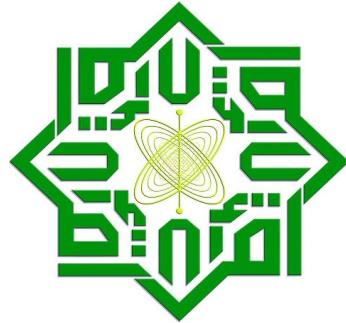


**Sistem Peringatan Kebakaran serta Kontrol dan  
Monitoring Penggunaan Listrik Rumah Tangga Berbasis IoT  
Terintegrasi dengan WhatsApp**

**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**

## DAFTAR ISI

<b>DAFTAR ISI .....</b>	<b>ii</b>
<b>DAFTAR GAMBAR .....</b>	<b>vi</b>
<b>DAFTAR TABEL.....</b>	<b>vii</b>
<b>DAFTAR LAMBANG/NOTASI .....</b>	<b>viii</b>
<b>DAFTAR SINGKATAN .....</b>	<b>ix</b>
 <b>BAB I PENDAHULUAN</b>	
1.1. Latar Belakang.....	I-1
1.2. Rumusan Masalah.....	I-4
1.3. Tujuan Penelitian.....	I-5
1.4. Batasan Masalah.....	I-5
1.5. Manfaat Penelitian.....	I-5
 <b>BAB II TINJAUAN PUSTAKA</b>	
2.1. Studi Literatur.....	II-1
2.2. <i>Warning System</i> .....	II-2
2.3. Mikrokontroler.....	II-2
2.3.1.ESP32.....	II-3
2.3.2.Skema Daya ESP32.....	II-4
2.3.3.Fitur ESP32 .....	II-6
2.4. <i>PZEM-004T</i> .....	II-7
2.5. MQ-2 .....	II-8
2.6. DHT-22.....	II-9
2.7. Flame Sensor .....	II-9
2.8. <i>Relay</i> .....	II-9
2.9. <i>Database</i> .....	II-11
2.9.1.Jenis Jenis <i>Database</i> .....	II-12
2.10. WhatsApp .....	II-13
 <b>BAB III METODE PENELITIAN</b>	
3.1. Diagram Alir Penelitian.....	III-1
3.2. Pengumpulan data.....	III-3

3.3.	Ilustrasi Sistem .....	III-3
3.4.	Perancangan <i>Hardware</i> .....	III-4
3.4.1.	Spesifikasi Perangkat yang digunakan.....	III-5
3.5.	Perancangan Listing Program Pada Kontroler .....	III-6
3.5.1.	Spesifikasi Program yang akan digunakan .....	III-6
3.6.	Pengujian Sistem .....	III-6
3.6.1.	Pengujian <i>Hardware</i> .....	III-6
3.6.2.	Pengujian Aplikasi/ <i>Software</i> .....	III-6
3.6.3.	Pengujian Keseluruhan Sistem.....	III-7
3.7.	Sekenario Pengujian .....	III-7
3.8.	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 <i>Layout</i> 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.	<i>Pinout</i> Relay SPDT .....	II-10
Gambar 2.11.	Modul Relay 1 <i>Channel</i> .....	II-10
Gambar 2.12.	Pin <i>Control</i> Pada Modul Relay .....	II-10
Gambar 2.13.	LED Indikator Pada Modul Relay.....	II-11
Gambar 2.14.	Terminal <i>Output</i> 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 <i>Hardware</i> .....	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 <i>Database</i> .....	II-12

## **DAFTAR LAMBANG/NOTASI**

$\mu\text{A}$  : *Micro Ampere*

$\mu\text{V}$  : *Micro Volt*

A : *Ampere*

C : *Celcius*

mA : *Mini Ampere*

Mhz : *Mega Hertz*

V : *Volt*

## DAFTAR SINGKATAN

AC : <i>Alternating Current</i>	LDO : <i>Low Drop Out</i>
ADC : <i>Analog to Digital Converter</i>	LED : <i>Light Emitting Diode</i>
ADT : <i>Android Developer Tool</i>	NC : <i>Normally Close</i>
ALU : <i>Aritmatic Logic Unit</i>	NO : <i>Normally Open</i>
AVL : <i>Automatic Vehicle Locator</i>	PCB : <i>Printed Circuit Bard</i>
BLE : <i>Bluetooth Low Energy</i>	PWM : <i>Pulse Width Modulation</i>
CPU : <i>Central Proccessing Unit</i>	RAM : <i>Random Access Memory</i>
CU : <i>Control Unit</i>	RTC : <i>Real Time Clock</i>
DBMS : <i>Database Management System</i>	SoC : <i>System on Chip</i>
DC : <i>Direct Current</i>	SPDT : <i>Single Pole Double Throw</i>
DPDT : <i>Doble Pole Doble Throw</i>	SPI : <i>Serial Pheriferal Interface</i>
DPST : <i>Doule Pole Single Throw</i>	SQL : <i>Structured Query Language</i>
GPIO : <i>General-Purpose Input/Output</i>	ULP : <i>Ultra Low Power</i>
GPRS : <i>General Package Radio Service</i>	UX : <i>User Experience</i>
I/O : <i>Input/Output</i>	VAC : <i>Volt AC</i>
I2C : <i>Inter-Integrated Circuit</i>	VCC : <i>Voltage Supply Colecotor</i>
IC : <i>Integreted Circuit</i>	VDC : <i>Volt DC</i>
IoT : <i>Internet of Things</i>	WIFI : <i>Wireless Fidelity</i>
IDE : <i>Integrated Development Environment</i>	
IEEE : <i>Institute of Electrical and Electronics Engineers</i>	

## **BAB I**

### **PENDAHULUAN**

#### **1.1. 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 tahunnya, 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 kerugian 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 dengan 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 menggunakan 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 diperkirakan 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 NRF24L01 yang 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 pendekripsi 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* berbasis 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 mendekripsi api, asap pada suhu yang tinggi.

## **1.2. Rumusan Masalah**

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

- a. Bagaimana cara merancang sistem kontrol dan monitoring pemakaian listrik yang terkoneksi dengan internet?
- b. Bagaimana cara merancang sistem peringatan kebakaran rumah dengan menggunakan Whatsapp sebagai media notifikasi?

### **1.3. 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.

### **1.4. Batasan Masalah**

Pada penelitian ini, dibuat batasan permasalahan sebagai berikut:

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

### **1.5. Manfaat Penelitian**

Diharapkan penelitian ini dapat memerikan manfaat sebagai berikut:

- a. Menurunkan tingkat konsumsi listrik rumah tangga pada peralatan yang tidak terpakai.
- b. Melakukan kontrol pada kelistrikan rumah dari jarak jauh.
- c. Meningkatkan keamanan dan kenyamanan dari potensi kebakaran rumah.

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **2.1. 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 lainnya. 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 pendekksi 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 Pendekksi Asap Kebakaran Menggunakan Sensor MQ-2 Berbasis Arduino Uno”, pada penelitian tersebut

sistem akan membaca nilai konsentrasi kadar CO<sup>2</sup> 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 output. 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 mendekripsi 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 Pendekripsi 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.

## 2.2. *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.

## 2.3. *Mikrokontroler*

Mikrokontroler ( $\mu$ C) 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]

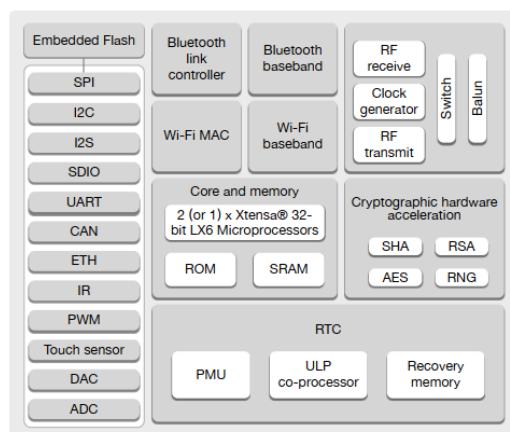
### 2.3.1.ESP32

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 40 $\mu$ m. 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).

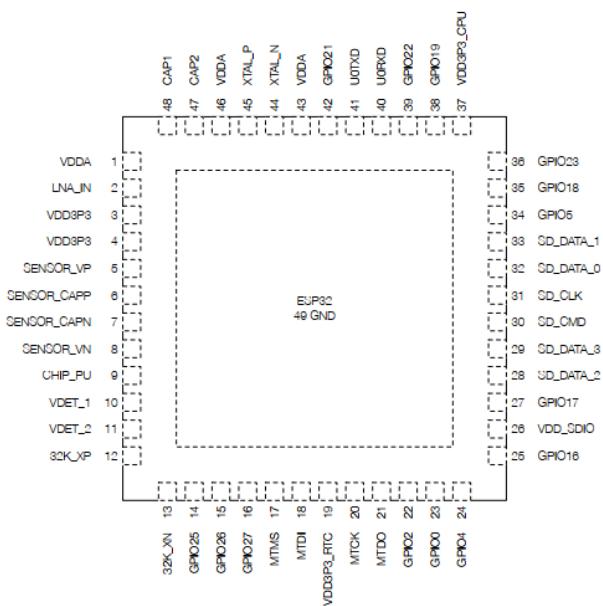


Gambar 2.1. ESP-WROOM-32[12]

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]

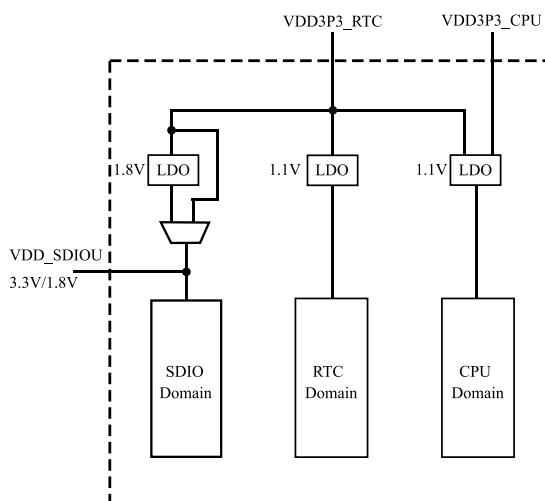


Gambar 2.3. Pin *Layout* ESP32[11]

### **2.3.2. Skema Daya**

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

- a. VDD3P3\_RTC yang merupakan catu daya input untuk RTC (*Real Time Clock*) dan CPU (*Central Processing Unit*).
  - b. VDD3P3\_CPU merupakan catu daya untuk CPU (*Central Processing Unit*).
  - c. 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.

