TUGAS PEMROGRAMAN KONTROLLER

Dosen : Ahmad Radhy, S.Si., M.Si

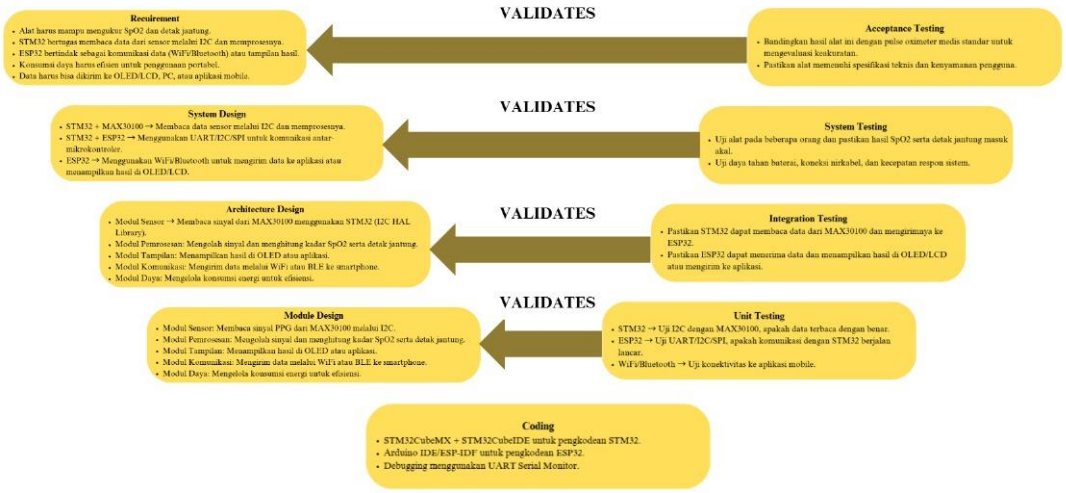
” Desain Alat Pemantauan Kadar Oksigen Darah (SpO2) Berbasis Microcontroller.”

Disusun Oleh :

Muhammad Yusron Maskur	**(2042231030)**
Rahmat	**(2042231050)**
Agus Wedi	**(2042231066)**

PRODI D4 TEKNOLOGI REKAYASA INSTRUMENTASI
DEPARTEMEN TEKNIK INSTRUMENTASI
FAKULTAS VOKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER

2025


Requirement
• Alat harus mampu mengukur SpO2 dan detak jantung.
• STM32 bertugas membaca data dari sensor melalui I2C dan memprosesnya.
• ESP32 bertindak sebagai komunikasi data (WiFi/Bluetooth) atau tampilan hasil.
• Konsumsi daya harus efisien untuk penggunaan portabel.
• Data harus bisa dikirim ke OLED/LCD, PC, atau aplikasi mobile.
VALIDATES
Acceptance Testing
• Bandingkan hasil alat ini dengan pulse oximeter medis standar untuk mengevaluasi keakuratan.
• Pastikan alat memenuhi spesifikasi teknis dan kenyamanan pengguna.
System Design
• STM32 + MAX30100 → Membaca data sensor melalui I2C dan memprosesnya.
• STM32 + ESP32 → Menggunakan UART/I2C/SPI untuk komunikasi antar-mikrokontroler.
• ESP32 → Menggunakan WiFi/Bluetooth untuk mengirim data ke aplikasi atau menampilkan hasil di OLED/LCD.
VALIDATES
System Testing
• Uji alat pada beberapa orang dan pastikan hasil SpO2 serta detak jantung masuk akal.
• Uji daya tahan baterai, koneksi nirkabel, dan kecepatan respon sistem.
Architecture Design
• Modul Sensor → Membaca sinyal dari MAX30100 menggunakan STM32 (I2C HAL Library).
• Modul Pemrosesan: Mengolah sinyal dan menghitung kadar SpO2 serta detak jantung.
• Modul Tampilan: Menampilkan hasil di OLED atau aplikasi.
• Modul Komunikasi: Mengirim data melalui WiFi atau BLE ke smartphone.
• Modul Daya: Mengelola konsumsi energi untuk efisiensi.
VALIDATES
Integration Testing
• Pastikan STM32 dapat membaca data dari MAX30100 dan mengirimnya ke ESP32.
• Pastikan ESP32 dapat menerima data dan menampilkan hasil di OLED/LCD atau mengirim ke aplikasi.
Module Design
• Modul Sensor: Membaca sinyal PPG dari MAX30100 melalui I2C.
• Modul Pemrosesan: Mengolah sinyal dan menghitung kadar SpO2 serta detak jantung.
• Modul Tampilan: Menampilkan hasil di OLED atau aplikasi.
• Modul Komunikasi: Mengirim data melalui WiFi atau BLE ke smartphone.
• Modul Daya: Mengelola konsumsi energi untuk efisiensi.
VALIDATES
Unit Testing
• STM32 → Uji I2C dengan MAX30100, apakah data terbaca dengan benar.
• ESP32 → Uji UART/I2C/SPI, apakah komunikasi dengan STM32 berjalan lancar.
• WiFi/Bluetooth → Uji konektivitas ke aplikasi mobile.
Coding
• STM32CubeMX + STM32CubeIDE untuk pengkodean STM32.
• Arduino IDE/ESP-IDF untuk pengkodean ESP32.
• Debugging menggunakan UART Serial Monitor.

Latar Belakang

Kadar oksigen dalam darah (SpO_2) merupakan indikator penting dalam pemantauan kesehatan, terutama bagi pasien dengan gangguan pernapasan seperti asma, pneumonia, dan penyakit paru obstruktif kronis (PPOK). Pemantauan kadar oksigen secara real-time dapat membantu tenaga medis dalam mengambil keputusan cepat terhadap kondisi pasien.

Saat ini, alat ukur kadar oksigen darah seperti oximeter sudah banyak digunakan, namun kebanyakan masih berbasis perangkat konvensional. Dengan perkembangan teknologi embedded system, perangkat pemantauan SpO_2 berbasis microcontroller dapat dikembangkan untuk meningkatkan efisiensi, portabilitas, dan konektivitas data ke sistem medis.

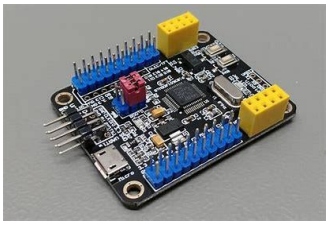
Alat pemantauan SpO2 ini menggunakan sensor MAX30100 atau MAX30102 untuk mendeteksi kadar oksigen dalam darah. Data sensor dikirim ke mikrokontroler STM32 atau ESP32 melalui komunikasi I2C. Setelah diproses, hasil pengukuran ditampilkan pada layar OLED atau LCD dan dapat dikirim ke perangkat lain melalui komunikasi Bluetooth/WiFi. Jika kadar oksigen dalam darah turun di bawah ambang batas, alat akan mengaktifkan alarm sebagai peringatan.

