**Sistem Peringatan Kebakaran serta Kontrol dan Monitoring**

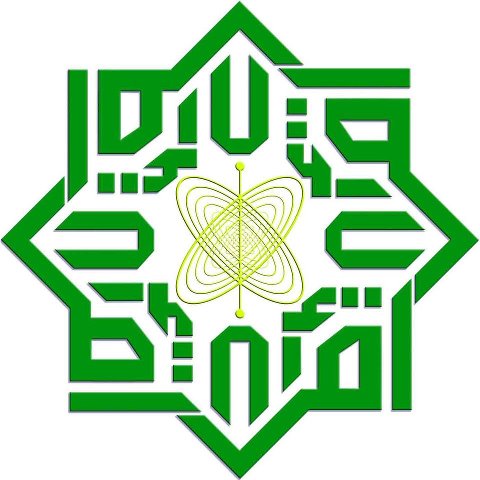
**Penggunaan Listrik Rumah Tangga Berbasis IoT**

**Terintegrasi dengan WhatsApp**

**PROPOSAL TUGAS AKHIR**

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik

Pada Program Studi Teknik Elektro Fakultas Sains dan Teknologi



Oleh:

**Zuhri Azharry**

**11655100074**

**PROGRAM STUDI TEKNIK ELEKTRO**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU**

**PEKANBARU**

**2022**

**LEMBAR HAK KEKAYAAN INTELEKTUAL**

Tugas Akhir ini terdaftar dan tersedia di Perpustakaan Universitas Islam Negeri Sultan Syarif Kasim Riau adalah terbuka untuk umum dengan ketentuan bahwa hak cipta pada penulis. Referensi diperkenankan dicatat, tetapi pengutipan atau ringkasan hanya dapat dilakukan seizin penulis dan harus disertai dengan kebiasaan ilmiah untuk menyebutkan sumbernya.

Penggandaan atau penerbitan sebagian atau seluruh Tugas Akhir ini harus memperoleh izin dari Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Sultan Syarif Kasim Riau. Perpustakaan yang meminjamkan Tugas Akhir ini untuk anggotanya diharapkan untuk mengisi nama, tanda peminjaman dan tanggal pinjam.

**Sistem Peringatan Kebakaran serta Kontrol dan Monitoring Penggunaan Listrik Rumah Tangga Berbasis IoT**

**Terintegrasi dengan WhatsApp**

**ZUHRI AZHARRY**

**NIM: 11655100074**

Tanggal Sidang: Januari 2023

Program Studi Teknik Elektro

Fakultas Sains dan Teknologi

Universitas Islam Negeri Sultan Syarif Kasim Riau

Jl. Soebrantas No. 155 Pekanbaru

Abstrak

Kebakaran merupakan bencana yang tidak dapat diprediksi kapan akan terjadinya salah satu faktor penyebabnya adalah korsleting listrik yang memicu terjadinya percikan api yang berakibat kebakaran rumah/pemukiman warga, sistem peringatan untuk mematikan peralatan yang tidak digunakan sekaligus dapat mengontrol dan memonitoring penggunaan listrik melalui Whatsapp secara real time diharapkan dapat menjadi pilihan untuk permasalahan ini. Aplikasi akan mengirimkan rangkuman pemakaian listrik harian dan memberikan notifikasi kepada pengguna yang telah didaftarkan apabila sensor mendeteksi api, asap pada suhu yang tinggi. Dalam pengiriman data ke server perangkat menggunakan metode komunikasi internet yang diakses langsung melalui mikrokontroler. Pengguna dapat mengontrol dan memonitoring penggunaan listrik dan status sensor melalui aplikasi WhatsApp. Hasil dari sistem yang telah diimplementasikan dapat membantu mengurangi penggunaan listrik. Sistem dapat mengirimkan notifikasi ke pengguna secara otomatis dalam waktu yang singkat yaitu antara 3 sampai 10 detik. Sistem dapat mengendalikan kelistrikan dari jarak jauh melalui pesan WhatsApp.

**Kata Kunci:** Listrik, IoT, Kontrol, Monitoring, Kebakaran.

**Fire Warning System and Control and Monitoring of IoT-Based Household Electricity Usage Integrated with WhatsApp**

**ZUHRI AZHARRY**

**Student Number: 11655100074**

Date of Exam: January 2023

*Department of Electrical Engineering*

*Faculty of Sains and Teknology*

*Syarif Kasim State Islamic University of Riau*

*Soebrantas Street , Number. 155 Pekanbaru*

Abstrak

*Fire is a disaster that cannot be predicted when it will occur, one of the causal factors is an electrical short circuit which triggers a spark which results in a house/residential fire, a warning system to turn off unused equipment while also being able to control and monitor electricity usage via Whatsapp in real time expected to be an option for this problem. The application will send a summary of daily electricity consumption and provide notifications to registered users when the sensor detects fire or smoke at high temperatures. In sending data to the device server using the internet communication method which is accessed directly through the microcontroller. Users can control and monitor electricity usage and sensor status through the WhatsApp application. The results of the implemented system can help reduce electricity usage. The system can send notifications to users automatically in a short time, which is between 3 and 10 seconds. The system can control electricity remotely via WhatsApp messages.*

**Keywords:** *Electricity, IoT, Control, Monitoring, Fire.*

**DAFTAR ISI**

**DAFTAR ISI ii**

**DAFTAR GAMBAR vi**

**DAFTAR TABEL vii**

**DAFTAR LAMBANG/NOTASI viii**

**DAFTAR SINGKATAN ix**

**BAB I PENDAHULUAN**

* 1. Latar Belakang I-1
  2. Rumusan Masalah I-4
  3. Tujuan Penelitian I-5
  4. Batasan Masalah I-5
  5. Manfaat Penelitian I-5

**BAB II TINJAUAN PUSTAKA**

* 1. Studi Literatur II-1
  2. *Warning System* II-2
  3. Mikrokontroler II-2
     1. ESP32 II-3
     2. Skema Daya ESP32 II-4
     3. Fitur ESP32 II-6
  4. *PZEM-004T* II-7
  5. MQ-2 II-8
  6. DHT-22 II-9
  7. Flame Sensor II-9
  8. *Relay* II-9
  9. *Database* II-11
     1. Jenis Jenis *Database* II-12
  10. WhatsApp II-13

**BAB III METODE PENELITIAN**

* 1. Diagram Alir Penelitian III-1
  2. Pengumpulan data III-3
  3. Ilustrasi Sistem III-3
  4. Perancangan *Hardware* III-4
     1. Spesifikasi Perangkat yang digunakan III-5
  5. Perancangan Listing Program Pada Kontroler III-6
     1. Spesifikasi Program yang akan digunakan III-6
  6. Pengujian Sistem III-6
     1. Pengujian *Hardware* III-6
     2. Pengujian Aplikasi/*Software* III-6
     3. Pengujian Keseluruhan Sistem III-7
  7. Sekenario Pengujian III-7
  8. Kebergunaan Sistem III-8

**DAFTAR PUSTAKA**

**LAMPIRAN**

**DAFTAR GAMBAR**

**HALAMAN**

Gambar 2.1. ESP-WROOM-32 II-3

Gambar 2.2. Blok Diagram ESP32 II-4

Gambar 2.3. Pin *Layout* ESP32 II-4

Gambar 2.4. Skema Daya Pada ESP32 II-5

Gambar 2.5. Wiring Diagram PZEM-004T II-7

Gambar 2.6. Blok Diagram PZEM-004T II-8

Gambar 2.7. Sensor MQ-2 II-8

Gambar 2.8. Sensor DHT-22 II-9

Gambar 2.9. Flame Sensor II-9

Gambar 2.10. Pinout Relay SPDT II-10

Gambar 2.11. Modul *Relay* 1 *Channel* II-10

Gambar 2.12. Pin *Control* Pada Modul *Relay* II-10

Gambar 2.13. LED Indikator Pada Modul *Relay* II-11

Gambar 2.14. Terminal *Output* Pada Modul *Relay* II-11

Gambar 3.1. Diagram Alir Penelitian III-2

Gambar 3.2. Ilustrasi Sistem III-3

Gambar 3.3. Blok Diagram *Hardware* III-4

Gambar 3.4. Sekema Keseluruhan Hardware III-5

**DAFTAR TABEL**

**HALAMAN**

Tabel 2.1. Konsumsi Daya berdasarkan Mode Daya Penggunaan II-6

Tabel 2.2. Fitur ESP32 II-7

Tabel 2.3. Jenis Jenis *Database* II-12

**DAFTAR LAMBANG/NOTASI**

µA : *Micro Ampere*

µV : *Micro Volt*

A : *Ampere*

C : *Celcius*

mA : *Mini Ampere*

Mhz : *Mega Hertz*

V : *Volt*

**DAFTAR SINGKATAN**

AC : *Alternating Current*

ADC : *Analog to Digital Converter*

ADT : *Android Developer Tool*

ALU : *Aritmatic Logic Unit*

AVL : *Automatic Vehicle Locator*

BLE : *Bluetooth Low Energy*

CPU : *Central Proccessing Unit*

CU : *Control Unit*

DBMS : *Database Management System*

DC : *Direct Current*

DPDT : *Doble Pole Doble Throw*

DPST : *Doule Pole Single Throw*

GPIO : *General-Purpose Input/Output*

GPRS : *General Package Radio Service*

I/O : *Input/Output*

I2C : *Inter-Integrated Circuit*

IC : *Integreted Circuit*

IoT : *Internet of Things*

LDO : *Low Drop Out*

LED : *Light Emitting Diode*

NC : *Normaly Close*

NO : *Normaly Open*

PCB : *Printed Circuit Bard*

PWM : *Pulse Width Modulation*

RAM : *Random Access Memory*

RTC : *Real Time Clock*

SoC : *System on Chip*

SPDT : *Single Pole Double Throw*

SPI : *Serial Pheriferal Interface*

SQL : *Structured Query Language*

ULP : *Ultra Low Power*

UX : *User Experience*

VAC : *Volt AC*

VCC : *Voltage Supply Colector*

VDC : *Volt DC*

WIFI : *Wireless Fidelity*

IDE : *Integrated Development Environment*

IEEE : *Institute of Electrical and Electronics Engineers*

**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang**

Energi listrik saat ini merupakan salah satu kebutuhan utama manusia dalam menjalankan aktivitas nya baik untuk kebutuhan rumah tangga hingga kebutuhan industri. Hal ini sejalan dengan laporan kenaikan konsumsi energi listrik setiap tahunya, 0.91 MWH/Kapita pada 2015, 1.02 MWH/Kapita pada 2017, 1.08 MWH/Kapita pada 2019, dan terus bertambah pada tahun 2021 menjadi 1.123 MWH/Kapita [1].

PT. Perusahaan Listrik Negara (PLN) sebagai pemasok utama energi listrik di Indonesia hingga saat ini menyediakan 2 skema pemakaian energi listrik yaitu prabayar/token dan sistem pasca bayar yang saat ini sudah tidak disarankan lagi untuk permintaan instalasinya kepada masyarakat retail[1]. Sering kali pelanggan dengan pasca bayar terkejut saat melakukan pembayaran ataupun mendapat tagihan penggunaan listriknya yang tidak sesuai ekspektasi. Berbeda halnya dengan pelanggan prabayar yang diharuskan melakukan pembelian token listrik sebelum dapat menggunakan listrik PLN. Namun kedua skema tersebut tidak memberikan pemberitahuan laporan penggunaan listrik secara harian maupun secara *realtime* yang dapat diakses dengan mudah oleh pelanggan, sehingga dapat membuat pelanggan terlena/lalai terhadap perangkat yang tidak digunakan namun tetap dinyalakan dan mengonsumsi listrik yang tinggi.

Disisi lain Kebakaran merupakan bencana yang tidak dapat diprediksi kapan akan terjadinya. Berdasarkan laporan tahunan Pemadam Kebakaran Kabupaten Indragiri Hilir dalam 5 tahun terakhir tercatat lebih dari 550 unit rumah masyarakat yang mengalami kebakaran dengan total nilai kerugiam mencapai ratusan juta rupiah. Dimana salah satu faktor penyebab nya adalah korsleting listrik yang memicu terjadinya percikan api yang berakibat kebakaran rumah/pemukiman warga.

Penelitian terkait pertama yang penulis temukan adalah “Rancang Bangun Alat Pengawas Pemakaian Listrik Rumah Tangga Menggunakan Sistem Internet of Thinks (IoT) Terintegrasi Web dan Telegram” oleh Deswita Adlyani Siregar, dalam penelitian tersebut sensor yang digunakan adalah PZEM004T sebagai sensor pengukuran penggunaan listrik, dan menggunakan NodeMCU/ESP8266 yang berfungsi sebagai Mikrokontroler sekaligus modul WiFi. Telegram dan Website digunakan sebagai Notifikasi untuk mengetahui penggunaan listrik harian. Hasil dari penelitian tersebut menyimpulkan pengurangan penggunaan listrik sebesar 15.25% dengan tingkat error rata-rata sebesar 3.8%[2]. Penelitian ini hanya dapat melakukan monitoring terhadap penggunaan listrik kemudian membandingkan konsumsi penggunaan sebelum dan sesudah sistem tersebut di implementasikan, dan masih menggunakan Telegram sebagai Message Broker nya.

Penelitian selanjutnya oleh Ronaldo Marcopolo Harianja denga judul “Rancang Bangun Monitoring Energi Listrik Pada Rumah Tangga Secara IoT Berbasis Mikrokontroler ATmega 328”. Penelitian tersebut menggunakan modul VR dan ACS712 sebagai sensor pembaca arus dan tegangan listrik. Pada sistem tersebut data input akan dikonversi menjadi data digital yang kemudian akan diproses untuk ditampilkan di LCD dan aplikasi android[3]. Penelitian ini masih mengguna sensor ACS712 yang hanya mampu mengukur arus hingga 5A di mana tidak efektif untuk digunakan pada rumah dengan daya 2200kwh. Sistem juga tidak memberikan notifikasi kepada pengguna, sehingga pengguna harus melihatnya secara manual.

Penelitian terkait selanjutnya adalah perancangan sistem pemakaian listrik dengan menggunakan sensor arus dan tegangan. Arduino Uno R3 digunakan sebagai mikrokontroler sistem ini, dan menggunakan modul SIM 800L sebagai media komunikasi antara sistem dan pengguna melalui layanan pesan singkat (SMS)[4]. Sistem ini depan berjalan dengan baik namun penggunaan SIM800L tidak efisien dan ekonomis pada zaman modern ini.

Penelitian berikutnya dengan topik Deteksi Kebakaran Rumah Tinggal berbasis WiFi dengan menggunakan Arduino UNO sebagai Mikrokontroler nya dan ESP8266 sebagai media komunikasi nya. Pada bagian input sensor menggunakan 3 buah sensor yaitu Sensor api, asap dan temperatur. Pada bagian output menggunakan LCD16x2 untuk menampilkan hasil data yang telah diproses [5]. Namun hasil dari penelitian tersebut menggunakan 2 buah mikrokontroller yang bertindak sebagai pengirim dan penerima. Pada akhir penelitian media komunikasi yang digunakan diubah menjadi modul NRF24Ll01yang mengakibatkan jarak komunikasi menjadi hanya 15m antar perangkat dan pengguna tidak dapat menerima informasi dari jarak jauh. Pada sisi sensor juga baru dapat beroperasi dengan optimal setelah perangkat dinyalakan dalam waktu 7.5 menit.

Penelitian selanjutnya dengan judul “Sistem Alarm Dan Monitoring Kebakaran Rumah Berbasis Nodemcu Dengan Komunikasi Android”. Pada penelitian tersebut menggunakan mikrokontroler NodeMCU sebagai media pemrosesannya dan Aplikasi Android sebagai output kepada pengguna. Penelitian tersebut hanya menggunakan sensor temperatur dan kelembaban DHT11[6] di mana hal tersebut akan berpotensi mengakibatkan terjadinya *fake alarm*/alarm palsu, dikarenakan indikator dari kebakaran tidak hanya pada temperatur yang ditinggi saja. Pada bagian aplikasi juga dibuat dengan APP Inventor di mana masih belum mendukung push notification/notifikasi yang akan langsung muncul pada layar smartphone pengguna, sehingga pengguna harus membuka aplikasi terlebih dahulu untuk mendapatkan notifikasi.

Pada penelitian sebelumnya komunikasi yang digunakan antara perangkat dan pengguna masih menggunakan layanan pesan singkat (SMS) di mana metode ini kurang efisien jika diterapkan pada saat ini. Penggunaan telegram masih kurang optimal untuk dilakukan di mana penulis lebih merekomendasikan menggunakan WhatsApp dikarenakan Di Indonesia WhatsApp lebih banyak digunakan dibandingkan telegram. Selanjutnya penggunaan Arduino dan IC ATmega juga kurang optimal untuk digunakan pada sistem ini dikarenakan masih menggunakan arsitektur 8-bit dan tidak menyertakan modul komunikasi WiFi secara bawaan. Mikrokontroler yang digunakan pada penelitian adalah mikrokontroler berbasis 32-bit yang diharapkan akan memberikan waktu pemrosesan dapat dilakukan lebih singkat yang telah termasuk dengan modul WiFi.

Berdasarkan referensi yang ada sebelumnya, peneliti akan melakukan pengembangan terhadap sistem monitoring dan kontrol terhadap pemakaian listrik serta sistem pendeteksi kebakaran. Penelitian yang akan dilakukan yaitu “***Rancang Bangun Sistem Peringatan Kebakaran serta Kontrol dan Monitoring Penggunaan Listrik Rumah Tangga Berbasis IoT Terintegrasi dengan WhatsApp***” di mana pada sistem ini memiliki prinsip kerja sebagai peringatan dan pengingat untuk mematikan peralatan yang tidak digunakan sekaligus dapat mengontrol dan memonitoring penggunaan listrik serta dapat memberikan peringatan terhadap kebakaran yang akan dikirimkan melalui Whatsapp dengan parameter-parameter yang telah ditetapkan secara *real* *time*. Sistem ini menggunakan mikrokontroler dengan prosesor *dual* *core* barbasis 32-bit yang akan mempercepat waktu pemrosesan dan diharapkan hasil dari monitoring dapat diterima pengguna dalam waktu sedini mungkin. Data dari sistem ini akan dikumpulkan pada *online database* *server* agar dapat diakses dengan mudah oleh pengguna. Dalam pengiriman data ke *server* perangkat menggunakan metode komunikasi internet gprs yang diakses langsung melalui mikrokontroler. Pengguna dapat mengontrol dan memonitoring penggunaan listrik dan status sensor melalui aplikasi WhatsApp. Aplikasi akan mengirimkan rangkuman pemakaian listrik harian dan memberikan notifikasi kepada pengguna yang telah didaftarkan apabila sensor mendeteksi api, asap pada suhu yang tinggi.

* 1. **Rumusan Masalah**

Berdasarkan latar belakang maka permasalahan yang ingin diatasi melalui penelitian ini adalah:

1. Bagaimana cara merancang sistem kontrol dan monitoring pemakaian listrik yang terkoneksi dengan internet?
2. Bagaimana cara merancang sistem peringatan kebakaran rumah dengan menggunakan Whatsapp sebagai media notifikasi?
   1. **Tujuan Penelitian**

Adapun tujuan yang ingin dicapai dalam penelitian ini adalah untuk membuat alat yang dapat melakukan kontrol dan juga monitoring penggunaan listrik sekaligus sistem peringatan kebakaran yang akan memberikan notifikasi melalui WhatsApp.

* 1. **Batasan Masalah**

Pada penelitian ini, dibuat batasan permasalahan sebagai berikut:

1. Modul yang digunakan adalah PZEM-004T sebagai sensor pembaca arus dan tegangan listrik.
2. Sistem memberikan informasi melalui aplikasi WhatsApp.
3. Percobaan dan pengembangan sistem ini dilakukan pada kamar kost dengan peralatan elektronik yang sama dan daya terpasang 900kwh.
4. Sistem menggunakan mikrokontroler berbasis 32-bit dual core dengan prosesor Xtensa LX6.
5. Sistem memberikan informasi melalui aplikasi android dan *website*.
   1. **Manfaat Penelitian**

Diharapkan penelitian ini dapat memerikan manfaat sebagai berikut:

1. Menurunkan tingkat konsumsi listrik rumah tangga pada peralatan yang tidak terpakai.
2. Melakukan kontrol pada kelistrikan rumah dari jarak jauh.
3. Meningkatkan keamanan dan kenyamanan dari potensi kebakaran rumah.

**BAB II**

**TINJAUAN PUSTAKA**

* 1. **Studi Literatur**

Dalam penelitian tugas akhir ini dilakukan studi literatur dengan tujuan mencari teori serta referensi yang relevan dengan kasus dan permasalahan yang akan diselesaikan, teori dan referensi yang didapatkan berasal dari artikel jurnal, *paper*, buku, penelitian terkait dan beberapa sumber lainya. Berikut ini disajikan beberapa penelitian terdahulu yang dikumpulkan dari berbagai sumber sebagai referensi dan dasar teori yang berkaitan dengan permasalahan yang akan diselesaikan.

Tujuan perancangan alat ini adalah untuk melakukan monitoring dan kontrol terhadap peralatan listrik rumah tangga sekaligus sebagai sistem pendeteksi kebakaran yang akan memberikan notifikasi ke pemilik rumah maupun pihak pemadam kebakaran untuk meminimalkan kerugian terhadap kejadian tersebut. Pada penelitian-penelitian sebelumnya memiliki beberapa perbedaan dan persamaan satu sama, baik pada sensor maupun pada mikrokontroler.

