BAB IV

HASIL DAN PEMBAHASAN

4.1 Implementasi

Pada bab ini membahas tentang proses implementasi dan pengujian alat monitoring detak jantung. Implementasi mencakup perancangan dan perakitan alat. Tujuan dari tahap implementasi ini yaitu untuk memastikan bahwa semua komponen sistem, baik perangkat keras dan perangkat lunak, bisa berfungsi secara terpadu sesuai pada diagram alir implementasi yang telah dirancang.

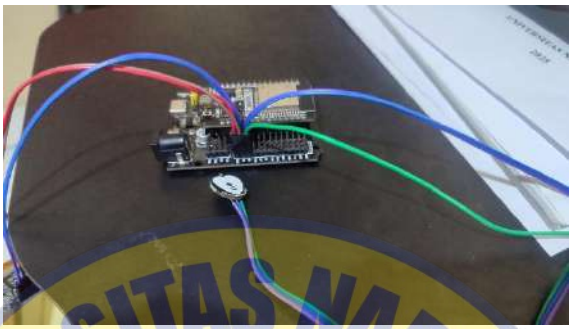
Pengujian dilakukan untuk mengevaluasi fungsionalitas dan kinerja dari sistem yang telah dibuat. Hal ini penting dilakukan guna memastikan bahwa alat bekerja sesuai dengan tujuan awal, serta untuk mendeteksi secara dini jika terdapat kesalahan pada salah satu bagian sistem. Dengan demikian, proses analisis dan perbaikan akan lebih mudah dilakukan.

4.1.2 Implementasi Perangkat Keras

Implementasi perangkat keras merupakan langkah awal untuk mewujudkan sistem monitoring detak jantung yang dirancang. Pada langkah ini, semua komponen fisik yang dibutuhkan dirakit dan dikonfigurasikan agar dapat bekerja secara terintegrasi. Perangkat keras utama yang dipakai dalam alat ini meliputi sensor detak jantung (Pulse Sensor) yang berfungsi untuk mendeteksi sinyal detak jantung pengguna secara real-time, mikrokontroler ESP32 yang memiliki kemampuan pemrosesan data serta mendukung konektivitas nirkabel melalui

Bluetooth Low Energy (BLE), dan layar OLED yang digunakan untuk menampilkan hasil pembacaan detak jantung secara langsung kepada pengguna.

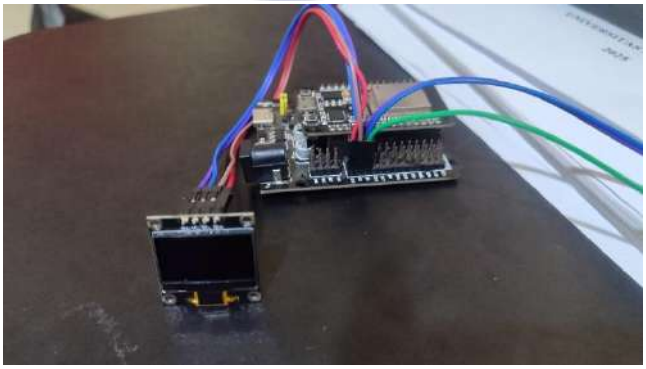
- Wiring Komponen Pulse Sensor ke ESP32



Gambar 4.1 Wiring Komponen Pulse Sensor ke ESP32

Pada Gambar 4.1 merupakan proses wiring Pulse Sensor dan ESP32 Pin VCC pada Pulse Sensor disambungkan ke pin 3.3V pada ESP32 untuk memberikan catu daya yang sesuai dengan logika kerja ESP32, yang beroperasi pada tegangan 3.3V. Pin GND disambungkan ke pin GND pada ESP32 untuk melengkapi sirkuit listrik. Sementara itu, pin Signal yang berfungsi mengirimkan sinyal detak jantung dalam bentuk analog dihubungkan ke GPIO 4.

- Wiring Komponen OLED ke ESP32



Gambar 4.2 Wiring Komponen OLED ke ESP32

Pada Gambar 4.2 merupakan proses wiring antara OLED dan mikrokontroler ESP32 pin VCC OLED dihubungkan ke pin 3.3V pada ESP32 untuk memberikan tegangan kerja yang sesuai. Pin GND disambungkan ke pin GND pada ESP32 untuk menyelesaikan sirkuit daya. Selanjutnya, pin SDA OLED dihubungkan ke GPIO 15 berfungsi untuk mengirimkan dan menerima data secara serial dan pin SCL OLED ke GPIO 2 untuk mengirimkan sinyal clock dari master untuk menyinkronkan pengiriman data.