Tujuan

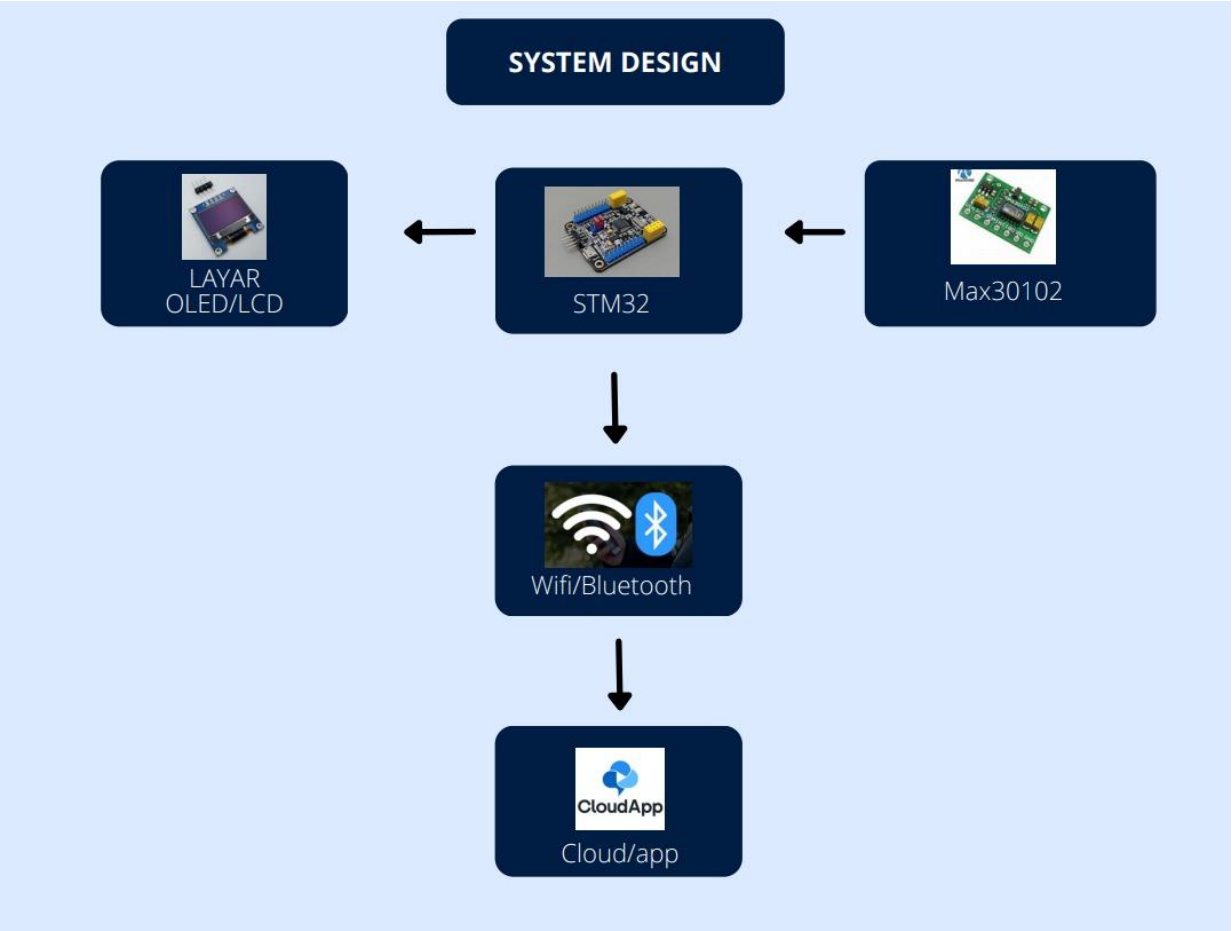
Proyek ini bertujuan untuk merancang dan mengembangkan perangkat pemantauan kadar oksigen darah (SpO_2) berbasis microcontroller, yang mampu Mengukur kadar oksigen dalam darah secara real-time. Menampilkan hasil pengukuran dalam bentuk digital pada layar LCD atau OLED. Menggunakan sensor pulse oximeter (misalnya MAX30100 atau MAX30102). Memiliki fitur penyimpanan data atau konektivitas IoT untuk pemantauan jarak jauh.