Penelitian terkait dengan topik pengawasan pemakaian listrik berbasis IoT yang melakukan monitoring pada penggunaan listrik menggunakan ESP8266 dan sensor PZEM-004T untuk membaca nilai arus dan tegangan yang digunakan. Penelitian tersebut juga mengintegrasikan penggunaan IoT sebagai antarmuka dengan pengguna untuk memudahkan dalam membaca hasil monitoring menggunakan Telegram bot dan Website yang akan dikirimkan ke pengguna setiap harinya[2]. Namun pada penelitian ini belum disematkan sistem kontrol yang dapat mengendalikan listrik dari jarak jauh dan belum memiliki sistem cut-off untuk memutus aliran listrik otomatis sesuai pengaturan yang diterapkan.

Penelitian selanjutnya dengan judul “Arduino Based Smart Energy Meter using GSM”, pada penelitian tersebut menggunakan sensor arus, tegangan dan juga RTC (Real Time Clock) untuk mengukur pemakaian listrik disertai dengan waktu penggunaannya. Sebagai *interface* dengan pengguna penelitian tersebut menggunakan modul GSM untuk mengirimkan SMS/Layanan Pesan Singkat ke pengguna dan pada alat juga disematkan LCD 16x12 sebagai tampilan debug maupun monitoring secara langsung. Sayangnya sistem tersebut masih menggunakan Arduino UNO sebagai mikrokontroler dan SMS sebagai interface yang kurang efektif untuk digunakan pada sistem tersebut untuk saat ini.

Penelitian berikutnya dengan judul “Rancang Bangun Alat Pendeteksi Asap Kebakaran Menggunakan Sensor MQ-2 Berbasis Arduino Uno”, pada penelitian tersebut sistem akan membaca nilai konsentrasi kadar CO2 pada udara sebagai *trigger* untuk mengaktifkan alarm sebagai peringatan kepada pengguna telah terjadi kebakaran,[7] namun pada penelitian ini hanya menggunakan MQ-2 sebagai input, Arduino UNO sebagai Proses dan Buzzer sebagai outuput. Di mana sistem akan tidak efektif dalam menentukan status kebakaran hanya dengan menggunakan parameter saja yakni asap.

Kemudian telah dilakukan juga penelitian dengan judul “Simulasi Alarm Kebakaran Menggunakan Sensor MQ-2, Falme Sensor Berbasis Mikrokontroler Arduino”, dimana pada penelitian tersebut ditambahkan sensor api untuk mendeteksi adanya api di sekitar ruangan sebagai parameter tambahan pada sensor asap untuk meningkatkan akurasi pengambilan keputusan terhadap status kebakaran. Pada sisi output menggunakan LCD 16x2 dan Buzzer yang akan mengeluarkan suara saat kondisi kebakaran terpenuhi[8]. Pada penelitian ini belum disematkan notifikasi dan monitoring sensor yang dapat diterima pengguna dari jarak jauh.

Penelitian selanjutnya berjudul “Rancang Bangun Prototype Alat Pendeteksi Kebakaran Menggunakan Arduino Uno Dilengkapi Pemadam Dan Notifikasi SMS Gateway” pada penelitian tersebut ditambahkan sebuah output berupa notifikasi SMS untuk memberitahukan kepada pengguna dari jarak jauh[9], namun untuk saat ini penggunaan SMS membutuhkan pulsa/biaya yang cukup mahal dan kurang efisien digunakan.

* 1. ***Warning System***

Sistem peringatan merupakan suatu sistem yang dapat memberitahukan suatu kejadian yang terjadi berdasarkan beberapa parameter-parameter yang berhubungan. Seperti kejadian tsunami yang biasanya didahului oleh gempa bumi ataupun pinggir pantai yang secara tiba-tiba surut dalam waktu singkat, Tanah longsor yang didahului oleh hujan lebat maupun gempa bumi, dan pada penelitian ini kebakaran rumah dengan parameter *temperature*, kelembaban udara, dan konsentrasi karbon dioksida dalam ruangan yang akan menggunakan parameter dari sensor untuk memberikan notifikasi kepada pemilik rumah dan dinas pemadam kebakaran.

* 1. **Mikrokontroler**

Mikrokontroler (µC) adalah komputer mini berbasis SoC (*single on chip*) yang dikemas dalam bentuk sebuah chip yang diprogram untuk menjalankan tugas tertentu dan merupakan salah satu syarat minimal dalam *system embedded.* Mikrokontroler berbeda dengan Mikroprosesor karena di dalam sebuah mikrokontroler umumnya juga telah terdapat komponen pendukung sistem minimal mikroprosesor, memori dan antarmuka I/O, sedangkan di dalam mikroprosesor umumnya hanya berisi CPU saja.[10]

* + 1. **ESP32**

ESP32 merupakan chip WiFi 2.4 GHz yang dikombinasikan dengan *Bluetooth* dalam *single* *on* *chip* (SoC) yang dirancang oleh Taiwan *Semiconductor* *Manufacturing* *Company* (TSMC) menggunakan konsumsi *Ultra* *Low* *Power* dengan teknologi 40µm. ESP32 dirancang untuk mencapai daya dan kinerja pada frekuensi radio yang baik, dengan ketahanan, keserbagunaan dan keandalan dalam berbagai macam aplikasi dan skenario penggunaan daya.[11] ESP32 Dikhususkan untuk perangkat bergerak/*mobile*, dan *Internet*-*of*-*Things* (IoT).

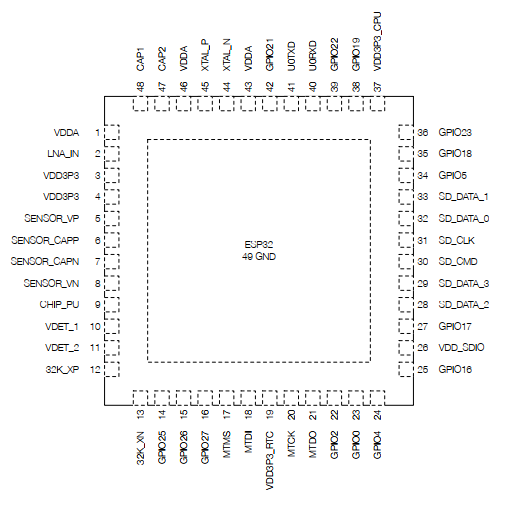


Gambar 2.1. ESP-WROOM-32[12]

Semua fitur mutakhir yang digunakan merupakan karakteristik dari *chip* dengan penggunaan daya yang rendah, termasuk sistem pewaktuan yang lebih halus, mode daya ganda, dan penyekalaan daya yang dinamis. Misalnya, dalam skenario aplikasi sensor IoT berdaya rendah, perangkat akan diaktifkan secara berkala dan hanya pada kondisi tertentu terdeteksi. Siklus tersebut digunakan untuk meminimalisir penggunaan daya pada chip.



Gambar 2.2. Blok Diagram ESP32[11]

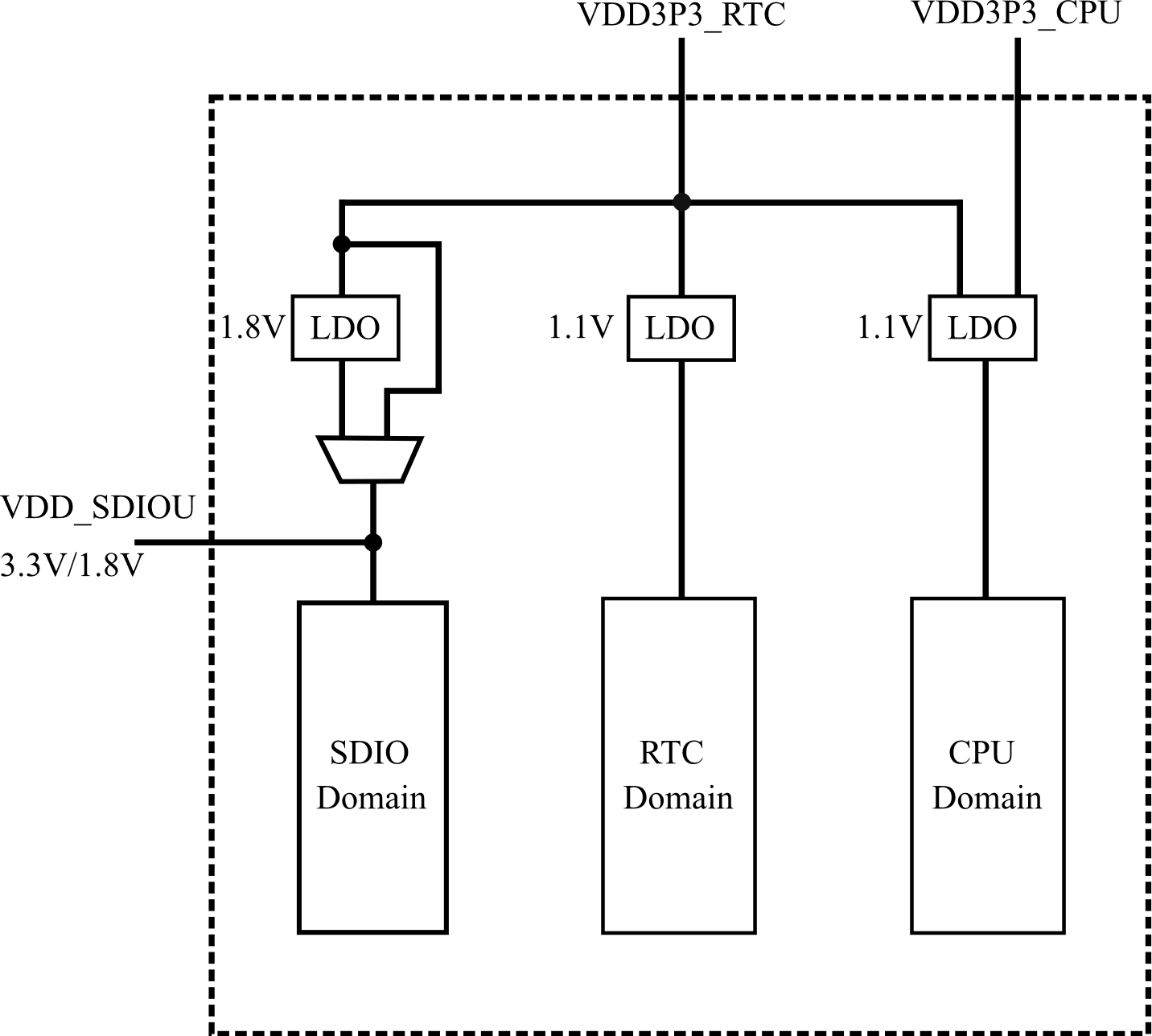


Gambar 2.3. Pin *Layout* ESP32[11]

* + 1. **Skema Daya**

Pada ESP32 skema daya pada pin digital dibagi menjadi 3 jenis domain:

1. VDD3P3\_RTC yang merupakan catu daya input untuk RTC (*Real* *Time* *Clock*) dan CPU (*Central Processing Unit*).
2. VDD3P3\_CPU merupakan catu daya untuk CPU (*Central Processing Unit*).
3. VDD\_SDIO terhubung ke output dari Regulator LDO (*Low Dropout*) dengan menggunakan *input* dari VDD3P3\_RTC. Saat VDD\_SDIO dan VDD3P3\_RTC terhubung dalam rangkaian yang sama maka Regulator LDO *internal* secara otomatis akan dinonaktifkan.

  
Gambar 2.4. Skema Daya Pada ESP32[11]

Dengan menggunakan teknologi manajemen daya yang canggih, ESP32 dapat beralih di antara berbagai mode daya. Berikut adalah mode daya yang ada pada ESP32.

1. **Mode Aktif:** Radio WiFi chip dihidupkan. Chip dapat menerima atau mengirim data.
2. ***Modem*-*sleep* mode:** CPU tetap beroperasi dan *Clock* dapat dikonfigurasi. Tetapi penggunaan Wi-Fi/*Bluetooth* dan radio dinonaktifkan.
3. ***Light*-*sleep* mode:** Penggunaan CPU terjeda, *Real* *Time* *Clock* pada memori dan *peripheral* tetap berjalan dengan normal, CPU akan diaktifkan kembali jika ada interupsi eksternal, RTC *Timer,* atau aktifitas jaringan yang mengguakan WiFi maupun *Bluetooth*.
4. ***Deep*-*sleep* mode:** Hanya RTC pada memori dan periferal yang diaktifkan. Koneksi Wi-Fi dan Bluetooth disimpan dalam memori. Prosesor *Ultra*-*Low* *Power* (ULP) tetap berfungsi.
5. **Mode Hibernasi:** Osilator 8-MHz *internal* dan co-prosesor ULP dinonaktifkan. Memori pemulihan RTC dimatikan. Hanya satu RTC *Timer* pada *clock* *speed* yang rendah dan GPIO RTC tertentu yang aktif. *Timer* RTC atau RTC GPIO dapat mengaktifkan kembali *chip* dari *mode* Hibernasi ke Mode Aktif.

Tabel 2.1. Konsumsi Daya berdasarkan Mode Daya Penggunaan

|  |  |  |
| --- | --- | --- |
| Mode Daya | Deskripsi | Konsumsi Daya |
| Aktif | Mengirim Data (Tx) | 180 ~ 240 mA |
| Menerima Data (Rx) | 95 ~ 100 mA |
| *Modem-sleep* | *Single-core chip* | 20 ~ 34 mA |
| *Dual-core chips* | 20 ~ 68 mA |
| *Light-sleep* |  | 0.8 mA |
| *Deep-sleep* | Prosesor ULP Diaktifkan | 150 µA |
| Sensor ULP dalam keadaan *monitoring* mode | 100 µA |
| RTC timer + *memory* | 10 µA |
| Hibernasi | Hanya RTC *Timer* yang aktif | 5 µA |
| *Power Off* | Chip *Pull* *Up* diatur menjadi *Low* Level.  Chip dimatikan | 1 µA |

Prosesor *Ultra-low Power* (ULP) dan memori RTC tetap dihidupkan selama mode deep-sleep. Oleh karena itu, pengembang dapat menyimpan program untuk prosesor ULP dalam memori RTC untuk mengakses perangkat periferal, *timer* internal dan sensor internal selama *mode* *deep*-*sleep*. Ini berguna untuk merancang aplikasi yang membutuhkan CPU untuk dibangunkan oleh aktivasi eksternal, ataupun melalui *timer*, atau kombinasi keduanya, dengan tetap mempertahankan konsumsi daya yang minimal.

* + 1. **Fitur ESP32**

ESP32 dapat beroperasi dengan handal di lingkungan industri, dengan suhu pengoperasian berkisar antara –40°C hingga +125°C. Didukung oleh sirkuit kalibrasi canggih, ESP32 secara dinamis dapat menghapus ketidaksempurnaan sirkuit eksternal dan beradaptasi dengan perubahan kondisi eksternal. Didesain untuk perangkat seluler, perangkat elektronik yang dapat dipakai dan aplikasi IoT, ESP32 mencapai konsumsi daya sangat rendah dengan kombinasi beberapa jenis perangkat lunak. ESP32 sangat terintegrasi dengan saklar antena *built*-*in*, penguat daya, *low-noise receive amplifier*, dan modul manajemen daya.

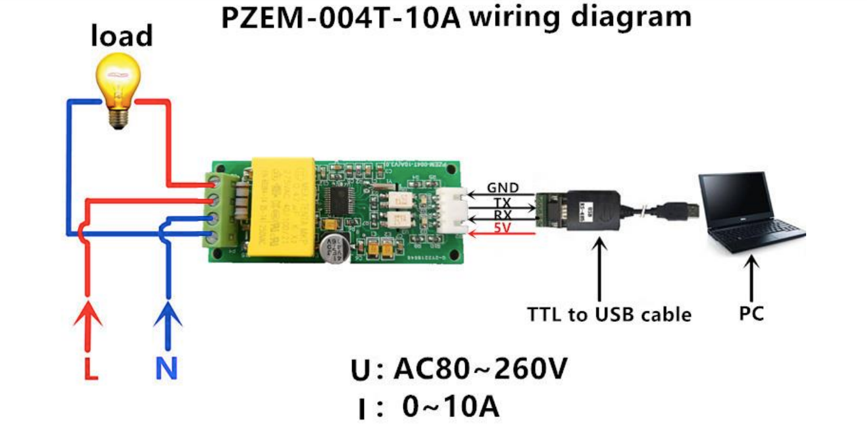
ESP32 dapat berfungsi sebagai sistem *standalone* yang lengkap atau sebagai perangkat *slave* untuk MCU *host*, mengurangi *overhead* dari komunikasi pada prosesor aplikasi utama. ESP32 dapat berinteraksi dengan sistem lain untuk menyediakan fungsi WiFi dan *Bluetooth* melalui antarmuka SPI/SDIO atau I2C/UART. Berikut ini adalah fitur dari ESP32:

Tabel 2.2. Fitur ESP32[13]

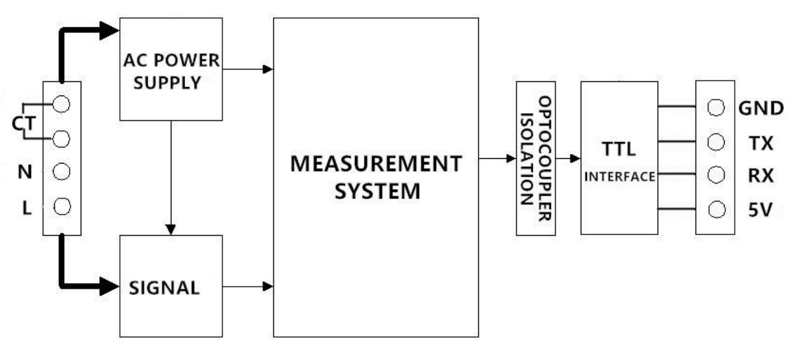
|  |  |  |
| --- | --- | --- |
| No. | Fitur | Deskripsi |
| 1 | *Prosesor* | *Xtensa dual-core 32-bit LX6 240 MHz*  *Ultralow power (ULP) co-processor* |
| 2 | *Memory* | *520 KB SRAM (Static Random Access Memory)* |
| 3 | *Wireless connectivity* | *WiFi: 802.11 b/g/n*  *Bluetooth: v4.2 BR/EDR dan BLE* |
| 4 | *Peripheral interfaces* | *12-bit ADC up to 18 channels*  *2 × 8-bit DAC*  *10 × touch sensors*  *4 × SPI*  *2 × I²S interfaces*  *2 × I²C interfaces*  *3 × UART*  *Motor PWM*  *LED PWM (up to 16 channels)*  *Hall effect sensor*  *Ultralow power analog pre-amplifier* |
| 5 | *Security* | *IEEE 802.11 WFA, WPA/WPA2 and WAPI*  *Secure boot*  *Flash encryption*  *1024-bit OTP, up to 768-bit for customers* |
| 6 | *Power management* | *Internal low-dropout regulator*  *Individual power domain for RTC* |

* 1. **PZEM-004T**

Sensor PZEM-004T dapat membaca tegangan AC mulai dari 80 ~260 V dengan resolusi 0.1v dan tingkat akurasi sebesar 0.5%. Dan dapat mengukur arus 0~10A mulai dari 0.01A dengan resolusi 0.001A dan akurasi pengukuran 0.5%. modul ini dapat beroperasi pada temperatur -20ºC ~ +60ºC.[14]



Gambar 2.5. Wiring Diagram PZEM-004T[14]



Gambar 2.6. Blok Diagram PZEM-004T[14]

Ketika melakukan pengukuran modul ini akan membaca voltase dan juga arus yang mengalir untuk selanjutnya akan dikonversi menjadi sinyal digital yang dapat dibaca oleh mikrokontroler dan akan diberikan proteksi antara tegangan tinggi 220v dan 5v-12v mengunakan isolator optocoupler sama halnya seperti penggunaan relay.

* 1. **MQ-2**

Modul MQ-2 merupakan sensor yang dapat mendeteksi kosentrasi LPG, Asap/CO2, Alkohol, Propana, Hydrogen dan Methana. MQ-2 termasuk dalam tipe *Metal Oxide Semiconductor (MOS)* yang dapat mendeteksi tingkat kosentrasi 200 ~ 1000ppm (*parts per million*) dengan tegangan operasi 5V yang dapat beroperasi selama 24jam.



Gambar 2.7. Sensor MQ-2

Saat mengukur gas seperti karbon dioksida, oksigen, atau metana, istilah konsentrasi digunakan untuk menggambarkan jumlah gas berdasarkan volume di udara. Satuan pengukuran yang paling umum adalah ppm, dan *percent concentration.* Parts-per-million (disingkat ppm) adalah rasio dari satu gas ke gas lainnya. Misalnya, 1.000 ppm CO berarti bahwa terdapat satu juta molekul gas, 1.000 di antaranya adalah karbon monoksida dan 999.000 molekul akan menjadi beberapa gas lainnya.

* 1. **DHT-22**

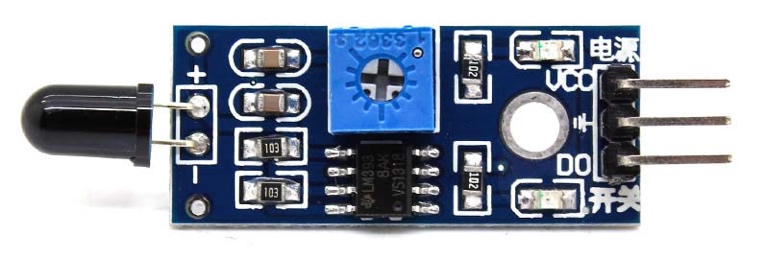
Modul DHT-22 merupakan sensor yang dapat membaca temperatur dan kelembaban ruangan berdasarkan prinsip kerja dari NTC (*Negative Temperature Coefficient*) Thermistor yang dapat mengukur temperatur -40ºC sampai 125ºC dengan tingat akurasi ±0.5º. Modul ini juga dapat mnegukur tingkat kelembaban ruangan mulai dari 0% sampai 100% dengan akurasi ±2-5%.