- a. **Mode Aktif:** Radio WiFi chip dihidupkan. Chip dapat menerima atau mengirim data.
- b. **Modem-sleep mode:** CPU tetap beroperasi dan *Clock* dapat dikonfigurasi. Tetapi penggunaan Wi-Fi/*Bluetooth* dan radio dinonaktifkan.
- c. **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 menggunakan WiFi maupun *Bluetooth*.
- d. **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.
- e. **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
<i>Modem-sleep</i>	<i>Single-core chip</i>	20 ~ 34 mA
	<i>Dual-core chips</i>	20 ~ 68 mA
<i>Light-sleep</i>		0.8 mA
<i>Deep-sleep</i>	Prosesor ULP Diaktifkan	150 $\mu$ A
	Sensor ULP dalam keadaan <i>monitoring mode</i>	100 $\mu$ A
	RTC timer + <i>memory</i>	10 $\mu$ A
Hibernasi	Hanya RTC <i>Timer</i> yang aktif	5 $\mu$ A
<i>Power Off</i>	Chip <i>Pull Up</i> diatur menjadi <i>Low Level</i> . Chip dimatikan	1 $\mu$ A

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.

### 2.3.3. Fitur ESP32

ESP32 dapat beroperasi dengan handal di lingkungan industri, dengan suhu pengoperasian berkisar antara  $-40^{\circ}\text{C}$  hingga  $+125^{\circ}\text{C}$ . 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]

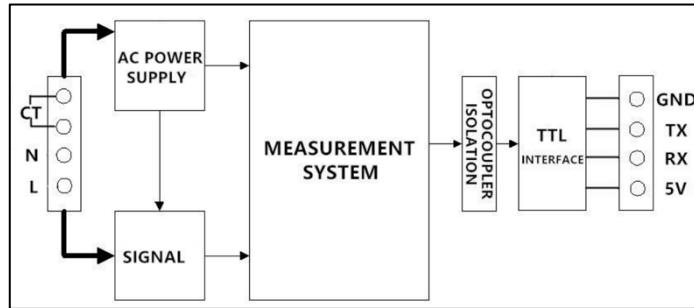
No.	Fitur	Deskripsi
1	<i>Prosesor</i>	<i>Xtensa dual-core 32-bit LX6 240 MHz Ultralow power (ULP) co-processor</i>
2	<i>Memory</i>	<i>520 KB SRAM (Static Random Access Memory)</i>
3	<i>Wireless connectivity</i>	<i>WiFi: 802.11 b/g/n Bluetooth: v4.2 BR/EDR dan BLE</i>
4	<i>Peripheral interfaces</i>	<i>12-bit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

#### 2.4. 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 -20°C ~ +60°C.[14]



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

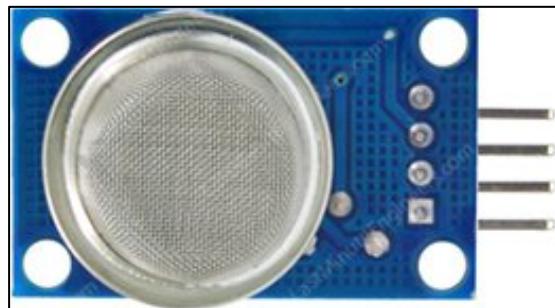


Gambar 2.6. Blok Diagram PZEM-004T[14]

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 menggunakan isolator optocoupler sama halnya seperti penggunaan relay.

## 2.5. MQ-2

Modul MQ-2 merupakan sensor yang dapat mendeteksi kosentrasi LPG, Asap/CO<sub>2</sub>, 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.

## 2.6. DHT-22

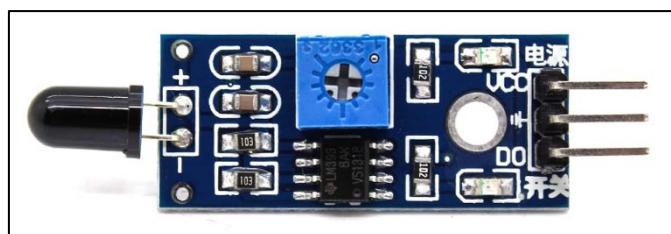
Modul DHT-22 merupakan sensor yang dapat membaca temperatur dan kelembaban ruangan berdasarkan prinsip kerja dari NTC (*Negative Temperature Coefficient*) Thermistor yang dapat mengukur temperatur -40°C sampai 125°C dengan tingkat akurasi  $\pm 0.5^\circ$ . Modul ini juga dapat mengukur tingkat kelembaban ruangan mulai dari 0% sampai 100% dengan akurasi  $\pm 2\text{-}5\%$ .



Gambar 2.8. Sensor DHT-22[15]

## 2.7. 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 transduser untuk mendeteksi nyala api dan menghasilkan output berupa sinyal listrik analog maupun digital.



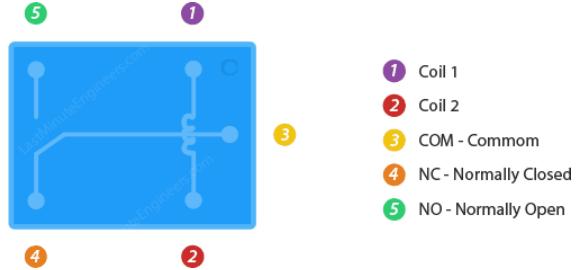
Gambar 2.9. Flame Sensor

Sensor ini dapat mendeteksi nyala api dengan besar sudut  $60^\circ$  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.

## 2.8. 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:

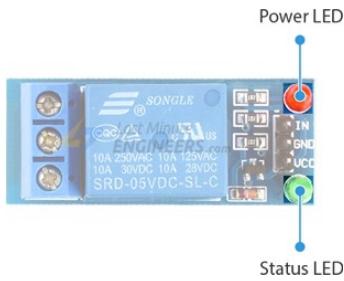


Gambar 2.11. Modul Relay 1 Channel[17]

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]

## 2.9. 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, memperbarui, 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]

### 2.9.1. 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 <i>Database</i>	Definisi
1	<i>Database</i> relasional	<i>Database</i> 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	<i>Database</i> berorientasi objek.	Informasi dalam <i>database</i> berorientasi objek direpresentasikan dalam bentuk objek, seperti dalam pemrograman berorientasi objek.
3	<i>Database</i> terdistribusi.	<i>Database</i> 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 <i>Warehouse</i>	Repositori pusat untuk data, gudang data adalah jenis <i>database</i> yang dirancang khusus untuk permintaan dan analisis cepat.
5	NoSQL <i>Database</i> .	NoSQL, atau <i>database</i> non-relasional, memungkinkan data yang tidak terstruktur dan terstruktur disimpan dan dimanipulasi (berbeda dengan <i>database</i> relasional, yang mendefinisikan bagaimana semua data yang dimasukkan ke dalam basis data harus dikomposisi). <i>Database</i> NoSQL semakin populer ketika aplikasi web menjadi lebih umum dan lebih kompleks.
6	<i>Database</i> grafik.	<i>Database</i> grafik menyimpan data dalam hal entitas dan hubungan antar entitas.
7	<i>Database</i> OLTP.	<i>Database Online Transaction Processing</i> (OLTP) adalah <i>database</i> analitik cepat yang dirancang untuk sejumlah besar transaksi yang dilakukan oleh banyak pengguna.

## **2.10. WhatsApp**

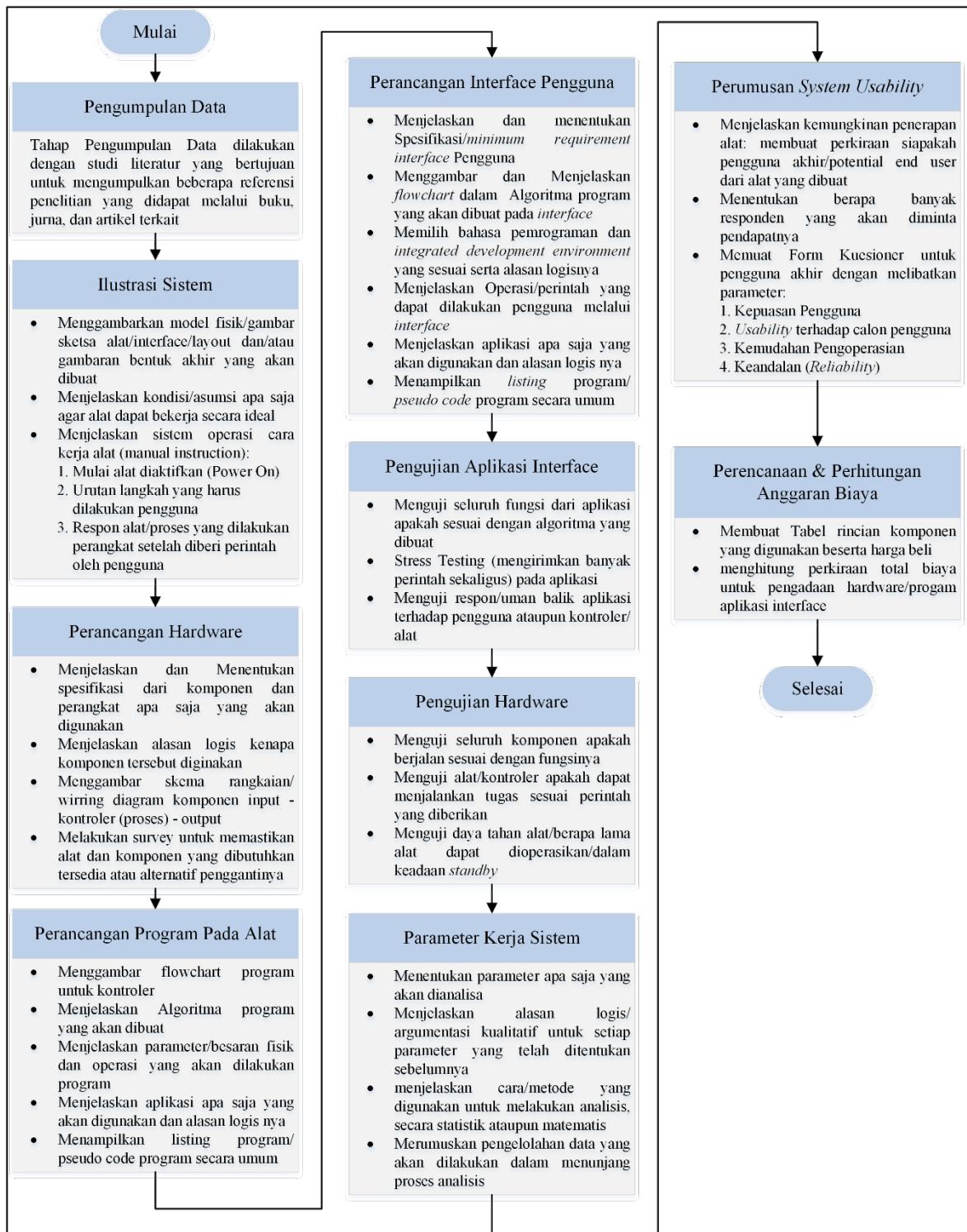
WhatsApp adalah aplikasi bertukar pesan yang dikembangkan oleh WhatsApp Inc. dan diakuisisis oleh Facebook inc pada 2014 (sekarang Meta Platform) dan telah diunduh lebih dari 5 Miliar kali di Play Store. Aplikasi WhatsApp telah digunakan lebih dari 84 Juta orang di Indonesia[20] yang dapat di install 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 lainnya[21].