Requirements

Komponen	Spesifikasi
Microcontroller	STM32
Sensor SpO_2	MAX30102
Layar Tampilan	OLED 0.96” atau LCD 1.8”
Sumber Daya	Baterai Li-ion 3.7V 1000mAh
Komunikasi Data	UART, I2C, atau SPI
Casing Perangkat	Plastik ABS atau silikon medis



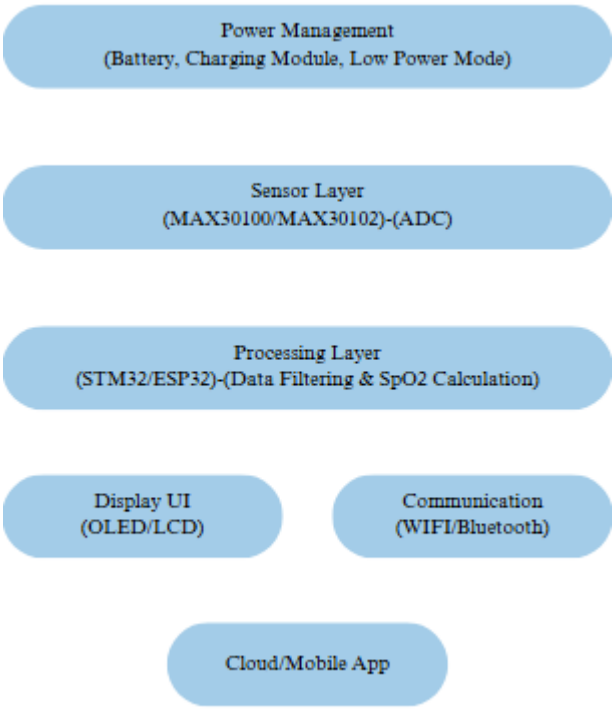
System Design



Alur Proses Diagram System Design

1. Sensor SpO_2 (MAX30102)
   - Sensor ini mendeteksi kadar oksigen dalam darah (SpO_2) serta detak jantung (BPM).
   - Data diambil melalui komunikasi I^2C dan dikirim ke microcontroller untuk diproses.
2. Microcontroller (STM32 / ESP32)
   - Micrrocontroller menerima data mentah dari sensor.
   - Data ini kemudian diolah untuk memastikan validitasnya dan dikonversi ke format yang dapat dibaca oleh pengguna.
   - Jika SpO_2 rendah (<90%), sistem dapat mengaktifkan alarm atau notifikasi.
3. Layar OLED/LCD
   - Setelah data diproses, microcontroller menampilkannya pada layar OLED/LCD.
   - Informasi yang ditampilkan biasanya berupa persentase SpO_2 dan detak jantung (BPM) pengguna.
4. Modul Komunikasi (WiFi/Bluetooth)
   - Untuk memungkinkan pemantauan jarak jauh, data dapat dikirim melalui WiFi atau Bluetooth.
   - Data ini bisa dikirim ke smartphone, laptop, atau sistem cloud untuk disimpan atau dianalisis lebih lanjut.
5. Cloud/App
   - Data dari sensor yang dikirim melalui modul komunikasi dapat diakses di aplikasi atau cloud.
   - Pengguna atau tenaga medis bisa memantau kondisi pasien secara real-time melalui aplikasi yang terhubung.

Arsitechture Disign



Alur Proses Diagram Arsitektur Sistem

1. Pengguna memasang sensor SpO_2 pada jari, sensor MAX30100/MAX30102 mulai membaca data.
2. Data sensor dikirim ke microcontroller melalui komunikasi I2C/SPI.
3. Microcontroller memproses data menggunakan algoritma filtering dan perhitungan SpO_2/BPM.
4. Hasil pengukuran ditampilkan di layar OLED/LCD dalam waktu nyata.
5. Jika SpO_2 rendah (< 90%), sistem akan memberikan peringatan menggunakan buzzer/LED.
6. IoT data dikirim ke aplikasi mobile/cloud untuk pemantauan jarak jauh.

