BAB IV
PEMBUATAN ALAT

4.1 Pembuatan Perangkat Keras

Pembuatan perangkat keras merupakan tahapan awal dalam proses perancangan sistem monitoring kedalaman dan suhu air berbasis kapal remote control menggunakan ESP32 dan komunikasi LoRa. Pada tahap ini dilakukan perakitan seluruh komponen yang mendukung kinerja alat, baik dari sisi kelistrikan maupun mekanik, yang terdiri dari mikrokontroler ESP32 sebagai pusat pengendali sistem, modul LoRa E220 sebagai media komunikasi data, sensor suhu DS18B20 untuk mengukur suhu air, sensor JSN-SR04T untuk mengukur kedalaman perairan, serta modul GPS untuk mengetahui titik koordinat lokasi pengukuran. Data hasil pembacaan sensor akan diproses oleh ESP32 kemudian dikirimkan melalui komunikasi LoRa menuju receiver untuk ditampilkan secara real-time pada LCD 20 x 4.

4.1.1 Alat dan Bahan

Proses selanjutnya yaitu melakukan persiapan alat dan bahan yang akan digunakan untuk perancangan sistem yang telah dibuat. Berikut daftar bahan dapat dilihat pada Tabel 4.1.

Tabel 4. 1 Keperluan Bahan

No	Nama Komponen	Jumlah
1	ESP32 Dev Kit V1	2 buah
2	Sensor JSN-SR04T	1 buah
3	Sensor DS18B20	1 buah
4	Modul GPS NEO-7M	1 buah
5	LoRa E220-900T22D	2 buah
6	XL4015 Step Down	2 buah
7	Modul Charger TP4056	2 buah
8	Baterai Li-ion 18650	4 buah

9	Antenna TX915_XPL-100	1 buah
10	Antenna TX915-JKD-20	1 buah
11	Holder Baterai	2 buah
12	Resistor 4.7K Ohm	1 buah
13	LCD 20 x 4 I2C	1 buah
14	Saklar	2 buah
15	Push Button	1 buah
16	Kabel Jumper	8 buah
17	Timah Solder	2 roll
18	Spacer	19 buah
19	Baut	19 buah
20	Mur	12 buah

Pada tabel 4.2 merupakan alat yang digunakan untuk pembuatan perangkat keras pada alat ini.

Tabel 4. 2 Keperluan Alat

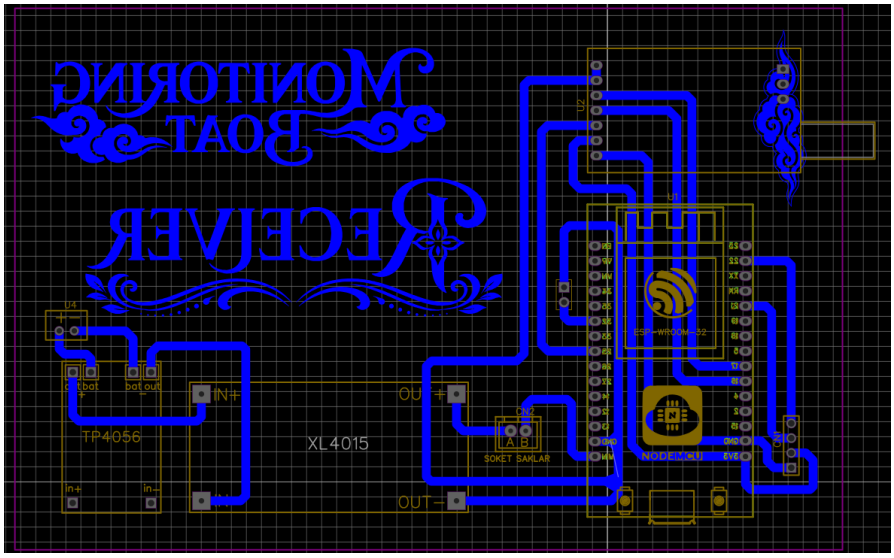
No	**Nama Alat**	**Jumlah**
1	Multimeter	1 buah
2	Set Obeng	1 buah
3	Gunting	1 buah
4	Tang potong	1 buah
5	Cutter	1 buah
6	Solder	1 buah
7	Bor	1 buah
8	Lem	1 buah
9	Meteran	1 buah
10	Alat Tulis	1 set
11	Akrilik 3 mm	5 buah

4.1.2 Pembuatan Perangkat Elektrik

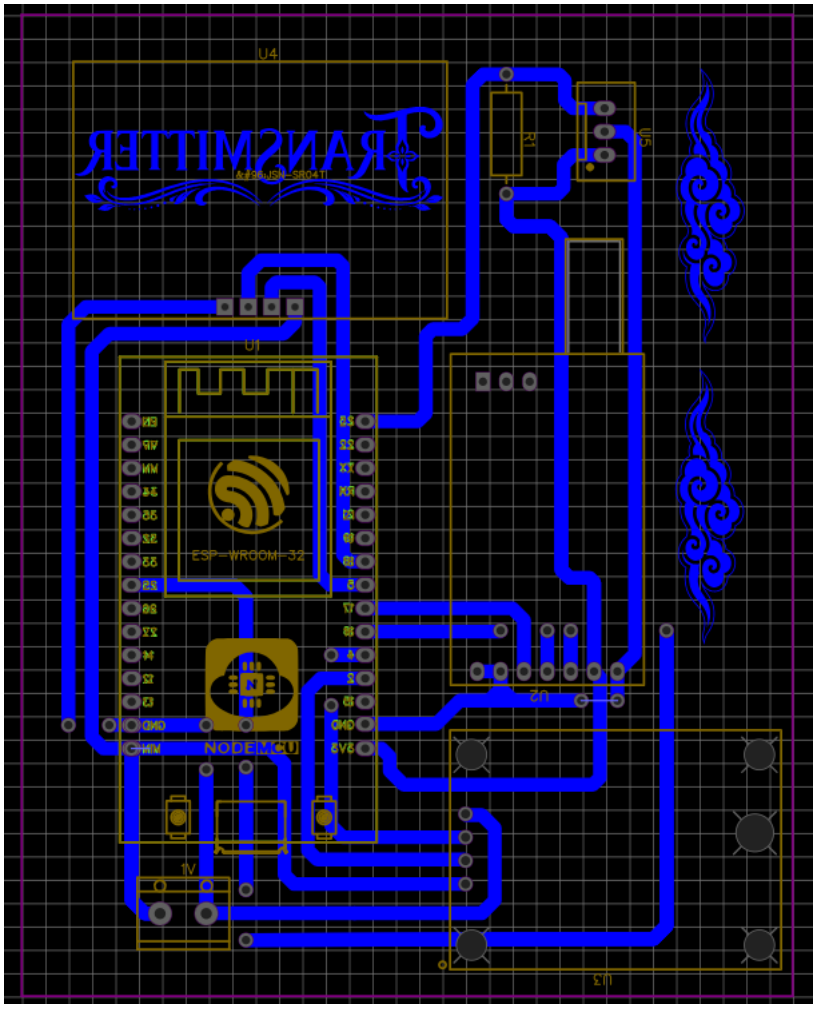
Pembuatan perangkat elektrik merupakan tahap penting dalam proses perakitan sistem monitoring kedalaman dan suhu air berbasis kapal remote control. Pada tahap ini dilakukan perancangan dan perakitan seluruh komponen elektronik yang berfungsi untuk membaca data sensor, mengolah data, serta mengirimkan data hasil pengukuran melalui komunikasi LoRa. Seluruh komponen elektrik dirangkai dalam satu sistem berbasis mikrokontroler ESP32 sebagai pusat pengendali utama. Adapun langkah – langkah pembuatan perangkat elektrik adalah sebagai berikut.