## BAB III

### METODE PENELITIAN

#### 3.1. 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.

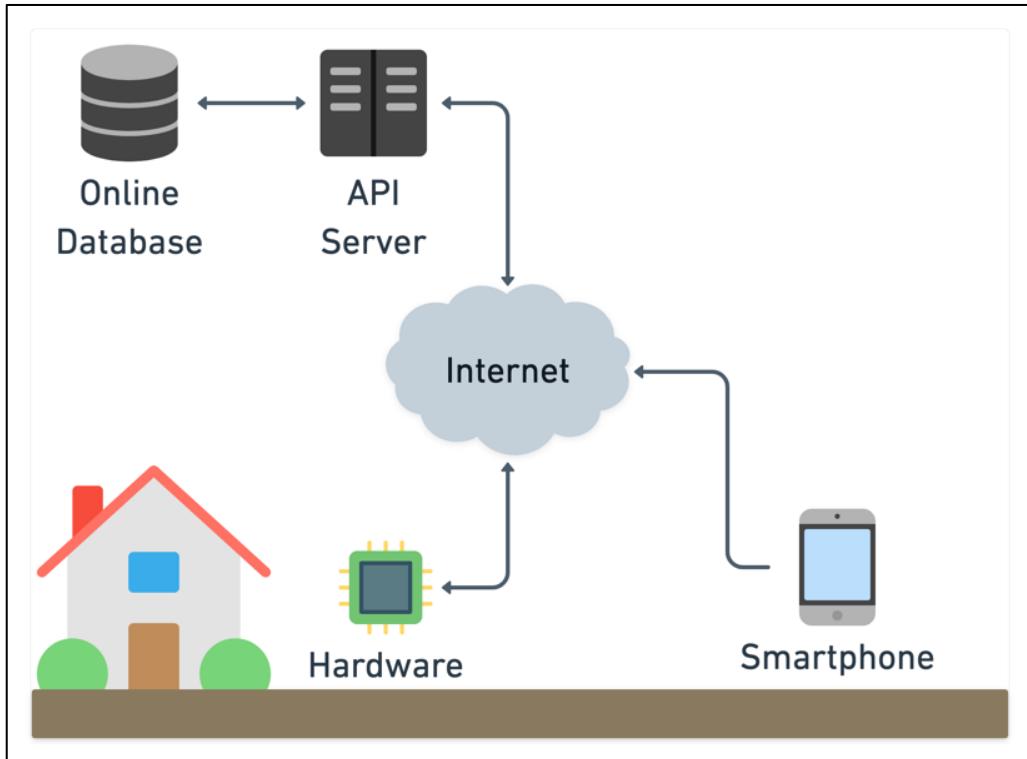
### **3.2. Pengumpulan Data**

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.

### **3.3. 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.



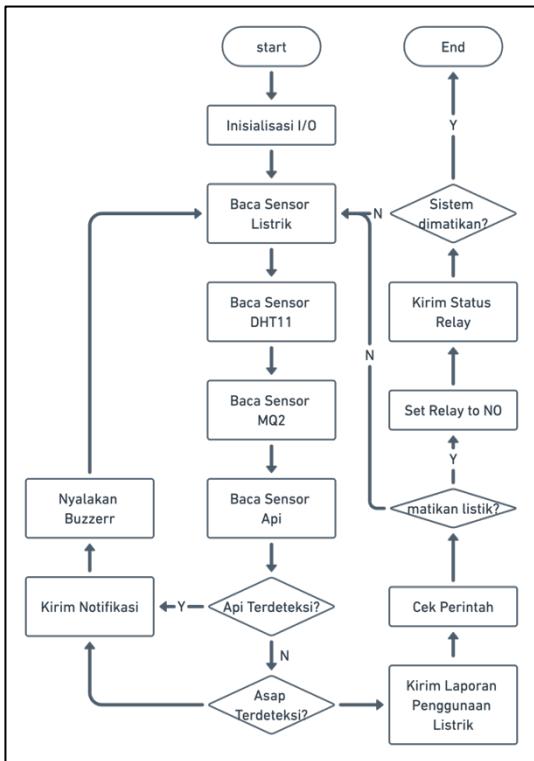
Gambar 3.2. Ilustrasi Sistem

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

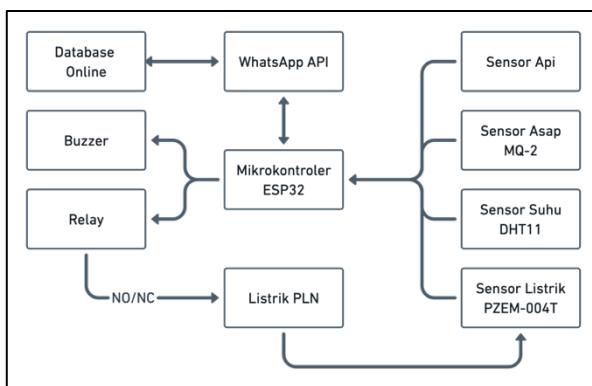
### 3.4. Perancangan Hardware

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.



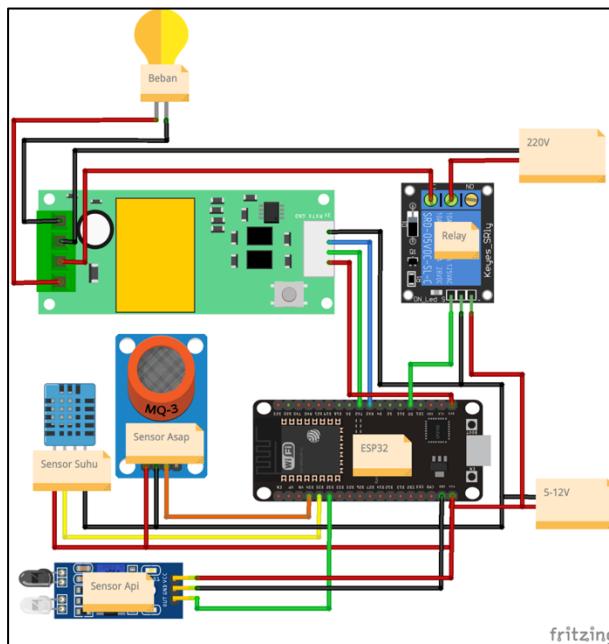
Gambar 3.3. Diagram Alir sistem

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.



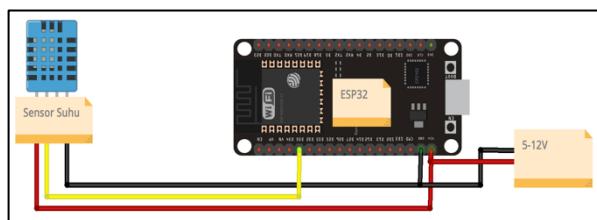
Gambar 3.5. Sekema Keseluruhan Hardware

### 3.4.1. Spesifikasi Perangkat yang akan digunakan

- 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.
- 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.
- Menggunakan sensor PZEM004T untuk membaca arus dan tegangan dari penggunaan listrik. Dengan tambahan kumparan yang dapat mendeteksi arus yang mengalir lebih presisi.
- 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.

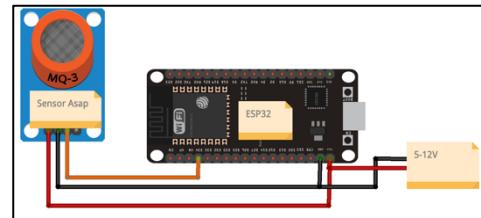
### 3.4.2. Skema Rangkaian

- Skema rangkaian DHT-22



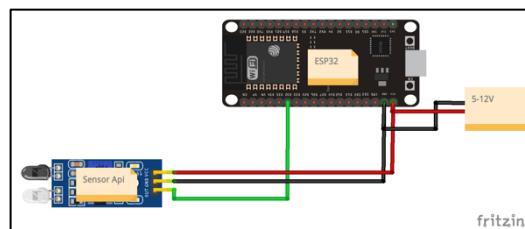
Gambar 3.6. Rangkaian Sensor Suhu DHT-22

b. Skema Rangkaian MQ-2



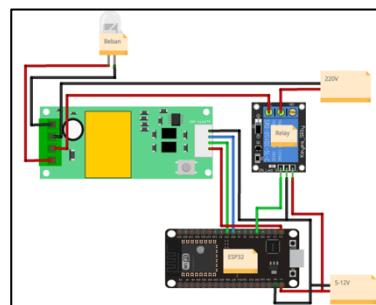
Gambar 3.7. Rangkaian Sensor Asap MQ-2

c. Skema Rangkaian Flame Sensor



Gambar 3.8. Rangkaian Sensor Api

d. Skema Rangkaian Sensor PZEM004T



Gambar 3.9. Rangkaian Sensor Arus & Tegangan dengan Relay

### 3.5. 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.

#### 3.5.1. 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*).

### **3.6. 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*.

#### **3.6.1. 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.

#### **3.6.2. 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.

### **3.6.3. 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.

### **3.7. 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:

- a. 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.

- b. 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.

- c. 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.

- d. *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,

- e. 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.

f. 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	>40°C	>300	True	Kebakaran Terdeteksi
2	>40°C	>300	False	Asap Terdeteksi & Suhu Tinggi
3	<40°C	>300	True	Asap & Api Terdeteksi
4	<40°C	>300	False	Asap Terdeteksi
5	>40°C	<300	True	Suhu Tinggi & Api Terdeteksi
6	>40°C	<300	False	Suhu Tinggi
7	<40°C	<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

### 3.8. 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:

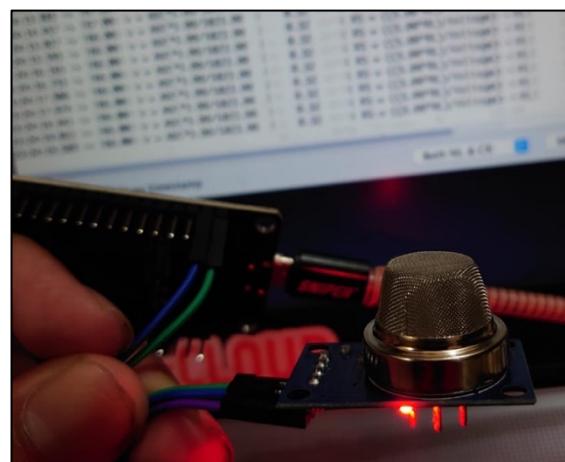
a. Sensor DHT-22



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.

b. 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 29°C-34°C 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.

