



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

**KOSEKI (Kontrol/Otomatis Suhu/Kelembaban Kar Intelligent)**

**GROUP 9**

<b>Adam Bintang Arafah Poernomo</b>	<b>2206029273</b>
<b>Edgrant Henderson Suryajaya</b>	<b>2206025016</b>
<b>Fairuz Muhammad</b>	<b>2206814324</b>
<b>Reiki Putra Darmawan</b>	<b>2206062882</b>

## PREFACE

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga kami dapat menyelesaikan laporan proyek akhir yang berjudul “KOSEKI: Kontrol/Otomatis Suhu/Kelembaban Kar Intelligent” tepat pada waktunya. Laporan ini disusun untuk memberikan pemahaman kepada pembaca mengenai konsep, desain, implementasi, dan pengujian sistem robot mobil yang dirancang untuk eksplorasi area sulit dijangkau manusia dengan memanfaatkan teknologi modern.

Proyek ini merupakan implementasi dari pengetahuan dan keterampilan yang kami peroleh selama mengikuti mata kuliah Sistem Embedded dan IoT. Kami juga ingin menyampaikan terima kasih yang sebesar-besarnya kepada dosen pembimbing, [Nama Dosen], serta rekan-rekan tim proyek yang telah mendukung dan memberikan masukan selama proses pengembangan proyek ini.

Meskipun kami telah berusaha menyusun laporan dengan cermat, kami menyadari bahwa masih terdapat kekurangan dan kelemahan dalam laporan proyek akhir ini. Oleh karena itu, kami sangat mengharapkan masukan dan saran yang konstruktif dari pembaca untuk membantu kami meningkatkan kualitas laporan ini. Kami berharap laporan ini dapat memberikan manfaat dalam pengembangan ilmu pengetahuan dan teknologi, khususnya di bidang IoT untuk aplikasi di medan yang menantang. Terima kasih atas perhatian dan waktu yang diberikan.

Depok, December 11, 2024

Group 9

## 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.....	6
<b>CHAPTER 2.....</b>	<b>7</b>
<b>IMPLEMENTATION.....</b>	<b>7</b>
2.1 HARDWARE DESIGN AND SCHEMATIC.....	7
2.2 SOFTWARE DEVELOPMENT.....	8
2.3 HARDWARE AND SOFTWARE INTEGRATION.....	17
<b>CHAPTER 3.....</b>	<b>23</b>
<b>TESTING AND EVALUATION.....</b>	<b>24</b>
3.1 TESTING.....	24
3.2 RESULT.....	25
3.3 EVALUATION.....	26
<b>CHAPTER 4.....</b>	<b>27</b>
<b>CONCLUSION.....</b>	<b>27</b>

## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1 PROBLEM STATEMENT**

Sistem pengumpulan data lingkungan di area berbahaya atau sulit dijangkau manusia, seperti zona bencana atau gua, sering kali masih memiliki keterbatasan baik dari segi fleksibilitas maupun efisiensi. Solusi konvensional biasanya hanya mendukung satu mode operasi, yaitu navigasi otonom atau kendali manual, sehingga kurang optimal untuk digunakan dalam berbagai skenario eksplorasi. Hal ini dapat memperlambat proses pengumpulan data dan pengambilan keputusan yang kritis dalam situasi darurat.

Misalnya, dalam banyak situasi:

- Perangkat gagal menavigasi medan kompleks karena kurangnya sistem kendali manual yang fleksibel.
- Data suhu dan kelembaban tidak dapat diperoleh secara real-time akibat keterbatasan integrasi perangkat keras dan perangkat lunak.
- Pemantauan dan kendali jarak jauh tidak memungkinkan karena ketergantungan pada metode kontrol lokal.

Kondisi ini mengakibatkan efisiensi dan efektivitas eksplorasi menjadi tidak maksimal, serta membatasi potensi teknologi untuk memberikan solusi yang praktis di medan berbahaya.

#### **1.2 PROPOSED SOLUTION**

Solusi yang diajukan adalah sistem robot mobil berbasis IoT bernama KOSEKI (Kontrol/Otomatis Suhu/Kelembaban Kar Intelligent), yang mampu bernavigasi secara otonom untuk menghindari rintangan atau dikendalikan secara manual melalui aplikasi dashboard Blynk. Sistem ini dirancang untuk mengatasi keterbatasan pada metode eksplorasi tradisional di area sulit dijangkau dengan mengintegrasikan sensor dan teknologi komunikasi real-time.

Robot ini dilengkapi dengan sensor DHT22 untuk mengukur suhu dan kelembaban, serta sensor inframerah dan ultrasonik untuk mendukung navigasi otonom. Data lingkungan seperti suhu dan kelembaban dapat dipantau secara real-time melalui aplikasi Blynk, memungkinkan pengguna untuk mengambil keputusan berdasarkan informasi yang tersedia.

Sistem ini juga memungkinkan pengendalian manual jika navigasi otonom tidak memungkinkan, memastikan fleksibilitas operasional yang lebih baik di berbagai medan. Dengan kombinasi antara kemampuan pengumpulan data, navigasi cerdas, dan kendali manual, KOSEKI dapat menjadi solusi praktis untuk mengatasi tantangan eksplorasi di area berbahaya dan meminimalkan risiko terhadap manusia.

### **1.3 ACCEPTANCE CRITERIA**

Kriteria yang ingin dicapai untuk proyek ini adalah:

1. Program harus dapat berjalan tanpa adanya kesalahan
2. Robot mobil harus dapat mengumpulkan data suhu dan kelembaban menggunakan sensor DHT22 dengan andal.
3. Robot harus mendukung navigasi otonom menggunakan sensor inframerah dan ultrasonik untuk menghindari rintangan.
4. Kendali manual robot harus dapat diakses melalui dashboard Blynk.
5. Sistem harus mengintegrasikan perangkat keras dan lunak dengan mulus, memastikan transmisi data secara real-time ke server Blynk.
6. Robot harus beroperasi dalam jarak dan durasi baterai yang cukup untuk penggunaan di lapangan.

