Merupakan nilai tetap (konstanta) yaitu 5000 milidetik (5 detik), yang digunakan sebagai durasi tunggu untuk mengamati keadaan ketidakseimbangan tegangan yang benar-benar terjadi, bukan karena fluktuasi tegangan sementara.

- waktuMulaiGabungan:

  Menyimpan waktu awal dimulainya pengamatan terhadap tegangan abnormal. Ini penting untuk menghitung apakah waktu tunggu sudah lewat agar bisa melanjutkan ke langkah berikutnya (aktivasi PIR).

- dalamPengamatanGabungan:

  Menandakan status bahwa sistem sedang dalam proses menunggu (selama 5 detik) untuk mengamati apakah kondisi ketidakseimbangan tegangan pada ketiga fasa memenuhi syarat sebagai pencurian kabel netral.

- abnormalR, abnormalS, abnormalT:

  Masing-masing menyimpan status apakah tegangan pada fasa R, S, dan T terdeteksi abnormal (misalnya terlalu rendah atau terlalu tinggi). Nilai true berarti fasa tersebut dianggap tidak normal.

- pirAktifNetral:

  Menyimpan status apakah sensor gerak (PIR) sedang diaktifkan dalam Mode 2, yaitu untuk memantau adanya gerakan mencurigakan saat terjadi tegangan abnormal.
- waktuMulaiPIRNetral:

  Mencatat waktu saat PIR Mode 2 mulai aktif. Berguna untuk menghitung durasi agar PIR hanya aktif selama 10 detik.
- sudahKirimNotifikasiNetral:

  Digunakan sebagai flag kontrol agar notifikasi pencurian kabel netral hanya dikirim satu kali dalam satu siklus deteksi.
- sudahSiklusPIRMode2:

  Menandakan bahwa sistem sudah menjalankan satu siklus penuh PIR Mode 2, termasuk aktivasi, pemantauan gerakan, dan pengiriman notifikasi (jika diperlukan). Ini mencegah pengulangan proses dalam satu kejadian.

```
// DEKLARASI VARIABEL UNTUK MODE PEMADAMAN LISTRIK
unsigned long waktuHilangR = 0;  // Menyimpan waktu saat fasa R pertama kali terdeteksi hilang
unsigned long waktuHilangS = 0;  // Menyimpan waktu saat fasa S pertama kali terdeteksi hilang
unsigned long waktuHilangT = 0;  // Menyimpan waktu saat fasa T pertama kali terdeteksi hilang
bool sudahKirimNotifikasiPadam = false; // Flag untuk menandai apakah notifikasi pemadaman sudah dikirim
```

Gambar 4. 26 Kode untuk Deklarasi Variabel Global Pemadaman

Bagian program yang ditunjukan pada Gambar 4.26 adalah program yang digunakan untuk mencatat kapan pertama kali tegangan pada masing-masing fasa hilang. Dengan mengetahui urutan waktu kehilangan tegangan, sistem dapat menentukan apakah ketiga fasa hilang dalam waktu yang hampir bersamaan yang dapat mengindikasikan pemadaman listrik. Berikut penjelasan lebih lanjutnya:

- waktuHilangR

  Menyimpan waktu (dalam milidetik) ketika tegangan pada fasa R pertama kali terdeteksi hilang. Waktu ini dibandingkan dengan waktu hilangnya fasa lain untuk menentukan jenis gangguan.
- waktuHilangS

  Menyimpan waktu saat fasa S kehilangan tegangan.

- waktuHilangT

  Menyimpan waktu saat fasa T kehilangan tegangan.
- sudahKirimNotifikasiPadam

  Merupakan flag atau penanda agar notifikasi pemadaman listrik hanya dikirim satu kali saat kondisi pemadaman terdeteksi. Hal ini untuk menghindari pengiriman pesan yang berulang-ulang dalam satu kejadian.

4.4.2.3 Pembuatan Program Sensor PIR HC-SR501

Pada bagian ini, program dirancang untuk mengintegrasikan sensor PIR (Passive Infrared Receiver) HC-SR501 dengan mikrokontroler ESP32. Sensor PIR digunakan untuk mendeteksi pergerakan manusia berdasarkan perubahan radiasi inframerah yang dipancarkan oleh tubuh. Program ini bertujuan untuk mengaktifkan sensor PIR pada kondisi-kondisi tertentu.

Ketika PIR mendeteksi gerakan dalam jangka waktu yang telah ditentukan, maka sistem akan mengirimkan notifikasi otomatis ke Telegram sebagai indikasi adanya aktivitas mencurigakan atau potensi pencurian kabel. Program juga mengatur logika agar sensor PIR hanya aktif dalam periode tertentu, serta memastikan agar notifikasi tidak dikirim secara berulang. Secara umum, bagian ini merupakan penghubung antara data kondisi kelistrikan dengan aktivitas fisik di lapangan, sehingga sistem dapat merespons situasi darurat dengan lebih akurat dan cepat.

```
#define PIR_PIN 13
```

Gambar 4. 27 Kode untuk mendefinisikan Pin sensor PIR

Bagian program yang ditunjukan pada Gambar 4.27 menunjukan pendefinisian bahwa pin GPIO 13 pada mikrokontroler ESP32 akan digunakan sebagai pin input untuk sensor PIR HC-SR501. Sensor PIR ini berfungsi untuk mendeteksi adanya gerakan fisik di sekitar PHB-TR. Dengan mendefinisikan PIR_PIN di awal program, penulisan kode selanjutnya menjadi lebih mudah dan rapi, karena cukup menggunakan nama PIR_PIN setiap kali ingin mengakses pin

tersebut, tanpa harus menulis angka 13 secara langsung. Pendeklarasian ini juga memudahkan pengubahan pin jika diperlukan, karena cukup mengganti nilainya di satu tempat saja.

```
pinMode(PIR_PIN, INPUT);
```

Gambar 4. 28 Kode untuk mengatur PIR sebagai input digital

Bagian program yang ditunjukan pada Gambar 4.28 berfungsi untuk mengatur pin PIR_PIN (dalam hal ini adalah GPIO 13) sebagai input digital, sehingga ESP32 dapat membaca sinyal dari sensor PIR HC-SR501. Dalam mode INPUT, pin ESP32 hanya akan membaca logika TINGGI (HIGH) atau RENDAH (LOW) berdasarkan sinyal yang diberikan oleh sensor. Sensor PIR akan mengeluarkan sinyal HIGH saat mendeteksi gerakan, dan LOW saat tidak ada gerakan. Secara umum, baris program ini adalah bagian penting untuk memulai komunikasi antara ESP32 dan sensor PIR, agar ESP32 dapat mengetahui kapan terjadi gerakan di sekitar sensor.