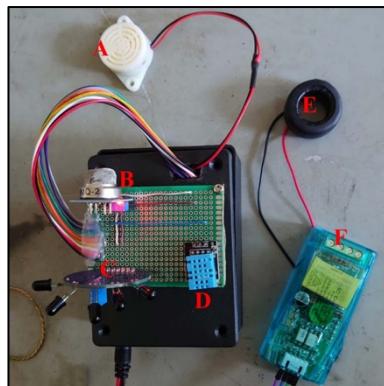
## BAB IV

### 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.

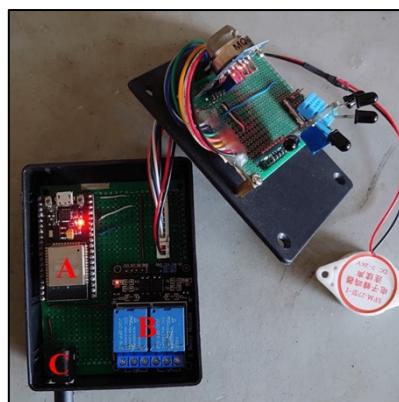
#### 4.1. Hasil Perancangan Hardware

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 *aktuuator* 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.

## 4.2. Hasil Pengujian Hardware

### 4.2.1. 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.

### 4.2.2. 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

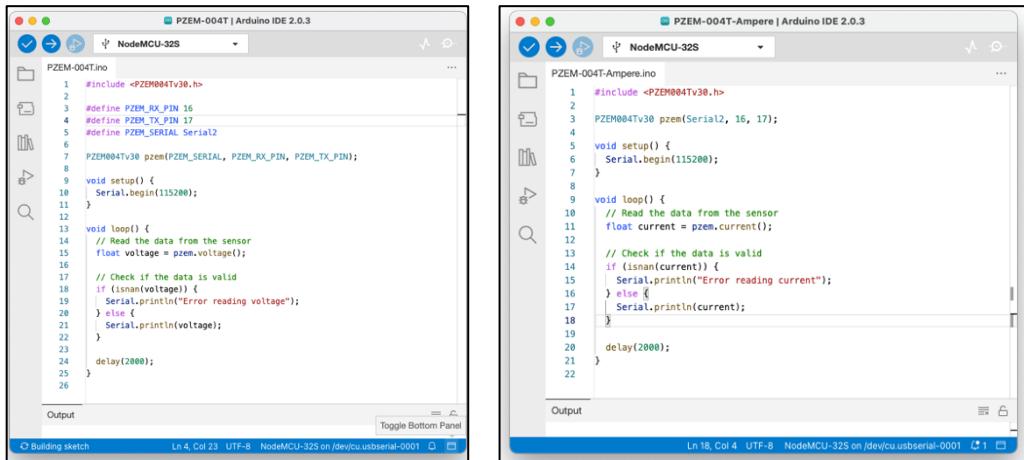
Tabel 4.1. Hasil Pengujian GPIO mikrokontroler ESP-32S

Label	GPIO	Dapat Digunakan?	Keterangan
D0	0	Ya	Aktif HIGH saat booting
TX0	1	Tidak	Digunakan Untuk Upload Program
D2	2	Ya	Terhubung ke <i>builtin</i> LED
RX0	3	Tidak	Digunakan untuk upload program
D4	4	Ya	
D5	5	Ya	Aktif HIGH saat booting
D6	6	Tidak	
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	
D35	35	Ya	
VP	36	Ya	
VN	39	Ya	Hanya dapat digunakan sebagai input

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.

#### 4.2.3. Pengujian PZEM-004T

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



```
PZEM-004Tino
1 #include <PZEM004Tv30.h>
2
3 #define PZEM_RX_PIN 16
4 #define PZEM_TX_PIN 17
5 #define PZEM_SERIAL Serial2
6
7 PZEM004Tv30 pzem(PZEM_SERIAL, PZEM_RX_PIN, PZEM_TX_PIN);
8
9 void setup() {
10   Serial.begin(115200);
11 }
12
13 void loop() {
14   // Read the data from the sensor
15   float voltage = pzem.voltage();
16
17   // Check if the data is valid
18   if (isnan(voltage)) {
19     Serial.println("Error reading voltage");
20   } else {
21     Serial.println(voltage);
22   }
23
24   delay(2000);
25 }
```

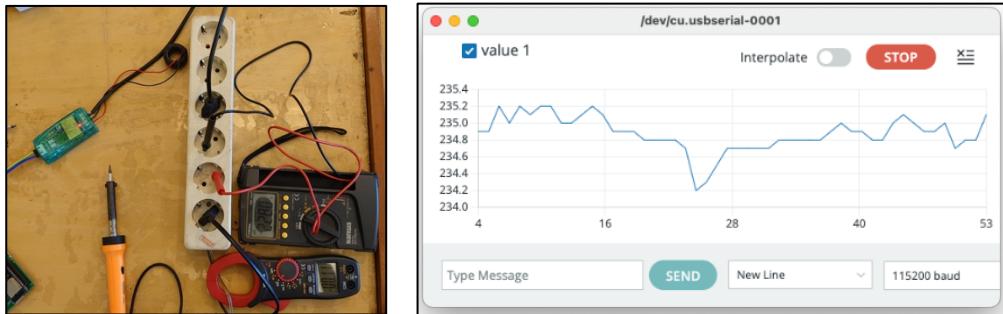
```
PZEM-004T-Ampere.ino
1 #include <PZEM004Tv30.h>
2
3 PZEM004Tv30 pzem(Serial2, 16, 17);
4
5 void setup() {
6   Serial.begin(115200);
7 }
8
9 void loop() {
10   // Read the data from the sensor
11   float current = pzem.current();
12
13   // Check if the data is valid
14   if (isnan(current)) {
15     Serial.println("Error reading current");
16   } else {
17     Serial.println(current);
18   }
19
20   delay(2000);
21 }
```

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.

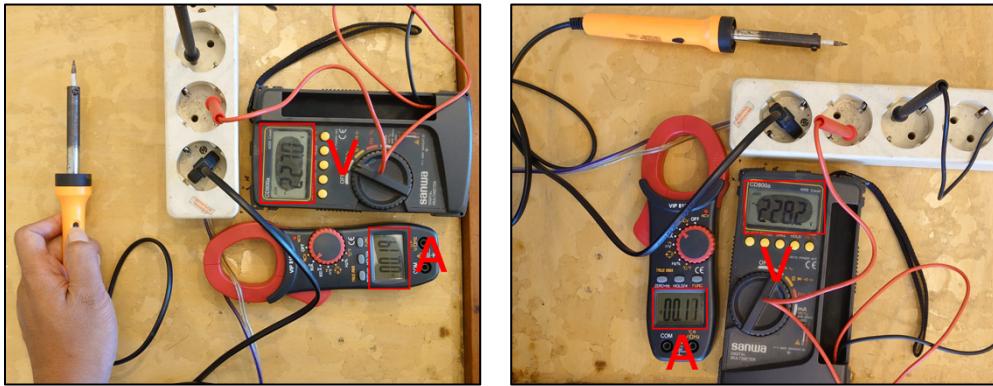


a. Tampilan Hardware

b. Tampilan Software

Gambar 4.6. Pengujian PZEM-004T

Penulis melakukan pengujian pada PZEM-004T tanpa beban dan dengan menggunakan beban berupa senter 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 senter 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.

#### 4.2.4. Pengujian Sensor Asap

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

```

MQ2_Sensor | Arduino IDE 2.0.3
NodeMCU-32S
MQ2_Sensor.ino
1 const int mq2 = 34;
2 float asap;
3
4 void setup() {
5   Serial.begin(115200);
6   pinMode(mq2, INPUT);
7   delay(1000);
8 }
9
10 void loop() {
11   float data = 0;
12
13   for (int i = 0; i < 20; i++) {
14     data += analogRead(mq2);
15     delay(25);
16   }
17
18   asap = data / 10;
19   Serial.println(data / 10);
20   delay(500);
21 }
22

```

Output

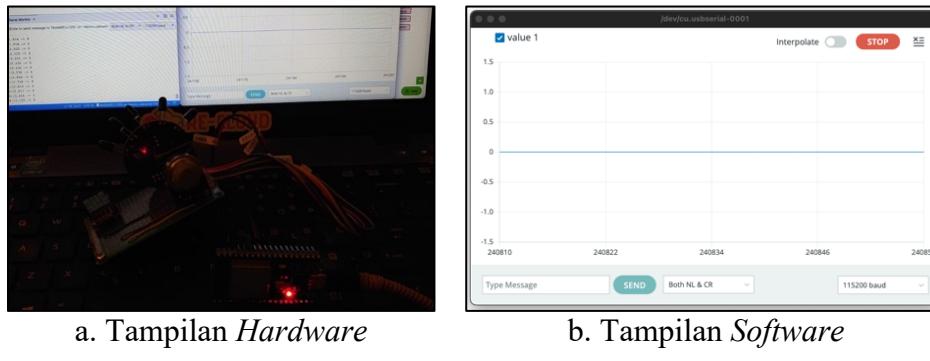
```

Writing at 0x00004000... (88 %)
Writing at 0x00004000... (88 %)
Writing at 0x00004000... (88 %)
Wrote 248784 bytes (137928 compressed) at 0x000010000 in 2.8 seconds (effective 717.7
Micros used)

```

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.

