



**REAL TIME SYSTEM AND INTERNET OF THINGS FINAL PROJECT REPORT  
DEPARTMENT OF ELECTRICAL ENGINEERING  
UNIVERSITAS INDONESIA**

**Smart Health Monitoring System for Real-time Heart Rate and Body  
Temperature Tracking**

**GROUP B3**

Lauren Christy Tanudjaja	2106707870
Mikhael Morris Hapataran Siallagan	2106731491
Rafi Fauzan Wirasena	2106656320
Zalfy Putra Rezky	2106731453

## PREFACE

Puji syukur kita panjatkan kehadirat Allah SWT, karena atas rahmat dan hidayah-Nya, yang telah membantu kelompok kami dalam menyelesaikan laporan proyek akhir praktikum Sistem Waktu Nyata dan IoT 2023 ini dengan lengkap.

Dalam era modern ini, perawatan kesehatan semakin menjadi fokus utama masyarakat global. Kehidupan yang cepat dan dinamis seringkali menyebabkan orang mengabaikan aspek penting kesehatan mereka. Oleh karena itu, pengembangan solusi yang memungkinkan pemantauan kesehatan secara real-time menjadi suatu kebutuhan mendesak. Proyek "Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking" bertujuan untuk mengatasi tantangan tersebut dengan memberikan solusi inovatif dan efektif.

Beberapa penelitian menunjukkan bahwa pemantauan terus-menerus terhadap parameter kesehatan tertentu, seperti detak jantung dan suhu tubuh, dapat memberikan informasi berharga tentang kondisi kesehatan seseorang. Melalui integrasi teknologi Internet of Things (IoT), proyek ini bertujuan untuk memberikan solusi yang praktis dan mudah digunakan untuk pemantauan kesehatan pribadi.

Selain itu, proyek ini dapat memberikan kontribusi positif terhadap upaya pencegahan dan manajemen penyakit kronis. Melalui pemantauan detak jantung dan suhu tubuh secara terus-menerus, user dapat mendeteksi anomali atau perubahan yang dapat menjadi indikator potensial masalah kesehatan. Hal ini dapat meningkatkan kesadaran user tentang kondisi kesehatan mereka dan memungkinkan tindakan pencegahan yang diperlukan.

Demikian laporan ini dibuat dengan penuh kesungguhan dan tanggung jawab. Kami berharap laporan ini dapat memberikan gambaran menyeluruh mengenai berbagai aspek dalam pengembangan proyek ini. Semoga laporan ini dapat memberikan pemahaman mendalam tentang konsep, teknologi yang digunakan, dan manfaat yang diharapkan dari proyek "Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking" ini.

Depok, 28 November 2023

Group B3

## TABLE OF CONTENTS

<b>CHAPTER 1.....</b>	<b>4</b>
<b>INTRODUCTION.....</b>	<b>4</b>
1.1    PROBLEM STATEMENT.....	4
1.3    ACCEPTANCE CRITERIA.....	5
1.4    ROLES AND RESPONSIBILITIES.....	5
1.5    TIMELINE AND MILESTONES.....	5
<b>CHAPTER 2.....</b>	<b>7</b>
<b>IMPLEMENTATION.....</b>	<b>7</b>
2.1    HARDWARE DESIGN AND SCHEMATIC.....	7
2.2    SOFTWARE DEVELOPMENT.....	7
2.3    HARDWARE AND SOFTWARE INTEGRATION.....	8
<b>CHAPTER 3.....</b>	<b>9</b>
<b>TESTING AND EVALUATION.....</b>	<b>9</b>
3.1    TESTING.....	9
3.2    RESULT.....	9
3.3    EVALUATION.....	10
<b>CHAPTER 4.....</b>	<b>11</b>
<b>CONCLUSION.....</b>	<b>11</b>

## CHAPTER 1

### INTRODUCTION

#### 1.1 PROBLEM STATEMENT

Di era modern ini, di mana kehidupan manusia berjalan secara cepat, perhatian terhadap pemantauan kesehatan seringkali kurang diperhatikan, menyebabkan penundaan dalam deteksi dini masalah kesehatan dan manajemen kondisi kronis. Masalah utama yang perlu diatasi adalah kurangnya alat pemantauan kesehatan yang praktis, akurat, dan dapat diakses secara real-time. Deteksi awal kondisi kesehatan seperti denyut jantung dan suhu tubuh dapat menjadi kunci untuk pencegahan penyakit dan manajemen kesehatan yang lebih efektif. Oleh karena itu, pengembangan *Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking* yang efisien dan mudah digunakan adalah sebuah proyek yang penting dan bermanfaat bagi banyak masyarakat.

Smart Health Monitoring System adalah alat yang dirancang untuk secara kontinu dan real-time memantau detak jantung seseorang. Sistem ini umumnya memanfaatkan teknologi sensor dan perangkat IoT yang dapat digunakan pada user untuk mengukur detak jantung mereka tanpa intervensi manual.

Body Temperature Tracking System adalah alat yang dirancang untuk melakukan proses pengukuran dan pencatatan suhu tubuh seseorang secara berkala. Sistem pelacakan suhu tubuh pada proyek ini menggunakan perangkat chip tertentu sebagai termometer digital, yang diintegrasikan dengan perangkat IoT oleh software yang diinginkan.

Selain kemudahan dan efektivitas yang ditawarkan oleh proyek ini, hasil pemantauan detak jantung dan suhu tubuh ditampilkan secara real-time kepada user melalui display yang responsif dan mudah diakses seperti Thingsboard. Lalu, user dapat mengakses data kesehatan mereka dengan mudah melalui smartphone mereka yang terhubung langsung ke perangkat. Hal ini memberikan pengalaman yang lebih interaktif dan informatif bagi user yang ingin memeriksa kondisi kesehatan mereka.

## **1.2 PROPOSED SOLUTION**

Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking merupakan sebuah sistem pemantauan detak jantung dan suhu tubuh yang menggunakan mikrokontroler ESP32 sebagai inti komponen nya. Sensor MAX30100 merupakan sensor Heart Rate Oximeter Sensor berbasis modul Arduino yang dirancang mengukur denyut jantung (heart rate) dan kadar oksigen dalam darah (SpO2), tetapi dalam proyek ini kami menggunakan sensor ini hanya untuk pengukuran detak jantung.

Rangkaian proyek ini juga dilengkapi dengan sensor LM35 yang merupakan sensor suhu analog yang menghasilkan output tegangan yang linear terhadap suhu dalam derajat Celcius. Sensor LM35 ini memiliki tingkat presisi yang baik, membuatnya cocok untuk aplikasi yang membutuhkan pengukuran suhu yang akurat.

Selain itu, rangkaian proyek ini diintegrasikan dengan platform Thingsboard agar data pengukuran detak jantung dan suhu tubuh dapat ditampilkan secara real-time kepada user dan agar user dapat mengakses hasil pengukuran mereka dengan mudah melalui smartphone. Platform ini memungkinkan pembuatan dasbor interaktif dengan widget yang dapat disesuaikan, memungkinkan pengalaman pemantauan yang lebih mendalam dan interaktif. Platform ini juga mendukung protokol komunikasi IoT standar seperti MQTT agar memudahkan integrasi dengan perangkat IoT dan sistem yang mendukung protokol tersebut.

Tujuan utama dari proyek "Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking" adalah membangun sistem pemantauan kesehatan yang efektif dan mudah diakses. Proyek ini dirancang untuk memberikan solusi inovatif yang memungkinkan individu memantau detak jantung dan suhu tubuh secara real-time, mendeteksi dini potensi masalah kesehatan, dan meningkatkan pemahaman tentang kondisi kesehatan pribadi.