4.4.2.4 Pembuatan Program Sensor PZEM-004T-V3.0

Bagian ini menjelaskan proses pembuatan program yang digunakan untuk membaca tegangan listrik dari setiap fasa (R, S, dan T) menggunakan sensor PZEM-004T V3.0. Sensor ini merupakan modul yang dirancang untuk memantau parameter listrik seperti tegangan, arus, daya, dan energi secara digital. Dalam konteks sistem deteksi pencurian kabel, PZEM difokuskan untuk memantau tegangan pada masing-masing kabel output transformator yang menuju panel hubung bagi tegangan rendah (PHB-TR). Data tegangan dari ketiga fasa ini sangat penting karena menjadi acuan utama dalam memonitoring tegangan.

```
#include <PZEM004Tv30.h>
```

Gambar 4. 29 Inisialisai Library sensor PZEM-004T-V3.0

Baris program yang ditunjukan pada Gambar 4.29 merupakan instruksi untuk menyertakan library PZEM004Tv30 ke dalam program ESP32. Library ini sangat penting karena menyediakan fungsi-fungsi siap pakai yang memudahkan komunikasi dan pengambilan data dari modul sensor PZEM-004T V3.0. Singkatnya, baris ini adalah langkah awal untuk mengaktifkan fungsi-fungsi komunikasi antara ESP32 dengan sensor PZEM-004T V3.0, sehingga sistem dapat membaca parameter kelistrikan secara akurat dan efisien.

```
#define RX_PIN 16   // Pin RX untuk komunikasi UART dengan PZEM
#define TX_PIN 17   // Pin TX untuk komunikasi UART dengan PZEM
```

Gambar 4. 30 Kode untuk mendefinisikan pin RX TX

Program yang ditunjukan pada Gambar 4.30 berfungsi untuk mendefinisikan pin yang akan digunakan oleh ESP32 dalam komunikasi serial (UART) dengan sensor PZEM-004T V3.0. Pada sistem ini, RX_PIN diatur ke GPIO 16 dan TX_PIN ke GPIO 17 pada ESP32. RX_PIN (Receive Pin) merupakan jalur tempat ESP32 menerima data dari sensor PZEM. Sedangkan TX_PIN (Transmit Pin) merupakan jalur tempat ESP32 mengirim data atau perintah ke sensor PZEM.

```
HardwareSerial pzemSerial(1);
```

Gambar 4. 31 Kode untuk komunikasi serial PZEM

Program yang ditunjukan pada Gambar 4.31 berfungsi untuk membuat objek komunikasi serial tambahan (UART1) pada ESP32 yang akan digunakan secara khusus untuk berkomunikasi dengan sensor PZEM-004T V3.0. Secara default, ESP32 memiliki lebih dari satu port UART, dan baris ini menggunakan port UART1 dengan cara membuat objek bernama pzemSerial. Objek ini nantinya akan digunakan untuk mengatur jalur komunikasi antara ESP32 dan sensor PZEM melalui pin RX dan TX yang telah ditentukan sebelumnya (GPIO 16 dan 17).

Dengan kata lain, baris program ini memberi tahu ESP32 bahwa komunikasi data dengan sensor PZEM akan dilakukan melalui port serial hardware ke-1, bukan port utama yang biasanya digunakan untuk komunikasi dengan komputer melalui

USB. Hal ini penting agar komunikasi dengan sensor dapat berjalan secara terpisah dan tidak mengganggu jalur komunikasi lainnya.

```
PZEM004Tv30 pzemR(pzemSerial, RX_PIN, TX_PIN, 0x01);
PZEM004Tv30 pzemS(pzemSerial, RX_PIN, TX_PIN, 0x02);
PZEM004Tv30 pzemT(pzemSerial, RX_PIN, TX_PIN, 0x03);
```

Gambar 4. 32 Kode untuk membuat objek sensor PZEM

Program yang ditunjukan pada Gambar 4.32 digunakan untuk membuat tiga objek sensor PZEM-004T V3.0 yang masing-masing mewakili pengukuran tegangan pada fasa R, S, dan T. Setiap objek menggunakan jalur komunikasi serial UART1 yang telah didefinisikan sebelumnya (pzemSerial), serta pin RX dan TX yang terhubung ke sensor. Masing-masing sensor PZEM diberikan alamat unik, 0x01 untuk fasa R, 0x02 untuk fasa S, dan 0x03 untuk fasa T.

Alamat ini penting karena ketiga sensor terhubung pada jalur komunikasi serial yang sama, sehingga ESP32 dapat membedakan dan berkomunikasi secara spesifik dengan setiap sensor berdasarkan alamatnya. Dengan pendekatan ini, ESP32 mampu membaca tegangan dari ketiga fasa listrik secara terpisah melalui satu jalur UART.

