BAB III

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Penulis bekerja pada Bidang Pengembangan dan Pengelolaan Sistem Informasi Pertahanan (BidBangola Sisfohan) pada divisi Subbidang Pengembangan dan Pengelolaan Sistem Aplikasi (Subbidang Bangola Siska) dibawah bimbingan Bapak Agus Wiji Suhariono S.Kom selaku pranata komputer terampil yang ditunjuk Bapak Eko Januardi S.Kom., M.A.P. selaku Kasubid (Kepala Sub Bidang).

Penulis ditugaskan untuk merancang suatu perangkat IoT yang dapat melakukan pemantau suhu, kelembapan dan api gas. Dalam perancangan, penulis berkoordinasi dengan Bapak Alif Bintoro A.Md.Kom Selaku Pranata TIK Subbid Siska Bid Banglolasisfohan yang memiliki tugas salah satunya adalah mengelola server Kemhan yang ada di Pusdatin untuk menghosting website tampilan. Penulis juga berkoordinasi dengan Bapak Rizqi Afdhani.ST selaku Pranata Komputer Ahli Pertama Pusdatin Kemhan untuk mengelola database yang didalamnya menyimpan data sensor seperti suhu, kelembapan, cahaya api dan gas, data ini nantinya ditampilkan pada halaman website.

3.2 Tugas dan Uraian Kerja Magang

Proyek perangkat IoT SiPantau merupakan salah satu proyek kerja dari divisi Subbidang Pengembangan dan Pengelolaan Sistem Aplikasi (Subbidang Bangola Siska), penulis tergabung dalam proyek perangkat IoT ini. Target dari proyek ini, diharapkan penulis dapat membuat rancangan prototipe alat pemantau rack data center berbasis IoT yang nantinya dapat membantu pengawas dalam melakukan pemantauan suhu, kelembapan, api dan gas.

3.2.1 Tugas Kerja Magang

Tugas yang dilakukan oleh penulis selama bekerja di Pusdatin Kemhan adalah merancang dan mengembangkan prototipe alat pengukur suhu, kelembapan dan api untuk ruangan data center milik Pusdatin Kemhan yang diberi nama SiPantau.

Tabel 2.1 Lini Masa Kerja Magang

Week	Onboarding	Virtual Design (Fritzing)	Hands on Design	Code program Software	Testing dan Fixing	Soldering	Finishing	Web Developing	Presentation
1.									
2									
3.									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									

Pada sesi *Onboarding*, penulis diminta untuk melakukan perkenalan diri dan dikenalkan lingkungan sekitar Pusdatin Kemhan untuk beradaptasi. Penulis juga dijelaskan mengenai prosedur kerja, aturan yang harus ditaati. Penulis melakukan studi literatur yang berkaitan dengan pembuatan perangkat monitoring *rack data center*. Penulis juga diberikan kesempatan untuk bertanya jika mengalami kesulitan.

Selama studi literatur, penulis mempelajari hal-hal yang berkaitan untuk pengembangan perangkat. Penulis mempelajari cara menggunakan mikrokontroler, mengkonfigurasi sensor, desain *website* dan pembuatan frontend *website*.

Penulis bekerja selama 16 minggu atau 3 bulan (31 Januari 2024-17 Mei 2024), berikut rangkuman kerja penulis.

Tabel 2.2 Rangkuman kerja

Minggu	Jenis Kegiatan Yang Dilakukan
1-2	- Perkenalan diri dan lingkungan tempat magang - Berdiskusi dengan supervisor dan pengawas ruang *data center* mengenai masalah yang ingin diselesaikan dan mencari solusinya. - Mendata alat dan bahan yang dibutuhkan dalam pembuatan project dan membelinya. - Memulai membuat rancangan skematik perangkat menggunakan aplikasi fritzing. - Memulai mengimplementasikan desain skematik perangkat dengan menggunakan papan *breadboard*.
3-5	- Berdiskusi dengan supervisor mengenai desain skematik yang sudah dirancang menggunakan *breadboard* - Mendesain ulang bentuk *case box* - Memulai *soldering* - Memulai melakukan pengkodean kode program mikrokontroler - Berdiskusi dengan supervisor mengenai *database* dan *web interface*

6-9	- Melakukan tes fungsi perangkat - Berdiskusi dengan supervisor mengenai perangkat - Finishing perangkat. - Mendesain *web Interface* - Melakukan implementasi *web interface* dan pengkodingan *Website*
10-15	- Melakukan implementasi *web interface* dan pengkodingan *Website* - Melakukan tes fungsi perangkat kembali - Berkoordinasi dengan bapak Rizqi untuk melakukan tes kirim data dari perangkat ke *database*. - Berkoordinasi dengan Pak Alif untuk melakukan hostingan. - Melakukan finishing pada perangkat dan web interface
16	Finishing perangkat dan *website* Presentasi atau pemaparan di kantor terkait produk yang telah dibuat

3.2.2 Uraian Kerja Magang

Dalam pengembangan perangkat IoT SiPantau, fitur yang tersedia telah disesuaikan dengan apa yang dibutuhkan oleh pengawas *rack data center Kemhan* di Pusdatin Kemhan. Dalam pengembangannya, penulis hanya diminta untuk membuat rangkaian perangkat yang siap pakai dan *website interface*. Untuk *database backend* penulis berkoordinasi dengan Bapak Rizqi karena *database* menggunakan *database management system* milik Kemhan. Dalam pembuatan dan pengelolaan *database* sangat dirahasiakan karena dikhawatirkan dapat terjadinya kebocoran data milik negara. Sama seperti *database backend*, untuk *hosting* website, penulis berkoordinasi dengan Bapak Alif karena *website* akan di *hosting* menggunakan *server* pribadi Kemhan sehingga untuk akses *server* hanya bisa dilakukan oleh pekerja di Pusdatin Kemhan. Fitur yang tersedia dalam perangkat IoT SiPantau adalah sebagai berikut.

1) Memiliki kemampuan pengukuran suhu pada *range* -40C hingga 80C. Pengukuran suhu dapat dilakukan hingga 20 meter. *Range* suhu tersebut sesuai dengan kriteria *maximal* suhu yang telah ditentukan pengawas *rack data center* yaitu 25C.
2) Memiliki kemampuan untuk melakukan pengukuran tingkat kelembapan pada range 0% hingga 100% RH. Pengukuran kelembapan dapat dilakukan hingga 20 meter. *Range* kelembapan tersebut sudah sesuai dengan kriteria minimal yang telah ditetapkan oleh pengawas *rack data center* pusdatin yakni 40% dan maximal 60%.
3) Memiliki kemampuan untuk mendeteksi adanya cahaya api dan kebocoran gas.

Dalam pengembangan perangkat IoT SiPantau, penulis menerapkan metode yang hampir sama dengan metode prototipe yaitu Requirements gathering and analysis, Quick design, Build prototype, User evaluation, Refining prototype, Implement product and maintain melakukan [1]. Namun penulis melakukan penyesuaian tahapan, berikut tahapan yang penulis gunakan:

1) Analisis permasalahan dan kebutuhan

Sebelum memulai pembuatan dan pengembangan perangkat IoT, penulis melakukan diskusi dengan supervisor dan perwakilan pengawasan *rack data center* untuk membicarakan permasalahan yang ada dan apa yang dibutuhkan. Dalam diskusi tersebut didapatkan permasalahan yakni, pernah terjadi overheat pada *rack data center* yang disebabkan tidak berfungsinya AC sentral pada ruang *data center* sehingga pekerja di Pusdatin Kemhan tidak dapat mengakses *server*, *website* dan akses *big data*. *Overheat* yang terjadi tidak diketahui oleh pengawas ruangan karena pengukur suhu pada ruangan data center yang sudah *built in* pada AC sentral ikut tidak berfungsi. Berdasarkan permasalahan tersebut, pengawas ruang *data center* meminta penulis untuk membuat suatu perangkat yang dapat melakukan pemantauan suhu kelembapan yang dapat diakses online dan memiliki fitur riwayat dengan tujuan agar dapat dilakukan pemantauan yang lebih *fleksibel* dan lebih sigap serta dapat melakukan rekap riwayat guna memonitoring pergerakan suhu kelembapan pada rack data center.

Pada tahap ini, setelah penulis selesai merancang perangkat maupun *website interface*, penulis melakukan diskusi untuk meminta evaluasi dengan meminta masukan

dan saran yang berkaitan dengan perangkat yang telah dirancang maupun website. Hal ini dilakukan agar penulis dapat lebih mengembangkan perangkat dan website sesuai dengan kebutuhan pengawas ruang data center kemhan.

2) Melakukan perancangan virtual (*Virtual Design*)

Sebelum melakukan perakitan penulis melakukan perancangan skematik rangkaian dengan menggunakan aplikasi simulator *fritzing* untuk mensimulasikan rangkaian. Hal ini dilakukan untuk menghindari resiko terjadinya korsleting saat melakukan perakitan. Sebelum perancangan dilakukan, penulis telah berdiskusi dengan supervisor dan pengawas ruang data center, dalam diskusi tersebut disepakati perangkat menggunakan 3 sensor dan untuk mikrokontroler menggunakan ESP 32, berikut sensor dan mikrokontroler yang digunakan:

- Sensor DHT 21

  Sensor DHT 21 merupakan salah satu jenis sensor suhu yang sering digunakan untuk mengukur suhu dan kelembapan pada suatu ruangan. Sensor ini memiliki kemampuan pengukuran suhu pada jangkauan -40ºC hingga 80ºC, kelembapan 0% hingga 100%. Jangkauan suhu dan kelembapan tersebut sesuai dengan kriteria maksimal dan minimal suhu dan kelembapan yang telah ditentukan pengawas *rack data center* yaitu 25ºC untuk suhu maximal, 40% hingga 60% untuk kelembapan.

- Sensor Gas MQ-2

  Sensor MQ-2 merupakan sensor yang digunakan untuk mengukur konsentrasi dari beberapa gas seperti alkohol, karbon Monoksida, Metana, Propana sehingga cocok digunakan untuk mendeteksi asap yang kemungkinan dapat muncul dari *rack data center* apabila *rack* terjadi kebakaran atau korsleting.

- Flame sensor KY-026

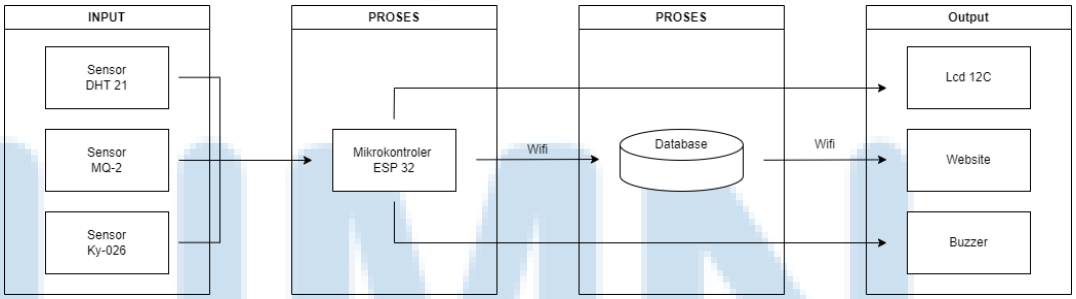
  Sensor KY-026 merupakan sensor yang dapat mendeteksi cahaya inframerah yang dipancarkan oleh api, sehingga sensor ini digunakan untuk

membantu pengawas dalam mendeteksi kebakaran yang bisa saja timbul dari *rack data center.*

- Mikrokontroler ESP32

  Sebelumnya pengawas ruangan mengusulkan untuk menggunakan mikrokontroler yang sudah ada yaitu ESP8266. Namun karena perangkat IoT SiPantau ingin dibuat menjadi aplikasi yang realtime, maka diputuskan untuk menggunakan ESP32 karena memiliki daya tangkap wifi yang lebih baik karena sudah menggunakan chip Wifi HT-40 yang memiliki penangkapan sinyal Wifi lebih baik dibanding ESP8266 yang masih menggunakan Wifi HT-20. Tidak hanya itu ESP32 memiliki lebih banyak pin GPIO 36 pin GPIO (General Purpose Input Output), sehingga bisa lebih banyak diberikan sensor atau modul lainnya jika dibutuhkan dan memiliki bluetooth yang kemungkinan untuk kedepannya bisa digunakan oleh pengawas ruang untuk melakukan penambahan fitur atau hal lainnya.