Proyek ini juga dapat menjadi inspirasi untuk pengembangan teknologi di masa depan yang dapat memperkuat kolaborasi antara inovasi kesehatan dan teknologi ramah lingkungan. Dengan demikian, diharapkan dapat mendorong pertumbuhan teknologi yang tidak hanya bermanfaat bagi kesehatan individu, tetapi juga berkontribusi pada upaya global untuk menciptakan dunia yang lebih hijau dan berkelanjutan.

### **1.3 ACCEPTANCE CRITERIA**

Berikut beberapa kriteria penerimaan pada proyek ini.

1. Mampu merancang dan membangun rangkaian *Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking* dengan menggunakan mikrokontroler ESP32 sebagai inti komponen berbasis modul Arduino..
2. Mampu merancang interface user yang interaktif dan mudah digunakan.
3. Mampu melakukan pengukuran detak jantung dengan sensor MAX30100.
4. Mampu melakukan pengukuran suhu tubuh dengan sensor LM35.
5. Mampu menampilkan bacaan detak jantung dan suhu tubuh melalui Serial Monitor pada Arduino IDE dan pada platform Thingsboard.
6. Mampu menyusun laporan proyek yang komprehensif dengan penjelasan rinci tentang desain, implementasi, dan hasil pengujian dari proyek.

### **1.4 ROLES AND RESPONSIBILITIES**

Berikut peran dan tanggung jawab untuk setiap anggota kelompok:

<b>Peran</b>	<b>Tanggung Jawab</b>	<b>Anggota</b>
Menyusun Rangkaian Fisik + Mengintegrasikan Komponen	Role 1 responsibilities	Lauren Christy T.
Merancang Thingsboard + Mengintegrasikan Komponen	Role 2 responsibilities	Zalfy Putra R.
Memprogram LM35 + Menyusun Laporan Proyek Akhir	Role 3 responsibilities	Rafi Fauzan W.
Memprogram MAX30100 + Menyusun PPT	Role 4 responsibilities	Mikhael Morris H.S.

## 1.5 TIMELINE AND MILESTONES

## CHAPTER 2

### IMPLEMENTATION

#### 2.1 HARDWARE DESIGN AND SCHEMATIC

Rangkaian *Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking* merupakan suatu sistem pemantauan kesehatan cerdas yang mampu secara real-time melacak detak jantung dan suhu tubuh dari user. Proyek ini bertujuan untuk memberikan informasi vital secara akurat dan instan untuk pemantauan kesehatan pribadi atau pengawasan medis. Untuk menerapkan desain proyek ini, diperlukan sejumlah komponen hardware yang meliputi ESP32, sensor MAX30100, dan sensor LM35.

ESP32 berfungsi sebagai komponen pusat yang menjadi otak utama dari proyek ini. Kemampuan ESP32 dalam mengelola konektivitas WiFi dan Bluetooth menjadi hal yang sangat penting dalam mengirimkan data ke platform cloud atau aplikasi mobile untuk menampilkan data ke user melalui antarmuka yang interaktif.

MAX30100 adalah sensor yang dirancang khusus untuk aplikasi pengukuran detak jantung dan saturasi oksigen dalam darah (SpO<sub>2</sub>). Sensor oximeter ini bekerja dengan bus I2C, yaitu menghubungkan pin I2C (SCL & SDA) dari modul oximeter dengan GPIO 21 dan GPIO 22 pada ESP32. Lalu, menyambungkan pin INT ke GPIO 19 pada ESP32 sebagai pin input/output analog untuk mengirimkan data.

LM35 adalah sensor suhu analog yang dirancang untuk mengukur suhu tubuh user dalam satuan derajat Celcius. Sensor LM35 memiliki tiga pin, yaitu pin VCC yang dihubungkan ke Vin, pin GND yang dihubungkan ke *ground*, dan pin OUT yang dihubungkan ke GPIO 36 atau ADC0 untuk pengiriman data hasil bacaan secara analog.

Hasil bacaan dari sensor MAX30100 (detak jantung) dan LM35 (suhu tubuh) dapat ditampilkan secara real-time melalui Serial Monitor pada Arduino IDE dan dikirim melalui internet ke aplikasi mobile menggunakan platform Thingsboard. Hal ini bertujuan agar memudahkan user untuk mengakses data bacaan mereka melalui *smartphone*.