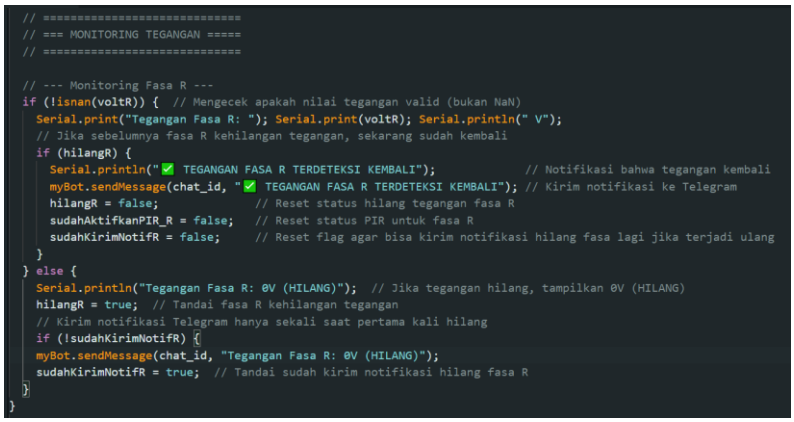
```
void loop() {
  // Pembacaan Tegangan 3 Fasa
  float voltR = pzemR.voltage();
  float voltS = pzemS.voltage();
  float voltT = pzemT.voltage();
```

Gambar 4. 33 Kode untuk mengambil data tegangan

Program yang ditunjukan pada Gambar 4.33 digunakan untuk mengambil data tegangan listrik dari ketiga sensor PZEM-004T V3.0 yang sebelumnya telah didefinisikan sebagai pzemR, pzemS, dan pzemT. Masing-masing sensor mengukur tegangan dari satu fasa listrik, yaitu fasa R, S, dan T. Nilai tegangan yang dibaca dari tiap sensor disimpan ke dalam variabel bertipe float, voltR menyimpan nilai tegangan dari fasa R, voltS menyimpan nilai tegangan dari fasa S, dan voltT menyimpan nilai tegangan dari fasa T. Dengan nilai-nilai ini, program dapat

melakukan analisis selanjutnya seperti mendeteksi kehilangan tegangan, tegangan abnormal, maupun indikasi pencurian kabel listrik berdasarkan perubahan tegangan pada setiap fasa.

1. Program Monitoring Tegangan



```
// ============================
// === MONITORING TEGANGAN =====
// ============================

// --- Monitoring Fasa R ---
if (!isnan(voltR)) {  // Mengecek apakah nilai tegangan valid (bukan NaN)
  Serial.print("Tegangan Fasa R: "); Serial.print(voltR); Serial.println(" V");
  // Jika sebelumnya fasa R kehilangan tegangan, sekarang sudah kembali
  if (hilangR) {
    Serial.println("✅ TEGANGAN FASA R TERDETEKSI KEMBALI");            // Notifikasi bahwa tegangan kembali
    myBot.sendMessage(chat_id, "✅ TEGANGAN FASA R TERDETEKSI KEMBALI"); // Kirim notifikasi ke Telegram
    hilangR = false;              // Reset status hilang tegangan fasa R
    sudahAktifkanPIR_R = false;   // Reset status PIR untuk fasa R
    sudahKirimNotifR = false;     // Reset flag agar bisa kirim notifikasi hilang fasa lagi jika terjadi ulang
  }
} else {
  Serial.println("Tegangan Fasa R: 0V (HILANG)");  // Jika tegangan hilang, tampilkan 0V (HILANG)
  hilangR = true;  // Tandai fasa R kehilangan tegangan
  // Kirim notifikasi Telegram hanya sekali saat pertama kali hilang
  if (!sudahKirimNotifR) {
  myBot.sendMessage(chat_id, "Tegangan Fasa R: 0V (HILANG)");
  sudahKirimNotifR = true;  // Tandai sudah kirim notifikasi hilang fasa R
  }
}
```

Gambar 4. 34 Kode untuk monitoring tegangan

Program yang ditunjukan pada Gambar 4.34 merupakan proses monitoring tegangan untuk fasa R yang dilakukan secara berkala oleh ESP32. Fungsinya adalah untuk mendeteksi apakah tegangan pada fasa R masih normal atau mengalami kehilangan (0 Volt), lalu menindaklanjuti kondisi tersebut dengan menampilkan informasi pada Serial Monitor dan mengirimkan notifikasi ke Telegram. Berikut rincian Penjelasan program untuk melakukan monitoring tegangan fasa R.

- Pengecekan Validitas Data Tegangan (!isnan(voltR))

  Program terlebih dahulu memeriksa apakah nilai tegangan yang dibaca dari sensor adalah angka yang valid (bukan NaN atau Not a Number). Jika valid, maka tegangan akan ditampilkan di Serial Monitor.

- Kondisi Tegangan Kembali Normal

  Jika sebelumnya fasa R dinyatakan hilang (variabel hilangR bernilai true), dan sekarang tegangan kembali terbaca, maka:

  - Ditampilkan pesan bahwa tegangan fasa R telah kembali normal.
  - Dikirim notifikasi ke akun Telegram yang dituju bahwa fasa R sudah kembali.

- Status kehilangan tegangan (hilangR) di-reset menjadi false. Hal ini menandakan bahwa tegangan fasa R sudah kembali normal.
- Status aktivasi sensor gerak PIR untuk fasa R di-reset (sudahAktifkanPIR_R = false) yang menandakan bahwa sensor PIR untuk fasa R belum aktif lagi. Ini penting jika nanti diperlukan logika aktivasi ulang saat tegangan hilang lagi.
- Flag sudahKirimNotifR di-reset agar notifikasi dapat dikirim lagi jika terjadi kehilangan fasa di lain waktu.

- Kondisi Tegangan Hilang

  Jika tegangan tidak valid (artinya sensor tidak membaca tegangan apapun), maka dianggap sebagai kondisi tegangan fasa R hilang (0 Volt). Dalam hal ini:
  - Serial Monitor akan menampilkan informasi "Tegangan Fasa R: 0V (HILANG)".
  - Variabel hilangR diatur menjadi true sebagai penanda kehilangan tegangan.
  - if (!sudahKirimNotifR) Jika notifikasi kehilangan tegangan belum dikirim sebelumnya, maka lanjut ke proses pengiriman.
  - Jika ini adalah pertama kalinya fasa R hilang dalam siklus saat ini, maka dikirimkan notifikasi ke Telegram.
  - flag sudahKirimNotifR = true; diatur agar notifikasi tidak terkirim berulang kali.

