

BAB IV

PEMBUATAN ALAT

4.1 Pembuatan *Prototype*

Pembuatan Tugas Akhir ini dilakukan berdasarkan hasil perancangan perangkat keras dan perangkat lunak yang telah dijelaskan pada bab sebelumnya. Tahap pembuatan prototipe bertujuan untuk merealisasikan rancangan sistem monitoring dan pengendalian laju tetes infus berbasis Internet of Things (IoT) menjadi sebuah perangkat yang dapat bekerja sesuai dengan fungsi yang telah ditentukan. Proses pembuatan meliputi persiapan dan pemasangan komponen, perakitan perangkat keras, pembuatan program, konfigurasi sistem IoT, serta integrasi seluruh komponen menjadi satu kesatuan sistem. Secara umum, proses pembuatan prototipe dibagi menjadi dua bagian utama, yaitu pembuatan perangkat keras (hardware) dan pembuatan perangkat lunak (software).

Pada tahap pembuatan perangkat keras (hardware), dilakukan proses identifikasi dan persiapan komponen yang digunakan pada prototipe, seperti ESP32 DevKit, load cell, modul HX711, sensor tetes LM393, servo MG90S, OLED, push button, modul power supply HLK-PM01, serta komponen pendukung lainnya. Setiap komponen dipasang dan dihubungkan berdasarkan wiring diagram yang telah dirancang pada BAB III. Load cell dipasang pada bagian mekanisme penyangga wadah infus sehingga dapat menerima beban dari wadah yang berisi cairan NaCl 0,9%. Load cell kemudian dihubungkan dengan modul HX711 untuk menguatkan dan mengubah sinyal hasil pengukuran menjadi data digital yang dapat dibaca oleh ESP32.

Selanjutnya, sensor tetes LM393 dipasang pada bagian drip chamber untuk mendeteksi tetesan cairan infus yang keluar dari wadah. Sensor tersebut dihubungkan dengan ESP32 sebagai sumber data untuk memperoleh nilai TPM aktual. Servo MG90S dipasang pada mekanisme pengatur aliran selang infus dan dihubungkan dengan ESP32 sebagai aktuator. Pergerakan servo digunakan untuk melakukan penyesuaian terhadap mekanisme pengatur aliran berdasarkan hasil pengolahan nilai

TPM aktual dan TPM target. OLED dipasang sebagai media tampilan lokal untuk menampilkan informasi kondisi sistem, sedangkan push button digunakan sebagai perangkat input untuk memilih kategori yang telah ditentukan pada prototipe.

Setelah seluruh komponen perangkat keras terpasang, tahap berikutnya adalah pembuatan perangkat lunak (software). Pemrograman dilakukan menggunakan Visual Studio Code (VS Code) dengan PlatformIO dan bahasa pemrograman C/C++. Program diimplementasikan pada ESP32 sebagai pusat pengolahan sistem. Perangkat lunak dibuat untuk mengatur proses inisialisasi komponen, pembacaan load cell melalui HX711, pendeteksian tetesan menggunakan sensor LM393, penghitungan TPM aktual, pembacaan push button, pengaturan tampilan OLED, pengendalian servo MG90S, serta komunikasi data melalui jaringan Wi-Fi dan sistem IoT.

Setelah program berhasil diimplementasikan, dilakukan proses integrasi antara perangkat keras dan perangkat lunak. ESP32 membaca data dari load cell melalui HX711 untuk memperoleh informasi perubahan berat wadah infus yang selanjutnya digunakan untuk menentukan estimasi sisa cairan dalam satuan mL. Pada saat yang sama, sensor LM393 mendeteksi setiap tetesan cairan yang melewati drip chamber. Data tetesan tersebut diproses oleh ESP32 untuk memperoleh nilai TPM aktual. Nilai tersebut kemudian digunakan sebagai parameter monitoring dan sebagai umpan balik dalam proses pengendalian laju tetesan.

Sistem juga diintegrasikan dengan jaringan internet untuk mendukung fungsi IoT. ESP32 melakukan koneksi Wi-Fi sehingga data hasil pembacaan dan pengolahan dapat dikirimkan ke sistem monitoring. Pengguna dapat memantau informasi seperti sisa cairan infus, TPM aktual, TPM target, dan kategori yang dipilih melalui perangkat yang terhubung dengan internet. Nilai TPM target yang diberikan oleh pengguna digunakan sebagai nilai acuan dalam proses pengendalian. ESP32 membandingkan nilai TPM aktual dengan TPM target, kemudian memberikan perintah kepada servo MG90S untuk melakukan penyesuaian posisi mekanisme pengatur aliran.

Setelah seluruh bagian berhasil diintegrasikan, dilakukan pemeriksaan awal untuk memastikan setiap komponen dapat bekerja sesuai dengan fungsinya. Pemeriksaan meliputi pengujian pembacaan load cell, pembacaan sensor tetes LM393,

pergerakan servo MG90S, tampilan OLED, pembacaan push button, serta koneksi IoT. Selain itu, dilakukan kalibrasi load cell untuk memperoleh hasil pembacaan sisa cairan yang sesuai dengan nilai referensi. Pengujian awal tersebut dilakukan sebelum sistem digunakan pada tahap pengujian dan pengambilan data.

Dengan demikian, pembuatan prototipe menghasilkan suatu sistem yang mengintegrasikan monitoring sisa cairan, monitoring TPM aktual, pengaturan TPM target, dan pengendalian laju tetesan dalam satu perangkat. Seluruh proses kerja dikendalikan oleh ESP32 dengan memanfaatkan data dari load cell dan sensor LM393 sebagai masukan, servo MG90S sebagai aktuator, OLED sebagai tampilan lokal, serta koneksi internet sebagai media monitoring. Prototipe yang telah selesai selanjutnya digunakan pada tahap pengujian untuk mengetahui kinerja masing-masing bagian dan kinerja sistem secara keseluruhan.

4.2 Alat dan Bahan

Pada proses pembuatan prototipe sistem monitoring dan pengendalian laju tetes infus, diperlukan berbagai alat dan bahan yang digunakan untuk mendukung proses perakitan, pemrograman, pengujian, dan pengoperasian sistem. Alat dan bahan yang digunakan dipilih berdasarkan kebutuhan masing-masing bagian sistem agar seluruh komponen dapat terintegrasi dan bekerja sesuai dengan rancangan yang telah dibuat pada bab sebelumnya. Alat yang digunakan meliputi perangkat yang menunjang proses perakitan dan pemrograman, sedangkan bahan terdiri atas komponen elektronik, sensor, aktuator, perangkat catu daya, serta komponen mekanis yang digunakan untuk membangun prototipe. Komponen utama sistem meliputi ESP32 DevKit sebagai mikrokontroler, load cell dan HX711 sebagai sistem pengukuran berat, sensor LM393 sebagai pendeteksi tetesan, servo MG90S sebagai aktuator pengatur laju tetesan, OLED sebagai media tampilan, serta push button sebagai perangkat input. Selain itu, digunakan pula modul power supply dan komponen pendukung lainnya untuk memenuhi kebutuhan catu daya dan koneksi antarbagian rangkaian. Berikut alat dan bahan yang digunakan untuk pembuatan Tugas Akhir ini ditunjukkan pada tabel ...

