DAFTAR PUSTAKA

Craig, J. J., Prentice, P., & Hall, P. P. (2005). *Introduction to Robotics Mechanics and Control Third Edition*.

Dewi, N. (2019). *PROTOTYPE SMART HOME DENGAN MODUL NODEMCU ESP8266 BERBASIS INTERNET OF THINGS (IOT)*.

Jazar, R. N. (2010). Theory of applied robotics: Kinematics, dynamics, and control (2nd Edition). In *Theory of Applied Robotics: Kinematics, Dynamics, and Control (2nd Edition)*. Springer US. https://doi.org/10.1007/978-1-4419-1750-8

liu, haoyu. (2018). *Solar Tracker*. https://digitalcommons.cwu.edu/undergradproj

Mirza, Y., & Firdaus, A. (2016). *Light Dependent Resistant (LDR) Sebagai Pendeteksi Warna*.

Patel, S., Salazar, C., Patel, K. K., Patel, S. M., & Scholar, P. G. (2016). Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges. *International Journal of Engineering Science and Computing*. https://doi.org/10.4010/2016.1482

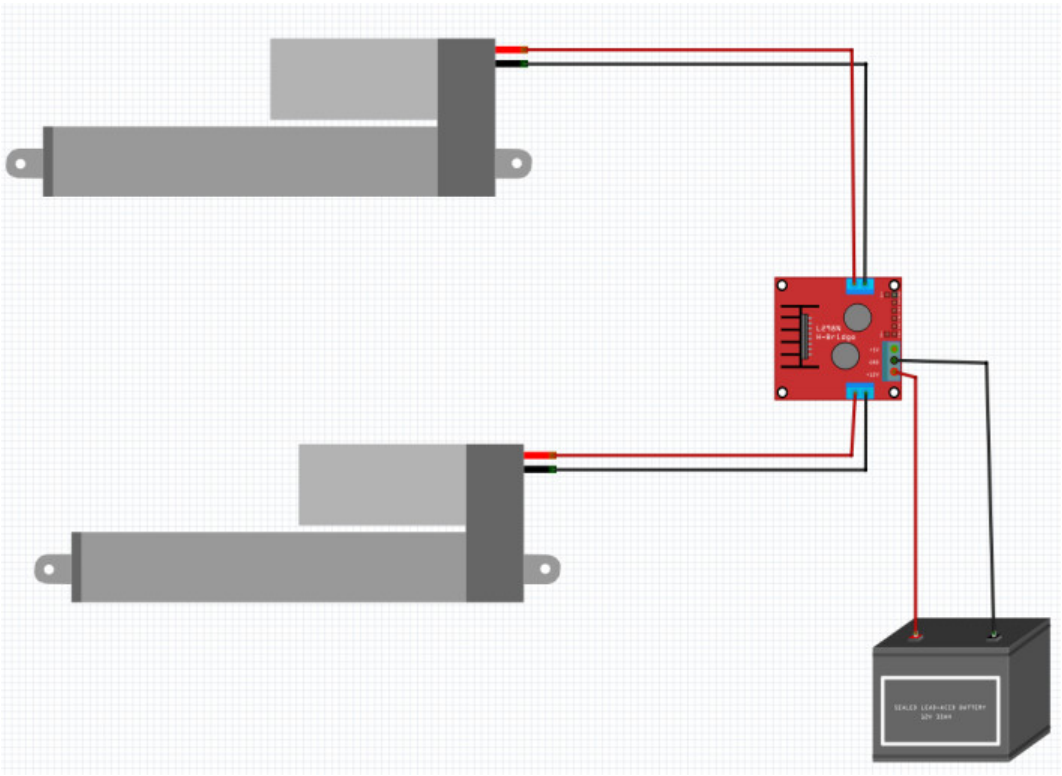
Sarkar, S., & Rahman, M. (2018). Power-Energy Optimization of Solar Photovoltaic Device Modeling. *Proceedings of International Conference on 2018 IEEE Electron Device Kolkata Conference, EDKCON 2018*, 541–546. https://doi.org/10.1109/EDKCON.2018.8770431

Sasmoko, D. (2021). *Arduino dan Sensor* (I. Dianta, Ed.; 1st ed., Vol. 1). Yayasan Prima Agus Teknik.