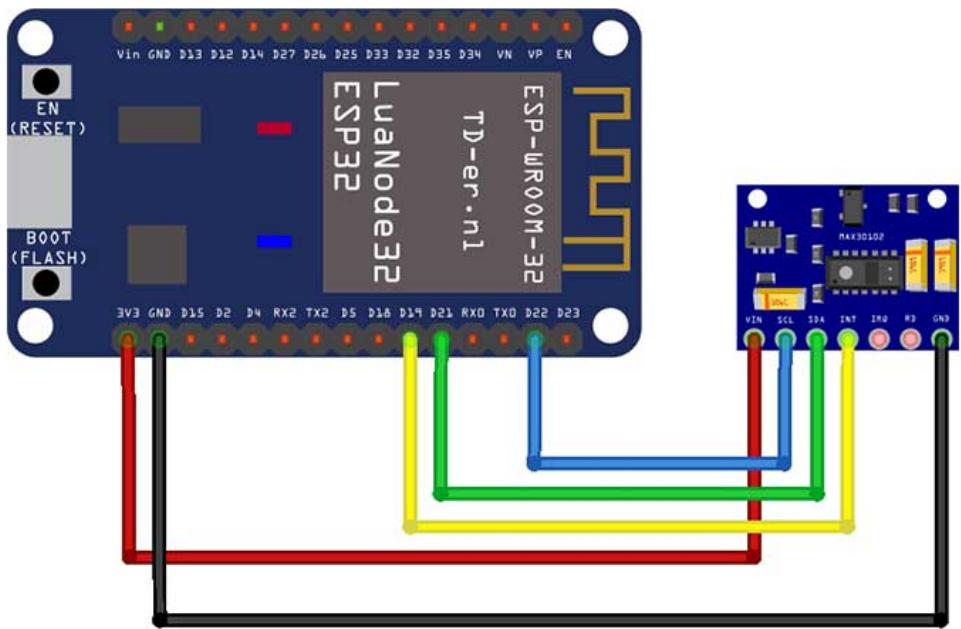


Figure 1. Schematic Design of ESP32 with MAX30100

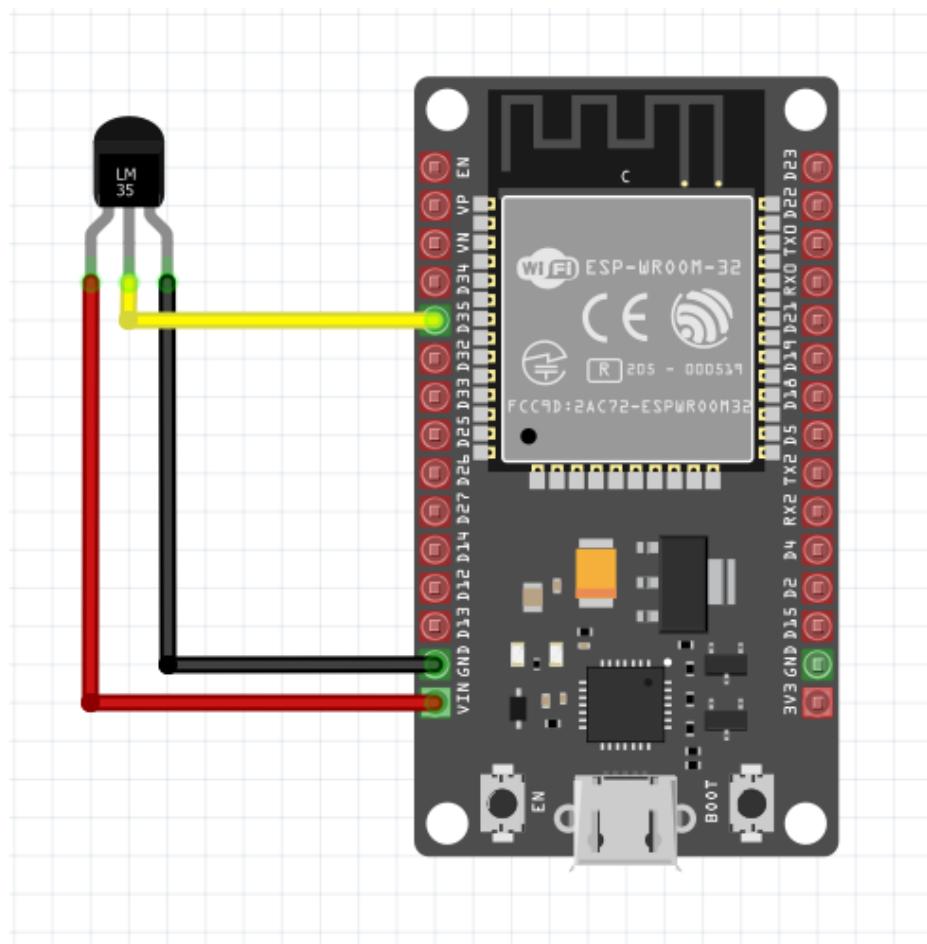


Figure 2. Schematic Design of ESP32 with LM35

## 2.2 SOFTWARE DEVELOPMENT

Software development pada project ini menggunakan bahasa C++ dimana akan mengimplementasikan modul-modul yang berkaitan dengan IoT. Disini kami menggunakan beberapa library, seperti:

```
#include <Wire.h>
#include <WiFi.h>
#include <ThingsBoard.h>
#include "Arduino_MQTT_Client.h"
#include "MAX30100_PulseOximeter.h"
```

- Wire, merupakan library yang berfungsi untuk mengakses protocol I2C pada ESP32.
- Wifi, merupakan library yang berfungsi untuk mengakses WiFi pada ESP32.
- ThingsBoard, merupakan library yang berfungsi untuk menghubungkan ESP32 dengan Thingsboard.
- Arduino\_MQTT\_Client, merupakan library yang berfungsi untuk mengakses dan implementasi dari protokol MQTT pada ESP32.
- MAX30100\_PulseOximeter, merupakan library yang berfungsi untuk terhubung dan membaca data dari sensor MAX30100 . .

Implementasi dari sisi software akan dibagi menjadi beberapa task yang akan menjalankan tasknya masing-masing. Terdapat 4 task yang dibuat yang masing-masing meng-handle Wifi, sensor LM35, sensor MAX30100.

### 1. TaskWifi

Task ini akan meng-handle proses penyambungan ESP32 terhadap jaringan Wifi. SSID dan password akan didefinisikan terlebih dahulu pada variabel sehingga credential wifi dapat disimpan dan dapat langsung terhubung.

```
// Put your SSID & Password
const char* ssid = "";          // Enter SSID here
const char* password = "";      // Enter Password here
```

Pada proses penyambungan, pertama akan dilakukan inisiasi WiFi dengan memanfaatkan built-in function yang berada pada library WiFi dan SSID dan Password akan digunakan. Lalu dibuat while loop yang berfungsi untuk mengecek

apakah ESP32 sudah terhubung dengan WiFi dan juga for loop yang berguna ketika WiFi belum berhasil terhubung maka akan dilakukan percobaan untuk menghubungkannya hingga koneksi berhasil dilakukan.

```
void WiFiTaskCode(void * parameter) {  
    // Connect to Wi-Fi  
  
    WiFi.begin(ssid, password);  
  
    while (WiFi.status() != WL_CONNECTED) {  
        vTaskDelay(1000 / portTICK_PERIOD_MS); // Delay for 1 second  
  
        Serial.println("Connecting to WiFi...");  
    }  
  
    Serial.println("Connected to WiFi");  
  
  
    for(;;) {  
        // Check if WiFi is connected, if not, then reconnect  
  
        while(WiFi.status() != WL_CONNECTED) {  
            Serial.println("Reconnecting WiFi...");  
  
            WiFi.reconnect();  
  
            vTaskDelay(5000 / portTICK_PERIOD_MS); // Delay for 5 seconds  
        }  
  
        vTaskDelay(3000 / portTICK_PERIOD_MS); // Delay for 3 seconds  
    }  
}
```

## 2. TaskMAX

Task ini berfungsi untuk mengambil data dari hasil sensor reading pada sensor MAX30100. Disini akan menggunakan penerapan dari function built-in pada library MAX30100. Seperti `pox.update()` untuk melakukan update data hasil reading sensor secara terus menerus, kemudian data tersebut disimpan pada variabel BPM dan SpO2.

getHeartRate() dan getSpO2 merupakan function untuk mendapatkan data heart rate dan kadar saturasi oksigen dalam darah dari sensor.

```
void MAX30100SensorReading(void * parameter) {  
    while(1){  
        if(xSemaphoreTake(oximeterMutex, portMAX_DELAY)){  
            pox.update();  
            BPM = pox.getHeartRate();  
            SpO2 = pox.getSpO2();  
  
            xSemaphoreGive(oximeterMutex);  
        }  
        vTaskDelay(10 / portTICK_PERIOD_MS);  
    }  
}
```

### 3. TaskLM

Task ini berfungsi untuk meng-handle sensor reading dari LM35. Metode pembacaan data dari sensor ini berbeda dengan MAX30100 dimana memiliki library sendiri sehingga lebih sederhana. Pembacaan data dari sensor LM35 didapat melalui nilai analog yang telah di-convert menjadi digital menggunakan ADC. Secara cara kerjanya, pada LM35 akan terjadi kenaikan sebesar 10 mV pada tiap satu derajat. Untuk itu kita perlu melakukan konversi dari hasil ADC ke miliVolt yang dapat dicari dengan menggunakan rumus:

$$miliVolt = ADCVal * \frac{ADC\ VREF}{ADC\ Resolution}$$

```
#define ADC_VREF_mV          3300.0           // in  
millivolt  
  
#define ADC_RESOLUTION       4096.0          // 12-bit  
ADC
```

Setelah didapatkan miliVolt-nya, maka kita bisa melakukan konversi menjadi celcius dimana dapat dicari dengan membagi total miliVolt yang didapat dengan 10. Hal ini dikarenakan tiap kenaikan/penurunan derajat akan terjadi perubahan sebesar 10 mV. Kemudian kita akan memberikan opsi lain seperti ukuran suhu dalam bentuk fahrenheit dan kedua data tersebut akan disimpan dalam variabel sehingga dapat digunakan.

```
void LM35SensorReading(void * parameter) {  
  
    for(;;) {  
  
        int adcVal = analogRead(PIN_LM35); // Read the ADC value from  
sensor  
  
        float milliVolt = adcVal * (ADC_VREF_mV / ADC_RESOLUTION); //  
Convert the ADC value to voltage in millivolt  
  
        float tempC = milliVolt / 10; // Convert the voltage to the  
temperature in °C  
  
        float tempF = tempC * 9 / 5 + 32; // convert the °C to °F  
  
  
        Temp temp;  
  
        temp.tempC = tempC;  
  
        temp.tempF = tempF;  
  
  
        // Mengirim data suhu ke dalam antrian  
  
        if (xQueueSend(tempQueue, &temp, portMAX_DELAY) == pdPASS) {  
  
            // Serial.println("Data suhu berhasil dikirim ke  
antrian.");  
  
        } else {  
  
            Serial.println("Gagal mengirim data suhu ke antrian.");  
  
        }  
  
        vTaskDelay(10 / portTICK_PERIOD_MS);  
  
    }  
}
```