Tabel 4.1 Bahan Pembuatan Tugas Akhir

No.	Nama	Jumlah
1	Infus NaCl 0,9%	1
2	ESP32 DevKit	1
3	Sensor LM393	1
4	Sensor Load Cell	1
5	Servo MG90S	1
6	Hi-Link PM01	1
7	Plat PCB	1
8	Box Panel	1
9	OLED 0.96 inch	1
10	Push Button	2
11	Terminal Block 2 pin	1
12	Powe Socket AC	1
13	Resistor 10k Ω	5
14	Kapasitor 100 μ F	3
15	JST 4 pin	1
16	JST 6 pin	1
17	JST 3 pin	1
18	Baut 10 mm	4
19	Kabel Spiral Putih 1m	1
20	Klem Ring Tiang	2
21	Isolasi	1

Berikutnya merupakan peralatan yang digunakan dalam proses pembuatan Tugas Akhir ditunjukkan pada Tabel 4.2.

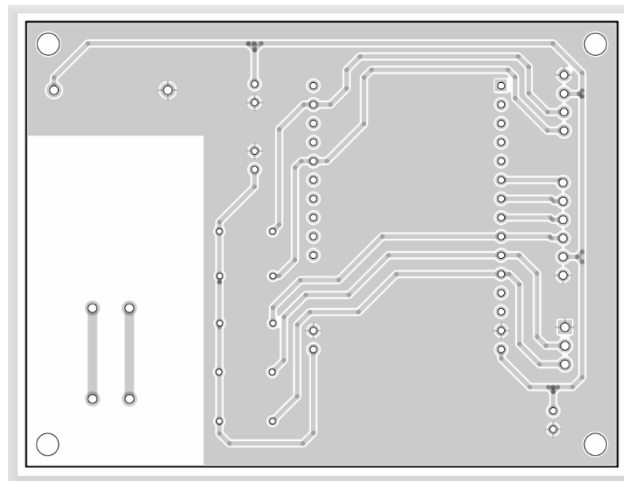
Tabel 4.2 Peralatan Pembuatan Tugas Akhir

No.	Nama	Jumlah
1	Solder	1
2	Obeng	1
3	Tang Potong	1
4	Bor	1
5	Gunting	1
6	Multimeter Digital	1
7	Laptop	1
8	HP	1

4.3 Perancangan dan Pembuatan PCB

Perancangan dan pembuatan Printed Circuit Board (PCB) dilakukan sebagai tahap realisasi rangkaian elektronik yang telah dirancang pada tahap sebelumnya. PCB digunakan sebagai media untuk menghubungkan beberapa jalur kelistrikan antar-komponen secara lebih terstruktur, sehingga rangkaian menjadi lebih rapi, kuat, dan mudah dalam proses pemasangan maupun pemeliharaan. Perancangan PCB dilakukan berdasarkan wiring diagram dan pembagian pin ESP32 yang telah ditentukan pada BAB III, dengan mempertimbangkan jalur catu daya, jalur input sensor, jalur output aktuator, serta jalur komunikasi antar-komponen.

Pada tahap perancangan, dilakukan penentuan posisi terminal, jalur track, titik pemasangan komponen, serta lubang pemasangan pada PCB. Jalur PCB dirancang agar hubungan antara ESP32, load cell/HX711, sensor LM393, servo MG90S, OLED, push button, dan sumber catu daya dapat terhubung sesuai dengan konfigurasi sistem. Selain itu, jalur tegangan dan ground dibuat dengan memperhatikan kebutuhan masing-masing komponen untuk mengurangi kemungkinan terjadinya kesalahan koneksi atau gangguan pada rangkaian.



Gambar 4.1 Perancangan Jalur PCB

Gambar 4.1 menunjukkan hasil rancangan jalur PCB yang digunakan sebagai dasar dalam proses pembuatan PCB. Rancangan tersebut kemudian direalisasikan

menjadi papan rangkaian yang digunakan pada prototipe sistem monitoring dan pengendalian laju tetes infus. Pembuatan PCB dilakukan melalui tahapan pencetakan atau fabrikasi jalur rangkaian, pengecekan jalur, pemasangan komponen, dan pemeriksaan kontinuitas untuk memastikan tidak terdapat jalur yang terputus maupun hubungan singkat (short circuit) sebelum PCB diintegrasikan dengan komponen lainnya.

Setelah PCB selesai dibuat dan diperiksa, papan rangkaian digunakan sebagai bagian dari sistem utama untuk menghubungkan komponen elektronik dengan ESP32. Dengan penggunaan PCB, proses perakitan prototipe menjadi lebih terorganisasi dan hubungan antar-komponen dapat dibuat sesuai dengan rancangan yang telah ditentukan.

4.4 Pembuatan Perangkat Lunak

Subbab ini menjelaskan proses implementasi perangkat lunak pada sistem monitoring dan pengendalian laju tetes infus. Program dikembangkan menggunakan Visual Studio Code dengan PlatformIO dan diimplementasikan pada mikrokontroler ESP32. Perangkat lunak berfungsi untuk mengatur proses inisialisasi sistem, membaca sensor load cell dan sensor tetes, mengolah data menjadi informasi volume dan TPM, mengatur posisi servo, menghubungkan ESP32 dengan jaringan Wi-Fi, serta melakukan pertukaran data dengan website dan Google Spreadsheet. Implementasi perangkat lunak dibuat berdasarkan rancangan sistem yang telah dijelaskan pada bab sebelumnya. Penggunaan program ESP32 untuk membaca sensor, mengendalikan aktuator, melakukan komunikasi IoT, dan menampilkan informasi pada OLED juga telah dijelaskan dalam rancangan perangkat lunak laporan.

4.4.1 Inisialisasi Sistem

Inisialisasi sistem merupakan tahap awal pada perangkat lunak yang bertujuan untuk mempersiapkan mikrokontroler ESP32 dan seluruh komponen yang digunakan sebelum sistem menjalankan proses monitoring dan pengendalian. Program utama dibuat pada file `main.cpp` dengan menggunakan library `Arduino.h` dan

MonitoringControlerInfus.h. Pada fungsi setup(), objek MonitoringControlerInfus dipanggil melalui fungsi INFUS.begin() untuk menjalankan seluruh proses inisialisasi sistem, sedangkan fungsi loop() digunakan untuk menjalankan menu utama sistem melalui fungsi INFUS.menuUtama().



```

main.cpp
src > main.cpp
1  #include <Arduino.h>
2  #include <MonitoringControlerInfus.h>
3
4  MonitoringControlerInfus INFUS;
5  void setup() {
6      INFUS.begin();
7  }
8
9  void loop() {
10     INFUS.menuUtama();
11 }

MonitoringControlerInfus.h
include > C MonitoringControlerInfus.h
71 class MonitoringControlerInfus {
72 public:
73     void begin() {
74         Serial.begin(115200);
75         // OLED
76         Wire.begin(
77             OLED_SDA,
78             OLED_SCL
79         );
80         Wire.setClock(400000);
81         display.begin(
82             SH1106_SWITCHCAPVCC,
83             SCREEN_ADDRESS
84         );
85         display.setTextColor(1);
86         display.setTextWrap(false);
87
88         // LITTLEFS
89         if (!LittleFS.begin(true)) {
90             Serial.println("LittleFS gagal");
91         }
92         File file = LittleFS.open("/kalibrasi.bin", FILE_READ);
93         if (file) {
94             calibration_factor = file.readStringUntil('\n').toFloat();
95             offset = file.readStringUntil('\n').toInt();
96             file.close();
97         }
98     }
99 }

```

Gambar 4.2 Proses Inisialisasi Sistem

Pada fungsi begin(), dilakukan beberapa proses inisialisasi, di antaranya komunikasi serial dengan baud rate 115200, konfigurasi OLED menggunakan komunikasi I2C, serta pengaturan tampilan awal OLED. Program juga melakukan inisialisasi sistem penyimpanan LittleFS yang digunakan untuk membaca data kalibrasi load cell yang sebelumnya telah tersimpan pada file kalibrasi.bin. Nilai calibration_factor dan offset yang diperoleh dari file tersebut kemudian digunakan pada

proses pembacaan sensor load cell. Dengan demikian, proses inisialisasi memastikan komponen dan parameter awal sistem telah tersedia sebelum perangkat memasuki proses pengoperasian.

