BAB IV

PROSES PEMBUATAN ALAT

4.1 Pembuatan Prototipe

Dalam proses perancangan dan implementasi proyek akhir ini, diperlukan suatu metode penelitian yang sistematis guna memastikan konsep yang matang serta struktur yang jelas dalam pengembangan sistem. Pada bab 4 menguraikan tahapan perancangan dan pembuatan perangkat, mencakup seluruh proses mulai dari perancangan awal, pemilihan komponen, hingga implementasi dan pengujian agar perangkat dapatt beroperasii sesuai dengan spesifiikasi yang telah ditentukan. Setiap tahapan dalam pengembangan sistem memiliki keterkaitan erat dalam membentuk solusi yang terintegrasi. Secara umum, prosedur pembuatan perangkat dalam tugas akhir ini terbagi menjadi dua tahapan utama, yaitu:

a. Pembuatan Perangkat Keras

Perangkat keras yanng digunakan dalam sistem ini meliputi seluruh komponen fisik yang berperan dalam proses transfer panas dan monitoring suhu. Tahapan pembuatan perangkat keras terdiri dari perancangan sistem sketsa, pemilihan material, proses perakitan, dan pengujian awal. Pembuatan perangkat keras melalui proses identifikasi dan penyediaan kebutuhan perangkat keras menjadi proses awal untuk memastikan bahwa desain alat dapat diimplementasikan secara optimal. Selanjutnya pembuatan perangkat keras, yang bertujuan untuk merealisasikan rancangan menjadi suatu sistem yang siap beroperasi.

b. Pembuatan Perangkat Lunak

Tahap selanjutnya adalah perangkat lunak dalam sistem ini berfungsi untuk membaca data dari sensor suhu termokopel, mengolah data tersebut, menampilkannya ke layar LCD dan mengirim data suhu yang terkumpul ke antarmuka WEB sederhana.

Langkah awal dalam menyelesaikan tugas akhir adalah dengan menyusun perencanaan yang matang serta merumuskan konsep yang jelas terkait perangkat atau produk yang akan dikembangkan. Tahapan ini sangat penting karena akan

menjadi dasar bagi keseluruhan proses, mulai dari analisis kebutuhan hingga implementasi. Pengembangan perangkat lunak dalam sistem ini melibatkan pemrograman menggunakan Arduino IDE dan mengembangkan antarmuka yang memungkinkan penulis untuk memantau suhu secara real-time.

4.2 Alat dan Bahan

Dalam menjalankan proyek tugas akhir, ada berbagai kebutuhan utama dan pendukung yang harus dipenuhi agar proses pembuatan tugas akhir berjalan optimal. Identifikasi kebutuhan ini menjadi langkah krusial karena akan menghasilkan daftar komprehensif mengenai komponen dan sumber daya yang diperlukan.

Tabel 4.1 memperlihatkan daftar kebutuhan tersebut, berperan sebagai landasan dalam perancangan dan pembuatan proyek tugas akhir ini, memastikan sistem yang dikembangkan sesuai dengan spesifikasi teknis serta tujuan penelitian. Dengan identifikasi yang jelas, risiko kesalahan dalam pemilihan komponen atau pengalokasian sumber daya dapat diminimalkan, sehingga proyek dapat berjalan lebih efisien dan sesuai dengan rencana.

Tabel 4. 1 Bahan pembuat tugas akhir

No	Nama Bahan	Jumlah
1	Panel surya	1 buah
2	Termokopel tipe k	1 buah
3	ESP 32 devkit V1	1 buah
4	LCD 12c 16x2	1 buah
5	Stepdown mini-360	1 buah
6	PCB fiber	1 buah
7	Kabel JST XH 2.54mm (4 pin)	1 buah
8	Box akrilik	1 buah
9	Pipa tembaga	1 buah
10	Kotak aluminium	1 buah
11	Timah solder	1 buah
12	Blower	1 buah