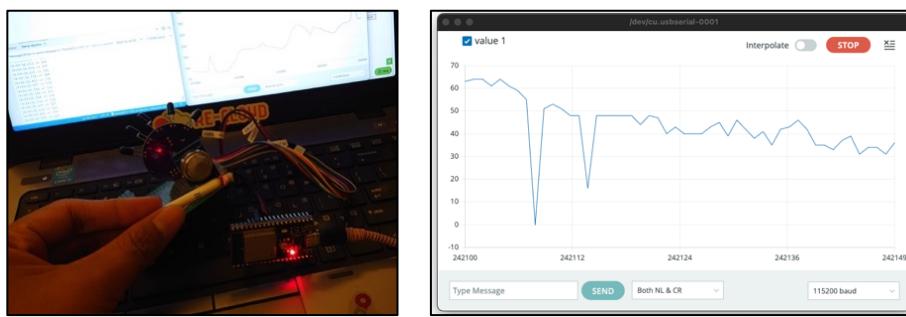


a. Tampilan *Hardware*

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

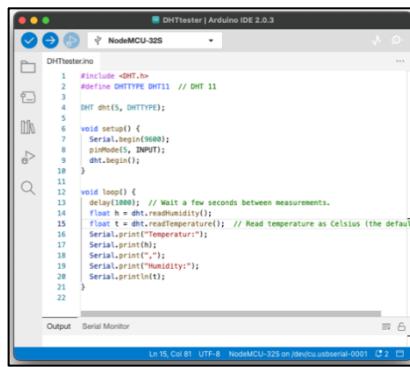
Mikrokontroler akan menampilkan grafik perubahan nilai dari sensor apabila asap terdeteksi seperti gambar 4.10. bagian (b) di mana penulis membakar kertas untuk menguji pembacaan sensor.

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

#### 4.2.5. Pengujian DHT

Sensor DHT dapat mendeteksi temperatur ruangan dengan akurasi  $2^{\circ}\text{C}$ , untuk memastikan akurasi tersebut perlu dilakukan pengujian secara langsung dengan pembanding berupa termometer digital.



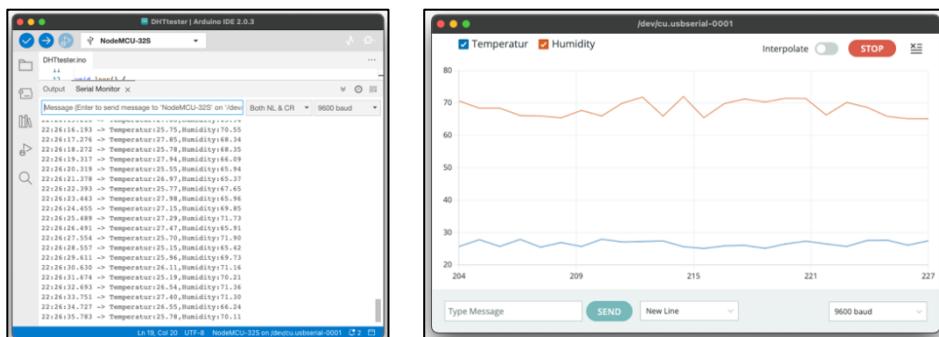
```

1 //<include <DHT.h>
2 #define DHTTYPE DHT11 // DHT 11
3
4 DHT dht11, DHTTYPE;
5
6 void setup() {
7   Serial.begin(9600);
8   pinMode(5, INPUT);
9   dht11.begin();
10 }
11
12 void loop() {
13   delay(1000); // Wait a few seconds between measurements.
14   float t = dht11.readTemperature(); // Read temperature as Celsius (the default)
15   Serial.print("Temperatur:");
16   Serial.print(t);
17   Serial.print(" ");
18   Serial.print("Humidity:");
19   Serial.print(dht11.readHumidity());
20   Serial.println(t);
21 }
22

```

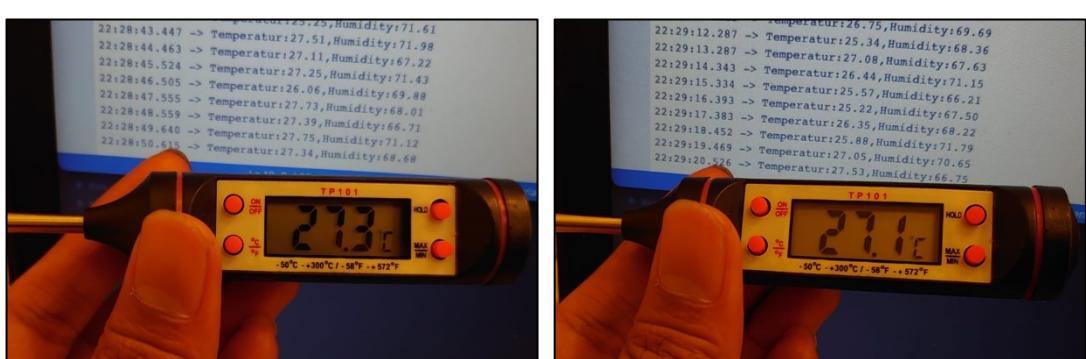
Gambar 4.11. Listing Program DHT

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.



Gambar 4.12. Pengujian Sensor DHT

Pada gambar 4.12. di atas merupakan hasil pembacaan sensor DHT yang menampilkan temperatur dengan tingkat *error* sebesar  $2^{\circ}\text{C}$  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.

$$|Error \text{ } ^\circ\text{C}| = sensor - thermometer \dots \dots \dots \quad (4.1)$$

$$|Error \%| = 1 - \left( \frac{sensor}{thermometer} \right) \cdot 100\% \dots \quad (4.2)$$

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

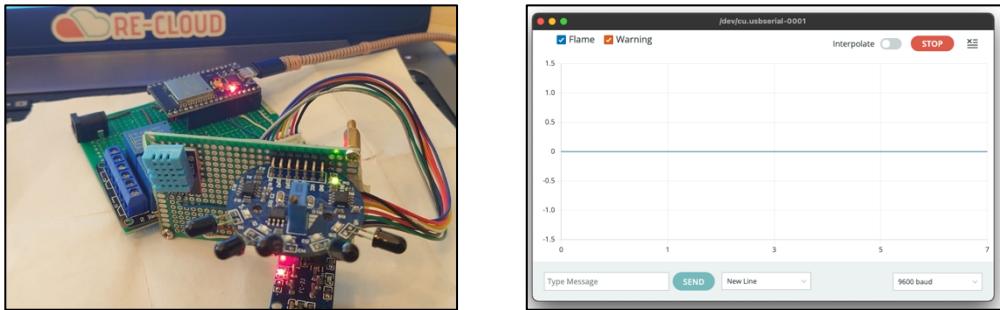
#### 4.2.6. Pengujian Sensor Api

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 dari sudut yang lebih besar.

Gambar 4.14. Listing Program Membaca Sensor Api

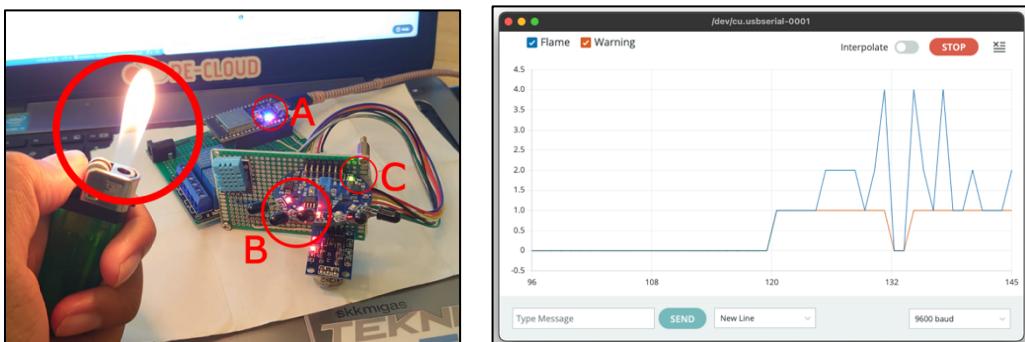
Untuk membaca sensor api penulis menggunakan data digital pada sensor api dengan menggunakan 5 channel, jika sensor mendeteksi api sensor akan memberikan

logika HIGH yang akan diubah 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 sehingga 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.

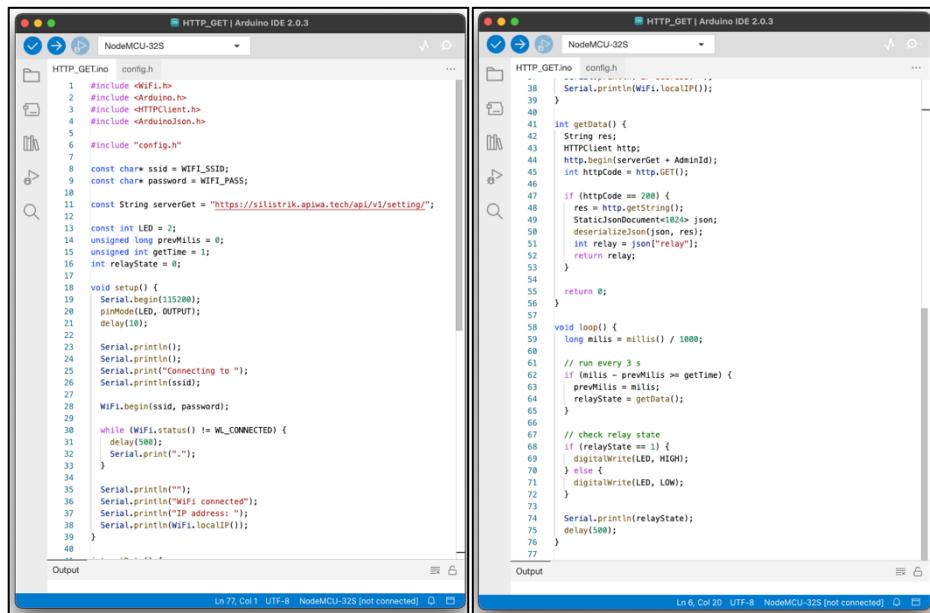
Tabel 4.4. Hasil pembacaan Sensor Api

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

## 4.3. Hasil Pengujian *Software*

### 4.3.1. 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.



```
HTTP.GET.ino config.h
1 #include <WiFi.h>
2 #include <Arduino.h>
3 #include <HTTPClient.h>
4 #include <ArduinoJson.h>
5
6 #include "config.h"
7
8 const char ssid = WIFI_SSID;
9 const char password = WIFI_PASS;
10
11 const String serverGet = "https://silistrik.apiwa.tech/api/v1/setting/";
12
13 const int LED = 2;
14 unsigned long prevMillis = 0;
15 unsigned int getTime = 1;
16 int relayState = 0;
17
18 void setup() {
19   Serial.begin(115200);
20   pinMode(LED, OUTPUT);
21   delay(10);
22
23   Serial.println();
24   Serial.println();
25   Serial.print("Connecting to ");
26   Serial.println(ssid);
27
28   WiFi.begin(ssid, password);
29
30   while (WiFi.status() != WL_CONNECTED) {
31     delay(500);
32     Serial.print(".");
33   }
34
35   Serial.println("");
36   Serial.println("WiFi connected");
37   Serial.println("IP address: ");
38   Serial.println(WiFi.localIP());
39 }
40

HTTP.GET.ino config.h
38   Serial.println(WiFi.localIP());
39 }
40
41 int getData() {
42   String res;
43   HTTPClient http;
44   http.begin(serverGet + AdminId);
45   int httpCode = http.GET();
46
47   if (httpCode == 200) {
48     res = http.getString();
49     StaticJsonDocument<100> json;
50     deserializeJson(json, res);
51     int relay = json["relay"];
52
53     return relay;
54   }
55
56   return 0;
57 }
58
59 void loop() {
60   long millis = millis() / 1000;
61
62   // run every 3 s
63   if (millis - prevMillis >= getTime) {
64     prevMillis = millis;
65     relayState = getData();
66
67     // check relay state
68     if (relayState == 1) {
69       digitalWrite(LED, HIGH);
70     } else {
71       digitalWrite(LED, LOW);
72     }
73
74     Serial.println(relayState);
75     delay(500);
76   }
77 }
```

Gambar 4.10. Listing Program http GET

Dari program di atas akan didapatkan respons dari server berupa data json (*javascript 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, setpoint temperatur maksimal, nilai asap maksimal, limit penggunaan daya dan status relay saat ini.