4.4.2 Konfigurasi Sensor Load Cell

Sensor load cell digunakan sebagai salah satu sensor utama pada sistem untuk mengetahui perubahan berat cairan infus yang terdapat pada wadah infus. Perubahan berat tersebut digunakan sebagai dasar untuk mengetahui jumlah cairan infus yang masih tersedia selama proses monitoring. Dalam penggunaannya, load cell dihubungkan dengan modul HX711 yang berfungsi sebagai antarmuka pembacaan sensor sebelum data diteruskan dan diolah oleh mikrokontroler ESP32. Konfigurasi perangkat lunak sensor load cell meliputi pendefinisian pin komunikasi, deklarasi objek HX711, pengaturan nilai faktor kalibrasi dan offset, serta proses pembacaan parameter kalibrasi yang telah tersimpan pada sistem. Program juga dilengkapi dengan fungsi kalibrasi yang digunakan untuk memperoleh faktor kalibrasi berdasarkan beban acuan. Dengan adanya proses tersebut, hasil pembacaan load cell dapat digunakan oleh sistem untuk memperoleh nilai pengukuran yang lebih sesuai dengan kondisi beban yang diberikan. Penjelasan lebih lanjut mengenai konfigurasi sensor load cell pada perangkat lunak ditunjukkan pada bagian berikut.

```

35  /*=====LOADCELL DEFINE=====*/
36  #define LOADCELL_DOUT_PIN 33
37  #define LOADCELL_SCK_PIN 32
38  HX711 scale;

```

Gambar 4.3 Inisialisasi Load Cell

Berdasarkan Gambar 4.3, konfigurasi sensor load cell diawali dengan menentukan pin yang digunakan untuk menghubungkan modul HX711 dengan ESP32. Pin `LOADCELL_DOUT_PIN` ditetapkan pada GPIO 33, sedangkan `LOADCELL_SCK_PIN` ditetapkan pada GPIO 32. Kedua pin tersebut digunakan sebagai jalur komunikasi antara ESP32 dengan modul HX711. Selanjutnya, program mendeklarasikan objek HX711 dengan nama `scale` yang digunakan untuk melakukan proses pembacaan data dari sensor load cell. Pada program juga terdapat variabel

calibration_factor dan offset yang digunakan sebagai parameter dalam proses pengolahan hasil pembacaan sensor.

```

75         float calibration_factor = -596.51;
76         long offset = 0;
113         // LITTLEFS
114         if (!LittleFS.begin(true)) {
115             Serial.println("LittleFS gagal");
116         }
117         File file = LittleFS.open("/kalibrasi.bin", FILE_READ);
118         if (file) {
119             calibration_factor = file.readStringUntil('\n').toFloat();
120             offset = file.readStringUntil('\n').toInt();
121             file.close();
122         }
123         else {
124             // Nilai Defaul kalibrasi
125             calibration_factor = -596.51;
126         }
158         // HX711
159         scale.begin(
160             LOADCELL_DOUT_PIN,
161             LOADCELL_SCK_PIN
162         );
163         scale.set_scale(calibration_factor);
164         scale.set_offset(offset);

```

Gambar 4.4 Inisialisasi Modul HX711

Selanjutnya pada Gambar 4.4 menunjukkan modul HX711 diinisialisasi menggunakan pin LOADCELL_DOUT_PIN dan LOADCELL_SCK_PIN. Setelah proses inisialisasi berhasil, nilai faktor kalibrasi diterapkan menggunakan fungsi set_scale(), sedangkan nilai offset diterapkan menggunakan fungsi set_offset(). Penerapan kedua parameter tersebut bertujuan agar hasil pembacaan load cell dapat diolah sesuai dengan karakteristik sensor yang digunakan. Dengan demikian, data yang diperoleh dari load cell dapat digunakan oleh ESP32 sebagai dasar dalam menentukan nilai berat cairan infus.

Selain proses konfigurasi dan pembacaan, perangkat lunak juga menyediakan fungsi kalibrasi load cell. Proses kalibrasi dilakukan melalui tahapan yang ditampilkan pada OLED, mulai dari persiapan sensor, proses tare, pemasangan beban acuan, pembacaan nilai sensor, perhitungan faktor kalibrasi, hingga penyimpanan hasil kalibrasi. Pada tahap tare, sistem melakukan pembacaan tanpa beban untuk menentukan titik nol sensor. Setelah beban acuan diberikan, sistem mengambil

beberapa kali pembacaan untuk memperoleh nilai rata-rata yang digunakan dalam menentukan faktor kalibrasi. Faktor kalibrasi yang telah diperoleh kemudian diterapkan pada sensor dan disimpan ke dalam kalibrasi.bin sehingga dapat digunakan kembali pada proses pengoperasian berikutnya.

Hasil konfigurasi dan kalibrasi tersebut menjadi bagian penting dalam proses monitoring volume cairan infus. Nilai pembacaan load cell yang telah dikalibrasi selanjutnya diproses oleh ESP32 dan digunakan untuk memperoleh informasi jumlah cairan infus dalam satuan mililiter (mL). Data tersebut kemudian dapat ditampilkan pada OLED dan dikirimkan ke website sebagai informasi monitoring kondisi cairan infus. Dengan demikian, konfigurasi perangkat lunak load cell tidak hanya digunakan untuk memperoleh nilai pembacaan sensor, tetapi juga menjadi bagian dari rangkaian proses monitoring volume infus pada sistem yang dikembangkan.

4.4.3 Konfigurasi Sensor Tetes

Sensor tetes digunakan untuk mendeteksi setiap tetesan cairan infus yang melewati bagian *drip chamber* dan menghitung jumlah tetesan yang terjadi dalam satu menit. Informasi jumlah tetesan tersebut digunakan oleh mikrokontroler ESP32 untuk memperoleh nilai Tetes Per Menit (TPM) aktual sebagai salah satu parameter monitoring sistem. Pada perangkat lunak, konfigurasi sensor tetes meliputi penentuan pin input sensor, pembuatan variabel penghitung jumlah tetesan, serta pemrograman fungsi interrupt untuk mendeteksi perubahan sinyal dari sensor. Setiap sinyal yang terdeteksi akan dihitung sebagai satu tetesan dan digunakan dalam proses perhitungan TPM. Dengan demikian, sensor tetes dapat memberikan informasi mengenai laju tetesan cairan infus selama sistem beroperasi. Penjelasan lebih lanjut mengenai konfigurasi sensor tetes pada perangkat lunak ditunjukkan pada bagian berikut.

```
32  /*=====BUTTON DEFINE=====*/
33  #define SENSOR_TETES 26
34
```

Gambar 4.5 Konfigurasi Sensor Tetes

Berdasarkan Gambar 4.5, sensor tetes dikonfigurasi pada GPIO 26 ESP32 melalui pendefinisian `SENSOR_TETES`. Program juga menggunakan variabel `jumlahTetesISR` sebagai penghitung jumlah tetesan yang terdeteksi oleh sensor. Variabel tersebut menggunakan tipe `volatile` karena nilainya dapat berubah pada saat fungsi interrupt dijalankan. Selain itu, terdapat variabel `waktuTetesTerakhir` yang digunakan untuk menyimpan waktu terjadinya deteksi tetesan sebelumnya.

```

50 void IRAM_ATTR sensorTetesISR() {
51     unsigned long sekarang = micros();
52     if(sekarang - waktuTetesTerakhir > 50000) {
53         jumlahTetesISR++;
54         waktuTetesTerakhir = sekarang;
55     }
56 }