1. Perancangan Skematik dan Layout PCB

   Langkah awal dimulai dengan membuat skematik rangkaian elektronik menggunakan perangkat lunak EasyEDA seperti pada Gambar 4.1 dan 4.2. Dalam skematik ini, seluruh komponen seperti ESP32, modul LoRa E220, sensor suhu DS18B20, sensor JSN-SR04T, modul GPS, serta sistem catu daya disusun secara terstruktur agar koneksi antar komponen dapat ditentukan dengan jelas. Setelah skematik selesai, dilakukan perancangan layout PCB berdasarkan koneksi pada skematik tersebut. Proses ini meliputi penempatan jalur (routing), pengaturan ukuran papan PCB, serta penyesuaian posisi konektor agar seluruh komponen dapat terhubung dengan baik dan mendukung pemasangan perangkat pada kapal RC.



Gambar 4. 1 Layout PCB Receiver



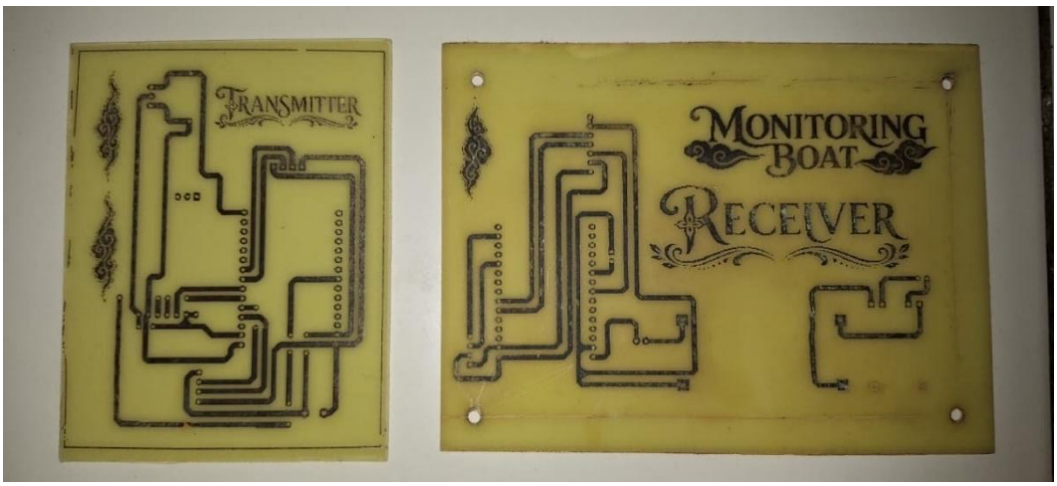
Gambar 4. 2 Layout PCB Transmitter

2. Proses Pencetakan dan Penyolderan PCB

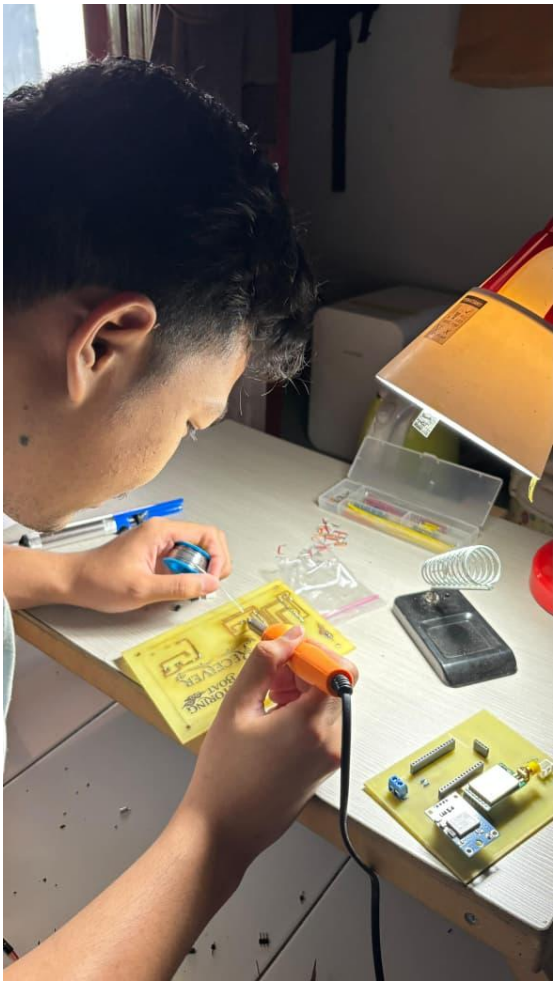
   Setelah layout selesai, file desain dicetak menggunakan printer laser kemudian dilakukan proses transfer jalur ke papan PCB dengan menggunakan autan dan mika bening lalu digosok menggunakan uang koin. Selanjutnya dilakukan proses etching menggunakan larutan ferric chloride ($FeCl_3$) untuk melarutkan bagian tembaga yang tidak terlindungi jalur rangkaian. Setelah proses etching selesai, PCB dibersihkan dan dilubangi menggunakan bor PCB untuk pemasangan komponen. Komponen seperti ESP32, modul LoRa E220, sensor suhu DS18B20, sensor JSN-SR04T, dan modul GPS. Komponen yang digunakan tidak disolder secara langsung ke PCB, melainkan menggunakan pin header dan konektor agar pemasangan serta perawatan komponen lebih

mudah dilakukan. Setelah seluruh komponen terpasang, dilakukan proses penyolderan dan pengecekan koneksi untuk memastikan seluruh rangkaian dapat bekerja dengan baik. Pada Gambar 4.3 ditunjukkan proses etching menggunakan larutan ferric chloride, lalu pada Gambar 4.4 menunjukkan hasil PCB yang telah selesai dirakit. Sedangkan Gambar 4.5 penyolderan komponen ke PCB.

