UNIVERSITAS INDONESIA

VitaSense: A Real-Time Health Monitoring Technology

TUGAS AKHIR SISTEM EMBEDDED BIOMEDIK DAN PRAKTIKUM

Surya Sasmita Vranken	**1906382334**
Aisyah Dzakiyah M	**2206060510**
Hasan Abdul Lathif	**2206815503**

FAKULTAS TEKNIK

DEPARTEMEN TEKNIK ELEKTRO

PROGRAM STUDI TEKNIK BIOMEDIK

DEPOK

DESEMBER 2024

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi telah mengubah berbagai aspek kehidupan kita, termasuk dalam bidang kesehatan. Saat ini, kesehatan menjadi salah satu perhatian utama di tengah peningkatan jumlah penyakit kronis. Pemantauan kondisi kesehatan secara real-time menjadi sangat penting untuk meningkatkan kualitas hidup dan mengurangi risiko komplikasi.

Namun, sebagian besar sistem kesehatan tradisional masih mengandalkan kunjungan berkala ke fasilitas kesehatan atau metode pemantauan manual yang sering kali tidak efisien. Hal ini dapat menyebabkan keterlambatan dalam mendeteksi gejala atau perubahan kondisi kesehatan pasien.

Dengan perkembangan teknologi Internet of Things (IoT), kecerdasan buatan (AI), dan sensor biomedis, muncul peluang besar untuk menciptakan solusi yang lebih efektif dan efisien dalam memantau kesehatan secara real-time. Teknologi ini memungkinkan pengumpulan data kesehatan secara kontinu, akurat, dan terintegrasi, yang dapat digunakan untuk mendeteksi anomali lebih awal, memberikan intervensi yang tepat waktu, serta mendukung pengambilan keputusan berbasis data oleh tenaga medis.

VitaSense dapat menjadi solusi atas tantangan ini. VitaSense dirancang untuk menyediakan solusi pemantauan kesehatan yang canggih, melibatkan perangkat wearable, aplikasi berbasis AI, serta sistem notifikasi yang terhubung langsung dengan tenaga medis atau keluarga. Dengan VitaSense, pengguna dapat memantau tanda-tanda vital mereka secara langsung, meningkatkan kesadaran akan kesehatan pribadi, dan mencegah kondisi medis yang lebih serius.

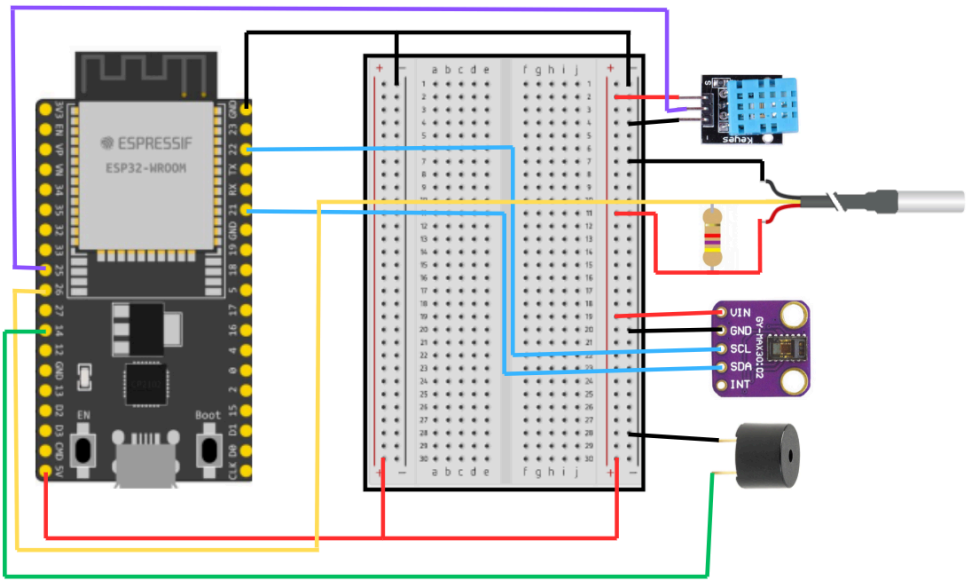
1.2 Tujuan Laporan

Tujuan dari laporan ini adalah untuk meningkatkan akses pemantauan kesehatan yang sederhana dan user-friendly sehingga memungkinkan deteksi dini perubahan kondisi kesehatan secara real-time dan untuk memberikan gambaran fitur pemantauan suhu dan kelembaban untuk memberikan gambaran kondisi lingkungan yang dapat mempengaruhi pasien.

BAB II
SYSTEM DESIGN AND COMPONENTS

2.1 Arsitektur Sistem

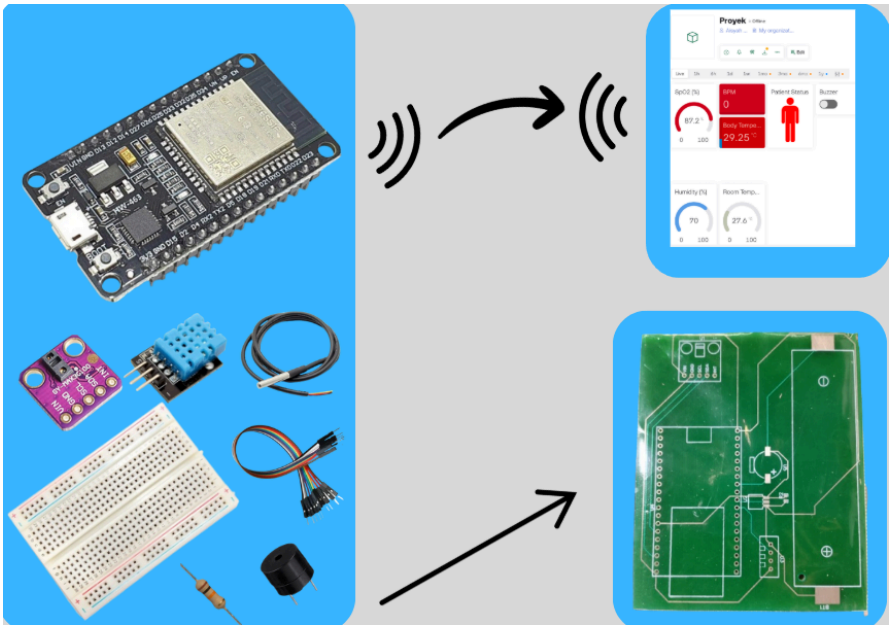
Sistem pemantauan kesehatan real-time ini bertujuan untuk mengukur tanda-tanda vital utama dan kondisi lingkungan sekaligus memanfaatkan IoT untuk pemantauan dan peringatan secara jarak jauh dan real-time. Pada intinya, sistem ini menggunakan mikrokontroler ESP32 untuk pemrosesan data dan komunikasi Wi-Fi dengan platform Blynk IoT. Sensor yang diintegrasikan ke dalam sistem meliputi sensorPulse Oximeter MAX30102, sensor Suhu DS18B20, dan sensor Suhu runagan dan Kelembaban DHT11.



Gambar 1. Diagram dari rangkaian VitaSense

Untuk meningkatkan kemudahan dan kepraktisan sistem, kami mencoba mendesain PCB khusus menggunakan EasyEDA dan casing cetak 3D menggunakan Fusion 360. Namun, karena keterbatasan waktu dan tantangan dalam mendesain PCB, kami menggunakan bread board untuk merangkai proyeknya dengan membuatnya tetap sederhana, teratur, dan enak dipandang dengan menggunakan kabel jumper tunggal. Usaha untuk mendesain 3D printied

casing tidak dapat diselesaikan tepat waktu, tetapi desain sistem memastikan portabilitas dan persiapan untuk perbaikan di masa mendatang.