```

Gambar 4.6 Proses Deteksi Tetesan

Proses pendeteksian tetesan dapat dilihat pada Gambar 4.6 dilakukan menggunakan fungsi interrupt `sensorTetesISR()`. Fungsi tersebut akan dijalankan ketika terjadi perubahan sinyal pada pin sensor sesuai dengan mode interrupt yang telah ditentukan. Pada fungsi tersebut, program mengambil waktu saat sinyal diterima menggunakan fungsi `micros()`. Nilai waktu tersebut kemudian dibandingkan dengan waktu pendeteksian tetesan sebelumnya. Program memberikan jeda minimum sebesar 50.000 mikrodetik untuk membatasi pendeteksian sinyal yang terlalu berdekatan. Mekanisme tersebut digunakan agar satu tetesan tidak dihitung lebih dari satu kali akibat adanya perubahan sinyal atau gangguan pembacaan sensor.

```

170 // SENSOR TETES
171 pinMode(SENSOR_TETES, INPUT);
172 attachInterrupt(
173     digitalPinToInterrupt(SENSOR_TETES),
174     sensorTetesISR,
175     RISING
176 );
510 noInterrupts();
511 uint32_t jumlahTetesSekarang = jumlahTetesISR;
512 interrupts();
513
514 jumlahTetesMenit = jumlahTetesSekarang - jumlahTetesSebelumnya;

```

Gambar 4.7 Proses Inisialisasi Sensor Tetes

Pada Gambar 4.7 terjadi proses inisialisasi, pin sensor tetes diatur sebagai input menggunakan `pinMode()`. Selanjutnya, fungsi `attachInterrupt()` digunakan untuk menghubungkan pin sensor dengan fungsi `sensorTetesISR()`. Interrupt dikonfigurasi menggunakan kondisi `RISING`, sehingga fungsi interrupt dijalankan ketika terjadi perubahan sinyal naik pada keluaran sensor. Setiap sinyal yang memenuhi kondisi tersebut akan menyebabkan nilai penghitung `jumlahTetesISR` bertambah satu.

Nilai jumlah tetesan yang diperoleh kemudian digunakan untuk menghitung jumlah tetesan per menit. Program mengambil nilai jumlah tetesan saat ini dan membandingkannya dengan jumlah tetesan pada periode sebelumnya. Selisih kedua nilai tersebut digunakan sebagai nilai `jumlahTetesMenit` yang merepresentasikan jumlah tetesan yang terdeteksi dalam periode pengukuran. Nilai tersebut selanjutnya digunakan sebagai TPM aktual pada sistem monitoring.

Hasil pembacaan sensor tetes kemudian digunakan bersama dengan nilai TPM target yang telah ditentukan berdasarkan mode pengoperasian. Nilai TPM aktual tersebut ditampilkan sebagai informasi monitoring dan juga dikirimkan ke website bersama dengan data sistem lainnya. Dengan demikian, konfigurasi perangkat lunak sensor tetes berfungsi untuk mengubah kejadian fisik berupa tetesan cairan infus menjadi data digital berupa jumlah tetesan yang dapat diproses dan ditampilkan oleh sistem.

4.4.4 Konfigurasi Servo MG90S

Servo motor MG90S digunakan sebagai aktuator pada sistem untuk mengatur mekanisme laju tetesan cairan infus. Servo dikendalikan oleh ESP32 melalui sinyal PWM sehingga posisi sudut servo dapat diatur sesuai dengan nilai yang telah ditentukan pada program. Pengaturan posisi servo disesuaikan dengan mode pengoperasian yang dipilih melalui menu OLED, yaitu mode Ginjal, Dewasa, Kecil, dan Custom. Setiap mode memiliki nilai posisi servo yang berbeda sesuai dengan pengaturan pada program. Konfigurasi perangkat lunak servo meliputi deklarasi objek servo, penentuan pin yang digunakan, pengaturan frekuensi PWM, proses *attach* servo, serta pengiriman nilai sudut melalui fungsi *write()*. Penjelasan lebih lanjut mengenai konfigurasi servo MG90S pada perangkat lunak ditunjukkan pada bagian berikut.

```

58  /*=====SERVO DEFINE=====*/
59  #define SERVO_PIN 25
60  Servo servoInfus;
86      uint16_t servoGinjal = 50;
87      uint16_t servoDewasa = 70;
88      uint16_t servoKecil = 60;
89      uint16_t targetServo = 90;
191     // SERVO
192     servoInfus.setPeriodHertz(50);
193     servoInfus.attach(
194         SERVO_PIN,
195         500,
196         2400
197     );

```

Gambar 4.8 Proses Konfigurasi Servo

Berdasarkan Gambar 4.8, servo MG90S dikonfigurasi menggunakan pin GPIO 25 yang ditetapkan melalui variabel `SERVO_PIN`. Program kemudian membuat objek Servo dengan nama `servoInfus` yang digunakan untuk mengendalikan posisi servo. Sebelum servo digunakan, frekuensi PWM diatur sebesar 50 Hz melalui fungsi `setPeriodHertz(50)`. Selanjutnya, fungsi `attach()` digunakan untuk menghubungkan objek servo dengan pin GPIO yang telah ditentukan serta menetapkan rentang pulsa yang digunakan oleh servo.

Program juga menyediakan beberapa nilai posisi servo yang disesuaikan dengan mode pengoperasian. Pada mode Ginjal digunakan variabel servoGinjal, sedangkan mode Dewasa menggunakan servoDewasa dan mode Kecil menggunakan servoKecil. Untuk mode Custom, posisi servo menggunakan nilai targetServo yang dapat diatur sesuai kebutuhan sistem. Nilai-nilai tersebut kemudian diberikan kepada servo menggunakan fungsi servoInfus.write() sehingga servo bergerak menuju posisi yang telah ditentukan.

Ketika pengguna memilih mode melalui OLED menggunakan tombol CURSOR dan ENTER, program akan menjalankan posisi servo yang sesuai dengan mode yang dipilih. Sebagai contoh, ketika mode Ginjal dipilih, program memberikan nilai servoGinjal kepada fungsi servoInfus.write(). Hal yang sama dilakukan pada mode Dewasa dan Kecil dengan menggunakan nilai posisi servo masing-masing. Pada mode Custom, sistem menggunakan nilai targetServo sebagai posisi servo yang digunakan.

```

200 void servoBolakBalikSmooth() {
201     static int posisi = 30;
202     static int arah = 1;
203     static unsigned long lastServo = 0;
204     if (millis() - lastServo >= 20) {
205         lastServo = millis();
206         posisi += arah;
207         if (posisi >= 100) {
208             posisi = 100;
209             arah = -1;
210         }
211         if (posisi <= 30) {
212             posisi = 30;
213             arah = 1;
214         }
215         servoInfus.write(posisi);
216     }
217 }

```

Gambar 4.9 Pengaturan Posisi Servo

Pada Gambar 4.9 selain pengaturan posisi berdasarkan mode, program juga memiliki fungsi servoBolakBalikSmooth() yang digunakan untuk mengatur pergerakan servo secara bertahap. Fungsi tersebut mengatur perubahan posisi servo sehingga pergerakan aktuator dapat dilakukan secara lebih terkontrol sesuai dengan logika yang telah diprogram. Posisi servo yang dihasilkan kemudian digunakan untuk menggerakkan mekanisme pengatur aliran pada selang infus.