2. Program Pemadaman Listrik

Pemadaman listrik dalam konteks sistem deteksi ini merujuk pada kondisi di mana ketiga fasa listrik R, S, dan T mengalami kehilangan tegangan secara serentak atau hampir bersamaan. Kejadian ini bisa disebabkan oleh gangguan jaringan dari pihak penyedia listrik (PLN) saat ada pemeliharaan sistem atau terjadi gangguan.

```
// ============================================================
// === MODE PEMADAMAN LISTRIK (TEGANGAN 3 FASA HILANG SERENTAK) ===
// ============================================================

// Catat waktu saat masing-masing fasa hilang, hanya sekali (jika belum tercatat)
if (hilangR && waktuHilangR == 0) waktuHilangR = millis();  // Jika fasa R hilang dan belum dicatat, simpan waktunya
if (hilangS && waktuHilangS == 0) waktuHilangS = millis();  // Jika fasa S hilang dan belum dicatat, simpan waktunya
if (hilangT && waktuHilangT == 0) waktuHilangT = millis();  // Jika fasa T hilang dan belum dicatat, simpan waktunya
```

Gambar 4. 35 Kode untuk mencatat waktu kehilangan tegangan

Program yang ditunjukan pada Gambar 4.35 merupakan awal dari logika Mode Pemadaman Listrik, yaitu kondisi ketika ketiga fasa (R, S, dan T) mengalami kehilangan tegangan secara bersamaan. Jika kondisi hilangR (status fasa R hilang) bernilai true dan variabel waktuHilangR masih nol (belum pernah dicatat sebelumnya), maka program akan menyimpan waktu saat kejadian tersebut menggunakan fungsi millis(), yaitu waktu dalam milidetik sejak ESP32 menyala. Logika yang sama diterapkan untuk fasa S dan fasa T. Jika tegangan fasa S dan T hilang dan belum tercatat sebelumnya, maka waktu kehilangan dicatat. Dengan mencatat waktu hilangnya tegangan dari masing-masing fasa, sistem bisa menganalisis selisih waktu antar kehilangan untuk menentukan apakah kejadian tersebut merupakan pemadaman total (serentak), atau terjadi secara bertahap (kemungkinan pencurian kabel fasa satu per satu).

```
// Jika ketiga fasa hilang DAN notifikasi belum pernah dikirim
if (hilangR && hilangS && hilangT && !sudahKirimNotifikasiPadam) {
// Ambil waktu hilangnya masing-masing fasa
unsigned long tR = waktuHilangR;
unsigned long tS = waktuHilangS;
unsigned long tT = waktuHilangT;

// Cari waktu paling awal dan paling akhir dari ketiga fasa
unsigned long maxT = max(tR, max(tS, tT)); // waktu hilang paling akhir
unsigned long minT = min(tR, min(tS, tT)); // waktu hilang paling awal

// Jika selisih waktu antara fasa paling cepat dan paling lambat <= 2 detik (2000 ms)
if (maxT - minT <= 2000) {
  // Maka dianggap PEMADAMAN LISTRIK (hilang serentak)
  String pesanPadam = "TEGANGAN 3 FASA HILANG SERENTAK\n ⚠ INDIKASI PEMADAMAN LISTRIK ⚠";
  Serial.println(pesanPadam);             // Tampilkan di Serial Monitor
  myBot.sendMessage(chat_id, pesanPadam);  // Kirim pesan ke Telegram
  sudahKirimNotifikasiPadam = true;        // Tandai bahwa notifikasi sudah dikirim agar tidak berulang
  }
}
```

Gambar 4. 36 Kode untuk mengidentifikasi kehilangan tegangan bersama

Program yang ditunjukan pada Gambar 4.36 merupakan lanjutan dari logika Mode Pemadaman Listrik, yang bertujuan untuk mengidentifikasi apakah

kehilangan tegangan pada ketiga fasa (R, S, dan T) terjadi secara serentak, sebagai indikator bahwa terjadi pemadaman listrik dan bukan pencurian kabel satu per satu.

- Pemeriksaan kondisi kehilangan tegangan pada semua fasa

```
// Jika ketiga fasa hilang DAN notifikasi belum pernah dikirim
if (hilangR && hilangS && hilangT && !sudahKirimNotifikasiPadam) {
```

Gambar 4. 37 Kode untuk memeriksa kehilangan tegangan bersamaan

Baris ini memeriksa apakah ketiga fasa (R, S, dan T) semuanya dalam kondisi hilang tegangan (bernilai true) dan notifikasi pemadaman belum pernah dikirim sebelumnya (!sudahKirimNotifikasiPadam). Ini untuk memastikan bahwa sistem hanya menjalankan proses ini satu kali saat kondisi pemadaman terjadi.

- Menyimpan waktu kehilangan masing-masing fasa ke dalam variabel sementara

```
unsigned long tR = waktuHilangR;
unsigned long tS = waktuHilangS;
unsigned long tT = waktuHilangT;
```

Gambar 4. 38 Kode untuk menyimpan waktu kehilangan tegangan

Waktu kehilangan tegangan yang sebelumnya sudah dicatat saat tiap fasa hilang, disimpan ke dalam variabel lokal tR, tS, dan tT agar lebih mudah dianalisis.

- Menentukan waktu paling awal dan paling akhir dari ketiga fasa

```
// Cari waktu paling awal dan paling akhir dari ketiga fasa
unsigned long maxT = max(tR, max(tS, tT)); // waktu hilang paling akhir
unsigned long minT = min(tR, min(tS, tT)); // waktu hilang paling awal
```

Gambar 4. 39 Kode untuk menghitung waktu kehilangan tegangan

Baris ini mencari:

minT: waktu fasa yang paling dahulu mengalami kehilangan tegangan.

maxT: waktu fasa yang paling akhir mengalami kehilangan tegangan.

Dengan membandingkan waktu ini, sistem dapat mengetahui selisih waktu antara ketiganya.

- Evaluasi apakah kehilangan terjadi serentak (<= 2 detik)

```
// Jika selisih waktu antara fasa paling cepat dan paling lambat <= 2 detik (2000 ms)
if (maxT - minT <= 2000) {
```

Gambar 4. 40 Kode untuk menentukan durasi serentak

Jika selisih antara waktu kehilangan paling cepat dan paling lambat kurang dari atau sama dengan 2000 milidetik (2 detik), maka dianggap hilang serentak**.** Batas waktu ini menjadi tolok ukur bahwa kejadian tersebut bukan pencurian bertahap, melainkan kemungkinan besar pemadaman.