### **1.4 ROLES AND RESPONSIBILITIES**

Roles	Responsibilities	Person
Pembuatan Laporan	Membuat Laporan	Adam Bintang Arafah Poernomo

Pendesain skematik dan rangkaian asli dan pembuat program	Membuat hardware dan juga kode program	Edgrant Henderson Suryajaya
Membuat Program	Membuat kode program	Fairuz Muhammad
Pembuatan Laporan	Membuat Laporan	Reiki Putra Darmawan

Table 1. Roles and Responsibilities

## 1.5 TIMELINE AND MILESTONES

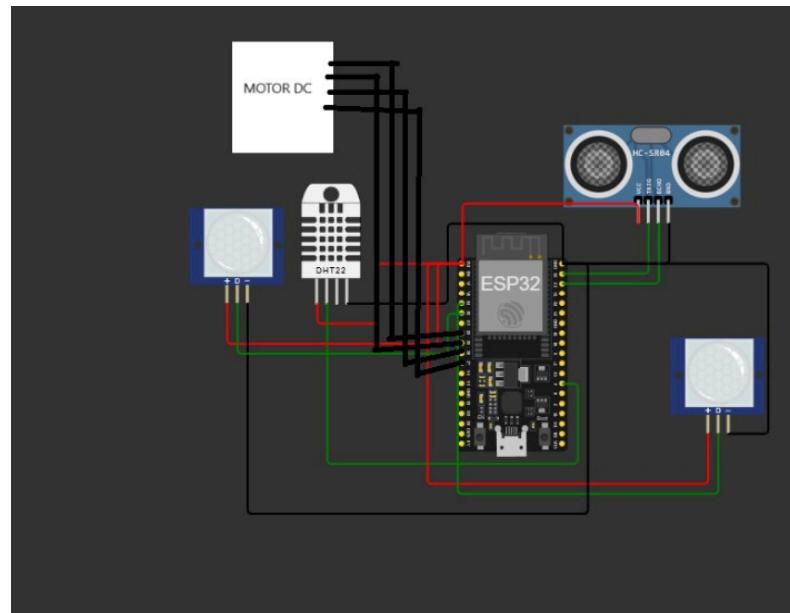
No	Kegiatan	November						Desember										
		25	26	27	28	29	30	1	2	3	4	5	6	7	8	9	10	11
1	Perumusan Ide																	
2	Finalisasi Ide																	
3	Hardware Desain																	
4	Pengembangan Software																	
5	Menyusun Rangkaian																	
6	Integrasi dan Uji coba Hardware dan Software																	
7	Perakitan dan Pengujian Produk Akhir																	



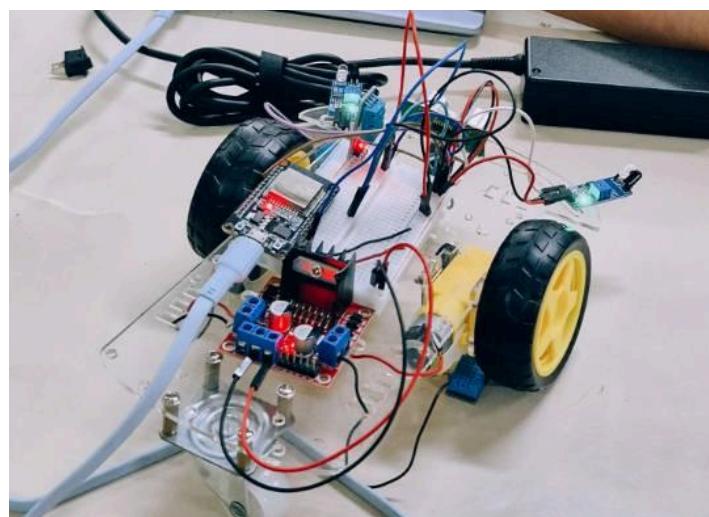
## CHAPTER 2

### IMPLEMENTATION

#### 2.1 HARDWARE DESIGN AND SCHEMATIC



*Figure 1. Schema*



*Figure 2. Hardware*

Untuk merancang sistem perangkat keras pada proyek KOSEKI, digunakan mikrokontroler ESP32 sebagai unit pusat pemrosesan data sensor dan komunikasi. ESP32 ini bertugas mengolah data dari beberapa sensor dan mengendalikan perangkat output seperti

motor dan modul PWM. Sensor DHT22 digunakan untuk mengukur suhu dan kelembaban udara, yang terhubung ke pin digital GPIO2 pada ESP32. Data yang dihasilkan dari sensor ini akan digunakan untuk pemantauan lingkungan sekitar robot.

Sensor inframerah terhubung ke pin GPIO15 pada ESP32 dan berfungsi untuk mendeteksi halangan di sekitar robot. Ketika sensor mendeteksi rintangan, ESP32 akan memproses sinyal tersebut untuk menghindari halangan dan menjaga pergerakan robot tetap lancar. Selain itu, sensor ultrasonik juga digunakan untuk mengukur jarak dengan presisi tinggi. Sensor ini terdiri dari dua pin, Trig dan Echo, yang masing-masing terhubung ke pin GPIO12 dan GPIO13 pada ESP32. Data jarak yang diperoleh akan membantu robot dalam menghindari objek di sekitarnya.

Motor DC digunakan untuk memberikan mobilitas pada robot, yang dikendalikan melalui motor driver seperti L298N atau L293D. Motor driver ini mengontrol arah dan kecepatan motor, dengan sinyal PWM yang dikirim dari pin GPIO18 dan GPIO19 pada ESP32. Modul PWM digunakan untuk mengatur kecepatan motor, memungkinkan robot bergerak dengan kecepatan yang dapat disesuaikan. Semua komponen ini terhubung ke sumber daya yang memadai, dengan memperhatikan kebutuhan tegangan dan arus masing-masing komponen agar sistem berfungsi dengan baik.

Desain perangkat keras ini memungkinkan robot untuk beroperasi secara mandiri dengan kemampuan untuk mendeteksi suhu, kelembaban, rintangan, dan jarak, serta menggerakkan motor untuk navigasi otomatis. Kombinasi dari sensor-sensor ini dan pengendalian motor membuat robot dapat beroperasi di area yang sulit dijangkau manusia, seperti di medan bencana atau di dalam gua.

## 2.2 SOFTWARE DEVELOPMENT

Berikut adalah source code untuk proyek kita:

```
/* Fill-in information from Blynk Device Info here */
#define BLYNK_TEMPLATE_ID "TMPL69TiULy6S"
#define BLYNK_TEMPLATE_NAME "Final Project IOT Group 5"
#define BLYNK_AUTH_TOKEN "jVextDFvO-MWXllCoVDrZ38SlrGaCeII"

#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClient.h>
```

```

#include <DHT.h>
#include <Ultrasonic.h>
#include <BlynkSimpleEsp32.h>

#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <freertos/queue.h>
#include <freertos/semphr.h>

// Motor pins
#define MOTOR1_PIN1 27
#define MOTOR1_PIN2 26
#define MOTOR2_PIN1 25
#define MOTOR2_PIN2 33

// IR Sensor pins
#define IR_SENSOR1 34
#define IR_SENSOR2 35

// Ultrasonic Sensor pins
#define ECHO_PIN 22
#define TRIG_PIN 23

// PWM Properties
#define PWM_FREQ 30000
#define PWM1_CHANNEL 0
#define PWM2_CHANNEL 1
#define PWM_RESOLUTION 8

// DHT Properties
#define DHT_PIN 4
#define DHT_TYPE DHT11
DHT dht(DHT_PIN, DHT_TYPE);
typedef struct
{
    float temperature;
    float humidity;
} DHTData;

// Ultrasonic Sensor
Ultrasonic ultrasonic(TRIG_PIN, ECHO_PIN);

/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial

// Forward declarations
void remoteControlTask(void *pvParameters);
void automaticControlTask(void *pvParameters);
void moveMotorTask(void *pvParameters);
void sendDHTDataTask(void *pvParameters);
void readSensorTask(void *pvParameters);