Berikut skema diagram blok dari perangkat SiPantau



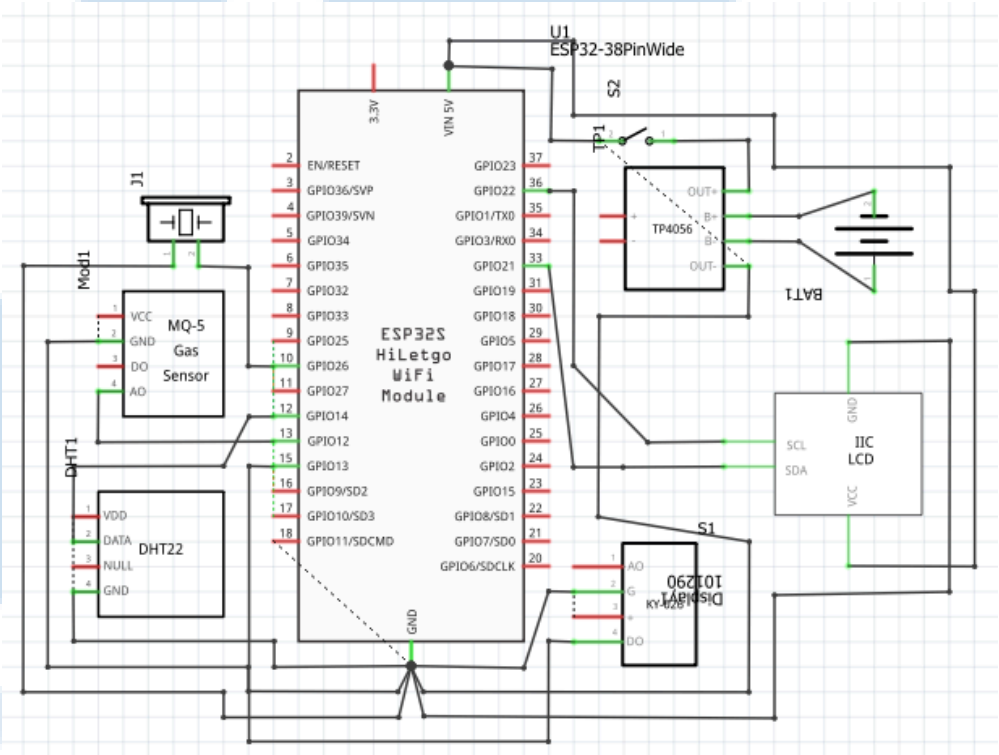
Gambar 3.1 Diagram Blok Sistem Perangkat

Penjelasan:

- Ketiga sensor yang digunakan yakni sensor DHT21, sensor MQ-2, sensor ky-026 merupakan sumber input data yang nantinya akan dikirimkan menuju Mikrokontroler ESP 32 untuk dilakukan pemprosesan.
- Ketiga input data yang telah dikirimkan dari 3 sensor tadi, akan diproses dan dikelola pada mikrokontroler esp 32. Lalu data inputan ini akan menghasilkan output sesuai yang diperintahkan.

- Data suhu, kelembapan, cahaya api dan gas yang diterima oleh mikrokontroler akan dikirimkan ke database untuk dapat ditampilkan pada halaman *website*. Hal ini dapat dilakukan apabila mikrokontroler ESP 32 telah terhubung pada jaringan internet dan terhubung pada database.
- Penulis juga merancang alat agar dapat mengirimkan output secara langsung dengan buzzer. Penulis telah melakukan pengkodean ketika sensor MQ-2 mendeteksi terdapatnya gas, maka buzzer akan berbunyi. Penulis juga melakukan pengkodean ketika sensor Ky-026 mendeteksi adanya cahaya api, maka buzzer akan berbunyi. Untuk suhu penulis tidak melakukan pengkodean output buzzer karena pengawas ruangan meminta untuk tidak diberikan output secara langsung, karena pengawas juga melakukan pemantauan via suara "bip" yang terdapat pada *rack data center* sehingga ditakutkan akan mengganggu pemantauan. Penggunaan buzzer hanya untuk darurat saja seperti terdapatnya api.

Berikut skematik rangkaian yang telah dibuat oleh penulis.



Gambar 3.2 Rancangan Skematik Rangkaian

Pengawas ruang mengharapkan perangkat yang dirancang dapat tetap aktif saat digunakan dalam mode portabel atau tanpa harus disambung secara langsung dengan terminal listrik. Hal ini bertujuan sebagai tindakan preventif ketika listrik yang berada pada ruangan data center mengalami pemadaman listrik. Pengawas ruang memberikan 2 buah baterai dengan tipe Li-ion 18650 dengan kapasitas total 7200 kepada penulis agar dapat digunakan sementara dalam melakukan perancangan masukan daya atau power pada prototipe perangkat. Penulis juga menambahkan modul charger TP4056 untuk modul *charging* baterai serta sebagai input power untuk perangkat. Penulis memilih TP4056 dikarenakan modul *charger* ini memiliki fitur proteksi yang dimana memberikan keamanan lebih saat melakukan charging karena ketika baterai dalam keadaan sudah penuh, aliran listrik akan disalurkan langsung ke beban sehingga baterai tidak akan mengalami kerusakan. Hal ini diperlukan karena nantinya perangkat IoT SiPantau akan dibiarkan selalu hidup sehingga memberikan rasa untuk pengawas. Penulis juga menambahkan saklar dengan 2 pin kaki dengan tujuan sebagai tombol *emergency* apabila terjadi keadaan darurat seperti terjadi korslet pada perangkat SiPantau.

Untuk sistem *output* secara langsung dari perangkat, penulis juga menambahkan modul *buzzer* sebagai sarana *output* suara yang akan berfungsi apabila perangkat mendeteksi adanya api atau gas dari asap api. *Buzzer* tidak digunakan untuk memperingati ketika suhu atau kelembapan sudah mencapai batas atau melebihi batas yang sudah ditentukan karena akan menimbulkan *spam* bunyi apabila perangkat mendeteksi suhu dan kelembapan sudah di atas batas, tentunya hal ini dapat mengganggu pemantauan pengawas yang mengandalkan suara “bip” dari *rack data center*. Penulis juga menambahkan LCD 12c dengan tujuan agar pengawas dapat melihat secara langsung berapa tingkat suhu dan kelembapan serta pengingat terdapatnya api atau gas.

Berikut pin yang penulis gunakan dalam rangkaian akan dijelaskan dalam tabel berikut.

- **Sensor MQ-2**

Pin ESP32	Sensor MQ-2
Vin	Vcc
Gnd	Gnd
A12 / D12	A0

Tabel 3.1 Pin Yang Terhubung Antara Sensor MQ-2 Dengan ESP 32

- **Sensor KY-026**

Pin ESP32	Sensor KY-026
Vin	+
Gnd	Gnd
A13 / D13	A0

Tabel 3.2 Pin Yang Terhubung Antara Sensor KY-026 Dengan ESP 32

- **Sensor DHT21**

Pin ESP32	Sensor DHT21
Vin	+
Gnd	-
A13 / D13	Data

Tabel 3.3 Pin Yang Terhubung Antara Sensor KY-026 Dengan ESP 32

- **Buzzer**

Pin ESP32	**Buzzer**
Vin	+
Gnd	-
A26/D26	+

Tabel 3.4 Pin Yang Terhubung Antara Buzzer Dengan ESP 32

- **LCD I2C**

Pin ESP32	**LCD 12C**
Vin	Vcc
Gnd	Gnd
A22/D22	SCL
A2/D21	SDA

Tabel 3.5 Pin Yang Terhubung Antara LCD I12C Dengan ESP 32

- **Power**

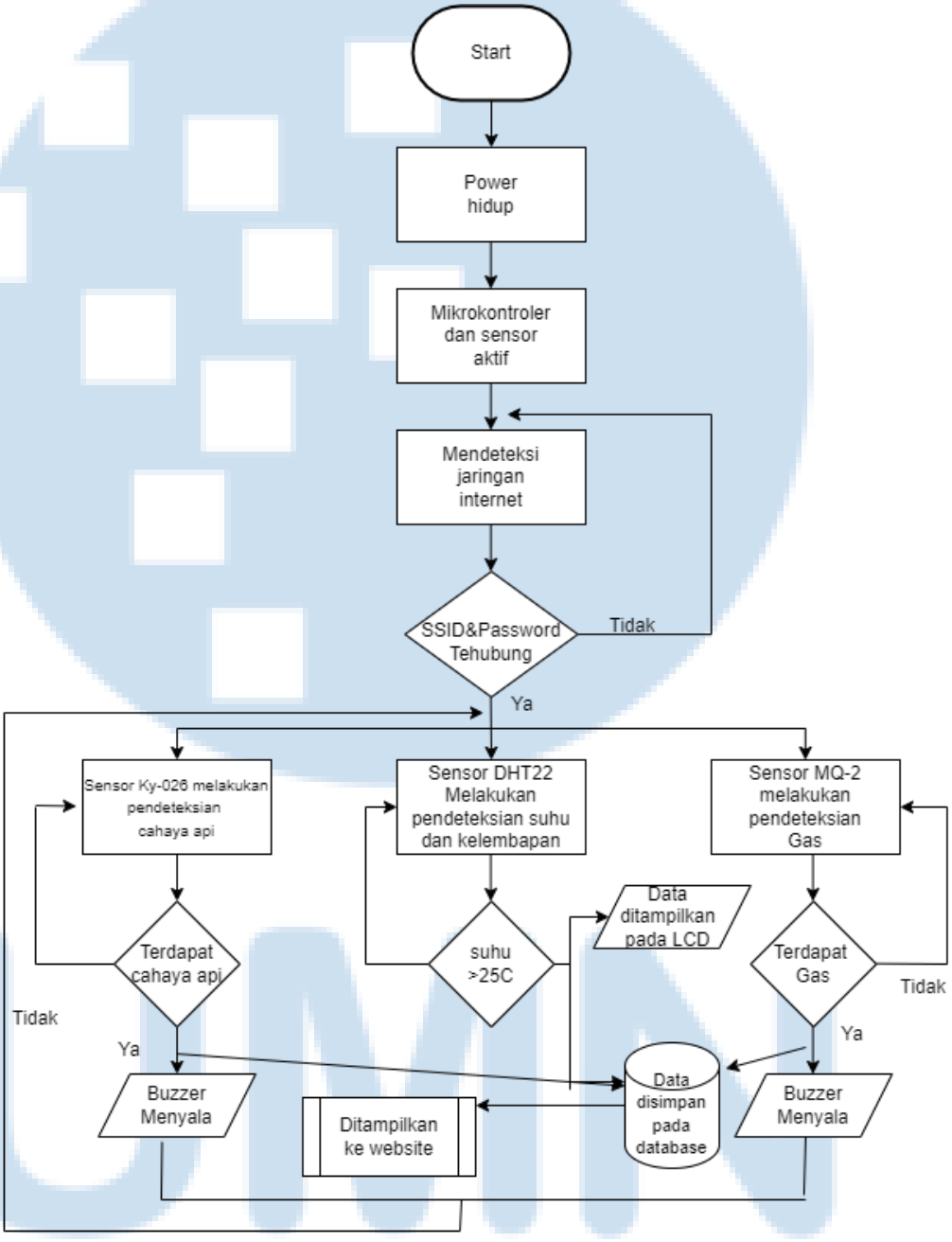
Pin ESP32	TP4056
Vin	+
Gnd	-

Tabel 3.6 Pin Yang Terhubung Antara Sensor TP4056 Dengan ESP 32

Pin TP4056	Baterai
Vin	Dihubungkan terlebih dahulu ke saklar lalu disambungkan ke pin + baterai.
Gnd	-

Tabel 3.7 Pin Yang Terhubung Antara Sensor TP4056 Dengan Baterai

Berikut alur proses kerja sistem



Gambar 3.3 Sistem Proses kerja sistem.