Gambar 2.8. Sensor DHT-22[15]

* 1. **Flame Sensor**

Modul Flame Sensor merupakan sensor yang dapat mendeteksi api berdasarkan pada spektrum panjang gelombang pada sumber cahaya dengan rentang 760nm sampai dengan 1100nm dengan menggunakan infrared yang berfungsi sebagai tranduser untuk mendeteksi nyala api dan menghasilkan output berupa sinyal listrik analog maupun digital.

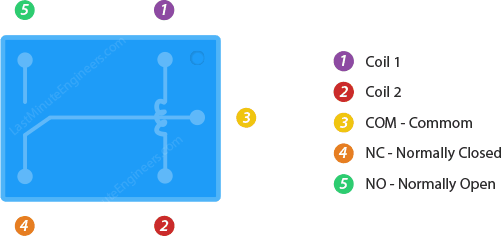


Gambar 2.9. Flame Sensor

Sensor ini dapat mendeteksi nyala api dengan besar sudut 60º yang dapat membedakan antara spektrum antara cahaya lampu penerangan dan cahaya yang ditimbulkan oleh nyala api. Terdapat juga sebuah potensiometer yang digunakan untuk mengatur tingkat sensitivitas pembacaan dari sensor.

* 1. **RELAY**

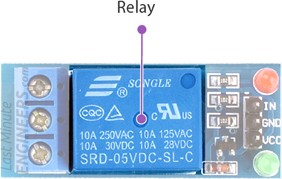
Relai adalah perangkat yang menggunakan elektromagnet untuk mengoperasikan kontak sakelar *output* nya. Susunan paling sederhana terdiri dari induktor kawat penghantar yang dililit pada inti besi. Bila kumparan ini dilewati oleh arus listrik, medan magnet yang terbentuk menarik sakelar yang digunakan sebagai pengungkit mekanisme sakelar magnet. Selain menggunakan jenis elektromagnet, relai juga telah dikembangkan dengan jenis *solid state relay* dan *relay* *numeric*. Konfigurasi pada *solid state relay* dan *numeric relay* dapat dilakukan dengan lebih mudah jika dibandingkan dengan konfigurasi rangkaian pada relai elektromagnet.[16]



Gambar 2.10. *Pinout Relay* SPDT[17]

Relai pada umumnya memiliki 5 pin, 3 diantara nya digunakan untuk terminal tegangan tinggi yang akan dikendalikan. Kabel listrik utama menggunakan terminal *common* (COM) yang ada pada relai. Sementara penggunaan terminal *Normaly Close* (NC) & *Normaly Open* (NO) tergantung pada apakah perangkat ingin dinyalakan atau dimatikan.

Bagian pada modul *Relay Single Pole Dual Throw* (SPDT) atau bisa juga disebut sebagai relai yang dapat mengendalikan 2 keadaan dengan 1 *input* saja adalah seperti Gambar di bawah ini:

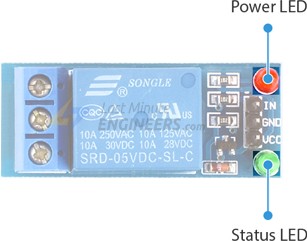


Gambar 2.11. Modul *Relay* 1 Channel[17]

Pada Gambar diatas merupakan *relay* dengan tegangan kerja 5 Volt DC yang dapat mengendalikan arus listrik maksimal 10A 250 Volt AC. Kemudian pada Gambar di bawah ini merupakan pin yang digunakan untuk mengendalikan saklar yang ada pada *relay* yaitu pin Input, VCC dan *Ground*.



Gambar 2.12. Pin *Control* Pada Modul *Relay*[17]



Gambar 2.13. LED Indikator Pada Modul *Relay*[17]

Pada Gambar dibawah ini merupakan terminal Output yang terdapat pada relai SPDT yaitu terminal *Common, Normaly Open* dan *Normaly Close*



Gambar 2.14. Terminal *Output* Pada Modul *Relay*[17]

* 1. ***DATABASE***

*Database*/Basis data adalah kumpulan data/informasi yang terstruktur dan terorganisir, *database* biasanya disimpan secara elektronik dalam sistem komputer. Basis data dikelola dan dimanajemen oleh perangkat lunak *Database management system* (DBMS) yang digunakan untuk berinteraksi dengan pengguna, aplikasi, dan basis data untuk membuat, menyimpan, memperbaharui, dan menganalisis data.[18]

Data dalam *database* yang paling umum digunakan saat ini biasanya dimodelkan dalam baris dan kolom di dalam serangkaian tabel untuk membuat pemrosesan dan kueri data menjadi efisien. Data kemudian dapat dengan mudah diakses, dikelola, dimodifikasi, diperbarui, dikendalikan, dan diatur. Sebagian besar *database* menggunakan bahasa *structured query language* (SQL).[19]

*Database* telah berkembang secara drastis sejak tahun 1960-an. Baru-baru ini, database NoSQL muncul sebagai tanggapan terhadap pertumbuhan internet dan kebutuhan akan kecepatan akses yang lebih cepat dan pemrosesan data yang tidak terstruktur. *Database* NoSQL dibuat dengan tujuan khusus yaitu untuk model data spesifik dan data yang memiliki skema fleksibel untuk membuat aplikasi modern. *Database* NoSQL dikenal secara luas karena kemudahan pengembangan, kinerja, dan fungsionalitas dalam berbagai skala.

Basis data dan *spreadsheet* (seperti Microsoft Excel) merupakan perangkat lunak yang digunakan untuk menyimpan informasi berdasarkan baris dan kolom dalam tabel. Perbedaan utama antara keduanya adalah Bagaimana data disimpan dan dimodifikasi, Siapa yang dapat mengakses data, dan Berapa banyak data yang dapat disimpan. *Spreadsheet* awalnya dirancang untuk satu pengguna, dan karakteristiknya mencerminkan hal itu. *Software* ini bagus untuk satu pengguna atau sejumlah kecil pengguna yang tidak perlu melakukan banyak modifikasi data yang sangat rumit. *Database*, di sisi lain, dirancang untuk menampung koleksi informasi yang jauh lebih besar dan ​​terkadang dalam jumlah besar. *Database* memungkinkan banyak pengguna pada saat yang sama dengan cepat dan aman mengakses dan meminta data menggunakan logika dan perintah yang sangat kompleks.[19]

* + 1. **Jenis-jenis *Database***

Ada banyak jenis *database*. Basis data terbaik untuk organisasi tertentu tergantung pada bagaimana organisasi bermaksud menggunakan data. Berikut ini merupakan beberapa jenis database dan definisinya:

Tabel 2.6. Jenis *Database*[19]

|  |  |  |
| --- | --- | --- |
| No. | Jenis *Database* | Definisi |
| 1 | *Database* relasional | *Database* relasional menjadi dominan pada 1980-an. Item dalam database relasional diatur sebagai satu set tabel dengan kolom dan baris. Teknologi basis data relasional menyediakan cara yang paling efisien dan fleksibel untuk mengakses informasi terstruktur. |
| 2 | *Database* berorientasi objek. | Informasi dalam *database* berorientasi objek direpresentasikan dalam bentuk objek, seperti dalam pemrograman berorientasi objek. |
| 3 | *Database* terdistribusi. | *Database* terdistribusi terdiri dari dua atau lebih file yang terletak di situs yang berbeda. Basis data dapat disimpan pada banyak komputer, terletak di lokasi fisik yang sama, atau tersebar di berbagai jaringan. |
| 4 | Data *Warehouse* | Repositori pusat untuk data, gudang data adalah jenis *database* yang dirancang khusus untuk permintaan dan analisis cepat. |
| 5 | NoSQL *Database*. | NoSQL, atau *database* non-relasional, memungkinkan data yang tidak terstruktur dan terstruktur disimpan dan dimanipulasi (berbeda dengan *database* relasional, yang mendefinisikan bagaimana semua data yang dimasukkan ke dalam basis data harus dikomposisi). *Database* NoSQL semakin populer ketika aplikasi web menjadi lebih umum dan lebih kompleks. |
| 6 | *Database* grafik. | *Database* grafik menyimpan data dalam hal entitas dan hubungan antar entitas. |
| 7 | *Database* OLTP. | *Database Online Transaction Processing* (OLTP) adalah *database* analitik cepat yang dirancang untuk sejumlah besar transaksi yang dilakukan oleh banyak pengguna. |

* 1. **WhatsApp**

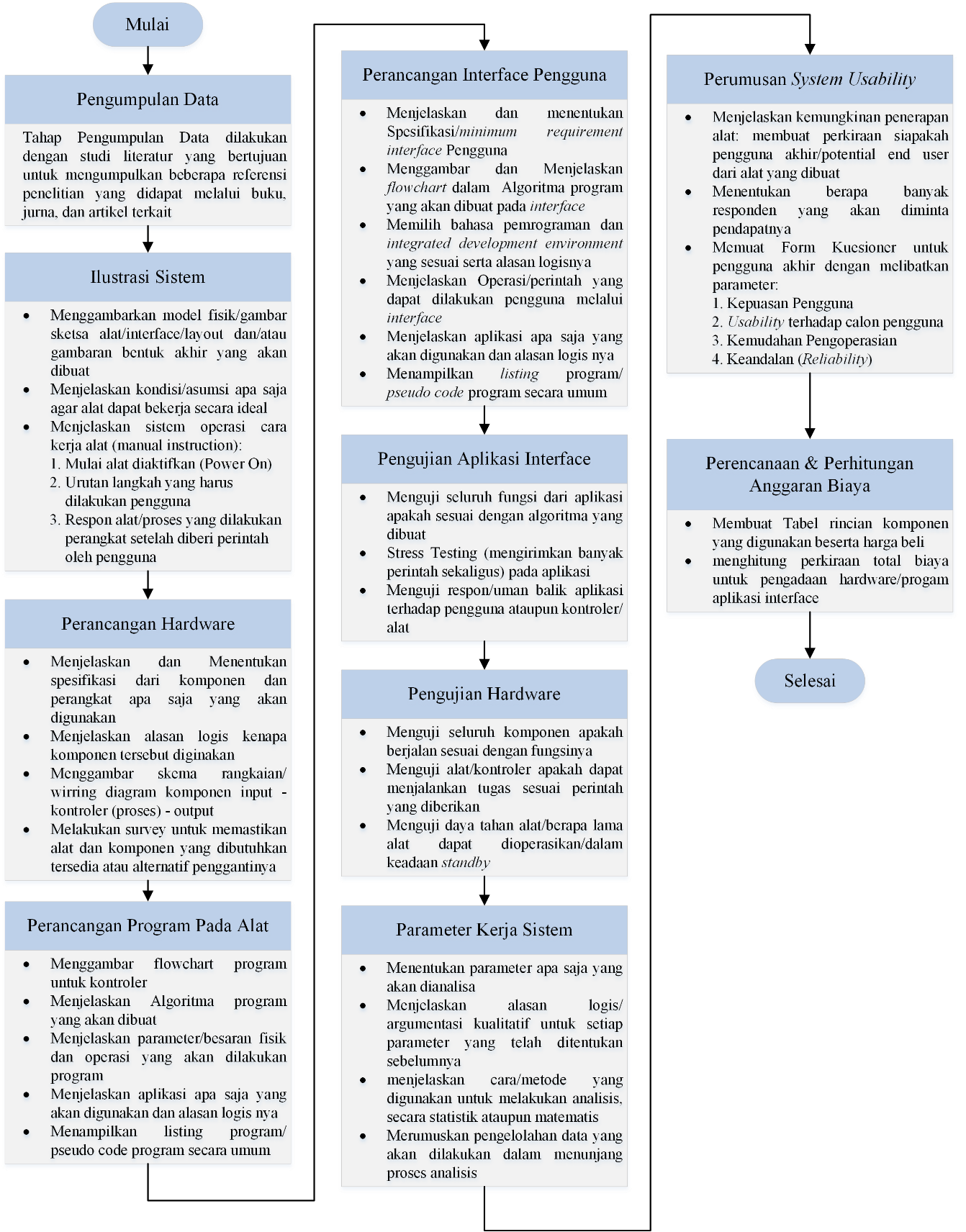
WhatsApp adalah aplikasi bertukar pesan yang dikembangkan oleh WhatsApp Inc. dan diakuisisis oleh Facebook inc pada 2014 (sekarang Meta Platform) dan telah diunduh lebih dari 5 Miliar kali di Play Store. Aplikasi WhatsApp telah digunakan lebih dari 84 Juta orang di Indonesia[20] yang dapat di install pada berbagai sistem operasi baik Android, iOS, Windows, MacOS, Linux bahkan melalui internet Browser seperti Chrome dan Firefox.

Whatsapp memiliki berbagai fitur pengiriman pesan multimedia seperti text, gambar, suara, link/url dan video. Whatsapp juga tidak membutuhkan trafik koneksi yang tinggi untuk melakukan pengiriman dan penerimaan pesan hanya memerlukan koneksi ~4KB/s dan tanpa iklan maupun biaya tambahan lainya[21].

**BAB III**

**METODE PENELITIAN**

* 1. **Diagram Alir Penelitian**



Gambar 3.1. Diagram Alir Penelitian

Dalam penelitian tugas akhir ini penulis menggunakan metode penelitian dan pengembangan (*Research and Development* atau *R&D*). Metode Penelitian dan Pengembangan merupakan metode yang digunakan dalam menghasilkan produk tertentu kemudian menguji tingkat efektivitas dan efisiensi produk tersebut. Penelitian ini dimulai dengan mengumpulkan data-data serta mempelajari teori yang relevan dengan penelitian ini seperti sistem kontrol, IoT, *monitoring*, dan kendali yang akan digunakan sebagai bahan penunjang dalam perancangan dan pembuatan penelitian ini.

Penelitian ini diawali dengan mengumpulkan data terkait topik penelitian, selanjutnya adalah proses perancangan sistem *hardware* dan *software.* Setelah tahapan perancangan selesai dilanjutkan dengan tahapan pengujian perangkat keras dan perangkat lunak, proses ini akan dilanjutkan jika tidak terdapat masalah ada sistem. Namun jika terdapat permasalahan dalam *hardware* maupun *software* akan dilakukan pengembangan ulang hingga sesuai dengan perencanaan. Tahap selanjutnya adalah mengimplementasikan sistem yang telah dibuat sekaligus evaluasi terhadap perangkat yang dikembangkan.

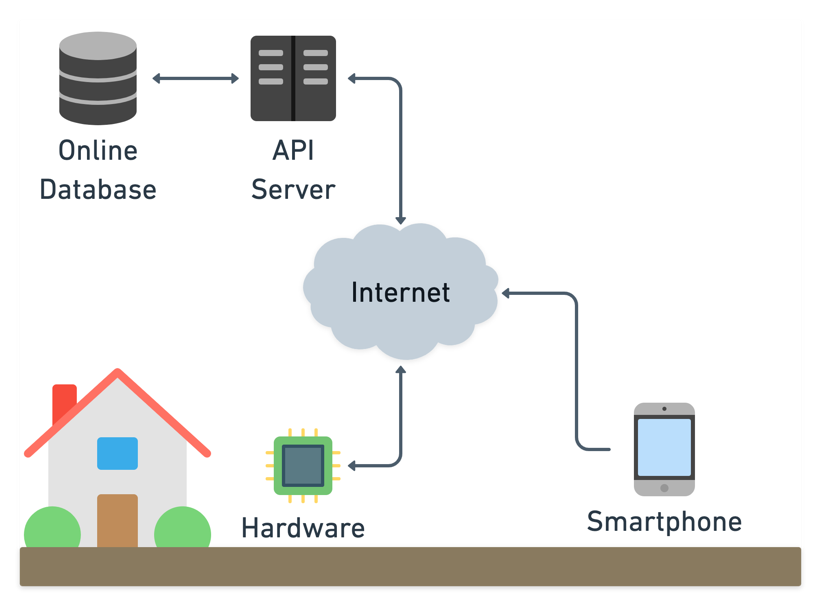
Untuk mendapatkan data dari hasil penelitian ini akan dilakukan analisa terhadap sistem sekaligus uji kelayakan terhadap alat yang di implementasikan dengan menggunakan kuesioner. Kritik dan saran juga akan dimasukkan ke dalam kuesioner agar dapat dijadikan sebagai evaluasi dan pengembangan terhadap alat yang telah dibuat pada penelitian ini. Diagram alir dalam penelitian ini akan ditunjukkan pada Gambar 3.1. yang sesuai dengan yang telah dijelaskan pada awal bab ini.

* 1. **Pengumpulan Data**

Metode Pengumpulan data yang dilakukan pada penelitian ini adalah dengan metode studi literatur. Metode ini berfungsi untuk mengumpulkan dan mempelajari teori-terori yang akan menjadi pendukung dalam penelitian ini. Studi dilakukan dari berbagai sumber literasi seperti buku, jurnal, artikel, *datasheet*, *manual* *book*, ataupun penelitian sejenis yang telah dilakukan sebelumnya. Tujuan dilakukannya studi literatur adalah untuk mencari data data yang berhubungan dengan topik penelitian ini.

* 1. **Ilustrasi Sistem**

Langkah awal dalam perancangan adalah dengan membuat blok diagram sebagai Gambaran umum dalam merancang suatu sistem sehingga keseluruhan blok diagram tersebut akan menghasilkan suatu sistem yang dapat berfungsi sesuai dengan perancangan di awal. Dalam penelitian ini tahapan perancangan terbagi menjadi beberapa bagian yaitu perancangan *hardware*, *listing* program untuk *hardware*, dan aplikasi sebagai *interface* dengan pengguna akhir.



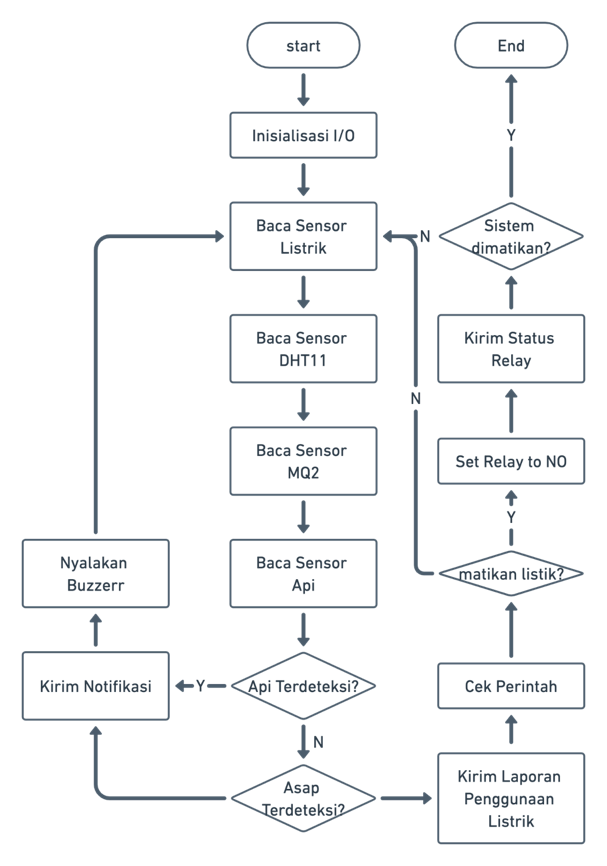
Gambar 3.2. Ilustrasi Sistem

Ketika perangkat diaktifkan secara otomatis akan mencari koneksi internet dan melakukan insialisasi sensor untuk menerima input, selanjutnya data input akan diproses untuk menentukan pengambilan keputusan kemudian data hasil proses akan dikirimkan ke *database* dan API untuk selanjutnya dikirimkan ke pengguna jika diperlukan/diminta.

Pada sisi pengguna dapat melakukan monitoring, kontrol hingga menerima notifikasi/pengingat pada *smartphone* untuk mencabut stop kontak jika penggunaan listrik berlebihan atau sesuai dengan pengaturan yang telah di buat

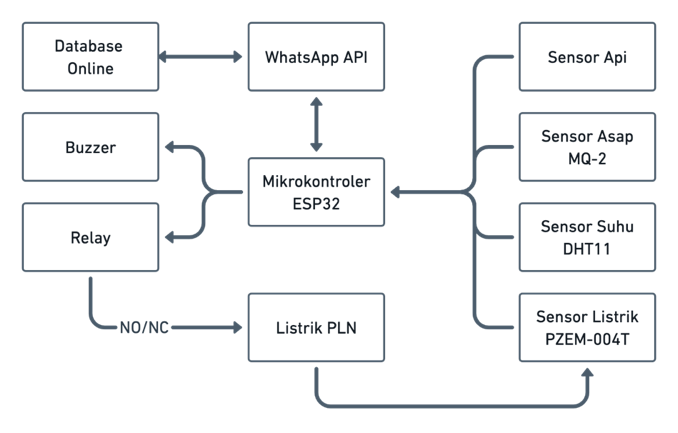
* 1. **Perancangan Hardware**

Perancangan sistem ini terdiri dari perangkat keras yang pengoperasiannya dilakukan oleh *listing* program yang akan ditanamkan ke dalam mikrokontroler. Lalu seluruh aktivitasnya juga dapat dikendalikan menggunakan aplikasi *front-end* yang ada di sisi pengguna.



Gambar 3.3. Diagram Alir sistem

Sistem yang akan dirancang dapat bekerja secara otomatis saat mendapatkan perintah/*trigger* *eksternal*. Secara blok diagram perancangan pada sisi perangkat keras penulis ilustrasikan pada Gambar 3.4. berikut ini.