Gambar 2. Diagram blok dari desain arsitektur

2.2 Komponen Hardware

Mikrokontroler ESP32 dipilih karena fitur-fiturnya yang canggih, termasuk Wi-Fi terintegrasi, pemrosesan yang cepat, pin pin GPIO yang banyak, dan mendukung untuk berbagai protokol komunikasi. Struktur dual-core-nya memastikan penanganan tugas-tugas yang bersamaan secara efisien, seperti perolehan data sensor dan komunikasi waktu nyata dengan platform IoT. Selain itu, kegunaannya dengan Arduino IDE menyederhanakan pemrograman dan debugging.

1. Sensors
    a. MAX30102 Pulse Oximeter Sensor: Sensor ini digunakan untuk mengukur detak jantung pasien (BPM) dan saturasi oksigen darah (SpO2). Sensor ini memanfaatkan teknologi photoplethysmography (PPG) memastikan akurasi dari pembacaannya
    b. DS18B20 Temperature Sensor: Sensor ini digunakan untuk pengukuran suhu tubuh manusia, dengan protokol komunikasi one-wire yang mengurangi kompleksitas kabel. Output digitalnya menghilangkan kebutuhan untuk konversi ADC, sehingga meningkatkan akurasi pengukuran.

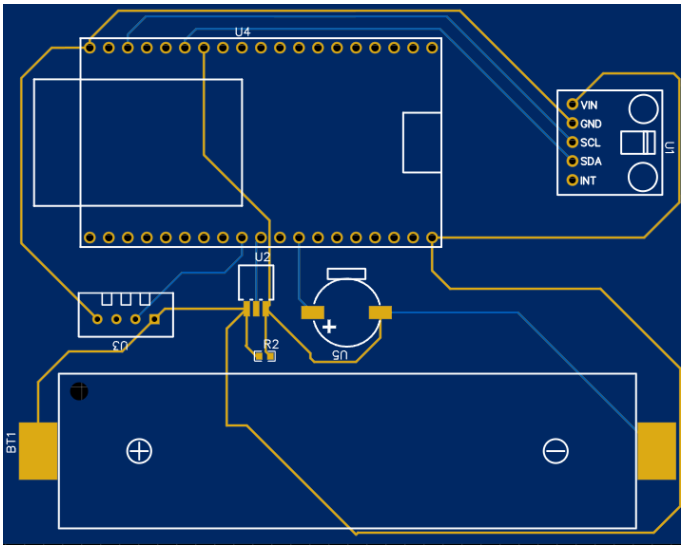
c. DHT11 Humidity and Temperature Sensor: Memantau kondisi ruangan sangat penting untuk kenyamanan pasien atau orang yang lagi sakit. DHT11, ini dapat digunaakn untuk memnataua kondisi ruangan tersebut dengan mengukur suhunya dan kelembaban, meskipun kurang akurat dibandingkan alternatif seperti DHT22, sensor ini kiat pilih karena harganya yang terjangkau.

2. Buzzer
   Buzzer ini kami gunakan sebagai sistem peringatan, mengeluarkan suara ketika pembacaan kesehatan yang tidak normal, seperti SpO2 yang rendah atau suhu tubuh yang tidak normal, terdeteksi. Pemberitahuan langsung ini memastikan masalah penting agara segera ditangani.

3. Komponen Lainnya
   Breadboard digunakan untuk membuat prototipe rangkaian, memastikan desain yang dapat disesuaikan dan dimodifikasi selama pengembangan. Untuk meminimalkan berantakan dan meningkatkan estetika, kabel jumper tunggal digunakan untuk menghubungkan komponen, membuat konfigurasi rangkaian yang rapi. Resistor juga diperlukan sebagai pull-up jika diperlukan untuk I2C dan jalur komunikasi one-wire untuk memastikan transmisi sinyal yang stabil.

2.3 PCB Desain

Untuk meningkatkan portabilitas sistem dan penampilan yang lebih profesional, kami mendesain PCB khusus untuk rangkaiannya menggunakan EasyEDA. PCB ini bertujuan untuk menggabungkan kabel dan mengganti konfigurasi rangkaian pada breadboard. Namun, karena keterbatasan waktu yang tersedia, kami menghadapi tantangan dalam mendesain PCB satu lapis, khususnya dalam merutekan semua koneksi secara efisien tanpa tumpang tindih. Akibatnya, PCB yang suda dicetak, tidak dapat diimplementasikan sepenuhnya dalam proyek akhir. Meskipun demikian, rangkaian breadboard dioptimalkan sebaik mungkin. Pengalaman ini menunjukkan area untuk perbaikan dalam pembuatan PCB di masa mendatang, seperti mengelola waktu dengan baik untuk meperkirakan cetak PCB yang baik tanpa dikejar waktu.



Gambar 3. Desain dari PCB

2.4 Protokol Komunikasi

Sistem ini menggunakan macam-macam protokol komunikasi untuk memastikan transfer data yang lancar antara ESP32 dan sensor:

1. Protokol I2C: Sensor MAX30102 berkomunikasi dengan ESP32 melalui protokol I2C, yang hanya memerlukan dua jalur (SCL dan SDA) untuk sinyal data dan clock, yang berada pada pin 21 dan 22 di ESP32.Protokol ini menyederhanakan pemasangan kabel sekaligus memungkinkan pertukaran data yang cepat.
2. Protokol One-Wire: Sensor DS18B20 menggunakan protokol one-wire, memungkinkan komunikasi dua arah melalui satu jalur data, yang mengurangi jumlah pin GPIO yang diperlukan.
3. Pin GPIO: Sensor DHT11 dan buzzer akan terhubung langsung ke pin GPIO ESP32. Pin GPIO memungkinkan ESP32 menerima data sensor digital dan mengontrol buzzer untuk fitur peringatannya.

2.5 Pertimbangan Power Supply

Sistem ini diberi sumber daya melalui port micro-USB ESP32, yang menyediakan sumber daya 5V yang cukup untuk menjalankan semua komponen. Kebutuhan daya setiap sensor diuji untuk memastikan kompatibilitas dengan output daya ESP32. Resistor pull-up dipasang pada jalur komunikasi untuk menstabilkan sinyal dan mencegah kerusakan data selama pengoperasian.

2.6 Percobaan Desain Casing

Untuk meningkatkan estetika proyek, kami telah mendesain casing khusus menggunakan Fusion 360. Casing ini bertujuan untuk menampung PCB dan sensor, membuat sistem lebih nyaman dan awet. Desain ini menggabungkan celah-celah untuk penempatan setiap sensornya dan buzzer. Akan tetapi, karena keterbatasan waktu, desain casing tidak dapat dicetak 3D untuk prototipe akhir. Namun, rancangan casing ini memberikan pelajaran dalam pengembangan proyek di masa mendatang yang memerlukan cetak 3D printing.

BAB III
HARDWARE AND SOFTWARE IMPLEMENTATION

3.1 Hardware Implementation

Integrasi perangkat keras untuk sistem pemantauan kesehatan real-time disusun dan dijalankan di breadboard dengan teliti untuk memastikan fungsionalitas, memudahkan pemasangan, dan melakukan debugging. Semua komponen perangkat keras dihubungkan ke mikrokontroler ESP32, memanfaatkan pin GPIO untuk komunikasi dan pengaturan. Pemasangan kabel didesain agar bersih dan ringkas dengan menggunakan kabel jumper tunggal, sehingga memungkinkan pengaturan yang terorganisir dan menarik secara visual.