#### 4. ThingsboardTask

Task ini berfungsi untuk menampilkan data hasil reading sensor pada Thingsboard. Pertama akan dilakukan pengecekan terlebih dahulu apakah sistem sudah terhubung dengan Thingsboard, jika belum maka akan dilakukan percobaan untuk melakukan koneksi hingga berhasil. Kemudian data dari sensor MAX30100 diambil menggunakan mutex dan data sensor LM35 diambil menggunakan Queue. Kemudian data akan dikirimkan menuju Thingsboard dengan menggunakan function dari library Thingsboard, yaitu tb.sendTelemetryData();

```
void ThingsBoardTaskCode(void * parameter) {  
  
    for(;;) {  
  
        // Connect to ThingsBoard  
  
        while(!tb.connected()) {  
  
            Serial.println("Connecting to thingsboard...");  
  
            tb.connect(TB_SERVER, TB_TOKEN);  
  
            vTaskDelay(3000 / portTICK_PERIOD_MS); // Delay for 5  
seconds  
  
        }  
  
        vTaskDelay(1000 / portTICK_PERIOD_MS); // Delay for 5 seconds  
  
  
        if (xSemaphoreTake(oximeterMutex, portMAX_DELAY)) {  
  
            if (millis() - tsLastReport > REPORTING_PERIOD_MS) {  
  
                Serial.print("Heart Rate: ");  
  
                Serial.print(BPM);  
  
                Serial.print(" BPM");  
  
  
                Serial.print(" | SpO2: ");  
  
                Serial.print(SpO2);  
  
                Serial.println("%");  
  
            }  
  
            xSemaphoreGive(oximeterMutex);  
        }  
    }  
}
```

```

        tsLastReport = millis();

        xSemaphoreGive(oximeterMutex);

    }

    vTaskDelay(2000 / portTICK_PERIOD_MS);

}

Temp receivedTemp;

if (xQueueReceive(tempQueue, &receivedTemp, portMAX_DELAY) == pdPASS) {

    if (millis() - tsLastReport > REPORTING_PERIOD_MS) {

        Serial.print("Temperature: ");

        Serial.print(receivedTemp.tempC); // print the temperature in °C

        Serial.print("°C");

        Serial.print(" ~ "); // separator between °C and °F

        Serial.print(receivedTemp.tempF); // print the temperature in °F

        Serial.println("°F");

        tsLastReport = millis();

    }

}

// Sending data to ThingsBoard

tb.sendTelemetryData("temperature_c", receivedTemp.tempC);

tb.sendTelemetryData("temperature_f", receivedTemp.tempF);

tb.sendTelemetryData("heart_rate_bpm", BPM);

tb.sendTelemetryData("heart_rate_oxy", SpO2);

tb.loop();

} k

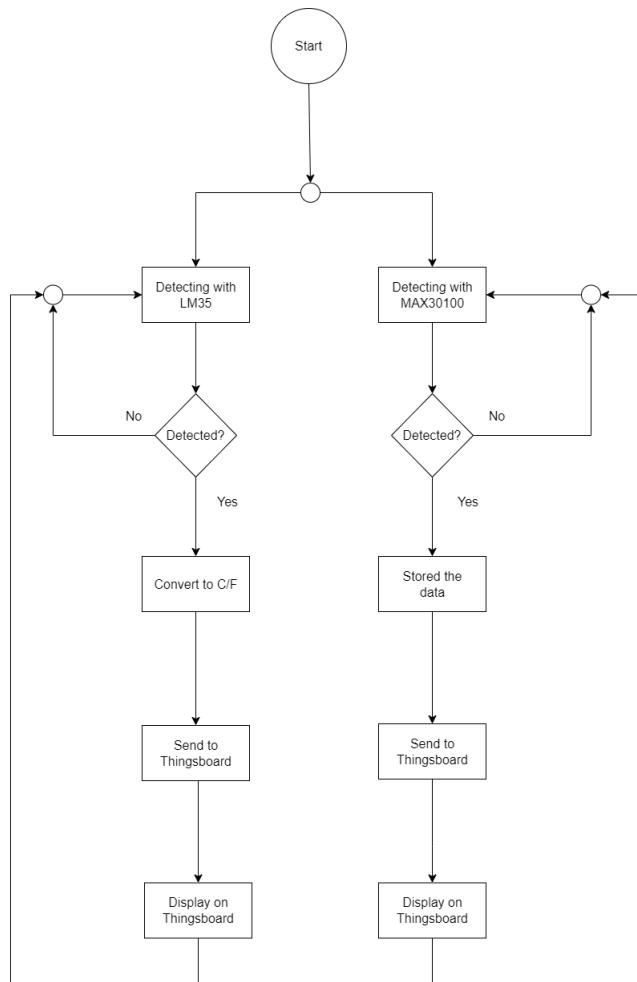
```

Queue diterapkan pada bagian sensor reading LM35 dengan tujuan agar tiap ada data baru yang dibaca oleh sensor akan langsung masuk kedalam queue dan langsung dikirimkan ke Thingsboard tanpa harus menunggu task lain mengakses data tersebut sehingga data bisa secara terus menerus di-update secara real-time.

Mutex diterapkan pada sensor reading MAX30100. Tujuan dari penerapan mutex ini adalah untuk mengatur dalam mengakses data yang disimpan dalam bentuk variable. useran take and give digunakan agar memastikan tiap task yang menggunakan variable terkait dapat berjalan dengan baik dan tidak ada intervensi antar task yang bisa saja mengubah isi dari variabel tersebut. Selain itu, penerapan mutex merupakan salah satu tindakan preventif dalam mencegah terjadinya deadlock.

Dari sisi prioritas antar task, keempat task tersebut memakai prioritas yang sama. Hal ini dikarenakan semua task yang akan dijalankan bersifat krusial dan jika dilakukan pada prioritas yang berbeda bisa saja respons time-nya lebih lama.

### **Flowchart:**



Implementasi modul:

## 1. Modul 1 (Introduction to RTOS & Task Scheduling)

useran Task dalam proyek ini adalah untuk membagi section berdasarkan klasifikasinya masing-masing dan memungkinkan untuk eksekusi secara independen dan bersamaan. Task MAX berfokus pada sensor MAX30100 dalam pemantauan heart rate dan oxygen rate dalam darah. TaskLM berfokus pada sensor suhu LM35 untuk mengukur suhu tubuh dan mengonversinya ke dalam skala Celcius dan Fahrenheit. TaskWifi berfungsi untuk mengelola koneksi WiFi dan terus-menerus memastikan keberlanjutan koneksi. Sementara ThingsBoardTask berfungsi untuk menerima data hasil bacaan sensor dari TaskMAX dan TaskLM dan mengirimkannya ke interface Thingsboard.

```
53 TaskHandle_t TaskMAX;
54 TaskHandle_t TaskLM;
55 TaskHandle_t TaskWifi;
56 TaskHandle_t ThingsBoardTask;
```