Gambar 4. 3 Proses Etching



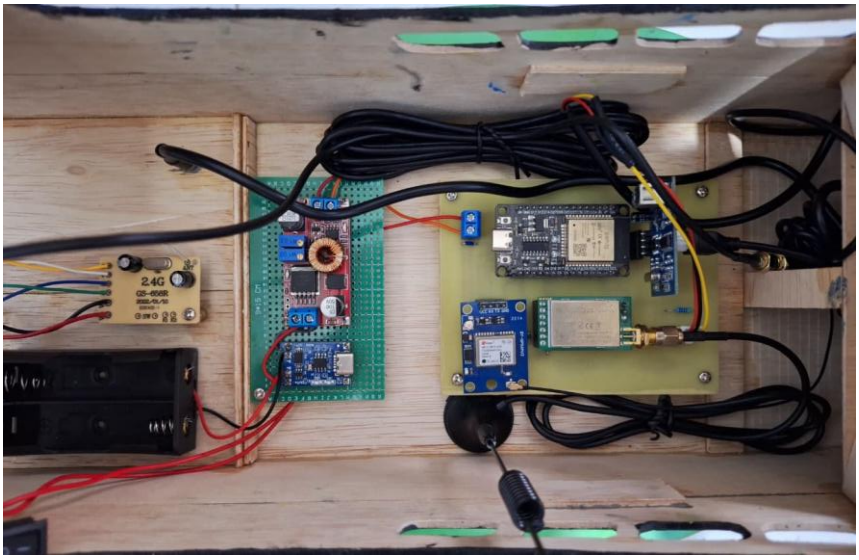
Gambar 4. 4 Hasil Cetak PCB



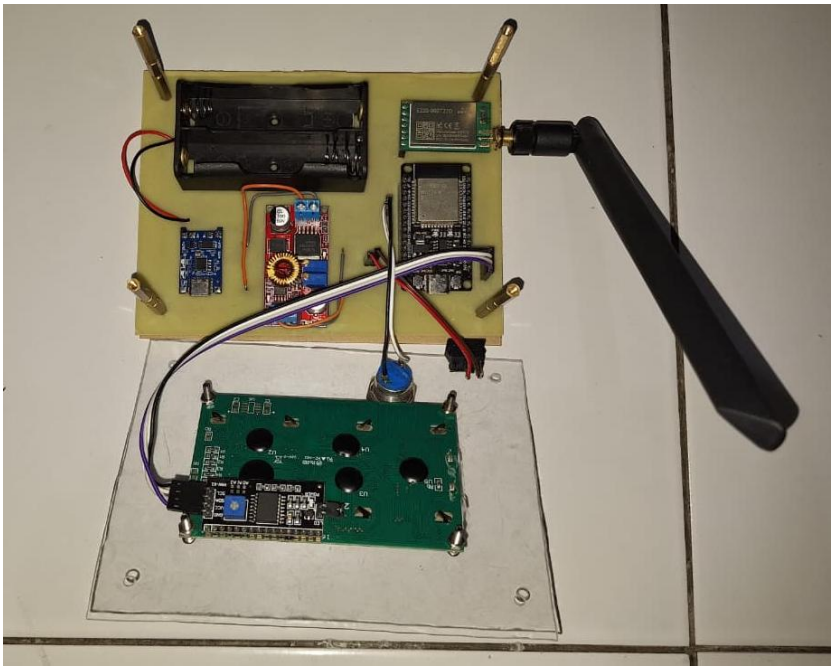
Gambar 4. 5 Penyolderan PCB

3. Integrasi Sistem

   Beberapa komponen seperti modul LoRa E220, sensor suhu DS18B20, sensor JSN-SR04T, modul GPS, dan LCD 20 x 4 tidak langsung disolder permanen ke PCB, melainkan dipasang menggunakan konektor agar posisi komponen lebih fleksibel saat dipasang pada kapal remote control serta memudahkan proses perawatan dan penggantian komponen apabila diperlukan. Penggunaan konektor juga membantu proses perakitan agar lebih rapi dan mempermudah pengecekan rangkaian ketika terjadi gangguan pada sistem. Gambar 4.6 dan 4.7 menunjukkan komponen yang telah terpasang pada PCB.



Gambar 4. 6 Komponen Yang Terpasang Pada Transmitter



Gambar 4. 7 Komponen Yang Terpasang Pada Receiver

4.1.3 Pembuatan Perangkat Mekanik

Perangkat mekanik pada sistem monitoring kedalaman dan suhu air dirancang untuk mendukung pemasangan dan perlindungan seluruh komponen elektronik agar dapat bekerja dengan baik di lingkungan perairan. Seluruh bagian mekanik meliputi dudukan sensor, tempat pemasangan PCB, wadah pelindung

komponen elektronik, serta penempatan antena dan baterai dirancang menyesuaikan bentuk body lambung kapal agar sistem tetap stabil saat dioperasikan. Adapun langkah – langkah pembuatan perangkat mekanik adalah sebagai berikut.

1. Proses Pemotongan Akrilik

   Proses pembuatan box receiver diawali dengan pemotongan akrilik sesuai ukuran yang telah dirancang. Pada Gambar 4.8 akrilik dipotong menggunakan bor tangan yang bisa digunakan untuk memotong akrilik agar menghasilkan bentuk yang presisi sesuai kebutuhan penempatan komponen receiver seperti ESP32, modul LoRa, LCD 20 x 4, dan catu daya. Setelah proses pemotongan selesai, setiap bagian akrilik dirakit dan direkatkan dengan menggunakan baut dan mur serta lem sehingga membentuk box receiver yang kokoh dan mampu melindungi komponen elektronik dari benturan maupun debu.

Gambar 4. 8 Pemotongan Akrilik

2. Proses Pendempulan dan Pengecetan Kapal

   Pada Gambar 4.9 pendempulan dilakukan pada body kapal untuk meratakan permukaan dan menutup bagian yang tidak tertutup rapat sebelum dilakukan pengecatan. Setelah dempul mengering, permukaan kapal diamplas hingga halus agar hasil pengecatan lebih rapi. Pada Gambar 4.10 yaitu pengecatan body bawah kapal menggunakan cat anti air agar dapat menghasilkan body bawah yang tidak tembus dari air

Gambar 4. 9 Pendempulan Body Kapal

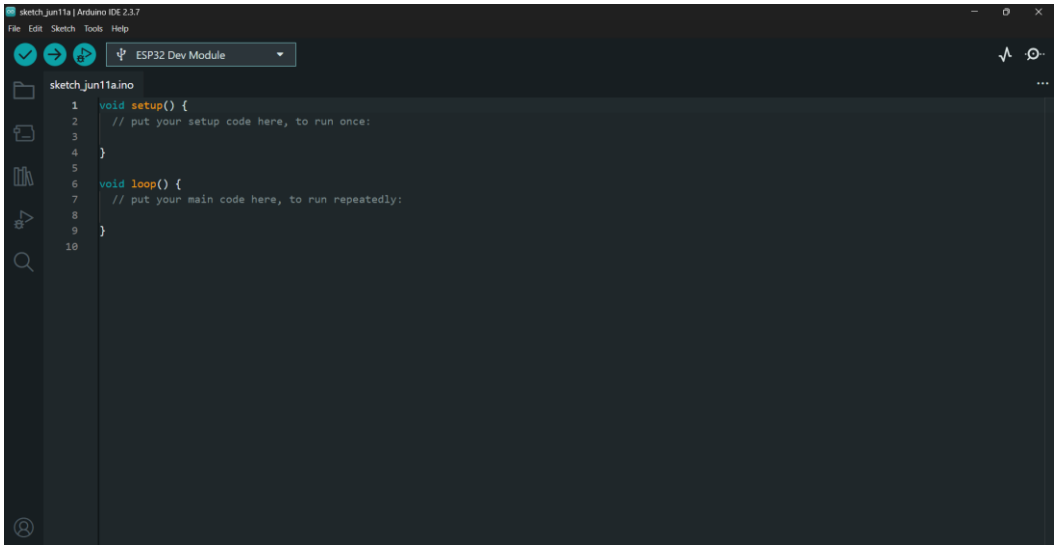
Gambar 4. 10 Pengecetan Bagian Body Bawah Kapal

4.2 Pembuatan Perangkat Lunak

Pembuatan perangkat lunak dalam proyek ini bertujuan untuk mengintegrasikan seluruh komponen sistem, baik pada sisi transmitter maupun receiver. Perangkat lunak dirancang untuk memastikan bahwa proses seluruh sistem dapat berjalan dengan baik.