Dengan demikian, konfigurasi perangkat lunak servo MG90S berfungsi sebagai penghubung antara keputusan sistem berupa mode pengoperasian dan gerakan mekanis pada alat. Nilai posisi servo yang telah ditentukan dalam program diteruskan oleh ESP32 melalui sinyal PWM sehingga servo dapat mengatur posisi mekanisme pengendali tetesan infus.

4.4.5 Konfigurasi Pengaturan TPM

Nilai Tetes Per Menit (TPM) merupakan salah satu parameter utama yang digunakan pada sistem untuk menentukan laju tetesan cairan infus. Pengaturan TPM pada perangkat lunak dilakukan berdasarkan mode pengoperasian yang dipilih melalui menu OLED, yaitu mode Ginjal, Dewasa, Kecil, dan Custom. Setiap mode memiliki nilai TPM target yang telah ditentukan pada program, sedangkan nilai TPM aktual diperoleh dari hasil pembacaan sensor tetes. Selain melalui pengaturan pada program, nilai TPM target juga dapat disesuaikan melalui antarmuka website yang terhubung dengan ESP32. Dengan demikian, konfigurasi pengaturan TPM digunakan untuk menentukan nilai target yang menjadi acuan monitoring laju tetesan pada sistem.

Penjelasan lebih lanjut mengenai konfigurasi pengaturan TPM ditunjukkan pada bagian berikut.

```

77      uint16_t tpmGinjal = 0;
78      uint16_t tpmDewasa = 0;
79      uint16_t tpmKecil = 0;
80      uint16_t tpmCustom = 0;
81      uint16_t targetGinjal = 7;
82      uint16_t targetDewasa = 20;
83      uint16_t targetKecil = 17;
84      uint16_t targetCustom = 25;

```

```

219     void updateTargetMode() {
220         modeCustom = false;
221         if (mainMenu == GINJAL) {
222             targetTPM = targetGinjal;
223         }
224         else if (mainMenu == DEWASA) {
225             targetTPM = targetDewasa;
226         }
227         else if (mainMenu == KECIL) {
228             targetTPM = targetKecil;
229         }
230         else if (mainMenu == CUSTOM_MODE) {
231             modeCustom = true;
232             targetTPM = targetCustom;
233         }
234     }

534     // --- SINKRONISASI TPM AKTUAL SECARA REALTIME ---
535     if (mainMenu == GINJAL) {
536         tpmGinjal = jumlahTetesMenit;
537     }
538     else if (mainMenu == DEWASA) {
539         tpmDewasa = jumlahTetesMenit;
540     }
541     else if (mainMenu == KECIL) {
542         tpmKecil = jumlahTetesMenit;
543     }
544     else if (mainMenu == CUSTOM_MODE) {
545         tpmCustom = jumlahTetesMenit;
546     }

```

Gambar 4.10 Penentuan TPM Target

Berdasarkan Gambar 4.10, program menentukan nilai TPM target untuk setiap mode pengoperasian melalui beberapa variabel, yaitu target Ginjal, target Dewasa, target Kecil, dan target Custom. Nilai tersebut digunakan sebagai nilai awal TPM target pada masing-masing mode. Ketika sistem beroperasi, fungsi update Target Mode() akan menentukan nilai target TPM berdasarkan mode yang sedang dipilih. Apabila mode yang dipilih adalah Ginjal, maka nilai target TPM akan menggunakan nilai target Ginjal. Demikian pula pada mode Dewasa, Kecil, dan Custom, nilai target akan disesuaikan dengan variabel masing-masing.

Pemilihan mode dilakukan melalui menu yang ditampilkan pada OLED dengan menggunakan tombol CURSOR dan ENTER. Tombol CURSOR digunakan untuk berpindah dari satu pilihan mode ke pilihan lainnya, sedangkan tombol ENTER digunakan untuk mengonfirmasi mode yang dipilih. Setelah mode ditentukan, program akan menjalankan fungsi pengaturan target sehingga nilai TPM yang sesuai dengan mode tersebut dapat digunakan oleh sistem.

```

718 server.on("/api/target", HTTP_GET, [this]() {
719     if (!server.hasArg("tpm")) {
720         server.send(400, "text/plain", "Parameter tpm tidak ada");
721         return;
722     }
723     int nilai = server.arg("tpm").toInt();
724     if (nilai < 1) {
725         nilai = 1;
726     }
727     if (nilai > 120) {
728         nilai = 120;
729     }
730     targetTPM = nilai;
731     targetCustom = nilai;
732     if (mainMenu == GINJAL && targetTPM == targetGinjal) {
733         modeCustom = false;
734     }
735     else if (mainMenu == DEWASA && targetTPM == targetDewasa) {
736         modeCustom = false;
737     }
738     else if (mainMenu == KECIL && targetTPM == targetKecil) {
739         modeCustom = false;
740     }
741     else {

```

Gambar 4.11 Proses Deteksi Tetesan

Pada Gambar 4.11 selain nilai TPM target, program juga memperoleh nilai TPM aktual berdasarkan jumlah tetesan yang terdeteksi oleh sensor tetes. Nilai jumlah tetesan yang diperoleh kemudian diproses oleh program dan disimpan sebagai nilai TPM aktual. Data tersebut digunakan untuk menampilkan kondisi laju tetesan yang sedang terjadi pada sistem dan dibandingkan secara informasi dengan nilai TPM target yang telah ditentukan.

Program juga menyediakan pengaturan nilai TPM melalui website. Komunikasi antara website dan ESP32 dilakukan melalui endpoint/api/target. Ketika pengguna memberikan nilai TPM baru melalui website, ESP32 menerima nilai tersebut dan memperbarui nilai target TPM yang digunakan dalam sistem. Pada mode Custom, nilai

yang diterima dapat digunakan sebagai target Custom sehingga pengguna dapat menentukan nilai TPM sesuai kebutuhan pengujian.

Dengan demikian, konfigurasi pengaturan TPM pada perangkat lunak terdiri atas penentuan target berdasarkan mode, pembacaan TPM aktual dari sensor tetes, serta pengaturan nilai target melalui website. Nilai TPM aktual digunakan sebagai informasi kondisi laju tetesan, sedangkan nilai TPM target digunakan sebagai parameter pengaturan yang dapat ditampilkan dan disesuaikan selama sistem beroperasi.

4.4.6 Konfigurasi Koneksi WiFi dan Komunikasi IoT

Koneksi Wi-Fi merupakan bagian penting dalam sistem karena digunakan sebagai media komunikasi antara mikrokontroler ESP32 dengan perangkat pengguna melalui jaringan. Pada sistem yang dirancang, ESP32 dikonfigurasi untuk terhubung ke jaringan Wi-Fi sehingga data hasil pembacaan sensor dan informasi sistem dapat diakses melalui website. Selain sebagai penghubung dengan jaringan, ESP32 juga dikonfigurasi sebagai **WebServer** yang menangani proses pertukaran data antara perangkat dan antarmuka website. Konfigurasi komunikasi IoT meliputi pengaturan jaringan Wi-Fi, proses koneksi ESP32, serta inisialisasi WebServer yang digunakan dalam sistem. Penjelasan lebih lanjut mengenai konfigurasi koneksi Wi-Fi dan komunikasi IoT ditunjukkan pada bagian berikut.

```

43  /*=====WIFI DEFINE=====*/
44  const char* WIFI_SSID = "HANIFAH";
45  const char* WIFI_PASSWORD = "hanifah123";
46  const char* GOOGLE_SCRIPT_URL = "https://script.google.com/macros/s/

128      // HOTSPOT & WIFI
129      WiFi.mode(WIFI_AP_STA);
130      WiFi.softAP(
131          "MONITORING & KONTROLER INFUS",
132          "12345678"
133      );
134      WiFi.begin(
135          WIFI_SSID,
136          WIFI_PASSWORD
137      );
748      server.begin();
749      Serial.println("WebServer aktif");