- Tindakan jika pemadaman terdeteksi

```
String pesanPadam = "TEGANGAN 3 FASA HILANG SERENTAK\n ⚠ INDIKASI PEMADAMAN LISTRIK ⚠";
Serial.println(pesanPadam);             // Tampilkan di Serial Monitor
myBot.sendMessage(chat_id, pesanPadam);  // Kirim pesan ke Telegram
sudahKirimNotifikasiPadam = true;        // Tandai bahwa notifikasi sudah dikirim agar tidak berulang
```

Gambar 4. 41 Kode untuk menampilkam notifikasi pemadaman

Jika kondisi terpenuhi:

- Sistem akan menampilkan pesan “TEGANGAN 3 FASA HILANG SERENTAK\n ⚠ INDIKASI PEMADAMAN LISTRIK ⚠” ke Serial Monitor.
- Mengirim notifikasi ke Telegram untuk memberi tahu bahwa ada indikasi pemadaman listrik.
- Mengatur status sudahKirimNotifikasiPadam menjadi true agar pesan ini tidak dikirim berulang kali untuk satu kejadian yang sama.

```
// RESET semua status jika ketiga fasa sudah normal kembali
if (!hilangR && !hilangS && !hilangT) {
// Semua fasa sudah kembali menyala
waktuHilangR = 0;                   // Reset waktu hilangnya R
waktuHilangS = 0;                   // Reset waktu hilangnya S
waktuHilangT = 0;                   // Reset waktu hilangnya T
sudahKirimNotifikasiPadam = false;  // Reset flag agar bisa mendeteksi pemadaman selanjutnya
}
```

Gambar 4. 42 Kode untuk mereset keadaan pemadaman

Bagian program ini berfungsi untuk mereset status sistem apabila kondisi tegangan pada ketiga fasa (R, S, dan T) telah kembali normal setelah sebelumnya mengalami kehilangan serentak, seperti saat terjadi pemadaman listrik.

- Pemeriksaan apakah semua fasa sudah normal kembali

  Baris ini memastikan bahwa tegangan pada semua fasa sudah terdeteksi kembali, dengan kata lain hilangR, hilangS, dan hilangT semuanya bernilai false. Kondisi ini menandakan bahwa pemadaman sudah berakhir, atau tegangan sudah kembali menyala di ketiga fasa.
- Reset waktu kehilangan tegangan pada masing-masing fasa

  Setiap fasa memiliki variabel waktu yang digunakan untuk mencatat kapan tegangan mulai hilang. Setelah tegangan kembali, nilai-nilai ini harus dikembalikan ke nol agar sistem siap untuk mendeteksi kejadian hilangnya tegangan berikutnya**.**
- Reset status notifikasi pemadaman

  Setiap fasa memiliki variabel waktu yang digunakan untuk mencatat kapan tegangan mulai hilang. Setelah tegangan kembali, nilai-nilai ini harus dikembalikan ke nol agar sistem siap untuk mendeteksi kejadian hilangnya tegangan berikutnya.

3. Program Pencurian Kabel Fasa

Program pencurian kabel power pada sistem ini dirancang untuk mendeteksi kehilangan tegangan secara individual pada masing-masing fasa R, S, dan T yang mengindikasikan kemungkinan pemutusan atau pencurian kabel power. Setiap fasa dimonitor menggunakan sensor PZEM-004T V3.0 yang mampu membaca tegangan secara real-time.

```
// ======================================================
// === MODE 1: DETEKSI PENCURIAN KABEL POWER (R / S / T) ===
// ======================================================
if (!pirAktif && !sudahKirimNotifikasiPadam) {
  // Jika PIR belum aktif (belum dalam mode deteksi gerakan)
  if ((hilangR && !sudahAktifkanPIR_R) || // Jika fasa R hilang dan PIR belum diaktifkan untuk R
      (hilangS && !sudahAktifkanPIR_S) || // Atau fasa S hilang dan PIR belum diaktifkan untuk S
      (hilangT && !sudahAktifkanPIR_T)) { // Atau fasa T hilang dan PIR belum diaktifkan untuk T
    pirAktif = true;  // Aktifkan PIR
    waktuMulaiPIR = millis();  // Simpan waktu mulai PIR aktif (waktu saat ini)
    pinMode(PIR_PIN, INPUT);   // Set pin PIR sebagai input aktif (mengaktifkan PIR sensor)
    Serial.println("🔥 PIR MODE 1 DI-AKTIFKAN SELAMA 10 DETIK UNTUK SIAGA PENCURIAN KABEL POWER"); // Tampilkan info aktivasi PIR
    if (hilangR) sudahAktifkanPIR_R = true; // Tandai PIR sudah diaktifkan untuk fasa R
    if (hilangS) sudahAktifkanPIR_S = true; // Tandai PIR sudah diaktifkan untuk fasa S
    if (hilangT) sudahAktifkanPIR_T = true; // Tandai PIR sudah diaktifkan untuk fasa T
  }
}
```

Gambar 4. 43 Kode untuk MODE 1 pencurian kabel power

Program yang ditunjukan pada Gambar 4.43 merupakan logika utama untuk mendeteksi kemungkinan pencurian kabel power pada salah satu fasa (R, S, atau T) secara individual. Program pertama-tama memeriksa apakah sensor PIR belum aktif (!pirAktif) dan sistem bukan dalam kondisi pemadaman listrik (!sudahKirimNotifikasiPadam). Selanjutnya, program memeriksa apakah ada salah satu fasa (R, S, atau T) yang kehilangan tegangan dan apakah sensor PIR belum diaktifkan sebelumnya untuk fasa tersebut. Jika kondisi ini terpenuhi, maka:

- Sensor PIR diaktifkan dengan mengatur pin PIR sebagai input (pinMode(PIR_PIN, INPUT)).
- Waktu saat PIR mulai aktif dicatat menggunakan millis() untuk menghitung durasi aktif selama 10 detik ke depan.
- Status aktivasi PIR untuk fasa yang hilang ditandai (sudahAktifkanPIR_R, sudahAktifkanPIR_S, atau sudahAktifkanPIR_T) agar tidak diaktifkan ulang saat kondisi belum pulih.

```
if (pirAktif) {
int pirState = digitalRead(PIR_PIN);
if (pirState == HIGH && millis() - waktuTerakhirNotifikasi >= intervalNotifikasi) {
  String pesan = "🚨 GERAKAN TERDETEKSI DI DEKAT PANEL 🚨\n⚠️ INDIKASI PENCURIAN KABEL FASA ⚠️";
  Serial.println(pesan);
  myBot.sendMessage(chat_id, pesan);
  waktuTerakhirNotifikasi = millis();  // update waktu terakhir kirim notifikasi
}

// Nonaktifkan PIR setelah 10 detik
if (millis() - waktuMulaiPIR >= 10000) {
  pirAktif = false;
  pinMode(PIR_PIN, INPUT_PULLUP);
  Serial.println("🔴 PIR DINONAKTIFKAN SETELAH 10 DETIK");
  sudahSiklusPIR = true;
}
}
```

Gambar 4. 44 Kode untuk aktivasi PIR MODE 1

Program yang ditunjukan pada Gambar 4.44 merupakan kelanjutan dari proses deteksi pencurian kabel power menggunakan sensor PIR (Passive Infrared). Setelah sensor PIR diaktifkan dalam mode siaga, program akan terus memantau keberadaan gerakan selama jangka waktu 10 detik. Pertama, program memeriksa apakah PIR sedang aktif (if (pirAktif)). Jika iya, maka sensor PIR dibaca menggunakan digitalRead(PIR_PIN) dan hasilnya disimpan dalam variabel pirState. Jika nilai pirState adalah HIGH, itu berarti ada gerakan yang terdeteksi oleh sensor. Agar notifikasi tidak terkirim berulang-ulang dalam waktu singkat,

sistem memeriksa juga apakah selang waktu sejak notifikasi terakhir telah lebih dari nilai intervalNotifikasi. Jika kedua kondisi ini terpenuhi, maka sistem akan mengirimkan notifikasi ke Telegram dengan isi pesan yang menunjukkan bahwa ada gerakan terdeteksi di dekat panel, sebagai indikasi pencurian kabel fasa. Waktu pengiriman notifikasi terakhir akan diperbarui (waktuTerakhirNotifikasi = millis()).

Setelah sensor PIR aktif selama 10 detik (if (millis() - waktuMulaiPIR >= 10000)), maka sistem akan menonaktifkan sensor PIR dengan menyetel ulang pin ke mode INPUT_PULLUP. Kemudian mengubah status pirAktif menjadi false untuk menandai bahwa sensor sudah tidak aktif. Sistem juga akan menandai bahwa satu siklus pendeteksian PIR telah selesai dengan sudahSiklusPIR = true.

Fungsi utama dari bagian ini adalah untuk menentukan apakah kehilangan tegangan pada salah satu fasa disebabkan oleh pencurian, yang dibuktikan dengan deteksi gerakan dalam waktu singkat setelah tegangan hilang. Jika tidak ada gerakan yang terdeteksi, maka kemungkinan besar kehilangan tegangan bukan disebabkan oleh pencurian.

4. Program Pencurian Kabel Netral

Bagian program ini menjalankan Mode 2, yaitu pendeteksian pencurian kabel netral berdasarkan ketidakseimbangan atau abnormalitas tegangan pada ketiga fasa (R, S, dan T) seperti yang ditunjukan pada Gambar 4.45.

```
// ============================================
// === MODE 2: DETEKSI PENCURIAN KABEL NETRAL ===
// ============================================


// Cek kondisi abnormal masing-masing fasa
abnormalR = (!isnan(voltR) && voltR != 0 && (voltR < 198.0 || voltR > 240.0)); // Tegangan R di luar batas normal
abnormalS = (!isnan(voltS) && voltS != 0 && (voltS < 198.0 || voltS > 240.0)); // Tegangan S di luar batas normal
abnormalT = (!isnan(voltT) && voltT != 0 && (voltT < 198.0 || voltT > 240.0)); // Tegangan T di luar batas normal

// Mulai pengamatan gabungan jika ada salah satu fasa abnormal dan belum dalam pengamatan dan PIR mode 2 belum aktif
if ((abnormalR || abnormalS || abnormalT) && !dalamPengamatanGabungan && !pirAktifNetral && !sudahSiklusPIRMode2) {
  waktuMulaiGabungan = millis();   // Simpan waktu mulai pengamatan gabungan abnormal
  dalamPengamatanGabungan = true;  // Tandai sedang dalam masa pengamatan gabungan
  sudahKirimInfoAbnormal = false;  // Reset flag agar pesan tegangan abnormal bisa dikirim sekali
}

// Setelah 5 detik pengamatan, jika belum kirim info abnormal, kirim sekarang
if (dalamPengamatanGabungan && !sudahKirimInfoAbnormal && (millis() - waktuMulaiGabungan >= waktuTungguAbnormal)) {
String pesan = "";  // Variabel untuk menyimpan pesan notifikasi abnormal tegangan
if (abnormalR) pesan += "⚠ TEGANGAN FASA R DI LUAR BATAS TOLERANSI (" + String(voltR) + "V)\n"; // Tambah pesan jika R abnormal
if (abnormalS) pesan += "⚠ TEGANGAN FASA S DI LUAR BATAS TOLERANSI (" + String(voltS) + "V)\n"; // Tambah pesan jika S abnormal
if (abnormalT) pesan += "⚠ TEGANGAN FASA T DI LUAR BATAS TOLERANSI (" + String(voltT) + "V)\n"; // Tambah pesan jika T abnormal
```

Gambar 4. 45 Kode untuk MODE 2 pencurian kabel netral

Langkah pertama dalam program adalah mengecek apakah tegangan pada masing-masing fasa masih berada di dalam atau sudah di luar rentang toleransi. Tegangan dianggap abnormal jika:

- Tidak NaN (berarti sensor berhasil membaca).
- Tidak sama dengan nol (berarti masih ada tegangan).
- Nilainya kurang dari 198V atau lebih dari 231V.