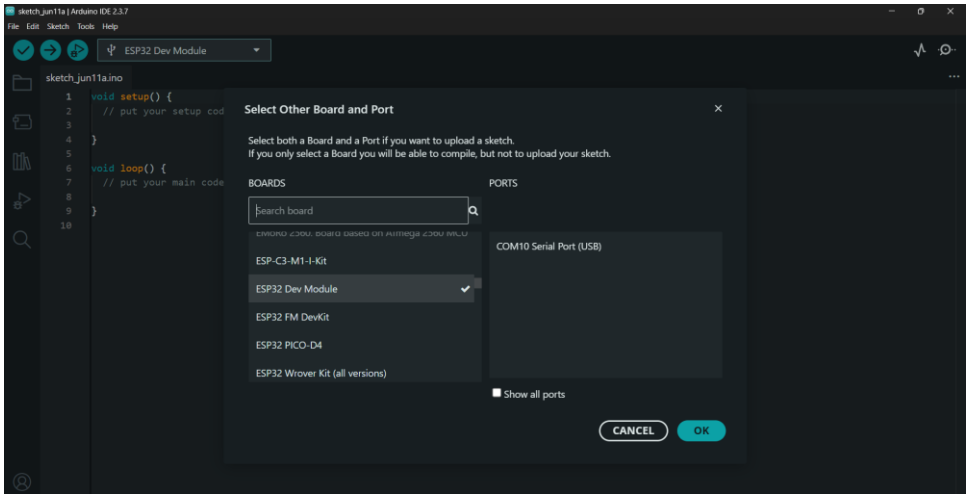
4.2.1 Pembuatan Program Pada Arduino IDE

Pada tahap ini, pembuatan perangkat lunak yang dapat mengendalikan sistem monitoring kedalaman dan suhu air dirancang menggunakan Arduino IDE. Proses diawali dengan membuka software Arduino IDE yang telah diinstal pada Laptop atau PC.



Gambar 4. 11 Tampilan Awal Software Arduino IDE

Setelah itu, lakukan konfigurasi awal dengan memilih board yang sesuai, yaitu ESP32 Dev Module, melalui menu Select Board dan menentukan port komunikasi yang terhubung dengan perangkat. Setelah pemilihan board ESP32 Dev Module, langkah selanjutnya adalah menulis program untuk Transmitter dan Receiver.



Gambar 4. 12 Pemilihan Board dan Port

4.2.2 Pembuatan Program Transmitter

Pada bagian transmitter berfungsi sebagai pusat pengambilan data serta pengiriman data yang terbaca pada kapal RC yang dikirim melalui komunikasi LoRa.

```
#include <HardwareSerial.h>
#include <LoRa_E220.h>
#include <TinyGPS++.h>
#include <OneWire.h>
#include <DallasTemperature.h>

// ─── PIN DEFINITION ──────────────────────────────────────
#define LORA_RX       16
#define LORA_TX       17
#define LORA_AUX      25
#define LORA_M0       -1
#define LORA_M1       -1

#define GPS_RX        4
#define GPS_TX        2

#define DS18B20_PIN   23
#define JSN_TRIG      5
#define JSN_ECHO      18
```

Gambar 4. 13 Inisialisasi Library dan Konfigurasi Pin Transmitter

Pada Gambar 4.13 menunjukkan pemanggilan beberapa library yang digunakan untuk mendukung komunikasi dan pembacaan sensor, yaitu HardwareSerial untuk komunikasi serial, LoRa_E220 untuk komunikasi LoRa, TinyGPS++ untuk pengolahan data GPS, OneWire dan DallasTemperature untuk

pembacaan sensor DS18B20. Selanjutnya dilakukan konfigurasi pin pada ESP32 yang digunakan untuk menghubungkan modul LoRa E220, GPS NEO-7M, sensor DS18B20, serta sensor JSN-SR04T. Konfigurasi ini bertujuan agar setiap perangkat dapat berkomunikasi dengan ESP32 sesuai fungsi masing – masing dalam proses pengambilan dan pengiriman data.

```
21  // ── SERIAL ──────────────────────────────────────────────
22  HardwareSerial LoRaSerial(2);   // UART2 untuk LoRa
23  HardwareSerial GPSSerial(1);    // UART1 untuk GPS
24
25  // ── OBJEK LIBRARY ───────────────────────────────────────
26  LoRa_E220     lora(&LoRaSerial, LORA_AUX, LORA_M0, LORA_M1);
27  TinyGPSPlus   gps;
28  OneWire        oneWire(DS18B20_PIN);
29  DallasTemperature ds18b20(&oneWire);
```

Gambar 4. 14 Deklarasi Komunikasi Serial dan Inisialisasi Objek Library

Pada Gambar 4.14 dilakukan deklarasi komunikasi serial dan pembuatan objek dari setiap library yang digunakan pada sistem transmitter. Objek LoRaSerial(2) mengaktifkan UART2 pada ESP32 untuk komunikasi dengan modul LoRa E220, sedangkan GPSSerial(1) menggunakan UART1 untuk menerima data dari modul GPS NEO-7M. Selanjutnya dibuat objek lora sebagai antarmuka komunikasi LoRa, objek gps untuk mengolah data koordinat yang diterima dari GPS, objek oneWire sebagai media komunikasi sensor DS18B20 melalui protokol One-Wire, serta objek ds18b20 untuk membaca data suhu air.

```
31  // ── INTERVAL (millis) ───────────────────────────────────
32  #define INTERVAL_GPS    1000
33  #define INTERVAL_JSN    1000
34  #define INTERVAL_DS     2000
35  #define INTERVAL_SEND   3000
36
37  unsigned long lastGPS  = 0;
38  unsigned long lastJSN  = 0;
39  unsigned long lastDS   = 0;
40  unsigned long lastSend = 0;
```