4.1.3 Implementasi Perangkat Lunak

Implementasi perangkat lunak merupakan langkah penting dalam mewujudkan sistem monitoring detak jantung yang dirancang. Pada tahap ini, perangkat lunak dikembangkan dan dikonfigurasikan agar dapat berfungsi secara terintegrasi dengan perangkat keras yang telah dibuat. Perangkat lunak utama yang dipakai pada sistem ini meliputi program *mikrokontroler* pada ESP32 yang berfungsi untuk membaca data dari *Pulse Sensor*, melakukan pemrosesan sinyal, serta menghitung nilai detak jantung per menit (BPM). Selain itu, perangkat lunak juga mengatur proses komunikasi data melalui *Bluetooth Low Energy (BLE)* untuk mengirimkan hasil pembacaan ke aplikasi mobile, dan menampilkan nilai BPM secara *real-time* pada layar OLED. Implementasi perangkat lunak ini dirancang agar sistem dapat bekerja secara otomatis, responsif, dan efisien dalam mendukung fungsi pemantauan detak jantung.

- Inisialisasi Komponen

  Langkah awal dalam implementasi perangkat lunak adalah melakukan inisialisasi komponen yang terhubung dengan mikrokontroler

ESP32. Inisialisasi ini bertujuan untuk memastikan bahwa semua perangkat keras, seperti *Pulse Sensor*, layar OLED, dan modul *Bluetooth Low Energy (BLE)*, telah dikonfigurasi dengan benar sebelum sistem mulai menjalankan fungsi utamanya.

a) Inisialisasi Pulse Sensor

```
// ===== PULSE SENSOR CONFIGURATION =====
#define PULSE_INPUT 4                    // PulseSensor terhubung ke GPIO 4
#define THRESHOLD 550                    // Batas sensitivitas deteksi detak jantung
PulseSensorPlayground pulseSensor;       // Objek sensor detak jantung
```

Gambar 4.3 Configuration Pulse Sensor

Pada Gambar 4.3 merupakan Konfigurasi awal *Pulse Sensor* dilakukan dengan mendefinisikan pin input dan ambang batas deteksi. Pin GPIO 4 pada ESP32 ditetapkan sebagai jalur input sinyal dari *Pulse Sensor* melalui #define PULSE_INPUT 4, sedangkan nilai threshold sebesar 550 ditentukan dengan #define THRESHOLD 550 untuk menyaring noise dan hanya mendeteksi puncak sinyal detak jantung yang valid. Selanjutnya, objek *pulseSensor* dari library *PulseSensorPlayground* dibuat untuk menangani pembacaan dan pemrosesan sinyal detak jantung secara terintegrasi. Konfigurasi ini menjadi dasar dalam proses inisialisasi dan pengolahan data sensor.

```
// Variabel Global untuk Fitur Kalibrasi Otomatis Threshold
int dynamicThreshold = INITIAL_THRESHOLD;
unsigned long calibrationStartTime = 0;
const unsigned long calibrationDuration = 7000; // 7 detik
int minRawSignalDuringCal = 4095;
int maxRawSignalDuringCal = 0;
const int MIN_SIGNAL_SWING_FOR_CALIBRATION = 100;
// BATAS ATAS BARU UNTUK THRESHOLD
const int MAX_CALIBRATED_THRESHOLD = 1950; // Tidak akan melebihi nilai ini
```

Gambar 4.4 Kalibrasi Otomatis Threshold

Pada Gambar 4.4 mengatur proses kalibrasi otomatis pada sensor detak jantung agar alat bisa membaca detak dengan lebih akurat. Ada beberapa bagian penting,

seperti *dynamicThreshold* yang menyimpan batas sensitivitas sensor yang bisa berubah selama kalibrasi. *calibrationStartTime* mencatat kapan kalibrasi dimulai, dan calibrationDuration menentukan lamanya waktu kalibrasi, yaitu selama 7 detik. Selama kalibrasi, alat mencatat nilai sinyal terendah dan tertinggi yang diterima sensor untuk mengetahui seberapa besar perubahan sinyalnya. Ada juga batas minimum perubahan sinyal (MIN_SIGNAL_SWING_FOR_CALIBRATION) yang harus terpenuhi supaya kalibrasi dianggap berhasil. Terakhir, ada batas maksimum untuk nilai sensitivitas sensor agar tetap stabil dan tidak terlalu sensitif. Jadi, semua ini membantu alat supaya bisa menyesuaikan diri dengan kondisi pengguna agar hasil pengukuran detak jantung lebih tepat..