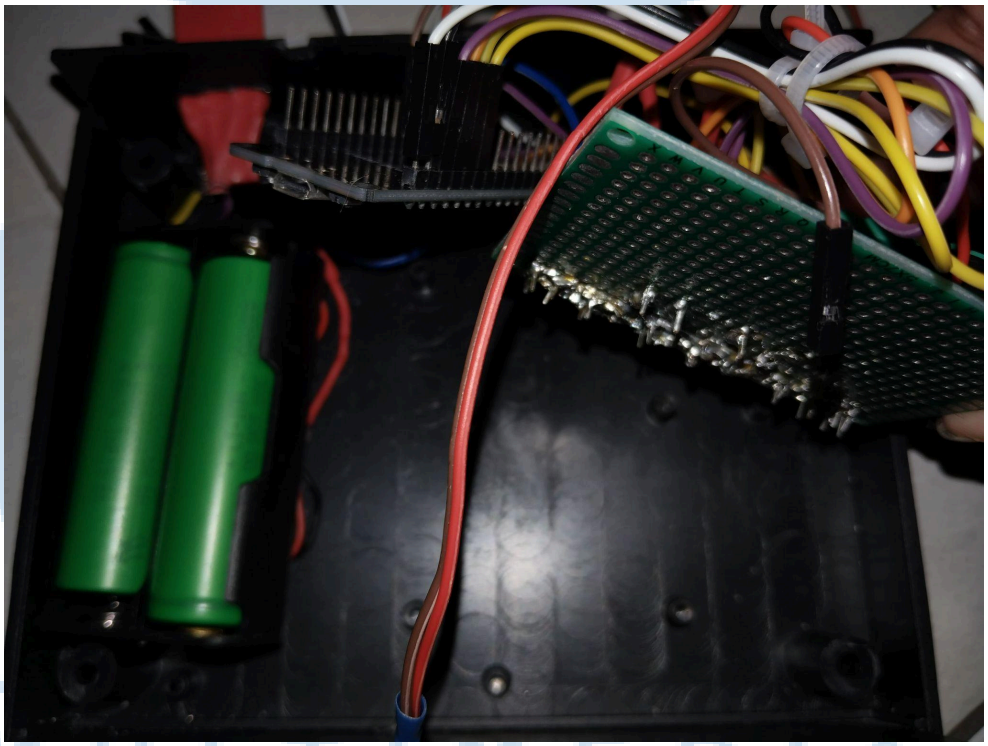
Penjelasan:

- Saat *power* hidup, mikrokontroler ESP32 dan sensor lainya akan aktif.
- Saat mikrokontroler ESP32 aktif, akan langsung menghubungkan ke wifi

- Ketika semua sensor dan mikrokontroler sudah aktif, semua data hasil pembacaan sensor seperti data suhu, kelembapan, cahaya api dan gas akan langsung dikirimkan ke database untuk ditampilkan ke halaman *website.* Data suhu dan kelembapan akan ditampilkan juga pada layar Lcd 12c.

3) Melakukan perancangan secara langsung (*hands on design*) dan menyolder (*soldering*)

Setelah melakukan perancangan skematik dan skematik tersebut telah disetujui, penulis melakukan tahap berikutnya yakni mencoba merangkai rangkaian berdasarkan rancangan skematik yang sudah di buat. Tahap ini dilakukan agar penulis dapat melakukan trial and error pada rangkaian. Setelah perakitan menggunakan *breadboard* dirasa sudah baik dan sesuai, penulis melanjutkan tahap selanjutnya yakni soldering. Pada tahap ini penulis melakukan soldering kabel pada papan PCB dengan tujuan agar kabel yang telah disusun tidak mudah lepas atau putus saat diimplementasikan secara langsung pada kotak *case*.



Gambar 3.4 Penulis Menyolder Rangkaian Perangkat

4) **Pengkodean rangkaian**

Dalam pengkodean rangkaian, penulis menggunakan papan mikrokontroler ESP32 sebagai media penyimpanan dan pemrosesan kode rangkaian. Penulis menggunakan aplikasi Arduino IDE sebagai media *compiler* kode yang akan dimasukkan ke dalam mikrokontroler ESP32. Sebelum melakukan pengkodean, penulis memasukkan *library* dari sensor yang akan digunakan dengan tujuan untuk memudahkan penulis dalam penulisan *sketch* atau program. Library yang dimasukan adalah *library* untuk LCD 12C, sensor DHT 21, sensor gas MQ2, *flame* sensor dan *buzzer.*

- **Melakukan *import library***

  Setelah memasukan *library*, *library* harus dilakukan *import*, untuk melakukannya dapat menggunakan function "#include", berikut kode untuk *include* yang telah ditulis oleh penulis:

```
#include <DHT.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>  //https://github.com/bbx10/WebServer_tng
#include <HTTPClient.h>
```

Gambar 3.5 Kode *Include*

- **Melakukan *define* pin**

  Setelah melakukan *import library*, selanjutnya diperlukan melakukan *define* pin yang terdapat pada mikrokontroler dengan pin yang ada pada sensor. Tujuan dari melakukan *define* ini ialah untuk mendefinisikan pin yang kita gunakan agar untuk kedepannya dalam melakukan pengkodean lebih mudah. Berikut kode *define* pin mikrokontroler yang ditulis oleh penulis.

```
#define DHTPIN 14      // Pin data DHT22 terhubung ke pin GPIO 4
#define DHTTYPE DHT22 // Jenis sensor DHT

#define FLAME_PIN 13  // Pin flame sensor terhubung ke pin GPIO 5
#define SMOKE_SENSOR_PIN 12 // Pin sensor asap terhubung ke pin GPIO 15
#define BUZZER_PIN 26       // Pin buzzer terhubung ke pin GPIO 18

#define I2C_ADDR    0x27 // Alamat LCD I2C
#define LCD_COLS    16   // Jumlah kolom pada LCD
#define LCD_ROWS    2    // Jumlah baris pada LCD
```

Gambar 3.6 Kode *Define* Pin

- **Inisialisasi Wifi**

Penulis melakukan inisialisasi wifi agar mikrokontroler dapat terhubung dengan Wifi dan *database*, berikut kode inisialisasi wifi yang telah penulis buat.

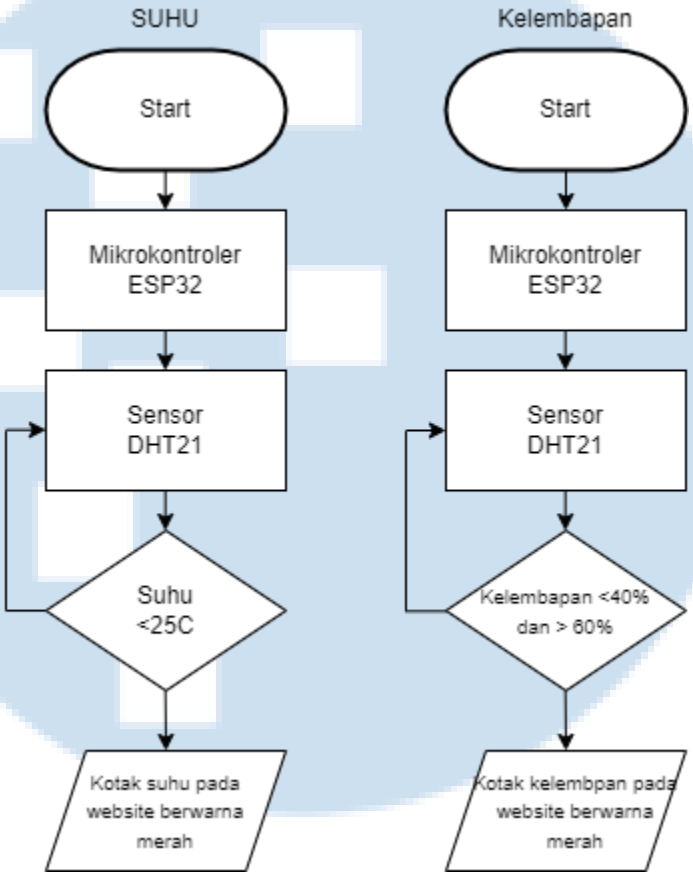
```
//wifi
const char* ssid = "Nama wifi";   // ganti dengan nama Wi-Fi hotspot
const char* password = "pasword wifi"; // ganti dengan password Wi-Fi
const char* host = "192.162.54.43"; // ip address komputer (lokal), jika database sudah di-hosting ganti ke alamat domain
IPAddress ip;
//------
```

Gambar 3.7 Kode Inisialisasi Wifi

Penjelasan:

- SSID digunakan untuk inisialisasi nama wifi, kita dapat memasukan nama wifi yang ingin digunakan.
- *Password* digunakan untuk inisialisasi password dari wifi yang akan kita gunakan.
- *Host* digunakan untuk inisialisasi *IP address* atau *domain* dari *database* management yang telah di-*hosting*. Tujuan dilakukan inisialisasi ini adalah untuk mengkoneksikan alat dengan *database* yang telah kita buat nantinya.

- **Kode DHT21**



Gambar 3.8 Gambar Flowchart Alur Sistem Kode Sensor DHT21

Penjelasan:

- Untuk suhu, penulis melakukan perkodean ketika sensor DHT21 mendeteksi suhu mencapai 25ºC dan lebih dari 25ºC maka kotak suhu pada halaman *website* akan berubah menjadi warna merah.
- Untuk kelembapan, penulis melakukan perkodean ketika sensor DHT21 mendeteksi kelembapan kurang dari 40% atau lebih dari 60% maka kotak kelembapan pada halaman *website* akan berubah menjadi warna merah.

Berikut code untuk DHT21

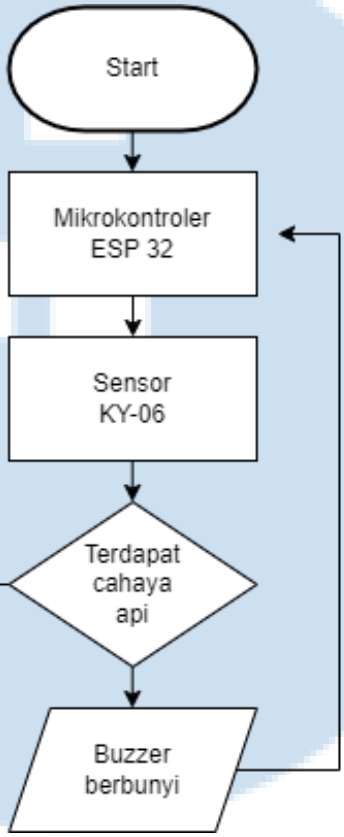
```
float temperature = dht.readTemperature(); // Membaca suhu dari sensor DHT22
float humidity = dht.readHumidity();       // Membaca kelembaban dari sensor DHT22

int flameValue = digitalRead(FLAME_PIN); // Membaca nilai dari sensor flame
int smokeValue = digitalRead(SMOKE_SENSOR_PIN); // Membaca nilai dari sensor asap

Serial.println("");

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Temp: " + String(temperature) + " C");
// lcd.print(temperature);
// lcd.print(" C");
Serial.println("Temp: " + String(temperature) + " C");

lcd.setCursor(0, 1);
lcd.print("Humidity: " + String(humidity) + "%");
// lcd.print(humidity);
// lcd.print("%");
Serial.println("Humidity: " + String(humidity) + "%");
```

Gambar 3.9 Kode Sensor DHT21

Penjelasan:

- float temperature = dht.readTemperature(); : barisan kode tersebut berfungsi untuk melakukan pembacaan nilai suhu dari sensor DHT21 dan menyimpannya dalam variabel *temperature* dalam tipe data *float*. Fungsi dht.readTemperature() digunakan untuk mendapatkan nilai suhu dalam derajat Celsius dari sensor.
- float humidity = dht.readHumidity(); : baris kode tersebut berfungsi untuk melakukan pembacaan nilai kelembapan dari sensor DHT21 dan menyimpannya dalam variabel *humidity*. Fungsi dht.readHumidity() digunakan untuk mendapatkan nilai kelembapan relatif (dalam persen) dari sensor.

- **Kode sensor api KY-026**



Gambar 3. 10 Flowchart Alur Sistem Kode Sensor KY-026

Penulis melakukan perkodean ketika sensor api KY-026 mendeteksi adanya cahaya api, maka mikrokontroler akan mengaktifkan buzzer sebagai alarm peringatan.