```

```
// Your WiFi credentials.
char ssid[] = ":"/;
char pass[] = "Amamyakokoro";

// Global variables
static int wirelessControl = 0;
static int remoteControl = 0;
static int manualControl = 0;
static int distance = 0;
static int irSensor1 = 0;
static int irSensor2 = 0;
static int readDHTbutton = 0;

// Queue
static QueueHandle_t xQueue;

// Manual Control VP
BLYNK_WRITE(V0)
{
    manualControl = param.asInt();
    xQueueSend(xQueue, &manualControl, portMAX_DELAY);
}

// Temperature VP
BLYNK_WRITE(V1)
{
}

// Humidity VP
BLYNK_WRITE(V2)
{
}

// Direction VP
BLYNK_WRITE(V3)
{
    wirelessControl = param.asInt();
}

// Manual DHT VP
BLYNK_WRITE(V4)
{
    readDHTbutton = param.asInt();
    xQueueSend(xQueue, &readDHTbutton, portMAX_DELAY);
}

void setup()
{
    // Debug console
    Serial.begin(115200);
```

```

Serial.println("Serial started");
dht.begin();
Serial.println("DHT started");

// Motor
pinMode(MOTOR1_PIN1, OUTPUT);
pinMode(MOTOR1_PIN2, OUTPUT);
pinMode(MOTOR2_PIN1, OUTPUT);
pinMode(MOTOR2_PIN2, OUTPUT);
pinMode(IR_SENSOR1, INPUT_PULLUP);
pinMode(IR_SENSOR2, INPUT_PULLUP);
pinMode(DHT_PIN, INPUT);

Serial.println("Motor setup");

Serial.println("PWM setup");

// Blynk
Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);

Serial.println("Blynk started");

// Queue
xQueue = xQueueCreate(8, sizeof(int));

// Task
xTaskCreatePinnedToCore(
    moveMotorTask, /* Task function. */
    "Move Motor", /* name of task. */
    10000,         /* Stack size of task */
    NULL,          /* parameter of the task */
    5,             /* max priority of the task */
    NULL,          /* Task handle */
    0);            /* core 0 */

Serial.println("Task created");

xTaskCreatePinnedToCore(
    sendDHTDataTask, /* Task function. */
    "Send DHT Data", /* name of task. */
    4096,           /* Stack size of task */
    NULL,           /* parameter of the task */
    1,              /* Low priority of the task */
    NULL,           /* Task handle */
    0);            /* core 0 */

Serial.println("Task created DHT");

xTaskCreatePinnedToCore(
    remoteControlTask, /* Task function. */
    "Remote Control", /* name of task. */

```

```

10240,           /* Stack size of task */
NULL,            /* parameter of the task */
4,               /* High priority of the task */
NULL,            /* Task handle */
0);              /* core 0 */

xTaskCreatePinnedToCore(
    readSensorTask, /* Task function. */
    "Read Sensor", /* name of task. */
    1024,           /* Stack size of task */
NULL,            /* parameter of the task */
3,               /* Very High priority of the task */
NULL,            /* Task handle */
0);              /* core 0 */

Serial.println("Task created Sensor");

xTaskCreatePinnedToCore(
    automaticControlTask, /* Task function. */
    "Automatic Control", /* name of task. */
    10240,             /* Stack size of task */
NULL,               /* parameter of the task */
2,                 /* Very High priority of the task */
NULL,               /* Task handle */
0);               /* core 0 */
}

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

void moveMotorTask(void *pvParameters)
{
    while (1)
    {
        // saat mendapat queue, motor akan bergerak sesuai dengan
queue yang diterima
        // xQueueReceive(xQueue, &remoteControl, portMAX_DELAY);
        // xQueueReceive(xQueue, &manualControl, portMAX_DELAY);
        Serial.println("Remote Control: " + String(remoteControl));

        if (remoteControl == 1)
        {
            // Maju
            digitalWrite(MOTOR1_PIN1, HIGH);
            digitalWrite(MOTOR1_PIN2, LOW);
            digitalWrite(MOTOR2_PIN1, HIGH);
            digitalWrite(MOTOR2_PIN2, LOW);
        }
        else if (remoteControl == 2)

```

```

    {
        // Mundur
        digitalWrite(MOTOR1_PIN1, LOW);
        digitalWrite(MOTOR1_PIN2, HIGH);
        digitalWrite(MOTOR2_PIN1, LOW);
        digitalWrite(MOTOR2_PIN2, HIGH);
    }
    else if (remoteControl == 3)
    {
        // Kanan
        digitalWrite(MOTOR1_PIN1, HIGH);
        digitalWrite(MOTOR1_PIN2, LOW);
        digitalWrite(MOTOR2_PIN1, LOW);
        digitalWrite(MOTOR2_PIN2, HIGH);
    }
    else if (remoteControl == 4)
    {
        // Kiri
        digitalWrite(MOTOR1_PIN1, LOW);
        digitalWrite(MOTOR1_PIN2, HIGH);
        digitalWrite(MOTOR2_PIN1, HIGH);
        digitalWrite(MOTOR2_PIN2, LOW);
    }
    else if (remoteControl == 5)
    {
        // Berhenti
        digitalWrite(MOTOR1_PIN1, LOW);
        digitalWrite(MOTOR1_PIN2, LOW);
        digitalWrite(MOTOR2_PIN1, LOW);
        digitalWrite(MOTOR2_PIN2, LOW);
    }

    // delay untuk menghindari motor bergerak terlalu cepat
    vTaskDelay(200 / portTICK_PERIOD_MS);
}
}

void sendDHTDataTask(void *pvParameters)
{
    while (1)
    {
        float temperature = dht.readTemperature();
        float humidity = dht.readHumidity();

        // Cek pembacaan
        if (isnan(temperature) || isnan(humidity))
        {
            Serial.println("Failed to read from DHT sensor!");
        }
        else
        {