Hal ini karena tegangan normal sistem 1 fasa umumnya berkisar sekitar 220V ± batas toleransi. Maka dari itu, dibuat batas 198V–240V sesuai dengan aturan toleransi yang berlaku dan menyesuaikan keadaan tegangan tempat melakukan uji coba yang dimana maksimal tegangannya sebesar 238V. Berdasarkan SPLN No.1:1978, batas toleransi tegangan pelayanan +5% dan -10% dari tegangan nominal. Tegangan nominal yang dimaksud adalah 220volt sehingga standar tersebut berarti tegangan listrik tidak boleh lebih dari 231volt atau kurang dari 198 volt.

Jika terjadi ketidakseimbangan tegangan dan sistem belum sedang melakukan pengamatan, maka sistem memulai pengamatan gabungan, dengan mencatat waktu saat pengamatan dimulai (waktuMulaiGabungan). Variabel dalamPengamatanGabungan diatur menjadi true sebagai penanda bahwa sistem sedang dalam masa pengamatan selama 5 detik. Flag sudahKirimInfoAbnormal diset ulang agar notifikasi bisa dikirim.

Setelah masa pengamatan selama 5 detik, jika kondisi tegangan masih tetap abnormal dan sistem belum mengirimkan pesan, maka program menyusun pesan notifikasi yang berisi fasa mana saja yang memiliki tegangan di luar batas toleransi, lengkap dengan nilai tegangannya. Pesan tersebut ditampilkan ke Serial Monitor, agar bisa dilihat oleh pengguna melalui kabel USB dan dikirim ke Telegram sebagai peringatan otomatis. Setelah itu, sistem menandai bahwa pesan sudah dikirim dengan menyetel sudahKirimInfoAbnormal = true, sehingga tidak dikirim berulang.

```
  // Setelah kirim pesan abnormal, aktifkan PIR mode 2 untuk pengamatan gerakan
  pirAktifNetral = true;          // Set PIR mode 2 aktif
  waktuMulaiPIRNetral = millis(); // Simpan waktu mulai PIR mode 2 aktif
  pinMode(PIR_PIN, INPUT);        // Aktifkan pin PIR sebagai input
  Serial.println("⏱️ PIR MODE 2 DI-AKTIFKAN SELAMA 10 DETIK UNTUK SIAGA PENCURIAN KABEL NETRAL"); // Info aktivasi PIR mode 2
  dalamPengamatanGabungan = false;  // selesai masa tunggu pengamatan
}
```

Gambar 4. 46 Kode untuk aktivasi PIR MODE 2

Program yang ditunjukan pada Gambar 4.46 merupakan program untuk mengaktifkan sensor PIR. Setelah sistem mendeteksi adanya tegangan abnormal pada salah satu atau lebih fasa, dan pesan notifikasi sudah dikirimkan, maka langkah selanjutnya adalah mengaktifkan sensor PIR untuk mengamati gerakan fisik di sekitar lokasi dalam kurun waktu tertentu. Hal ini dilakukan untuk memastikan apakah kondisi tegangan abnormal tersebut memang disebabkan oleh pencurian kabel netral, bukan gangguan teknis biasa. Berikut penjelasan program per baris:

- pirAktifNetral = true;

  Menandai bahwa sensor PIR sedang diaktifkan khusus untuk Mode 2, yaitu untuk mendeteksi pencurian kabel netral. Mode ini berbeda dari aktivasi PIR pada Mode 1 (kabel power R/S/T).
- waktuMulaiPIRNetral = millis();

  Menyimpan waktu saat PIR mulai aktif. Nilai millis() digunakan sebagai acuan waktu sejak ESP32 menyala. Program ini akan digunakan nanti untuk menghitung durasi aktif PIR (dalam sistem ini adalah 10 detik).
- pinMode(PIR_PIN, INPUT);

  Mengatur pin input PIR agar aktif dan siap membaca gerakan. PIR akan mendeteksi apakah ada perubahan sinyal.
- Serial.println("⏱️ PIR MODE 2 DI-AKTIFKAN SELAMA 10 DETIK UNTUK SIAGA PENCURIAN KABEL NETRAL");

  Menampilkan informasi di Serial Monitor bahwa sensor PIR telah diaktifkan selama 10 detik sebagai bagian dari proses pendeteksian pencurian kabel netral.

- dalamPengamatanGabungan = false;

  Menandai bahwa masa pengamatan tegangan abnormal telah selesai. Sistem tidak lagi menunggu dan mulai fokus pada pemantauan gerakan fisik melalui PIR.

```
// Jika PIR MODE 2 aktif
if (pirAktifNetral) {
int pirState = digitalRead(PIR_PIN);
if (pirState == HIGH && millis() - waktuTerakhirNotifikasiNetral >= intervalNotifikasi) {
  String pesanGerakan = "🚨 GERAKAN TERDETEKSI DI DEKAT PANEL 🚨\n⚠️ INDIKASI PENCURIAN KABEL NETRAL ⚠️";
  Serial.println(pesanGerakan);
  myBot.sendMessage(chat_id, pesanGerakan);
  waktuTerakhirNotifikasiNetral = millis();  // update waktu terakhir
}

if (millis() - waktuMulaiPIRNetral >= 10000) {
  pirAktifNetral = false;
  pinMode(PIR_PIN, INPUT_PULLUP);
  Serial.println("🔴 PIR MODE 2 DINONAKTIFKAN SETELAH 10 DETIK");
  sudahSiklusPIRMode2 = true;
 }
}
```

Gambar 4. 47 Kode untuk mengelola sensor PIR MODE 2

Program yang ditunjukan pada Gambar 4.47 merupakan program untuk mengelola aktivitas sensor PIR selama proses deteksi pencurian kabel netral, khususnya setelah PIR diaktifkan dalam Mode 2. Tujuannya adalah untuk menangkap gerakan fisik mencurigakan di sekitar panel gardu distribusi yang bisa mengindikasikan adanya upaya pencurian. Berikut penjelasan lebih rinci dari program tersebut.