b) Inisialisasi OLED

```
// ===== OLED CONFIGURATION =====
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET    -1        // Tidak pakai reset pin
#define SCREEN_ADDRESS 0x3C     // Alamat I2C OLED SSD1306

// Buat objek OLED
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
```

Gambar 4.5 Configuration OLED

Pada Gambar 4.5 merupakan proses Konfigurasi awal OLED dilakukan dengan menetapkan ukuran layar menggunakan #define SCREEN_WIDTH 128 dan #define SCREEN_HEIGHT 64, yang menunjukkan resolusi OLED dalam piksel. Pin reset tidak digunakan, sehingga diset ke -1. Alamat I2C OLED, yaitu 0x3C, ditentukan melalui #define SCREEN_ADDRESS 0x3C agar ESP32 dapat berkomunikasi dengan modul OLED. Selanjutnya, objek display dari library Adafruit_SSD1306 dibuat untuk mengontrol tampilan layar, dengan parameter

resolusi, jalur komunikasi I2C (&Wire), dan konfigurasi pin reset. Konfigurasi ini memungkinkan ESP32 menampilkan informasi detak jantung secara real-time melalui layar OLED.

```
// --- FUNGSI-FUNGSI UNTUK MENAMPILKAN PESAN DI LAYAR OLED (Diperbarui) ---
void showInitialMessageOLED() {
  display.clearDisplay();
  display.drawBitmap(56, 8, heart_icon_bmp, 16, 14, SSD1306_WHITE); // Hati di tengah atas
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(30, 35);
  display.println("Detak Jantung");
  display.setCursor(45, 50);
  display.println("Mulai?");
  display.display();
}

void showCalibratingMessageOLED() {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(20, 8);
  display.println("Kalibrasi Sensor");
```

Gambar 4.6 Potongan Kode Pengelolaan Tampilan Status dan Informasi pada OLED

Pada Gambar 4.6 merupakan potongan code yang berisi fungsi-fungsi untuk menampilkan berbagai pesan dan status sistem pada layar OLED. Fungsi-fungsi tersebut meliputi tampilan pesan awal, proses kalibrasi dengan progress bar, instruksi menyiapkan sensor, tampilan pengukuran detak jantung lengkap dengan animasi dan peringatan jika sensor bermasalah, serta ringkasan statistik hasil pengukuran. Selain itu, ada juga tampilan peringatan timeout jika pengukuran gagal selesai tepat waktu. Semua tampilan ini membantu pengguna memantau kondisi alat secara real-time dengan jelas dan mudah dipahami.

c) Inisialisasi BLE.

```
// ===== KONFIGURASI BLUETOOTH (BLE) =====
BLEServer* pServer = NULL;
BLECharacteristic* pCharacteristic = NULL;
bool deviceConnected = false;
#define SERVICE_UUID        "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"
```

Gambar 4.7 Konfigurasi BLE

Pada Gambar 4.7 merupakan konfigurasi dasar untuk mengaktifkan komunikasi *Bluetooth Low Energy (BLE)* pada ESP32. Objek pServer digunakan untuk membuat server BLE, sementara pCharacteristic digunakan untuk mendefinisikan karakteristik data yang akan dikirim, dalam hal ini nilai detak jantung. Variabel *deviceConnected* berfungsi sebagai penanda status koneksi antara ESP32 dan perangkat penerima. Selain itu, SERVICE_UUID dan CHARACTERISTIC_UUID adalah identitas unik yang digunakan untuk mengatur layanan dan karakteristik BLE sesuai standar UART-over-BLE, sehingga perangkat penerima dapat mengenali dan membaca data yang dikirimkan dengan benar.

```
// --- Callback BLE (Sama) ---
class MyServerCallbacks : public BLEServerCallbacks { void
onConnect(BLEServer* pServer) { deviceConnected = true;
Serial.println("BLE terhubung."); } void onDisconnect(BLEServer*
pServer) { deviceConnected = false; Serial.println("BLE terputus.");
BLEDevice::startAdvertising(); } };
class MyCharacteristicCallbacks : public BLECharacteristicCallbacks
{ void onWrite(BLECharacteristic* pCharacteristic) { String rxValue
= String(pCharacteristic->getValue().c_str()); rxValue.trim(); if
(rxValue.equalsIgnoreCase("start")) { startMeasurement("BLE"); } }
};
```