1. Sambungan Power Supply:

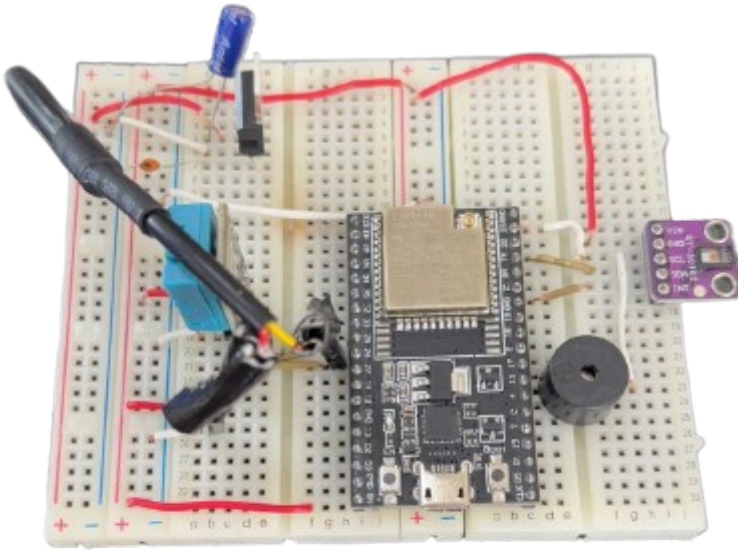
Semua sensor diberi daya melalui pin 5V (Vin) dan GND ESP32 untuk memastikan penyediaan tegangan yang stabil.

- Sensor MAX30102: Terhubung ke 5V dan GND
- Sensor DHT11: Terhubung ke 5V dan GND
- Sensor DS18B20: Terhubung ke 5V dan GND
- Buzzer: Terhubung ke GND

2. Koneksi GPIO pada Komponen:

   Setiap sensor dan buzzer diberi pin GPIO spesifik pada ESP32, dihubungkannya sesuai protokol komunikasi dan fungsinya:

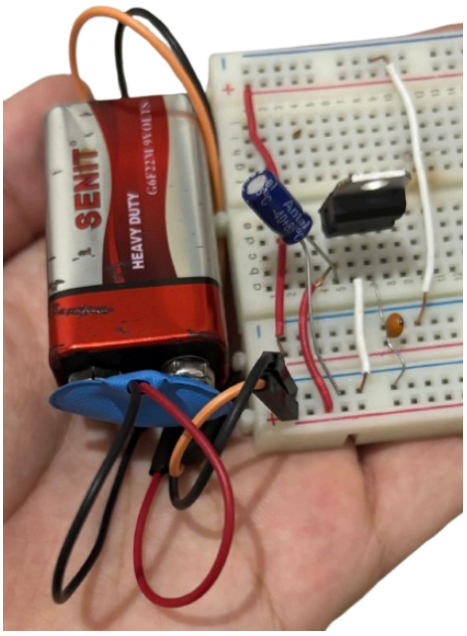
   - Sensor MAX30102
     - SCL (Serial Clock Line): Terhubung ke GPIO 22. Pin ini dipilih karena GPIO 22 adalah salah satu pin clock I2C khusus pada ESP32, memastikan komunikasi yang optimal dengan sensor
     - SDA (Jalur Data Serial): Terhubung ke GPIO 21. Pin ini dipilih karena merupakan pin data I2C default untuk ESP32, sehingga memungkinkan integrasi yang mulus dengan protokol I2C
   - Sensor DHT11
     - Pin Data: Terhubung ke GPIO 25, yang disediakan dan dikonfigurasikan untuk input digital. Pin ini menangani transmisi pembacaan suhu dan kelembaban ruangan.
   - Sensor DS18B20
     - Pin Data (Protokol One-Wire): Terhubung ke GPIO 26. Resistor pull-up 4,7kΩ ditempatkan di antara pin data dan Vin sensor untuk menstabilkan sinyal komunikasi setiap sensor dan mencegah perilaku yang tidak menentu.
   - Buzzer:
     - Pin Kontrol: Terhubung ke GPIO 14, yang ditetapkan sebagai output untuk mengaktifkan peringatan suara jika terjadi abnormalitas pada data kesehatan pasien.



Gambar 4. Implementasi komponen pada breadboard

3. Power Supply Portabel:

   Meskipun sistem ini pada dasarnya diberi daya melalui port micro-USB ESP32 selama pengujian, namun ada upaya untuk memberi daya pada sistem dengan menggunakan baterai agar mudah dibawa-bawa. Kami berusaha untuk menggunakan sumber daya dari baterai 9V yang diregulasikna menjadi 5V dengan IC 7805 dan kapasitor. Namun karena saat diukur pembacaan rangkaian tegangan regulator dengan multimeter, didapatkan nilai 5,21V, sehingga kelompok kami tidak berani untuk memasang baterai ke dalam rangkaian, dengan khawatirnya akan terjadi *short circuit*



Gambar 5. Implementasi baterai 9V dengan rangkaian voltage regulator

3.2 Software Implementation

3.2.1 Aplikasi Software

1. EasyEDA:

   EasyEDA digunakan untuk merancang PCB khusus demi komponen perangkat kerasnya. Prosesnya desain PCB dimulai dengan membuat diagram skematik rangkaian, yang seudah mewakili semua komponen dan koneksi pada breadboard. Setelah skematik selesai, komponen ditempatkan pada susunan PCB dengan mengoptimalkan perutean PCB satu lapis karena keterbatasan dana dan waktu. Banayk waktu yang dihabiskan untuk menyusun ulang komponen dan jalur PCB agar meminimalkan tumpang tindih dan memastikan konektivitas listrik yang baik.

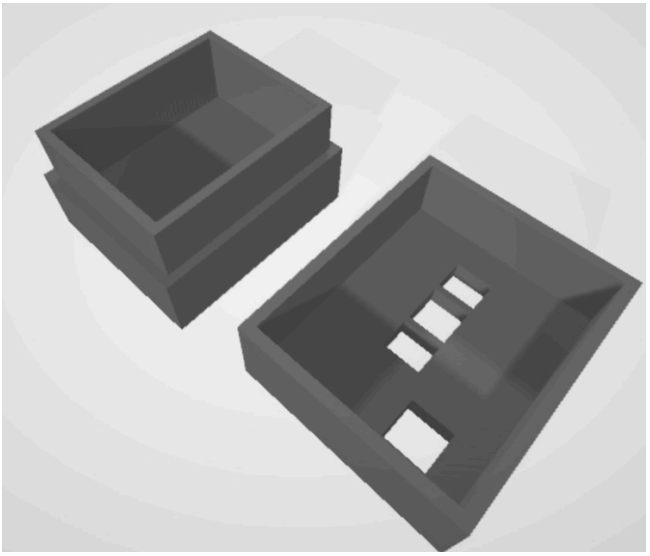
   Terlepas dari perencanaan yang sangat teliti, kesulitan rangkaian dan keterbatasan lapisan satu lapis menyebabkan kendala dalam penjaluran PCB sehingga, PCB yang sudah di cetak tidak dapat di implemnetasikan ke komponen pearngkat keras dalam proyek.

2. Fusion 360

   Untuk meningkatkan tampilan proyek dan melindungi rangkaian, casing 3D dirancang menggunakan Autodesk Fusion 360. Proses desain

difokuskan pada aspek fungsionalitas dan kepraktisan, dengan menyatukan berbagai fitur seperti:

- Slot Sensor: Bukaan untuk sensor MAX30102, DS18B20, dan DHT11 untuk memastikan pengambilan data yang optimal.
- Lubang Ventilasi: Untuk menjaga aliran udara, khususnya untuk DHT11, yang sensitif terhadap kondisi lingkungan.



Gambar 6. Implementasi Fusion 360 untuk mendesain casing

Akan tetapi, pencetakan 3D pada casing tidak dapat diselesaikan karena keterbatasan waktu dan antrian yang panjang dalam penggunaan mesin 3D Printing.