Module Disign

Nama Modul	Fungsi
Modul Sensor & Akuisisi Data	Membaca data kadar oksigen (SpO_2) dan denyut jantung dari sensor MAX30102.
Modul Pemrosesan Data	Mengolah data sensor dengan filtering dan perhitungan SpO_2/BPM.
Modul Tampilan UI	Menampilkan data hasil pemantauan di layar OLED/LCD.
Modul Notifikasi & Alarm	Mengaktifkan buzzer atau LED jika SpO_2 turun di bawah batas normal.
Modul Komunikasi (Opsional - IoT)	Mengirim data ke aplikasi mobile atau cloud menggunakan MQTT/HTTP.
Modul Manajemen Daya	Mengatur konsumsi daya untuk efisiensi baterai.

Coding

```
#include <Wire.h>
#include "MAX30102.h"
#include "SSD1306Wire.h"

// Inisialisasi sensor dan layar OLED
MAX30102 sensor;
SSD1306Wire display(0x3C, SDA, SCL);

void setup() {
  Serial.begin(115200);
  Wire.begin();

  // Inisialisasi sensor MAX30102
  if (!sensor.begin()) {
    Serial.println("Sensor MAX30102 tidak terdeteksi!");
    while (1);
  }

  // Inisialisasi OLED
  display.init();
  display.flipScreenVertically();
}
```

```
void loop() {
    int spo2, heartRate;

    if (sensor.check() == true) {
        spo2 = sensor.getSpO2();
        heartRate = sensor.getHeartRate();

        // Tampilkan hasil di Serial Monitor
        Serial.print("SpO2: "); Serial.print(spo2);
        Serial.print("% | BPM: "); Serial.println(heartRate);

        // Tampilkan di OLED
        display.clear();
        display.setFont(ArialMT_Plain_16);
        display.drawString(10, 10, "SpO2: " + String(spo2) +
"%");
        display.drawString(10, 30, "BPM: " +
String(heartRate));
        display.display();

        // Alarm jika SpO2 rendah
        if (spo2 < 90) {
            tone(5, 1000, 500);
        }
    }
    delay(1000);
}
```

Unit Testing

```
#include <Wire.h>
#include "MAX30102.h"
#include "SSD1306Wire.h"
#include <assert.h>

// Inisialisasi sensor dan layar OLED
MAX30102 sensor;
SSD1306Wire display(0x3C, SDA, SCL);

void setup() {
  Serial.begin(115200);
  Wire.begin();

  // Inisialisasi sensor MAX30102
  assert(sensor.begin() == true);
  Serial.println("Sensor MAX30102 berhasil diinisialisasi.");

  // Inisialisasi OLED
  display.init();
  display.flipScreenVertically();
}
```

```
void loop() {
  int spo2, heartRate;

  if (sensor.check() == true) {
    spo2 = sensor.getSpO2();
    heartRate = sensor.getHeartRate();

    // Unit Test: Pastikan nilai yang dibaca valid
    assert(spo2 >= 0 && spo2 <= 100);
    assert(heartRate > 30 && heartRate < 200);

    Serial.print("SpO2: "); Serial.print(spo2);
    Serial.print("%  | BPM: "); Serial.println(heartRate);

    // Tampilkan di OLED
    display.clear();
    display.setFont(ArialMT_Plain_16);
    display.drawString(10, 10, "SpO2: " + String(spo2) + "%");
    display.drawString(10, 30, "BPM: " + String(heartRate));
    display.display();

    // Unit Test: Pastikan tampilan OLED berfungsi
    assert(display.getWidth() > 0 && display.getHeight() > 0);

    // Alarm jika SpO2 rendah
    if (spo2 < 90) {
      tone(5, 1000, 500);
      // Unit Test: Pastikan alarm berbunyi ketika SpO2 rendah
      Serial.println("ALARM: SpO2 rendah!");
    }
  }
  delay(1000);
```

