

BAB IV

PEMBUATAN ALAT

4.1 Pembuatan Perangkat Keras

4.1.1 Alat dan Bahan

Pada pelaksanaan Tugas Akhir ini digunakan beberapa alat untuk mendukung proses perakitan sistem pengharum ruangan adaptif berbasis waktu sholat dan perubahan kondisi udara ruangan. Alat yang digunakan ditunjukkan pada Tabel 4-1.

***Tabel 4- 1** Alat-alat yang Digunakan*

No	Nama Alat	Jumlah
1	Multimeter	1 Buah
2	Solder	1 Buah
3	Tang Potong	1 Buah
4	Obeng Set	1 Set
5	Gunting	1 Buah

Tabel 4-2 menunjukkan bahan-bahan utama yang digunakan dalam pembuatan sistem pengharum ruangan adaptif berbasis waktu sholat dan perubahan kondisi udara ruangan menggunakan ESP32.

***Tabel 4- 2** Bahan-bahan yang Digunakan*

No	Nama Bahan	Jumlah
1	Papan PCB	1 Lembar
2	Timah Solder	Secukupnya
3	Pengharum Ruangan Otomatis	1 Buah
4	ESP32-S2	1 Buah
5	RTC DS3231SN	1 Buah
6	Sensor MQ-135	1 Buah

No	Nama Bahan	Jumlah
7	OLED 128 x 64	1 Buah
8	Transistor S9012	1 Buah
9	MOSFET AO3401A	1 Buah
10	Baterai Lithium Polymer 801530	3 Buah
11	Boost Converter	1 Buah
12	USB Type-C TP4056	1 Buah

4.1.2 Pembuatan Perangkat Elektrik

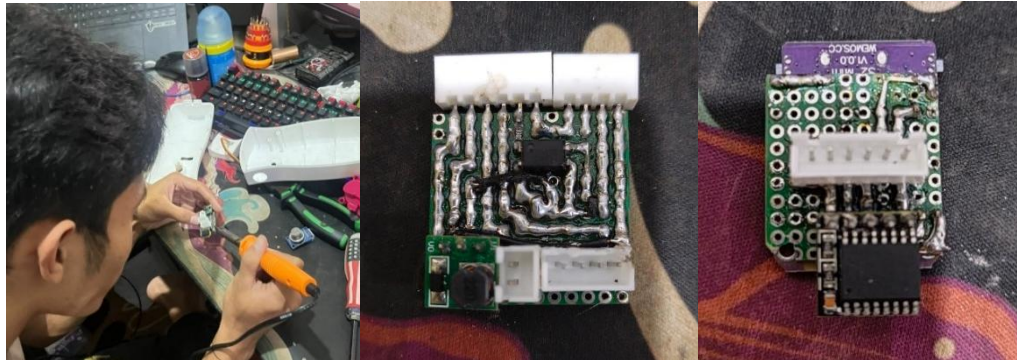
Pembuatan perangkat elektrik dilakukan untuk merealisasikan rangkaian sistem yang telah dirancang pada tahap sebelumnya. Proses ini meliputi perancangan skematik dan layout PCB, pencetakan serta penyolderan PCB, hingga integrasi seluruh rangkaian ke dalam bodi Stella Matic. Tahapan pembuatan perangkat elektrik dijelaskan sebagai berikut.

1. Perancangan skematik dan layout PCB

Perancangan skematik dan layout PCB dilakukan menggunakan aplikasi EasyEDA [20]. Pada tahap ini ditentukan hubungan antar komponen, jalur rangkaian, serta posisi komponen pada PCB agar sesuai dengan fungsi masing-masing dan ruang yang tersedia di dalam bodi Stella Matic. Setelah skematik selesai dibuat, rangkaian kemudian disusun menjadi layout PCB sebagai acuan pada proses pembuatan papan rangkaian.

Rangkaian sistem dibagi menjadi tiga bagian, yaitu Main PCB, Secondary PCB, dan Motor PCB. Main PCB digunakan sebagai bagian utama yang berisi ESP32-S2, RTC DS3231SN, dan rangkaian pendukung. Secondary PCB digunakan untuk menghubungkan sensor MQ-135, OLED 128 × 64, serta rangkaian pengendali daya, sedangkan Motor PCB digunakan untuk mengendalikan motor DC sebagai penggerak mekanisme penyemprotan. Pembagian tersebut dilakukan agar susunan komponen lebih mudah disesuaikan dengan ruang di dalam bodi pengharum ruangan.