3. Arduino IDE

Arduino IDE digunakan dalam proyek ini karena software ini merupakan platform yang sederhana dan efektif untuk melakukan pengembangan perangkat lunak pada mikrokontroler, seperti ESP32. Software ini mempermudah proses pengkodean, kompilasi, dan pengunggahan kode ke perangkat lunak. Dengan pustaka bawaan, seperti DHT.H, DallasTemperature.h, MAX30105.h dan BlynkSimpleEsp32.h yang memungkinkan untuk dilakukan integrasi perangkat keras pada sensor suhu (DHT11 dan DS18B20), sensor detak jantung (MAX30102), dan modul IoT (Blynk) dengan efisien.

Implementasi Arduino IDE dalam proyek ini dimulai dengan menulis kode yang mencakup konfigurasi perangkat keras, inisiasi pustaka, dan pengaturan komunikasi dengan server Blynk. Pada tahap awal, inisiasi semua perangkat keras dilakukan melalui fungsi setup(). Setelah itu, Arduino IDE akan menjalankan alur utama program, seperti membaca data dari sensor, memproses informasi, dan mengirimkan hasilnya ke dashboard Blynk melalui fungsi Blynk.virtualWrite().

4. Blynk

Blynk digunakan dalam proyek ini karena kemampuannya yang sangat baik dalam mempermudah integrasi perangkat berbasis IoT dengan antarmuka pengguna secara real-time melalui aplikasi seluler atau web. Implementasi Blynk dalam proyek ini dengan menambahkan pustaka BlynkSimpleEsp32.h ke dalam program menggunakan Arduino IDE. Setelah itu, token autentikasi dari aplikasi Blynk, serta kredensial jaringan Wi-Fi digunakan untuk menghubungkan ESP32 ke server blynk menggunakan fungsi Blynk.begin (). Data dari sensor akan dikirim ke dashboard menggunakan fungsi Blynk.virtualwrite() ke virtual pins yang telah ditentukan. Terakhir pengguna dapat melihat hasil data secara real time melalui aplikasi seluler atau web yang divisualisasikan dalam bentuk indikator status atau nilai numerik.

3.2.1 Kode

a. Definisi Template Blynk

```
#define BLYNK_TEMPLATE_NAME "Quickstart Template"
#define BLYNK_TEMPLATE_ID "TMPL6wFf7wO8V"
#define BLYNK_PRINT Serial
```

Fungsi ini digunakan untuk mendefinisikan nama template dan ID untuk menghubungkan Arduino IDE ke server Blynk

b. Pustaka yang digunakan

```
#include <DHT.h>
#include <OneWire.h>
#include <DallasTemperature.h>
```

```
#include <Wire.h>
#include "MAX30105.h"
#include <Adafruit_Sensor.h>
#include <BlynkSimpleEsp32.h>
```

Kode ini mencakup pustaka yang diperlukan untuk mengintegrasikan dan mengontrol perangkat keras ke dalam sistem. Pustaka DHT.h digunakan untuk membaca data dari sensor DHT11, untuk mengukur suhu dan kelembapan udara. Pustaka OneWire.h dan DallasTemperature.h digunakan untuk mengatur komunikasi dengan sensor suhu DS18B20, untuk pengambilan data suhu tubuh secara presisi. Pustaka Wire.h digunakan untuk mendukung komunikasi I2C yang digunakan oleh sensor MAX30102. Pustaka MAX30105.h digunakan untuk membaca data inframerah untuk menghitung detak jantung dan saturasi oksigen. Pustaka Adafruit_sensor.h adalah pustaka yang digunakan untuk mendukung kompatibilitas. Terakhir, pustaka BlynkSimpleEsp32.h digunakan untuk memungkinkan koneksi antara ESP32 dan server Blynk.

c. Konfigurasi Blynk dan Jaringan internet

```
char auth[] = "_tOy-PDlMW-0FmRkBjAWQev3MGu0ebMq";
char ssid[] = "xxx";
char pass[] = "xxxxx";
```

Fungsi auth[] digunakan untuk menghubungkan perangkat ke akun blynk. ssid[] dan pass[] digunakan untuk menghubungkan ke koneksi internet

d. Konfigurasi Sensor DHT11

```
#define DHTTYPE DHT11
#define DHTPIN 19
DHT dht(DHTPIN, DHTTYPE);
```

Fungsi ini digunakan untuk mengkonfigurasi sensor DHT11. Sensor DHT11 digunakan untuk membaca suhu dan kelembapan lingkungan. Konfigurasi dimulai dengan mendefinisikan jenis sensor menggunakan DHTTYPE. Pin ESP32 yang digunakan untuk berkomunikasi dengan sensor adalah pin digital 19. Objek dht dibuat untuk memungkinkan pembacaan suhu dan kelembapan secara langsung dari sensor.

e. Konfigurasi Sensor DS18B20

```
#define ONE_WIRE_BUS 26
OneWire oneWire(ONE_WIRE_BUS);
```

```
DallasTemperature sensors(&oneWire);
```

Fungsi ini digunakan untuk menggunakan sensor DS18B20 dengan pin komunikasi ditentukan pada pin 26. Onewire dibuat untuk mengelola koneksi dan objek sensor yang digunakan untuk memungkinkan pembacaan suhu tubuh yang lebih presisi.

f. Konfigurasi Sensor MAX30102

```
MAX30105 particleSensor;
const byte RATE_SIZE = 8;
byte rates[RATE_SIZE];
byte rateSpot = 0;
long lastBeat = 0;
float beatsPerMinute;
int beatAvg;
```

Fungsi ini digunakan untuk melakukan konfigurasi sensor MAX30102. Objek particleSensor dibuat untuk mengelola komunikasi dengan sensor, sementara variabel seperti beatsPerMinute, beatAvg, dan spo2Value digunakan untuk menyimpan hasil pengukuran.

g. Variabel Global

```
float spo2Value = 0.0;
float temperatureDS18B20 = 0.0;
float humidity = 0.0;
float temperatureDHT = 0.0;

int patientStatus = 0;
int buzzerEnabled = 0;
```

Fungsi ini digunakan untuk menyimpan hasil pembacaan dari berbagai sensor dan untuk sebagai status kontrol kondisi pasien serta untuk kontrol buzzer.

h. Konfigurasi Buzzer

```
#define BUZZER_PIN 16
```

Fungsi ini digunakan untuk menentukan pin 16 sebagai output untuk mengontrol buzzer.

i. Fungsi Deteksi Detak Jantung

```
bool customCheckForBeat(long irValue) {
 static long prevIrValue = 0;
```

```
 static bool isPeak = false;
 static unsigned long lastPeakTime = 0;

 if (irValue > 50000 && irValue > prevIrValue && !isPeak) {
   unsigned long currentTime = millis();
   if (currentTime - lastPeakTime > 300) {
     isPeak = true;
     lastPeakTime = currentTime;
     prevIrValue = irValue;
     return true;
   }
 } else if (irValue < prevIrValue) {
   isPeak = false;
 }

 prevIrValue = irValue;
 return false;
}
```

Fungsi ini digunakan untuk menganalisis data inframerah (IR) dari MAX30102 untuk mendeteksi puncak yang sesuai dengan detak jantung. Hasil analisis ini digunakan untuk menghitung detak jantung (BPM)

j. Fungsi Evaluasi Status Pasien

```
void evaluatePatientStatus() {
 if (beatAvg < 60 || beatAvg > 100 || spo2Value < 95.0 ||
temperatureDS18B20 > 37.5) {
   patientStatus = 1; // Unhealthy
 } else {
   patientStatus = 0; // Healthy
 }

 Blynk.virtualWrite(V7, patientStatus);
 Serial.print("Patient Status: ");
 Serial.println(patientStatus == 0 ? "Healthy" : "Unhealthy");

 if (patientStatus == 1 && buzzerEnabled == 1) {
   digitalWrite(BUZZER_PIN, HIGH);
 } else {
   digitalWrite(BUZZER_PIN, LOW);
 }
}
```