Intergration Testing

```
#include <Wire.h>
#include "MAX30102.h"
#include "SSD1306Wire.h"
#include <assert.h>

// Inisialisasi sensor dan layar OLED
MAX30102 sensor;
SSD1306Wire display(0x3C, SDA, SCL);

void setup() {
    Serial.begin(115200);
    Wire.begin();

    // Integration Test: Pastikan sensor MAX30102 dapat diinisialisasi
    assert(sensor.begin() == true);
    Serial.println("[TEST] Sensor MAX30102 berhasil diinisialisasi.");

    // Integration Test: Pastikan OLED dapat diinisialisasi
    display.init();
    display.flipScreenVertically();
    assert(display.getWidth() > 0 && display.getHeight() > 0);
    Serial.println("[TEST] OLED berhasil diinisialisasi.");
}

void loop() {
    int spo2, heartRate;

    if (sensor.check() == true) {
        spo2 = sensor.getSpO2();
        heartRate = sensor.getHeartRate();
```

```
    // Integration Test: Pastikan nilai yang dibaca dari sensor valid
    assert(spo2 >= 0 && spo2 <= 100);
    assert(heartRate > 30 && heartRate < 200);
    Serial.println("[TEST] Data sensor valid.");

    Serial.print("SpO2: "); Serial.print(spo2);
    Serial.print("% | BPM: "); Serial.println(heartRate);

    // Integration Test: Pastikan OLED dapat menampilkan data
    display.clear();
    display.setFont(ArialMT_Plain_16);
    display.drawString(10, 10, "SpO2: " + String(spo2) + "%");
    display.drawString(10, 30, "BPM: " + String(heartRate));
    display.display();
    assert(display.getWidth() > 0 && display.getHeight() > 0);
    Serial.println("[TEST] Data ditampilkan di OLED.");

    // Integration Test: Pastikan buzzer berbunyi saat SpO2 rendah
    if (spo2 < 90) {
      tone(5, 1000, 500);
      Serial.println("[TEST] ALARM: SpO2 rendah!");
    }
  }

  // Integration Test: Pastikan komunikasi I2C berjalan normal
  assert(Wire.available() >= 0);
  Serial.println("[TEST] Komunikasi I2C berjalan dengan baik.");

  delay(1000);
}
```

System testing

```
#include <Wire.h>
#include "MAX30102.h"
#include "SSD1306Wire.h"
#include <assert.h>

// Inisialisasi sensor dan layar OLED
MAX30102 sensor;
SSD1306Wire display(0x3C, SDA, SCL);

// Variabel untuk sistem testing
unsigned long testStartTime;
bool systemStable = true;

void setup() {
  Serial.begin(115200);
  Wire.begin();

  // System Test: Pastikan sensor MAX30102 dapat diinisialisasi
  assert(sensor.begin() == true);
  Serial.println("[TEST] Sensor MAX30102 berhasil diinisialisasi.");

  // System Test: Pastikan OLED dapat diinisialisasi
  display.init();
  display.flipScreenVertically();
  assert(display.getWidth() > 0 && display.getHeight() > 0);
  Serial.println("[TEST] OLED berhasil diinisialisasi.");

  // Catat waktu mulai pengujian
  testStartTime = millis();
```

```
}

void loop() {
  int spo2, heartRate;

  if (sensor.check() == true) {
    spo2 = sensor.getSpO2();
    heartRate = sensor.getHeartRate();

    // System Test: Pastikan nilai dari sensor berada dalam rentang yang valid
    assert(spo2 >= 0 && spo2 <= 100);
    assert(heartRate > 30 && heartRate < 200);
    Serial.println("[TEST] Data sensor valid.");

    Serial.print("SpO2: "); Serial.print(spo2);
    Serial.print("% | BPM: "); Serial.println(heartRate);

    // System Test: Pastikan OLED dapat menampilkan data
    display.clear();
    display.setFont(ArialMT_Plain_16);
    display.drawString(10, 10, "SpO2: " + String(spo2) + "%");
    display.drawString(10, 30, "BPM: " + String(heartRate));
    display.display();
    assert(display.getWidth() > 0 && display.getHeight() > 0);
    Serial.println("[TEST] Data ditampilkan di OLED.");

    // System Test: Pastikan buzzer berbunyi saat SpO2 rendah
    if (spo2 < 90) {
      tone(5, 1000, 500);
      Serial.println("[TEST] ALARM: SpO2 rendah!");
    }
```

```
    }

    // System Test: Pastikan komunikasi I2C berjalan normal
    assert(Wire.available() >= 0);
    Serial.println("[TEST] Komunikasi I2C berjalan dengan baik.");

    // System Test: Uji stabilitas sistem dalam jangka waktu lama
    if (millis() - testStartTime > 60000) { // Setelah 1 menit
        Serial.println("[TEST] Sistem berjalan stabil selama 1 menit.");
        assert(systemStable == true);
        testStartTime = millis(); // Reset timer
    }

    delay(1000);
}
```