Gambar 3.4. Blok Diagram *Hardware*

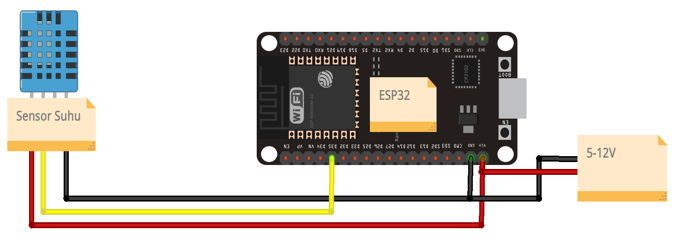
Blok diagram diatas dibuat berdasarkan perencanaan cara kerja rangkaian pada bagian perangkat keras yang terdiri dari 3 bagian yaitu *Input*, *Output*, dan Kontroler/Proses. Pada bagian *Input* terdiri dari berbagai sensor yang akan membaca parameter yang dibutuhkan kemudian data tersebut akan diproses oleh ESP32 lalu akan dikirimkan ke Whatsapp *API* dan disimpan dalam *database.* Terdapat juga relay yang digunakan sebagai kontaktor yang akan mengendalikan kelistrikan. Wiring diagram pada *hardware* ditunjukkan pada gambar 3.5. berikut ini.



Gambar 3.5. Sekema Keseluruhan Hardware

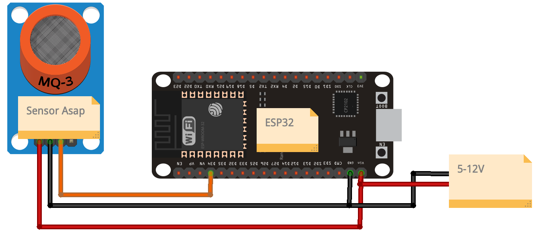
* + 1. **Spesifikasi Perangkat yang akan digunakan**

1. Perangkat menggunakan mikrokontroler ESP32 berbasis 32-bit yang diharapkan dapat melakukan pemrosesan 4 kali lebih cepat dibanding prosesor berbasis 8-bit yang umumnya disematkan pada Arduino.
2. Menggunakan 3 buah sensor yang dapat membaca temperatur, kelembaban, konsentrasi karbon dioksida, dan api untuk meminimalisir terjadinya fake alarm karena Kurangnya data parameter yang dibutuhkan.
3. Menggunakan sensor PZEM004T untuk membaca arus dan tegangan dari penggunaan listrik. Dengan tambahan kumparan yang dapat mendeteksi arus yang mengalir lebih presisi.
4. *Relay* jenis SPDT sebagai saklar menghubung dan pemutus daya, digunakan karena *relay* jenis ini memiliki 2 *state*/keadaan dengan 1 jenis *input Low-High* sebagai penentu saklar nya.
   * 1. **Skema Rangkaian**
5. Skema rangkaian DHT-22



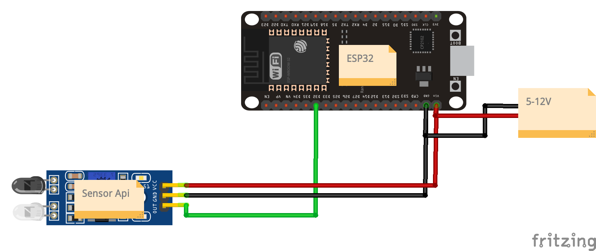
Gambar 3.6. Rangkaian Sensor Suhu DHT-22

1. Skema Rangkaian MQ-2



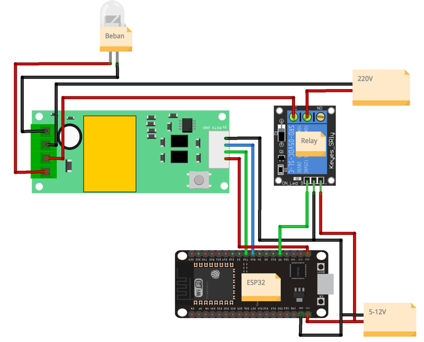
Gambar 3.7. Rangkaian Sensor Asap MQ-2

1. Skema Rangkaian Flame Sensor



Gambar 3.8. Rangkaian Sensor Api

1. Skema Rangkaian Sensor PZEM004T



Gambar 3.9. Rangkaian Sensor Arus & Tegangan dengan Relay

* 1. **Perancangan Listing Program Kontroler**

Sebelum melakukan penelitian penulis akan membuat perancangan terhadap program yang akan dimasukkan ke dalam *Hardware* dengan cara membuat program satu per satu untuk setiap sensor dan aktuator yang digunakan. Hal ini diharapkan dapat menghindari mal fungsi dari seluruh perangkat saat telah digabungkan.

* + 1. **Spesifikasi Program yang akan digunakan**

Bahasa pemrograman yang akan digunakan pada mikrokontroler adalah bahasa pemrograman C yang telah dimodifikasi. Pemilihan bahasa pemrograman C dikarenakan Bahasa C merupakan bahasa tingkat tinggi yang sangat mudah untuk digunakan dan dipahami. Selain itu banyak IDE untuk pemrograman mikrokontroler yang menggunakan bahasa C yang akan diimplementasikan ke dalam sistem tertanam (*embedded system*).

* 1. **Pengujian Sistem**

Pengujian ini dilakukan untuk mengetahui sejauh mana keberhasilan sistem apakah dapat beroperasi sesuai dengan perencanaan. Adapun peralatan konsumsi daya yang akan digunakan selama pengujian adalah sebagai berikut:

Tabel. 3.1. List Peralatan konsumsi daya.

|  |  |  |
| --- | --- | --- |
| **No.** | **Peralatan** | **Konsumsi Daya** |
| 1. | Lampu | 20 Watt |
| 2. | Rice Cooker | 380 Watt |
| 3. | Kipas Angin | 100 Watt |
| 4. | Laptop | 65 Watt |
| 5. | Charger hp | 33 Watt |
| 6. | Dispenser | 250 Watt  6 Watt (stand by) |

Pengujian dilakukan dengan cara mengoperasikan alat yang telah dibuat dengan beberapa skenario seperti perangkat dipindah posisinya, diletakan di dalam ruangan tertutup atau terbuka, dan sebagainya. Proses pengujian ini terbagi menjadi 2 bagian yaitu *hardware* dan *software*.

* + 1. **Pengujian *Hardware***

Pada bagian perangkat keras dilakukan pengujian di setiap sub komponen yang terpasang pada perangkat. Tahapan ini akan dilakukan untuk mengetahui apakah sistem pada mikrokontroler berjalan sesuai dengan perintah yang diberikan, apakah koneksi internet dapat diterima dengan baik, berapa lama perangkat dapat beroperasi saat pertama kali dinyalakan apakah temperatur operasional perangkat masih dalam batas wajar sesuai dengan *datasheet*, Hingga pengujian apakah seluruh komponen yang terpasang dapat berfungsi dengan baik atau tidak.

* + 1. **Pengujian Aplikasi/*Software***

Pengujian pada bagian *software* dilakukan dengan menguji eksekusi sub-sub program dan fungsi dari keseluruhan dari keseluruhan program yang telah dibuat. Hal ini dilakukan untuk mengetahui apakah program yang telah dibuat mengalami kendala *error*, *delay,* atau hasil yang tidak sesuai dengan data yang dimasukkan.

Pengujian ini juga dilakukan untuk mengetahui kompatibilitas aplikasi dengan perangkat yang berbeda, apakah data dapat dimuat dengan benar atau adakah permasalahan lain yang berkaitan dengan aplikasi.

* + 1. **Pengujian Keseluruhan Sistem**

Setelah melakukan pengujian pada masing-masing *hardware* dan *software* dengan hasil sesuai harapan dan tanpa ada kendala, proses selanjutnya adalah melakukan pengujian terhadap perangkat yang telah dirangkai keseluruhannya. Hal ini diharapkan dapat meminimalisir terjadinya kesalahan pada sistem saat diimplementasikan pada keadaan yang sesungguhnya.

* 1. **Skenario Pengujian**

Untuk dapat melakukan analisa dengan mudah maka diperlukan parameter parameter apa saja yang akan digunakan dalam penelitian ini, adapun beberapa parameter yang akan di analisa adalah:

1. Efektivitas biaya dari mikrokontroler yang digunakan

Penelitian ini menggunakan unit pemrosesan dengan arsitektur 32-bit di mana secara teoretis prosesor jenis ini memiliki lebar jalur pemrosesan yang cukup besar dibandingkan dengan mikrokontroler pada umumnya yang masih menggunakan arsitektur 8-bit ataupun 16-bit. Sehingga untuk mengetahui efektivitas penggunaan prosesor jenis ini perlu dilakukan analisa dengan parameter rasio biaya produksi antara prosesor 32-bit dengan 8-bit dan/atau 16-bit.

1. Waktu pemrosesan data yang dibutuhkan

Unit pemrosesan yang digunakan pada penelitian ini memiliki 2 buah inti (*core*) yang dapat menjalankan *task*/tugas berbeda di waktu yang sama. Dengan demikian sangat memungkinkan pemrosesan secara paralel dilakukan untuk mempercepat waktu pemrosesan data pada sistem ini.

1. Akurasi data yang diterima

Data yang diterima dan data yang akan ditampilkan pada sisi pengguna haruslah sama sehingga tidak menimbulkan kerancuan. Maksudnya saat perangkat mendeteksi maka pada aplikasi pengguna harus menampilkan keadaan yang sama pula.

1. *Latency* proses transmisi data

Sistem ini berkaitan erat dengan keamanan sehingga diperlukan data yang *realtime* atau tepat waktu, karena data yang dibutuhkan sudah terlalu lama menjadi tidak terlalu dibutuhkan untuk beberapa hari ke depan,

1. Konsumsi Daya pada sistem

Sistem ini menggunakan komponen elektronik yang tentunya membutuhkan energi listrik untuk mengoperasikannya. Maka perlu dilakukan analisa megenai konsumsi daya yang dibutuhkan oleh sistem ini.

1. Pengujian *User Experience* (UX)

Setiap aplikasi yang digunakan memiliki kesan tersendiri bagi penggunanya, pengujian ini dilakukan dengan cara menggunakan partisipan untuk mencoba menggunakan aplikasi *interface.* Parameter yang akan di analisa adalah usabilitas, interaktivitas, dan *simplicity* dalam penggunaan sistem ini.

Tabel 3.2. Respons Sistem Terhadap kondisi ruangan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Suhu | Asap | Api | Respon Sistem |
| 1 | >40ºC | >300 | True | Kebakaran Terdeteksi |
| 2 | >40ºC | >300 | False | Asap Terdeteksi & Suhu Tinggi |
| 3 | <40ºC | >300 | True | Asap & Api Terdeteksi |
| 4 | <40ºC | >300 | False | Asap Terdeteksi |
| 5 | >40ºC | <300 | True | Suhu Tinggi & Api Terdeteksi |
| 6 | >40ºC | <300 | False | Suhu Tinggi |
| 7 | <40ºC | <300 | False | Keadaan Normal |

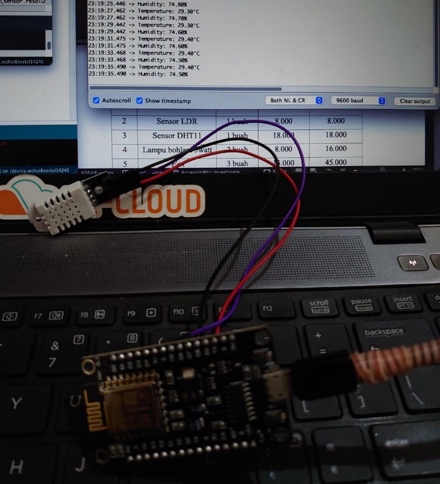
Tabel 3.3. Respon Sistem Terhadap Penggunaan listrik

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Pemakaian (Watt) | Limit (Watt) | Respon Sistem |
| 1 | 400 | 500 | Normal |
| 2 | 600 | 500 | Pemakaian Listrik berlebih |

* 1. **Initial Result**

Pada tahapan ini, akan dilakukan pengujian pada beberapa komponen yang akan digunakan dalam penelitian ini, hal ini dilakukan untuk mengetahui apakah seluruh komponen dapat berfungsi dengan baik atau tidak. Pengujian dilakukan pada beberapa komponen sebagai berikut:

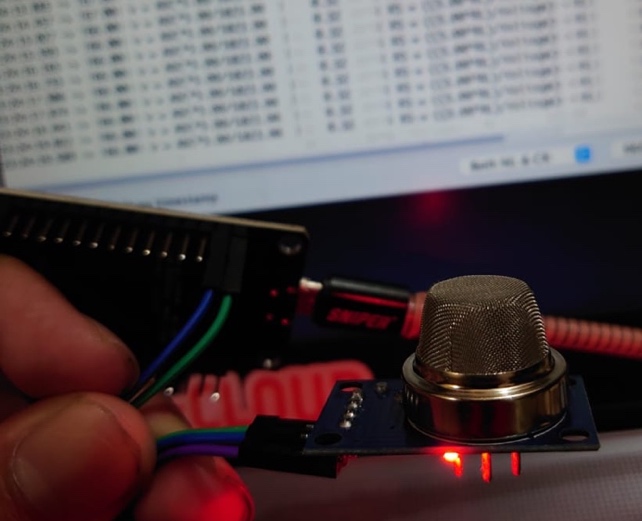
1. Sensor DHT-22



Gambar 3.10. Pengujian Sensor DHT-22

Pengujian pada DHT-22 untuk mendapatkan temperatur dan kelembaban udara pada ruangan. Mikrokontroler akan membaca temperatur dan kelembaban udara melalui sensor DHT yang selanjutnya akan dikirimkan ke *database* *online* untuk disimpan dan dilakukan pengecekan apakah suhu saat ini melebihi *set* *point* atau tidak.

1. Sensor MQ-2



Gambar 3.11. Pengujian Sensor Asap MQ-2

Pengujian sensor asap digunakan untuk mengetahui konsentrasi asap ataupun CO yang terkandung dalam udara.

Pengujian sensor DHT-22 berfungsi untuk mendapatkan temperatur dan kelembaban ruangan secara ideal dengan nilai 29ºC-34ºC dan kelembaban antara 60% - 80%. Sensor MQ-2 berfungsi untuk mendeteksi konsentrasi asap pada udara dalam satuan ppm (*part per million*) di mana MQ-2 dapat mendeteksi konsentrasi asap mulai 200ppm – 1000ppm. Pada Sensor MQ-2 akan dilakukan konversi terhadap *input* dari signal analog menjadi nilai ppm berdasarkan *datasheet*.

Tahapan selanjutnya penulis akan merangkai seluruh komponen yang akan digunakan seperti pada gambar 3.9. dan selanjutnya akan dilakukan pengujian sistem secara keseluruhan sesuai dengan skenario yang telah dituliskan sebelumnya.

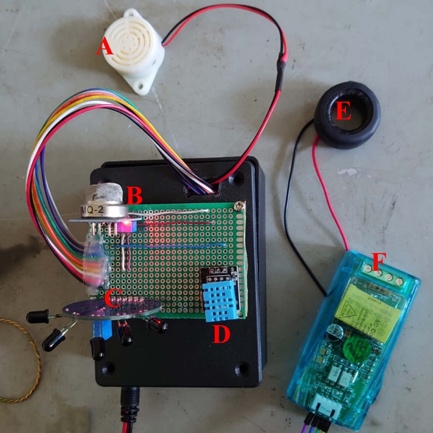
**BAB IV**

**HASIL DAN ANALISA**

Pada bab ini akan dilakukan pembahasan hasil dari perancangan prototipe sistem kontrol dan monitoring yang telah dibuat sesuai dengan metode perancangan yang telah dibahas sebelumnya, dengan demikian diharapkan dapat diketahui kesesuaian, ketelitian dan kesalahan prototipe yang telah dirancang sebelumnya.

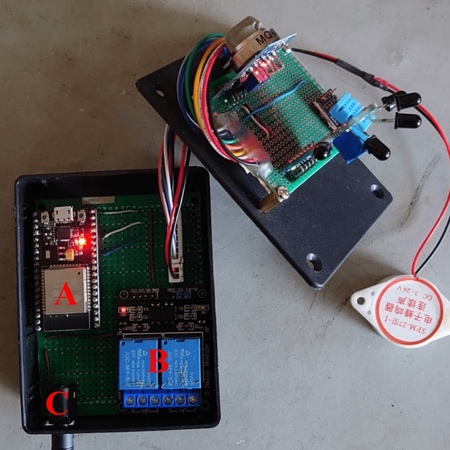
* 1. **Hasil Perancangan Hardware**

Berikut ini adalah hasil perancangan *hardware* dalam sistem kontrol dan monitoring penggunaan listrik menggunakan ESP32 yang telah dibuat, hasil dapat dilihat pada gambar 4.1. di bawah ini.



Gambar 4.1. Tampilan Hardware dari atas

Pada gambar 4.1 terdapat 6 bagian yang merupakan sensor dan *aktuator* pada sistem ini, bagian (A) merupakan *buzzer* yang akan menyala ketika sebuah kondisi terpenuhi seperti penggunaan listrik berlebih, ataupun terdeteksi kebakaran. Bagian (B) merupakan sensor MQ-2 yang dapat mendeteksi asap, bagian (C) merupakan sensor api yang dapat mendeteksi api hingga 300 derajat, bagian (D) merupakan kumparan akan mendeteksi induksi listrik yang akan diproses oleh bagian (F) untuk menghasilkan output data.



Gambar 4.2. Tampilan dalam Hardware

Gambar 4.2. menampilkan bagian dalam sistem yang terdiri dari 3 bagian yaitu (A) mikrokontroler ESP-32S yang berfungsi sebagai pengolahan data input dan output, (B) *relay* 2 channel yang berfungsi sebagai *switch* untuk menyalakan dan mematikan listrik, (C) Power *jack* sebagai input daya untuk dapat mengoperasikan sistem ini.

* 1. **Hasil Pengujian Hardware**
     1. **Pengujian Power Supply**

Suplai daya yang digunakan pada sistem ini memiliki tegangan sebesar 5V DC untuk dapat mengoperasikan sistem ini. Pengujian dilakukan dengan menggunakan multimeter untuk mengetahui sekaligus menghindari kelebihan tegangan oleh adaptor (*overvoltage*) yang dapat merusak komponen yang terpasang.



Gambar 4.3. Hasil pengujian Tegangan

Hasil pengujian tegangan pada gambar 4.3. menampilkan tegangan yang dihasilkan oleh *power* *supply* adalah sebesar 5.14 Volt, tegangan tersebut masih dapat digunakan oleh sistem ini.

* + 1. **Pengujian Mikrokontroler ESP-32S**

Pengujian mikrokontroler dilakukan untuk mengetahui *mapping* pada GPIO yang akan digunakan saat melakukan upload program ke dalam mikrokontroler.



Gambar 4.4. Pengujian Mikrokontroler ESP-32S

Tabel 4.1. Hasil Pengujian GPIO mikrokontroler ESP-32S

|  |  |  |  |
| --- | --- | --- | --- |
| Label | GPIO | Dapat  Digunakan? | Keterangan |
| D0 | 0 | Ya | Aktif HIGH saat booting |
| TX0 | 1 | Tidak | Digunakan Untuk Upload Program |
| D2 | 2 | Ya | Terhubung ke *builtin* LED |
| RX0 | 3 | Tidak | Digunakan untuk upload program |
| D4 | 4 | Ya |  |
| D5 | 5 | Ya | Aktif HIGH saat booting |
| D6 | 6 | Tidak | Terhubung ke memori flash |
| D7 | 7 | Tidak |
| D8 | 8 | Tidak |
| D9 | 9 | Tidak |
| D10 | 10 | Tidak |
| D11 | 11 | Tidak |
| D12 | 12 | Ya | Aktif LOW saat booting |
| D13 | 13 | Ya |  |
| D14 | 14 | Ya |  |
| D15 | 15 | Ya | Aktif HIGH saat booting |
| RX2 | 16 | Ya |  |
| TX2 | 17 | Ya |  |
| D18 | 18 | Ya |  |
| D19 | 19 | Ya |  |
| D21 | 21 | Ya |  |
| D22 | 22 | Ya |  |
| D23 | 23 | Ya |  |
| D25 | 25 | Ya |  |
| D26 | 26 | Ya |  |
| D27 | 27 | Ya |  |
| D32 | 32 | Ya |  |
| D33 | 33 | Ya |  |
| D34 | 34 | Ya | Hanya dapat digunakan  sebagai input |
| D35 | 35 | Ya |
| VP | 36 | Ya |
| VN | 39 | Ya |

Berdasarkan hasil pengujian yang ditampilkan pada tabel 4.1. di atas didapat hasil tidak semua pinout mikrokontroler dapat digunakan, karena sebagian telah dialokasikan untuk keperluan pengembangan dan penyimpanan, sedangkan GPIO 34, 35, 36, 39 tidak dapat digunakan sebagai output, pin tersebut hanya dapat digunakan sebagai input saja baik input digital maupun analog.