Penulis melakukan pengkodean untuk sensor api KY-026

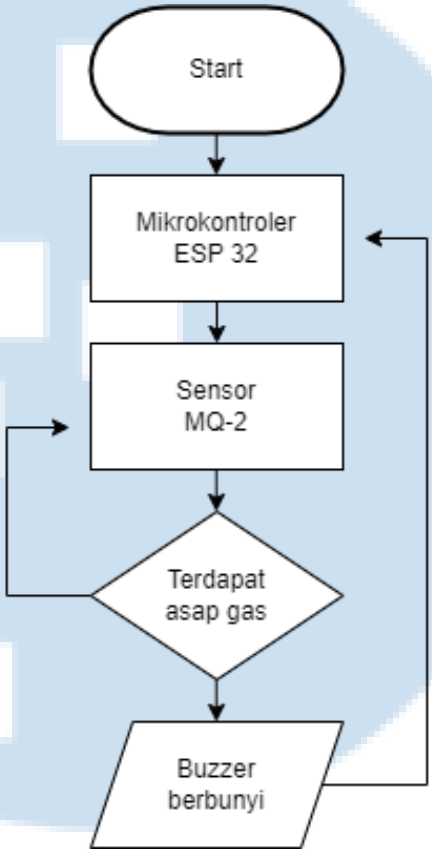
```
// Code sensor pendeteksi cahaya api
  flameValue = digitalRead(FLAME_PIN);
    if (flameValue == HIGH) {

      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Terdeteksi api");

      Serial.println("FLAME DETECTED!");
      digitalWrite(BUZZER_PIN, HIGH);
    } else {
      Serial.println("Tidak ada api.");
      digitalWrite(BUZZER_PIN, LOW);
    }
```

Gambar 3.11 Kode Sensor Api KY-026

Penjelasan:

- flameValue = digitalRead(FLAME_PIN); : pada baris kode ini nilai dari pin yang terhubung ke sensor api akan dibaca dan disimpan kedalam variabel flameValue. digitalRead(FLAME_PIN) akan mengembalikan HIGH jika sensor mendeteksi api, dan LOW jika tidak ada api.
- if (flameValue == HIGH) {, pada fungsi ini, program akan memeriksa apakah nilai flameValue adalah High yang diartikan terdapat api yang terdeteksi.

- **Kode sensor MQ-2**



Gambar 3.12 Flowchart Alur Sistem Kode Sensor MQ-2

Penulis melakukan perkodean ketika sensor api KY-026 mendeteksi adanya cahaya api, maka mikrokontroler akan mengaktifkan buzzer sebagai alarm peringatan.

Penulis melakukan pengkodean pada sensor MQ-2

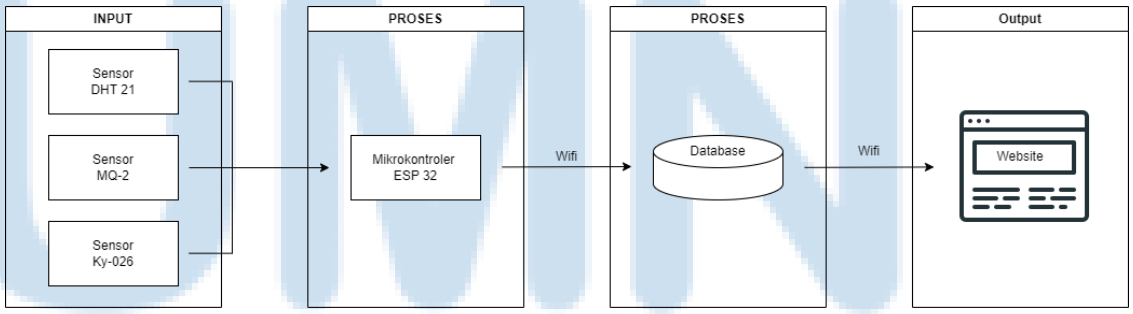
```
// Code mq2
  smokeSensorValue = analogRead(SMOKE_SENSOR_PIN);
  if (smokeSensorValue > 2000) {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Terdeteksi gas"); // Tampilkan pesan jika gas terdeteksi

    Serial.println("TERDETEKSI GAS!");
    digitalWrite(BUZZER_PIN, HIGH);
  } else {
    Serial.println("Tidak ada Gas.");
    digitalWrite(BUZZER_PIN, LOW);
  }
```

Gambar 3.13 Kode sensor MQ-2

Penjelasan:

- smokeSensorValue = analogRead(SMOKE_SENSOR_PIN); : baris kode ini akan melakukan pembacaan nilai analog dari pin yang terhubung ke sensor asap (SMOKE_SENSOR_PIN). Nilai ini disimpan dalam variabel smokeSensorValue.
- if (smokeSensorValue > 2000) {, fungsi ini berfungsi untuk melakukan pemeriksaan apakah nilai yang dibaca dari sensor lebih dari 2000, jika nilai yang dibaca lebih dari 2000 maka sensor mendeteksi adanya gas.

- **Kode Pengiriman Data Sensor**



Gambar 3.14 Flowchart Alur Sistem Kode Kirim Data Server Dari ESP

Penulis juga melakukan pengkodean untuk menghubungkan alat dengan database agar nantinya output seperti presentasi suhu, kelembapan, hasil pembacaan cahaya api dan gas dapat dikirimkan ke database secara langsung.

```
  // kirim data ke server
  WiFiClient client;
  const int httpPort = 80;
  if(!client.connect(host, httpPort)) {
    Serial.println("Connection to web failed.");
    return;
  }

  String Link;
  HTTPClient http;

  //ketika nama folder diganti, jangan lupa ganti nama(kemhan5)
  Link = "http://" + String(host) + "/kemhan5/kirimdata.php?suhu=" + String(temperature) + "&kelembapan=" + String(humidity)
  + "&api=" + String(flameValue) + "&asap=" + String(smokeSensorValue);
  http.begin(Link);
  http.GET();

  String respons = http.getString();
  Serial.println(respons);
  http.end();

  count = 0; //kirim ke database setiap 10 detik (iterasi)
}

delay(1000); // 1 menit
```

Gambar 3.15 Kode Pengiriman Data Sensor

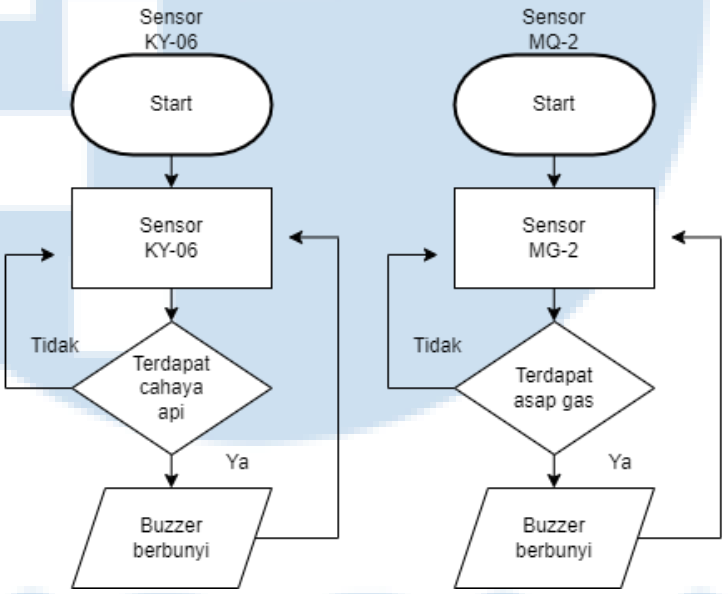
Penjelasan:

- WiFiClient client; : baris kode ini berfungsi untuk membuat objek WiFiClient yang akan digunakan untuk koneksi ke server.
- const int httpPort = 80; : baris kode ini berfungsi untuk menetapkan *port* HTTP (80) yang umum digunakan untuk koneksi web.
- if (!client.connect(host, httpPort)) { ... } : fungsi ini berfungsi untuk mencoba menghubungkan ke *host* pada *port* yang ditentukan. Jika gagal, akan mencetak pesan kesalahan dan keluar dari fungsi.
- String Link; : baris kode ini berfungsi untuk melakukan deklarasi variabel Link untuk menyimpan URL yang akan diakses.
- HTTPClient http; : baris kode ini berfungsi untuk membuat objek HTTPClient untuk mengelola permintaan HTTP.
- Link = "http://" + String(host) + "/kemhan5/kirimdata.php?suhu=" + String(temperature) + "&kelembapan=" + String(humidity) + "&api=" + String(flameValue) + "&asap=" + String(smokeSensorValue); : baris kode ini berfungsi untuk membangun URL lengkap dengan parameter *query* yang

mengandung data sensor (temperature, humidity, flameValue, dan smokeSensorValue).

- http.begin(Link); : baris kode ini berfungsi untuk memulai koneksi HTTP ke URL yang dibangun.
- http.GET(); : baris kode ini berfungsi untuk mengirimkan permintaan GET ke server.

- ***Buzzer***



Gambar 3.16 Flowchart Alur Sistem Kode Buzzer

Kode *buzzer* tidak berdiri sendiri, melainkan bersamaan dengan kode pada *flame sensor KY-026* dan sensor MQ-2, hal ini dikarenakan *buzzer* digunakan hanya sebagai hasil *output*. Penulis melakukan perkodean ketika sensor api mendeteksi terdapat cahaya api maka buzzer akan berbunyi. Penulis juga melakukan perkodean ketika sensor MQ-2 mendeteksi adanya asap gas maka buzzer akan berbunyi.

```
    Serial.println("FLAME DETECTED!");
    digitalWrite(BUZZER_PIN, HIGH);
  } else {
    Serial.println("Tidak ada api.");
    digitalWrite(BUZZER_PIN, LOW);
  }
```

Gambar 3.17 Kode *Buzzer* Untuk Sensor KY-26

```
    Serial.println("TERDETEKSI GAS!");
    digitalWrite(BUZZER_PIN, HIGH);
  } else {
    Serial.println("Tidak ada Gas.");
    digitalWrite(BUZZER_PIN, LOW);
  }
```

Gambar 3.18 Kode *Buzzer* Untuk sensor MQ-2

5) Uji coba dan perbaikan (*testing* and *fixing*)

Tahap ini sudah dilakukan penulis saat melakukan perancangan langsung menggunakan papan *breadboard* dan *soldering*. Tahap ini merupakan tahapan uji coba pada perangkat yang telah dirancang oleh penulis dengan tujuan untuk mengetahui apakah perangkat sudah berjalan dengan baik atau belum. Pada saat kegiatan uji coba berlangsung, penulis didampingi langsung oleh supervisor dan pengawas ruang.

Dalam uji coba yang dilakukan oleh penulis, perangkat yang dirancang akan dibandingkan hasil pembacaan suhu dan kelembapannya dengan termometer digital milik Pusdatin. Saat uji coba berlangsung, perangkat yang dirancang dapat melakukan pendeteksian suhu dan kelembapan dengan baik, hal ini telah disampaikan dan disetujui oleh pengawas ruang saat kegiatan uji coba berlangsung. Penggunaan 2 buah baterai bertipe *Li ion* 18650 dengan kapasitas total 7200mAH pada rancangan prototipe perangkat, merupakan daya sementara yang digunakan penulis saat merancang masukan daya atau *power* pada perangkat, agar perangkat dapat digunakan dalam mode portabel. Penggunaan jenis baterai dan besaran kapasitas yang akan digunakan pada rancangan

final, akan ditentukan oleh tim *IoT developer* pada divisi Subbidang Bangola Siska dalam pengembangan lebih lanjut dari perangkat yang akan dilakukan oleh mereka. Karena penggunaan jenis baterai dan kapasitas belum diketahui pasti oleh penulis, maka penulis tidak melakukan pengujian dari daya tahan baterai saat perangkat digunakan dalam mode portabel. Walaupun begitu, pengukurannya dapat dilakukan dengan konsep berikut.