Gambar 4.8 Callback BLE

Pada Gambar 4.8 merupakan kode agar perangkat ESP32 bisa terhubung dan dikendalikan lewat Bluetooth. Ada dua bagian utama dalam kode ini. Bagian pertama mengatur saat Bluetooth tersambung atau terputus. Ketika HP atau perangkat lain berhasil terhubung ke ESP32 lewat Bluetooth, akan muncul tulisan “BLE terhubung” di komputer. Jika koneksi terputus, maka akan muncul pesan “BLE terputus” dan ESP32 akan bersiap untuk disambungkan kembali. Bagian

kedua dari kode ini membaca perintah yang dikirim lewat Bluetooth. Jika pengguna mengirim kata “start” dari aplikasi Bluetooth di HP, maka ESP32 akan langsung memulai proses pengukuran detak jantung. Jadi, dengan kode ini, ESP32 bisa dikendalikan dari jarak jauh hanya dengan mengirim perintah lewat Bluetooth.

- Pengolahan Data BPM

```
// ===== PENGATURAN APLIKASI & PENGUKURAN DATA BPM =====
const int MAX_BPM_DATA = 20;
int bpmData[MAX_BPM_DATA];
int dataCount = 0;

const int BPM_FILTER_WINDOW_SIZE = 5;
float bpmHistory[BPM_FILTER_WINDOW_SIZE];
int bpmHistoryIndex = 0;
bool bpmFilterPrimed = false;

int medianBPM = 0;
float trimmedMeanBPM = 0.0;
float simpleAverageBPM = 0.0;

enum AppState { IDLE, CALIBRATING_THRESHOLD, PREPARING_SENSOR, MEASURING, CALCULATING_STATS, DISPLAYING_RESULT, TIMEOUT_STATE };
AppState currentAppState = IDLE;

unsigned long measurementStartTime = 0;
const unsigned long measurementTimeoutDuration = 40000; // 40 detik
unsigned long displayResultStartTime = 0;
const unsigned long displayResultDuration = 8000; // 8 detik
unsigned long lastDataPointTime = 0;
const unsigned long stuckDetectionTimeout = 10000; // 10 detik
unsigned long preparingSensorStartTime = 0;
const unsigned long preparingSensorDuration = 3000; // 3 detik
```

Gambar 4.9 Pengolahan Data BPM

Pada Gambar 4.9 berfungsi untuk mengatur proses pengukuran dan pengolahan data detak jantung dalam aplikasi monitoring. Pertama, data detak jantung yang terkumpul disimpan dalam sebuah array dengan kapasitas maksimal 20 data. Untuk menghasilkan pembacaan yang lebih stabil dan akurat, digunakan teknik penyaringan data dengan menyimpan beberapa nilai detak jantung terakhir, lalu menghitung nilai rata-rata, median, dan trimmed mean dari data tersebut. Selanjutnya, kode ini juga mengatur berbagai kondisi atau tahap aplikasi melalui sebuah variabel status, mulai dari keadaan diam (idle), kalibrasi sensor, persiapan sensor, proses pengukuran, penghitungan statistik, hingga menampilkan hasil pengukuran dan menangani kondisi waktu habis (timeout). Selain itu, kode ini

menetapkan batas waktu pada tiap tahap, seperti durasi pengukuran selama 40 detik, tampilan hasil selama 8 detik, dan persiapan sensor selama 3 detik, agar proses pengukuran berjalan secara teratur dan efisien. Dengan begitu, aplikasi dapat menjalankan monitoring detak jantung secara sistematis dan memberikan hasil yang lebih akurat kepada pengguna.