Fungsi ini digunakan untuk menentukan status kesehatan pasien berdasarkan

- BPM (<60 atau >100 dianggap tidak sehat)
- SpO2 (<95% dianggap tidak sehat)
- Suhu tubuh (>37,5°C dianggap tidak sehat)

Berdasarkan status tersebut, jika status tidak sehat, buzzer akan diaktifkan.

k. Pembacaan Sensor DHT11

```
void readDHT11() {
 humidity = dht.readHumidity();
 temperatureDHT = dht.readTemperature();

 if (!isnan(humidity) && !isnan(temperatureDHT)) {
   Serial.print("Temperature (DHT11): ");
   Serial.print(temperatureDHT);
   Serial.print(" °C, Humidity: ");
   Serial.print(humidity);
   Serial.print(" %");

   // Send data to Blynk
   Blynk.virtualWrite(V5, humidity);
   Blynk.virtualWrite(V6, temperatureDHT);
 } else {
   Serial.print("DHT11 Sensor Error");
 }
}
```

Fungsi ini digunakan untuk membaca suhu dan kelembapan dari sensor DHT11, memvalidasi hasil pembacaan, dan mengirimkan data ke dashboard Blynk.

l. Pembacaan Sensor DS18B20

```
#void readDS18B20() {
 sensors.requestTemperatures();
 temperatureDS18B20 = sensors.getTempCByIndex(0);
 if (temperatureDS18B20 != DEVICE_DISCONNECTED_C) {
   Serial.print(", Temperature (DS18B20): ");
   Serial.print(temperatureDS18B20);
   Serial.print(" °C");

   // Send data to Blynk
   Blynk.virtualWrite(V4, temperatureDS18B20);
```

```
 } else {
   Serial.print(", DS18B20 Sensor Error");
 }
}
```

Fungsi ini digunakan untuk membaca data suhu tubuh dari DS18B20. Setelah pembacaan dilakukan, data ini dikirim ke dashboard Blynk melalui virtual pins

m. Pembacaan Sensor MAX30102

```
void readMAX30102() {
 if (millis() - maxSensorLastRead >= maxSensorInterval) {
   maxSensorLastRead = millis();
   long irValue = particleSensor.getIR();
   long redValue = particleSensor.getRed();

   if (customCheckForBeat(irValue)) {
     long delta = millis() - lastBeat;
     lastBeat = millis();

     beatsPerMinute = 60.0 / (delta / 1000.0);
     if (beatsPerMinute < 255 && beatsPerMinute > 40) {
       rates[rateSpot++] = (byte)beatsPerMinute;
       rateSpot %= RATE_SIZE;

       beatAvg = 0;
       for (byte x = 0; x < RATE_SIZE; x++)
         beatAvg += rates[x];
       beatAvg /= RATE_SIZE;
     }
   }

   if (redValue > 0 && irValue > 0) {
     float ratio = (float)redValue / (float)irValue;
     spo2Value = 97.0 - (ratio * 30.0); //Rumus kalibrasi dasar
   }

   Serial.print("BPM: ");
   Serial.print(beatsPerMinute);
   Serial.print(", Avg BPM: ");
   Serial.print(beatAvg);
   Serial.print(", SpO2: ");
   Serial.print(spo2Value, 2);
```

```
    Serial.println(" %");

    // Send data to Blynk
    Blynk.virtualWrite(V0, beatAvg);
    Blynk.virtualWrite(V1, spo2Value);

    // Evaluate patient status
    evaluatePatientStatus();
  }
}
```

Fungsi ini digunakan untuk membaca data inframerah (IR) dan cahaya merah dari MAX30102. Data ini digunakan untuk menghitung BPM dan SpO2. Hasil data ini kemudian akan dikirim ke server Blynk untuk divisualisasikan

n. Kontrol Input dari Dashboard

```
BLYNK_WRITE(V8) {
 buzzerEnabled = param.asInt();
 Serial.print("Buzzer Enabled: ");
 Serial.println(buzzerEnabled);
}
```

Fungsi ini digunakan untuk menerima input dari pengguna melalui dashboard Blynk pada virtual pin V8. Input ini memungkinkan pengguna untuk mengontrol status buzzer (mengaktifkan atau menonaktifkan) secara manual pada dashboard.

o. Inisialisasi Sistem

```
void setup() {
 Serial.begin(115200);
 Serial.println("Initializing sensors...");

 Blynk.begin(auth, ssid, pass);

 // Inisiasi DHT11
 dht.begin();

 //Inisiasi DS18B20
 sensors.begin();

 //Inisiasi MAX30102
 if (!particleSensor.begin()) {
```

```
    Serial.println("MAX30102 was not found. Please check the
connections.");
    while (1);
  }
  particleSensor.setup();
  particleSensor.setPulseAmplitudeRed(0x0A);
  particleSensor.setPulseAmplitudeIR(0x1F);

  //Inisiasi Buzzer
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);

  Serial.println("All sensors initialized successfully.");
}
```

Fungsi ini dijalankan sekali pada awal sistem. Sistem ini akan menginisiasikan koneksi serial, jaringan Wi-Fi, dan server Blynk, termasuk sensor DHT11, DS18B20, dan MAX30102.

p. Fungsi Loop

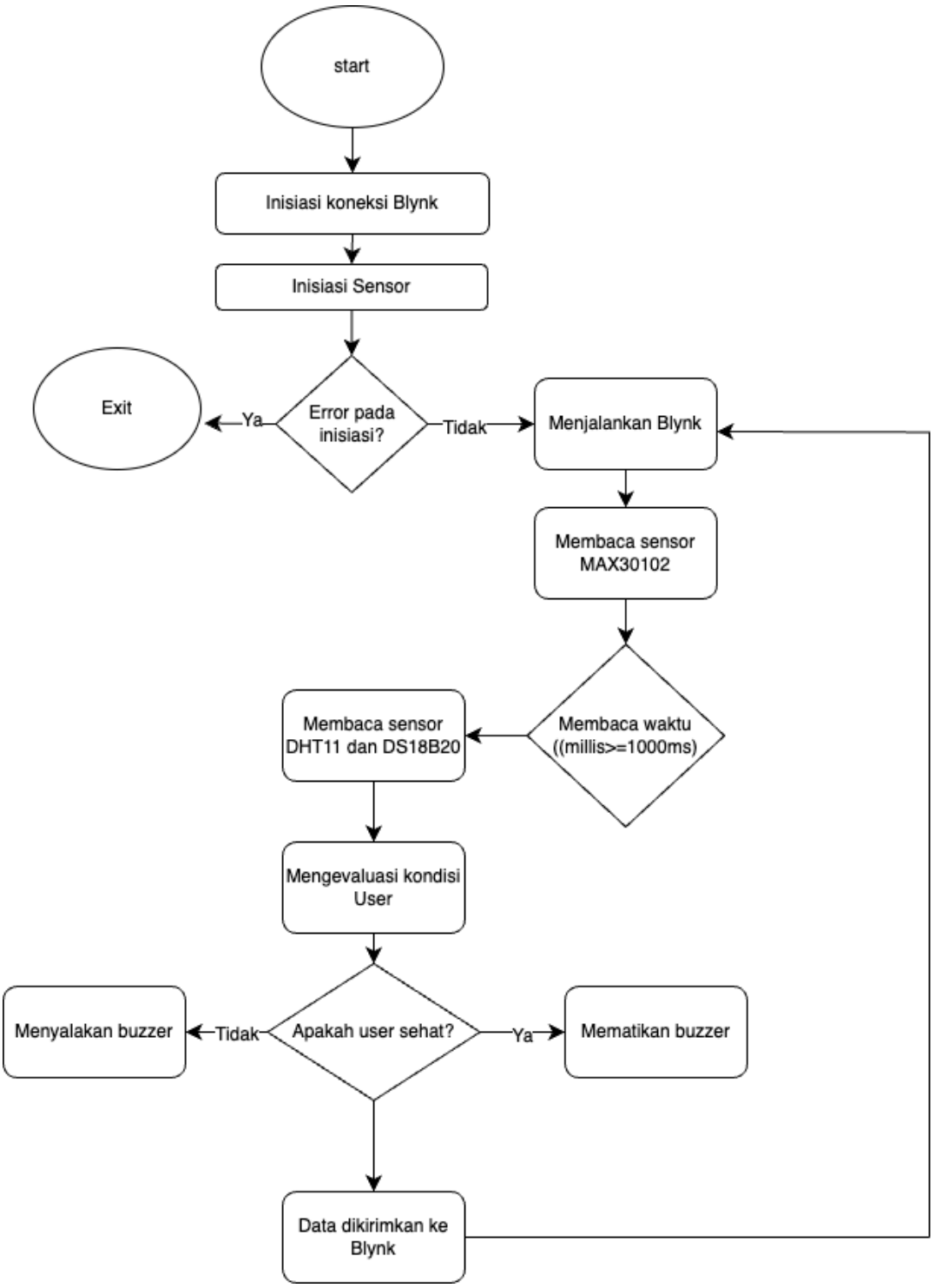
```
void loop() {
  Blynk.run();

  readMAX30102();

  // Read DHT11 and DS18B20 sensors with delay to avoid
interference
  static unsigned long lastSensorRead = 0;
  if (millis() - lastSensorRead >= 1000) { // Run DHT11 and DS18B20
every 1 second
    lastSensorRead = millis();
    Serial.print("\n[Sensor Data] ");
    readDHT11();
    readDS18B20();
    Serial.println();
  }
}
```