## 2. Modul 2 (Memory Management & Queue)

useran queue dilakukan untuk mengatur dan mengirim data suhu antara TaskLM ke ThingsBoardTask untuk ditampilkan di interface Thingsboard. Queue didefinisikan pada fungsi setup dan membuat antrian dengan kapasitas maksimum 5 elemen, untuk menghemat memory (Memory Management), dengan setiap elemen memiliki size sebesar struktur data dari Temp yang adalah suhu dalam derajat Celcius dan Fahrenheit.

```
tempQueue = xQueueCreate(5, sizeof(Temp));
```

Dari TaskLM, akan dilakukan pembacaan suhu, setelah membaca suhu dan mengonversinya, data suhu dimasukkan ke dalam antrian menggunakan fungsi xQueueSend

```
if (xQueueSend(tempQueue, &temp, portMAX_DELAY) == pdPASS) {
```

Sementara dari sisi ThingsboardTask, akan diterima data suhu dari queue untuk dikirim ke interface Thingsboard

```
if (xQueueReceive(tempQueue, &receivedTemp, portMAX_DELAY) == pdPASS) {
    if (millis() - tsLastReport > REPORTING_PERIOD_MS){
        Serial.print("Temperature: ");
        Serial.print(receivedTemp.tempC); // print the temperature in °C
        Serial.print("°C");
        Serial.print(" ~ "); // separator between °C and °F
        Serial.print(receivedTemp.tempF); // print the temperature in °F
        Serial.print(" °F");
    }
}
```

### 3. Modul 3 (Mutex & Semaphore)

Dalam kode ini, mutex (mutual exclusion) digunakan untuk mengamankan critical section ketika akses secara bersamaan terjadi, terutama dalam pembacaan dan update data dari sensor MAX30100. Penggunaan mutex ini dilakukan dengan mendeklarasi variabel oximeterMutex. Mutex ini dibuat untuk melindungi variabel global yang berisi data BPM dan SpO2 saat task ini membaca dan memperbarui data dari sensor Pulse Oximeter.

```
oximeterMutex = xSemaphoreCreateMutex();
```

Ketika task MAX30100SensorReading memasuki critical section, yakni pembacaan dan pembaruan data sensor, mutex diambil menggunakan xSemaphoreTake(oximeterMutex). Hal ini memastikan bahwa tidak ada task lain yang dapat mengakses data yang sama pada saat bersamaan.

```
if (xSemaphoreTake(oximeterMutex, portMAX_DELAY)){
```

Setelah selesai, mutex dilepas menggunakan fungsi xSemaphoreGive(oximeterMutex), membebaskan akses bagi task-task lain yang menunggu.

```
xSemaphoreGive(oximeterMutex);
```

## 4. Modul 5 (Deadlock and Starvation)

Dalam kode ini, terdapat useran mutex (mutual exclusion) untuk melindungi critical section dari kode yang melibatkan akses dan pembaruan data bersamaan oleh beberapa task. Useran mutex bertujuan untuk mencegah

kondisi deadlock dan memastikan konsistensi data. Pada khususnya, mutex yang digunakan adalah oximeterMutex, yang melibatkan pembacaan dan pembaruan data dari sensor Pulse Oximeter MAX30100.

Melalui useran mutex , implementasi ini mencegah deadlock yang dapat terjadi ketika dua atau lebih task saling menunggu satu sama lain untuk melepaskan mutex yang mereka pegang.

## 5. Modul 6 (Priority Inversion & Multicore Systems)

Pada kode ini, konsep multicore diaplikasikan melalui useran fungsi xTaskCreatePinnedToCore. Fungsi ini digunakan untuk menciptakan dan menjalankan task-task secara bersamaan pada core yang ditentukan. Sebagai contoh, task pembacaan dari sensor Pulse Oximeter MAX30100 (MAX30100SensorReading) ditempatkan pada core 0, sementara task pembacaan dari sensor suhu LM35 (LM35SensorReading) ditempatkan pada core 1. Dengan demikian, task-task ini dapat beroperasi secara paralel, meningkatkan kinerja sistem secara keseluruhan.

```
xTaskCreatePinnedToCore(  
    WiFiTaskCode,  
    "WiFiTask",  
    10000,  
    NULL,  
    1,  
    &TaskWifi,  
    0  
);
```

```
xTaskCreatePinnedToCore(  
    LM35SensorReading,  
    "LM35SensorReading",  
    10000,  
    NULL,  
    1,  
    &TaskLM,  
    1  
);
```

## 6. Modul 8 (WiFi, HTTP, & MQTT)

Dalam kode ini, WiFi diintegrasikan melalui task TaskWifi. Task ini bertanggung jawab untuk menginisialisasi koneksi Wi-Fi, mengelola koneksi ulang jika diperlukan, dan memastikan konsistensi koneksi ke jaringan. Dengan ini, device dapat terus terhubung ke jaringan, memfasilitasi pengiriman data ke ThingsBoard dan menjaga fungsionalitas sistem secara keseluruhan.

```
void WiFiTaskCode(void * parameter) {
    // Connect to Wi-Fi
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        vTaskDelay(1000 / portTICK_PERIOD_MS); // Delay for 1 second
        Serial.println("Connecting to WiFi...");
    }
    Serial.println("Connected to WiFi");
```

Implementasi MQTT dalam kode ini menggunakan library Arduino\_MQTT\_Client untuk mengelola komunikasi antara ESP32 dan broker MQTT, yang dalam konteks ini adalah ThingsBoard.

```
Arduino_MQTT_Client mqttClient(wifiClient);

// Initialize ThingsBoard instance
ThingsBoard tb(mqttClient, 256U);
```

## 7. Modul 9 (Blynk)

Implementasi ThingsBoard dalam kode ini memungkinkan ESP32 berkomunikasi dengan ThingsBoard dan mengirimkan data secara real-time. Pada tahap awal, koneksi ke ThingsBoard diinisialisasi melalui objek ThingsBoard yang menggunakan klien MQTT sebagai parameter. Kode ini mencakup konfigurasi seperti server ThingsBoard (TB\_SERVER) dan token akses device (TB\_TOKEN), yang memberikan identifikasi ke perangkat dan mengizinkannya untuk berkomunikasi dengan ThingsBoard.

```
// ThingsBoard configuration
#define TB_SERVER          "thingsboard.cloud"
#define TB_TOKEN           "xSvHRJysdZ8Czi7RhBn2"
#define REPORTING_PERIOD_MS 1000
#define ADC_VREF_mV         3300.0
#define ADC_RESOLUTION       4096.0
```

Implementasi ThingsBoard dalam kode ini menciptakan interface antara ESP32 dan ThingsBoard, memungkinkan monitoring dan analisis data kesehatan secara real-time.

```
void ThingsBoardTaskCode(void * parameter) {
    for(;;) {
        // Connect to ThingsBoard
        while(!tb.connected()){
            Serial.println("Connecting to thingsboard...");
            tb.connect(TB_SERVER, TB_TOKEN);
        }
    }
}
```

Dalam proyek ini, kami memilih untuk menggunakan Thingsboard daripada Blynk dikarenakan preferensi fitur yang lebih bervariasi pada Thingsboard dan platformnya yang berbasis web sehingga tidak perlu untuk menginstall aplikasinya dari device yang digunakan untuk monitoring. Selain itu, untuk fitur user interfacenya lebih bervariasi dan cepat diimplementasikan. Thingsboard pun memiliki sinkronisasi yang cepat antar device dalam satu account, sehingga jika user login ke beberapa device, tidak diperlukan setup ulang untuk dashboard pada setiap perangkat, karena pada konfigurasi Thingsboard sudah dilakukan sinkronisasi otomatis pada device yang terhubung pada account tersebut.