```

Gambar 4.12 Proses Konfigurasi Jaringan

Berdasarkan Gambar 4.12, proses konfigurasi jaringan diawali dengan menentukan nama jaringan Wi-Fi melalui parameter `WIFI_SSID` dan kata sandi melalui `WIFI_PASSWORD`. Kedua parameter tersebut digunakan oleh ESP32 untuk melakukan koneksi ke jaringan yang telah ditentukan. Program kemudian mengatur mode Wi-Fi menggunakan fungsi `WiFi.mode(WIFI_AP_STA)`, sehingga ESP32 dapat menjalankan koneksi sebagai perangkat yang terhubung ke jaringan sekaligus menyediakan akses jaringan melalui mode Access Point.

Selanjutnya, program menjalankan fungsi `WiFi.begin()` dengan memasukkan parameter SSID dan password yang telah ditentukan. ESP32 kemudian melakukan proses koneksi ke jaringan Wi-Fi. Program melakukan pemeriksaan terhadap status koneksi menggunakan `WiFi.status()` hingga ESP32 berhasil terhubung dengan jaringan. Setelah koneksi berhasil, sistem dapat memperoleh alamat IP yang digunakan sebagai alamat akses untuk komunikasi dengan perangkat pengguna.

Selain koneksi Wi-Fi, program juga menginisialisasi objek `WebServer` dengan menggunakan port 80. `WebServer` digunakan untuk menyediakan layanan komunikasi antara ESP32 dengan website. Melalui `WebServer` tersebut, ESP32 dapat menerima permintaan dari website maupun memberikan respons berupa data hasil pembacaan sistem. Data yang dipertukarkan dapat berupa nilai volume infus, TPM aktual, TPM target, mode pengoperasian, posisi servo, dan kondisi infus.

Pada implementasinya, komunikasi antara ESP32 dan website dilakukan menggunakan endpoint yang telah ditentukan pada program. Website mengirimkan permintaan kepada ESP32 melalui jaringan Wi-Fi, kemudian ESP32 memproses permintaan tersebut dan memberikan respons sesuai dengan fungsi yang dipanggil. Salah satu bentuk komunikasi yang digunakan adalah pertukaran data dalam format JSON, sehingga data yang diperoleh dari sistem dapat diproses dan ditampilkan pada antarmuka website.

Dengan demikian, konfigurasi Wi-Fi dan komunikasi IoT memungkinkan ESP32 menjadi pusat komunikasi antara perangkat keras sistem dengan antarmuka website. Koneksi jaringan digunakan untuk mendukung proses monitoring dan

pengaturan sistem, sedangkan WebServer berfungsi sebagai penghubung dalam proses pertukaran data antara ESP32 dan perangkat pengguna.

4.4.7 Monitoring dan Pengiriman Data ke Website

Sistem monitoring pada perangkat lunak digunakan untuk menampilkan kondisi alat dan hasil pembacaan sensor melalui website. Data yang diperoleh dari sensor dan hasil pengolahan ESP32 dikumpulkan menjadi informasi monitoring yang meliputi volume infus, TPM aktual, TPM target, mode pengoperasian, posisi servo, dan kondisi infus. Data tersebut kemudian dikirimkan dari ESP32 ke website melalui komunikasi WebServer sehingga pengguna dapat memantau kondisi sistem menggunakan perangkat yang terhubung pada jaringan. Selain menampilkan data monitoring, komunikasi ini juga memungkinkan website mengirimkan nilai pengaturan tertentu kepada ESP32. Proses monitoring dan pengiriman data ke website ditunjukkan pada bagian berikut.

```

633 server.on("/api/data", HTTP_GET, [this]() {
653     json += "\",\"mode\":\":";
654     json += mode;
655     json += "\"";
656
657     json += "\",\"servo\":\":";
658     json += String(servoPos);
659
660     json += "\",\"targetServo\":\":";
661     json += String(targetServo);
662
663     json += "\",\"kondisi\":\":";
664     json += kondisi;
665     json += "\"";
666
667     json += "}";
668
669     server.send(
670         200,
671         "application/json",
672         json
673     );
674 });

```

Gambar 4.13 Proses Pengiriman Data ke Website

Berdasarkan Gambar 4.13, ESP32 menyediakan endpoint `/api/data` yang digunakan sebagai jalur permintaan data monitoring dari website. Ketika website meminta data tersebut, program mengambil nilai hasil pembacaan dan pengolahan sistem, kemudian menyusunnya ke dalam format JSON. Data yang dikirimkan meliputi nilai berat atau volume cairan, TPM aktual, jumlah tetesan per menit, TPM target, mode yang sedang digunakan, posisi servo, target posisi servo, serta kondisi infus.

Setelah seluruh data disusun, ESP32 mengirimkan respons kepada website menggunakan `server.send()` dengan tipe data `application/json`. Penggunaan format JSON memungkinkan setiap parameter sistem dikirimkan dalam bentuk data terstruktur sehingga dapat dibaca dan diolah oleh program JavaScript pada website.

Pada sisi website, proses pengambilan data dilakukan menggunakan fungsi `fetch()` yang mengakses endpoint `/api/data`. Data yang diterima kemudian diubah menjadi objek JSON dan dimasukkan ke dalam variabel sesuai dengan parameter masing-masing. Nilai tersebut selanjutnya digunakan untuk memperbarui tampilan monitoring pada website, seperti volume infus, TPM aktual, mode, posisi servo, dan kondisi infus.



```

void running() {
    unsigned long lastLoadCell = 0;
    unsigned long lastOLED = 0;
    float berat = 0;
    unsigned long lastGoogleSend = 0;

    noInterrupts();
    uint32_t jumlahTetesSebelumnya = jumlahTetesISR;
    interrupts();

    awalMenit = millis();
    jumlahTetesMenit = 0;

    while (1) {
        server.handleClient();
        if (millis() - lastGoogleSend >= 5000) {
            lastGoogleSend = millis();
            kirimKeGoogleSheets();
        }
    }
}

```

```

void running() {
    unsigned long lastLoadCell = 0;
    unsigned long lastOLED = 0;
    float berat = 0;
    unsigned long lastGoogleSend = 0;

    noInterrupts();
    uint32_t jumlahTetesSebelumnya = jumlahTetesISR;
    interrupts();

    awalMenit = millis();
    jumlahTetesMenit = 0;

    while (1) {
        server.handleClient();
        if (millis() - lastGoogleSend >= 5000) {
            lastGoogleSend = millis();
            kirimKeGoogleSheets();
        }
    }
}

```

Gambar 4.14 Proses Pengambilan Data

Selanjutnya, pada Gambar 4.14 dilakukan proses pengambilan data secara berkala sehingga informasi yang ditampilkan pada website dapat diperbarui selama sistem beroperasi. Dengan demikian, pengguna tidak perlu melakukan pembaruan halaman secara manual untuk mengetahui perubahan data yang berasal dari sensor. Mekanisme tersebut memungkinkan website berfungsi sebagai antarmuka monitoring yang menampilkan kondisi sistem berdasarkan data yang diperoleh secara langsung dari ESP32.