Fungsi ini dijalankan terus-menerus untuk membaca data dari semua sensor dan mengirimkan hasil pembacaan ke dashboard Blynk secara berkala.

3.2.3 Alur Program



Gambar 7. Alur program

BAB IV
TESTING AND DEBUGGING

5.1. Metode Pengujian

A. Sensor DHT11
   Sensor diuji untuk membaca suhu dan kelembaban ruangan. Proses pengujian fungsionalitas dilakukan dengan memeriksa kesesuaian hasil data suhu dan kelembaban yang diukur oleh DHT11 dengan suhu dan kelembaban lingkungan aslinya. Sensor diuji dalam lingkungan dengan perubahan suhu dan kelembaban yang bervariasi untuk memeriksa *reliability.*
B. Sensor DS18B20
   Sensor diuji untuk membaca suhu tubuh. Pengujian dilakukan dengan mendekatkan sensor pada tubuh pengguna. Selain itu, sensor juga diuji pada suhu yang bervariasi untuk memeriksa akurasi dari sensor.
C. Sensor MAX301012
   Sensor diuji untuk mendeteksi detak jantung dan saturasi oksigen. Pengujian dilakukan langsung pada pengguna untuk memastikan sensor dapat mendeteksi detak jantung dan saturasi oksigen dalam kondisi nyata. Akurasi dari detak jantung masih perlu diuji lebih lanjut untuk menyesuaikan dengan kondisi nyatanya. Namun, sensor ini sudah dapat mendeteksi detak jantung pada berbagai kondisi, misalnya pada saat kondisi rileks atau pada saat kondisi beraktivitas ringan. Untuk saturasi oksigen, akurasi masih sangat rendah karena belum menggunakan algoritma yang sesuai untuk mendeteksi saturasi oksigen.
D. Koneksi dengan Blynk
   Pengujian koneksi Blynk melibatkan pengukuran stabilitas dan kemampuan sistem dalam mempertahankan komunikasi real-time antara data dari Arduino IDE dengan server pada Blynk. Sistem diuji akurasinya dengan memeriksa konsistensi dan kesesuaian antara data yang ditampilkan pada serial monitor pada Arduino IDE dan pada dashboard Blynk.
E. Buzzer
   Pengujian buzzer mencakup penilaian terhadap fungsionalitas aktivasi otomatis buzzer berdasarkan data yang didapatkan dari sensor serta kontrol manual melalui dashboard Blynk.

5.2. Hasil Pengujian

Hasil pengujian menunjukkan sistem dapat bekerja dengan baik, dalam hal mendeteksi dan menampilkan data dari sensor secara real-time. Dengan detail sebagai berikut:

A. Sensor DHT11

Sensor berhasil membaca suhu dan kelembaban lingkungan dalam kondisi ruangan normal dengan stabil. Data suhu dan kelembaban ditampilkan di dashboard Blynk secara real-time.

B. Sensor DS18B20
Sensor berhasil membaca suhu tubuh pengguna dengan stabil. Data suhu tubuh dikirimkan ke dashboard Blynk secara real time. Sensor dapat memberikan hasil yang konsisten dalam berbagai kondisi.

C. Sensor MAX30102
Sensor berhasil membaca detak jantung dan saturasi oksigen dari pengguna. Data juga dapat dikirimkan secara konsisten ke dashboard Blynk secara real-time. Untuk hasil pembacaan dari detak jantung, akurasi dari pembacaan sudah cukup akurat dapat membaca detak jantung dalam berbagai kondisi yang berbeda. Namun, untuk pembacaan saturasi oksigen, akurasi masih cukup rendah karena masih menggunakan rumus sederhana untuk melakukan perhitungan saturasi oksigen

D. Koneksi Blynk
Sistem berhasil mengirimkan data dari semua sensor ke dashboard blynk dan data dapat ditampilkan secara real-time tanpa error.

E. Buzzer
Buzzer berhasil ditampilkan secara otomatis ketika status pasien “tidak sehat”. Buzzer dapat dimatikan melalui dashboard Blynk dengan baik tanpa error.