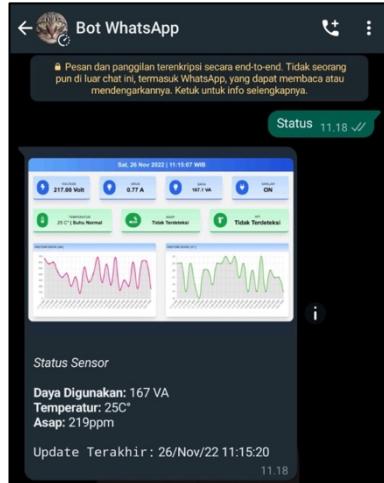
### 4.3.2. Pengujian POST data ke Database

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

```
POST1.ino config.h
1 //include <Arduino.h>
2 #include <WiFi.h>
3 #include <HTTPClient.h>
4 #include <StaticJsonDocument.h>
5 #include <config.h>
6 // WiFi Config
7 const char ssid = "NETFLIX";
8 const char password = "NETFLIX_PASS";
9 // Your domain name with URL path or IP address with path
10 const String serverPost = "https://sillistrik.apilwa.tech/api/v1/data";
11 const int LED = 2;
12 const int LED2 = 1;
13 // Variable
14 float flame = 0.8;
15 float volt = 215.0;
16 float smoke = 237.0;
17 float hum = 45.0;
18 float amper = 2.2;
19 float temperature = 25.0;
20 float power = 0.0;
21 unsigned long preMillis = 0;
22 // Post data every? (second)
23 unsigned int postTime = 30;
24
25 > void setup() {
26   > }
27   > float getTemperature() {
28     > }
29   > float getHumidity() {
30     > }
31   > float getFlame() {
32     > }
33   > float getVolt() {
34     > }
35   > float getAmper() {
36     > }
37   > float getSmoke() {
38     > }
39
40 void postData() {
41   Serial.println("Post data");
42   String params;
43   String response;
44   HTTPClient http;
45   StaticJsonDocument<200> buf;
46   buf["v1"] = flame;
47   buf["volt"] = volt;
48   buf["smoke"] = smoke;
49   buf["temperature"] = temp;
50   buf["amper"] = amper;
51   buf["power"] = power;
52   buf["token"] = "X10";
53   buf["username"] = "admin";
54   buf["password"] = "admin";
55   http.begin(serverPost);
56   http.addHeader("Content-Type", "application/json");
57   http.POST(params);
58   response = http.getString();
59   StaticJsonDocument<200> json;
60   deserializeJson(json, response);
61
62   void loss() {
63     long millis = millis() / 1000;
64     volt = getvolt();
65     flame = getflame();
66     flame = getflame();
67     hum = gethumidity();
68     amper = getamper();
69     temperature = gettemperature();
70     power = volt * amper;
71     if (power > 0.0 || smoke > 300 || flame > 0) {
72       postdata();
73     }
74   }
75   // Post data every 30 s
76   if (millis - preMillis >= postTime) {
77     preMillis = millis();
78     postdata();
79   }
80   delay(5000);
81 }
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
625
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
```

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

b. 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.

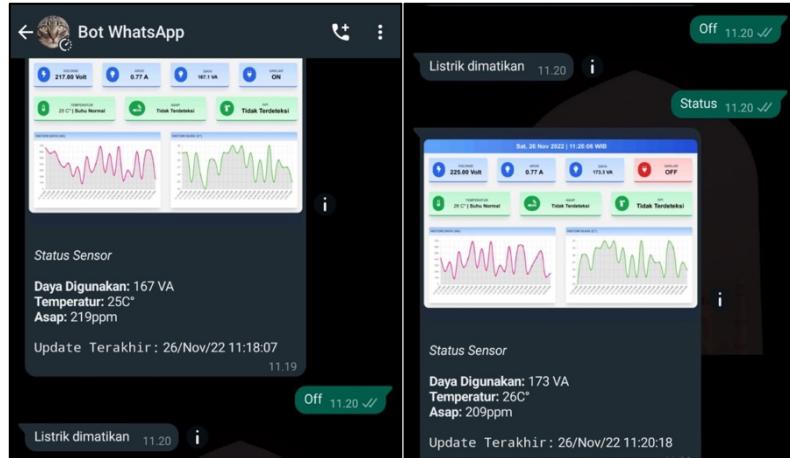
c. Menyalakan Relay



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.

#### d. Mematikan Relay

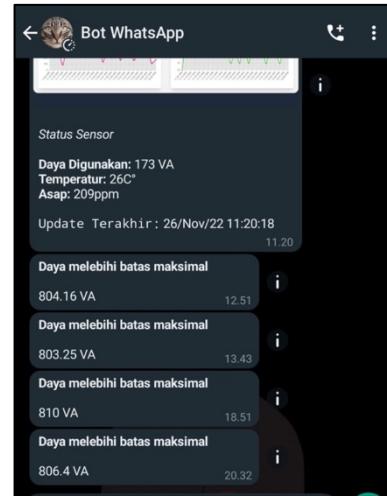


Gambar 4.16. Mematikan Relay melalui WhatsApp

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.

#### 4.3.4. 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 setpoint 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.

#### 4.3.5. 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	<i>Message sent</i>
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	<i>Message sent</i>
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	<i>Message sent</i>
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	<i>Message sent</i>
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	<i>Message sent</i>
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	<i>Message sent</i>
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.

#### 4.4. 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.

#### 4.4.1. 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
			<b>Min</b>	0.44 kWh	Rp. 595
			<b>Max</b>	2.73 kWh	Rp. 3,691
			<b>Rata-Rata</b>	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.

PENETAPAN PENYESUAIAN TARIF TENAGA LISTRIK (TARIFF ADJUSTMENT)					
OKTOBER - DESEMBER 2022					
NO.	GOL. TARIF	BATAS DAYA	REGULER		PRA BAYAR (Rp/kWh)
			BIAYA BEBAN (Rp/kVA/bulan)	BIAYA PEMAKAIAN (Rp/kWh) DAN BIAYA kVAhr (Rp/kVAhr)	
1.	R-1/TR	900 VA-RTM	*)	1.352,00	1.352,00
2.	R-1/TR	1.300 VA	*)	1.444,70	1.444,70
3.	R-1/TR	2.200 VA	*)	1.444,70	1.444,70
4.	R-2/TR	3.500 VA s.d. 5.500 VA	*)	1.699,53	1.699,53
5.	R-3/TR	6.600 VA ke atas	*)	1.699,53	1.699,53
6.	B-2/TR	6.600 VA s.d. 200 kVA	*)	1.444,70	1.444,70
7.	B-3/TM	di atas 200 kVA	**) )	Blok WBP = K x 1.035,78 Blok LWBP = 1.035,78 kVAhr = 1.114,74 ****)	-
8.	I-3/TM	di atas 200 kVA	**) )	Blok WBP = K x 1.035,78 Blok LWBP = 1.035,78 kVAhr = 1.114,74 ****)	-
9.	I-4/TT	30.000 kVA ke atas	***) )	Blok WBP dan Blok LWBP = 996,74 kVAhr = 996,74 ****)	-
10.	P-1/TR	6.600 VA s.d. 200 kVA	*)	1.699,53	1.699,53
11.	P-2/TM	di atas 200 kVA	**) )	Blok WBP = K x 1.415,01 Blok LWBP = 1.415,01 kVAhr = 1.522,88 ****)	-
12.	P-3/TR		*)	1.699,53	1.699,53
13.	L/TR, TM, TT		-	1.644,52	-

Catatan :

- \*) Diterapkan Rekening Minimum (RM):  
RM1 = 40 (Jam Nyala) x Daya tersambung (kVA) x Biaya Pemakaian.
- \*\*) Diterapkan Rekening Minimum (RM):  
RM2 = 40 (Jam Nyala) x Daya tersambung (kVA) x Biaya Pemakaian LWBP.  
Jam nyala : kWh per bulan dibagi dengan kVA tersambung.
- \*\*\*) Diterapkan Rekening Minimum (RM):  
RM3 = 40 (Jam Nyala) x Daya tersambung (kVA) x Biaya Pemakaian WBP dan LWBP.  
Jam nyala : kWh per bulan dibagi dengan kVA tersambung.
- \*\*\*\*) Biaya kelebihan pemakaian daya reaktif (kVAhr) dikenakan dalam hal faktor daya rata-rata setiap bulan kurang dari 0,85 (delapan puluh lima perseratus).

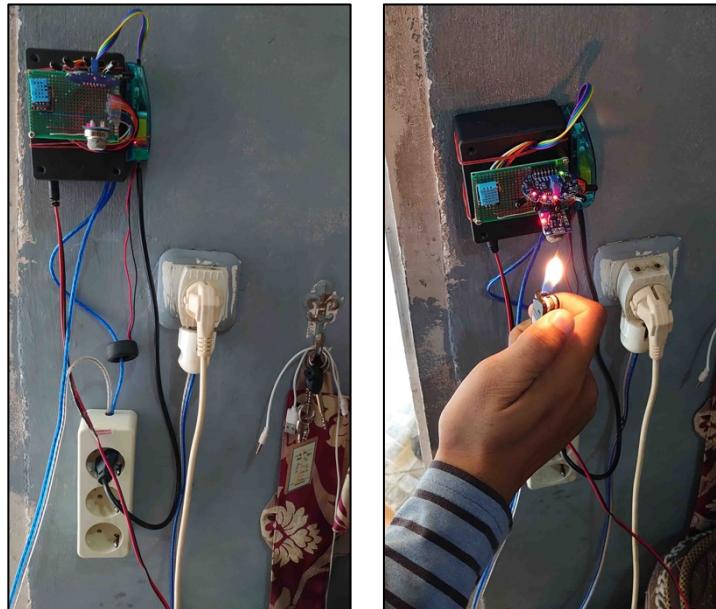
K : Faktor perbandingan antara harga WBP dan LWBP sesuai dengan karakteristik beban sistem kelistrikan setempat (1,4 ≤ K ≤ 2), ditetapkan oleh Direksi Perusahaan Perseroan (Persero) PT Perusahaan Listrik Negara.