Selain mengirimkan data dari ESP32 menuju website, sistem juga menyediakan komunikasi dari website menuju ESP32 untuk pengaturan nilai tertentu. Salah satunya adalah pengaturan TPM target melalui endpoint yang telah disediakan pada WebServer. Nilai yang diberikan melalui website diterima oleh ESP32 dan digunakan untuk memperbarui parameter TPM target pada sistem.

Dengan demikian, proses monitoring dan pengiriman data pada sistem terdiri atas pembacaan data oleh ESP32, pengolahan data, penyusunan data dalam format JSON, pengiriman melalui WebServer, penerimaan data oleh website, dan pembaruan tampilan monitoring. Proses tersebut memungkinkan kondisi alat dan parameter pengoperasian dapat dipantau melalui website selama sistem terhubung dengan jaringan Wi-Fi.

4.4.8 Konfigurasi Antarmuka Website

Antarmuka website merupakan bagian yang digunakan sebagai media monitoring antara sistem dengan pengguna. Website dirancang agar informasi yang diperoleh dari ESP32 dapat ditampilkan dalam bentuk yang lebih mudah dipahami. Antarmuka website menampilkan beberapa parameter utama sistem, seperti volume cairan infus, TPM aktual, TPM target, mode pengoperasian, posisi servo, dan kondisi infus. Selain digunakan untuk monitoring, website juga menyediakan bagian pengaturan yang memungkinkan pengguna mengubah nilai TPM target yang akan digunakan oleh sistem. Konfigurasi antarmuka website terdiri atas penyusunan tampilan menggunakan HTML dan CSS serta pengolahan data menggunakan JavaScript. Tampilan antarmuka website ditunjukkan pada bagian berikut.

```

15         <section class="grid2">
16             <div class="card blueCard">
17                 <h2>VOLUME INFUS</h2>
18                 <div class="big">
19                     <span id="berat">MEMUAT...</span>
20                     <small>ml</small>
21                 </div>
22                 <p>Volume tersisa dalam wadah</p>
23             </div>
24             <div class="card greenCard">
25                 <h2>TPM AKTUAL</h2>
26                 <div class="big">
27                     <span id="tpm">0</span> <small>tpm</small></div><p>Tetesan per menit</p>
28                 </div>
29             </section>

```

Gambar 4.15 Informasi pada Website

Berdasarkan Gambar 4.15, tampilan website terdiri atas beberapa bagian informasi yang digunakan untuk menunjukkan kondisi sistem secara keseluruhan. Informasi volume infus digunakan untuk menampilkan jumlah cairan yang masih tersedia, sedangkan nilai TPM aktual menunjukkan jumlah tetesan yang terdeteksi oleh sensor dalam satu menit. Selain itu, terdapat informasi TPM target yang menunjukkan nilai laju tetesan yang telah ditentukan pada sistem.

Website juga menampilkan informasi mengenai mode pengoperasian yang sedang digunakan. Mode tersebut mengikuti pilihan yang telah ditentukan melalui menu pada OLED, yaitu mode Ginjal, Dewasa, Kecil, atau Custom. Dengan demikian, website berfungsi sebagai media untuk menampilkan mode yang sedang aktif, bukan sebagai media pemilihan kategori pasien.

Pada bagian monitoring juga ditampilkan informasi posisi servo yang menunjukkan nilai posisi aktuator yang sedang digunakan oleh sistem. Data tersebut diperoleh dari ESP32 dan diperbarui pada halaman website berdasarkan data yang diterima melalui endpoint /api/data. Selain itu, website menampilkan kondisi infus berdasarkan informasi yang diberikan oleh sistem sehingga pengguna dapat mengetahui keadaan alat selama proses monitoring berlangsung.

Konfigurasi antarmuka website juga menggunakan JavaScript untuk mengambil dan mengolah data dari ESP32. Proses pengambilan data dilakukan melalui fungsi fetch() terhadap endpoint yang telah disediakan oleh WebServer. Data dalam format JSON yang diterima kemudian digunakan untuk memperbarui elemen-elemen

pada halaman website sehingga informasi monitoring dapat berubah sesuai dengan kondisi sistem.

```

89  /* TARGET TPM */
90  function adjustTarget(value) {
91
92      data.target += value;
93
94
95      if (data.target < 1) {
96
97          data.target = 1;
98      }
99
100
101
102      if (data.target > 120) {
103
104          data.target = 120;
105      }
106
107
108
109      update();
110
111
112      fetch(
113          "/api/target?tpm=" +
114          data.target
115      )

```

Gambar 4.16 Pengaturan TPM

Selain menampilkan data, pada Gambar 4.16 website juga menyediakan bagian pengaturan TPM yang dapat digunakan untuk memasukkan nilai target baru. Nilai tersebut kemudian dikirimkan kembali ke ESP32 melalui endpoint pengaturan TPM. Dengan mekanisme tersebut, website tidak hanya berfungsi sebagai tampilan monitoring, tetapi juga sebagai antarmuka untuk melakukan pengaturan parameter TPM pada sistem.

Secara keseluruhan, konfigurasi antarmuka website mengintegrasikan HTML sebagai struktur halaman, CSS sebagai pengatur tampilan, dan JavaScript sebagai pengolah serta penghubung data dengan ESP32. Integrasi tersebut menghasilkan

antarmuka yang dapat digunakan untuk memantau parameter sistem dan melakukan pengaturan TPM melalui perangkat yang terhubung ke jaringan.

4.4.9 Konfigurasi Menu dan Tampilan OLED

OLED digunakan sebagai antarmuka lokal pada alat untuk menampilkan informasi dan menyediakan menu pengoperasian yang dapat dipilih oleh pengguna secara langsung pada perangkat. Pada sistem yang dirancang, OLED digunakan untuk menampilkan menu utama, pilihan mode pengoperasian, informasi hasil pembacaan sensor, serta proses kalibrasi load cell. Pengoperasian menu dilakukan menggunakan dua tombol, yaitu CURSOR untuk berpindah pilihan dan ENTER untuk mengonfirmasi pilihan. Dengan adanya OLED dan tombol tersebut, pengguna dapat melakukan pengaturan dasar alat tanpa harus menggunakan website. Konfigurasi menu dan tampilan OLED ditunjukkan pada bagian berikut.

```

62  /*=====ENUM MENU=====*/
63  enum MainDisplay {
64      GINJAL,
65      DEWASA,
66      KECIL,
67      CUSTOM_MODE,
68      KALIBRASILOADCELL
69  };

281  void menuUtama() {
282      // DISPLAY MENU
283      auto displayMenu = [&]() {
284          display.clearDisplay();
285          display.setCursor(53, 0);
286          display.print("MENU");
287          display.setCursor(21, 12);
288          display.print("GINJAL");
289          display.setCursor(21, 22);
290          display.print("DEWASA");
291          display.setCursor(21, 32);
292          display.print("KECIL");
293          display.setCursor(21, 42);
294          display.print("CUSTOM");
295          display.setCursor(21, 52);
296          display.print("KALIBRASI LC.");
297          display.display();
298      };
299      while(1) {
300          if (digitalRead(CURSOR) == LOW) {
301              mainMenu++;
302              if (mainMenu == 5) {
303                  mainMenu = 0;
304              }
305              delay(300);
306          }
307      }
308  }

```

Gambar 4.17 Inisialisasi OLED

Berdasarkan Gambar 4.17, OLED diinisialisasi pada saat sistem mulai dijalankan melalui fungsi `begin()`. Program menggunakan komunikasi I2C untuk menghubungkan ESP32 dengan OLED SH1106. Setelah proses inisialisasi berhasil, OLED digunakan untuk menampilkan informasi dan pilihan menu yang diperlukan selama proses pengoperasian alat.