## 2.3 HARDWARE AND SOFTWARE INTEGRATION

Pada perancangan Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking, kami menggunakan software dan hardware yang terintegrasi agar dapat saling terhubung dan berjalan satu sama lain. ESP32 berperan sebagai pengendali utama sistem, bertanggung jawab untuk membaca data dari sensor, mengolahnya, dan mengirimkannya ke platform ThingsBoard.

Dalam melakukan integrasi hardware, sensor MAX30100 Pulse Oximeter digunakan untuk mengukur detak jantung (BPM) dan tingkat oksigen dalam darah (SpO<sub>2</sub>). Komunikasi antara ESP32 dan sensor MAX30100 menggunakan protokol I2C. Data sensor dibaca dalam task terpisah (MAX30100SensorReading) dan dilindungi oleh mutex (oximeterMutex) untuk memastikan akses data yang aman. Sensor suhu LM35 juga digunakan untuk mengukur suhu tubuh. Sensor ini terhubung ke pin analog pada ESP32, dan data suhu dibaca dalam task terpisah (LM35SensorReading). Data suhu kemudian ditempatkan dalam sebuah antrian (tempQueue) untuk diproses oleh task ThingsBoard.

Dalam melakukan integrasi software, penggunaan FreeRTOS memungkinkan manajemen eksekusi task secara bersamaan. Berbagai task dibuat untuk tujuan yang berbeda, seperti membaca data sensor, mengelola koneksi WiFi, dan berkomunikasi dengan ThingsBoard. Setiap task berjalan secara independen pada inti tertentu dari ESP32. Mutex (oximeterMutex) digunakan untuk melindungi sumber daya bersama, yaitu variabel BPM dan SpO2, yang diakses oleh task pembacaan sensor MAX30100 dan task ThingsBoard. Ini memastikan konsistensi data selama akses bersama. Antrian (tempQueue) digunakan untuk mengirimkan data suhu dari task pembacaan sensor LM35 ke task ThingsBoard. Antrian memfasilitasi komunikasi antar task dalam lingkungan multitasking.

Integrasi dengan ThingsBoard dilakukan melalui platform IoT ThingsBoard, yang digunakan untuk menyimpan dan memvisualisasikan data pemantauan kesehatan. Library ThingsBoard digunakan untuk membangun koneksi dan mengirimkan data. Task ThingsBoard (ThingsBoardTask) bertanggung jawab atas komunikasi dengan ThingsBoard, mengirimkan data suhu dan detak jantung secara berkala.

## **CHAPTER 3**

### **TESTING AND EVALUATION**

#### **3.1 TESTING**

Proyek Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking dikembangkan menggunakan perakitan ESP32 untuk menciptakan sistem otomatis untuk memonitor kesehatan user dengan menggunakan parameter detak jantung serta suhu tubuh. Untuk memastikan keandalan dan fungsionalitas sistem, pengujian yang ketat dilakukan pada berbagai tahap pengembangan. Bagian ini menguraikan metodologi pengujian yang digunakan dan hasil yang diperoleh.

##### **Unit Testing**

Unit testing dilakukan pada komponen-komponen individu dari Sistem Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking untuk memverifikasi fungsi mereka dengan benar. Setiap modul, seperti sensor kelembaban, sensor pengukur suhu tubuh LM35 serta sensor pengukur detak jantung MAX30100 , diuji secara terpisah menggunakan kasus uji yang sesuai. Pengujian ini memastikan bahwa setiap komponen dikalibrasi dengan benar dan menghasilkan pembacaan yang akurat.

##### **Integration Testing**

Integration testing bertujuan untuk menilai integrasi yang mulus dari semua komponen dan modul individu dari Sistem Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking. Tujuannya adalah untuk memastikan bahwa aliran data dan komunikasi antara berbagai modul berfungsi seperti yang diharapkan. Kasus uji dirancang untuk mensimulasikan skenario-skenario yang berbeda. Sistem diuji untuk stabilitas, responsivitas, dan akurasi data selama skenario-skenario ini.

Dalam integration testing, kami menguji sistem dengan mengintegrasikan hanya sensor LM35 dan MAX30100 untuk melihat apakah kedua komponen tersebut dapat melakukan pengukuran suhu badan untuk LM35 serta pengukuran detak jantung MAX30100 dengan benar dan apakah dapat dijadikan parameter untuk kesehatan user.

### 3.2 RESULT

Semua komponen individu berhasil melewati fase pengujian unit dengan sukses. Sensor pengukur suhu badan serta sensor pengukur detak jantung memberikan pembacaan mengenai suhu tubuh dan juga detak jantung.

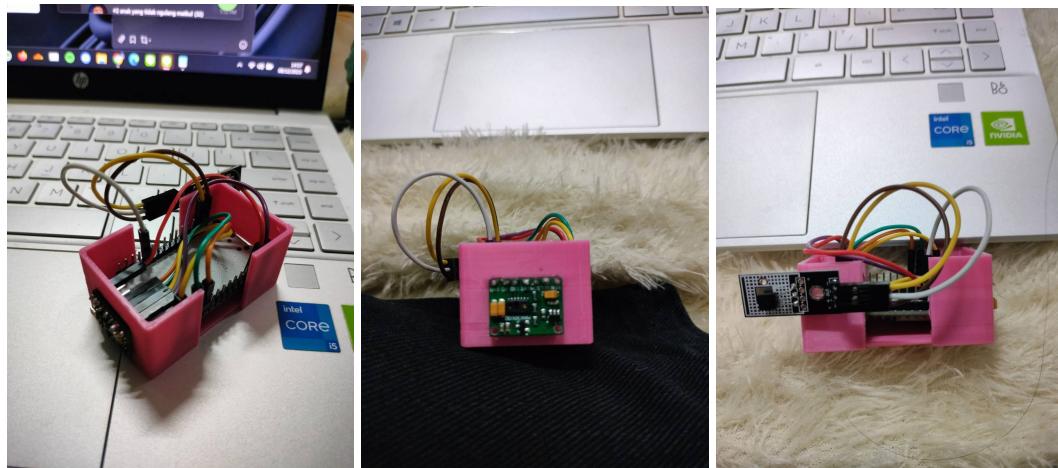


Fig 3.1 Testing Result

Fase pengujian integrasi memverifikasi bahwa komponen-komponen dari Sistem Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking terintegrasi secara efektif. Aliran data antara modul-modul berjalan dengan lancar, dan sistem memberikan pembacaan mengenai suhu tubuh serta detak jantung yang akurat dan real-time