Konsep tersebut adalah dengan menentukan dahulu "rata-rata konsumsi daya per jam" dengan membagi "konsumsi daya selama sekian jam" dengan "jumlah jam pengukuran".

$$Rata - rata\ Konsumsi\ Daya\ Per\ Jam = \frac{Konsumsi\ Daya\ selama\ Sekian\ Jam}{Jumlah\ Jam\ Pengukuran}$$

Kemudian, hitung "daya tahan baterai (jam)" dengan membagi "kapasitas baterai (mAh)" dengan rata-rata "konsumsi daya per jam (W)" yang dikali dengan "konversi mAh ke W".

$$Daya\ Tahan\ Baterai\ (jam) = \frac{Kapasitas\ Baterai\ (mAh)}{Rata{-}rata\ Konsumsi\ Daya\ Per\ Jam\ (W) \times Konversi\ mAh\ ke\ W}$$

"Konversi mAh ke W" dapat dilakukan dengan membagi Kapasitas Baterai (mAh) yang dikali dengan Tegangan Baterai (V), dengan 1000.

$$Daya\ (W) = \frac{Kapasitas\ Baterai\ (mAh) \times Tegangan\ Baterai\ (V)}{1000}$$

Dengan konsep ini, daya tahan baterai (jam) bisa didapatkan.

Untuk mengetahui perkiraan kapasitas baterai yang diperlukan untuk menyediakan daya yang cukup saat listrik padam, dibutuhkan durasi listrik padam di Pusdatin pada umumnya. Namun, saat penulis menanyakan hal ini kepada pengawas ruang, beliau tidak bisa memperkirakannya.

6) **Penyelesaian (*Finishing*)**

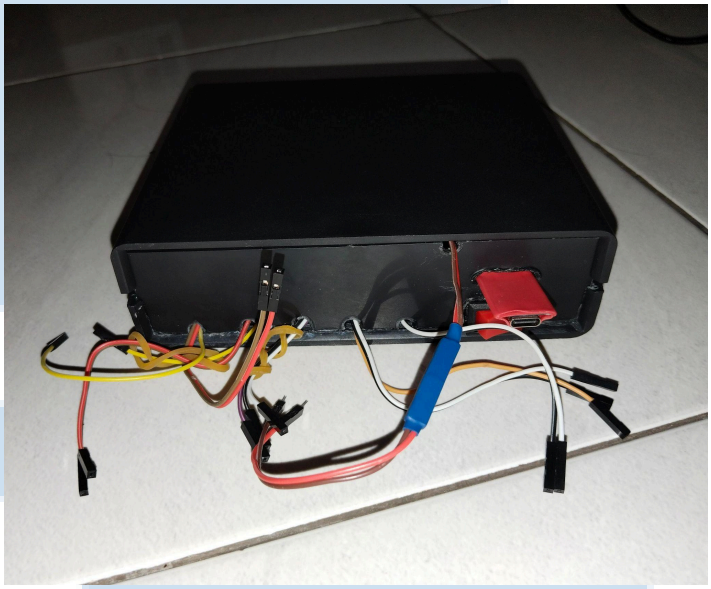
Pada tahap ini, penulis merapikan kabel pada perangkat dan diimplementasikan rangkaian pada kotak *case.*

- **Perapihan kabel**

Gambar 3.19 Perapihan kabel

Penulis melakukan perapihan kabel menggunakan kabel ties agar rangkaian kabel terlihat lebih rapih dan lebih terstruktur dan juga mempermudah melakukan perbaikan apabila perangkat SiPantau mengalami kerusakan.

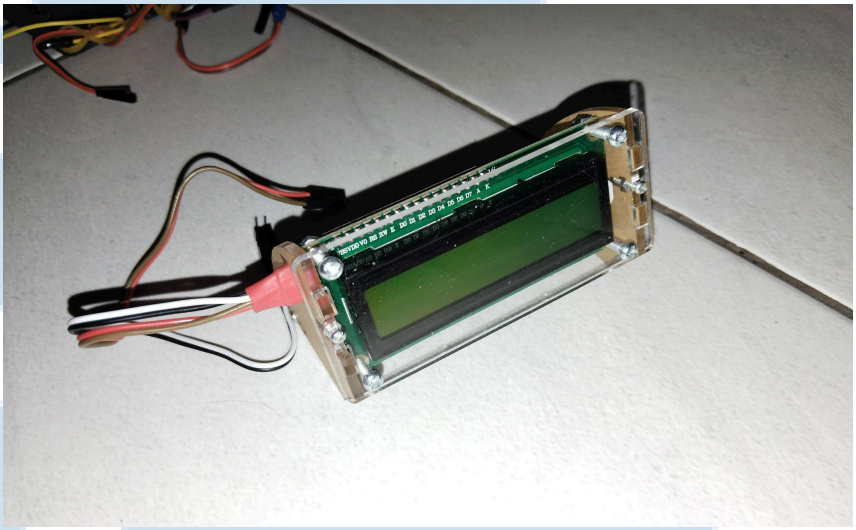
- **Desain kotak *case***



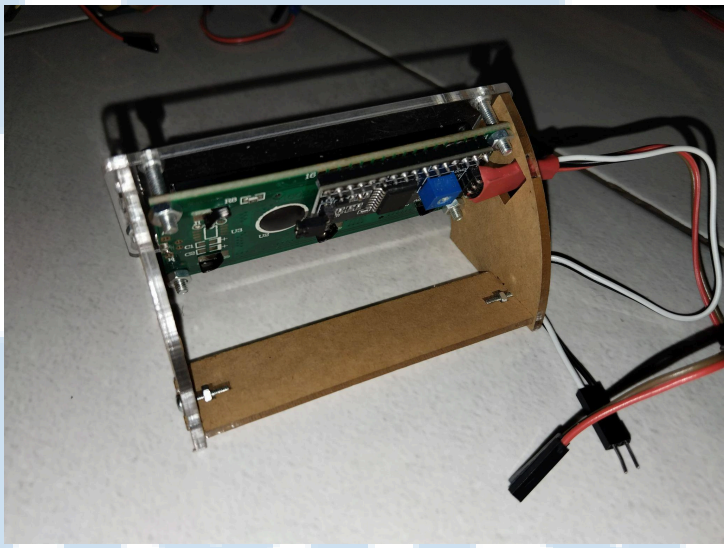
Gambar 3.20 Desain Kotak *Case*

Penulis menerapkan desain mini PC agar peletakannya fleksibel di mana saja. Penulis juga memberikan lubang pada sisi depan bawah untuk memberikan jalur kabel sensor nantinya.

- **Desain LCD 12C *Case***



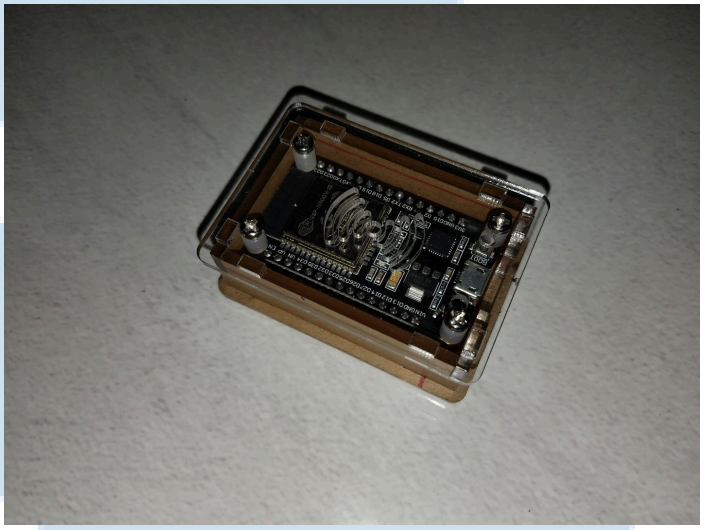
Gambar 3.21 Lcd *Stand Case* Tampak Depan



Gambar 3.22 Lcd Stand *Case* Tampak Belakang

Penulis merancang untuk Lcd tidak dijadikan satu dalam kotak *case* melainkan dirancang terpisah agar nantinya pengawas ruangan bisa meletakan posisi di luar rack atau pada posisi yang diinginkan. Penulis memberikan stand case dengan menggunakan akrilik.

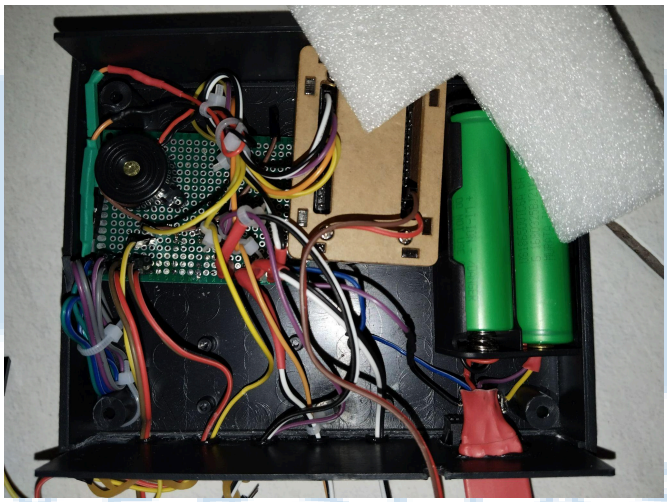
- **Desain kotak case untuk mikrokontroler ESP32**



Gambar 3.23 Kotak *Case* Mikrokontroler

Penulis memberikan kotak case pada mikrokontroler ESP 32 dengan tujuan untuk memberikan perlindungan lebih pada mikrokontroler ESP32 saat perangkat SiPantau terjatuh.

- **Desain Keamanan Tambahan**

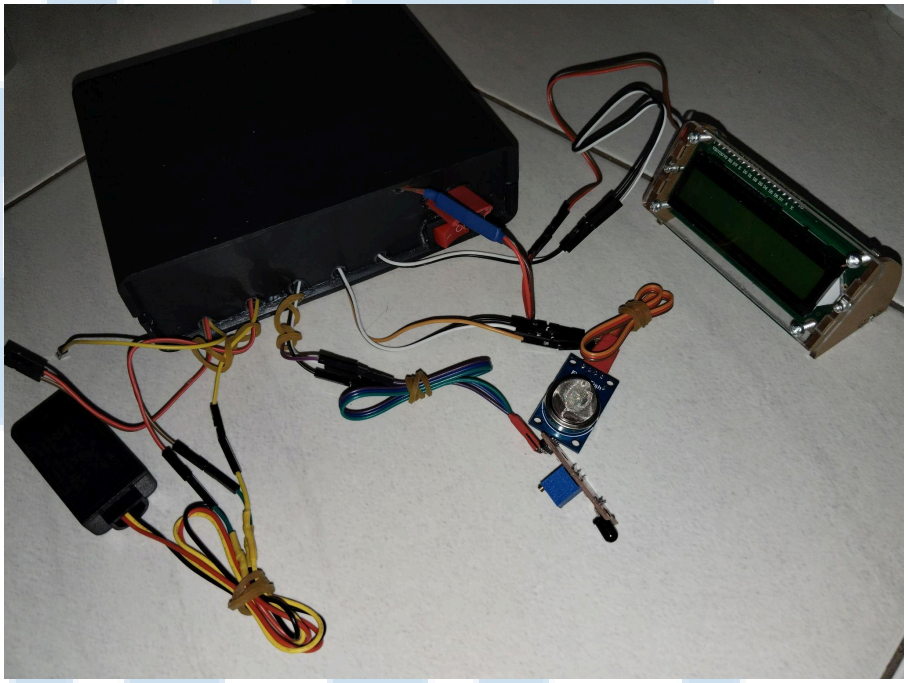


Gambar 3.24 Keamanan Tambahan

Pada bagian dalam perangkat SiPantau, penulis memberikan busa pelindung agar posisi baterai dan mikrokontroler tidak berpindah saat perangkat

SiPantau terjatuh dan menghindari benturan pada baterai dan mikrokontroler ESP32. Penulis juga memberikan heat shrink pada setiap ujung soket kabel agar setiap koneksi kabel tidak mudah terputus dan menghindari adanya arus pendek ketika antar kabel bersentuhan.