2. Proses Pencetakan dan Penyolderan PCB



Gambar 4. 1 *Proses Pencetakan dan Penyolderan PCB*

Gambar 4.1 menunjukkan proses pencetakan dan penyolderan PCB yang dilakukan setelah tahap perancangan skematik dan *layout* selesai. Setelah PCB selesai dibuat, setiap komponen dipasang pada posisi yang telah ditentukan kemudian dilakukan proses penyolderan agar komponen terhubung dengan jalur rangkaian. Setelah seluruh komponen terpasang, dilakukan pemeriksaan terhadap hasil penyolderan dan jalur PCB untuk memastikan tidak terdapat sambungan yang terputus maupun hubungan antarjalur yang tidak diinginkan.

3. Integrasi Sistem



Gambar 4. 2 *Integrasi Sistem*

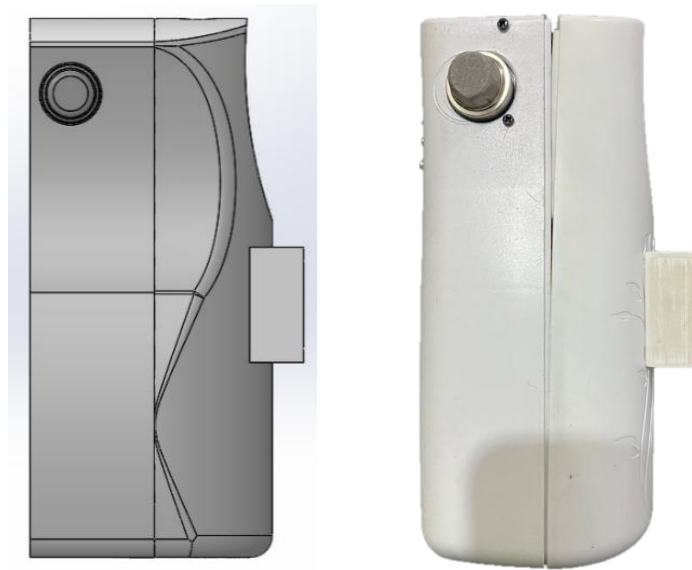
Gambar 4.2 menunjukkan proses integrasi seluruh rangkaian dan komponen ke dalam bodi asli pengharum ruangan Stella Matic. Sensor MQ-135 ditempatkan pada bagian samping alat agar bagian sensor dapat berhubungan dengan udara di sekitar ruangan, sedangkan OLED 128×64 ditempatkan pada bagian depan agar

informasi sistem dapat dilihat dengan mudah. Bagian pengisian daya ditempatkan pada posisi yang mudah dijangkau ketika baterai perlu diisi ulang. Main PCB, Secondary PCB, Motor PCB, baterai, serta jalur penghubung ditempatkan di dalam bodi dengan menyesuaikan ruang yang tersedia dan tetap memperhatikan mekanisme penyemprotan bawaan Stella Matic. Ketiga PCB dihubungkan menggunakan konektor dan kabel sesuai dengan rangkaian yang telah dibuat. Dengan susunan tersebut, seluruh bagian sistem dapat dipasang di dalam bodi Stella Matic tanpa memerlukan wadah tambahan.

4.1.3 Pembuatan Perangkat Mekanik



Gambar 4. 3 Tampak Depan



Gambar 4. 4 Tampak Samping

Berdasarkan Gambar 4.3 dan Gambar 4.4, hasil pembuatan perangkat mekanik ditunjukkan melalui tampak depan dan tampak samping alat yang telah terintegrasi dengan seluruh komponen. Bentuk utama alat tetap dipertahankan, kemudian dilakukan beberapa penyesuaian pada bagian bodi berdasarkan desain 3D yang telah dibuat. OLED ditempatkan pada bagian depan agar informasi sistem dapat dilihat dengan mudah, sedangkan sensor MQ-135 ditempatkan pada bagian samping agar bagian sensornya dapat berhubungan dengan udara di sekitar ruangan. Penyesuaian juga dilakukan pada bagian lubang dan ruang di dalam bodi agar PCB, baterai, jalur kabel, serta mekanisme penyemprotan bawaan dapat terpasang tanpa saling mengganggu.