- Fungi Setup()

```
// Fungsi setup()
void setup() {
  Wire.begin(15, 2); Serial.begin(115200); delay(1000);

  pinMode(blinkPin_amped, OUTPUT); pinMode(fadePin_amped, OUTPUT);
  digitalWrite(blinkPin_amped, LOW); digitalWrite(fadePin_amped, LOW);

  if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) { Serial.println("OLED gagal"); while (true); }

  pulseSensor.analogInput(PULSE_INPUT);
  dynamicThreshold = INITIAL_THRESHOLD;
  pulseSensor.setThreshold(dynamicThreshold);
  if (!pulseSensor.begin()) {
    Serial.println("PulseSensor gagal!");
    display.clearDisplay(); display.setCursor(0,0); display.println("PulseSensor ERROR!"); display.display();
    while (true);
  }
  Serial.print("Threshold awal: "); Serial.println(dynamicThreshold);

  BLEDevice::init("DetakJantungV3");
  pServer = BLEDevice::createServer(); pServer->setCallbacks(new MyServerCallbacks());
  BLEService* pService = pServer->createService(SERVICE_UUID);
  pCharacteristic = pService->createCharacteristic(CHARACTERISTIC_UUID, BLECharacteristic::PROPERTY_READ | BLECharacteristic::PROPERTY_NOTIFY | BLECharacteristic::PROPERTY_WRITE);
  pCharacteristic->addDescriptor(new BLE2902()); pCharacteristic->setCallbacks(new MyCharacteristicCallbacks());
  pService->start();
  BLEAdvertising* pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID); pAdvertising->setScanResponse(true);
  BLEDevice::startAdvertising();
  Serial.println("BLE siap. Nama: DetakJantungV3");

  showInitialMessageOLED();
  currentAppState = IDLE;
}
```

Gambar 4.10 Fungsi Setup()

Fungsi setup() Menyiapkan semua komponen utama yaitu OLED yang akan menampilkan informasi secara langsung di layar, serta sensor detak jantung PulseSensor yang bertugas mendeteksi sinyal denyut nadi dari pengguna., dan BLE agar perangkat siap digunakan untuk mendeteksi dan mengirimkan detak jantung melalui Bluetooth ke aplikasi Serial Bluetooth Terminal.

- Fungi loop()

```
// Fungsi loop()
void loop() {
  if (Serial.available() && (currentAppState == IDLE || currentAppState == DISPLAYING_RESULT || currentAppState == TIMEOUT_STATE )) {
    String command = Serial.readStringUntil('\n'); command.trim();
    if (command.equalsIgnoreCase("start")) { startMeasurement("Serial"); }
  }
  updateFadeLEDs(); // Selalu update status LED fade

  switch (currentAppState) {
    case IDLE:
      // Tidak ada update OLED khusus di loop, sudah ditampilkan saat masuk state
      break;

    case CALIBRATING_THRESHOLD:
      showCalibratingMessageOLED(); // Update progress bar di OLED
      if (millis() - calibrationStartTime >= calibrationDuration) { // Cek >= agar pasti selesai
        Serial.print("Kalibrasi Selesai. Min: "); Serial.print(minRawSignalDuringCal);
        Serial.print(", Max: "); Serial.println(maxRawSignalDuringCal);
        if ((maxRawSignalDuringCal > minRawSignalDuringCal) && (maxRawSignalDuringCal - minRawSignalDuringCal) >= MIN_SIGNAL_SWING_FOR_CALIBRATION) {
          int calculatedNewThreshold = minRawSignalDuringCal + (int)((maxRawSignalDuringCal - minRawSignalDuringCal) * 0.60);
          if(calculatedNewThreshold < 50) calculatedNewThreshold = 50;
          if(calculatedNewThreshold > MAX_CALIBRATED_THRESHOLD) calculatedNewThreshold = MAX_CALIBRATED_THRESHOLD; // Batas atas baru
          dynamicThreshold = calculatedNewThreshold;
          pulseSensor.setThreshold(dynamicThreshold);
          Serial.print("Threshold baru: "); Serial.println(dynamicThreshold);
        } else {
          Serial.println("Kalibrasi gagal, pakai threshold lama.");
          pulseSensor.setThreshold(dynamicThreshold); // Pastikan threshold lama (atau initial) tetap dipakai
          Serial.print("Threshold tetap: "); Serial.println(dynamicThreshold);
        }
        currentAppState = PREPARING_SENSOR;
        preparingSensorStartTime = millis();
        showPreparingSensorOLED();
      } else { // Selama kalibrasi, baca sampel mentah
        int currentSample = pulseSensor.getLatestSample();
        if (currentSample < minRawSignalDuringCal) minRawSignalDuringCal = currentSample;
        if (currentSample > maxRawSignalDuringCal) maxRawSignalDuringCal = currentSample;
      }
```