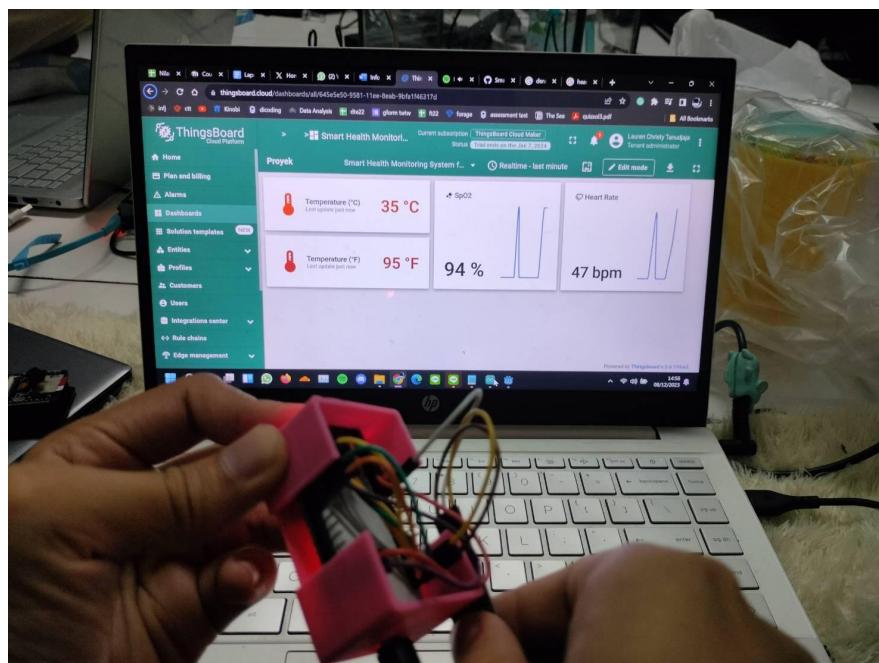


Fig 3.2 (a) Integration and Result on Desktop Displays

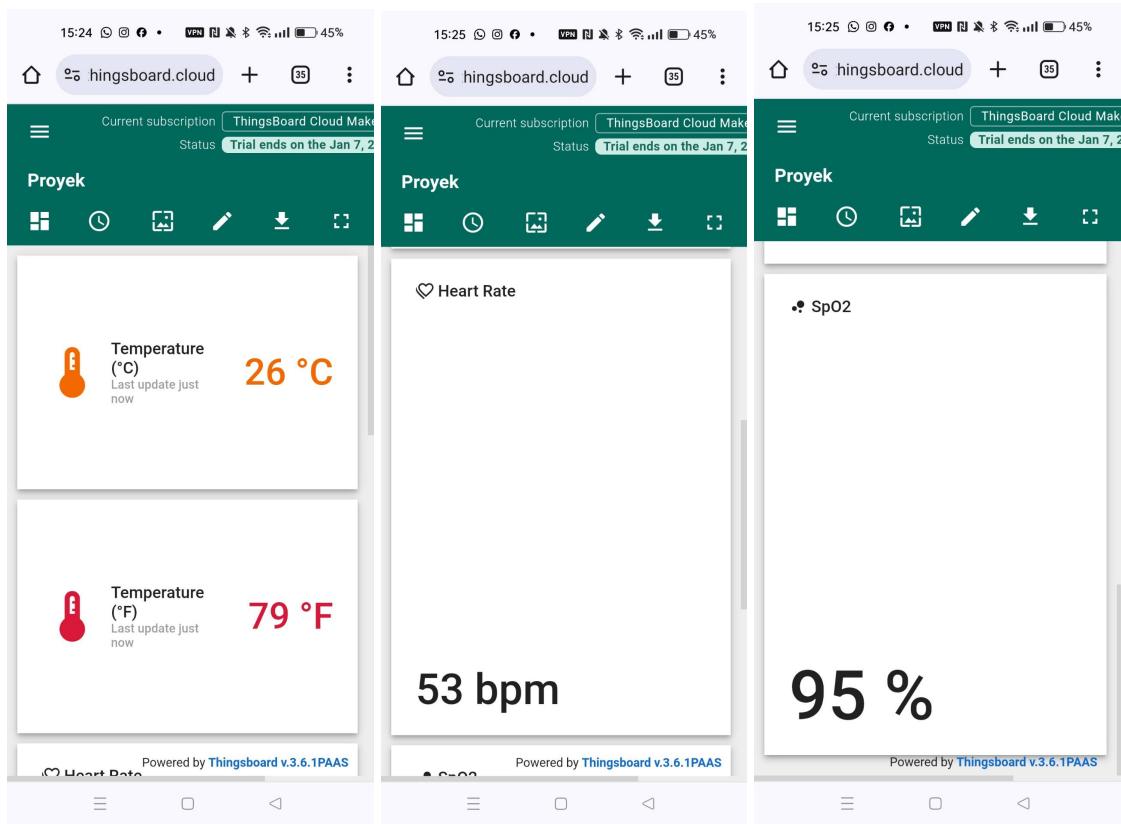


Fig 3.2 (a) Integration and Result on Mobile Displays

User acceptance testing menunjukkan bahwa sistem, ketika semua komponen terhubung dan digunakan bersama-sama, berhasil menghasilkan pembacaan yang benar untuk suhu badan serta detak jantung yang nantinya akan ditampilkan di dalam user interface yang menggunakan software thingsboard.

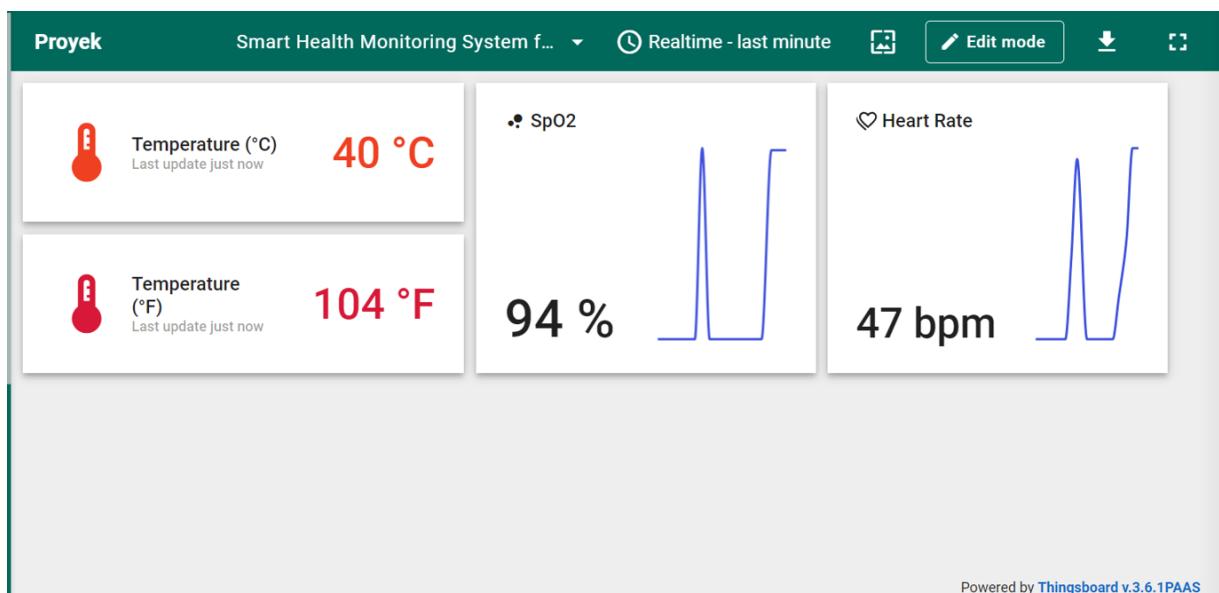


Fig 3.3 Integration and Full Result

Secara keseluruhan, proses pengujian untuk Sistem Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking melibatkan unit testing, integration testing, dan pengujian penerimaan user User acceptance testing. Sistem berhasil melewati semua tahap pengujian dengan sukses, menunjukkan pembacaan yang akurat, integrasi yang mulus, kinerja yang efisien, dan umpan balik positif dari user. Pengujian ini menjamin keandalan dan fungsionalitas pada sistem yang telah dirancang.