- **Desain Akhir Perangkat**



Gambar 3.25 Desain Akhir Perangkat

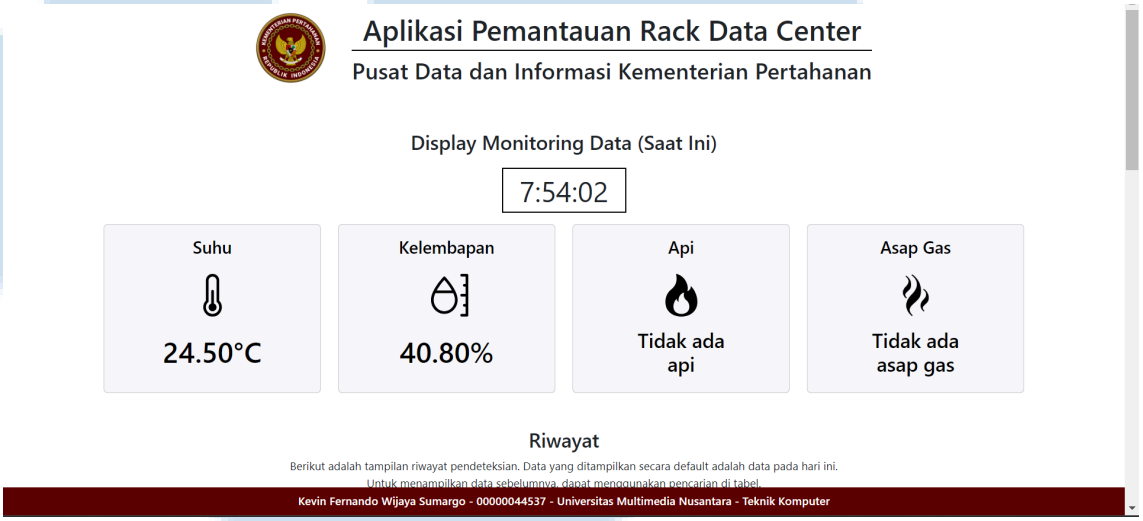
7) Pengkodean *Website*

Perancangan Frontend Halaman *Website Interface*

Untuk pengkodean antarmuka website, penulis menggunakan HTML untuk struktur web sesuai dengan permintaan pihak pengawas ruang, Bootstrap untuk *framework* CSS, dan JavaScript untuk menampilkan data suhu, kelembapan, api, dan gas secara *real time*. Halaman website ini nantinya akan dihosting melalui server pribadi Kemhan, yang akan dilakukan oleh Pak Alif Sehingga proses untuk hosting penulis tidak bisa tampilkan pada laporan ini karena tidak diperbolehkan untuk dipublikasi.

Untuk menampilkan semua data suhu, kelembapan, api dan gas, penulis menggunakan protokol Rest API. Halaman Website akan meminta data dari database untuk ditampilkan pada halaman *website.*

- **Rancangan Tampilan Bagian Indikator**



Gambar 3.26 Panel Indikator

Halaman ini menampilkan suhu, kelembapan, api, asap gas, dan waktu *real time*.

```
<div id="clock" class="mb-3"></div>
  <script>
    function startTime() {
      const today = new Date();
      let h = today.getHours();
      let m = today.getMinutes();
      let s = today.getSeconds();
      m = checkTime(m);
      s = checkTime(s);
      document.getElementById('clock').innerHTML =  h + ":" + m + ":" + s;
      setTimeout(startTime, 1000);
    }
    function checkTime(i) {
      if (i < 10) {i = "0" + i};  // add zero in front of numbers < 10
      return i;
    }
  </script>
</div>
```

Gambar 3.27 Kode untuk Jam

Kode ini mengatur jam *real-time* yang diperbarui setiap detik, menampilkan waktu saat ini dalam format "HH:MM:SS". Kode ini mencakup hal berikut:

- HTML: adalah pengganti jam.
- JavaScript:
  - startTime(): Mendapatkan waktu saat ini, memformatnya, dan memperbarui HTML bagian dalam elemen jam. Ia menggunakan setTimeout untuk memanggil dirinya sendiri setiap detik.
  - checkTime(i): Menambahkan angka nol di depan ke angka kurang dari 10.

Dengan pengaturan ini, jam akan mulai secara otomatis dan diperbarui setiap detik.

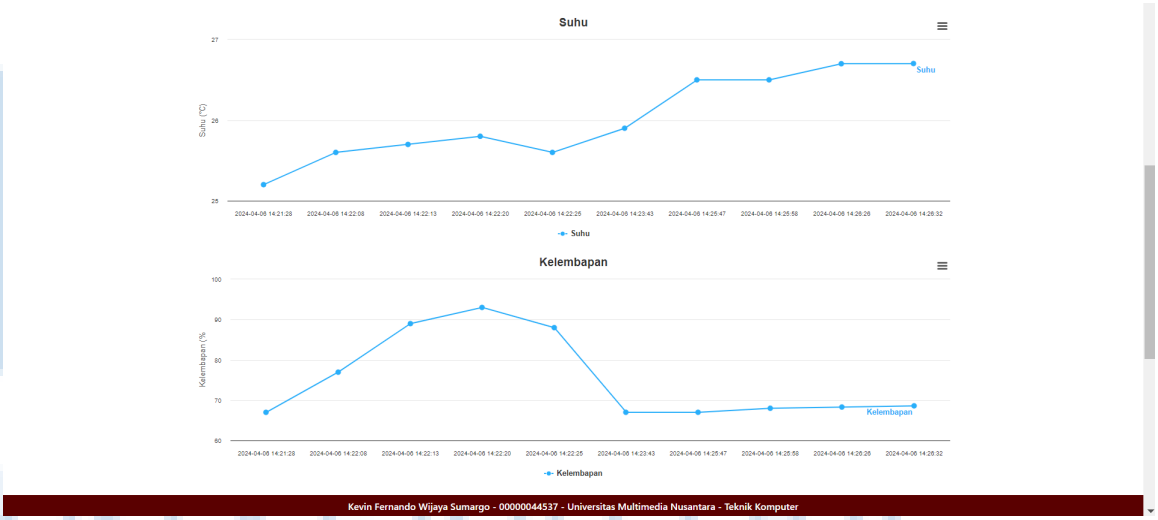
```
<script>
  $(document).ready(function() {
    setInterval(function() {
      $("#suhu").load("suhu.php");
      $("#kelembapan").load("kelembapan.php");
      $("#api").load("api.php");
      $("#asap").load("asap.php");
    }, 0 ); // delay 0 detik
  });
</script>
```

Gambar 3.28 Kode *Update* Isi *Card* Secara *Real-Time*

Kode ini menggunakan jQuery untuk memuat konten secara berkala dari file-file PHP yang mem-*fetch database* ke dalam elemen *card*.

- **Rancangan Tampilan Bagian Riwayat**

Riwayat ditampilkan dalam 2 bentuk, yaitu bentuk diagram *chart* dan juga dalam bentuk tabel. Untuk bentuk diagram, penulis menggunakan *tools plugin* dari Highcharts.



Gambar 3.29 Diagram *Chart* Riwayat

Chart tersebut menampilkan visualisasi nilai suhu dan kelembapan yang ditampilkan di tabel di bawahnya. Dengan mengklik tombol garis tiga di pojok

atas kanan, pengguna dapat mengunduh *chart* tersebut dalam berbagai macam format dan menge-*print*-nya.

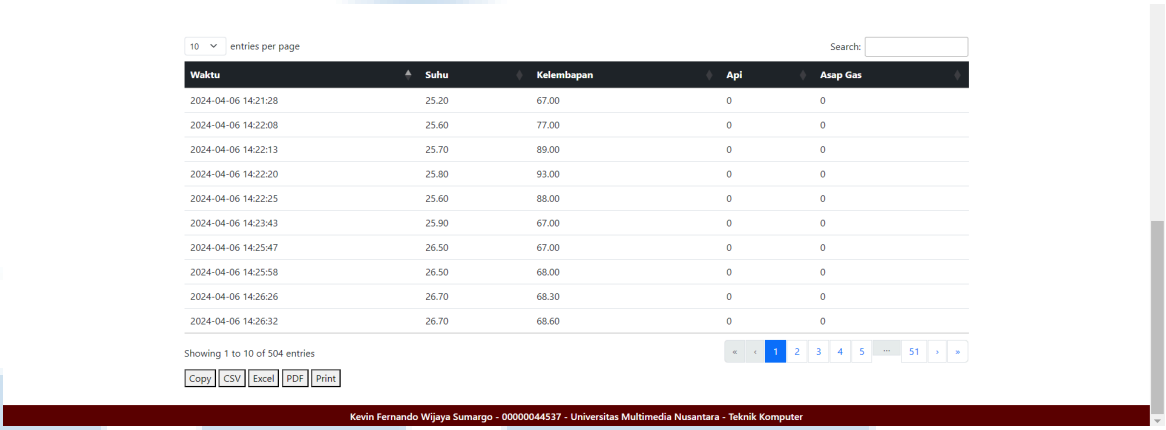
```
<script>
  function updateChartsFromTable() {
    var tableRows = document.querySelectorAll('#table tbody tr');

    var dataSuhu = [];
    var dataKelembapan = [];

    tableRows.forEach(function(row) {
      var waktu = row.cells[0].textContent;
      var suhu = parseFloat(row.cells[1].textContent);
      var kelembapan = parseFloat(row.cells[2].textContent);

      dataSuhu.push([waktu, suhu]);
      dataKelembapan.push([waktu, kelembapan]);
    });

    chartSuhu.update({series: [{data: dataSuhu }]});
    chartKelembapan.update({series: [{data: dataKelembapan}]});
  }

  var chartSuhu = Highcharts.chart('chartSuhu', {
    title: {text: 'Suhu'},
    xAxis: {type: 'category'},
    yAxis: {title: {text: 'Suhu (°C)'}},
    series: [{
      name: 'Suhu',
      data: []
    }],
    credits: {enabled: false}
  });

  var chartKelembapan = Highcharts.chart('chartKelembapan', {
    title: {text: 'Kelembapan'},
    xAxis: {type: 'category'},
    yAxis: {title: {text: 'Kelembapan (%'}},
    series: [{
      name: 'Kelembapan',
      data: []
    }],
    credits: {enabled: false}
  });
```

```
var observer = new MutationObserver(function(mutationsList, observer) {
  updateChartsFromTable();
});

observer.observe(document.getElementById('table').querySelector('tbody'), {
  childList: true,
  subtree: true
});

updateChartsFromTable();
</script>
```

Gambar 3.30 Kode *Chart*

Kode ini memperbarui *line chart* Highcharts secara dinamis berdasarkan data dari DataTable.

- fungsi updateChartsFromTable: Fungsi ini mengekstrak data dari baris tabel dan memperbarui grafik Highcharts.
- Inisialisasi Highcharts: Dua grafik diinisialisasi, chartSuhu dan chartKelembapan, dengan judul, sumbu, dan seri data kosong yang sesuai.
- MutationObserver: Pengamat ini mengamati perubahan pada isi tabel (seperti baris yang ditambahkan atau dihapus) dan memanggil updateChartsFromTable setiap kali perubahan terdeteksi.
- Pembaruan *Chart* Awal: updateChartsFromTable dipanggil satu kali pada awalnya untuk mengisi *chart* dengan data yang ada.



Gambar 3.31 Tabel Riwayat

Riwayat dalam bentuk tabel menggunakan *plugin* dari DataTable. Pengguna dapat memilih berapa banyak data yang ditampilkan, mencari data tertentu, mengurutkan setiap kolom secara naik atau turun, menyalinnya, mengunduhnya dalam berbagai format data, dan menge-*print*-nya.