4.2 Pembuatan Perangkat Lunak

Pembuatan perangkat lunak dilakukan menggunakan Arduino IDE [21] untuk menerapkan alur kerja sistem pada ESP32-S2 sesuai dengan rancangan yang telah dibuat. Program digunakan untuk mengatur pembacaan waktu dari RTC DS3231SN, pengambilan jadwal sholat melalui koneksi internet, pembacaan sensor MQ-135, proses penyemprotan, tampilan OLED, pengiriman data ke Google Spreadsheet, serta pengaturan mode *deep sleep*. Perangkat lunak dirancang agar

sistem dapat bekerja secara otomatis berdasarkan jadwal sholat yang telah tersimpan. Pada setiap waktu sholat, sistem memasuki waktu operasi selama 15 menit untuk melakukan pembacaan MQ-135 dan menentukan kondisi penyemprotan berdasarkan nilai ambang batas serta kondisi *cooldown*. Setelah waktu operasi dan proses pengiriman data selesai, ESP32-S2 kembali masuk ke mode *deep sleep* hingga waktu sholat berikutnya.

4.2.1 Inisialisasi Sistem

```
// 1. POWER UP I2C BUS (OLED & RTC)
pinMode(DsPw, OUTPUT);
digitalWrite(DsPw, HIGH);
delay(50);
Wire.begin(SDA, SCL);
// Inisialisasi OLED
if (!display.begin(SSD1306_SWITCHCAPVCC, OLED_ADDR)) {
  Serial.println("OLED failed");
} else {
  display.clearDisplay();
  display.display();
}
// 2. INITIALIZE PINS
pinMode(MQ, INPUT);
pinMode(USB_DETECT_PIN, INPUT_PULLDOWN);

pinMode(MqPw, OUTPUT);
digitalWrite(MqPw, HIGH); // MQ-135 tetap ON terus
pinMode(MTR, OUTPUT);
digitalWrite(MTR, HIGH); // MOSFET OFF (Active LOW)
if (!rtc.begin()) {
  Serial.println("Couldn't find RTC");
}
```

Gambar 4. 5 Inisialisasi Sistem

Gambar 4.5 menunjukkan bagian program yang digunakan untuk melakukan proses inisialisasi sistem pada ESP32-S2. Komponen yang diinisialisasi meliputi RTC DS3231SN, OLED 128×64 , sensor MQ-135, motor penyemprot, serta jalur komunikasi yang digunakan. Komunikasi RTC Pembacaan data waktu dari RTC DS3231SN pada program menggunakan library RTCLib [22] dan OLED menggunakan I2C dengan SDA pada GPIO 39 dan SCL pada GPIO 40. RTC DS3231SN digunakan sebagai sumber waktu utama, sedangkan motor penyemprot ditempatkan pada kondisi awal OFF karena rangkaian pengendali motor menggunakan logika *active low*. Sensor MQ-135 diatur agar tetap mendapatkan daya selama sistem digunakan, termasuk ketika ESP32-S2 berada dalam mode *deep sleep*. Hal tersebut dilakukan agar sensor tetap berada dalam kondisi aktif dan dapat langsung digunakan ketika sistem memasuki waktu operasi. Setelah seluruh komponen selesai diinisialisasi, ESP32-S2 membaca waktu dari RTC dan memeriksa data jadwal sholat yang digunakan sebagai acuan.