### **3.3 EVALUATION**

Proyek ini berhasil mengambil data dari bacaan kedua sensor dan mengirimkannya secara real-time ke platform Thingsboard untuk keperluan *user monitoring*. Namun, kedua sensor tidak dapat mengukur data secara konsisten sehingga seringkali ada *delay* antara pengambilan bacaan data dengan pengiriman data ke Thingsboard. Hal ini disebabkan oleh useran sensor yang lebih murah yang dapat menimbulkan pembacaan data yang kurang konsisten dan kurang akurat setiap kali user meletakkan jari ke sensor. Meskipun begitu, sistem tetap dapat menampilkan informasi heart rate, SpO<sub>2</sub>, dan suhu tubuh user pada platform Thingsboard

## **CHAPTER 4**

### **CONCLUSION**

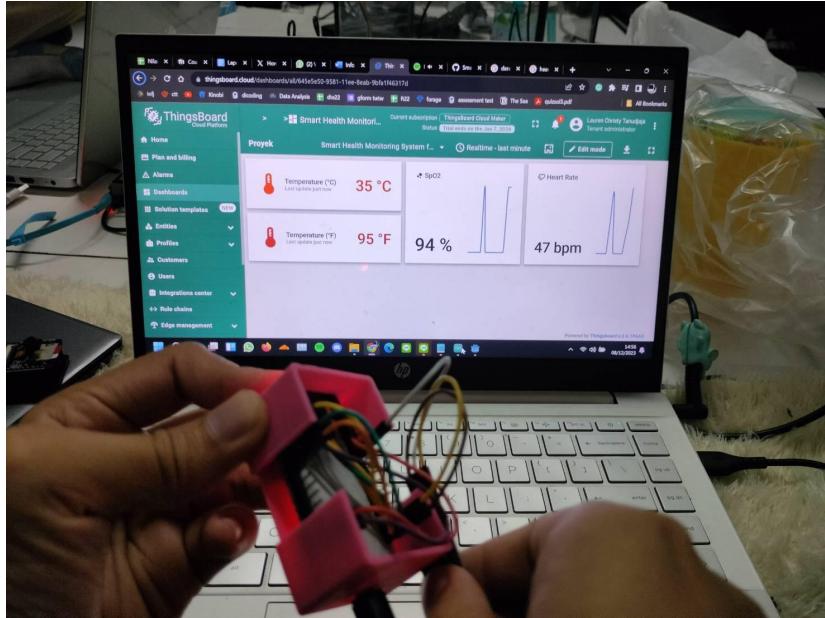
Smart Health Monitoring System for Real-time Heart Rate and Body Temperature Tracking ini bertujuan untuk mengembangkan sistem pemantauan kesehatan pintar untuk pemantauan detak jantung dan suhu tubuh secara real-time menggunakan teknologi IoT. Proyek ini berhasil mengintegrasikan komponen hardware dan software seperti mikrokontroler, sensor, dan perangkat IoT untuk menciptakan solusi yang praktis dan mudah digunakan. Data yang dihasilkan dapat diakses melalui antarmuka user yang interaktif, termasuk aplikasi smartphone, yang memungkinkan user untuk memantau kondisi kesehatan mereka dan mengambil tindakan pencegahan jika diperlukan. Proyek ini juga memberikan kontribusi pada pengembangan teknologi IoT dan perangkat pemantauan kesehatan, yang dapat berkolaborasi untuk menciptakan masa depan yang lebih sehat dan berkelanjutan. Dengan demikian, proyek ini berhasil mencapai tujuannya dan dapat menjadi inspirasi untuk pengembangan teknologi kesehatan dan lingkungan di masa depan.

## REFERENCES

- [1] IoT Project Ideas. “ESP32 based MAX30100 Pulse Oximeter Webserver”. Iotprojectideas.com. [Online]. Available: <https://iotprojectsideas.com/esp32-based-max30100-pulse-oximeter-webserver/>. [Accessed: Nov. 28, 2023].
- [2] ESP32IO. “ESP32 LM35 Temperature Sensor”. Esp32io.com. [Online]. Available: <https://esp32io.com/tutorials/esp32-lm35-temperature-sensor>. [Accessed: Nov. 28, 2023].
- [3] Digital Laboratory. “Modul 1: Introduction to RTOS & Task Scheduling,” [Online]. Available:[https://emas2.ui.ac.id/pluginfile.php/3996927/mod\\_resource/content/3/Modul%201%20RTS-IoT\\_%20Introduction%20to%20RTOS%20%20Task%20Scheduling.pdf](https://emas2.ui.ac.id/pluginfile.php/3996927/mod_resource/content/3/Modul%201%20RTS-IoT_%20Introduction%20to%20RTOS%20%20Task%20Scheduling.pdf) . [Accessed: Dec. 9, 2023].
- [4] Digital Laboratory. “Modul 2: Memory Management & Queue,” [Online]. Available: [https://emas2.ui.ac.id/pluginfile.php/3997136/mod\\_resource/content/3/Modul%202%20RTS-IoT\\_%20Memory%20Management%20%20Queue.pdf](https://emas2.ui.ac.id/pluginfile.php/3997136/mod_resource/content/3/Modul%202%20RTS-IoT_%20Memory%20Management%20%20Queue.pdf) . [Accessed: Dec. 9, 2023].
- [5] Digital Laboratory. “Modul 3: Mutex & Semaphore,” [Online]. Available: [https://emas2.ui.ac.id/pluginfile.php/4068093/mod\\_resource/content/2/Modul%203%20RTS-IoT-%20Mutex%20%20Semaphore.pdf](https://emas2.ui.ac.id/pluginfile.php/4068093/mod_resource/content/2/Modul%203%20RTS-IoT-%20Mutex%20%20Semaphore.pdf) . [Accessed: Dec. 9, 2023].
- [6] Digital Laboratory. “Modul 5: Deadlock & Starvation,” [Online]. Available: [https://emas2.ui.ac.id/pluginfile.php/4089016/mod\\_resource/content/1/Modul%205%20RTS-IoT-%20Deadlock%20%20Starvation.pdf](https://emas2.ui.ac.id/pluginfile.php/4089016/mod_resource/content/1/Modul%205%20RTS-IoT-%20Deadlock%20%20Starvation.pdf) . [Accessed: Dec. 9, 2023].
- [7] Digital Laboratory. “Modul 6: Priority Inversion & Multicore Systems,” [Online]. Available:[https://emas2.ui.ac.id/pluginfile.php/4104862/mod\\_resource/content/1/Modul%206%20RTS-IoT-%20Priority%20Inversion%20%20Multicore%20Systems.pdf](https://emas2.ui.ac.id/pluginfile.php/4104862/mod_resource/content/1/Modul%206%20RTS-IoT-%20Priority%20Inversion%20%20Multicore%20Systems.pdf) . [Accessed: Dec. 9, 2023].
- [8] Digital Laboratory. “Modul 8 RTS-IoT: WiFi, HTTP, and MQTT,” [Online]. Available:[https://emas2.ui.ac.id/pluginfile.php/4159959/mod\\_resource/content/2/Modul%208%20RTS-IoT-%20WiFi%2C%20HTTP%2C%20and%20MQTT.pdf](https://emas2.ui.ac.id/pluginfile.php/4159959/mod_resource/content/2/Modul%208%20RTS-IoT-%20WiFi%2C%20HTTP%2C%20and%20MQTT.pdf) . [Accessed: Dec. 9, 2023].
- [9] Digital Laboratory. “Modul 9: Blynk ,” [Online]. Available: [https://emas2.ui.ac.id/pluginfile.php/4168627/mod\\_resource/content/1/Modul%209%20RTS-IoT\\_%20Blynk.pdf](https://emas2.ui.ac.id/pluginfile.php/4168627/mod_resource/content/1/Modul%209%20RTS-IoT_%20Blynk.pdf) . [Accessed: Dec. 9, 2023].

## APPENDICES

### Appendix A: Project Schematic



### Appendix B: Documentation

