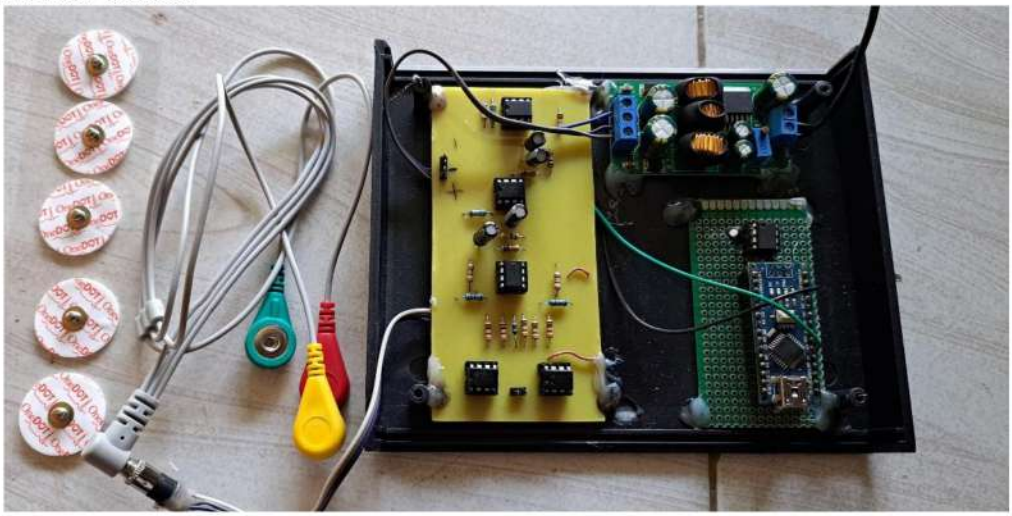
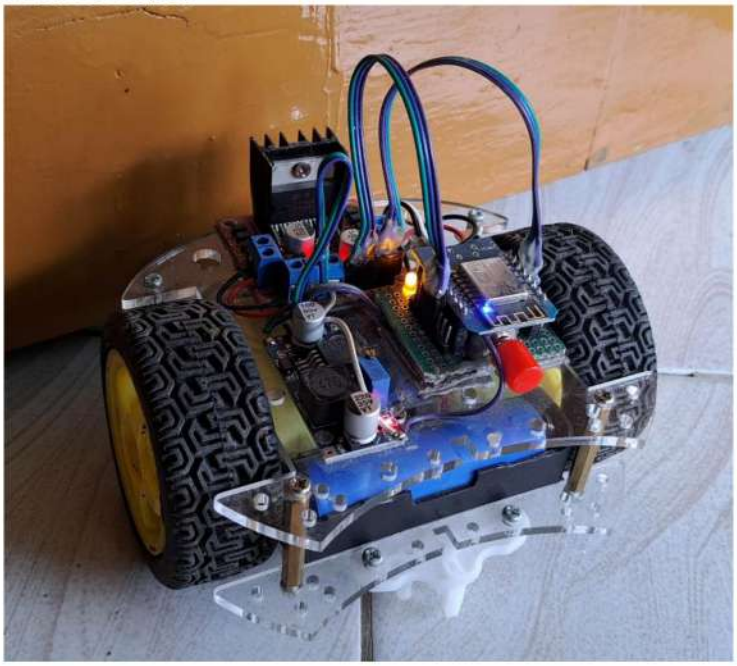
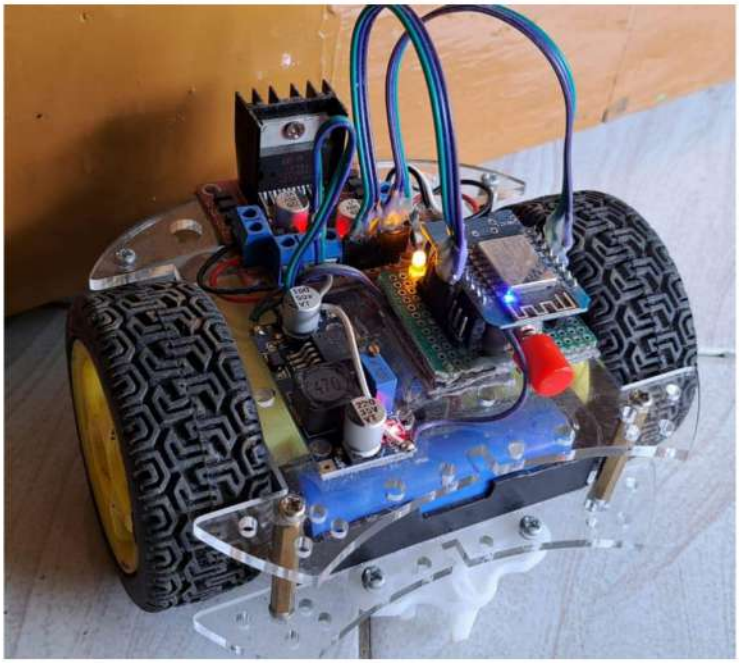
LAMPIRAN