13	Pasir silika	15 kg
14	Heather 240 V	1 buah
15	Breaker	1 buah

Tabel 4. 2 Alat pembuat tugas akhir

NO	Nama alat	Jumlah
1	Multimeter	1 buah
2	Solder	1 buah
3	Bor	1 buah
4	Gunting	1 buah
5	Cutter	1 buah
6	Alat tulis	1 buah

4.3 Pembuatan Perangkat keras

Proses pembuatan perangkat keras diawali dengan pengumpulan literatur dan referensi dari sumber-sumber terpercaya, seperti buku, jurnal ilmiah, serta studi terkait produk serupa yang telah dikembangkan sebelumnya. Tahap ini dilanjutkan dengan analisis mendalam untuk memahami prinsip kerja sistem, mengidentifikasi kebutuhan komponen seperti jenis sensor, mikrokontroler, modul komunikasi, dan sumber daya listrik yang diperlukan. Selanjutnya, dilakukan perancangan rangkaian elektronik dengan mempertimbangkan koneksi antar komponen secara optimal. Proses ini juga melibatkan pemilihan board mikrokontroler yang sesuai dengan kebutuhan aplikasi, serta modul pendukung seperti sensor dan aktuator yang akan digunakan. Setelah perancangan selesai, tahap berikutnya adalah perakitan perangkat keras dengan menggunakan *breadboard* atau PCB, pengujian koneksi antar komponen, dan verifikasi fungsi dasar perangkat agar dapat berjalan sesuai dengan spesifikasi yang diinginkan.

4.3.1 Perancangan Mekanik

Perancangan mekanik merupakan tahap awal dalam pembuatan sistem, yang berfokus pada penyusunan struktur fisik alat agar dapat bekerja secara efisien dan aman. Pada proyek tugas akhir ini, aliran udara panas dari media pasir silika diarahkan ke kotak penyimpanan panas menggunakan bantuan blower sebagai pendorong udara panas.

Gambar 4. 1Rancang mekanik

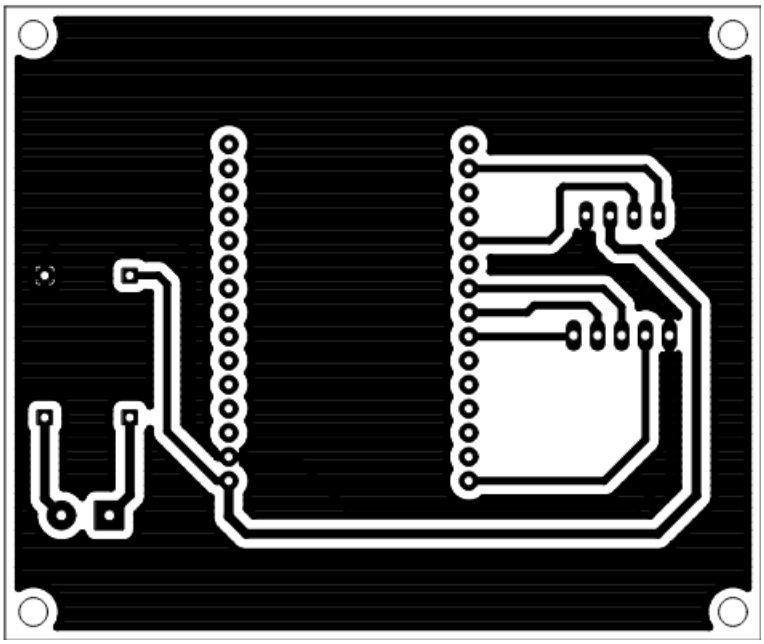
Pada penyusunan tugas akhir ini tabung media pasir silika berbentuk silinder bahan *stainless steel* dengan diameter 30 cm dan tinggi 30 cm. Berdasarkan dimensi perancangan, tabung media pasir silika memiliki bentuk silinder dengan diameter 30 cm dan tinggi 30 cm. Dengan menggunakan rumus volume tabung ($V=\pi r2t$), di mana jari-jari (r) adalah 15 cm, volume geometris tabung dihitung sebesar 21.195 cm^3. Volume ini setara dengan sekitar 21.1 Liter. Tabung ini diisi pasir silika seberat 15 kg, yang berfungsi sebagai material penyimpan energi panas karena sifat kapasitas panas spesifiknya yang baik. Di dalam tabung dipasang spiral pipa tembaga berdiameter 0,25 inch yang melilit secara merata untuk memaksimalkan perpindahan panas dari pasir silika ke udara dalam pipa.Di dalam tabung ini juga terdapat Spiral Pipa Tembaga yang dililitkan secara melingkar.