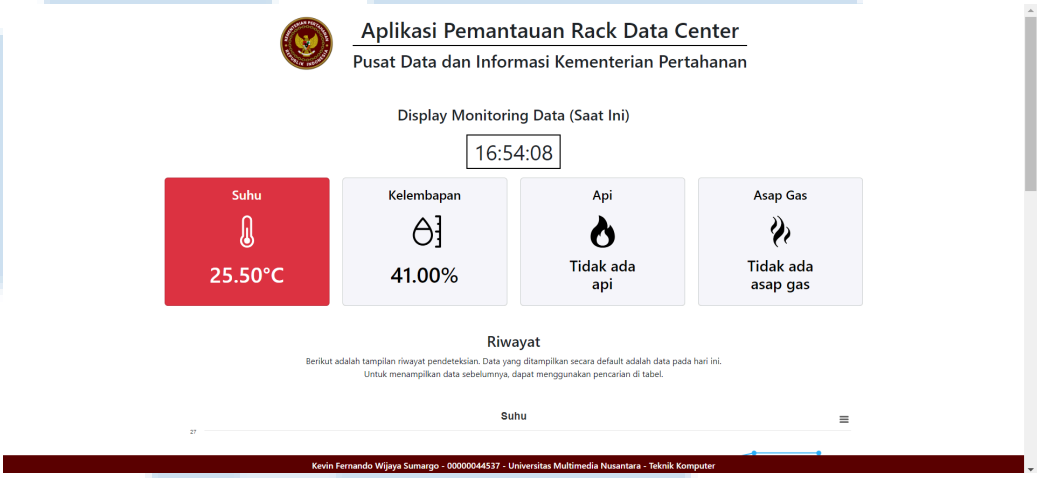
```
<?php include"tabel.php";?>

<script>
  $(document).ready(function() {
    $('#table').DataTable({
      ajax: 'server_processing.php',
      processing: true,
      serverSide: true,
      layout: {
        bottom2: {
          buttons: ['copy', 'csv', 'excel', 'pdf', 'print']
        }
      },
      createdRow: function(row, data, dataIndex) {
        if(data[1] >= 25.00 || data[2] < 40 || data[2] > 60 || data[3] === 1 || data[4] === 1) {
          $(row).addClass('table-danger');
        }
      }
    });
  });
</script>
```

Gambar 3.32 Kode Datatable

Kode ini mengintegrasikan *plugin* DataTables untuk menampilkan dan mengelola tabel dengan data yang diambil dari *server* menggunakan AJAX.

- **Rancangan Fitur Peringatan**

Penulis juga memberikan fitur peringatan, yaitu kotak yang berisikan hasil pengukuran suhu, kelembapan, api dan gas dapat berubah warna menjadi merah jika melewati batas aman.



Gambar 3.33 Fitur Peringatan Suhu Melewati Batas Yang Sudah Ditentukan

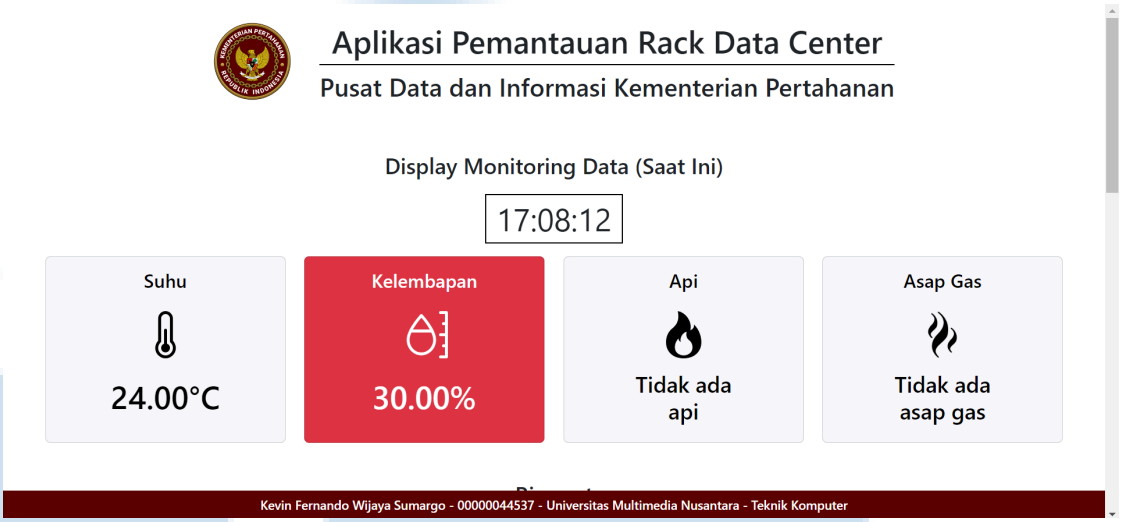
Penulis memberikan fitur kotak suhu yang akan berubah ketika suhu sudah lebih tinggi dari 25ºC.

```
<div class="card text-bg-light h-100 w-60" id="cardSuhu">
  <script>
    function updateCardClassSuhu() {
      fetch('suhu.php')
        .then(response => response.text())
        .then(suhu => {
          const card = document.getElementById('cardSuhu');
          if(parseFloat(suhu) >= 25) {
            card.classList.remove('text-bg-light');
            card.classList.add('text-bg-danger');
          } else {
            card.classList.remove('text-bg-danger');
            card.classList.add('text-bg-light');
          }
        });
    }
    setInterval(updateCardClassSuhu, 0);
  </script>
```

Gambar 3.34 Kode Ganti Warna *Card* Suhu

Kode ini melakukan hal-hal berikut.

- *Fetch* Suhu: Fungsi updateCardClassSuhu mem-*fetch* suhu *real-time* dari suhu.php.
- Perbarui *Class Card*: Ini memperbarui *class card* berdasarkan suhu. Jika suhunya 25°C atau lebih tinggi, kelas text-bg-danger akan ditambahkan; jika tidak, maka menggunakan kelas text-bg-light.

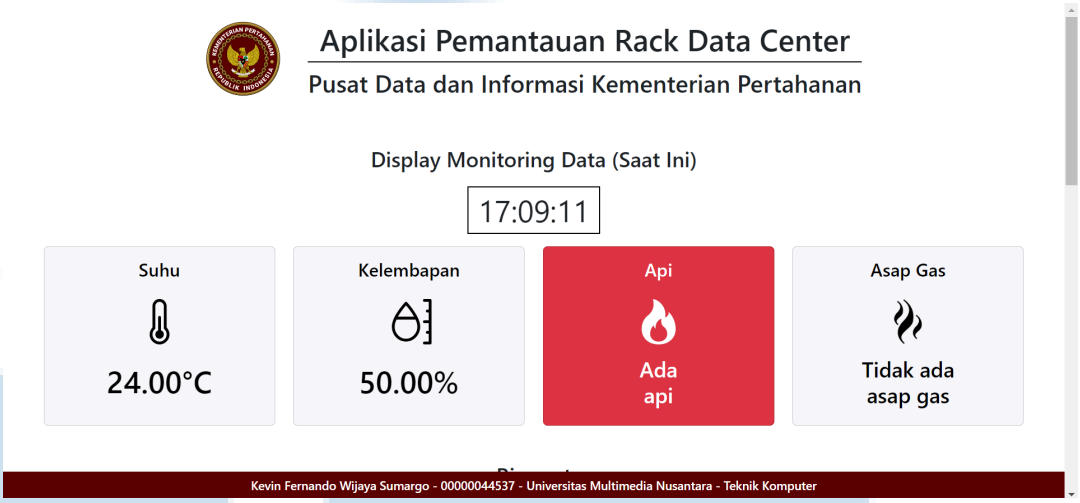


Gambar 3.35 Fitur Peringatan Kelembapan Melewati Batas Yang Sudah Ditentukan

Penulis memberikan fitur kotak kelembapan yang akan berubah menjadi merah ketika kelembapan sudah di luar batas yang ditentukan yaitu 40%-60%.

```
<div class="card text-bg-light h-100 w-60" id="cardKelembapan">
  <script>
    function updateCardClassKelembapan() {
      fetch('kelembapan.php')
        .then(response => response.text())
        .then(kelembapan => {
          const card = document.getElementById('cardKelembapan');
          if(parseFloat(kelembapan) < 40 || parseFloat(kelembapan) > 60) {
            card.classList.remove('text-bg-light');
            card.classList.add('text-bg-danger');
          } else {
            card.classList.remove('text-bg-danger');
            card.classList.add('text-bg-light');
          }
        });
    }
    setInterval(updateCardClassKelembapan, 0);
  </script>
```

Gambar 3.36 Kode Ganti Warna *Card* Kelembapan

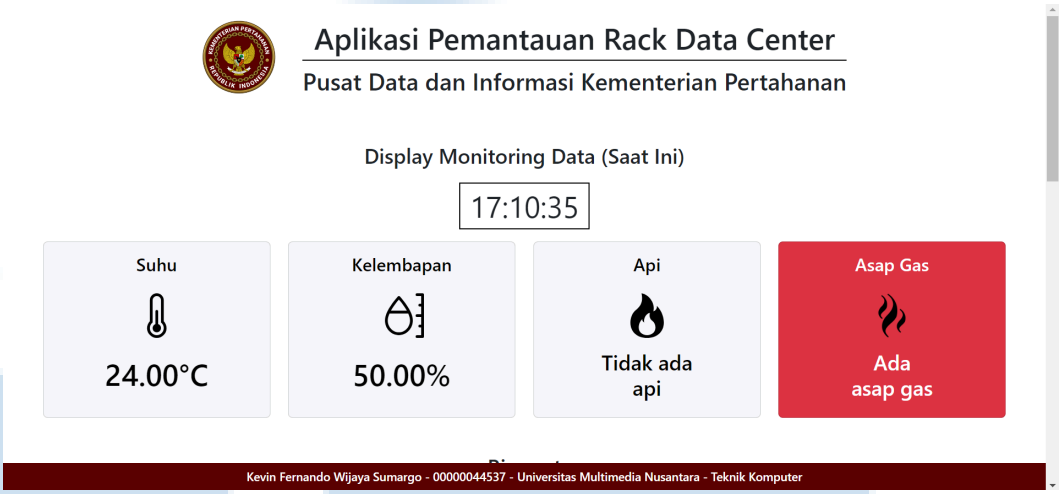


Gambar 3.37 Fitur Peringatan Terdapat Cahaya Api

Penulis memberikan fitur kotak api yang akan berubah ketika sensor mendeteksi cahaya api.

```
<div class="card text-bg-light h-100 w-60" id="cardApi">
  <script>
    function updateCardClassApi() {
      fetch('api.php')
        .then(response => response.text())
        .then(api => {
          const card = document.getElementById('cardApi');
          if(api.trim() === "Ada") {
            card.classList.remove('text-bg-light');
            card.classList.add('text-bg-danger');
          } else {
            card.classList.remove('text-bg-danger');
            card.classList.add('text-bg-light');
          }
        });
    }
    setInterval(updateCardClassApi, 0);
  </script>
```

Gambar 3.38 Kode Ganti Warna *Card* Api

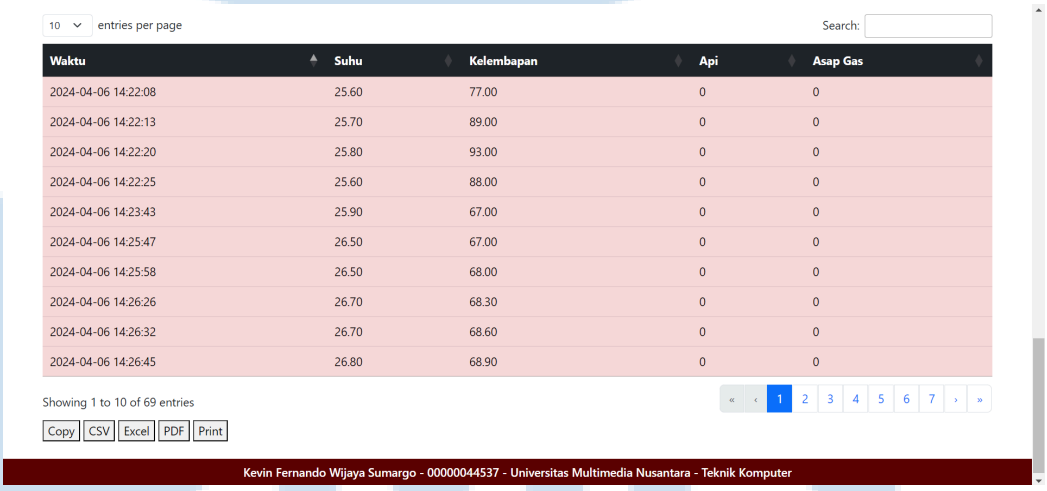


Gambar 3.39 Fitur Peringatan Terdapat Asap Gas

Penulis memberikan fitur kotak asap gas yang akan berubah ketika sensor mendeteksi asap api.

```
<div class="card text-bg-light h-100 w-60" id="cardAsap">
  <script>
    function updateCardClassAsap() {
      fetch('asap.php')
        .then(response => response.text())
        .then(asap => {
          const card = document.getElementById('cardAsap');
          if(asap.trim() === "Ada") {
            card.classList.remove('text-bg-light');
            card.classList.add('text-bg-danger');
          } else {
            card.classList.remove('text-bg-danger');
            card.classList.add('text-bg-light');
          }
        });
    }
    setInterval(updateCardClassAsap, 0);
  </script>
```

Gambar 3.40 Kode Ganti Warna *Card* Asap



Gambar 3.41 Fitur Tabel

Penulis memberikan fitur tabel yang akan berubah menjadi merah ketika data menunjukkan suhu dan kelembapan melewati batas, serta terdeteksi api dan asap gas.