Gambar 4. 15 Pengaturan Interval Pembacaan Sensor dan Pengiriman Data

Pada Gambar 4.15 ditentukan interval pembacaan sensor dan pengiriman data menggunakan satuan milidetik (milliseconds). Interval pembacaan GPS dan sensor JSN-SR04T diatur setiap 1 detik, sensor DS18B20 setiap 2 detik, sedangkan pengiriman data melalui LoRa dilakukan setiap 3 detik. Selain itu, variabel lastGPS,

lastJSN, lastDS, dan lastSend bertipe unsigned long digunakan untuk menyimpan waktu pembacaan atau pengiriman terakhir, dengan nilai awal 0 yang menandakan proses tersebut belum pernah dijalankan sejak sistem dinyalakan.

```
// ── VARIABEL DATA GLOBAL ──────────────────────────────
// GPS
float  gpsLat    = 0.0;
float  gpsLon    = 0.0;
int    gpsHour   = 0;
int    gpsMin    = 0;
int    gpsSec    = 0;
bool   gpsValid  = false;

// JSN-SR04T
float  jsnDepth  = 0.0;   // cm

// DS18B20
float  dsTemp    = 0.0;   // C
```

Gambar 4. 16 Deklarasi Variabel Global Sistem Transmitter

Pada Gambar 4.16 deklarasi variabel – variabel global yang digunakan untuk menyimpan data GPS, kedalaman air, dan suhu air selama program berjalan. Variabel tersebut menyimpan informasi koordinat, waktu, status validitas GPS, hasil pengukuran kedalaman dari sensor JSN-SR04T, serta suhu air dari sensor DS18B20. Deklarasi variabel global memungkinkan data diakses oleh berbagai fungsi dalam proses pembacaan sensor, pengolahan data, dan pengiriman melalui komunikasi LoRa.

```
#define SOUND_SPEED_WATER  0.1670886f
#define JSN_MIN_DEPTH_CM    90.0f
#define JSN_MAX_DEPTH_CM    600.0f
#define JSN_TIMEOUT_US      60000
#define JSN_MEDIAN_SAMPLES  5
#define WIB_OFFSET          7
```

Gambar 4. 17 Konfigurasi Parameter Pengukuran Sensor dan Zona Waktu

Pada Gambar 4.17 ini didefinisikan beberapa konstanta yang digunakan dalam pembacaan sensor kedalaman JSN-SR04T, meliputi kecepatan rambat gelombang ultrasonik di dalam air, batas minimum dan maksimum kedalaman yang valid, batas waktu tunggu pembacaan (*timeout*), serta jumlah sampel pada metode filter median. Selain itu, konstanta WIB_OFFSET digunakan untuk mengonversi waktu GPS dari UTC ke zona waktu Indonesia Barat (UTC+7).

```
void setup() {
  Serial.begin(115200);
  Serial.println(F("[BOOT] Transmitter Monitoring Boat (Simplified)"));

  // Init LoRa Serial
  LoRaSerial.begin(9600, SERIAL_8N1, LORA_RX, LORA_TX);
  delay(100);

  // Init GPS Serial
  GPSSerial.begin(9600, SERIAL_8N1, GPS_RX, GPS_TX);
  delay(100);

  // Init LoRa E220
  lora.begin();
  Serial.println(F("[LoRa] Init OK"));

  // Init DS18B20
  ds18b20.begin();
  Serial.println(F("[DS18B20] Init OK"));

  // Init JSN-SR04T
  pinMode(JSN_TRIG, OUTPUT);
  pinMode(JSN_ECHO, INPUT);
  digitalWrite(JSN_TRIG, LOW);
  Serial.println(F("[JSN] Init OK"));

  Serial.println(F("[BOOT] Semua sensor siap."));
}
```

Gambar 4. 18 Inisialisasi Sistem Transmitter pada Fungsi setup()

Pada Gambar 4.18 fungsi setup() digunakan untuk menginisialisasi seluruh perangkat yang digunakan pada sistem transmitter, meliputi komunikasi serial, modul LoRa E220, GPS NEO-7M, sensor suhu DS18B20, serta sensor kedalaman JSN-SR04T. Setiap proses inisialisasi ditampilkan pada Serial Monitor sebagai indikator bahwa perangkat telah aktif. Setelah seluruh proses selesai, sistem menyatakan bahwa semua sensor siap digunakan untuk melakukan pembacaan dan pengiriman data melalui komunikasi LoRa.

```
void loop() {
  unsigned long now = millis();

  feedGPS();

  if (now - lastGPS >= INTERVAL_GPS) {
    lastGPS = now;
    readGPS();
  }

  if (now - lastJSN >= INTERVAL_JSN) {
    lastJSN = now;
    readJSN();
  }

  if (now - lastDS >= INTERVAL_DS) {
    lastDS = now;
    readDS();
  }

  if (now - lastSend >= INTERVAL_SEND) {
    lastSend = now;
    sendLoRaPacket();
  }
}
```

Gambar 4. 19 Fungsi loop() untuk Pembacaan Sensor dan Pengiriman Data

Pada Gambar 4.19 fungsi loop() digunakan untuk menjalankan proses utama sistem secara berulang. Program membaca data GPS, sensor kedalaman JSN-SR04T, dan sensor suhu DS18B20 sesuai interval yang telah ditentukan menggunakan fungsi millis(). Setelah seluruh data diperoleh, fungsi sendLoRaPacket() akan mengirimkan data ke receiver melalui modul LoRa. Penggunaan metode millis() memungkinkan proses pembacaan sensor dan pengiriman data berlangsung secara periodik tanpa menggunakan fungsi delay(), sehingga sistem bekerja lebih responsif.

```
119  void feedGPS() {
120    while (GPSSerial.available() > 0) {
121      gps.encode(GPSSerial.read());
122    }
123  }
```

Gambar 4. 20 Fungsi feedGPS() untuk Pembaruan Data GPS

Pada Gambar 4.20 Fungsi feedGPS() digunakan untuk membaca seluruh data yang diterima dari modul GPS melalui komunikasi serial. Data tersebut kemudian diproses oleh library TinyGPS++ menggunakan fungsi gps.encode(), sehingga informasi koordinat dan waktu GPS selalu diperbarui secara terus-menerus.

```
125  void readGPS() {
126    if (gps.location.isValid() && gps.location.isUpdated()) {
127      gpsLat   = gps.location.lat();
128      gpsLon   = gps.location.lng();
129      gpsValid = true;
130    }
131
132    if (gps.time.isValid() && gps.time.isUpdated()) {
133      // Konversi UTC ke WIB (UTC+7)
134      int utcHour = gps.time.hour();
135      gpsHour = (utcHour + WIB_OFFSET) % 24;
136      gpsMin  = gps.time.minute();
137      gpsSec  = gps.time.second();
138    }
139  }
```

Gambar 4. 21 Fungsi readGPS() untuk Pembacaan Data GPS

Pada Gambar 4.21 Fungsi readGPS() digunakan untuk membaca data koordinat dan waktu dari modul GPS yang telah diproses oleh TinyGPS++. Jika data valid, sistem akan menyimpan nilai *latitude*, *longitude*, serta mengonversi

waktu GPS dari UTC ke WIB (UTC+7) sebelum digunakan pada proses monitoring dan pengiriman data.

```
long jsnSingleRead() {
  digitalWrite(JSN_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(JSN_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(JSN_TRIG, LOW);
  return pulseIn(JSN_ECHO, HIGH, JSN_TIMEOUT_US);
}

float medianOf(float arr[], int n) {
  for (int i = 0; i < n - 1; i++) {
    for (int j = 0; j < n - i - 1; j++) {
      if (arr[j] > arr[j + 1]) {
        float tmp = arr[j]; arr[j] = arr[j + 1]; arr[j + 1] = tmp;
      }
    }
  }
  return arr[n / 2];
}
```