* + 1. **Pengujian PZEM-004T**

Untuk mendeteksi penggunaan daya listrik penulis menggunakan PZEM-004T sehingga perlu dilakukan pengujian apakah sensor dapat berfungsi dengan baik.

|  |  |
| --- | --- |
| a. Program membaca tegangan | b. Program membaca arus |

Gambar 4.5. *Listing* program untuk PZEM-004T

Penggunaan listrik akan terbaca oleh PZEM-004T dengan menggunakan *library* yang telah disediakan, program akan membaca beberapa parameter yang ada di antaranya adalah voltase, arus, dan daya.

|  |  |
| --- | --- |
| a. Tampilan *Hardware* | b. Tampilan *Software* |

Gambar 4.6. Pengujian PZEM-004T

Penulis melakukan pengujian pada PZEM-004T tanpa beban dan dengan menggunakan beban berupa solder dengan daya 25-80 Watt, hasil pembacaan sensor akan ditampilkan pada program. Pada gambar 4.6. bagian (a) menampilkan voltase yang terbaca sebesar 228 V dan arus sebesar 0,11 A yang artinya solder menggunakan beban sebesar 25 Watt. Hasil pengukuran ditampilkan pada program seperti ditunjukkan pada gambar 4.6. bagian (b).

|  |  |
| --- | --- |
| a. pengukuran saat *switch* ditekan | b. pengujian keadaan normal |

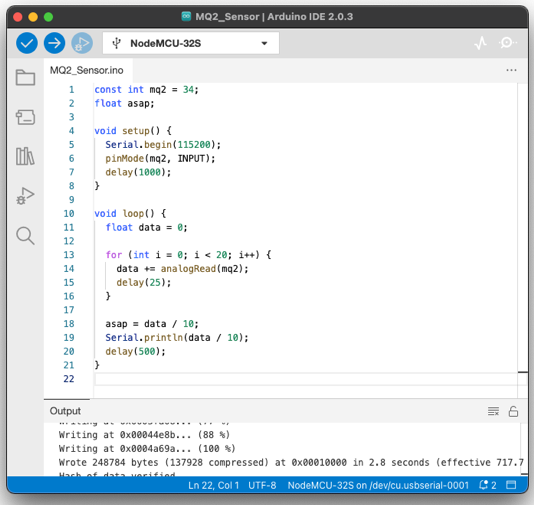
Gambar 4.7. Menguji Akurasi PZEM-004T

Untuk mengetahui akurasi PZEM-004T dilakukan pengujian dengan membandingkan menggunakan multimeter digital untuk mengukur tegangan dan arus yang mengalir. Pengujian dilakukan sebanyak 2 kali menggunakan beban solder dengan daya 25-80 Watt, dalam keadaan normal solder membutuhkan daya sebesar 25 watt dan saat tombol *boost* ditekan daya solder akan meningkat hingga 80 watt.

Berdasarkan gambar 4.7. di atas bagian (a) membutuhkan daya 43 watt dengan tegangan 227 V dan arus 0,19 A saat tombol *boost* ditekan. Namun saat dalam keadaan normal hanya membutuhkan daya 25 watt dengan tegangan sebesar 228 V dan arus 0,11 A.

* + 1. **Pengujian Sensor Asap**

Untuk memastikan apakah sensor asap berfungsi dengan baik dan dapat mendeteksi asap perlu dilakukan pengujian.



Gambar 4.8. *Listing* Program Sensor Asap

Asap akan dideteksi oleh sensor menggunakan sinyal analog, sinyal tersebut yang nantinya dibaca oleh mikrokontroler dan akan dikirimkan ke server secara berkala maupun saat terdeteksi asap.

|  |  |
| --- | --- |
| a. Tampilan *Hardware* | b. Tampilan *Software* |

Gambar 4.9. Asap tidak terdeteksi

Saat tidak mendeteksi asap sensor akan menunjukkan grafik, seperti pada gambar 4.9. di atas grafik berbentuk lurus karena tidak ada asap yang terdeteksi.

|  |  |
| --- | --- |
| a. Tampilan *Hardware* | b. Tampilan *Software* |

Gambar 4.10. Asap terdeteksi

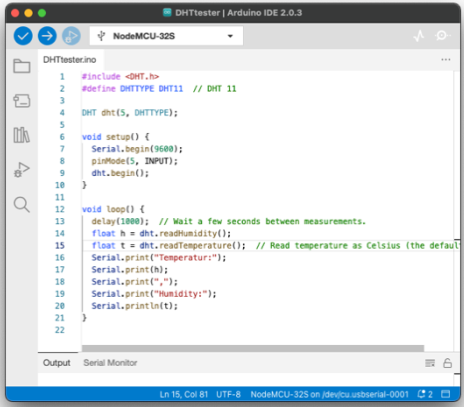
Mikrokontroler akan menampilkan grafik perubahan nilai dari sensor apabila asap terdeteksi seperti gambar 4.10. bagian (b) di mana penulis membakar kertas untuk menguji pembacaan sensor.

Tabe. 4.2. Tabel Hasil Sensor Asap

|  |  |
| --- | --- |
| Sensor  Asap | Keterangan |
| 50 | Tidak Terdeteksi |
| 100 | Tidak Terdeteksi |
| 150 | Tidak Terdeteksi |
| 200 | Tidak Terdeteksi |
| 250 | Tidak Terdeteksi |
| 299 | Tidak Terdeteksi |
| 300 | Asap Terdeteksi |
| 350 | Asap Terdeteksi |
| 400 | Asap Terdeteksi |
| 450 | Asap Terdeteksi |
| 500 | Asap Terdeteksi |
| 550 | Asap Terdeteksi |
| 600 | Asap Terdeteksi |
| 650 | Asap Terdeteksi |
| 700 | Asap Terdeteksi |

* + 1. **Pengujian DHT**

Sensor DHT dapat mendeteksi temperatur ruangan dengan akurasi 2°C, untuk memastikan akurasi tersebut perlu dilakukan pengujian secara langsung dengan pembanding berupa termometer digital.



Gambar 4.11. Listing Program DHT

Sensor DHT dapat mendeteksi temperatur dan kelembaban, namun pada penelitian ini hanya akan menggunakan temperatur saja. Hasil pembacaan sensor akan ditampilkan pada program dalam bentuk grafik dan angka.

|  |  |
| --- | --- |
| a. Tampilan Sensor DHT *Text* | b. Tampilan Sensor DHT Grafik |

Gambar 4.12. Pengujian Sensor DHT

Pada gambar 4.12. di atas merupakan hasil pembacaan sensor DHT yang menampilkan temperatur dengan tingkat *error* sebesar 2°C berdasarkan *datasheet*.

|  |  |
| --- | --- |
|  |  |

Gambar 4.13. Perbandingan Akurasi

Untuk memastikan tingkat akurasi sensor DHT penulis melakukan perbandingan dengan termometer digital, dan didapatkan hasil seperti gambar 4.13. di atas di mana hanya terdapat sedikit perbedaan yang sesuai dengan *datasheet* pada DHT.

(4.1)

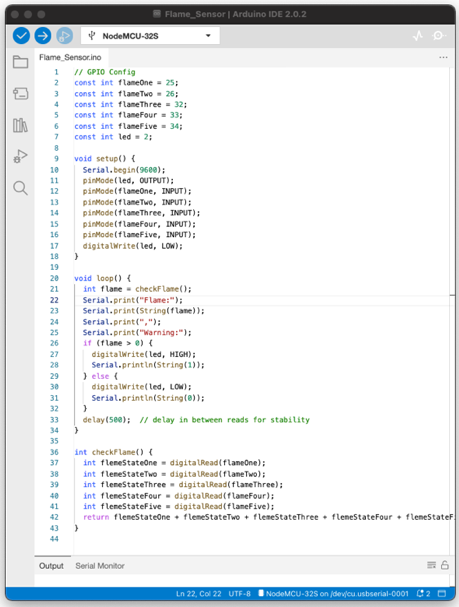
(4.2)

Tabel 4.3. Perbandingan Hasil Sensor Suhu

|  |  |  |  |
| --- | --- | --- | --- |
| Sensor | Thermometer | Error °C | Error % |
| 27.5 | 28.1 | 0.6 | 2.1 |
| 26.3 | 27.4 | 1.1 | 4.0 |
| 27.7 | 27.5 | 0.2 | 0.7 |
| 27.5 | 27.7 | 0.2 | 0.7 |
| 25.3 | 26.8 | 1.5 | 5.6 |
| 26.6 | 26.5 | 0.1 | 0.4 |
| 25.9 | 27.9 | 2.0 | 7.2 |
| 26.8 | 24.4 | 2.4 | 9.8 |
| 27.5 | 25.9 | 1.6 | 6.2 |
| 25.3 | 26.9 | 1.6 | 5.9 |

* + 1. **Pengujian Sensor Api**

Untuk memastikan apakah sensor api dapat mendeteksi api dengan baik penulis melakukan pengujian dengan menggunakan korek api secara langsung. Jenis sensor yang digunakan memiliki 5 Channel yang dapat mendeteksi api dara sudut yang lebih besar.



Gambar 4.14. Listing Program Membaca Sensor Api

Untuk membaca sensor api penulis menggunakan data digital pada sensor api dengan menggunakan 5 channel, jika sensor mendeteksi api sensor akan memberikan logika HIGH yang akan diuban menjadi angka 1, namun apabila tidak mendeteksi adanya api sensor akan memberikan logika LOW yang akan diubah menjadi angka 0. Data dari semua channel akan digabungkan yang selanjutnya akan digunakan sebagai level dari sensor api.

|  |  |
| --- | --- |
|  |  |

Gambar 4.15. Pengujian Tanpa Api

Saat tidak mendeteksi api sensor juga tidak akan menyalakan LED indikatornya sehinggan akan terlihat seperti gambar 4.15. di atas, nilai yang di tampilkan program juga menunjukkan angka 0 yang berarti api tidak terdeteksi.

|  |  |
| --- | --- |
|  |  |

Gambar 4.16. Pengujian Menggunakan Api

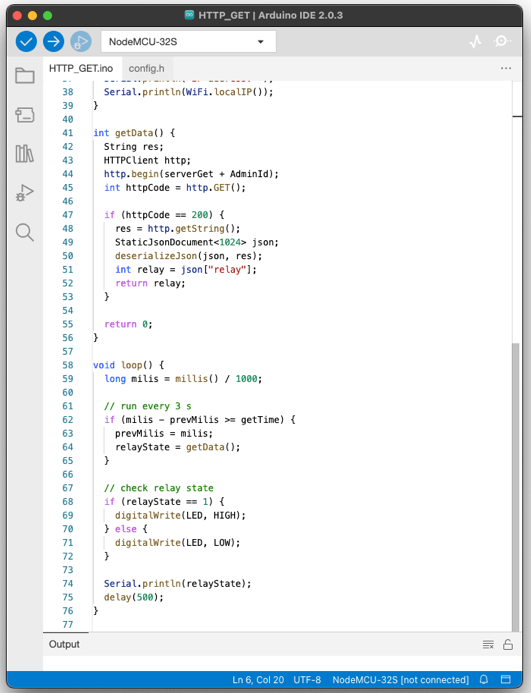
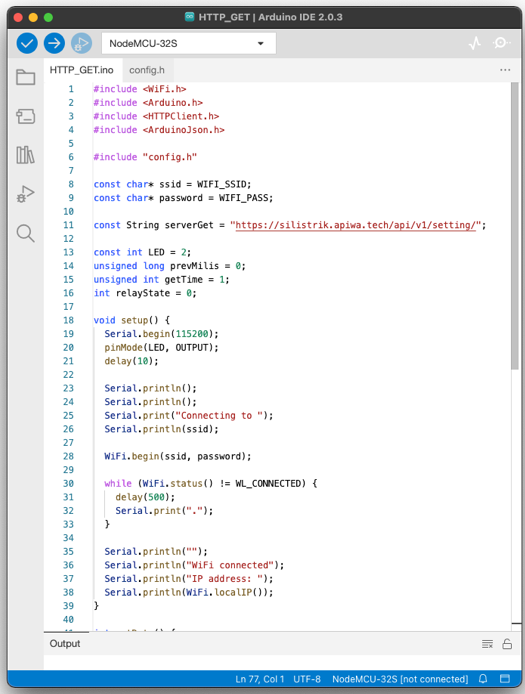
Penulis melakukan pengujian apabila api terdeteksi oleh sensor seperti gambar 4.16. di atas, program menampilkan level dari api yang terdeteksi dan indikator LED yang terpasang pada sensor.

Tabel 4.4. Hasil penbacaan Sensor Api

|  |  |  |
| --- | --- | --- |
| Ukuran Api | Level  Sensor | Keterangan |
| Besar | 5 | Api Terdeteksi |
| Besar | 4 | Api Terdeteksi |
| Besar | 4 | Api Terdeteksi |
| Sedang | 4 | Api Terdeteksi |
| Sedang | 3 | Api Terdeteksi |
| Kecil | 2 | Api Terdeteksi |
| Kecil | 1 | Api Terdeteksi |

* 1. **Hasil Pengujian *Software***
     1. **Pengujian GET data dari database**

Untuk mendapatkan nilai dari status relay dan juga set poin yang telah ditetapkan maka diperlukan pengambilan data dari database secara online, metode yang digunakan adalah dengan http GET di mana mikrokontroler akan melakukan komunikasi http menggunakan method GET dengan server.

****

Gambar 4.10. Listing Program http GET

Dari program di atas akan didapatkan respons dari server berupa data json (*javascrip object notation*) yang akan *diencode* untuk didapatkan nilainya. Nilai yang didapatkan akan disimpan ke dalam variabel yang telah dibuat.

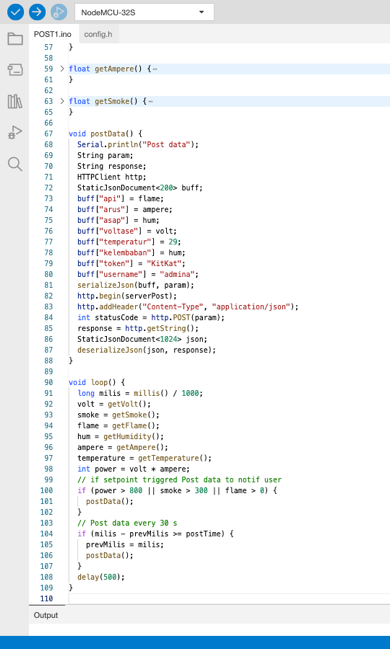
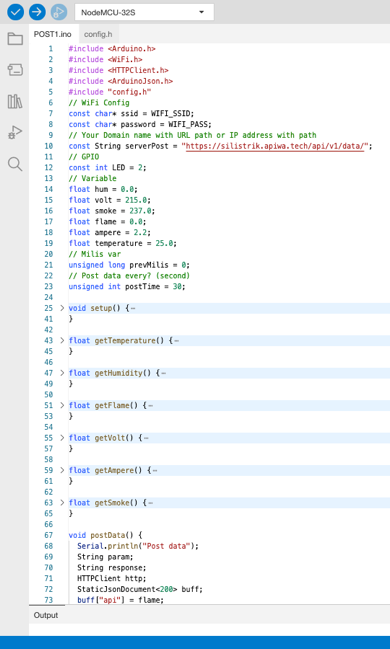


Gambar 4.11. Respons http GET dari server

Respon yang akan diberikan oleh server berupa nilai daya yang terpasang, setpoin temperatur maksimal, nilai asap maksimal, limit penggunaan daya dan status relay saat ini.

* + 1. **Pengujian POST data ke Database**

Berbeda dengan http method sebelumnya yang menggunakan GET untuk mengirimkan data sensor ke server database digunakan method yang berbeda yaitu http POST method.



Gambar 4.12. Program POST data ke Server

Pada gambar 4.12. program akan mengirimkan data ke database dimulai dengan melakukan koneksi ke jaringan WiFi, menghubungi server, mengambil data dari semua sensor yang selanjutnya akan dikirimkan ke server dalam bentuk json.

* + 1. **Pengujian Perintah ke Bot WhatsApp**

Untuk memastikan apakah Bot WhatsApp berjalan dengan baik, akan dilakukan pengujian mengirimkan perintah ke Bot, perintah yang akan diberikan adalah meminta data saat ini, meminta status sensor saat ini, menyalakan dan mematikan relay.

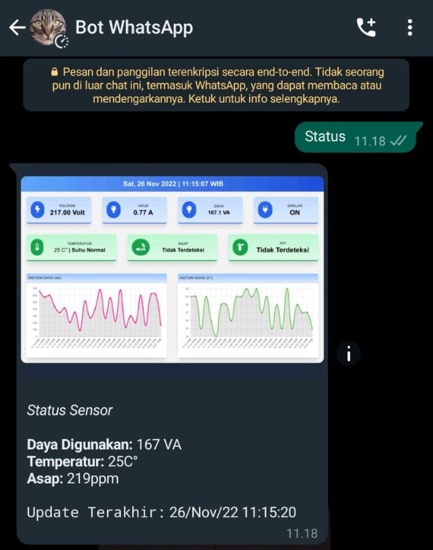
1. Meminta data sensor melalui Bot WhatsApp



Gambar 4.13. Meminta data sensor

Pada gambar 4.13. saat bot menerima pesan text “**data**”, bot akan memproses pesan tersebut kemudian akan mengirimkan pesan balasan berisi data sensor.

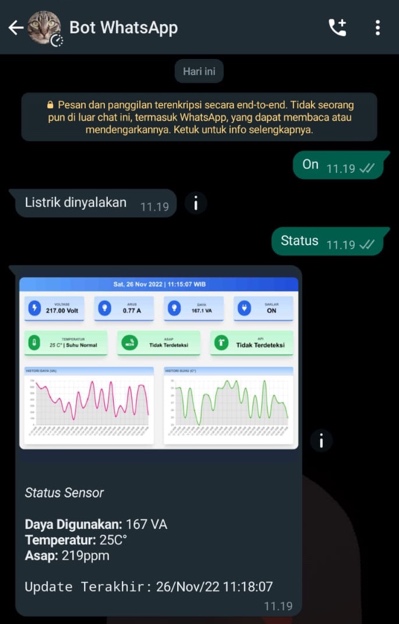
1. Meminta Status Sensor melalui Bot WhatsApp



Gambar 4.14. Meminta Status sensor

Pada gambar 4.14. saat bot menerima pesan text dengan ini “**status**”, bot akan memproses pesan tersebut dan mengembalikan pesan balasan berupa status sensor beserta dengan ilustrasi berupa gambar dan pesan text nya.

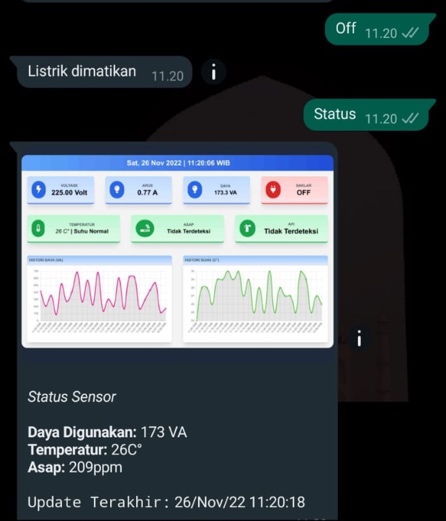
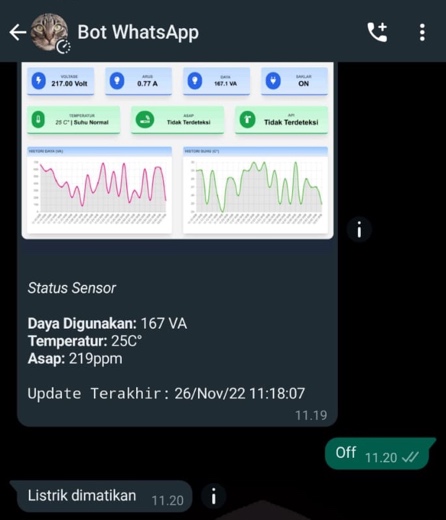
1. Menyalakan Relay



Gambar 4.15. Menyalakan Relay melalui bot WhatsApp

Pada gambar 4.15. saat bot menerima pesan dengan ini “**on**”, bot akan memproses pesan tersebut dan mengubah state relay menjadi **HIGH** yang berarti menyalakan kelistrikan pada sistem kemudian akan memberi pesan balasan seperti pada gambar.

1. Mematikan Relay

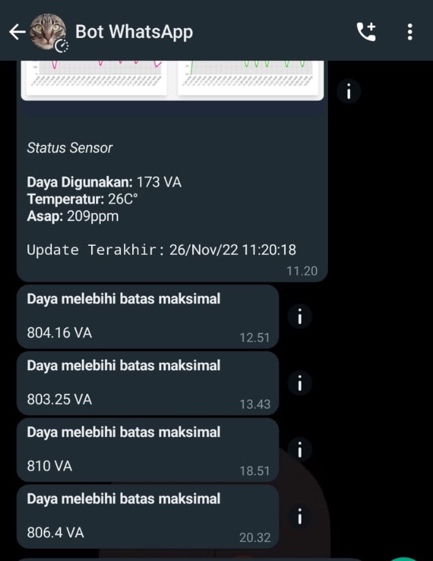


Gambar 4.16. Mematikan Relay melalui WhatsApp

Pada gambar 4.15. saat bot menerima pesan dengan ini “**off**”, bot akan memproses pesan tersebut dan mengubah state relay menjadi **LOW** yang berarti mematikan kelistrikan pada sistem kemudian akan memberi pesan balasan seperti pada gambar.

* + 1. **Pengujian Notifikasi Dari Bot WhatsApp**

Saat suatu keadaan terdeteksi bot akan mengirimkan notifikasi kepada pengguna melalui WhatsApp sehingga pengguna dapat langsung mengetahui informasi tersebut.



Gambar 4.17. Notifikasi Melalui WhatsApp

Pengguna mendapatkan notifikasi sesuai dengan setpoin yang telah ditetapkan sebelumnya, pada gambar 4.17. pengguna mengatur limit penggunaan listrik sebesar 800 VA, sehingga apabila penggunaan listrik melebihi 800 VA bot akan mengirimkan notifikasi kepada pengguna.

* + 1. **Pengujian Respons Bot WhatsApp**

Untuk mengukur waktu yang dibutuhkan oleh bot dalam memproses pesan dilakukan pengujian dengan mengirim pesan beberapa kali sehingga didapatkan hasil seperti Tabel 4.2.