Gambar 4.11 Fungsi Loop()

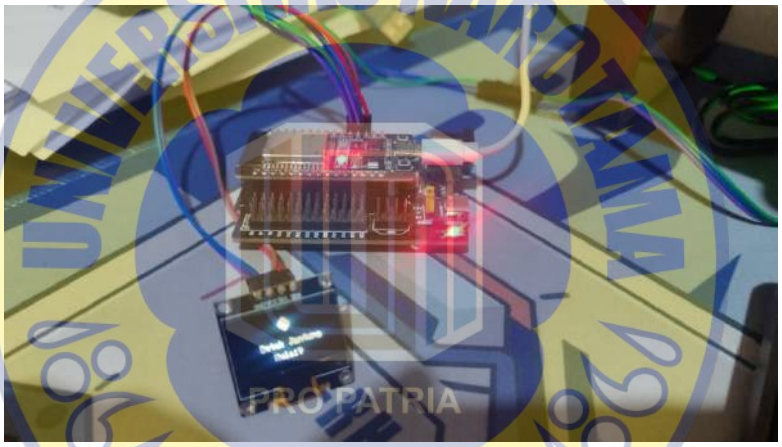
Fungsi loop() inti dari sistem pemantauan detak jantung yang mengatur alur kerja berdasarkan status sistem. Dimulai dari menerima perintah "start", melakukan kalibrasi sensor, mempersiapkan, hingga mengukur BPM secara real-time. Data BPM difilter, dihitung statistiknya, ditampilkan di OLED, dan dikirim via BLE jika tersedia. Fungsi ini juga menangani kondisi timeout serta memperbarui LED sebagai indikator visual, memastikan proses berjalan teratur dan responsif

4.2 Pengujian Alat

Pada tahap ini, seluruh komponen yang dipakai sebagai alat serta bahan dalam sistem akan melalui tahap pengujian dan pemeriksaan yang dilakukan secara berkala untuk memastikan bahwa setiap fungsi dari komponen tersebut berjalan dengan baik serta sesuai dengan yang diinginkan. Pemeriksaan ini mencakup berbagai aspek teknis, mulai dari kestabilan sinyal sensor, respons mikrokontroler terhadap input data, kejelasan tampilan informasi pada layar OLED, hingga

kelancaran proses komunikasi data antar perangkat. Pengecekan rutin ini sangat penting dilakukan guna meminimalisir kemungkinan terjadinya kerusakan, gangguan, atau malfungsi yang dapat menghambat kinerja sistem secara keseluruhan, terutama pada bagian-bagian vital yang memiliki peran krusial dalam operasional perangkat. Serta melakukan perbandingan pengujian alat dengan alat monitoring detak jantung lain.