Gambar 4. 22 Fungsi Pembacaan Sensor dan Filter Median JSN-SR04T

Pada Gambar 4.22 Fungsi jsnSingleRead() digunakan untuk mengirimkan pulsa trigger dan membaca waktu pantulan gelombang ultrasonik dari sensor JSN-SR04T. Selanjutnya, fungsi medianOf() mengurutkan beberapa hasil pembacaan dan mengambil nilai tengah (median) sebagai hasil akhir untuk mengurangi noise serta meningkatkan kestabilan pengukuran kedalaman.

```
void readJSN() {
  float vSuara = SOUND_SPEED_WATER;

  float samples[JSN_MEDIAN_SAMPLES];
  int   validCount = 0;

  for (int i = 0; i < JSN_MEDIAN_SAMPLES; i++) {
    long durasi = jsnSingleRead(); // µs — durasi echo raw dari sensor

    if (durasi > 0) {
      float depth = (durasi * vSuara) / 2.0f;  // cm

      if (depth >= JSN_MIN_DEPTH_CM && depth <= JSN_MAX_DEPTH_CM) {
        samples[validCount++] = depth;
      }
    }
    delay(50);
  }

  if (validCount == 0) {
    jsnDepth = -1.0f; // tidak ada pembacaan valid (kemungkinan blind zone)
    Serial.println(F("[JSN] Tidak ada sample valid"));
    return;
  }

  jsnDepth = medianOf(samples, validCount);

  Serial.printf("[JSN] depth=%.2fcm | valid=%d/%d\n", jsnDepth, validCount, JSN_MEDIAN_SAMPLES);
}
```

Gambar 4. 23 Fungsi Pembacaan dan Validasi Data Kedalaman

Pada Gambar 4.23 Fungsi readJSN() digunakan untuk membaca data kedalaman dari sensor JSN-SR04T dengan mengambil beberapa sampel pengukuran. Data yang berada dalam rentang kedalaman yang valid akan disimpan dan diolah menggunakan metode median untuk memperoleh hasil yang lebih stabil. Apabila tidak terdapat data yang valid, sistem akan memberikan informasi bahwa pembacaan gagal dan menghentikan proses pembacaan.

```
192  void readDS() {
193    ds18b20.requestTemperatures();
194    float t = ds18b20.getTempCByIndex(0);
195
196    if (t == DEVICE_DISCONNECTED_C) {
197      dsTemp = -127.0;
198      return;
199    }
200
201    dsTemp = t;
202  }
```

Gambar 4. 24 Fungsi Pembacaan Suhu Menggunakan Sensor DS18B20

Pada Gambar 4.24 Fungsi readDS() digunakan untuk membaca suhu air dari sensor DS18B20. Program meminta sensor melakukan pengukuran, kemudian mengambil nilai suhu yang terbaca. Jika sensor tidak terhubung atau terjadi kegagalan pembacaan, nilai suhu diatur menjadi -127.0 sebagai penanda kesalahan. Apabila pembacaan berhasil, nilai suhu disimpan ke dalam variabel dsTemp untuk digunakan pada proses monitoring dan pengiriman data.

```
204  void sendLoRaPacket() {
205    char latStr[12], lonStr[12];
206
207    dtostrf(gpsLat, 9, 6, latStr);
208    dtostrf(gpsLon, 10, 6, lonStr);
209
210    char depthStr[8], tempStr[8];
211    dtostrf(jsnDepth, 5, 2, depthStr);
212    dtostrf(dsTemp,   5, 2, tempStr);
213
214    String latS = String(latStr);  latS.trim();
215    String lonS = String(lonStr);  lonS.trim();
216    String depS = String(depthStr); depS.trim();
217    String tmpS = String(tempStr);  tmpS.trim();
218
219    String gpsTag = gpsValid ? "G" : "X";
220
221    String msg = gpsTag + "," + latS + "," + lonS + ","
222               + String(gpsHour) + ":" + (gpsMin < 10 ? "0" : "") + String(gpsMin)
223               + ":" + (gpsSec < 10 ? "0" : "") + String(gpsSec)
224               + "|J," + depS
225               + "|D," + tmpS;
226
227    ResponseStatus rs = lora.sendMessage(msg);
228    delay(50);
229
230    Serial.print(F("[TX] "));
231    Serial.print(msg);
232    Serial.print(F(" → "));
233    Serial.println(rs.getResponseDescription());
234  }
```

Gambar 4. 25 Fungsi Penyusunan dan Pengiriman Paket Data LoRa

Pada Gambar 4.25 Fungsi sendLoRaPacket() digunakan untuk menyusun data GPS, kedalaman, dan suhu ke dalam satu paket data (*string*) sebelum dikirim melalui modul LoRa E220. Selanjutnya, paket data dikirim menggunakan fungsi lora.sendMessage(), kemudian status pengiriman ditampilkan pada Serial Monitor untuk memastikan proses transmisi berhasil.

4.2.3 Pembuatan Program Receiver

Pada bagian receiver berfungsi untuk menerima data yang dikirim oleh transmitter melalui komunikasi LoRa, kemudian mengolah dan menampilkan informasi berupa koordinat GPS, nilai kedalaman air, suhu air, serta status koneksi LoRa pada LCD secara real-time.

```
#include <HardwareSerial.h>
#include <LoRa_E220.h>
#include <LiquidCrystal_I2C.h>

// ── PIN DEFINITION ──────────────────────────────────────
#define LORA_RX     16
#define LORA_TX     17
#define LORA_AUX    25
#define LORA_M0     -1
#define LORA_M1     -1

#define BTN_PIN     32
#define LCD_ADDR    0x27
#define LCD_COLS    20
#define LCD_ROWS    4
```

Gambar 4. 26 Konfigurasi Library dan Definisi Pin Receiver

Pada Gambar 4.26 menunjukkan bagian awal program receiver yang berisi pemanggilan *library* dan konfigurasi pin yang digunakan pada sistem. *Library* HardwareSerial digunakan untuk komunikasi serial, LoRa_E220 untuk komunikasi data melalui modul LoRa E220, dan LiquidCrystal_I2C untuk mengendalikan LCD 20×4. Selain itu, pada bagian ini juga ditentukan alamat I2C LCD sebesar 0x27 dengan ukuran tampilan 20 kolom × 4 baris, sehingga seluruh komponen pada receiver dapat beroperasi sesuai fungsinya.

```
// ── SERIAL & OBJEK ──────────────────────────────────────
HardwareSerial  LoRaSerial(2);
LoRa_E220       lora(&LoRaSerial, LORA_AUX, LORA_M0, LORA_M1);
LiquidCrystal_I2C lcd(LCD_ADDR, LCD_COLS, LCD_ROWS);

// ── STATE HALAMAN ───────────────────────────────────────
int  currentPage   = 0;
#define TOTAL_PAGE 2

// ── TOMBOL ──────────────────────────────────────────────
bool     lastBtnState   = HIGH;
unsigned long lastDebounce = 0;
#define DEBOUNCE_MS     50
```

