LAMPIRAN

```
#include <SoftwareSerial.h>
SoftwareSerial BT_Serial(2, 3); // RX, TX

#define enA 10 // Pin L298 untuk mengatur kecepatan motor 1
#define in1 9  // Pin L298 untuk menggerakkan motor 1 maju
#define in2 8  // Pin L298 untuk menggerakkan motor 1 mundur
#define in3 7  // Pin L298 untuk menggerakkan motor 2 maju
#define in4 6  // Pin L298 untuk menggerakkan motor 2 mundur
#define enB 5  // Pin L298 untuk mengatur kecepatan motor 2

// Pin untuk sensor ultrasonik
#define triggerPin1 11
#define echoPin1 12
#define triggerPin2 13
#define echoPin2 A0
#define triggerPin3 A1
#define echoPin3 A2

long duration1;
long duration2;
long duration3;
float distanceleft;
float distancefront;
float distanceright;
int set = 20 ; // Jarak ambang batas untuk penghindaran hambatan

int bt_data; // Variabel untuk menerima data dari serial port (Aplikasi Android)
int Speed = 180;
int Speed2 = 200;
int Speed3 = 200;
int mode = 0;
```

```
void setup()
{
  pinMode(triggerPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(triggerPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  pinMode(triggerPin3, OUTPUT);
  pinMode(echoPin3, INPUT);

  pinMode(enA, OUTPUT); // Deklarasikan pin enA sebagai output untuk
mengatur kecepatan motor 1
  pinMode(in1, OUTPUT); // Deklarasikan pin in1 sebagai output untuk
menggerakkan motor 1 maju
  pinMode(in2, OUTPUT); // Deklarasikan pin in2 sebagai output untuk
menggerakkan motor 1 mundur
  pinMode(in3, OUTPUT); // Deklarasikan pin in3 sebagai output untuk
menggerakkan motor 2 maju
  pinMode(in4, OUTPUT); // Deklarasikan pin in4 sebagai output untuk
menggerakkan motor 2 mundur
  pinMode(enB, OUTPUT); // Deklarasikan pin enB sebagai output untuk
mengatur kecepatan motor 2

  Serial.begin(9600);   // Memulai komunikasi serial dengan kecepatan 9600bps
  BT_Serial.begin(9600); // Memulai komunikasi serial Bluetooth dengan
kecepatan 9600bps
}

void loop()
{
  // Periksa apakah ada data yang diterima dari aplikasi Android
  if (BT_Serial.available() > 0)
  {
```

```
    bt_data = BT_Serial.read();
    Serial.println(bt_data);

    // Periksa data yang diterima untuk memilih mode
    if (bt_data == 1)
    {
      mode = 0; // Mode manual
    }
    else if (bt_data == 2)
    {
      mode = 1; // Mode Halang rintang
    }
  }

  // Pemilihan mode
  if (mode == 0)
  {
    // Mode Manual: Kendalikan robot menggunakan perintah dari aplikasi
Android
    manualControl();
  }
  else if (mode == 1)
  {
    // Mode Penghindaran Hambatan: Gunakan sensor ultrasonik untuk
menghindari hambatan
    obstacleAvoidance();
  }
}

// Fungsi untuk mengendalikan robot secara manual berdasarkan perintah dari
aplikasi Android
void manualControl()
{
```

```
  if (bt_data == 3)
  {
    analogWrite(enA, Speed);
    analogWrite(enB, Speed);
    forward();
  }
  else if (bt_data == 4)
  {
    analogWrite(enA, Speed);
    analogWrite(enB, Speed);
    backward();
  }
  else if (bt_data == 5)
  {
    analogWrite(enA, Speed2);
    analogWrite(enB, Speed2);
       turnLeft();
  }
  else if (bt_data == 6)
  {
    analogWrite(enA, Speed2);
    analogWrite(enB, Speed2);
    turnRight();
  }
  else if (bt_data == 7)
  {
    stopRobot();
  }
}

void obstacleAvoidance()
{
  // Mengukur jarak ke hambatan menggunakan sensor ultrasonik
```

```
  distancefront = calculateDistance(triggerPin1, echoPin1);
  distanceleft = calculateDistance(triggerPin2, echoPin2);
  distanceright = calculateDistance(triggerPin3, echoPin3);

  // Jika ada hambatan di depan dan jarak kurang dari ambang batas set
  if (distancefront >= set)
     {analogWrite(enA, Speed);
     analogWrite(enB, Speed);
     forward();
    }
    // Menghindari hambatan dengan belok ke kanan atau kiri (Pilih sesuai
keadaan)
    if (distancefront <= set && distanceright >= set && distanceleft <= set)
    {
     // Hindari hambatan dengan berbelok ke kanan
     analogWrite(enA, Speed3);
     analogWrite(enB, Speed3);
     turnRight();
    }
    if (distancefront >= set && distanceright <= set && distanceleft >= set)
    {
     // Hindari hambatan dengan berbelok ke kanan
     analogWrite(enA, Speed3);
     analogWrite(enB, Speed3);
     turnLeft();
    }
    if (distancefront <= set && distanceright < distanceleft )
    {
     // Hindari hambatan dengan berbelok ke kanan
     analogWrite(enA, Speed3);
     analogWrite(enB, Speed3);
     turnLeft();
    }
```

```
  if (distancefront >= set && distanceright > distanceleft )
    {
      // Hindari hambatan dengan berbelok ke kanan
      analogWrite(enA, Speed3);
      analogWrite(enB, Speed3);
      turnRight();
    }
}
float calculateDistance(int triggerPin, int echoPin)
{
  // Mengirimkan pulsa ultrasonik untuk mengukur jarak hambatan
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);

  // Menerima waktu pantulan kembali (echo) dan menghitung jarak
  long duration = pulseIn(echoPin, HIGH);
  float distance = duration * 0.034 / 2; // Mengubah waktu menjadi jarak dalam cm

  return distance;
}

// Functions to control the robot's movements
void forward()
{
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}
```

```
void backward()
{
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}

void turnLeft()
{
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}

void turnRight()
{
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
}

void stopRobot()
{
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}
```