5.3. Tantangan yang Dihadapi

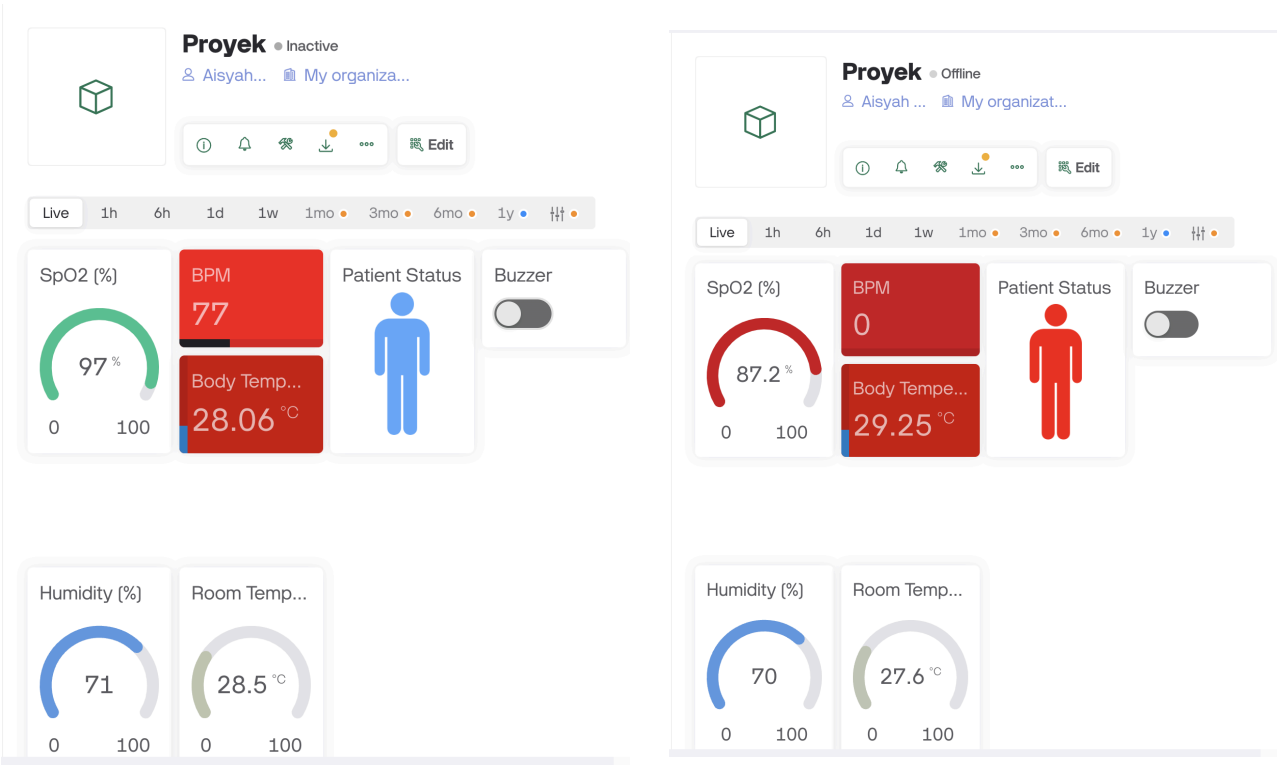
Selama pengembangan sistem ini, terdapat beberapa tantangan yang dihadapi, terutama dalam hal integrasi komponen dan pengembangan algoritma. Tantangan pertama adalah integrasi beberapa sensor, seperti DHT11,DS18B20, dan MAX30102 ke dalam satu sistem berbasis ESP32. Dalam menjalankan sensor MAX30102 diperlukan algoritma yang cukup sensitif terhadap waktu. Pada awalnya sensor ini tidak dapat dijalankan,terutama dalam hal membaca data detak jantung. Hal ini karena semua sensor dijalankan dalam waktu bersamaan. Oleh karena itu, untuk mengatasi hal ini, dibuatlah delay selama satu detik dalam menjalankan kembali sensor DHT11 dan DS18B20. Hal ini untuk memberikan waktu yang maksimal kepada sensor MAX30102 dalam memproses data detak jantung dengan baik. Tantangan lainnya yang dihadapi adalah dalam mengembangkan algoritma perhitungan SpO_2, Ketika dilakukan uji pembuatan kode menggunakan algoritma yang ada di literatur, kode tidak dapat dijalankan. Oleh karena itu, karena keterbatasan waktu untuk melakukan pengembangan lebih lanjut, algoritma perhitungan SpO_2 ini hanya menggunakan algoritma perhitungan yang cukup sederhana, tetapi diperkirakan masih dapat merepresentasikan kondisi kesehatan pengguna.

Selain itu, terdapat kendala dalam mendesain PCB yang disebabkan karena keterbatasan waktu. Proses desain dan fabrikasi PCB diselesaikan dalam waktu singkat yang menyebabkan komponen tidak dapat dijalankan ketika menggunakan PCB. Hal ini diperkirakan karena terdapat kesalahan dalam tata letak pada PCB yang mempengaruhi aliran listrik tidak dapat berjalan dengan baik. Tantangan terakhir adalah dalam melakukan 3D printing untuk casing perangkat. Desain casing ini sudah dibentuk, tetapi karena keterbatasan waktu tidak dapat dilakukan printing.

BAB VI
USER INTERFACE

6.1. Deskripsi Interface

Sistem ini dirancang dengan menggunakan interface pada platform IoT Blynk yang memungkinkan pengguna untuk memantau data sensor secara real time melalui aplikasi seluler atau melalui website.



Gambar 8. Interface pada blynk pada kondisi sehat dan pada kondisi tidak sehat

Kedua gambar di atas adalah interface untuk pengguna pada dua kondisi, yaitu sehat dan saat kondisi tidak sehat. Pada kondisi sehat, tampilan akan menunjukkan

status pasien dengan logo berwarna biru dan ketika tidak sehat ditampilkan dengan logo berwarna merah. Selain itu, *interface* ini juga menyajikan data suhu dan kelembaban ruangan, suhu tubuh, detak jantung, serta saturasi oksigen (SpO_2) dalam bentuk display label dan gauge untuk menunjukkan perubahan data dari waktu ke waktu. Hal ini diterapkan untuk mempermudah pemantauan data kepada pengguna. Pada kontrol alarm,berupa buzzer menggunakan kontrol switch yang memungkinkan pengguna untuk mengaktifkan atau menonaktifkan alarm dengan mudah.

BAB VII
DISKUSI DAN PEMBAHASAN

7.1. Aplikasi Dunia Nyata

Pada dunia nyata, VitaSense dapat bermanfaat pada aplikasi pemantauan kesehatan pasien di rumah dan di daerah terpencil. Selain itu juga pada pemantauan kesehatan di rumah sakit untuk pasien non-kritis.

7.1. Ruang Lingkup Pengembangan di Masa Depan

Untuk ruang lingkup pengembangan di masa depan, VitaSense dapat digunakan untuk menambahkan sensor tambahan untuk mengukur parameter seperti tekanan darah, ECG, dan kadar gula darah. serta, mengintegrasikan sistem ke perangkat wearable. Selain itu juga dapat digunakan untuk mengimplementasikan algoritma AI dan machine learning untuk menganalisis tren data pasien, memprediksi risiko kesehatan, serta melakukan penilaian perubahan kondisi lingkungan mempengaruhi kesehatan pasien.

BAB VIII
PENUTUP

8.1. Kesimpulan

Kita berhasil mengembangkan sistem pemantauan kesehatan real-time yang dapat mengukur tanda-tanda vital dan kondisi lingkungan, serta mengirimkan data ke platform jarak jauh untuk pemantauan dan peringatan real-time. Sistem ini menggunakan berbagai sensor, termasuk oksimeter denyut nadi, sensor suhu, serta sensor kelembaban dan suhu. Data dikumpulkan oleh mikrokontroler ESP32 dan dikirimkan ke platform IoT Blynk. Platform Blynk memungkinkan kita untuk melihat data secara real-time di smartphone atau komputer. Kita juga mengembangkan prototipe PCB dan casing untuk sistem tersebut. Namun, karena keterbatasan waktu, PCB tidak sepenuhnya diterapkan dan casing tidak dicetak 3D. Kita berencana untuk terus mengembangkan sistem VitaSense di masa mendatang, termasuk meningkatkan desain

PCB dan menambahkan fitur-fitur baru. Sistem VitaSense berpotensi menjadi alat yang berharga untuk memantau kesehatan pasien, terutama mereka yang sudah lanjut usia atau memiliki kondisi kesehatan kronis. Sistem ini dapat membantu mendeteksi masalah kesehatan sejak dini, yang dapat mengarah pada perawatan lebih dini dan hasil yang lebih baik.

8.2. Saran

Beberapa saran dari proyek ini adalah kita dapat terus mengembangkan sistem VitaSense dengan menyempurnakan desain PCB dan menambahkan fitur-fitur baru. Kita dapat melakukan uji klinis untuk menguji efektivitas sistem VitaSense dalam memantau kesehatan pasien. Kita dapat memasarkan sistem VitaSense kepada penyedia layanan kesehatan dan pasien. Dan kita juga dapat menjadikan sistem VitaSense lebih terjangkau dan dapat diakses oleh lebih banyak orang.

Sistem VitaSense berpotensi menjadi alat yang berharga untuk memantau kesehatan pasien, terutama mereka yang sudah lanjut usia atau memiliki kondisi kesehatan kronis. Sistem ini dapat membantu mendeteksi masalah kesehatan sejak dini, yang dapat mengarah pada perawatan lebih dini dan hasil yang lebih baik.