A. Modul Sensor



B. Mobile Robot





C. Code Program Mobile Robot

```
/******************** WiFi Robot Remote Control Mode ********************/
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ArduinoOTA.h>

// connections for drive Motors
int PWM_A = D1;
int PWM_B = D2;
int DIR_A = D3;
int DIR_B = D4;

int PWM_R = D4;
int DIR1_R = D3;
int DIR2_R = D2;

int PWM_L = D8;
int DIR1_L = D7;
int DIR2_L = D6;

const int buzPin = D5;      // set digital pin D5 as buzzer pin (use active buzzer)
const int ledPin = D8;      // set digital pin D8 as LED pin (use super bright LED)
```

```
const int wifiLedPin = D0;  // set digital pin D0 as indication, the LED turn on if NodeMCU connected to WiFi as STA mode

String command;          // String to store app command state.
int SPEED = 150; //1023;        // 330 - 1023.
int speed_Coeff = 3;

ESP8266WebServer server(80);     // Create a webserver object that listens for HTTP request on port 80

unsigned long previousMillis = 0;

//String sta_ssid = "$your_ssid_maximum_32_characters";       // set Wifi networks you want to connect to
//String sta_password = "$your_pswd_maximum_32_characters";   // set password for Wifi networks

String sta_ssid = "SLIMS";      // set Wifi networks you want to connect to
String sta_password = "bitarumi";  // set password for Wifi networks


void setup(){
  Serial.begin(115200);    // set up Serial library at 115200 bps
  Serial.println();
  Serial.println("*WiFi Robot Remote Control Mode*");
  Serial.println("-------------------------------------");

  pinMode(buzPin, OUTPUT);      // sets the buzzer pin as an Output
  pinMode(ledPin, OUTPUT);      // sets the LED pin as an Output
  pinMode(wifiLedPin, OUTPUT);  // sets the Wifi LED pin as an Output
  digitalWrite(buzPin, LOW);
  digitalWrite(ledPin, LOW);
  digitalWrite(wifiLedPin, HIGH);

  // Set all the motor control pins to outputs
  pinMode(PWM_A, OUTPUT);
  pinMode(PWM_B, OUTPUT);
  pinMode(DIR_A, OUTPUT);
  pinMode(DIR_B, OUTPUT);

  pinMode(PWM_R, OUTPUT);
  pinMode(PWM_L, OUTPUT);
  pinMode(DIR1_R, OUTPUT);
  pinMode(DIR2_R, OUTPUT);
```

```
  pinMode(DIR1_L, OUTPUT);
  pinMode(DIR2_L, OUTPUT);

  // Turn off motors - Initial state
  digitalWrite(DIR1_R, LOW);
  digitalWrite(DIR2_R, LOW);
  digitalWrite(DIR1_L, LOW);
  digitalWrite(DIR2_L, LOW);
  analogWrite(PWM_R, 0);
  analogWrite(PWM_L, 0);

  // set NodeMCU Wifi hostname based on chip mac address
  String chip_id = String(ESP.getChipId(), HEX);
  int i = chip_id.length()-4;
  chip_id = chip_id.substring(i);
  chip_id = "robocar-" + chip_id;
  String hostname(chip_id);

  Serial.println();
  Serial.println("Hostname: "+hostname);

  // first, set NodeMCU as STA mode to connect with a Wifi network
  WiFi.mode(WIFI_STA);
  WiFi.begin(sta_ssid.c_str(), sta_password.c_str());
  Serial.println("");
  Serial.print("Connecting to: ");
  Serial.println(sta_ssid);
  Serial.print("Password: ");
  Serial.println(sta_password);

  // try to connect with Wifi network about 10 seconds
  unsigned long currentMillis = millis();
  previousMillis = currentMillis;
  while (WiFi.status() != WL_CONNECTED && currentMillis -
previousMillis <= 10000) {
    delay(500);
    Serial.print(".");
    currentMillis = millis();
  }

  // if failed to connect with Wifi network set NodeMCU as AP mode
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("");
    Serial.println("*WiFi-STA-Mode*");
```

```
    Serial.print("IP: ");
    Serial.println(WiFi.localIP());
    digitalWrite(wifiLedPin, LOW);     // Wifi LED on when connected to
Wifi as STA mode
    delay(3000);
  } else {
    WiFi.mode(WIFI_AP);
    WiFi.softAP(hostname.c_str());
    IPAddress myIP = WiFi.softAPIP();
    Serial.println("");
    Serial.println("WiFi failed connected to " + sta_ssid);
    Serial.println("");
    Serial.println("*WiFi-AP-Mode*");
    Serial.print("AP IP address: ");
    Serial.println(myIP);
    digitalWrite(wifiLedPin, HIGH);    // Wifi LED off when status as AP
mode
    delay(3000);
  }


  server.on ( "/", HTTP_handleRoot );       // call the 'handleRoot' function
when a client requests URI "/"
  server.onNotFound ( HTTP_handleRoot );    // when a client requests an
unknown URI (i.e. something other than "/"), call function
"handleNotFound"
  server.begin();                           // actually start the server

  ArduinoOTA.begin();                       // enable to receive update/uploade
firmware via Wifi OTA
}


void loop() {
//  Forward();
    ArduinoOTA.handle();        // listen for update OTA request from clients
    server.handleClient();      // listen for HTTP requests from clients