Sementara itu, kotak penyimpanan panas bahan *stainless steel* berbentuk

kubus berukuran 30 × 30 × 30 cm dengan volume sebesar 27.000 cm^3 atau 27 liter. Bagian dalam kotak juga dilengkapi dengan spiral pipa tembagadengan ukuran 0.25 inch, yang terhubung langsung dengan spiral pipa dalam tabung melalui saluran pipa tembaga dan dialiri udara panas menggunakan blower. Blower berperan dalam mentransfer udara panas dari tabung media pasir silika ke kotak penyimpanan, sehingga memungkinkan panas yang diserap media pasir silika untuk dilepaskan dan disimpan di dalam kotak. Udara yang telah dipanaskan oleh media pasir kemudian dialirkan menuju spiral pipa tembaga di dalam kotak penyimpanan, sehingga terjadi transfer dan penyimpanan energi panas

4.3.2 Pembuatan PCB

Setelah proses pembuatan perangkat keras dan perancangan alat selesai, langkah berikutnya adalah melakukan perancangan PCB (*Printed Circuit Board*) sebagai media untuk mengintegrasikan semua komponen elektronik secara rapi dan efisien. Perancangan PCB bertujuan untuk menyatukan seluruh rangkaian elektronik dalam satu papan sirkuit yang rapi dan terorganisasi, sehingga memudahkan dalam proses instalasi, perawatan, dan pengujian sistem. Dengan perancangan PCB yang tepat, perangkat keras dan alat yang telah dirancang dapat diwujudkan dalam bentuk produk akhir yang memiliki kualitas koneksi listrik optimal serta kemudahan dalam perakitan dan pemeliharaan, serta mengurangi kemungkinan kesalahan koneksi yang dapat terjadi pada sistem berbasis breadboard.



Gambar 4. 2 Rancangan PCB

4.4 Pembuatan Perangkat Lunak

Pada tahap ini dilakukan pembuatan perangkat lunak pada sistem ini bertujuan untuk proses monitoring suhu secara otomatis dan real-time dari kotak penyimpanan panas. Proyek ini adalah implementasi Data Logger Suhu real-time menggunakan mikrokontroler ESP32 dengan sensor termokopel tipe K melalui modul *MAX*6675 dan tampilan LCD 16x2 I2C**.** Data suhu yang dikumpulkan akan secara berkala dikirim dan disimpan ke database Google *Firebase Firestore* melalui koneksi WiFi.Dilengkapi dengan sebuah aplikasi web sederhana untuk memantau data secara *real-time* dan kemampuan untuk mengunduh seluruh data dalam format Excel. Dalam pengembangannya, perangkat lunak ini dirancang menggunakan bahasa pemrograman pada platform Arduino IDE.

```
#include <Arduino.h>
#include <WiFi.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <time.h>

// Firebase Library
#include <Firebase_ESP_Client.h>
#include <addons/TokenHelper.h>

// Hardware Peripheral Libraries
#include <LiquidCrystal_I2C.h>
#include "max6675.h"

// Project Configuration
#include "secrets.h"

// ===================================
// GLOBAL OBJECTS & CONSTANTS
// ===================================

FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", 7 * 3600); // UTC+7 (WIB) offset

// MAX6675 Pin Definitions (CLK, CS, DO)
const int MAX6675_SCK_PIN = 18;
const int MAX6675_CS_PIN = 5;
const int MAX6675_MISO_PIN = 17;

MAX6675 thermocouple(MAX6675_SCK_PIN, MAX6675_CS_PIN, MAX6675_MISO_PIN);

// LCD 16x2 I2C Object (Address, Columns, Rows)
```