DAFTAR REFERENSI

1. R. Uddin and I. Koo, “Real-time remote patient monitoring: A review of biosensors integrated with multi-hop IoT systems via cloud connectivity,” *Appl. Sci. (Basel)*, vol. 14, no. 5, p. 1876, 2024.
2. “IoT patient health monitoring system project using ESP32,” *DIY Projects Lab*, 17-May-2023. [Online]. Available: https://diyprojectslab.com/esp32-iot-patient-health-monitoring-system/. [Accessed: 27-Dec-2024].
3. T. K. Wong, H. K. Mun, S. K. Phang, K. L. Lum, and W. Q. Tan, “Real-time machine health monitoring system using machine learning with IoT technology,” *MATEC Web Conf.*, vol. 335, p. 02005, 2021.

https://drive.google.com/drive/folders/1nM0o1kuHp8whtikB_f4Jl3ZJDGZcWWPr?usp=sharing

DAFTAR LAMPIRAN

Lampiran Video:
https://drive.google.com/drive/folders/1nM0o1kuHp8whtikB_f4Jl3ZJDGZcWWPr?usp=sharing

Lampiran Kode

```
#define BLYNK_TEMPLATE_NAME "Quickstart Template"
#define BLYNK_TEMPLATE_ID "TMPL6wFf7wO8V"
#define BLYNK_PRINT Serial

#include <DHT.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include "MAX30105.h"
#include <Adafruit_Sensor.h>
#include <BlynkSimpleEsp32.h>

// Blynk configuration
char auth[] = "_tOy-PDlMW-0FmRkBjAWQev3MGu0ebMq";
char ssid[] = "Ke";
char pass[] = "melkeisya";

// DHT11 Configuration
#define DHTTYPE DHT11
#define DHTPIN 19
DHT dht(DHTPIN, DHTTYPE);

// DS18B20 Configuration
#define ONE_WIRE_BUS 26
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

// MAX30102 Configuration
MAX30105 particleSensor;
const byte RATE_SIZE = 8;
byte rates[RATE_SIZE];
byte rateSpot = 0;
long lastBeat = 0;
float beatsPerMinute;
int beatAvg;
```

```
float spo2Value = 0.0;
float temperatureDS18B20 = 0.0;
float humidity = 0.0;
float temperatureDHT = 0.0;

int patientStatus = 0;
int buzzerEnabled = 0;

// Buzzer pin
#define BUZZER_PIN 16

// Timing variables
unsigned long maxSensorLastRead = 0;
const unsigned long maxSensorInterval = 100;

// Fungsi untuk mendeteksi BPM
bool customCheckForBeat(long irValue) {
 static long prevIrValue = 0;
 static bool isPeak = false;
 static unsigned long lastPeakTime = 0;

 if (irValue > 50000 && irValue > prevIrValue && !isPeak) {
   unsigned long currentTime = millis();
   if (currentTime - lastPeakTime > 300) {
     isPeak = true;
     lastPeakTime = currentTime;
     prevIrValue = irValue;
     return true;
   }
 } else if (irValue < prevIrValue) {
   isPeak = false;
 }

 prevIrValue = irValue;
 return false;
}

void evaluatePatientStatus() {
 if (beatAvg < 60 || beatAvg > 100 || spo2Value < 95.0 ||
temperatureDS18B20 > 37.5) {
   patientStatus = 1; // Unhealthy
 } else {
```

```
    patientStatus = 0; // Healthy
  }

  Blynk.virtualWrite(V7, patientStatus);
  Serial.print("Patient Status: ");
  Serial.println(patientStatus == 0 ? "Healthy" : "Unhealthy");

  if (patientStatus == 1 && buzzerEnabled == 1) {
    digitalWrite(BUZZER_PIN, HIGH);
  } else {
    digitalWrite(BUZZER_PIN, LOW);
  }
}

void readDHT11() {
  humidity = dht.readHumidity();
  temperatureDHT = dht.readTemperature();

  if (!isnan(humidity) && !isnan(temperatureDHT)) {
    Serial.print("Temperature (DHT11): ");
    Serial.print(temperatureDHT);
    Serial.print(" °C, Humidity: ");
    Serial.print(humidity);
    Serial.print(" %");

    // Send data to Blynk
    Blynk.virtualWrite(V5, humidity);
    Blynk.virtualWrite(V6, temperatureDHT);
  } else {
    Serial.print("DHT11 Sensor Error");
  }
}

void readDS18B20() {
  sensors.requestTemperatures();
  temperatureDS18B20 = sensors.getTempCByIndex(0);
  if (temperatureDS18B20 != DEVICE_DISCONNECTED_C) {
    Serial.print(", Temperature (DS18B20): ");
    Serial.print(temperatureDS18B20);
    Serial.print(" °C");

    // Send data to Blynk
    Blynk.virtualWrite(V4, temperatureDS18B20);
```

```
  } else {
    Serial.print(", DS18B20 Sensor Error");
  }
}

void readMAX30102() {
  if (millis() - maxSensorLastRead >= maxSensorInterval) {
    maxSensorLastRead = millis();
    long irValue = particleSensor.getIR();
    long redValue = particleSensor.getRed();

    if (customCheckForBeat(irValue)) {
      long delta = millis() - lastBeat;
      lastBeat = millis();

      beatsPerMinute = 60.0 / (delta / 1000.0);
      if (beatsPerMinute < 255 && beatsPerMinute > 40) {
        rates[rateSpot++] = (byte)beatsPerMinute;
        rateSpot %= RATE_SIZE;

        beatAvg = 0;
        for (byte x = 0; x < RATE_SIZE; x++)
          beatAvg += rates[x];
        beatAvg /= RATE_SIZE;
      }
    }

    if (redValue > 0 && irValue > 0) {
      float ratio = (float)redValue / (float)irValue;
      spo2Value = 97.0 - (ratio * 30.0); //Rumus kalibrasi dasar
    }

    Serial.print("BPM: ");
    Serial.print(beatsPerMinute);
    Serial.print(", Avg BPM: ");
    Serial.print(beatAvg);
    Serial.print(", SpO2: ");
    Serial.print(spo2Value, 2);
    Serial.println(" %");

    // Send data to Blynk
    Blynk.virtualWrite(V0, beatAvg);
    Blynk.virtualWrite(V1, spo2Value);
```

```
    // Evaluate patient status
    evaluatePatientStatus();
  }
}

BLYNK_WRITE(V8) {
  buzzerEnabled = param.asInt();
  Serial.print("Buzzer Enabled: ");
  Serial.println(buzzerEnabled);
}

void setup() {
  Serial.begin(115200);
  Serial.println("Initializing sensors...");

  Blynk.begin(auth, ssid, pass);

  // Inisiasi DHT11
  dht.begin();

  //Inisiasi DS18B20
  sensors.begin();

  //Inisiasi MAX30102
  if (!particleSensor.begin()) {
    Serial.println("MAX30102 was not found. Please check the
connections.");
    while (1);
  }
  particleSensor.setup();
  particleSensor.setPulseAmplitudeRed(0x0A);
  particleSensor.setPulseAmplitudeIR(0x1F);

  //Inisiasi Buzzer
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);

  Serial.println("All sensors initialized successfully.");
}

void loop() {
  Blynk.run();
```

```
 readMAX30102();

 // Read DHT11 and DS18B20 sensors with delay to avoid interference
 static unsigned long lastSensorRead = 0;
 if (millis() - lastSensorRead >= 1000) { // Run DHT11 and DS18B20
every 1 second
   lastSensorRead = millis();
   Serial.print("\n[Sensor Data] ");
   readDHT11();
   readDS18B20();
   Serial.println();
 }
}
```

PEMBAGIAN TUGAS

1. **PENDAHULUAN**, **DISKUSI DAN PEMBAHASAN**, **PENUTUP** : Surya
2. **SYSTEM DESIGN AND COMPONENTS, HARDWARE AND SOFTWARE IMPLEMENTATION :** Hasan
3. **HARDWARE AND SOFTWARE IMPLEMENTATION, USER INTERFACE** : Aisyah