      command = server.arg("State");          // check HTPP request, if has
arguments "State" then saved the value
      if (command == "F") Forward();       // check string then call a function
or set a value
      else if (command == "B") Backward();
      else if (command == "R") TurnRight();
```

```
      else if (command == "L") TurnLeft();
      else if (command == "g") ForwardLeft();
      else if (command == "G") ForwardLeft1();
      else if (command == "H") BackwardLeft();
      else if (command == "i") ForwardRight();
      else if (command == "I") ForwardRight1();
      else if (command == "J") BackwardRight();
      else if (command == "S") Stop();
      else if (command == "V") BeepHorn();
      else if (command == "W") TurnLightOn();
      else if (command == "w") TurnLightOff();
//      else if (command == "0") SPEED = 330;
//      else if (command == "1") SPEED = 400;
//      else if (command == "2") SPEED = 470;
//      else if (command == "3") SPEED = 540;
//      else if (command == "4") SPEED = 610;
//      else if (command == "5") SPEED = 680;
//      else if (command == "6") SPEED = 750;
//      else if (command == "7") SPEED = 820;
//      else if (command == "8") SPEED = 890;
//      else if (command == "9") SPEED = 960;
//      else if (command == "q") SPEED = 1023;
}

// function prototypes for HTTP handlers
void HTTP_handleRoot(void){
  server.send ( 200, "text/html", "" );      // Send HTTP status 200 (Ok) and
send some text to the browser/client

  if( server.hasArg("State") ){
    Serial.println(server.arg("State"));
  }
}

void handleNotFound(){
  server.send(404, "text/plain", "404: Not found"); // Send HTTP status 404
(Not Found) when there's no handler for the URI in the request
}

// function to move forward
void Forward(){
  digitalWrite(DIR1_L, HIGH);
  digitalWrite(DIR2_L, LOW);
  digitalWrite(DIR1_R, HIGH);
```

```
  digitalWrite(DIR2_R, LOW);
  analogWrite(PWM_L, SPEED);
  analogWrite(PWM_R, SPEED);
}

// function to move backward
void Backward(){
  digitalWrite(DIR1_L, LOW);
  digitalWrite(DIR2_L, HIGH);
  digitalWrite(DIR1_R, LOW);
  digitalWrite(DIR2_R, HIGH);
  analogWrite(PWM_L, SPEED);
  analogWrite(PWM_R, SPEED);
}

// function to turn right
void TurnRight(){
  digitalWrite(DIR1_L, HIGH);
  digitalWrite(DIR2_L, LOW);
  digitalWrite(DIR1_R, LOW);
  digitalWrite(DIR2_R, HIGH);
  analogWrite(PWM_L, SPEED);
  analogWrite(PWM_R, SPEED);
}

// function to turn left
void TurnLeft(){
  digitalWrite(DIR1_L, LOW);
  digitalWrite(DIR2_L, HIGH);
  digitalWrite(DIR1_R, HIGH);
  digitalWrite(DIR2_R, LOW);
  analogWrite(PWM_L, SPEED);
  analogWrite(PWM_R, SPEED);
}

// function to move forward left
void ForwardLeft(){
  digitalWrite(DIR1_L, HIGH);
  digitalWrite(DIR2_L, LOW);
  digitalWrite(DIR1_R, HIGH);
  digitalWrite(DIR2_R, LOW);
  analogWrite(PWM_L, SPEED/speed_Coeff);
  analogWrite(PWM_R, SPEED);
}
```

```
void ForwardLeft1(){
  digitalWrite(DIR1_L, HIGH);
  digitalWrite(DIR2_L, LOW);
  digitalWrite(DIR1_R, HIGH);
  digitalWrite(DIR2_R, LOW);
  analogWrite(PWM_L, SPEED/2);
  analogWrite(PWM_R, SPEED);
}

// function to move backward left
void BackwardLeft(){
  digitalWrite(DIR1_L, LOW);
  digitalWrite(DIR2_L, HIGH);
  digitalWrite(DIR1_R, LOW);
  digitalWrite(DIR2_R, HIGH);
  analogWrite(PWM_L, SPEED);
  analogWrite(PWM_R, SPEED/speed_Coeff);
}

// function to move forward right
void ForwardRight(){
  digitalWrite(DIR1_L, HIGH);
  digitalWrite(DIR2_L, LOW);
  digitalWrite(DIR1_R, HIGH);
  digitalWrite(DIR2_R, LOW);
  analogWrite(PWM_L, SPEED);
  analogWrite(PWM_R, SPEED/speed_Coeff);
}
void ForwardRight1(){
  digitalWrite(DIR1_L, HIGH);
  digitalWrite(DIR2_L, LOW);
  digitalWrite(DIR1_R, HIGH);
  digitalWrite(DIR2_R, LOW);
  analogWrite(PWM_L, SPEED);
  analogWrite(PWM_R, SPEED/2);
}

// function to move backward left
void BackwardRight(){
  digitalWrite(DIR1_L, LOW);
  digitalWrite(DIR2_L, HIGH);
  digitalWrite(DIR1_R, LOW);
  digitalWrite(DIR2_R, HIGH);
  analogWrite(PWM_L, SPEED/speed_Coeff);
```

```
  analogWrite(PWM_R, SPEED);
}

// function to stop motors
void Stop(){
  digitalWrite(DIR1_L, LOW);
  digitalWrite(DIR2_L, LOW);
  digitalWrite(DIR1_R, LOW);
  digitalWrite(DIR2_R, LOW);
  analogWrite(PWM_L, 0);
  analogWrite(PWM_R, 0);
}

// function to beep a buzzer
void BeepHorn(){
  digitalWrite(buzPin, HIGH);
  delay(150);
  digitalWrite(buzPin, LOW);
  delay(80);
}

// function to turn on LED
void TurnLightOn(){
  digitalWrite(ledPin, HIGH);
}

// function to turn off LED
void TurnLightOff(){
  digitalWrite(ledPin, LOW);
}
```