- if (pirAktifNetral) {

  Memastikan bahwa blok kode ini hanya dijalankan saat PIR dalam mode 2 (deteksi pencurian kabel netral) sedang aktif.

- int pirState = digitalRead(PIR_PIN);

  Membaca status logika dari pin PIR. Nilai HIGH menunjukkan bahwa gerakan terdeteksi, sedangkan LOW berarti tidak ada gerakan.

- if (pirState == HIGH && millis() - waktuTerakhirNotifikasiNetral >= intervalNotifikasi) {

  Jika terdeteksi gerakan dan sudah cukup waktu sejak notifikasi terakhir (menghindari spam pesan), maka sistem akan menindaklanjuti dengan mengirim peringatan.

- String pesanGerakan = "🚨 GERAKAN TERDETEKSI DI DEKAT PANEL 🚨\n⚠️ INDIKASI PENCURIAN KABEL NETRAL ⚠️";
  Menyusun isi pesan notifikasi yang menjelaskan bahwa ada gerakan terdeteksi, dan kondisi ini dapat menjadi indikasi pencurian kabel netral.
- Serial.println(pesanGerakan);
  Menampilkan pesan tersebut ke Serial Monitor (untuk pemantauan langsung via komputer).
- myBot.sendMessage(chat_id, pesanGerakan);
  Mengirim pesan ke akun Telegram yang sudah terdaftar, agar petugas bisa segera merespons kemungkinan kejadian pencurian.
- waktuTerakhirNotifikasiNetral = millis();
  Menyimpan waktu terakhir notifikasi dikirim agar selanjutnya bisa dilakukan jeda (delay) sebelum mengirim ulang jika masih terdeteksi gerakan.
- if (millis() - waktuMulaiPIRNetral >= 10000) {
  Mengecek apakah waktu 10 detik sejak PIR Mode 2 aktif telah berlalu. Bila sudah berlalu selama 10 detik, maka PIR akan dinonaktifkan.
- pirAktifNetral = false;
  Menandai bahwa PIR mode 2 sudah tidak aktif lagi.
- pinMode(PIR_PIN, INPUT_PULLUP);
  Mengatur ulang pin PIR sebagai input dengan resistor pull-up internal. Ini penting agar pin tidak berada dalam keadaan "mengambang" (floating) setelah PIR dinonaktifkan.
- Serial.println("🛑 PIR MODE 2 DINONAKTIFKAN SETELAH 10 DETIK");
  Menampilkan informasi bahwa sensor PIR mode 2 telah dimatikan setelah 10 detik.
- sudahSiklusPIRMode2 = true;
  Menandai bahwa satu siklus lengkap PIR Mode 2 telah selesai. Hal ini bertujuan untuk mencegah aktivasi ulang PIR secara terus-menerus untuk kasus yang sama.

```
// Reset flag dan kondisi mode 2 jika tegangan semua fasa sudah normal (antara 198 - 231 Volt)
if ((voltR >= 198.0 && voltR <= 231.0) &&
    (voltS >= 198.0 && voltS <= 231.0) &&
    (voltT >= 198.0 && voltT <= 231.0)) {
  sudahKirimInfoAbnormal = false;       // Reset flag sudah kirim info tegangan abnormal
  sudahKirimNotifikasiNetral = false;   // Reset flag sudah kirim notifikasi pencurian kabel netral
  dalamPengamatanGabungan = false;      // Reset status pengamatan gabungan
  pirAktifNetral = false;               // Pastikan PIR mode 2 nonaktif
  sudahSiklusPIRMode2 = false;          // Reset agar siklus PIR mode 2 bisa berjalan lagi di masa depan
  pinMode(PIR_PIN, INPUT_PULLUP);       // Set pin PIR ke mode pull-up (nonaktif)
}
```

Gambar 4. 48 Kode untuk mereset keadaan pencurian kabel netral

Program yang ditunjukan pada Gambar 4.48 merupakan mengembalikan kondisi sistem ke keadaan awal setelah tegangan semua fasa (R, S, dan T) kembali normal, yaitu berada dalam rentang 198–231 Volt. Hal ini penting untuk memastikan bahwa sistem siap mendeteksi ulang apabila terjadi gangguan atau pencurian kabel netral di kemudian hari. Berikut penjelasan lebih rinci dari program tersebut.

- if ((voltR >= 198.0 && voltR <= 231.0) && (voltS >= 198.0 && voltS <= 231.0) && (voltT >= 198.0 && voltT <= 231.0)) {

  Baris ini mengecek apakah semua fasa (R, S, dan T) memiliki tegangan yang sudah normal kembali. Jika sudah, maka seluruh sistem deteksi Mode 2 akan direset.
- sudahKirimInfoAbnormal = false;

  Reset status bahwa pesan tegangan abnormal telah dikirim. Hal ini memungkinkan sistem mengirim pesan kembali jika terjadi abnormal di masa depan.
- sudahKirimNotifikasiNetral = false;

  Reset status bahwa notifikasi indikasi pencurian kabel netral telah dikirim, agar bisa mengirim ulang jika terjadi kasus baru.
- dalamPengamatanGabungan = false;

  Menandakan bahwa sistem sudah tidak lagi berada dalam masa pengamatan gabungan, yang sebelumnya digunakan untuk memantau tegangan abnormal selama 5 detik.
- pirAktifNetral = false;

  Memastikan bahwa sensor PIR mode 2 tidak lagi aktif setelah masa pengamatan selesai.

- sudahSiklusPIRMode2 = false;
  Reset siklus PIR mode 2 agar dapat digunakan kembali jika di masa mendatang jika terjadi kondisi serupa.
- pinMode(PIR_PIN, INPUT_PULLUP);
  Mengatur kembali pin sensor PIR ke mode INPUT_PULLUP, yang artinya sensor dinonaktifkan sementara, untuk menjaga kestabilan dan menghindari pembacaan tak disengaja.