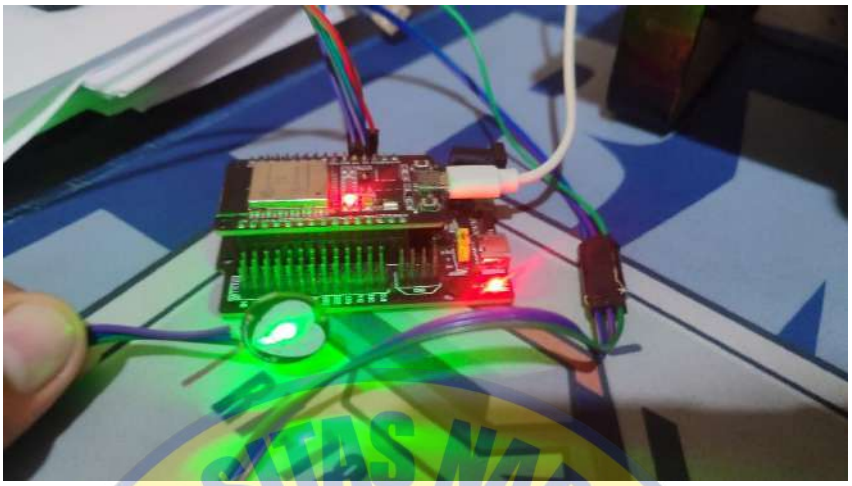
- Pengujian OLED



Gambar 4.12 Pengujian OLED

Pada Gambar 4.12 merupakan pengujian secara terhubung *Mikrokontroler* ESP32 dengan OLED pin VCC OLED disambungkan ke pin 3.3V pada ESP32, Pin GND disambungkan ke pin GND pada ESP32, pin SDA OLED disambungkan ke GPIO 15 dan pin SCL OLED ke GPIO 2. Tujuan diadakan nya uji coba pada Gambar 4.9 adalah untuk mengetahui apakah OLED berfungsi dengan baik.

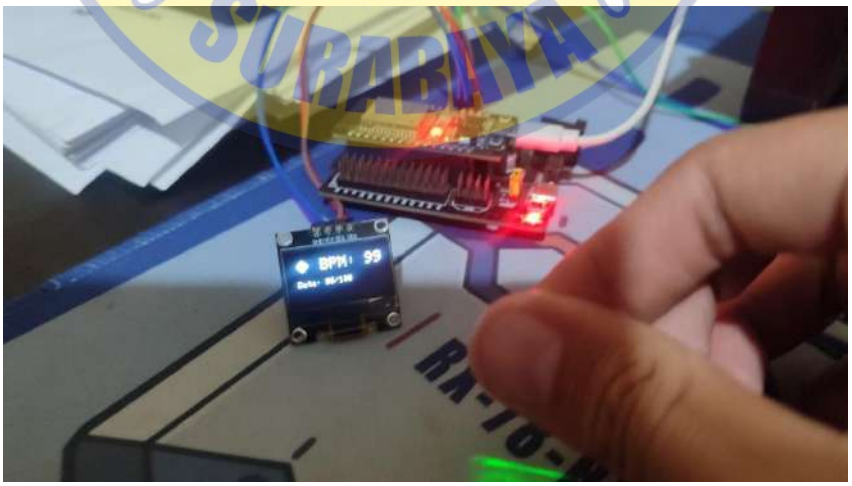
- Pengujian Pulse Sensor



Gambar 4.13 Pengujian Pulse Sensor

Pada Gambar 4.13 merupakan pengujian secara terhubung Mikrokontroler ESP32 dengan *Pulse Sensor* Pin VCC pada *Pulse Sensor* disambungkan ke pin 3.3V pada ESP32,.Pin GND disambungkan ke pin GND pada, Pin Signal disambungkan ke GPIO 4.

- Pengujian Pengambilan Data Detak Jantung



Gambar 4.14 Pengujian Pengambilan Data Detak Jantung

Pada Gambar 4.14 merupakan pengujian pengambilan data detak jantung. Pulse Sensor mengambil data detak jantung kemudian dikirim ke ESP32 guna di proses setelah itu ESP32 mengirimkan hasil data detak jantung ke OLED dan Serial Bluetooth Terminal menggunakan *Bluetooth Low Energy (BLE)*.

- Pengujian Koneksi BLE dan Pengiriman Data BLE

Gambar 4.15 Pengujian Koneksi BLE dan Pengiriman Data BLE

Pada Gambar 4.15 merupakan pengujian koneksi BLE dan pengiriman data BLE Saat koneksi berhasil terjalin, sistem secara otomatis mengirimkan data denyut jantung yang telah dihitung dalam satuan *BPM (Beats Per Minute)* melalui karakteristik BLE yang telah dikonfigurasi dengan fitur notifikasi. Setiap kali detak jantung terdeteksi, nilai BPM diperbarui dan dikirim ke Serial Bluetooth Terminal dalam bentuk teks, misalnya "BPM: 78". Data ini kemudian ditampilkan secara real-time di layar aplikasi Serial Bluetooth Terminal, memungkinkan pengguna buat mengawasi kondisi detak jantung mereka.

- Pengujian Alat dan Perbandingan Alat

Tahap selanjutnya adalah menguji tingkat akurasi sensor, yang dilakukan dengan cara membandingkan perangkat hasil rancangan penelitian ini dengan alat pemantau detak jantung *Smart Watch Laxsafit*.