Tabel 4.2. Hasil pengukuran waktu respon

| Waktu | Penerima/  Penerima | Pesan | *Type* | Status | Keterangan |
| --- | --- | --- | --- | --- | --- |
| 27/11/22  00:27:54 | 6282 31\*\*  \*\*99 | Listrik  dimatikan | Pesan  Keluar | *Success* | *Message sent* |
| 27/11/22  00:27:53 | 6282 31\*\*  \*\*99 | off | Pesan  Masuk | *Success* | Pesan Masuk |
| 27/11/22  00:27:50 | 6282 31\*\*  \*\*99 | Listrik  dinyalakan | Pesan  Keluar | *Success* | *Message sent* |
| 27/11/22  00:27:48 | 6282 31\*\*  \*\*99 | on | Pesan  Masuk | *Success* | Pesan Masuk |
| 27/11/22  00:27:48 | 6282 31\*\*  \*\*99 | Listrik  dimatikan | Pesan  Keluar | *Success* | *Message sent* |
| 27/11/22  00:27:47 | 6282 31\*\*  \*\*99 | off | Pesan  Masuk | *Success* | Pesan Masuk |
| 27/11/22  00:27:46 | 6282 31\*\*  \*\*99 | Listrik  dinyalakan | Pesan  Keluar | *Success* | *Message sent* |
| 27/11/22  00:27:45 | 6282 31\*\*  \*\*99 | on | Pesan  Masuk | *Success* | Pesan Masuk |
| 27/11/22  00:27:44 | 6282 31\*\*  \*\*99 | Listrik  dimatikan | Pesan  Keluar | *Success* | *Message sent* |
| 27/11/22  00:27:43 | 6282 31\*\*  \*\*99 | off | Pesan  Masuk | *Success* | Pesan Masuk |
| 27/11/22  00:27:40 | 6282 31\*\*  \*\*99 | Listrik  dinyalakan | Pesan  Keluar | *Success* | *Message sent* |
| 27/11/22  00:27:37 | 6282 31\*\*  \*\*99 | on | Pesan  Masuk | *Success* | Pesan Masuk |

Berdasarkan tabel 4.2. di atas waktu rata-rata yang diperlukan untuk memproses pesan oleh bot adalah 2 detik dengan waktu tercepat 1 detik sedangkan waktu maksimal sebesar 3 detik. Hal tersebut bergantung dengan koneksi data ke server bot dan juga server database.

* 1. **Hasil Pengujian Keseluruhan**

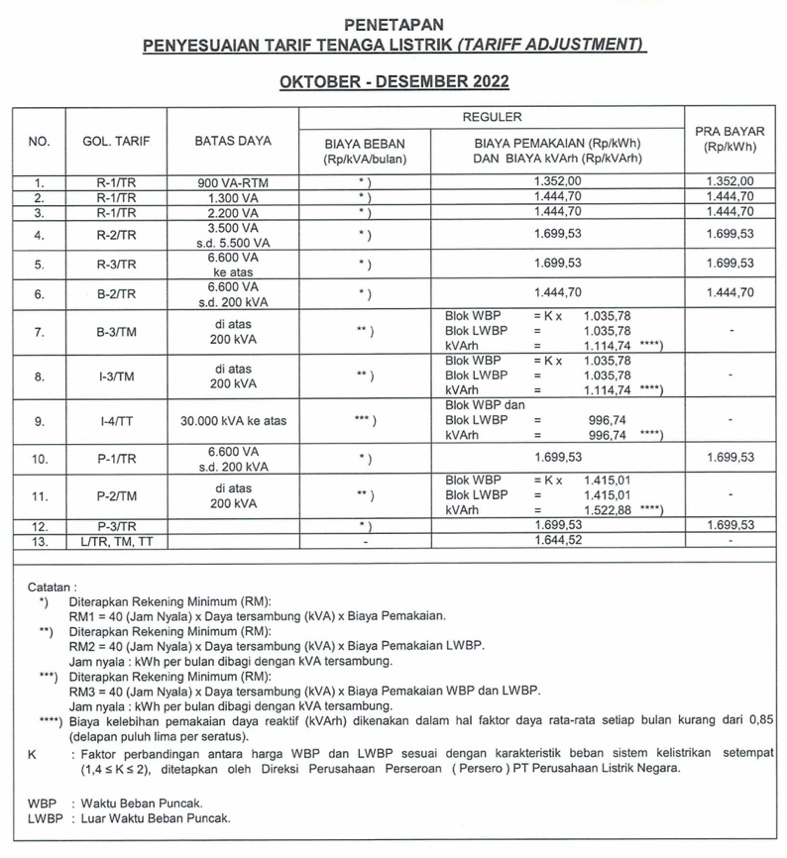
Dalam mengumpulkan data hasil pengujian keseluruhan penulis melakukukan pengambilan data salama 10 hari mulai dari 12 Oktober 2022 hingga 22 Oktober 2022. Hal ini dilakukan untuk mengetahui penggunaan listrik rata-rata harian.

* + 1. **Observasi Penggunaan Listrik**

Tabel 4.3. Observasi penggunaan Listrik

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Tanggal** | **Gambar** | **Sisa Listrik** | **Penggunaan Harian** | **Harga** |
| 1 | 12-11-22 |  | 18.20 kWh | 0 kWh | Rp. 0 |
| 2 | 13-11-22 |  | 17.76 kWh | 0.44 kWh | Rp. 594.88 |
| 3 | 14-11-22 |  | 16.95 kWh | 0.81 kWh | Rp. 1,095.12 |
| 4 | 15-11-22 |  | 16.30 kWh | 0.65 kWh | Rp. 878.80 |
| 5 | 16-11-22 |  | 14.62 kWh | 1.68 kWh | Rp. 2,271.36 |
| 6 | 17-11-22 |  | 11.89 kWh | 2.73 kWh | Rp. 3,690.96 |
| 7 | 18-11-22 |  | 10.70 kWh | 1.19 kWh | Rp. 1,608.88 |
| 8 | 19-11-22 |  | 8.13 kWh | 2.57 kWh | Rp. 3,474.64 |
| 9 | 20-11-22 |  | 6.70 kWh | 1.43 kWh | Rp. 1,933.36 |
| 10 | 21-11-22 |  | 5.05 kWh | 1.65 kWh | Rp. 2,230.8 |
| 11 | 22-11-22 |  | 3.81 kWh | 1.24 kWh | Rp. 1,676.48 |
| **Min** | | | | 0.44 kWh | Rp. 595 |
| **Max** | | | | 2.73 kWh | Rp. 3,691 |
| **Rata-Rata** | | | | 1.439 kWh | Rp. 1,946 |

Pengamatan yang dilakukan selama 10 hari mendapatkan hasil penggunaan listrik selama pengamatan didapatkan penggunaan daya terendah 0.44 kWh, dengan harga Rp. 595, daya tertinggi 2.73 kWh dengan biaya Rp. 3,691 dan rata-rata 1.439 kWh dengan harga Rp. 1,946. Dalam penentuan harga listrik diambil melalui website resmi tarif *adjustmen* PT. PLN 900 VA periode Oktober sampai Desember 2022 seperti pada gambar 4.18 berikut ini.



Gambar 4.18. Penyesuaian Tarif Listrik[22]

Untuk harga listrik dengan daya terpasang 900 VA memiliki harga Rp. 1,353 per kWh.

* + 1. **Hasil Pengujian Keseluruhan alat**

|  |  |
| --- | --- |
| (a) | (b) |

Gambar 4.19. Tampilan alat saat pemasangan

Alat yang telah dirangkai secara keseluruhan akan dipasangkan langsung ke listrik utama seperti pada gambar 4.19. pada gambar bagian (b) merupakan pengujian deteksi api menggunakan mancis di mana terlihat lampu indikator menyala.



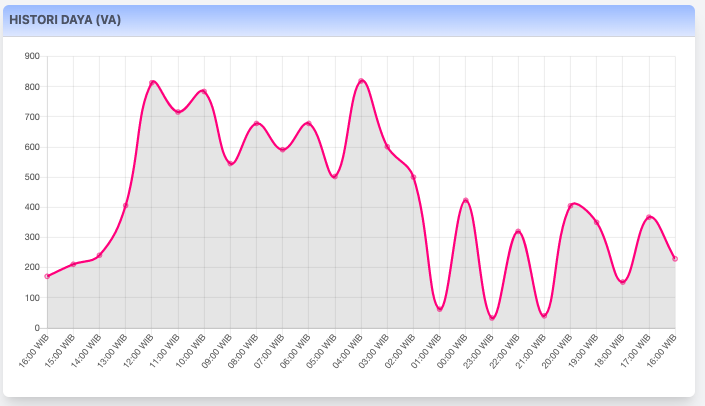
Gambar 4.20. Tampilan Status Sensor pada Website

Status dari sensor akan dapat dilihat secara langsung melalui website yang akan secara otomatis di perbarui. Pada website pengguna hanya dapat melihat status dari sensor dan tidak dapat mengontrol kelistrikan.



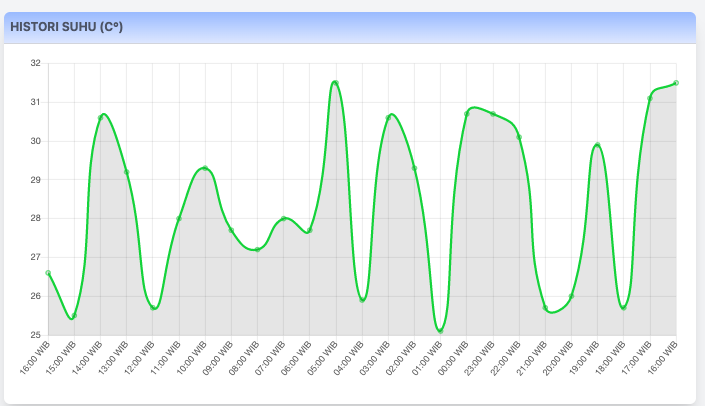
Gambar 4.21. Tampilan log notifikasi

Pengguna juga dapat melihat log atau riwayat notifikasi yang dikirimkan oleh sistem melalui website seperti yang ditunjukkan pada gambar 4.21. melalui menu ini pengguna dapat melihat kapan notifikasi dikirimkan dan apa pesan yang dikirimkan.



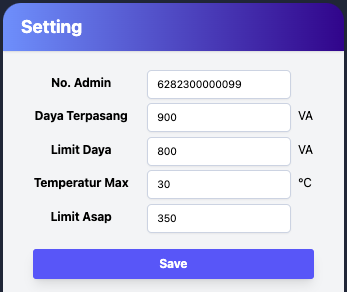
Gambar 4.22. Histori penggunaan daya listrik

Pengguna dapat melihat riwayat penggunaan daya listrik untuk mengetahui kapan waktu yang membutuhkan konsumsi listrik yang tinggi.



Gambar 4.23. Riwayat temperatur ruangan

Riwayat temperatur tuangan juga akan dapat dilihat melalui website dalam bentuk grafik yang mudah untuk dipahami.



Gambar 4.24. Menu pengaturan pada Website

Notifikasi dari sistem juga akan dikirimkan ke nomor telah didaftarkan pada website, nomor yang didaftarkan akan menerima notifikasi secara langsung apabila terdapat keadaan yang memicu trigger seperti asap terdeteksi, temperatur tinggi, maupun penggunaan daya listrik yang melebihi set point.



Gambar 4.25. Notifikasi Pada sisi Pengguna

Gambar 4.25. menunjukkan pengguna mendapatkan pesan notifikasi dari sistem bahwa sensor mendeteksi temperatur yang lebih tinggi melebihi setpoint yang telah ditetapkan.

Tabel 4.3. Hasil Penghujian Keseluruhan Alat pada 01 Januari 2023

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Waktu | Voltase | Arus | Daya | Suhu | Api | Asap | Notifikasi |
| 21:00:02 | 219 | 3.2 | 690 | 31 | 0 | 295 | Temperatur Tinggi |
| 20:00:02 | 222 | 2.4 | 524 | 31 | 0 | 214 | Temperatur Tinggi |
| 19:00:03 | 221 | 3.6 | 787 | 30 | 0 | 254 | Temperatur Tinggi |
| 18:00:02 | 221 | 1.6 | 349 | 27 | 0 | 265 | Aman |
| 17:00:03 | 219 | 1.9 | 414 | 26 | 0 | 269 | Aman |
| 16:00:03 | 220 | 1.0 | 229 | 32 | 0 | 274 | Temperatur Tinggi |
| 15:00:03 | 220 | 1.7 | 367 | 31 | 0 | 307 | Temperatur Tinggi |
| 14:00:03 | 220 | 0.69 | 152 | 26 | 0 | 200 | Aman |
| 13:00:03 | 220 | 1.6 | 350 | 30 | 0 | 273 | Aman |
| 12:00:04 | 220 | 1.8 | 405 | 26 | 0 | 206 | Aman |
| 11:00:04 | 222 | 0.18 | 40 | 26 | 0 | 300 | Aman |
| 10:00:07 | 222 | 1.4 | 320 | 30 | 0 | 220 | Temperatur Tinggi |
| 09:00:06 | 219 | 0.15 | 33 | 31 | 0 | 248 | Temperatur Tinggi |
| 08:00:09 | 219 | 1.9 | 423 | 31 | 0 | 257 | Temperatur Tinggi |
| 07:00:05 | 220 | 0.28 | 62 | 25 | 0 | 224 | Aman |
| 06:00:07 | 222 | 2.3 | 500 | 29 | 0 | 251 | Aman |
| 05:00:05 | 221 | 2.7 | 601 | 31 | 0 | 200 | Temperatur Tinggi |
| 04:00:17 | 222 | 3.7 | 819 | 26 | 0 | 309 | Penggunaan Listrik Tinggi |
| 03:00:05 | 222 | 2.3 | 502 | 32 | 0 | 226 | Temperatur Tinggi |
| 02:00:03 | 221 | 3.1 | 678 | 28 | 0 | 211 | Aman |
| 01:00:03 | 222 | 2.7 | 591 | 28 | 0 | 253 | Aman |
| 00:00:04 | 221 | 3.1 | 678 | 27 | 0 | 293 | Aman |

Berdasarkan tabel 4.3. sistem akan secara otomatis mengirimkan pesan berdasarkan kondisi yang dikirimkan oleh sensor. Saat penggunaan daya melebihi 800 VA sistem akan mengirimkan notifikasi “Daya melebihi batas maksimal 819.18 VA” dan “Temperatur Tinggi 30.6 C” saat temperatur yang terbaca melebihi set point yang ditetapkan.



Gambar 4.26. Notifikasi dari sistem

Sistem mengirimkan notifikasi secara langsung sesaat setelah sensor mengirimkan data yang memantik kondisi tertentu seperti temperatur tinggi, asap terdeteksi, ataupun penggunaan daya yang melebihi setpoint.

* 1. **Analisa Keseluruhan**

Hasil implementasi sistem peringatan kebakaran serta kontrol dan monitoring penggunaan listrik yang terintegrasi dengan WhatsApp dapat memberitahukan kondisi terkini dari ruangan yang dipasangkan sistem ini. Sistem juga dapat mengendalikan kelistrikan jarak jauh melalui WhatsApp yang dapat mengurangi penggunaan listrik. Sistem juga akan mengirimkan notifikasi apabila terjadi kondisi tertentu kepada pengguna dalam waktu yang singkat melalui pesan WhatsApp.

Pengguna dapat melakukan monitoring secara manual dengan mengirimkan perintah melalui WhatsApp untuk mendapatkan data terkini ruangan yang dibaca oleh sensor. Data yang dikirimkan oleh sensor terbagi menjadi 2 jenis yaitu data teks dan data gambar yang dapat dengan mudah dimengerti oleh pengguna yang masih awam. Sistem akan menyimpan data yang dikirimkan oleh sensor sehingga pengguna dapat melihat riwayat apabila dibutuhkan suatu saat nanti.

**BAB V**

**PENUTUP**

* 1. **Kesimpulan**

Berdasarkan penelitian yang dilakukan pada Sistem Peringatan Kebakaran serta Kontrol dan Monitoring Penggunaan Listrik Rumah Tangga Berbasis IoT Terintegrasi dengan WhatsApp penulis menyimpulkan.

1. Hasil dari sistem yang telah diimplementasikan dapat membantu mengurangi penggunaan listrik.
2. Sistem dapat mengirimkan notifikasi ke pengguna secara otomatis dalam waktu yang singkat yaitu antara 3 sampai 10 detik.
3. Sistem dapat mengendalikan kelistrikan dari jarak jauh melalui pesan WhatsApp.
   1. **Saran**

Untuk menyempurnakan penelitian ini terdapat beberapa saran yang dapat dipertimbangkan dalam penelitian selanjutnya sebagai berikut.

1. Pengiriman data dari sensor dilakukan secara sekuensial sehingga data yang didapatkan berupa realtime data tanpa delay, dimana sistem ini menggunakan delay 30 detik dalam setiap pengiriman data tanpa adanya triger.
2. Sistem memberikan notifikasi kepada pengguna apabila dalam waktu tertentu sensor tidak mengirimkan data, sehingga pengguna dapat melakukan pengecekan pada sensor apakah terdapat masalah.
3. Bot WhatsApp terkadang memiliki responyang lambat dikarenakan koneksi internet yang tidak stabil.
4. Melakukan autentikasi ataupun filter terhadap perintah yang dikirimkan oleh pengguna, sehingga tidak setiap pengguna dapat mengirimkan pesan tertentu.

**DAFTAR PUSTAKA**

[1] Badan Pusat Statistik, “Konsumsi Listrik per Kapita (MWH/Kapita),” 2020. Diakses: Jun 28, 2022. [Daring]. Available: https://www.bps.go.id/indicator/7/1156/1/konsumsi-listrik-per-kapita.html

[2] D. A. Siregar, “Rancang Bangun Alat Pengawas Pemakaian Listrik Rumah Tangga Menggunakan Sistem Internet Of Things (IoT) Terintegrasi Web Dan Telegram (Studi Kasus: Perumahan Green Panam Regency),” Universitas Islam Negeri Sultan Syarif Kasim, Pekanbaru, 2020.

[3] R. M. Harianja, “Rancang Bangun Monitoring Energi Listrik Pada Rumah Tangga Secara Iot Berbasis Mikrokontroler Atmega 328.” Diakses: Apr 13, 2022. [Daring]. Available: https://repositori.usu.ac.id/handle/123456789/32370

[4] M. Juhan, D. Suryanto, dan T. Rijanto, “Rancang Bangun Alat Pencatat Biaya Pemakaian Energi Listrik Pada Kamar Kos Menggunakan Modul Global System For Mobile Communications(Gsm) 800L Berbasis Arduino Uno.” Diakses: Apr 13, 2022. [Daring]. Available: https://jurnalmahasiswa.unesa.ac.id/index.php/JTE/article/view/25600

[5] B. T. P. Guritno, “Deteksi Kebakaran Rumah Tinggal Berbasis WiFi,” Universitas Sanata Dharma, Yogyakarta, 2017. Diakses: Apr 08, 2022. [Daring]. Available: https://repository.usd.ac.id/12260/2/135114024\_full.pdf

[6] M. Imamuddin dan Z. Zulwisli, “Sistem Alarm Dan Monitoring Kebakaran Rumah Berbasis Nodemcu Dengan Komunikasi Android,” *Voteteknika (Vocational Teknik Elektronika dan Informatika)*, vol. 7, no. 2, hlm. 40–45, Jun 2019, doi: 10.24036/VOTETEKNIKA.V7I2.104093.

[7] “Rancang Bangun Alat Pendeteksi Asap Kebakaran Menggunakan Sensor Mq-2 Berbasis Arduino Uno.” Diakses: Apr 13, 2022. [Daring]. Available: https://repositori.usu.ac.id/handle/123456789/3810

[8] J. Mulyono dan E. Apriaskar, “Simulasi Alarm Kebakaran Menggunakan Sensor Mq-2, Falme Sensor Berbasis Mikrokontroler Arduino,” vol. 14, no. 1, hlm. 16–25, 2021, [Daring]. Available: http://journal.stekom.ac.id/index.php/elkom■page16

[9] “Rancang Bangun Prototype Alat Pendeteksi Kebakaran Menggunakan Arduino Uno Dilengkapi Pemadam Dan Notifikasi Sms Gateway”, Diakses: Apr 13, 2022. [Daring]. Available: https://ejournal.um-sorong.ac.id/index.php/insect/article/download/1280/702

[10] Wikipedia, “Pengendali mikro,” *Wikipedia, Ensiklopedia Bebas*. 2020.

[11] Espressif, “ESP32 Series Datasheet,” Shanghai, 2020.

[12] Espressif, “The Internet of Things with ESP32,” *Compact Surface-Mount PCB Modules*, 2016. http://esp32.net/images/\_resources/tiny/Espressif\_ESP-WROOM-32\_Shield\_FCC.svg (diakses Mei 19, 2020).

[13] Wikipedia, “ESP32,” *Wikipedia, Ensiklopedia Bebas*. 2020.

[14] PZEM, “PZEM-004T V3.0 User Manual.” Diakses: Apr 08, 2022. [Daring]. Available: https://innovatorsguru.com/wp-content/uploads/2019/06/PZEM-004T-V3.0-Datasheet-User-Manual.pdf

[15] LastMinuteEngineers, “How DHT11 DHT22 Sensors Work & Interface With Arduino.”

[16] Wikipedia, “Relai,” *Wikipedia, Ensiklopedia Bebas*. 2020.

[17] Last Minute Engineers, “Interface One Channel Relay Module with Arduino,” 2020. https://lastminuteengineers.com/one-channel-relay-module-arduino-tutorial/ (diakses Mei 21, 2020).

[18] Wikipedia, “Database,” *Wikipedia, the free encyclopedia*. 2020.

[19] Oracle Corporation, “Database,” *Oracle Corporation*, 2020. https://www.oracle.com/database/what-is-database.html (diakses Mei 21, 2020).