4.2.2 Konfigurasi Pengaturan Waktu dan Jadwal Sholat

```
// Mulai tepat pada HH:MM:00
if (now.hour() == prayer_times[i][0] &&
    now.minute() == prayer_times[i][1] &&
    now.second() == 0) {

    // Supaya satu waktu sholat
    // hanya dijalankan satu kali
    if (lastPrayerDay != now.day() ||
        lastPrayerIndex != i) {

        currentPrayerIndex = i;

        lastPrayerDay = now.day();
        lastPrayerIndex = i;

        runActiveWindow();
    }

    bool fetchPrayerTimes() {
        WiFiClientSecure secureClient;
        secureClient.setInsecure();
        HTTPClient http;

        String url = "https://api.aladhan.com/v1/timingsByCity?city=" + String(CITY) +
            "&country=" + String(COUNTRY) + "&method=" + String(CALC_METHOD);
        http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
        http.begin(secureClient, url);
        int httpCode = http.GET();
        if (httpCode == 200 || httpCode == 201) {
            String payload = http.getString();
            DynamicJsonDocument doc(4096);
            DeserializationError error = deserializeJson(doc, payload);

            if (!error) {
                JsonObject timings = doc["data"]["timings"];
                parsePrayerString(timings["Fajr"].as<String>(), 0);
                parsePrayerString(timings["Dhuhr"].as<String>(), 1);
                parsePrayerString(timings["Asr"].as<String>(), 2);
                parsePrayerString(timings["Maghrib"].as<String>(), 3);
                parsePrayerString(timings["Isha"].as<String>(), 4);
            }
        }
    }
}
```

Gambar 4. 6 Konfigurasi Pengaturan Waktu dan Jadwal Sholat

Gambar 4.6 menunjukkan konfigurasi program pengaturan waktu RTC dan jadwal sholat yang digunakan sebagai acuan waktu operasi sistem. Jadwal lima waktu sholat, yaitu Subuh, Dzuhur, Ashar, Maghrib, dan Isya', diperoleh melalui koneksi internet menggunakan API AlAdhan [24]. Data jadwal yang diterima kemudian diproses oleh ESP32-S2 dan disimpan pada variabel `prayer_times`. Pembaruan jadwal dilakukan setiap hari sehingga waktu operasi sistem dapat mengikuti jadwal sholat pada tanggal tersebut. Setelah jadwal tersimpan, waktu aktual dari RTC dibandingkan dengan masing-masing jadwal sholat. Ketika waktu RTC telah sesuai dengan jadwal yang tersimpan, ESP32-S2 menjalankan fungsi `runActiveWindow()` dan sistem memasuki waktu operasi selama 15 menit. Selama periode tersebut, sistem melakukan pembacaan MQ-135, menentukan kondisi penyemprotan, serta menampilkan kondisi kerja alat pada OLED. Setelah waktu operasi selesai, data hasil kerja sistem dikirim ke Google *Spreadsheet* sebelum ESP32-S2 kembali ke mode *deep sleep*.

4.2.3 Konfigurasi Pembacaan MQ-135 dan Sistem Penyemprotan

```

if (currentMinute < 15 &&
    nowMs - lastSampleTime >= 1000) {
    lastSampleTime += 1000;
    mqr = analogRead(MQ);
    minuteSum += mqr;

    sampleCount++;
}

// =====
// SETIAP 1 MENIT
// =====

if (currentMinute < 15 &&
    nowMs - minuteStart >= 60000) {

    float minuteAverage = 0;

    // Hitung rata-rata 1 menit
    if (sampleCount > 0) {
        minuteAverage =
            minuteSum / sampleCount;
    }

    // Simpan rata-rata ke data Spreadsheet
    mqReadings[currentMinute] =
        minuteAverage;

    currentMinute++;

    if (minuteAverage >= mqSet &&
        !Sprayer &&
        (nowMs - SpryTmrInv >= SpryInv)) {

        Sprayer = true;

        sprayCount++;

        SpryTmr = millis();

        digitalWrite(MTR, LOW);
    }

    // =====
    // RESET DATA UNTUK MENIT BERIKUTNYA
    // =====

    minuteSum = 0;

    sampleCount = 0;

    minuteStart += 60000;
}