Gambar 4. 3 Program monitoring suhu

```
LiquidCrystal_I2C lcd(0x27, 16, 2);

struct DateTimeComponents
{
    int year;
    int month;
    int day;
    int hour;
    int minute;
    int second;
    String iso8601;
};

// Firebase data send interval
unsigned long lastFirebaseSendMillis = 0;
const unsigned long FIREBASE_SEND_INTERVAL_MS = 60000; // 1 minute

// NTP Sync status
bool isTimeSynced = false;
unsigned long lastNtpAttemptMillis = 0;
const unsigned long NTP_RETRY_INTERVAL_MS = 5000; // Retry NTP every 5 seconds if not synced

// ===================================
// HELPER FUNCTIONS
// ===================================

void printLcd(const char *line1, const char *line2 /*= ""*/)
{
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(line1);
    if (line2[0] != '\0')
    {
        lcd.setCursor(0, 1);
        lcd.print(line2);
    }
```

Gambar 4. 4 Program monitoring suhu

```
}

void connectToWiFi()
{
    printLcd("Connecting WiFi...", WIFI_SSID);
    Serial.print("Connecting to WiFi: ");
    Serial.println(WIFI_SSID);
    WiFi.mode(WIFI_STA);
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
}

void WiFiEvent(WiFiEvent_t event)
{
    Serial.printf("[WiFi-event] event: %d\n", event);
    switch (event)
    {
    case ARDUINO_EVENT_WIFI_STA_GOT_IP:
        Serial.println("WiFi connected, IP address:");
        Serial.println(WiFi.localIP());
        isTimeSynced = false;
        lastNtpAttemptMillis = 0;
        break;
    case ARDUINO_EVENT_WIFI_STA_DISCONNECTED:
        Serial.println("WiFi lost connection.");
        isTimeSynced = false;
        break;
    default:
        break;
    }
}

DateTimeComponents getLocalTimeComponents()
{
    DateTimeComponents dt;
    dt.year = 0;
    dt.month = 0;
```

Gambar 4. 5 Program monitoring suhu

```
    dt.day = 0;
    dt.hour = 0;
    dt.minute = 0;
    dt.second = 0;
    dt.iso8601 = "Invalid Time";

    if (!isTimeSynced && (millis() - lastNtpAttemptMillis >= NTP_RETRY_INTERVAL_MS))
    {
        timeClient.update();
        lastNtpAttemptMillis = millis();
        Serial.println("NTP update attempt initiated...");
    }

    time_t epochTime = timeClient.getEpochTime();
    struct tm *timeinfo = localtime(&epochTime);

    if (timeinfo == nullptr || epochTime < 100000)
    {
        Serial.println("Warning: Current time is invalid. NTP not synced or conversion failed.
        isTimeSynced = false;
        return dt;
    }

    isTimeSynced = true;

    dt.year = timeinfo->tm_year + 1900;
    dt.month = timeinfo->tm_mon + 1;
    dt.day = timeinfo->tm_mday;
    dt.hour = timeinfo->tm_hour;
    dt.minute = timeinfo->tm_min;
    dt.second = timeinfo->tm_sec;

    char dateTimeStr[30];
    sprintf(dateTimeStr, "%04d-%02d-%02dT%02d:%02d:%02d",
            dt.year, dt.month, dt.day,
            dt.hour, dt.minute, dt.second);
```