WBP : Waktu Beban Puncak.  
LWBP : Luar Waktu Beban Puncak.

Gambar 4.18. Penyesuaian Tarif Listrik[22]

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

#### 4.4.2. 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.



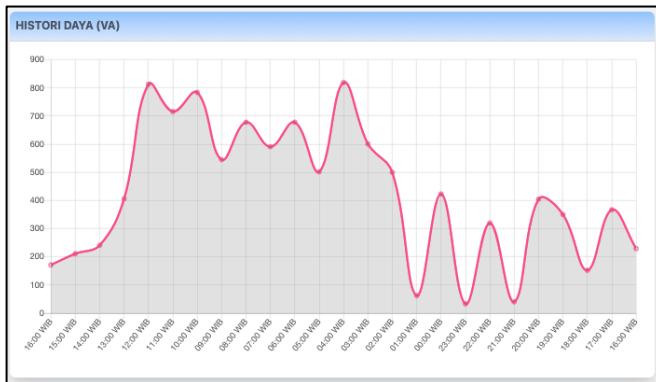
Gambar 4.20. Tampilan Status Sensor pada Website

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.

Notifikasi Sensor							
Waktu	Voltase	Arus	Daya	Suhu	Api	Asap	Status
01/01/23 16:00:03	220 V	1.0 A	229 VA	32 C°	0	274	Temperatur Tinggi
01/01/23 15:00:03	220 V	1.7 A	367 VA	31 C°	0	307	Temperatur Tinggi
01/01/23 14:00:03	220 V	0.69 A	152 VA	26 C°	0	200	Aman
01/01/23 13:00:03	220 V	1.6 A	350 VA	30 C°	0	273	Aman
01/01/23 12:00:04	220 V	1.8 A	405 VA	26 C°	0	206	Aman
01/01/23 11:00:04	222 V	0.18 A	40 VA	26 C°	0	300	Aman
01/01/23 10:00:07	222 V	1.4 A	320 VA	30 C°	0	220	Temperatur Tinggi
01/01/23 09:00:06	219 V	0.15 A	33 VA	31 C°	0	248	Temperatur Tinggi
01/01/23 08:00:09	219 V	1.9 A	423 VA	31 C°	0	257	Temperatur Tinggi
01/01/23 07:00:05	220 V	0.28 A	62 VA	25 C°	0	224	Aman
01/01/23 06:00:07	222 V	2.3 A	500 VA	29 C°	0	251	Aman
01/01/23 05:00:05	221 V	2.7 A	601 VA	31 C°	0	200	Temperatur Tinggi
01/01/23 04:00:17	222 V	3.7 A	819 VA	26 C°	0	309	Penggunaan Listrik Tinggi
01/01/23 03:00:05	222 V	2.3 A	502 VA	32 C°	0	226	Temperatur Tinggi
01/01/23 02:00:03	221 V	3.1 A	678 VA	28 C°	0	211	Aman
01/01/23 01:00:03	222 V	2.7 A	591 VA	28 C°	0	253	Aman
01/01/23 00:00:04	221 V	3.1 A	678 VA	27 C°	0	293	Aman
31/12/22 23:00:03	219 V	2.5 A	545 VA	28 C°	0	231	Aman
31/12/22 22:00:03	219 V	3.6 A	784 VA	29 C°	0	207	Aman
31/12/22 21:00:03	221 V	3.2 A	716 VA	28 C°	0	222	Aman
31/12/22 20:00:04	222 V	3.7 A	813 VA	26 C°	0	271	Penggunaan Listrik Tinggi
31/12/22 19:00:04	222 V	1.8 A	406 VA	29 C°	0	246	Aman
31/12/22 18:00:04	221 V	1.1 A	241 VA	31 C°	0	297	Temperatur Tinggi
31/12/22 17:00:04	220 V	0.96 A	211 VA	26 C°	0	296	Aman

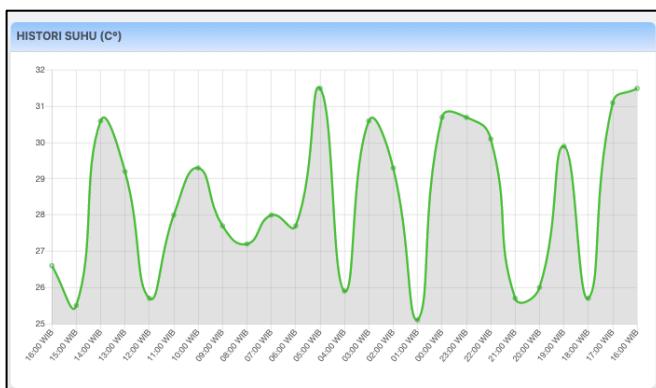
Gambar 4.21. Tampilan log notifikasi

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.



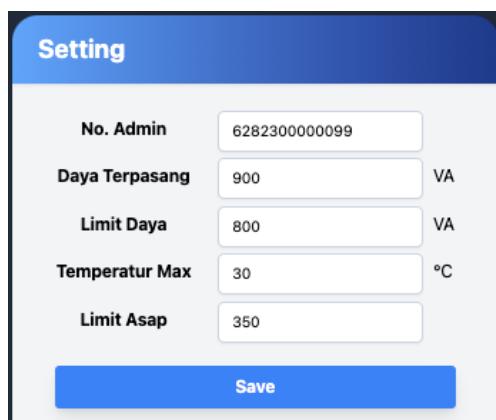
Gambar 4.22. Histori penggunaan daya listrik

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.



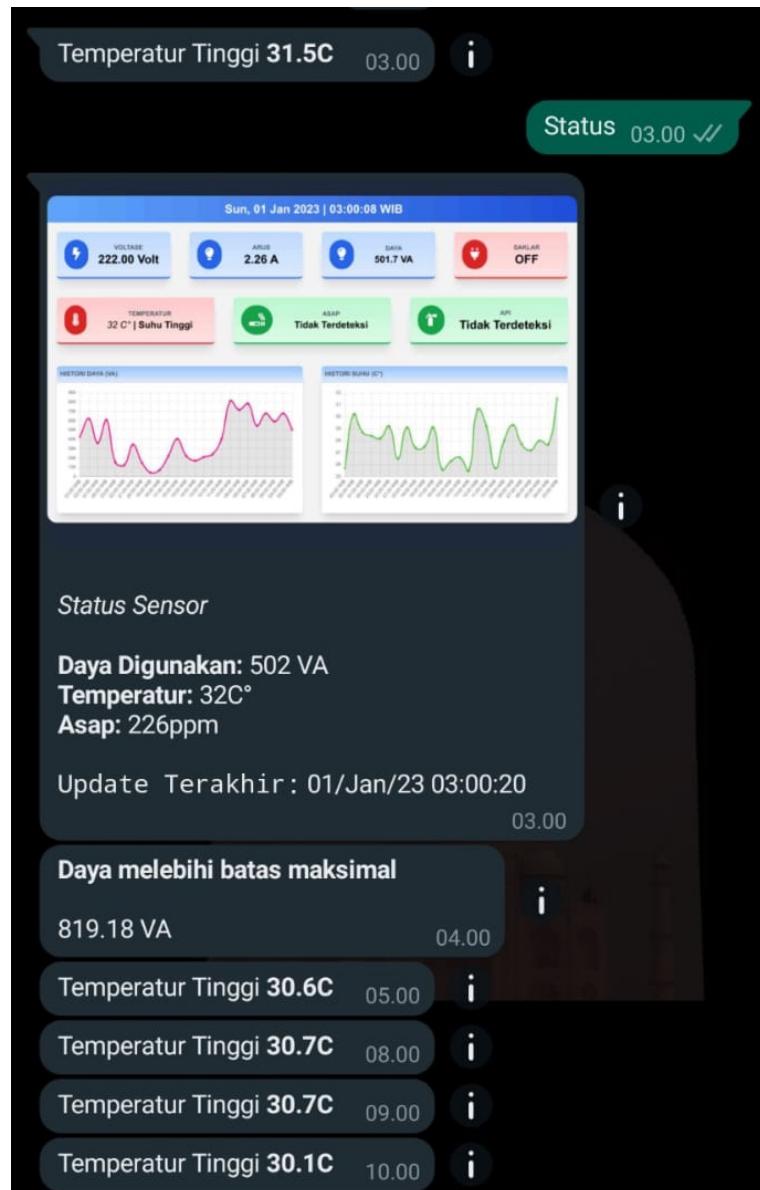
Gambar 4.25. Notifikasi Pada sisi Pengguna

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 Pengujian 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.