```

```
        Serial.println("Temperature: " + String(temperature) + " C");
        Serial.println("Humidity: " + String(humidity) + " %");
        // Kirim data DHT ke Blynk
        Blynk.virtualWrite(V1, temperature);
        Blynk.virtualWrite(V2, humidity);
    }

    // delay 1 detik
    vTaskDelay(1000 / portTICK_PERIOD_MS);
}
}

void remoteControlTask(void *pvParameters)
{
    while (1)
    {
        // kalau manual control aktif, meneruskan nilai remote
control ke moveMotorTask
        if (manualControl == 1)
        {
            // Maju
            if (wirelessControl == 1)
            {
                remoteControl = 1;
            }
            // Mundur
            else if (wirelessControl == 2)
            {
                remoteControl = 2;
            }
            // Kanan
            else if (wirelessControl == 3)
            {
                remoteControl = 3;
            }
            // Kiri
            else if (wirelessControl == 4)
            {
                remoteControl = 4;
            }
            // Berhenti
            else if (wirelessControl == 5)
            {
                remoteControl = 5;
            }
        }

        // delay 10 ms
        vTaskDelay(100 / portTICK_PERIOD_MS);
    }
}
```

```

}

void readSensorTask(void *pvParameters)
{
    while (1)
    {
        // Serial.println("Reading sensor...");
        // Serial.println("Distance: " + String(distance) + " cm");
        // Serial.println("IR Sensor 1: " + String(irSensor1));
        // Serial.println("IR Sensor 2: " + String(irSensor2));

        // Baca sensor ultrasonik
        distance = ultrasonic.read(); // baca jarak dalam cm

        // Baca sensor IR
        irSensor1 = digitalRead(IR_SENSOR1);
        irSensor2 = digitalRead(IR_SENSOR2);

        // delay 100 ms
        vTaskDelay(200 / portTICK_PERIOD_MS);
    }
}

void automaticControlTask(void *pvParameters)
{
    while (1)
    {
        // kalau manual control tidak aktif, maka robot akan bergerak
        otomatis
        if (manualControl == 0)
        {
            // Jika sensor ultrasonik mendeteksi objek di depan
            if (distance < 10)
            {
                // Berhenti
                remoteControl = 5;
            }
            else
            {
                // Jika kedua sensor IR mendeteksi objek (IR Kiri dan
                Kanan)
                if (irSensor1 == 0 && irSensor2 == 0)
                {
                    // Mundur
                    remoteControl = 2;
                }
                // Jika sensor IR1 mendeteksi objek (IR Kiri)
                else if (irSensor1 == 0)
                {
                    // Kanan
                    remoteControl = 3;
                }
            }
        }
    }
}

```

```

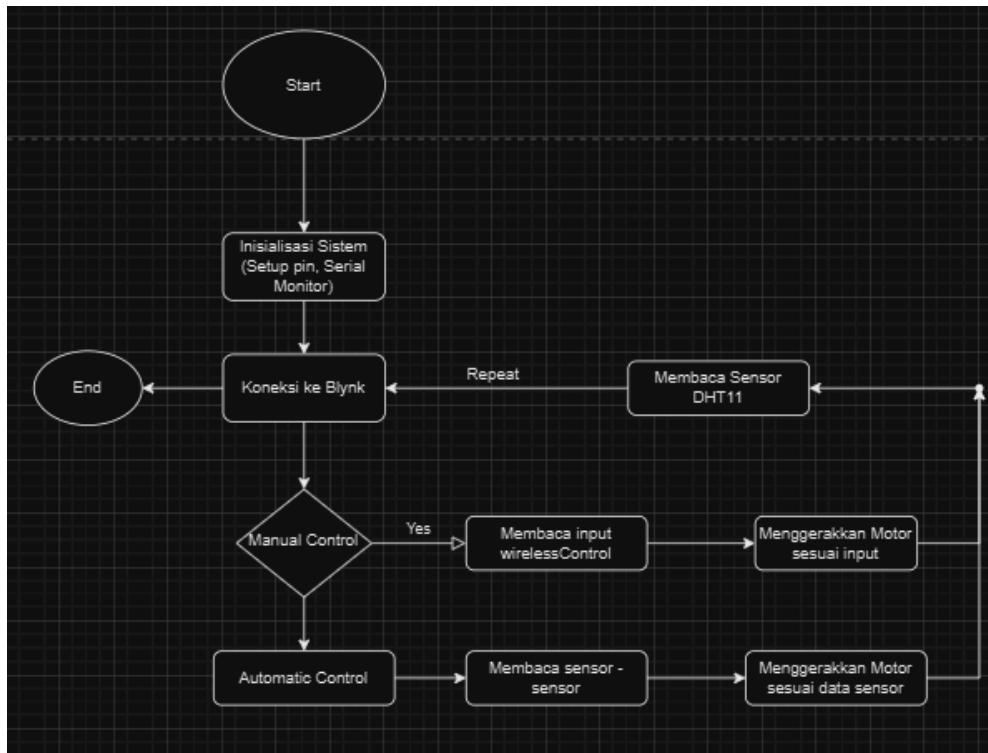
        }
        // Jika sensor IR2 mendeteksi objek (IR Kanan)
        else if (irSensor2 == 0)
        {
            // Kiri
            remoteControl = 4;
        }
        else
        {
            // Maju
            remoteControl = 1;
        }
    }

    // delay 10 ms
    vTaskDelay(10 / portTICK_PERIOD_MS);
}
}

```

Kode ini dilengkapi dengan sensor ultrasonik untuk mendeteksi jarak, sensor inframerah (IR) untuk deteksi rintangan, serta sensor DHT untuk membaca suhu dan kelembaban lingkungan. Sistem ini menggabungkan kontrol manual melalui aplikasi Blynk dan kontrol otomatis menggunakan algoritma yang dipicu oleh data sensor. Pada mode manual control, pengguna dapat mengontrol pergerakan robot (maju, mundur, kiri, kanan, atau berhenti) melalui aplikasi Blynk menggunakan virtual pin. Dalam mode automatic control, robot bergerak secara mandiri dengan logika berbasis kondisi sensor. Jika objek terdeteksi di depan (dengan jarak tertentu oleh sensor ultrasonik) atau jika rintangan terdeteksi oleh sensor IR, robot akan merespons dengan berhenti, berbelok, atau mundur.

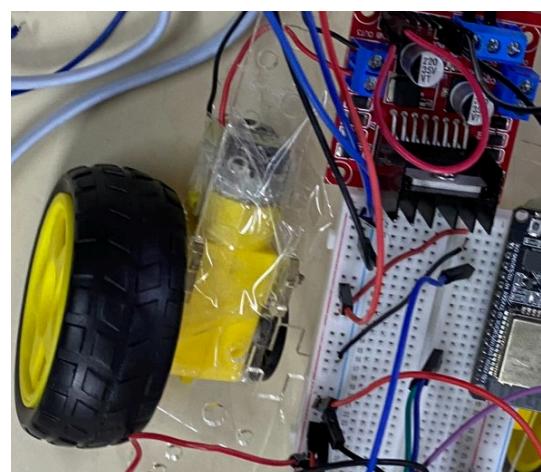
Proyek ini menggunakan FreeRTOS untuk menjalankan beberapa task secara paralel, seperti: moveMotorTask untuk mengontrol motor, sendDHTDataTask untuk mengirimkan data suhu dan kelembaban ke Blynk, remoteControlTask untuk membaca perintah dari aplikasi, readSensorTask untuk membaca data dari sensor ultrasonik dan IR, serta automaticControlTask untuk menjalankan logika pergerakan otomatis. Komunikasi antar task dilakukan melalui queue, sedangkan pemrosesan logika kontrol menggunakan variabel global. Penggunaan pinned task memungkinkan pembagian tugas ke core prosesor ESP32 yang berbeda, sehingga meningkatkan performa dan efisiensi pemrosesan. Secara keseluruhan, kode ini mencerminkan implementasi sistem IoT terintegrasi dengan kontrol robotik berbasis sensor untuk aplikasi di area yang sulit dijangkau.