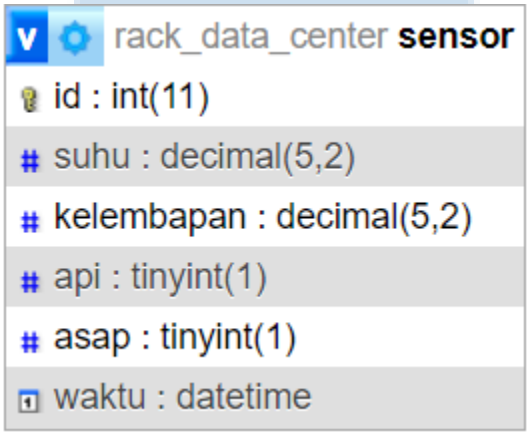
```
createdRow: function(row, data, dataIndex) {
  if(data[1] >= 25.00 || data[2] < 40 || data[2] > 60 ||
     data[3] === 1 || data[4] === 1) {
    $(row).addClass('table-danger');
  }
}
```

Gambar 3.42 Kode Ganti Warna Baris Tabel

Database dan Backend Website SiPantau

- ***Database***

Untuk *database* penulis menggunakan *database* lokal MySQL karena pada perancangan ini penulis hanya melakukan *testing*. Untuk *database* asli yang digunakan, telah dikelola oleh Pak Rizqi dan tidak dapat dipublikasikan.



Gambar 3.43 Skema *database*

Nama dari *database* yang penulis gunakan untuk menyimpan data dari sensor adalah "rack_data_center". *Database* itu hanya memiliki satu tabel, yaitu "sensor". Tabel ini berisi enam buah atribut, yaitu :

- "id" dengan tipe data *integer* sebagai *primary key* yang mengidentifikasi tiap *record*.
- "suhu" dengan tipe data desimal 5 digit dengan 2 digit di belakang koma.

- “kelembapan” dengan tipe data desimal 5 digit dengan 2 digit di belakang koma.
- “api” dengan tipe data *tinyint* yang hanya memuat 1 digit, yaitu 1 dan 0.
- “asap” dengan tipe data *tinyint* yang hanya memuat 1 digit, yaitu 1 dan 0.
- “waktu” dengan tipe data *datetime* yang mencakup tanggal dan waktu dari masuknya data ke *database* itu.

koneksi.php > ...

```php
<?php
    $konek = mysqli_connect(
        hostname: 'localhost',
        username: 'root',
        password: '',
        database: 'rack_data_center'
    );
?>
```

Gambar 3.44 Kode di “koneksi.php”

“koneksi.php” membuat koneksi ke database MySQL bernama rack_data_center menggunakan mysqli_connect().

- mysqli_connect('localhost', 'root', '', 'rack_data_center'): Fungsi ini mencoba untuk terhubung ke database MySQL.
  - 'localhost': Server database di-*hosting* di perangkat yang sama.
  - 'root': Nama pengguna untuk terhubung ke *database*.
  - '': Kata sandi untuk pengguna *database*. Dalam hal ini, kosong.
  - 'rack_data_center': Nama *database* yang akan dihubungkan.
- mysqli_connect mengembalikan objek koneksi jika berhasil, atau salah jika koneksi gagal. Objek ini disimpan dalam variabel $konek.

kirimdata.php > ...

```php
<?php
    include"koneksi.php";

    $suhu = $_GET['suhu'];
    $kelembapan = $_GET['kelembapan'];
    $api = $_GET['api'];
    $asap = $_GET['asap'];

    mysqli_query( mysql: $konek,  query: "ALTER TABLE sensor AUTO_INCREMENT=1");
    $simpan = mysqli_query( mysql: $konek,  query: "INSERT INTO sensor(suhu, kelembapan, api, asap)
                                                    VALUES('$suhu', '$kelembapan', '$api', '$asap')");

    if($simpan) {
        echo "Berhasil dikirim.";
    } else {
        echo "Gagal dikirim.";
    }
?>
```

Gambar 3.45 Kode di "kirimdata.php"

"kirimdata.php" memasukkan data sensor ke dalam tabel sensor. Program ini mengambil nilai untuk suhu, kelembapan, api, dan asap dari parameter *query* URL dan kemudian melakukan *insert*.

suhu.php > ...

```php
<?php
    include"koneksi.php";

    date_default_timezone_set( timezoneId: 'Asia/Jakarta');

    $sql = mysqli_query( mysql: $konek,  query: "SELECT * FROM sensor ORDER BY waktu DESC");

    $data = mysqli_fetch_array( result: $sql);
    $suhu = $data['suhu'];

    mysqli_close( mysql: $konek);

    echo $suhu;
?>
```

Gambar 3.46 Kode di "suhu.php"

"suhu.php" mengambil nilai suhu terbaru dari tabel sensor dan meng-*echo*-nya di *card*. Kode itu juga menetapkan zona waktu *default* ke Asia/Jakarta dan mengurutkan hasilnya berdasarkan waktu dalam urutan menurun untuk mendapatkan *entry* terbaru.

kelembapan.php > ...

```php
<?php
    include"koneksi.php";

    date_default_timezone_set( timezoneId: 'Asia/Jakarta');

    $sql = mysqli_query( mysql: $konek,  query: "SELECT * FROM sensor ORDER BY waktu DESC");

    $data = mysqli_fetch_array( result: $sql);
    $kelembapan = $data['kelembapan'];

    mysqli_close( mysql: $konek);

    echo $kelembapan;
?>
```

Gambar 3.47 Kode di "kelembapan.php"

"kelembapan.php" mengambil nilai suhu terbaru dari tabel sensor dan meng-*echo*-nya di *card*. Kode itu juga menetapkan zona waktu *default* ke Asia/Jakarta dan mengurutkan hasilnya berdasarkan waktu dalam urutan menurun untuk mendapatkan *entry* terbaru.

api.php > ...

```php
<?php
    include"koneksi.php";

    date_default_timezone_set( timezoneId: 'Asia/Jakarta');

    $sql = mysqli_query( mysql: $konek,  query: "SELECT * FROM sensor ORDER BY waktu DESC");

    $data = mysqli_fetch_array( result: $sql);
    $api = $data['api'];

    if($api == "0") {$api = "Tidak ada";}
    else if($api == "1") {$api = "Ada";}

    mysqli_close( mysql: $konek);

    echo $api;
?>
```

Gambar 3.48 Kode di "api.php"

"api.php" mem-*fetch* data api yang terakhir di *database* untuk ditampilkan di *card*. Jika nilainya adalah 0, yang ditampilkan di *card* adalah tulisan "Tidak ada". Jika 1, yang ditampilkan adalah tulisan "Ada".

asap.php > ...

```php
<?php
    include"koneksi.php";

    date_default_timezone_set( timezoneId: 'Asia/Jakarta');

    $sql = mysqli_query( mysql: $konek, query: "SELECT * FROM sensor ORDER BY waktu DESC");

    $data = mysqli_fetch_array( result: $sql);
    $asap = $data['asap'];

    if($asap == "0") {$asap = "Tidak ada";}
    else if($asap == "1") {$asap = "Ada";}

    mysqli_close( mysql: $konek);

    echo $asap;
?>
```

Gambar 3.49 Kode di "asap.php"

"asap.php" mem-*fetch* data asap yang terakhir di *database* untuk ditampilkan di *card.* Jika nilainya adalah 0, yang ditampilkan di *card* adalah tulisan "Tidak ada". Jika 1, yang ditampilkan adalah tulisan "Ada".

tabel.php > ...

```php
<?php include"koneksi.php"; ?>
<div id="table_to_chart" class="chart-display"></div>
    <table id="table" class="table">
        <thead class="table-dark">
            <tr>
                <th>Waktu</th>
                <th>Suhu</th>
                <th>Kelembapan</th>
                <th>Api</th>
                <th>Asap Gas</th>
            </tr>
        </thead>
        <tbody>
            <?php
                $sql = $konek->query( query: "SELECT * FROM sensor");
                while ($rView=$sql->fetch_array()) {
            ?>
            <tr>
                <td><?=$rView['waktu']?></td>
                <td><?=$rView['suhu']?></td>
                <td><?=$rView['kelembapan']?></td>
                <td><?=$rView['api']?></td>
                <td><?=$rView['asap']?></td>
            </tr>
            <?php
                }
            ?>
        </tbody>
    </table>
</div>
```

Gambar 3.50 Kode di "tabel.php"

"tabel.php" meng-*query*/mem-*fetch* data di *database* dan menampilkannya di tabel HTML.

server_processing.php > ...

```php
<?php

$table = 'sensor';

$primaryKey = 'id';

$columns = array(
    array('db' => 'waktu',      'dt' => 0 ),
    array('db' => 'suhu',       'dt' => 1 ),
    array('db' => 'kelembapan', 'dt' => 2 ),
    array('db' => 'api',        'dt' => 3 ),
    array('db'  => 'asap',      'dt' => 4)
);

$sql_details = array(
    'user' => 'root',
    'pass' => '',
    'db'   => 'rack_data_center',
    'host' => 'localhost'
    // ,'charset' => 'utf8' // Depending on your PHP and MySQL config, you may need this
);

require( 'ssp.class.php' );

echo json_encode(
     value: SSP::simple( request: $_GET, conn: $sql_details, $table, $primaryKey, $columns )
);
```

Gambar 3.51 Kode di "server_processing.php"

"server_processing.php" mengirimkan respons berkode JSON ke klien. Respons ini dihasilkan oleh metode SSP::simple, yang mengambil parameter berikut:

- $_GET: Parameter *query* dari permintaan HTTP klien. Ini termasuk parameter untuk mengurutkan, memfilter, dan memberi nomor halaman pada data.
- $sql_details: Detail koneksi *database*.
- $table: Nama tabel *database* yang akan digunakan.
- $primaryKey: Kunci utama tabel.
- $columns: Kolom database yang akan disertakan dalam DataTable.

3.3 Kendala yang Ditemukan

Selama masa kerja magang di Pusdatin Kemhan berlangsung, dalam melakukan perancangan dan pengembangan perangkat IoT SiPantau, penulis menemukan kendala yang dihadapi, yaitu sering terjadinya *error* pada mikrokontroler ESP32. *Error* yang sering terjadi adalah ketika mikrokontroler sudah diprogram dan berhasil, saat diterapkan dengan sensor, ESP 32 tidak berfungsi. Kendala lainnya yang ditemukan penulis adalah pengiriman data sensor ke *database* yang nantinya akan diteruskan ke halaman *website* sangatlah cepat sehingga membuat website menjadi lambat.

3.4 Solusi atas Kendala yang Ditemukan

Berdasarkan permasalahan yang ditemukan penulis selama melakukan perancangan, untuk permasalahan sering terjadinya *error* pada ESP32, penulis memberikan tombol saklar yang berfungsi sebagai tombol *on-off* sekaligus tombol *reset*. Lalu, untuk permasalahan pengiriman data dari sensor menuju *database* yang terlalu cepat, penulis berdiskusi dengan Bapak Rizqi untuk mencari solusi dan mencoba memberikan *count* yang disisipkan pada kode mikrokontroler selama 10 detik dengan tujuan agar data yang dikirim ke *database* tidak terlalu cepat, melainkan setiap 10 detik. Penulis juga memberikan *page navigation* dan *server side processing* agar data yang berada di tabel tidak ditampilkan semua melainkan setiap 10 data pada setiap laman tabelnya, karena jika tidak memakai *server side processing*, data yang ditampilkan merupakan data keseluruhan, hanya saja dengan bantuan CSS, data yang ditampilkan menjadi sedikit.