Gambar 4. 27 Inisialisasi Objek dan Variabel Receiver

Pada Gambar 4.27 menunjukkan inisialisasi objek komunikasi LoRa dan LCD, serta deklarasi variabel untuk pengaturan halaman LCD dan pembacaan push button. Selain itu, ditentukan jumlah halaman tampilan dan nilai *debounce* tombol agar perpindahan halaman berlangsung dengan stabil.

```
// ── DATA TERIMA ─────────────────────────────────────────
// GPS
bool   gpsValid  = false;
String gpsLat    = "---";
String gpsLon    = "---";
String gpsTime   = "--:--:--";

// JSN
String jsnDepth  = "---";

// DS18B20
String dsTemp    = "---";
String dsTempSt  = "---";

// LoRa Status
int    lastRSSI   = 0;
String loraStatus = "Offline";
```

Gambar 4. 28 Deklarasi Variabel Data Sensor pada Program Receiver

Pada Gambar 4.28 menunjukkan deklarasi variabel yang digunakan untuk menyimpan data yang diterima dari transmitter melalui komunikasi LoRa. Variabel tersebut meliputi data GPS berupa *latitude*, *longitude*, dan waktu, data sensor JSN-SR04T berupa kedalaman, serta data suhu dari sensor DS18B20. Selain itu, terdapat variabel untuk menyimpan nilai RSSI dan status komunikasi LoRa sebagai indikator kondisi koneksi antara transmitter dan receiver.

```
51  unsigned long lastReceive = 0;
52  #define OFFLINE_TIMEOUT  5000
53
54  // LCD refresh
55  unsigned long lastLCDUpdate = 0;
56  #define LCD_REFRESH_MS   500
```

Gambar 4. 29 Pengaturan Waktu Timeout dan Refresh LCD

Pada Gambar 4.29 menunjukkan deklarasi variabel yang digunakan untuk mengatur status komunikasi LoRa dan pembaruan tampilan LCD. Variabel lastReceive digunakan untuk menyimpan waktu penerimaan data terakhir, sedangkan OFFLINE_TIMEOUT sebesar 5000 ms digunakan sebagai batas waktu untuk mengubah status komunikasi menjadi Offline apabila tidak ada data yang diterima. Selain itu, variabel lastLCDUpdate dan LCD_REFRESH_MS sebesar 500 ms digunakan untuk mengatur interval pembaruan tampilan LCD agar informasi yang ditampilkan selalu diperbarui secara berkala.

```
58  void setup() {
59    Serial.begin(115200);
60    Serial.println(F("[BOOT] Receiver Monitoring Boat"));
61
62    // Init Button
63    pinMode(BTN_PIN, INPUT_PULLUP);
64
65    // Init LCD
66    lcd.init();
67    lcd.backlight();
68    lcd.clear();
69    lcd.setCursor(0, 0); lcd.print(F("===================="));
70    lcd.setCursor(0, 1); lcd.print(F("MONITORING  PERAIRAN"));
71    lcd.setCursor(0, 2); lcd.print(F("Kapal Remote Control"));
72    lcd.setCursor(0, 3); lcd.print(F("===================="));
73    delay(3000);
74
75    // Init LoRa
76    LoRaSerial.begin(9600, SERIAL_8N1, LORA_RX, LORA_TX);
77    delay(100);
78    lora.begin();
79    Serial.println(F("[LoRa] Init OK"));
80
81    lcd.clear();
82    showPage(currentPage);
83  }
```

Gambar 4. 30 Inisialisasi Sistem pada Fungsi setup()

Pada Gambar 4.30 menunjukkan fungsi setup() yang digunakan untuk menginisialisasi komunikasi serial, push button, LCD 20×4, dan modul LoRa E220. Setelah seluruh komponen berhasil diinisialisasi, LCD akan menampilkan halaman utama sehingga sistem receiver siap menerima data dari transmitter.

```
85  void loop() {
86    unsigned long now = millis();
87
88    handleButton(now);
89
90  if (lora.available() > 0) {
91    ResponseContainer rc = lora.receiveMessage();
92    if (rc.status.code == E220_SUCCESS) {
93
94      lastReceive = now;
95      loraStatus  = "Online";
96
97      parsePacket(rc.data);
98      Serial.println("========== DATA MASUK ==========");
99      Serial.println(rc.data);
100
101   } else {
102
103     Serial.print("[LoRa ERROR] ");
104     Serial.println(rc.status.getResponseDescription());
105   }
106 }
107   if ((now - lastReceive) > OFFLINE_TIMEOUT && lastReceive != 0) {
108     loraStatus = "Offline";
109   }
110   if (now - lastLCDUpdate >= LCD_REFRESH_MS) {
111     lastLCDUpdate = now;
112     showPage(currentPage);
113   }
114 }
```

Gambar 4. 31 Proses Utama pada Fungsi loop()

Pada Gambar 4.31 menunjukkan fungsi loop() yang dijalankan secara berulang untuk membaca tombol, menerima data dari transmitter melalui LoRa, serta memperbarui status komunikasi. Jika data berhasil diterima, program akan memproses dan menampilkan data pada LCD, sedangkan jika tidak ada data yang diterima dalam waktu tertentu, status komunikasi akan berubah menjadi Offline.

```
116  void handleButton(unsigned long now) {
117    static bool buttonPressed = false;
118    bool reading = digitalRead(BTN_PIN);
119
120
121    if (reading == LOW && !buttonPressed) {
122      buttonPressed = true;
123      currentPage++;
124      if (currentPage >= TOTAL_PAGE) {
125        currentPage = 0;
126      }
127
128      lcd.clear();
129      showPage(currentPage);
130      Serial.print("Page: ");
131      Serial.println(currentPage);
132      delay(200); // debounce sederhana
133    }
134
135    if (reading == HIGH) {
136      buttonPressed = false;
137    }
138  }
```

Gambar 4. 32 Fungsi Pembacaan Tombol pada Receiver

Pada Gambar 4.32 menunjukkan fungsi handleButton() yang digunakan untuk membaca status push button dan mengatur perpindahan halaman pada LCD. Setiap kali tombol ditekan, halaman tampilan akan berpindah ke halaman berikutnya dan kembali ke halaman awal setelah mencapai batas halaman yang ditentukan. Selain itu, fungsi ini menerapkan debounce sederhana untuk mencegah pembacaan tombol secara berulang akibat pantulan sinyal.

```
140  void parsePacket(String raw) {
141    raw.trim();
142
143    int sep1 = raw.indexOf('|');
144    int sep2 = raw.indexOf('|', sep1 + 1);
145
146    if (sep1 < 0 || sep2 < 0) {
147      Serial.println(F("[PARSE] Format paket tidak valid"));
148      return;
149    }
150
151    String segGPS = raw.substring(0, sep1);
152    String segJSN = raw.substring(sep1 + 1, sep2);
153    String segDS  = raw.substring(sep2 + 1);
```