*Figure 3. Flowchart*

## 2.3 HARDWARE AND SOFTWARE INTEGRATION

Ada beberapa fitur yang disediakan dalam proyek kami, fitur - fitur ini merupakan hasil integrasi antara hardware dan juga software. Berikut adalah deskripsi dari beberapa komponen dan bagaimana mereka saling berinteraksi.



```

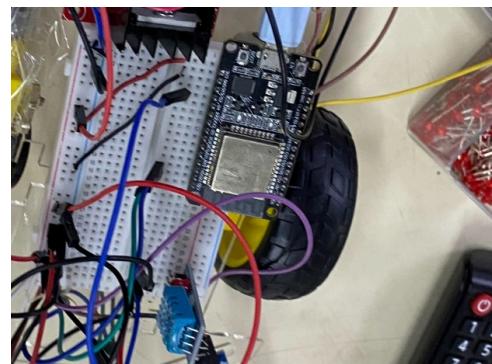
void moveMotorTask(void *pvParameters)
{
    while (1)

```

```
{  
    // saat mendapat queue, motor akan bergerak sesuai dengan  
queue yang diterima  
    // xQueueReceive(xQueue, &remoteControl, portMAX_DELAY);  
    // xQueueReceive(xQueue, &manualControl, portMAX_DELAY);  
    Serial.println("Remote Control: " + String(remoteControl));  
  
    if (remoteControl == 1)  
    {  
        // Maju  
        digitalWrite(MOTOR1_PIN1, HIGH);  
        digitalWrite(MOTOR1_PIN2, LOW);  
        digitalWrite(MOTOR2_PIN1, HIGH);  
        digitalWrite(MOTOR2_PIN2, LOW);  
    }  
    else if (remoteControl == 2)  
    {  
        // Mundur  
        digitalWrite(MOTOR1_PIN1, LOW);  
        digitalWrite(MOTOR1_PIN2, HIGH);  
        digitalWrite(MOTOR2_PIN1, LOW);  
        digitalWrite(MOTOR2_PIN2, HIGH);  
    }  
    else if (remoteControl == 3)  
    {  
        // Kanan  
        digitalWrite(MOTOR1_PIN1, HIGH);  
        digitalWrite(MOTOR1_PIN2, LOW);  
        digitalWrite(MOTOR2_PIN1, LOW);  
        digitalWrite(MOTOR2_PIN2, HIGH);  
    }  
    else if (remoteControl == 4)  
    {  
        // Kiri  
        digitalWrite(MOTOR1_PIN1, LOW);  
        digitalWrite(MOTOR1_PIN2, HIGH);  
        digitalWrite(MOTOR2_PIN1, HIGH);  
        digitalWrite(MOTOR2_PIN2, LOW);  
    }  
    else if (remoteControl == 5)  
    {  
        // Berhenti  
        digitalWrite(MOTOR1_PIN1, LOW);  
        digitalWrite(MOTOR1_PIN2, LOW);  
        digitalWrite(MOTOR2_PIN1, LOW);  
        digitalWrite(MOTOR2_PIN2, LOW);  
    }  
  
    // delay untuk menghindari motor bergerak terlalu cepat  
vTaskDelay(200 / portTICK_PERIOD_MS);  
}
```

```
}
```

Kode ini bertugas mengontrol pergerakan motor robot berdasarkan nilai kontrol (variabel remoteControl) yang diterima dari queue atau dari logika kontrol otomatis/manual. Task ini menentukan apakah robot akan maju, mundur, berbelok ke kiri atau kanan, atau berhenti dengan mengatur pin motor secara tepat. Selain itu, terdapat jeda (vTaskDelay) selama 200 ms untuk mencegah pergerakan yang terlalu cepat.

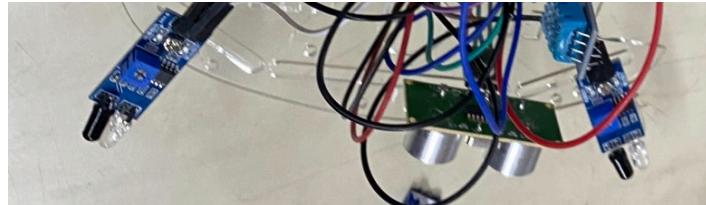


```
void sendDHTDataTask(void *pvParameters)
{
    while (1)
    {
        float temperature = dht.readTemperature();
        float humidity = dht.readHumidity();

        // Cek pembacaan
        if (isnan(temperature) || isnan(humidity))
        {
            Serial.println("Failed to read from DHT sensor!");
        }
        else
        {
            Serial.println("Temperature: " + String(temperature) + " C");
            Serial.println("Humidity: " + String(humidity) + " %");
            // Kirim data DHT ke Blynk
            Blynk.virtualWrite(V1, temperature);
            Blynk.virtualWrite(V2, humidity);
        }

        // delay 1 detik
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

Kode ini membaca data suhu dan kelembaban dari sensor DHT11. Data tersebut dikirim ke server Blynk melalui virtual pin sehingga dapat dimonitor oleh pengguna pada aplikasi Blynk. Jika pembacaan sensor gagal, task ini akan mencetak pesan error ke serial monitor. Task ini diatur untuk dijalankan setiap 1 detik (1000 ms) untuk memastikan pengambilan data sensor secara berkala.



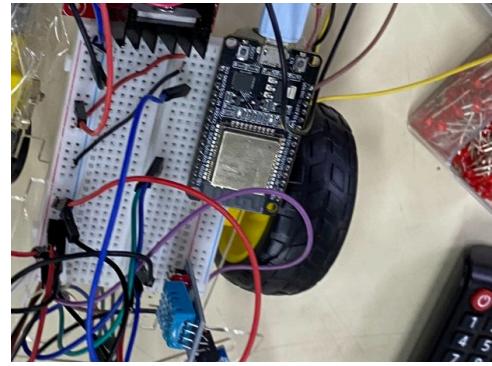
```
void readSensorTask(void *pvParameters)
{
    while (1)
    {
        // Serial.println("Reading sensor...");
        // Serial.println("Distance: " + String(distance) + " cm");
        // Serial.println("IR Sensor 1: " + String(irSensor1));
        // Serial.println("IR Sensor 2: " + String(irSensor2));