Tabel 4.1 Indikator Detak Jantung

Detak Jantung	**Kategori**
<60 BPM	Rendah
61-100 BPM	Normal
>100	Tinggi

Dalam pengujian terhadap akurasi sensor, penulis menghitung persentase *error* dan tingkat akurasi dengan membandingkan hasil pembacaan sensor terhadap alat konvensional sebagai acuan (Isyanto et al., 2022). Perhitungan ini dilakukan menggunakan rumus persamaan berikut:

$$Error(\%) = (\underline{A - B})\times 100\% \quad \text{.....(1)}$$

A

$$Akurasi(\%) = 100\% - Error(\%) \quad \text{.....(2)}$$

Keterangan :

A = Hasil dari pengukuran menggunakan smartwatch

B = Hasil dari pengukuran Pulse Sensor

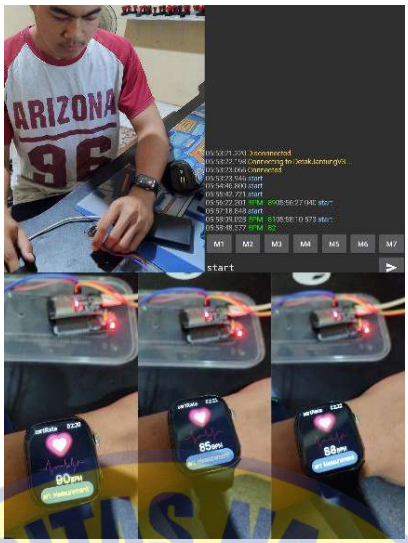
Tabel 4.2 Tabel Hasil Pengujian Detak Jantung

NO	Nama dan Umur	Detak Jantung Pulse Sensor (BPM)	Detak Jantung Smart Watch Laxsafit (BPM)	Selisih	Error (%)	Akurasi (%)
1	Yoel (15 Tahun)	77	83	6	7,23%	92,77%
2	Yoel (15 Tahun)	83	84	1	1,19%	98,81%
3	Yoel (15 Tahun)	79	84	5	5,95%	94,05%
4	Aldy (21 Tahun)	89	90	1	1,11%	98,89%
5	Aldy (21 Tahun)	81	85	4	4,71%	95,29%
6	Aldy (21 Tahun)	82	88	6	6,82%	93,18%
7	Tina (46 Tahun)	83	84	1	1.19%	98,81%
8	Tina (46 Tahun)	79	84	5	5,95%	94,05%
9	Tina (46 Tahun)	78	85	7	9,24%	91,76%
RATA - RATA					**4,83%**	**95,17%**

Mengacu pada Tabel 4.2 yang memuat hasil pengujian *Pulse Sensor* dan *Smartwatch Laxsafit* dalam membaca detak jantung, dari 9 kali percobaan diperoleh rata-rata *error* sebesar 4,83% dan tingkat *akurasi* rata-rata sebesar 95,17%. Nilai *error* tertinggi tercatat sebesar 9,24%, sedangkan yang terendah mencapai 1,11%. Sementara itu, akurasi tertinggi berada di angka 98,89%, dan akurasi terendah sebesar 91,76%.

Gambar 4.16 Pengujian Detak Jantung 1

Berdasarkan Gambar 4.16, ditunjukkan proses pengujian alat *Telemedicine Monitoring Detak Jantung* yang dilakukan untuk mengevaluasi kinerja sistem secara menyeluruh. Pengujian ini dilakukan oleh subjek bernama Yoel, seorang remaja berusia 15 tahun, sebagai pengguna uji coba Percobaan dilakukan sebanyak tiga kali untuk memastikan konsistensi hasil dan kestabilan sistem.



Gambar 4.17 Pengujian Detak Jantung 2

Berdasarkan Gambar 4.17, ditunjukkan proses pengujian alat *Telemedicine Monitoring Detak Jantung*. Pengujian ini dilakukan oleh subjek bernama Aldy, seorang berusia 21 tahun, sebagai pengguna uji coba Percobaan dilakukan sebanyak tiga kali untuk memastikan konsistensi hasil dan kestabilan sistem.

Gambar 4.18 Pengujian Detak Jantung 3

Berdasarkan Gambar 4.18, Pengujian ini dilakukan oleh subjek bernama Tina , seorang berusia 46 tahun, sebagai pengguna uji coba Percobaan dilakukan sebanyak tiga kali untuk memastikan konsistensi hasil dan kestabilan sistem.