Menu utama dijalankan melalui fungsi `menuUtama()` yang dipanggil secara berulang pada fungsi `loop()`. Pada menu tersebut, pengguna dapat memilih beberapa pilihan pengoperasian yang tersedia, yaitu Ginjal, Dewasa, Kecil, Custom, dan Kalibrasi LC. Pemilihan menu dilakukan menggunakan tombol CURSOR untuk menggeser pilihan, kemudian tombol ENTER digunakan untuk mengonfirmasi pilihan yang telah ditentukan.

Apabila pengguna memilih mode Ginjal, Dewasa, Kecil, atau Custom, program akan menjalankan konfigurasi sesuai dengan mode yang dipilih. Nilai target TPM kemudian disesuaikan melalui fungsi `updateTargetMode()`. Selain pengaturan nilai TPM, pemilihan mode juga berkaitan dengan posisi servo yang telah ditentukan pada program. Dengan demikian, menu OLED menjadi bagian yang digunakan untuk menentukan mode pengoperasian utama pada alat.

```

557 // Update OLED
558 if (millis() - lastOLED >= 500) {
559     lastOLED = millis();
560
561     // Update Status Kondisi Infus
562     String kondisi = getKondisiInfus(berat);
563     display.fillRect(0, 0, 128, 9, BLACK);
564     display.setCursor(0, 0);
565     display.print("STATUS: ");
566     display.print(kondisi);
567
568     display.fillRect(83, 37, 45, 10, BLACK);
569     display.setCursor(83, 37);
570     display.print(berat, 1);
571     display.print("ml");
572     display.fillRect(83, 52, 45, 10, BLACK);
573     display.setCursor(83, 52);
574     display.print(jumlahTetesMenit);
575     display.display();
576 }

```

Gambar 4.18 Menampilkan hasil monitoring

Selain menu pengoperasian, pada Gambar 4.18 OLED juga digunakan untuk menampilkan hasil monitoring dari sistem. Informasi yang dapat ditampilkan meliputi volume cairan infus, TPM aktual, TPM target, mode pengoperasian, serta informasi posisi servo sesuai dengan tampilan yang telah diprogram. Data tersebut berasal dari hasil pembacaan sensor dan pengolahan ESP32 sehingga pengguna dapat mengetahui kondisi sistem secara langsung melalui tampilan pada alat.

OLED juga digunakan dalam proses kalibrasi load cell. Ketika menu Kalibrasi LC dipilih, sistem menampilkan tahapan kalibrasi pada OLED, mulai dari persiapan sensor, proses tare, pemberian beban acuan, proses pembacaan sensor, hingga hasil kalibrasi. Informasi tersebut digunakan sebagai panduan bagi pengguna agar setiap tahapan kalibrasi dapat dilakukan sesuai dengan urutan yang telah ditentukan pada program.

Dengan demikian, konfigurasi menu dan tampilan OLED berfungsi sebagai antarmuka lokal alat yang memungkinkan pengguna melakukan pemilihan mode, melihat informasi monitoring, serta menjalankan proses kalibrasi load cell. OLED dan tombol CURSOR serta ENTER bekerja bersama dengan program ESP32 sehingga pengoperasian dasar alat dapat dilakukan secara langsung tanpa bergantung pada antarmuka website.

4.4.10 Pengiriman Data ke Google Sheets

Pengiriman data ke Google Sheets merupakan bagian dari sistem yang digunakan untuk menyimpan hasil monitoring alat secara daring. Data hasil pembacaan sensor dan parameter sistem yang telah diproses oleh ESP32 dikirimkan secara berkala ke Google Sheets sehingga data pengujian dapat terdokumentasi dan digunakan untuk proses analisis. Data yang dikirimkan berasal dari parameter monitoring sistem, seperti volume cairan infus, TPM aktual, TPM target, mode pengoperasian, dan informasi lain yang telah ditentukan pada program. Proses pengiriman data ke Google Sheets ditunjukkan pada bagian berikut.

```

479 void running() {
480     unsigned long lastLoadCell = 0;
481     unsigned long lastOLED = 0;
482     float berat = 0;
483     unsigned long lastGoogleSend = 0;
484
485     noInterrupts();
486     uint32_t jumlahTetesSebelumnya = jumlahTetesISR;
487     interrupts();
488
489     awalMenit = millis();
490     jumlahTetesMenit = 0;
491
492     while (1) {
493         server.handleClient();
494         if (millis() - lastGoogleSend >= 5000) {
495             lastGoogleSend = millis();
496             kirimKeGoogleSheets();
497         }

```

Gambar 4.19 Pengiriman ke GoogleSheets

Berdasarkan Gambar 4.19, proses pengiriman data dilakukan melalui fungsi `kirimKeGoogleSheets()`. Fungsi tersebut digunakan untuk mengumpulkan data hasil monitoring yang terdapat pada ESP32 sebelum dikirimkan ke layanan Google Sheets. Data yang akan dikirim terlebih dahulu disusun berdasarkan parameter yang telah ditentukan pada program sehingga setiap hasil pembacaan dapat dicatat secara terstruktur.

ESP32 menggunakan koneksi Wi-Fi sebagai media komunikasi untuk mengirimkan data menuju layanan Google Sheets. Setelah ESP32 terhubung dengan jaringan internet, fungsi pengiriman data dapat dijalankan untuk meneruskan hasil monitoring ke alamat layanan yang telah dikonfigurasi pada program. Dengan demikian, proses pencatatan data tidak dilakukan secara manual, tetapi dapat dilakukan oleh sistem selama perangkat memiliki koneksi jaringan.

```

752 void kirimKeGoogleSheets() {
792
793     WiFiClientSecure client;
794     client.setInsecure();
795     HTTPClient http;
796     http.begin(
797         client,
798         GOOGLE_SCRIPT_URL
799     );
800     http.setFollowRedirects(
801         HTTPC_DISABLE_FOLLOW_REDIRECTS
802     );
803     http.addHeader(
804         "Content-Type",
805         "application/json"
806     );
807
808     int httpCode = http.POST(json);
809     Serial.print("HTTP Google: ");
810     Serial.println(httpCode);
811     if (httpCode == 302) {
812         Serial.println("Data Terkirim");
813     }
814     http.end();
815 }

```

Gambar 4.20 Data Monitoring

Pada Gambar 4.20 data monitoring dikirimkan secara berkala selama alat beroperasi. Mekanisme pengiriman berkala tersebut memungkinkan setiap perubahan kondisi sistem dapat terdokumentasi dalam Google Sheets sesuai dengan waktu pengiriman data. Data yang tersimpan selanjutnya dapat digunakan untuk melihat perubahan volume infus, nilai TPM aktual, nilai TPM target, serta parameter lain yang diperlukan dalam proses pengujian alat.

Google Sheets juga digunakan sebagai media penyimpanan data hasil pengujian sehingga data yang telah dikirim dapat dikumpulkan dalam bentuk tabel. Penyimpanan tersebut memudahkan proses pemeriksaan dan pengolahan data karena hasil monitoring dapat diakses kembali untuk kebutuhan analisis pada tahap pengujian dan pembahasan.

Dengan demikian, konfigurasi pengiriman data ke Google Sheets berfungsi sebagai media pencatatan dan penyimpanan data monitoring secara daring. Integrasi

antara ESP32, koneksi Wi-Fi, dan Google Sheets memungkinkan hasil pembacaan sistem terdokumentasi secara berkala sehingga dapat mendukung proses pengambilan dan analisis data pada pengujian alat.