#### **4.5. 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**

#### **5.1. 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.
- 4.

#### **5.2. 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 trigger.
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 respon yang 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](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 Pendekripsi 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 Pendekripsi 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](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

**REKAPITULASI DATA KEBAKARAN PEMUKIMAN PERTAHUN  
KABUPATEN INDRAGIRI HILIR  
TAHUN 2016**

No	Nama Kecamatan	Jumlah Kejadian	Penyebab Kebakaran			Jumlah Korban (Jiwa)			Jumlah Sarana Yang Rusak/Hangus Terbakar				Prediksi Nilai Kerugian
			Korsleting Listrik	Tabung Gas	Lain-lain	Meninggal	Hilang	Luka	Rumah Tinggal	Ruko/ Kios	Gedung	Lain-lain	
1	Januari	3	3	-	-	-	-	-	2	-	-	-	1 Gudang Rp 350,000,000
2	Februari	2	2	-	-	-	-	-	8	-	-	-	Rp 650,000,000
3	Maret	7	7	-	-	-	-	2	27	-	-	-	Rp 1,456,000,000
4	April	3	3	-	-	-	-	-	3	-	-	-	Rp 325,000,000
5	Mei	1	1	-	-	-	-	-	15	-	-	-	Rp 900,000,000
6	Juni	-	-	-	-	-	-	-	-	-	-	-	Rp -
7	Juli	-	-	-	-	-	-	-	-	-	-	-	Rp -
8	Agustus	-	-	-	-	-	-	-	-	-	-	-	Rp -
9	September	-	-	-	-	-	-	-	-	-	-	-	Rp -
10	Okttober	-	-	-	-	-	-	-	-	-	-	-	Rp -
11	November	-	-	-	-	-	-	-	-	-	-	-	Rp -
12	Desember	-	-	-	-	-	-	-	-	-	-	-	Rp -
<b>Total</b>		<b>16</b>	<b>16</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>2</b>	<b>55</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>Rp 3,681,000,000</b>

**REKAPITULASI DATA KEJADIAN KEBAKARAN PEMUKIMAN PERTAHUN  
KABUPATEN INDRAGIRI HILIR  
TAHUN 2017**

No	Bulan	Jumlah Kejadian	Penyebab Kebakaran			Jumlah Korban (Jiwa)			Jumlah Sarana Yang Rusak/Hangus Terbakar				Prediksi Nilai Kerugian
			Korsleting Listrik	Tabung Gas	Lain-lain	Meninggal	Hilang	Luka	Rumah Tinggal	Ruko/ Kios	Gedung	Lain-lain	
1	Januari	6	6	-	-	1	-	1	-	-	-	-	1 Tongkang SPBB -
2	Februari	5	5	-	-	-	-	-	5	-	-	-	Rp 720,000,000
3	Maret	5	5	-	-	-	-	-	9	3	1	-	Rp 1,900,000,000
4	April	2	2	-	-	-	-	-	7	1	-	-	Rp 450,000,000
5	Mei	2	2	-	-	-	-	-	1	-	1	-	Rp 800,000,000
6	Juni	2	2	-	-	-	-	-	60	3	-	-	Rp 1,000,000,000
7	Juli	1	1	-	-	-	-	1	10	-	-	-	Rp 1,000,000,000
8	Agustus	2	2	-	-	-	-	-	1	8	3	-	Rp 950,000,000
9	September	4	4	-	-	-	-	-	13	-	-	-	Rp 855,000,000
10	Okttober	3	3	-	-	-	-	-	2	4	-	-	Rp 570,000,000
11	November	-	-	-	-	-	-	-	4	-	-	-	Rp 600,000,000
12	Desember	3	3	-	-	-	-	-	23	5	-	-	Rp 3,100,000,000
<b>Total</b>		<b>35</b>	<b>35</b>	<b>0</b>	<b>0</b>	<b>1</b>	<b>1</b>	<b>3</b>	<b>142</b>	<b>19</b>	<b>2</b>	<b>0</b>	<b>Rp 11,945,000,000</b>

**REKAPITULASI DATA KEBAKARAN PEMUKIMAN PERTAHUN  
KABUPATEN INDRAGIRI HILIR  
TAHUN 2018**

No	Bulan	Jumlah Kejadian	Penyebab Kebakaran			Jumlah Korban (Jiwa)			Jumlah Sarana Yang Rusak/Hangus Terbakar				Prediksi Nilai Kerugian
			Korsleting Listrik	Tabung Gas	Lain-lain	Meninggal	Hilang	Luka	Rumah Tinggal	Ruko/ Kios	Gedung	Lain-lain	
1	Januari	4	4	-	-	-	-	-	66	-	-	-	Rp 5,000,000,000
2	Februari	4	4	-	-	-	-	-	20	-	-	-	Rp 3,000,000,000
3	Maret	4	4	-	-	-	-	-	15	-	-	-	Rp 2,250,000,000
4	April	-	-	-	-	-	-	-	-	-	-	-	Rp -
5	Mei	5	5	-	-	-	-	-	17	-	-	-	Rp 2,550,000,000
6	Juni	2	2	-	-	-	-	-	9	-	-	-	Rp 1,350,000,000
7	Juli	-	-	-	-	-	-	-	-	-	-	-	Rp -
8	Agustus	-	-	-	-	-	-	-	-	-	-	-	Rp -
9	September	1	1	-	-	-	-	-	1	-	-	-	Rp 150,000,000
10	Okttober	-	-	-	-	-	-	-	-	-	-	-	Rp -
11	November	1	1	-	-	-	-	-	1	-	-	-	Rp 150,000,000
12	Desember	-	-	-	-	-	-	-	-	-	-	-	Rp -
<b>Total</b>		<b>21</b>	<b>21</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>129</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>Rp 14,450,000,000</b>

**REKAPITULASI DATA KEBAKARAN PEMUKIMAN PERTAHUN  
KABUPATEN INDRAGIRI HILIR  
TAHUN 2019**

No	Nama Kecamatan	Jumlah Kejadian	Penyebab Kebakaran			Jumlah Korban (Jiwa)			Jumlah Sarana Yang Rusak/Hangus Terbakar				Prediksi Nilai Kerugian
			Korsleting Listrik	Tabung Gas	Lain-lain	Meninggal	Hilang	Luka	Rumah Tinggal	Ruko/ Kios	Gedung	Lain-lain	
1	Januari	1	1	-	-	-	-	-	5	-	-	-	Rp 750,000,000
2	Februari	1	-	1	-	-	-	-	11	-	-	-	Rp 1,650,000,000
3	Maret	1	1	-	-	-	-	-	-	15	-	2 Gudang	Rp 2,000,000,000
4	April	-	-	-	-	-	-	-	-	-	-	-	Rp -
5	Mei	-	-	-	-	-	-	-	-	-	-	-	Rp -
6	Juni	4	4	-	-	-	-	-	4	-	-	-	Rp 600,000,000
7	Juli	9	9	-	-	-	-	-	13	160	-	-	Rp 12,000,000,000
8	Agustus	4	4	-	-	-	-	-	19	20	-	-	696 Los sembako
9	September	2	2	-	-	-	-	-	10	-	-	-	Rp 1,500,000,000
10	Okttober	2	2	-	-	-	-	-	1	1	-	-	Rp 250,000,000
11	November	-	-	-	-	-	-	-	3	-	-	-	Rp 450,000,000
12	Desember	-	-	-	-	-	-	-	-	-	-	-	Rp -
	Total	24	23	1	1	0	0	0	66	196	0	696	Rp 79,200,000,000

**REKAPITULASI DATA KEJADIAN KEBAKARAN PEMUKIMAN PERTAHUN  
KABUPATEN INDRAGIRI HILIR  
TAHUN 2020**

No	Nama Kecamatan	Jumlah Kejadian	Penyebab Kebakaran			Jumlah Korban (Jiwa)			Jumlah Sarana Yang Rusak/Hangus Terbakar				Prediksi Nilai Kerugian
			Arus Pendek Listrik	Tabung Gas	Lain-lain	Meninggal	Hilang	Luka	Rumah Tinggal	Ruko/ Kios	Gedung	Lain-lain	
1	Januari	1	1	-	-	-	-	-	1	-	-	-	Rp 70,000,000
2	Februari	2	2	-	-	-	-	-	1	20 Kios	-	-	Rp 1,300,000,000
3	Maret	2	2	-	-	-	-	-	8	1 Kios	-	-	Rp 200,000,000
4	April	2	2	-	-	-	-	-	8	1 Ruko	-	-	Rp 2,200,000,000
5	Mei	2	2	-	-	-	-	-	4	-	-	-	Rp 250,000,000
6	Juni	3	2	1	-	-	-	-	2	1 Toko	-	-	Rp 320,000,000
7	Juli	3	3	-	-	-	-	-	5	-	-	-	Rp 1,170,000,000
8	Agustus	6	6	1	-	-	-	-	82	-	-	-	Rp 2,780,000,000
9	September	2	2	-	-	3	-	-	3	2 Kios	-	-	Rp 150,000,000
10	Okttober	2	2	-	-	-	-	-	7	-	-	-	Rp 500,000,000
11	November	4	2	1	-	-	-	-	4	-	-	-	Rp 250,000,000
12	Desember	6	6	-	-	7	-	-	10	41 Kios	1 Wisma	1 Warung Kopi	Rp 2,000,000,000
	Total	35	32	3	0	10	-	-	135	66	1	1	Rp 11,190,000,000

**REKAPITULASI DATA KEJADIAN KEBAKARAN PEMUKIMAN PERTAHUN  
KABUPATEN INDRAGIRI HILIR  
TAHUN 2021**

No	Nama Kecamatan	Jumlah Kejadian	Penyebab Kebakaran			Jumlah Korban (Jiwa)			Jumlah Sarana Yang Rusak/Hangus Terbakar				Prediksi Nilai Kerugian
			Arus Pendek Listrik	Tabung Gas	Lain-lain	Meninggal	Hilang	Luka	Rumah Tinggal	Ruko/ Kios	Gedung	Lain-lain	
1	Januari	2	2	-	-	-	-	-	2	-	-	-	Rp 310,000,000
2	Februari	3	1	1	1	-	-	-	2	-	-	-	Rp 900,000,000
3	Maret	4	4	-	-	-	-	-	14	-	-	-	Rp 1,200,000,000
4	April	4	2	-	2	-	-	-	9	-	-	-	Rp 990,000,000
5	Mei	3	1	-	2	-	-	-	9	-	-	-	Rp 670,000,000
6	Juni	-	-	-	-	-	-	-	-	-	-	-	-
7	Juli	-	-	-	-	-	-	-	-	-	-	-	-
8	Agustus	-	-	-	-	-	-	-	-	-	-	-	-
9	September	-	-	-	-	-	-	-	-	-	-	-	-
10	Okttober	-	-	-	-	-	-	-	-	-	-	-	-
11	November	-	-	-	-	-	-	-	-	-	-	-	-
12	Desember	-	-	-	-	-	-	-	-	-	-	-	-
	Total	16	10	1	5	-	-	-	36	-	-	3	Rp 4,070,000,000

## 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 flameStateOne = digitalRead(flameOne);
    int flameStateTwo = digitalRead(flameTwo);
    int flameStateThree = digitalRead(flameThree);
    int flameStateFour = digitalRead(flameFour);
    int flameStateFive = digitalRead(flameFive);
    return flameStateOne + flameStateTwo + flameStateThree + flameStateFour +
flameStateFive;
}
```

## SENSOR PZEM VOLTAGE

```
#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://solistrik.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://solistrik.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://sulistrik.apiwa.tech/api/v1/data";
const String serverGet = "https://sulistrik.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 flameStateOne = digitalRead(flameOne);
    int flameStateTwo = digitalRead(flameTwo);
    int flameStateThree = digitalRead(flameThree);
    int flameStateFour = digitalRead(flameFour);
    int flameStateFive = digitalRead(flameFive);
    return flameStateOne + flameStateTwo + flameStateThree + flameStateFour +
flameStateFive;
}

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**