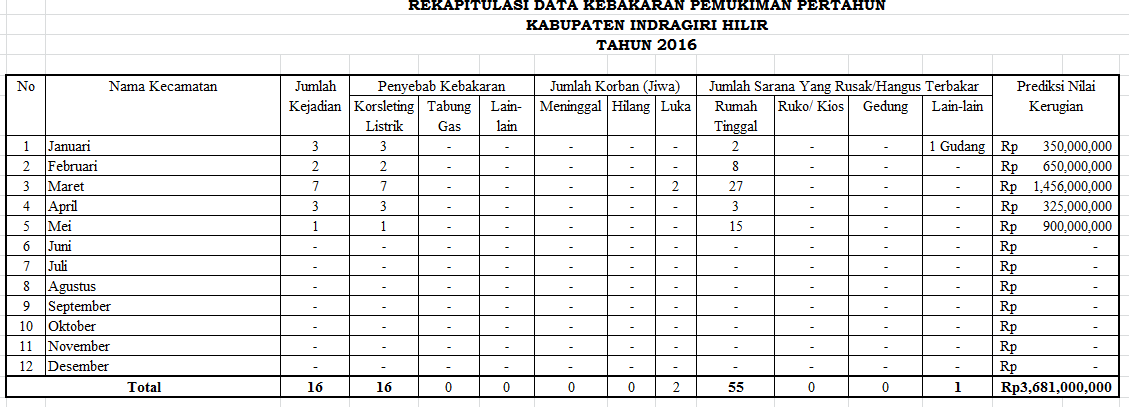
[20] Katadata., “Indonesia Pengguna WhatsApp Terbesar Ketiga di Dunia | Databoks,” *Statista*, 2021. https://databoks.katadata.co.id/datapublish/2021/11/23/indonesia-pengguna-whatsapp-terbesar-ketiga-di-dunia (diakses Apr 05, 2022).

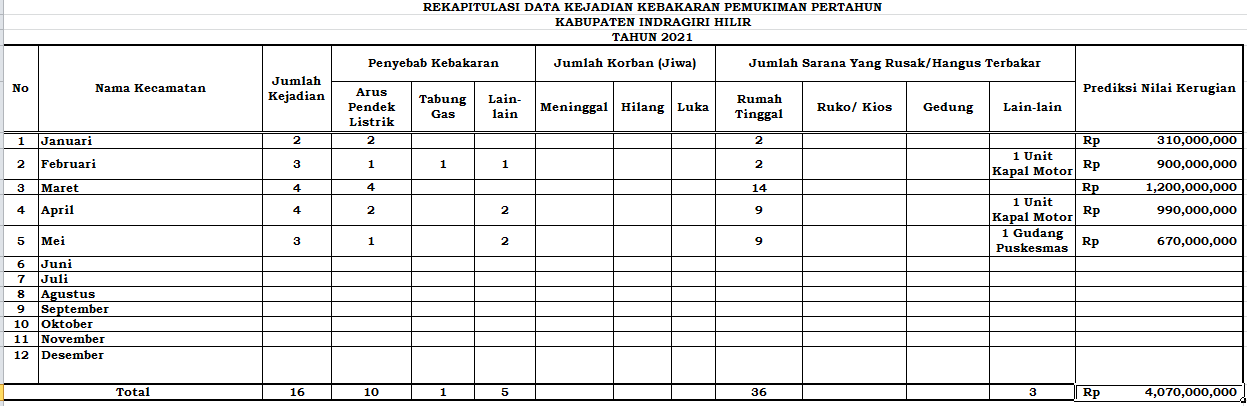
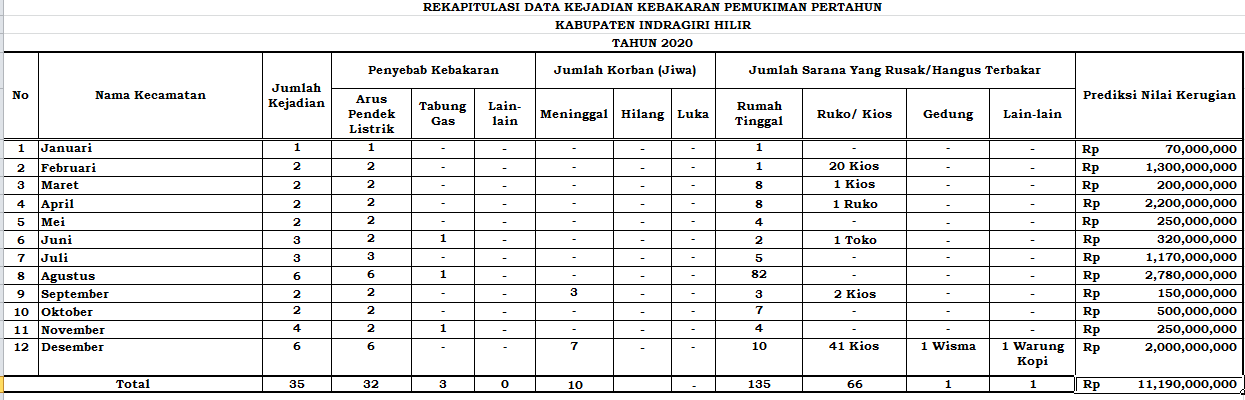
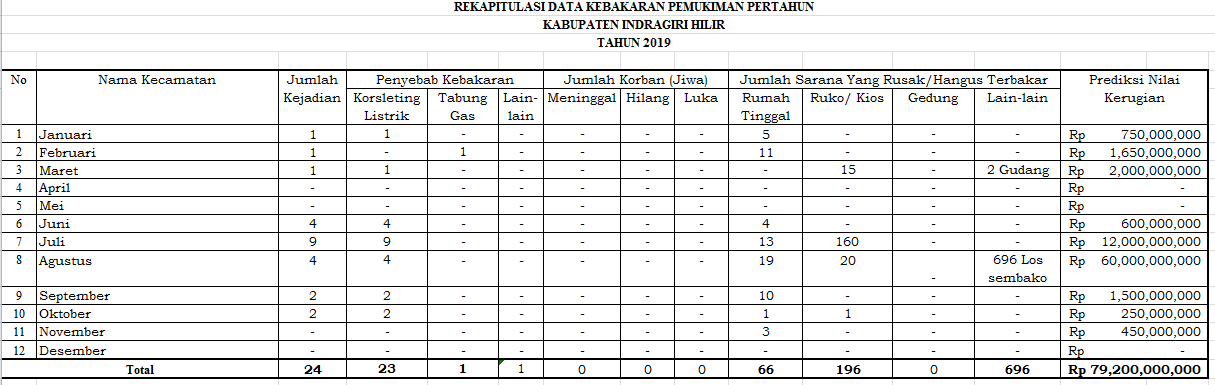
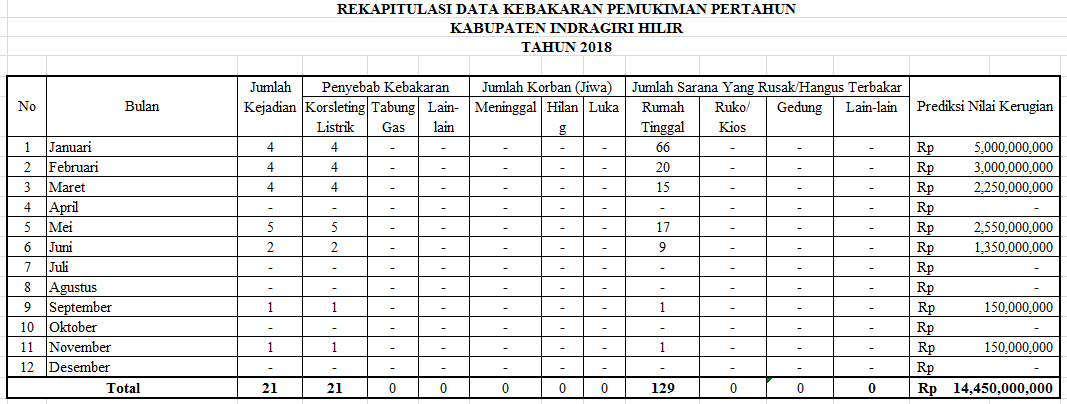
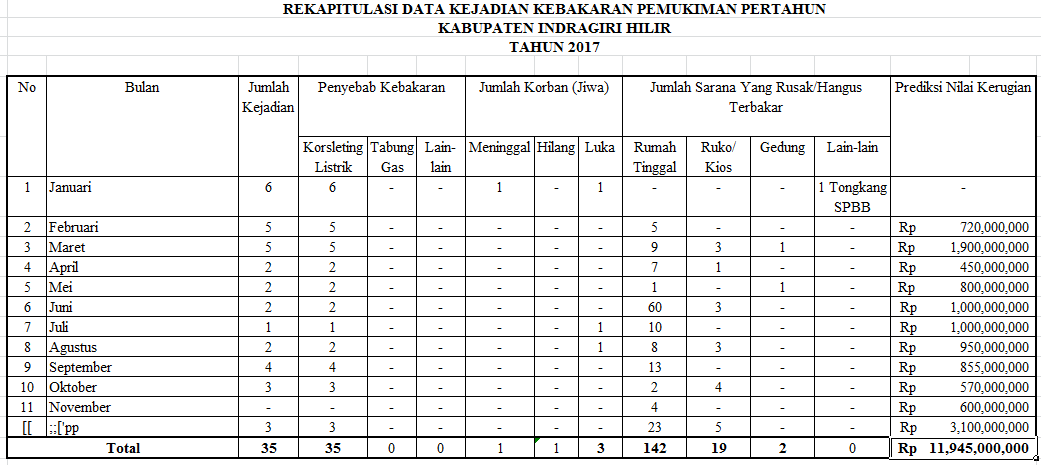
[21] WhatsApp. Inc, “WhatsApp Help Center - About sending messages on WhatsApp.” https://faq.whatsapp.com/general/account-and-profile/about-sending-messages-on-whatsapp/?lang=en (diakses Apr 08, 2022).

[22] PT Perusahaan Listrik Negara, “Tarif Adjustment.” https://web.pln.co.id/pelanggan/tarif-tenaga-listrik/tariff-adjustment (diakses Des 01, 2022).

**Lampiran I**

**Data kebakaran 2016-2021 indragiri hilir**





**Lampiran II**

**Program**

Sensor dht

|  |
| --- |
| #include <Adafruit\_Sensor.h>  #include <DHT.h>  #include <DHT\_U.h>  #define DHTPIN 2 // Digital pin connected to the DHT sensor  #define DHTTYPE DHT11 // DHT 11  DHT\_Unified dht(DHTPIN, DHTTYPE);  uint32\_t delayMS;  void setup() {  Serial.begin(9600);  dht.begin();  pinMode(DHTPIN, INPUT);  sensor\_t sensor;  // Set delay between sensor readings based on sensor details.  delayMS = sensor.min\_delay / 1000;  }  void loop() {  // Delay between measurements.  delay(delayMS);  // Get temperature event and print its value.  sensors\_event\_t event;  dht.temperature().getEvent(&event);  if (isnan(event.temperature)) {  Serial.println(F("Error reading temperature!"));  } else {  Serial.print(F("Temperature: "));  Serial.print(event.temperature);  Serial.println(F("°C"));  }  } |

sensor asap

|  |
| --- |
| const int mq2 = 34;  float asap;  void setup() {  Serial.begin(115200);  pinMode(mq2, INPUT);  delay(1000);  }  void loop() {  data = analogRead(mq2);  Serial.print("Smoke:");  Serial.println(data);  delay(500);  } |

sensor api

|  |
| --- |
| // GPIO Config  const int flameOne = 25;  const int flameTwo = 26;  const int flameThree = 32;  const int flameFour = 33;  const int flameFive = 34;  const int led = 2;  void setup() {  Serial.begin(9600);  pinMode(led, OUTPUT);  pinMode(flameOne, INPUT);  pinMode(flameTwo, INPUT);  pinMode(flameThree, INPUT);  pinMode(flameFour, INPUT);  pinMode(flameFive, INPUT);  digitalWrite(led, LOW);  }  void loop() {  int flame = checkFlame();  Serial.print("Flame:");  Serial.print(String(flame));  Serial.print(",");  Serial.print("Warning:");  if (flame > 0) {  digitalWrite(led, HIGH);  Serial.println(String(1));  } else {  digitalWrite(led, LOW);  Serial.println(String(0));  }  delay(500); // delay in between reads for stability  }  int checkFlame() {  int flemeStateOne = digitalRead(flameOne);  int flemeStateTwo = digitalRead(flameTwo);  int flemeStateThree = digitalRead(flameThree);  int flemeStateFour = digitalRead(flameFour);  int flemeStateFive = digitalRead(flameFive);  return flemeStateOne + flemeStateTwo + flemeStateThree + flemeStateFour + flemeStateFive;  } |

sensor pzem Voltase

|  |
| --- |
| #include <PZEM004Tv30.h>  #define PZEM\_RX\_PIN 16  #define PZEM\_TX\_PIN 17  #define PZEM\_SERIAL Serial2  PZEM004Tv30 pzem(PZEM\_SERIAL, PZEM\_RX\_PIN, PZEM\_TX\_PIN);  void setup() {  Serial.begin(115200);  }  void loop() {  // Read the data from the sensor  float voltage = pzem.voltage();  // Check if the data is valid  if (isnan(voltage)) {  Serial.println("Error reading voltage");  } else {  Serial.println(voltage);  }  delay(2000);  } |

sensor pzem arus

|  |
| --- |
| #include <PZEM004Tv30.h>  PZEM004Tv30 pzem(Serial2, 16, 17);  void setup() {  Serial.begin(115200);  }  void loop() {  // Read the data from the sensor  float current = pzem.current();  // Check if the data is valid  if (isnan(current)) {  Serial.println("Error reading current");  } else {  Serial.println(current);  }  delay(2000);  } |

kirim data

|  |
| --- |
| #include <WiFi.h>  #include <Arduino.h>  #include <HTTPClient.h>  #include <ArduinoJson.h>  #include "config.h"  const char\* ssid = WIFI\_SSID;  const char\* password = WIFI\_PASS;  const String admin = AdminId;  const String token = TokenId;  const String serverPost = "https://silistrik.apiwa.tech/api/v1/data";  unsigned long postMilis = 0;  unsigned int postTime = 30;  void setup() {  Serial.begin(115200);  wifiConnect();  }  void wifiConnect() {  Serial.print("\n\nConnecting to ");  Serial.println(ssid);  WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  Serial.println("\nWiFi connected");  Serial.print("IP address: ");  Serial.println(WiFi.localIP());  }  void postData() {  Serial.println("Post Data");  String param;  String res;  HTTPClient http;  StaticJsonDocument<200> buff;  buff["api"] = 0;  buff["arus"] = 1;  buff["asap"] = 250;  buff["voltase"] = 222;  buff["temperatur"] = 29;  buff["kelembaban"] = 80;  buff["token"] = token;  buff["username"] = admin;  serializeJson(buff, param);  http.begin(serverPost);  http.addHeader("Content-Type", "application/json");  int statusCode = http.POST(param);  res = http.getString();  StaticJsonDocument<1024> json;  deserializeJson(json, res);  Serial.println(statusCode);  Serial.println(res);  Serial.println(param);  }  void loop() {  long milis = millis() / 1000;  if (milis - postMilis >= postTime) {  postMilis = milis;  postData();  }  delay(500);  } |