        // Baca sensor ultrasonik
        distance = ultrasonic.read(); // baca jarak dalam cm

        // Baca sensor IR
        irSensor1 = digitalRead(IR_SENSOR1);
        irSensor2 = digitalRead(IR_SENSOR2);

        // delay 100 ms
        vTaskDelay(200 / portTICK_PERIOD_MS);
    }
}
```

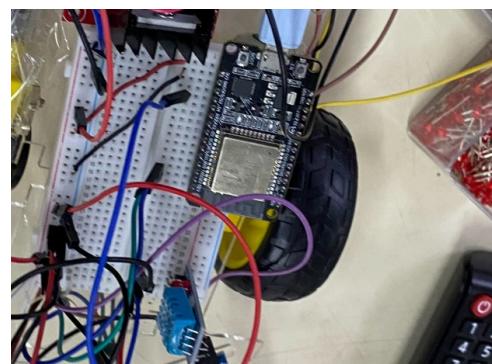
Kode ini membaca data dari sensor ultrasonik (jarak dalam cm) dan dua sensor IR (untuk deteksi rintangan). Data sensor ini disimpan dalam variabel global (distance, irSensor1, dan irSensor2) untuk digunakan oleh task lain, terutama automaticControlTask. Dengan jeda 200 ms, task ini memastikan pembacaan sensor dilakukan secara periodik tanpa menghabiskan terlalu banyak siklus prosesor.



```
void remoteControlTask(void *pvParameters)
{
    while (1)
    {
        // kalau manual control aktif, meneruskan nilai remote
control ke moveMotorTask
        if (manualControl == 1)
        {
            // Maju
            if (wirelessControl == 1)
            {
                remoteControl = 1;
            }
            // Mundur
            else if (wirelessControl == 2)
            {
                remoteControl = 2;
            }
            // Kanan
            else if (wirelessControl == 3)
            {
                remoteControl = 3;
            }
            // Kiri
            else if (wirelessControl == 4)
            {
                remoteControl = 4;
            }
            // Berhenti
            else if (wirelessControl == 5)
            {
                remoteControl = 5;
            }
        }

        // delay 10 ms
        vTaskDelay(100 / portTICK_PERIOD_MS);
    }
}
```

Kode ini menangani kontrol manual melalui aplikasi Blynk. Ketika mode manual aktif (manualControl == 1), task ini membaca nilai kontrol arah (wirelessControl) dari aplikasi dan meneruskannya ke variabel global remoteControl. Variabel ini kemudian digunakan oleh moveMotorTask untuk menggerakkan robot sesuai perintah pengguna. Task ini memiliki jeda delay sebesar 100 ms untuk mengurangi beban prosesor.



```
void automaticControlTask(void *pvParameters)
{
    while (1)
    {
        // kalau manual control tidak aktif, maka robot akan bergerak
        otomatis
        if (manualControl == 0)
        {
            // Jika sensor ultrasonik mendeteksi objek di depan
            if (distance < 10)
            {
                // Berhenti
                remoteControl = 5;
            }
            else
            {
                // Jika kedua sensor IR mendeteksi objek (IR Kiri dan
                Kanan)
                if (irSensor1 == 0 && irSensor2 == 0)
                {
                    // Mundur
                    remoteControl = 2;
                }
                // Jika sensor IR1 mendeteksi objek (IR Kiri)
                else if (irSensor1 == 0)
                {
                    // Kanan
                    remoteControl = 3;
                }
            }
        }
    }
}
```

```
// Jika sensor IR2 mendeteksi objek (IR Kanan)
else if (irSensor2 == 0)
{
    // Kiri
    remoteControl = 4;
}
else
{
    // Maju
    remoteControl = 1;
}
}