```

Gambar 4. 7 Konfigurasi Pembacaan MQ-135 dan Sistem Penyemprotan

Gambar 4.7 menunjukkan konfigurasi program pembacaan sensor MQ-135 dan proses penyemprotan pengharum ruangan. Selama waktu operasi berlangsung, sensor MQ-135 melakukan pembacaan menggunakan fungsi `analogRead()` dengan interval setiap 1 detik. Setiap hasil pembacaan sensor dijumlahkan pada variabel `minuteSum`, sedangkan jumlah data yang terbaca dihitung menggunakan variabel `sampleCount`. Setelah proses pembacaan berlangsung selama 1 menit, sistem menghitung nilai rata-rata dari seluruh data yang diperoleh pada periode tersebut. Nilai rata-rata ini digunakan agar data yang dipakai sebagai dasar pengambilan keputusan tidak hanya berasal dari satu kali pembacaan sesaat, tetapi mewakili kondisi sensor selama satu menit. Hasil perhitungan kemudian disimpan pada array `mqReadings` sebagai data pembacaan untuk setiap menit. Proses tersebut dilakukan

secara berulang selama waktu operasi 15 menit, sehingga pada setiap waktu sholat diperoleh 15 nilai rata-rata MQ-135 yang selanjutnya digunakan untuk proses analisis dan penentuan kondisi penyemprotan.

Nilai rata-rata MQ-135 setiap 1 menit kemudian dibandingkan dengan nilai ambang batas sebesar 186 ADC. Nilai di bawah 186 ADC dinyatakan tidak mencapai nilai ambang, sedangkan nilai yang mencapai atau melebihi 186 ADC dinyatakan mencapai nilai ambang. Apabila nilai sensor mencapai nilai ambang dan sistem tidak berada dalam kondisi *cooldown*, motor penyemprot diaktifkan selama 2 detik. Setelah penyemprotan selesai, motor kembali dimatikan dan sistem menerapkan *cooldown* selama 5 menit. Selama *cooldown* berlangsung, pembacaan sensor tetap dilakukan, tetapi penyemprotan berikutnya tidak dapat dilakukan sampai waktu *cooldown* selesai. Mekanisme tersebut digunakan untuk mencegah motor melakukan penyemprotan secara berulang dalam waktu yang terlalu dekat.

4.2.4 Konfigurasi Tampilan OLED

```
// =====
// WAKTU RTC
// =====
char timeBuffer[9];

sprintf(
    timeBuffer,
    "%02d:%02d:%02d",
    now.hour(),
    now.minute(),
    now.second()
);

printCentered(
    String(timeBuffer),
    11,
    2
);

// =====
// MQ-135
// =====
display.setTextSize(1);

display.setCursor(0, 35);
display.print("MQ-135 : ");
```

Gambar 4. 8 Konfigurasi Tampilan OLED

Gambar 4.8 menunjukkan konfigurasi program yang digunakan untuk menampilkan informasi kondisi kerja sistem. OLED digunakan sebagai media untuk menampilkan informasi kondisi kerja sistem secara langsung. Pengendalian tampilan OLED pada program menggunakan library Adafruit SSD1306 [23]. Informasi yang ditampilkan meliputi jenis sholat, waktu dari RTC, nilai pembacaan MQ-135, serta status kerja sistem. Status yang ditampilkan terdiri dari *Standby*,

Spray, dan *Cooldown*. Tampilan tersebut digunakan agar kondisi alat dapat diketahui tanpa harus melihat data melalui *Google Spreadsheet*. Status *Standby* ditampilkan ketika sistem berada dalam waktu operasi tetapi motor tidak sedang melakukan penyemprotan dan *cooldown* tidak aktif. Ketika syarat penyemprotan terpenuhi dan motor diaktifkan, OLED menampilkan status *Spray*. Setelah penyemprotan selesai, tampilan berubah menjadi *Cooldown* dan menunjukkan waktu jeda sebelum penyemprotan berikutnya dapat dilakukan. OLED hanya digunakan sebagai media tampilan dan tidak digunakan sebagai dasar dalam menentukan keputusan penyemprotan.