Gambar 4. 6 Program monitoring suhu

```
    dt.iso8601 = String(dateTimeStr);

    return dt;
}

// ================================
// SETUP FUNCTION
// ================================

void setup()
{
    Serial.begin(115200);
    Serial.println("\n--- Starting ESP32 Data Logger ---");

    // Initialize LCD
    lcd.init();
    lcd.backlight();
    printLcd("Initializing...", "");
    delay(1000);

    // WiFi Connection
    WiFi.setHostname(WIFI_HOSTNAME);
    WiFi.onEvent(WiFiEvent);
    connectToWiFi();

    printLcd("Connecting WiFi...", "Waiting for IP...");
    Serial.print("Waiting for WiFi connection...");
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
        lcd.setCursor(15, 1);
        lcd.print(".");
    }
    String ipAddr = WiFi.localIP().toString();
    Serial.println("\nWiFi Connected. IP: " + ipAddr);
```

Gambar 4. 7 Program monitoring suhu

```
    printLcd("WiFi Connected!", ipAddr.c_str());
    delay(2000);

    // Initial NTP Sync (Blocking in Setup for initial Firebase auth)
    printLcd("Syncing Time...", "Blocking once...");
    timeClient.begin();
    Serial.println("Initial NTP sync (blocking in setup)...");
    unsigned long startTime = millis();
    while (!timeClient.update() && (millis() - startTime < 15000))
    { // Max 15s blocking
        timeClient.forceUpdate();
        Serial.print(".");
        lcd.setCursor(15, 1);
        lcd.print(".");
        delay(500);
    }
    if (timeClient.update())
    {
        isTimeSynced = true;
        String formattedTime = timeClient.getFormattedTime();
        Serial.println("\nInitial NTP time synced: " + formattedTime);
        printLcd("Time Synced!", formattedTime.c_str());
        delay(2000);
    }
    else
    {
        isTimeSynced = false;
        Serial.println("\nInitial NTP sync failed. Firebase connections might fail.");
        printLcd("NTP Sync Failed!", "Continuing...");
        delay(2000);
    }
    lastNtpAttemptMillis = millis();

    // Firebase Initialization
    printLcd("Connecting Firebase", "Auth...");
    config.api_key = API_KEY;
```

Gambar 4. 8 Program monitoring suhu

```
217     auth.user.email = USER_EMAIL;
218     auth.user.password = USER_PASSWORD;
219     config.token_status_callback = tokenStatusCallback;
220
221     Firebase.begin(&config, &auth);
222     Firebase.reconnectNetwork(true);
223
224     Serial.print("Waiting for Firebase to be ready...");
225     while (!Firebase.ready())
226     {
227         delay(500);
228         Serial.print(".");
229         lcd.setCursor(15, 1);
230         lcd.print(".");
231     }
232     Serial.println("\nFirebase is ready and authenticated.");
233     printLcd("Firebase Ready!", "Authenticated.");
234     delay(2000);
235
236     // Firebase data buffer configuration
237     fbdo.setBSSLBufferSize(4096, 1024);
238     fbdo.setResponseSize(2048);
239
240     // Initialize MAX6675 Sensor
241     Serial.println("Initializing MAX6675...");
242     delay(500); // MAX6675 stabilization time
243
244     double initialTemp = thermocouple.readCelsius();
245     if (isnan(initialTemp))
246     {
247         Serial.println("ERROR: Failed to read from MAX6675 sensor! Check wiring.");
248         printLcd("MAX6675 Error!", "Check wiring.");
249         while (1)
250         {
251             delay(1000);
252         }
```

Gambar 4. 9 Program monitoring suhu

```
253     }
254     else
255     {
256         Serial.print("MAX6675 OK. Initial Temp: ");
257         Serial.print(initialTemp);
258         Serial.println(" C");
259         char tempStr[17];
260         sprintf(tempStr, "Temp: %.2f C", initialTemp);
261         printLcd("MAX6675 OK!", tempStr);
262         delay(2000);
263     }
264
265     Serial.println("\n--- System Ready! Entering Loop ---");
266     printLcd("System Ready!", "Temp Logger");
267     delay(2000);
268 }
269
270 // ====================================
271 // LOOP FUNCTION
272 // ====================================
273
274 void loop()
275 {
276     // Check main connections (WiFi & Firebase)
277     if (WiFi.isConnected() && Firebase.ready())
278     {
279         // Update NTP time non-blockingly and check sync status
280         DateTimeComponents now = getLocalTimeComponents();
281
282         // Only proceed if NTP time is synced
283         if (isTimeSynced)
284         {
285             // Read Temperature
286             double temperature = thermocouple.readCelsius();
287
288             if (isnan(temperature))
```