get data

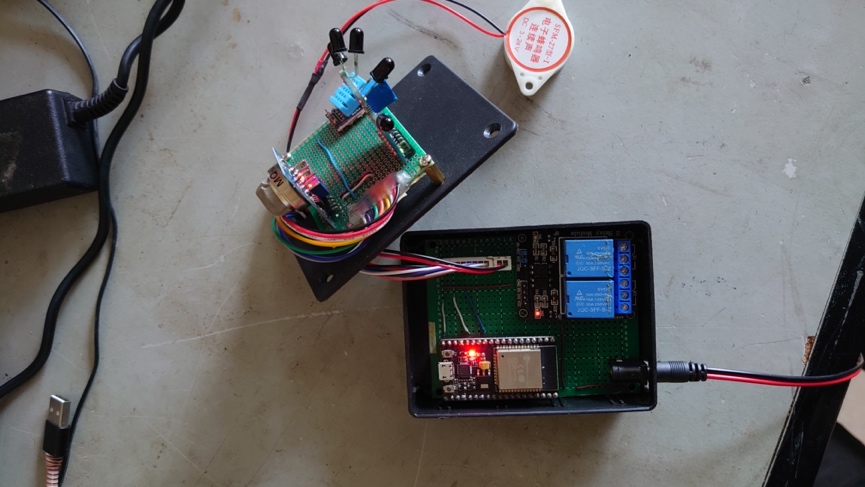
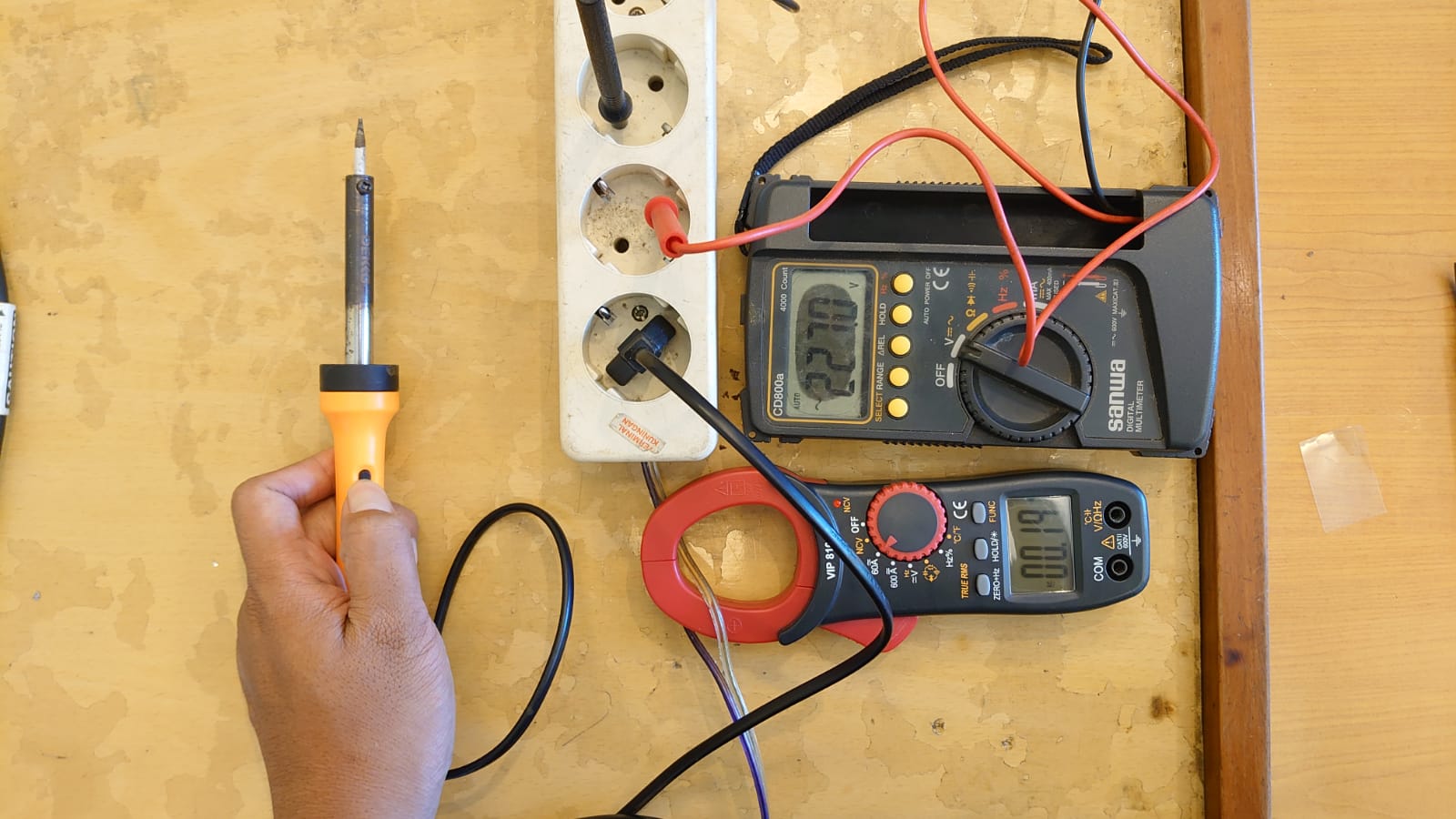
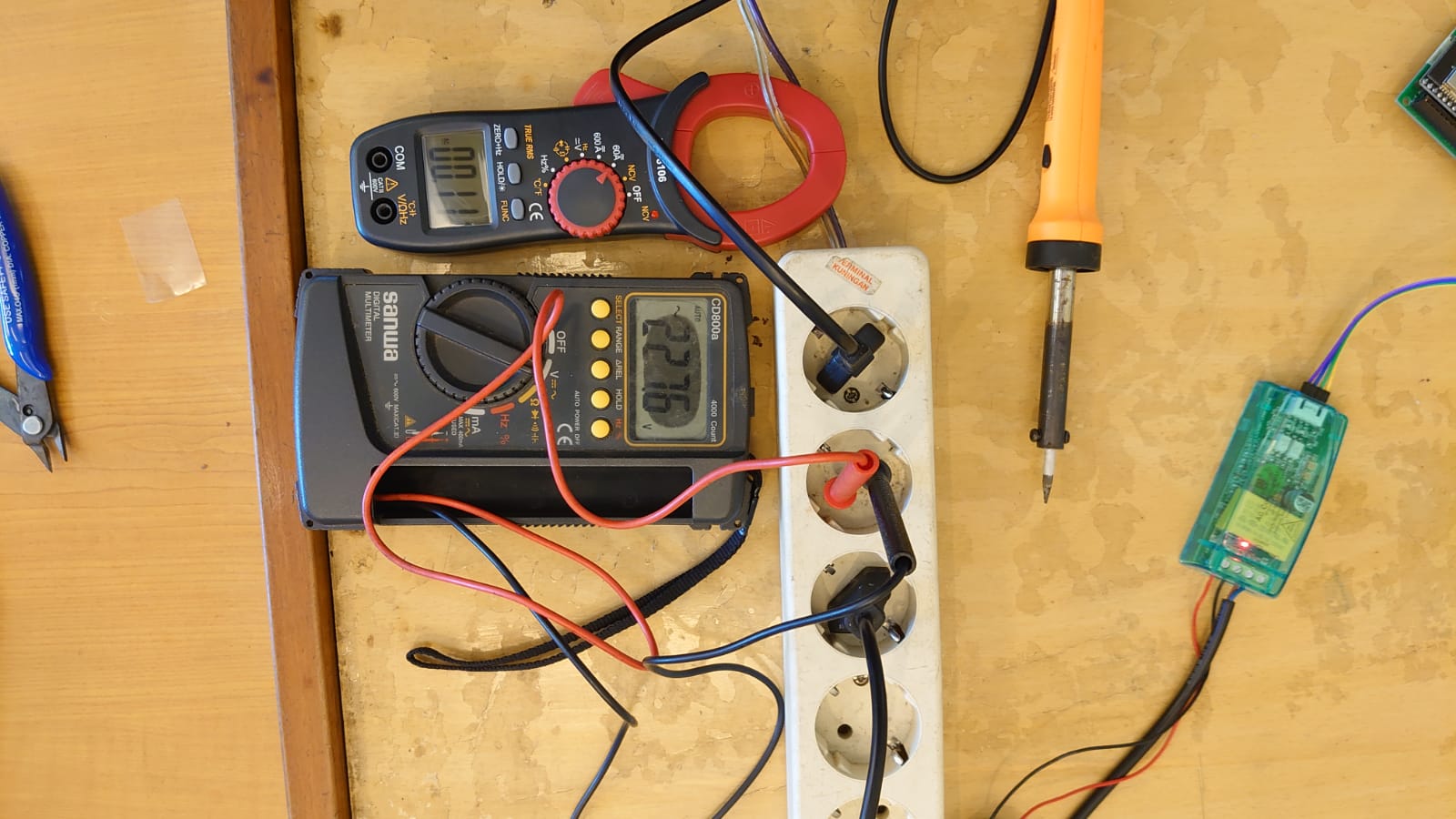
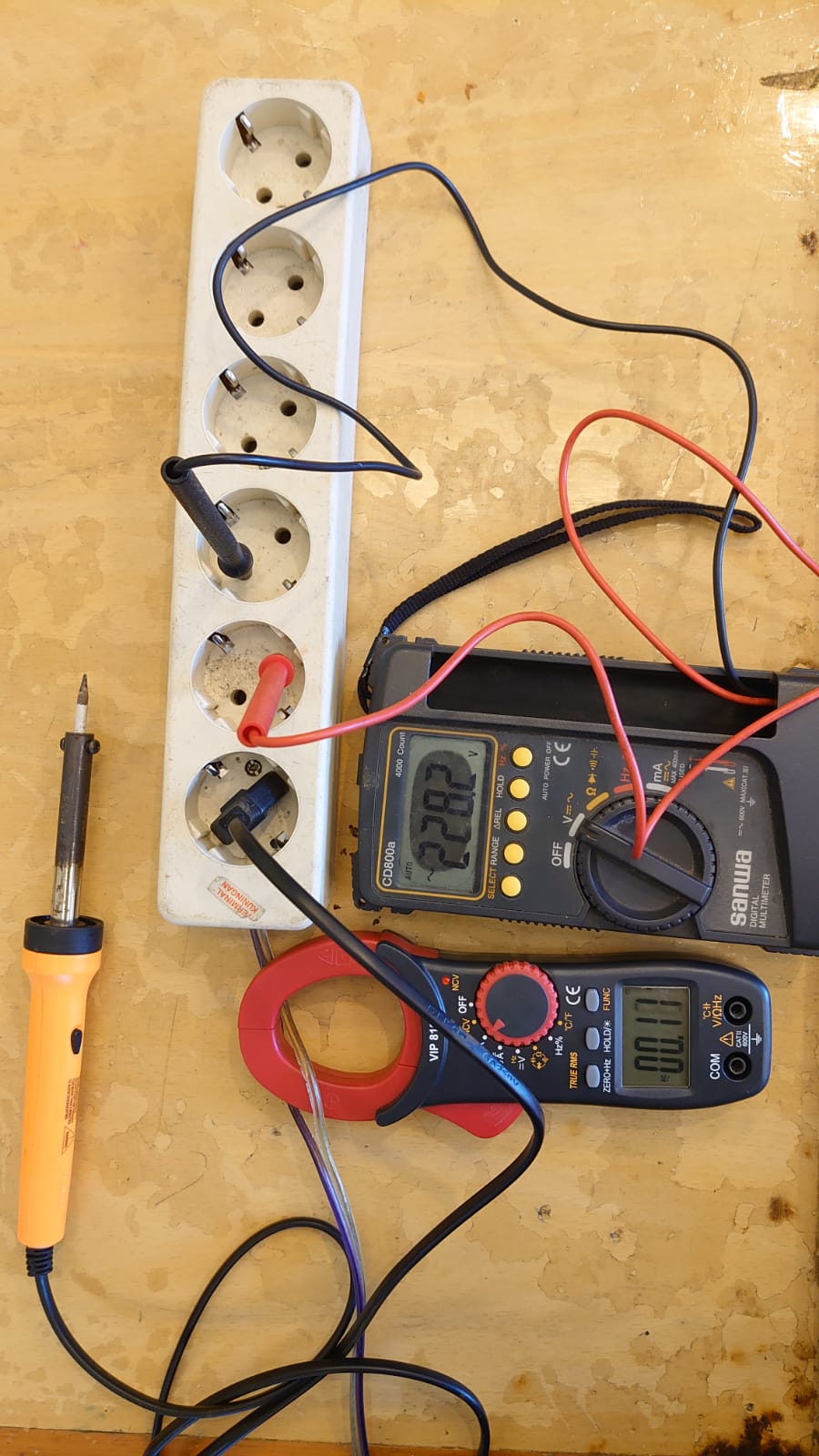
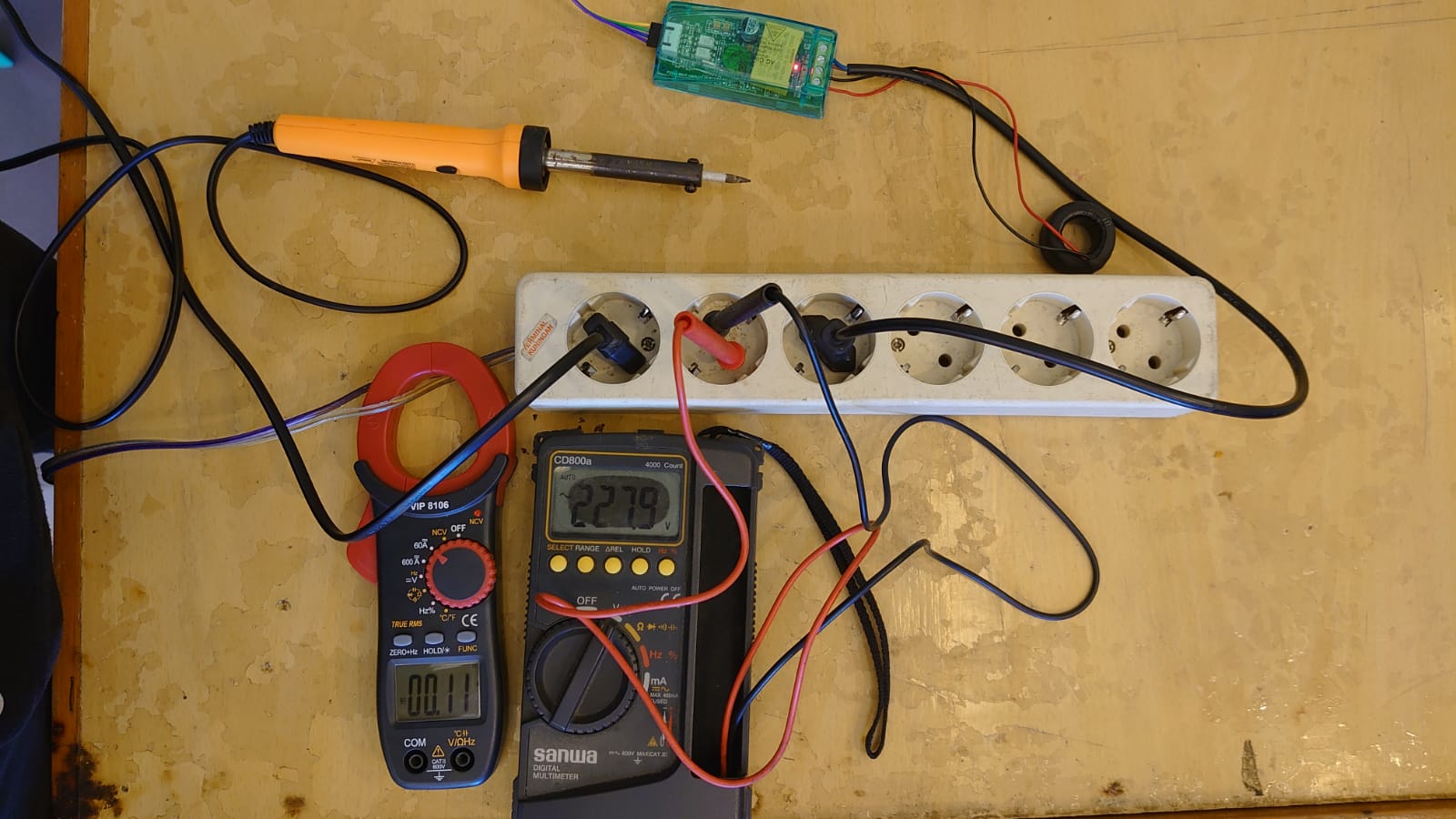
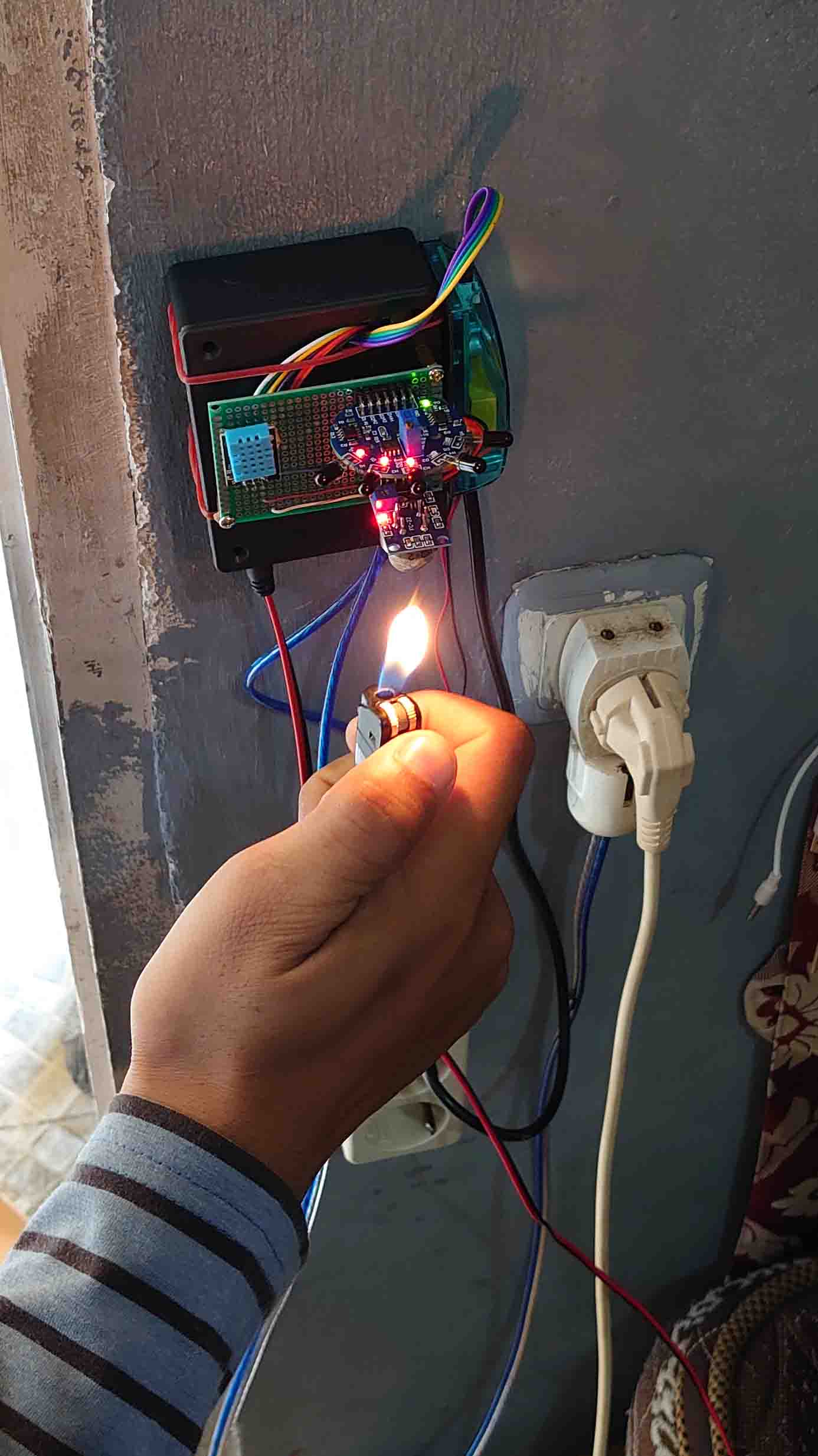
|  |
| --- |
| #include <WiFi.h>  #include <Arduino.h>  #include <HTTPClient.h>  #include <ArduinoJson.h>  #include "config.h"  const char\* ssid = WIFI\_SSID;  const char\* password = WIFI\_PASS;  const String serverGet = "https://silistrik.apiwa.tech/api/v1/setting/";  const int LED = 2;  const int relay = 14;  unsigned long prevMilis = 0;  unsigned int getTime = 1;  int relayState = 0;  void setup() {  Serial.begin(115200);  pinMode(LED, OUTPUT);  pinMode(relay, OUTPUT);  delay(10);  Serial.println();  Serial.println();  Serial.print("Connecting to ");  Serial.println(ssid);  WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  Serial.println("");  Serial.println("WiFi connected");  Serial.println("IP address: ");  Serial.println(WiFi.localIP());  }  int getData() {  String res;  HTTPClient http;  http.begin(serverGet + AdminId);  int httpCode = http.GET();  if (httpCode == 200) {  res = http.getString();  StaticJsonDocument<1024> json;  deserializeJson(json, res);  int relay = json["relay"];  return relay;  }  return 0;  }  void loop() {  long milis = millis() / 1000;  // run every 3 s  if (milis - prevMilis >= getTime) {  prevMilis = milis;  relayState = getData();  }  // check relay state  if (relayState == 1) {  digitalWrite(LED, HIGH);  digitalWrite(relay, HIGH);  } else {  digitalWrite(LED, LOW);  digitalWrite(relay, LOW);  }  Serial.println(relayState);  delay(500);  } |

program keseluruhan

|  |
| --- |
| #include <DHT.h>  #include <WiFi.h>  #include <Arduino.h>  #include <HTTPClient.h>  #include <ArduinoJson.h>  #include <PZEM004Tv30.h>  #include "config.h"  #define DHTTYPE DHT11 // DHT 11  const char\* ssid = WIFI\_SSID;  const char\* password = WIFI\_PASS;  const String admin = AdminId;  const String token = TokenId;  const String serverPost = "https://silistrik.apiwa.tech/api/v1/data";  const String serverGet = "https://silistrik.apiwa.tech/api/v1/setting/";  DHT dht(5, DHTTYPE);  PZEM004Tv30 pzem(Serial2, 16, 17);  // Deklarasi GPIO  const int led = 2;  const int buzz = 0;  const int mq2 = 34;  const int dhtPin = 5;  const int flameOne = 33;  const int flameTwo = 25;  const int flameThree = 26;  const int flameFour = 27;  const int flameFive = 15;  const int relayOne = 14;  const int relayTwo = 13;  // Deklarasi Variabel  float smoke = 200;  float current = 0;  float voltage = 220;  float humidity = 75;  float temperature = 20;  int flame = 0;  int power = 200;  int relayState = 0;  // Setting data  int powerMax = 800;  int smokeMax = 300;  int temperatureMax = 30;  // Milis config  unsigned int getTime = 1;  unsigned int postTime = 30;  unsigned long getMilis = 0;  unsigned long postMilis = 0;  void setup() {  Serial.begin(115200);  configGPIO();  wifiConnect();  dht.begin();  }  void wifiConnect() {  Serial.print(F("\n\nConnecting to "));  Serial.println(ssid);  WiFi.begin(ssid, password);  while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }  Serial.println(F("\nWiFi connected"));  Serial.print(F("IP address: "));  Serial.println(WiFi.localIP());  }  void configGPIO() {  pinMode(mq2, INPUT); // MQ2 as Input  pinMode(dhtPin, INPUT); // DHT as Input  pinMode(flameOne, INPUT);  pinMode(flameTwo, INPUT);  pinMode(flameThree, INPUT);  pinMode(flameFour, INPUT);  pinMode(flameFive, INPUT);  pinMode(led, OUTPUT);  pinMode(buzz, OUTPUT);  pinMode(relayTwo, OUTPUT);  pinMode(relayOne, OUTPUT);  digitalWrite(led, LOW);  digitalWrite(buzz, LOW);  digitalWrite(relayTwo, HIGH);  digitalWrite(relayOne, HIGH);  }  float getCurrent() {  // Read the data from the sensor  float ampere = pzem.current();  // Check if the data is valid  if (isnan(ampere)) {  Serial.println(F("Error reading current"));  return current;  } else {  return ampere;  }  }  float getVoltage() {  // Read the data from the sensor  float volt = pzem.voltage();  // Check if the data is valid  if (isnan(volt)) {  Serial.println(F("Error reading Voltage"));  return voltage;  } else {  return volt;  }  }  float getTemperature() {  float t = dht.readTemperature();  if (isnan(t)) {  t = random(2500, 2800) / 100.0;  }  return t;  }  float getHumidity() {  float h = dht.readHumidity(); // read humidity from sensor  if (isnan(h)) {  h = random(6500, 7200) / 100.0;  }  return h;  }  float getSmoke() {  float data;  float input = 0;  for (int i = 0; i < 20; i++) {  input += analogRead(mq2);  delay(25);  }  data = input / 10;  return data;  }  int getFlame() {  int flemeStateOne = digitalRead(flameOne);  int flemeStateTwo = digitalRead(flameTwo);  int flemeStateThree = digitalRead(flameThree);  int flemeStateFour = digitalRead(flameFour);  int flemeStateFive = digitalRead(flameFive);  return flemeStateOne + flemeStateTwo + flemeStateThree + flemeStateFour + flemeStateFive;  }  int getData() {  String res;  HTTPClient http;  http.begin(serverGet + AdminId);  int httpCode = http.GET();  if (httpCode == 200) {  res = http.getString();  StaticJsonDocument<1024> json;  deserializeJson(json, res);  int relay = json["relay"];  powerMax = json["limit"];  smokeMax = json["asap"];  temperatureMax = json["tmax"];  return relay;  }  return 0;  }  void postData() {  Serial.println(F("Post Data"));  String param;  String res;  HTTPClient http;  StaticJsonDocument<200> buff;  buff["api"] = flame;  buff["asap"] = smoke;  buff["token"] = token;  buff["arus"] = current;  buff["username"] = admin;  buff["voltase"] = voltage;  buff["kelembaban"] = humidity;  buff["temperatur"] = temperature;  serializeJson(buff, param);  http.begin(serverPost);  http.addHeader("Content-Type", "application/json");  int statusCode = http.POST(param);  res = http.getString();  StaticJsonDocument<1024> json;  deserializeJson(json, res);  Serial.print(statusCode + " ");  Serial.println(res);  }  void checkData() {  power = current \* voltage;  if ((flame > 0) || (power > powerMax) || (temperature > temperatureMax) || (smoke > smokeMax)) {  digitalWrite(buzz, HIGH);  postData();  } else {  digitalWrite(buzz, LOW);  }  }  void printData() {  Serial.print(F("Relay:"));  Serial.println(relayState);  Serial.print(F("Flame:"));  Serial.println(flame);  Serial.print(F("Smoke:"));  Serial.println(smoke);  Serial.print(F("Temperatur:"));  Serial.println(temperature);  Serial.print(F("Humidity:"));  Serial.println(humidity);  Serial.print(F("Volt:"));  Serial.println(voltage);  Serial.print(F("Ampere:"));  Serial.println(current);  }  void loop() {  Serial.println("=======================");  long milis = millis() / 1000;  flame = getFlame(); // Read Flame  smoke = getSmoke(); // Read Smoke  voltage = getVoltage(); // read Voltage  current = getCurrent(); // Read Current  humidity = getHumidity(); // Read Humidity  temperature = getTemperature(); // Read Temperature  // Get data run every 3 s  if (milis - getMilis >= getTime) {  getMilis = milis;  relayState = getData();  // check relay state  if (relayState == 1) {  digitalWrite(led, HIGH);  digitalWrite(relayOne, HIGH);  } else {  digitalWrite(led, LOW);  digitalWrite(relayOne, LOW);  }  }  // run every 30 s  if (milis - postMilis >= postTime) {  postMilis = milis;  postData();  }  checkData();  printData();  delay(1000);  } |

**LAMPIRAN III**

**Dokumentasi**

****