4.2.5 Konfigurasi Mode *Deep Sleep*

```
// Motor tetap HIGH = OFF
gpio_hold_en(
  (gpio_num_t)MTR
);

// OLED tetap LOW = OFF
gpio_hold_en(
  (gpio_num_t)DsPw
);

gpio_deep_sleep_hold_en();

// =====
// SET TIMER WAKE-UP
// =====
esp_sleep_enable_timer_wakeup(
  (uint64_t)nextEventSeconds *
  1000000ULL
);

// =====
// MASUK DEEP SLEEP
// =====
esp_deep_sleep_start();
```

Gambar 4. 9 Konfigurasi Mode *Deep Sleep*

Gambar 4.9 menunjukkan konfigurasi program mode *deep sleep* yang digunakan ketika ESP32-S2 berada di luar waktu operasi. Setelah waktu operasi selama 15 menit dan proses pengiriman data selesai, program mencari waktu sholat berikutnya berdasarkan jadwal yang telah tersimpan. Selisih antara waktu RTC saat itu dengan waktu sholat berikutnya digunakan untuk menentukan durasi *deep sleep*. Dengan mekanisme tersebut, ESP32-S2 dapat kembali aktif tepat pada waktu sholat berikutnya, bukan beberapa saat sebelum waktu operasi. Sebelum memasuki *deep sleep*, motor penyemprot dipastikan berada dalam kondisi OFF dan OLED dinonaktifkan. Sensor MQ-135 tetap mendapatkan daya selama ESP32-S2 berada dalam *deep sleep*, dengan kondisi pin dipertahankan menggunakan fitur GPIO *hold*.

Jika seluruh waktu sholat pada hari tersebut telah selesai, ESP32-S2 berada dalam mode *deep sleep* hingga pergantian hari untuk melakukan pembaruan jadwal berikutnya. Dengan demikian, mode *deep sleep* digunakan untuk mengurangi aktivitas ESP32-S2 di luar waktu operasi.

4.2.6 Pengiriman Data ke Google *Spreadsheet*

```
String pName = "Unknown";
if (currentPrayerIndex == 0) pName = "Subuh";
else if (currentPrayerIndex == 1) pName = "Zuhur";
else if (currentPrayerIndex == 2) pName = "Ashar";
else if (currentPrayerIndex == 3) pName = "Maghrib";
else if (currentPrayerIndex == 4) pName = "Isya";
char pTime[6];
if (currentPrayerIndex >= 0 && currentPrayerIndex <= 4) {
    sprintf(pTime, "%02d:%02d", prayer_times[currentPrayerIndex][0],
    prayer_times[currentPrayerIndex][1]);
} else {
    strcpy(pTime, "00:00");
}
String url = String(gscriptUrl) + "?type=log";
url += "&prayerTime=" + String(pTime) + "&prayerName=" + pName;
url += "&sprayCount=" + String(sprayCount) + "&battery=" + String(batPercentage);

for (int i = 0; i < 15; i++) {
    url += "&m" + String(i + 1) + "=" + String(mqReadings[i], 0);
}
http.begin(secureClient, url);
http.GET();
http.end();
```

Gambar 4. 10 Pengiriman Data ke Google *Spreadsheet*

Gambar 4.10 menunjukkan bagian program yang digunakan untuk mengirimkan data hasil kerja sistem. Google *Spreadsheet* digunakan sebagai media penyimpanan dan pemantauan data hasil kerja sistem. Data yang dikirim terdiri dari data jadwal harian dan data hasil pengujian pada setiap waktu sholat. Data jadwal meliputi tanggal, waktu pengiriman, serta jadwal lima waktu sholat yang diperoleh melalui koneksi internet. Sementara itu, setelah waktu operasi selama 15 menit selesai, ESP32-S2 mengaktifkan koneksi WiFi untuk mengirimkan nama dan waktu sholat, 15 nilai rata-rata pembacaan MQ-135, jumlah penyemprotan, serta persentase baterai melalui Google Apps Script [25]. Nilai MQ-135 dikirim dalam bentuk bilangan bulat tanpa angka di belakang koma agar data lebih sederhana dan mudah digunakan dalam proses analisis. Google *Spreadsheet* hanya digunakan sebagai media pencatatan dan pemantauan hasil kerja alat, sedangkan keputusan penyemprotan tetap dilakukan secara langsung oleh ESP32-S2 berdasarkan waktu operasi, nilai MQ-135, dan kondisi *cooldown*. Setelah proses pengiriman selesai, koneksi WiFi dinonaktifkan dan ESP32-S2 melanjutkan proses menuju mode *deep sleep* hingga waktu sholat berikutnya.