acceptance testing

```
#include <Wire.h>
#include "MAX30102.h"
#include "SSD1306Wire.h"
#include <assert.h>

// Inisialisasi sensor dan layar OLED
MAX30102 sensor;
SSD1306Wire display(0x3C, SDA, SCL);

// Variabel untuk pengujian
unsigned long testStartTime;
bool systemStable = true;

void setup() {
  Serial.begin(115200);
  Wire.begin();

  // Acceptance Test: Sensor harus dapat diinisialisasi
  assert(sensor.begin() == true);
  Serial.println("[ACCEPTANCE TEST] Sensor MAX30102 berhasil diinisialisasi.");

  // Acceptance Test: OLED harus dapat diinisialisasi
  display.init();
  display.flipScreenVertically();
  assert(display.getWidth() > 0 && display.getHeight() > 0);
  Serial.println("[ACCEPTANCE TEST] OLED berhasil diinisialisasi.");

  // Catat waktu mulai pengujian
  testStartTime = millis();
}

void loop() {
  int spo2, heartRate;

  if (sensor.check() == true) {
    spo2 = sensor.getSpO2();
    heartRate = sensor.getHeartRate();

    // Acceptance Test: Pastikan nilai dari sensor berada dalam rentang yang valid
    assert(spo2 >= 0 && spo2 <= 100);
    assert(heartRate > 30 && heartRate < 200);
    Serial.println("[ACCEPTANCE TEST] Data sensor valid.");

    Serial.print("SpO2: "); Serial.print(spo2);
    Serial.print("% | BPM: "); Serial.println(heartRate);
```

```
      // Acceptance Test: Data harus ditampilkan di OLED dalam format yang jelas
      display.clear();
      display.setFont(ArialMT_Plain_16);
      display.drawString(10, 10, "SpO2: " + String(spo2) + "%");
      display.drawString(10, 30, "BPM: " + String(heartRate));
      display.display();
      assert(display.getWidth() > 0 && display.getHeight() > 0);
      Serial.println("[ACCEPTANCE TEST] Data ditampilkan di OLED.");

      // Acceptance Test: Alarm berbunyi jika SpO2 rendah
      if (spo2 < 90) {
        tone(5, 1000, 500);
        Serial.println("[ACCEPTANCE TEST] ALARM: SpO2 rendah!");
      }
    }

    // Acceptance Test: Sistem harus responsif dengan update setiap detik
    assert(millis() - testStartTime < 2000);
    Serial.println("[ACCEPTANCE TEST] Sistem responsif.");

    // Acceptance Test: Pastikan sistem berjalan stabil dalam waktu lama
    if (millis() - testStartTime > 3600000) { // Setelah 1 jam
      Serial.println("[ACCEPTANCE TEST] Sistem berjalan stabil selama 1 jam.");
      assert(systemStable == true);
      testStartTime = millis(); // Reset timer
    }

    delay(1000);
}
```