// delay 10 ms
vTaskDelay(10 / portTICK_PERIOD_MS);
}
}
```

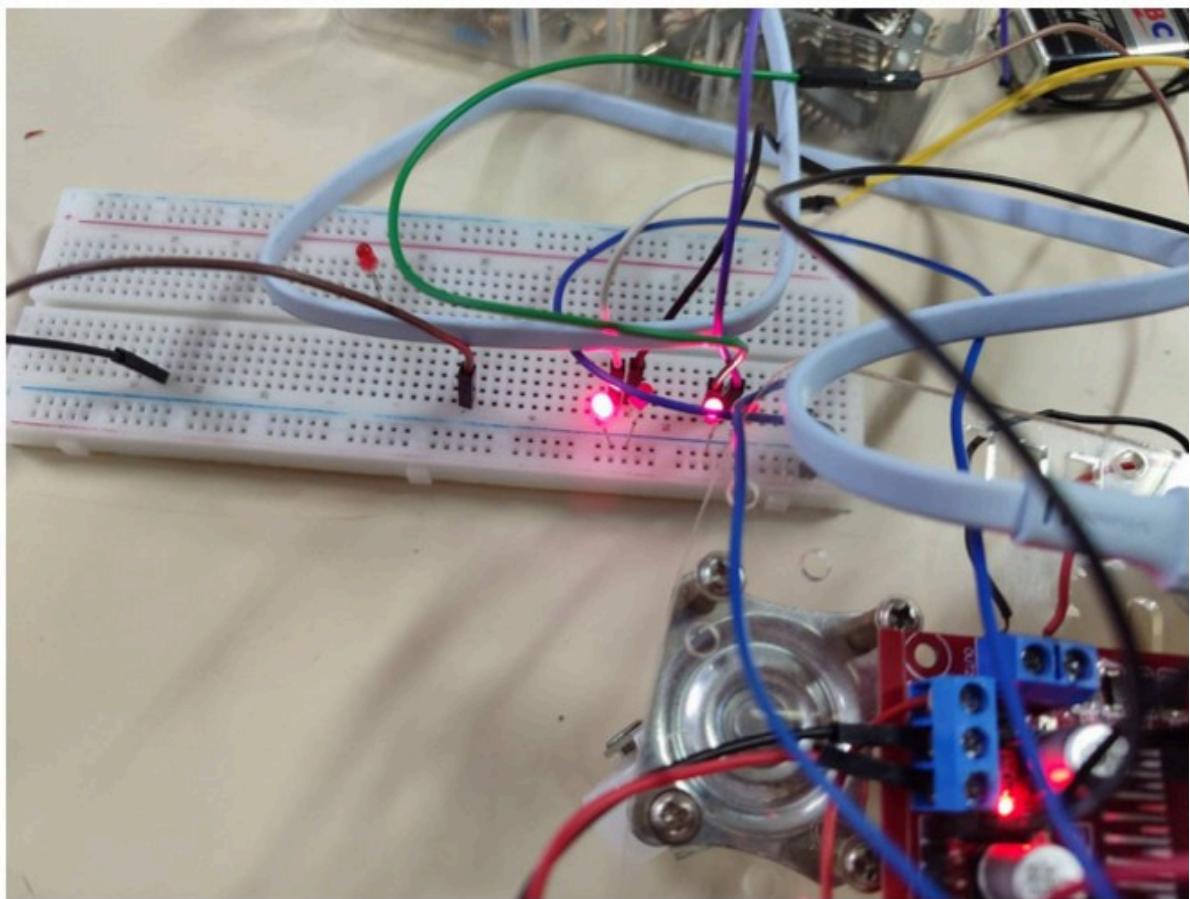
Kode ini bertanggung jawab atas logika pergerakan otomatis robot ketika mode manual tidak aktif (manualControl == 0). Berdasarkan data dari sensor ultrasonik dan IR, task ini menentukan pergerakan robot. Misalnya, jika ada rintangan di depan (jarak sangat dekat), robot akan berhenti; jika hanya satu sisi terdeteksi oleh sensor IR, robot akan berbelok; jika tidak ada rintangan, robot akan maju. Task ini berjalan dengan jeda sangat kecil (10 ms) untuk memastikan respons cepat terhadap perubahan kondisi lingkungan.

## CHAPTER 3

### TESTING AND EVALUATION

#### 3.1 TESTING

Setelah menyelesaikan pembuatan kode dan juga menyelesaikan rangkaian, maka dilakukanlah testing untuk menguji apakah program dapat berjalan dengan sesuai. Testing dilakukan dengan menjalankan program lalu menggunakan Blynk sebagai alat manual control dari robot mobil tersebut. Jika berhasil maka output akan menampilkan nilai dari arah yang dijalankan saat itu, semua setup berhasil dilakukan, kondisi mobil robot, dan hasil deteksi sensor DHT11.



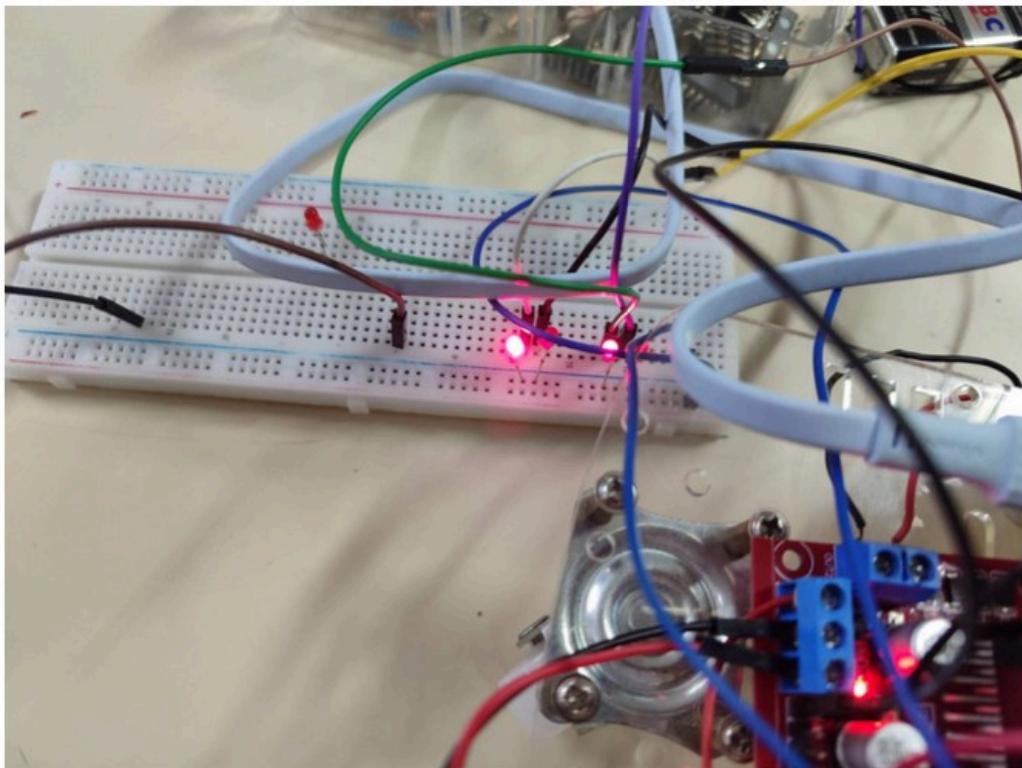
```
// Maju
Serial.println("\nMaju\n");
remoteControl = 1;
vTaskDelay(2000 / portTICK_PERIOD_MS);