D. Code Program Pengujian Sensor EOG

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Kelas Kalman Filter yang sudah ada
class SimpleKalmanFilter:
    def __init__(self, Q, R, P, x0):
        self.Q = Q  # Process noise covariance
        self.R = R  # Observation noise covariance
        self.P = P  # Estimate error covariance
        self.x = x0 # Initial state estimate
    def predict(self):
```

```
        # Predict the state/Prediksi keadaan
        self.P = self.P + self.Q
    def update(self, z):
        # Update the state
        K = self.P / (self.P + self.R)
        self.x = self.x + K * (z - self.x)
        self.P = (1 - K) * self.P
# Define the parameters of the Simple Kalman Filter
Q = 1     # Process noise covariance
R = 10    # Observation noise covariance
P = 1     # Estimate error covariance
x0 = 0    # Initial state estimate

kf = SimpleKalmanFilter(Q, R, P, x0)

# Baca file CSV
file_path = 'arduino_data.csv'
data = pd.read_csv(file_path)

# Mengambil 100 baris pertama dari kolom "Original Data"
original_data = data['Original Data'].iloc[:115]

# Terapkan Kalman Filter pada data
filtered_data = []
for z in original_data:
    kf.predict()
    kf.update(z)
    filtered_data.append(kf.x)

# Plot data original dan hasil Kalman Filter
plt.figure(figsize=(10, 6))
plt.plot(original_data.values, label='Original Data')
plt.plot(filtered_data, label='Kalman Filtered Data', linestyle='--')
plt.title('Original Data dan Kalman Filtered Data')
plt.xlabel('Index')
plt.ylabel('Data EOG')
plt.ylim(0, 300)  # Mengatur skala sumbu y dari 0 sampai 300
plt.legend()
plt.grid(True)
plt.gca().spines[['top', 'right']].set_visible(False)

# Tampilkan grafik
plt.show()
```