Gambar 4. 33 Fungsi Parsing Data yang Diterima dari LoRa

Pada Gambar 4.33 menunjukkan fungsi parsePacket() yang digunakan untuk memproses data yang diterima dari transmitter melalui LoRa. Pada fungsi ini, data diperiksa formatnya kemudian dipisahkan menjadi tiga bagian, yaitu data GPS, sensor JSN-SR04T, dan sensor DS18B20. Jika format data tidak valid, program akan menampilkan pesan kesalahan pada Serial Monitor.

```
155   if (segGPS.length() > 2) {
156     char gpsFlag = segGPS.charAt(0);
157     gpsValid = (gpsFlag == 'G');
158
159     String gpsBody = segGPS.substring(2);
160     int c1 = gpsBody.indexOf(',');
161     int c2 = gpsBody.indexOf(',', c1 + 1);
162
163     if (c1 > 0 && c2 > 0) {
164       gpsLat  = gpsBody.substring(0, c1);
165       gpsLon  = gpsBody.substring(c1 + 1, c2);
166       gpsTime = gpsBody.substring(c2 + 1);
167     }
168   }
```

Gambar 4. 34 Proses Pengolahan Data GPS pada Fungsi parsePacket()

Pada Gambar 4.34 menunjukkan proses pengolahan data GPS yang telah dipisahkan dari paket data LoRa. Program memeriksa status validitas GPS, kemudian mengekstrak nilai *latitude*, *longitude*, dan waktu pembacaan untuk disimpan ke dalam variabel sehingga dapat ditampilkan pada LCD.

```
170   if (segJSN.startsWith("J,")) {
171     String body = segJSN.substring(2);
172     int c1 = body.indexOf(',');
173     int c2 = body.indexOf(',', c1 + 1);
174
175     if (c1 > 0 && c2 > 0) {
176       jsnDepth  = body.substring(0, c1);
177     }
178   }
```

Gambar 4. 35 Proses Pengolahan Data Sensor JSN-SR04T

Pada Gambar 4.35 menunjukkan proses pengolahan data yang diterima dari sensor JSN-SR04T. Program memeriksa format data, kemudian mengambil nilai

kedalaman air dari paket yang diterima dan menyimpannya ke dalam variabel jsnDepth untuk digunakan pada proses monitoring dan ditampilkan pada LCD.

```
180   if (segDS.startsWith("D,")) {
181     String body = segDS.substring(2);
182     int c1 = body.indexOf(',');
183
184     if (c1 > 0) {
185       dsTemp   = body.substring(0, c1);
186       dsTempSt = body.substring(c1 + 1);
187     }
188   }
189 }
```

Gambar 4. 36 Proses Pengolahan Data Sensor DS18B20

Pada Gambar 4.36 menunjukkan proses pengolahan data yang diterima dari sensor DS18B20. Program memeriksa format data, kemudian mengambil nilai suhu air dan status sensor dari paket yang diterima untuk disimpan ke dalam variabel dsTemp dan dsTempSt, sehingga dapat ditampilkan pada LCD sebagai informasi monitoring suhu air.

```
191   void showPage(int page) {
192     switch (page) {
193
194       case 0: pageStatus();  break;
195       case 1: pagePosisi();  break;
196     }
197   }
```

Gambar 4. 37 Fungsi Pemilihan Halaman Tampilan LCD

Pada Gambar 4.37 menunjukkan fungsi showPage() yang digunakan untuk memilih halaman yang ditampilkan pada LCD berdasarkan nilai variabel page. Pada program ini terdapat dua halaman, yaitu pageStatus() untuk menampilkan informasi status sistem dan pagePosisi() untuk menampilkan informasi posisi kapal.

```
199 void lcdCenter(int row, String text) {
200   int len = text.length();
201   if (len > LCD_COLS) len = LCD_COLS;
202   int pad = (LCD_COLS - len) / 2;
203   lcd.setCursor(pad, row);
204   lcd.print(text);
205 }
```

Gambar 4. 38 Fungsi Menampilkan Teks di Tengah LCD

Pada Gambar 4.38 menunjukkan fungsi lcdCenter() yang digunakan untuk menampilkan teks di bagian tengah LCD. Fungsi ini menghitung panjang teks dan menentukan posisi awal tampilan sehingga teks dapat ditampilkan secara rata tengah pada baris LCD yang dipilih.

```
207 void lcdLabelVal(int row, String label, String val, String unit = "") {
208   String line = label + ": " + val;
209   if (unit.length() > 0) line += " " + unit;
210   // Pad spasi agar bersih
211   while (line.length() < LCD_COLS) line += " ";
212   line = line.substring(0, LCD_COLS);
213   lcd.setCursor(0, row);
214   lcd.print(line);
215 }
```

Gambar 4. 39 Fungsi Menampilkan Label dan Nilai pada LCD

Pada Gambar 4.39 menunjukkan fungsi lcdLabelVal() yang digunakan untuk menampilkan data pada LCD dalam format label dan nilai. Fungsi ini juga menambahkan satuan jika diperlukan serta menyesuaikan panjang teks agar tampilan pada LCD tetap rapi dan mudah dibaca.

```
217 void pageStatus() {
218   lcd.setCursor(0, 0); lcd.print(F("====   STATUS   ===="));
219   lcdLabelVal(1, "Kedalaman", jsnDepth, "cm");
220   lcdLabelVal(2, "Suhu      ", dsTemp, "C");
221   lcdLabelVal(3, "LoRa      ", loraStatus);
222 }
```

Gambar 4. 40 Fungsi Tampilan Halaman Status pada LCD

Pada Gambar 4.40 menunjukkan fungsi pageStatus() yang digunakan untuk menampilkan halaman status pada LCD. Halaman ini menampilkan informasi utama hasil monitoring, yaitu nilai kedalaman air, suhu air, dan status komunikasi LoRa sehingga pengguna dapat memantau kondisi sistem secara langsung.

```
224 void pagePosisi() {
225   String latDisp  = gpsValid ? gpsLat  : "Searching...";
226   String lonDisp  = gpsValid ? gpsLon  : "Searching...";
227   String timDisp  = gpsValid ? gpsTime : "--:--:--";
228
229   lcd.setCursor(0, 0); lcd.print(F("=== POSISI KAPAL ==="));
230   lcdLabelVal(1, "Lat ", latDisp);
231   lcdLabelVal(2, "Long", lonDisp);
232   lcdLabelVal(3, "WIB ", timDisp);
233 }
```

Gambar 4. 41 Fungsi Tampilan Halaman Posisi Kapal pada LCD

Pada Gambar 4.41 menunjukkan fungsi pagePosisi() yang digunakan untuk menampilkan informasi posisi kapal pada LCD. Halaman ini menampilkan koordinat latitude, longitude, dan waktu pembacaan GPS. Apabila data GPS belum valid, LCD akan menampilkan keterangan Searching... sebagai indikasi bahwa modul GPS masih mencari sinyal satelit.