Gambar 4. 10 Program monitoring suhu

```
289             {
290                 Serial.println("Failed to read temperature from MAX6675!");
291                 printLcd("Read Error!", "MAX6675 Failed!");
292             }
293             else
294             {
295                 Serial.print("Current Temp: ");
296                 Serial.print(temperature);
297                 Serial.println(" C");
298 
299                 char tempStr[17];
300                 sprintf(tempStr, "Temp: %.2f C", temperature);
301 
302                 // Send data to Firebase Firestore (based on interval 60s)
303                 if (millis() - lastFirebaseSendMillis >= FIREBASE_SEND_INTERVAL_MS)
304                 {
305                     lastFirebaseSendMillis = millis();
306 
307                     Serial.print("Sending data to Firestore. Time: ");
308                     Serial.println(now.iso8601);
309 
310                     FirebaseJson content;
311                     content.set("fields/timestamp/stringValue", now.iso8601);
312                     content.set("fields/suhu/doubleValue", temperature);
313 
314                     String documentPath = "suhuData/" + String(timeClient.getEpochTime());
315 
316                     Serial.print("Document Path: ");
317                     Serial.println(documentPath);
318 
319                     if (Firebase.Firestore.createDocument(&fbdo, PROJECT_ID, "", documentPath.c_str(), content.raw()))
320                     {
321                         Serial.println("Data successfully sent to Firestore.");
322                         printLcd("Data Sent!", "Firebase OK!");
323                         delay(1000);
324                     }
```

Gambar 4. 11 Program monitoring suhu

```
325                     else
326                     {
327                         Serial.print("Failed to send data to Firestore: ");
328                         Serial.println(fbdo.errorReason());
329                         printLcd("Send Failed!", fbdo.errorReason().c_str());
330                         delay(2000);
331                     }
332                 }
333                 else
334                 { // If not time to send data
335                     // Display temperature and countdown on LCD
336                     char timeBuf[17];
337                     // Mengambil waktu dari 'now' yang sudah didapatkan dari getLocalTimeComponents()
338                     // dan memastikan waktu ditampilkan dari komponen yang valid.
339                     if (now.iso8601 != "Invalid Time")
340                     {
341                         unsigned long remainingTimeSec = (FIREBASE_SEND_INTERVAL_MS - (millis() - lastFirebaseSendMillis)) / 100
342                         sprintf(timeBuf, "%02d:%02d:%02d Next:%lds", now.hour, now.minute, now.second, remainingTimeSec);
343                     }
344                     else
345                     {
346                         // Fallback jika waktu masih invalid
347                         sprintf(timeBuf, "Time Invalid!");
348                     }
349                     printLcd(tempStr, timeBuf);
350                 }
351             }
352         }
353         else
354         { // NTP not synced
355             Serial.println("NTP not synced. Displaying status, not sending data.");
356             printLcd("NTP Syncing...", "Please wait...");
357         }
358     }
359     else
360     { // WiFi or Firebase not ready
```

Gambar 4. 12 Program monitoring suhu

```
361             Serial.println("WiFi or Firebase not ready. Waiting for connection...");
362             if (!WiFi.isConnected())
363             {
364                 printLcd("WiFi Disconnected!", "Reconnecting...");
365             }
366             else if (!Firebase.ready())
367             {
368                 printLcd("Firebase Offline!", "Reconnecting...");
369             }
370             delay(1000);
371         }
372
373         delay(100);
374     }
```

Gambar 4. 13 Program monitoring suhu