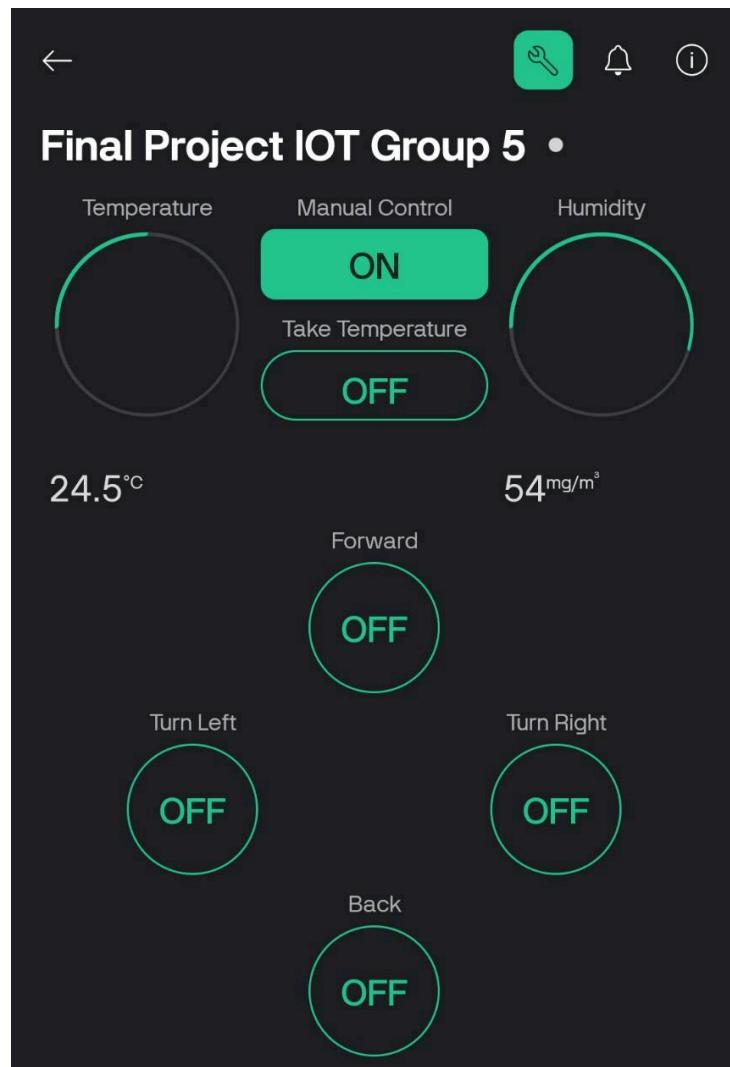
// Berhenti
```

```
Motor setup
PWM setup
Blynk started
Task created
Task created DHT
Task created Sensor
```

Testing dilakukan dengan menjalankan program di VS code menggunakan platform.io. Kita juga harus menjalankan Blynk untuk mencoba mode manual control.

### 3.2 RESULT





Dari hasil testing yang dilakukan terdapat output yang menunjukkan bahwa mobil sedang berjalan maju (Remote Control:1), ini menandakan tidak ada rintangan atau halangan yang mengganggu jalan mobil tersebut. Dan seharusnya terdapat output yang menampilkan suhu namun tidak ditampilkan karena dari Blynk, user tidak menyalakan “Take Temperature”.

### 3.3 EVALUATION

Dari hasil testing yang dilakukan, terbukti bahwa proyek ini telah berhasil bekerja sesuai dengan kriteria yang ditentukan. Sensor yang digunakan dapat bekerja dengan seharusnya, yaitu sensor IR, Ultrasonic, dan DHT11. Tetapi ada beberapa kendala yang ditemukan saat mencoba melakukan testing pada proyek ini. Daya yang digunakan yaitu baterai 9 V, tidak mencukupi untuk menjalankan motor dan roda untuk robot bisa berjalan. Daya yang digunakan haruslah 12 V agar mobil dapat berjalan dengan lancar. Untuk kedepannya seharusnya kita juga mempertimbangkan hal tersebut.

## **CHAPTER 4**

### **CONCLUSION**

Proyek KOSEKI (Kontrol/Otomatis Suhu/Kelembaban Kar Intelligent) yang telah kami buat dirancang untuk mendukung akses ke area sulit dijangkau manusia, seperti medan bencana atau dalam gua. Robot ini dilengkapi dengan berbagai fitur, termasuk mode kontrol otomatis dan manual yang dapat diakses melalui aplikasi Blynk, serta sensor DHT11 untuk pengukuran suhu dan kelembaban, sensor ultrasonik untuk pengukuran jarak, dan sensor inframerah untuk deteksi rintangan. Dalam mode otomatis, robot memanfaatkan data dari sensor-sensor ini untuk menghindari rintangan dan menentukan arah gerakannya. Sementara dalam mode manual, pengguna dapat mengontrol robot secara langsung menggunakan aplikasi Blynk dengan mengirimkan perintah ke sistem melalui WiFi.

Proyek ini mengimplementasikan algoritma kontrol berbasis multitasking dengan FreeRTOS, yang memungkinkan robot untuk membaca sensor, mengontrol motor, dan berkomunikasi dengan server Blynk secara efisien. Data suhu dan kelembaban dikirim ke aplikasi Blynk untuk dipantau pengguna, sementara kontrol pergerakan robot dilakukan secara responsif baik dalam mode manual maupun otomatis.

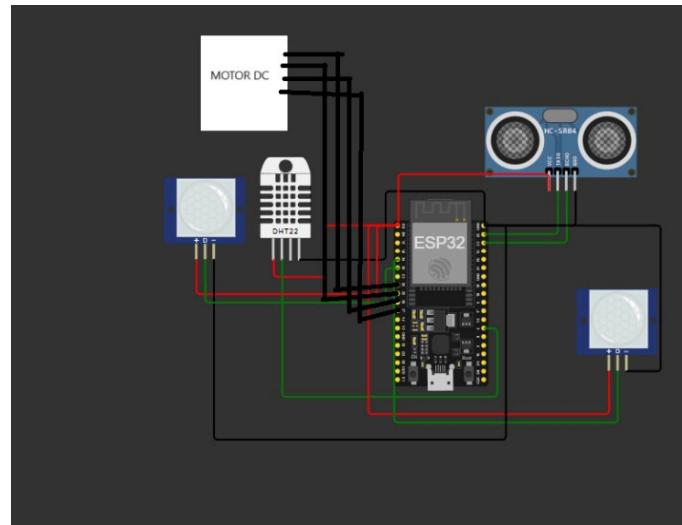
Dari pengujian yang telah dilakukan, dapat disimpulkan bahwa proyek “Robot Cerdas untuk Eksplorasi Otomatis dan Manual” telah berfungsi sesuai target dan parameter yang ditentukan. Robot berhasil membaca data sensor dengan akurat, bergerak secara otomatis untuk menghindari rintangan, dan merespons perintah manual melalui aplikasi Blynk. Harapannya, proyek ini dapat menjadi prototipe yang berguna untuk berbagai aplikasi, seperti penanganan bencana atau eksplorasi lingkungan berisiko, serta memberikan kontribusi dalam pengembangan teknologi IoT dan robotika.

## REFERENCES

- [1] Blynk IoT Platform, "Blynk - IoT Platform," [Online]. Available: <https://blynk.io/>. [Accessed: Dec. 11, 2024].
- [2] FreeRTOS, "FreeRTOS Official Website," [Online]. Available: <https://www.freertos.org/>. [Accessed: Dec. 11, 2024].
- [3] FreeRTOS, "Queues, Mutexes, and Semaphores," [Online]. Available: <https://www.freertos.org/Documentation/02-Kernel/02-Kernel-features/02-Queues-mutexes-and-semaphores/01-Queues>. [Accessed: Dec. 11, 2024].
- [4] FreeRTOS, "Queue Management API References," [Online]. Available: <https://freertos.org/Documentation/02-Kernel/04-API-references/06-Queues/00-QueueManagement>. [Accessed: Dec. 11, 2024].
- [5] Arduino, "Pulse Width Modulation (PWM) Tutorial," [Online]. Available: <https://www.arduino.cc/en/Tutorial/PWM>. [Accessed: Dec. 11, 2024].
- [6] Espressif Systems, "FreeRTOS API Reference - ESP-IDF v4.3," [Online]. Available: <https://docs.espressif.com/projects/esp-idf/en/v4.3/esp32/api-reference/system/freertos.html>. [Accessed: Dec. 11, 2024].
- [7] FreeRTOS, "xTaskCreate API Reference," [Online]. Available: <https://www.freertos.org/Documentation/02-Kernel/04-API-references/01-Task-creation/01-xTaskCreate>. [Accessed: Dec. 11, 2024].

## APPENDICES

### Appendix A: Project Schematic



### Appendix B: Documentation



### Appendix C: Related Link

- Finpro IoT
- <https://github.com/EdgrantHS/Finpro-IOT>