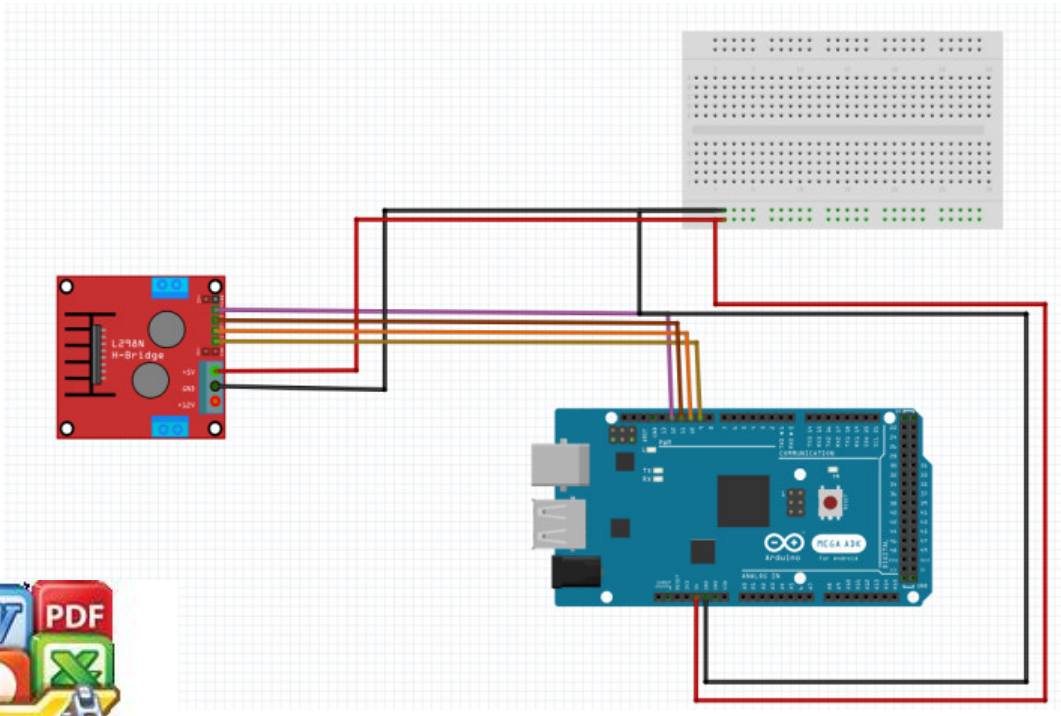
Selviyani, S. (2016). *RANCANG BANGUN SISTEM MONITORING ARUS DAN TEGANGAN DC BERBASIS MIKROKONTROLER ATMEGA32 PADA TURBIN ANGIN HORIZONTAL AXIS*.

LAMPIRAN RANGKAIAN SKEMATIK

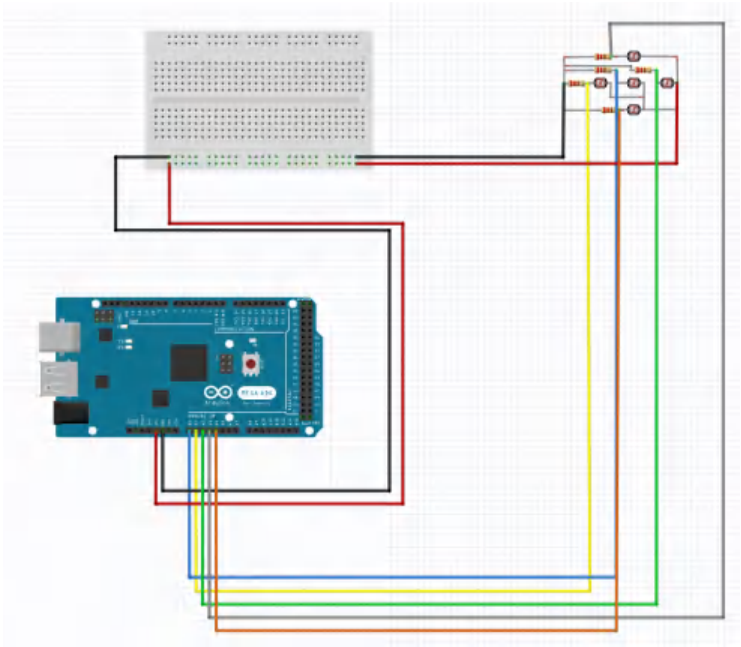
Linear actuator to motor driver L298N



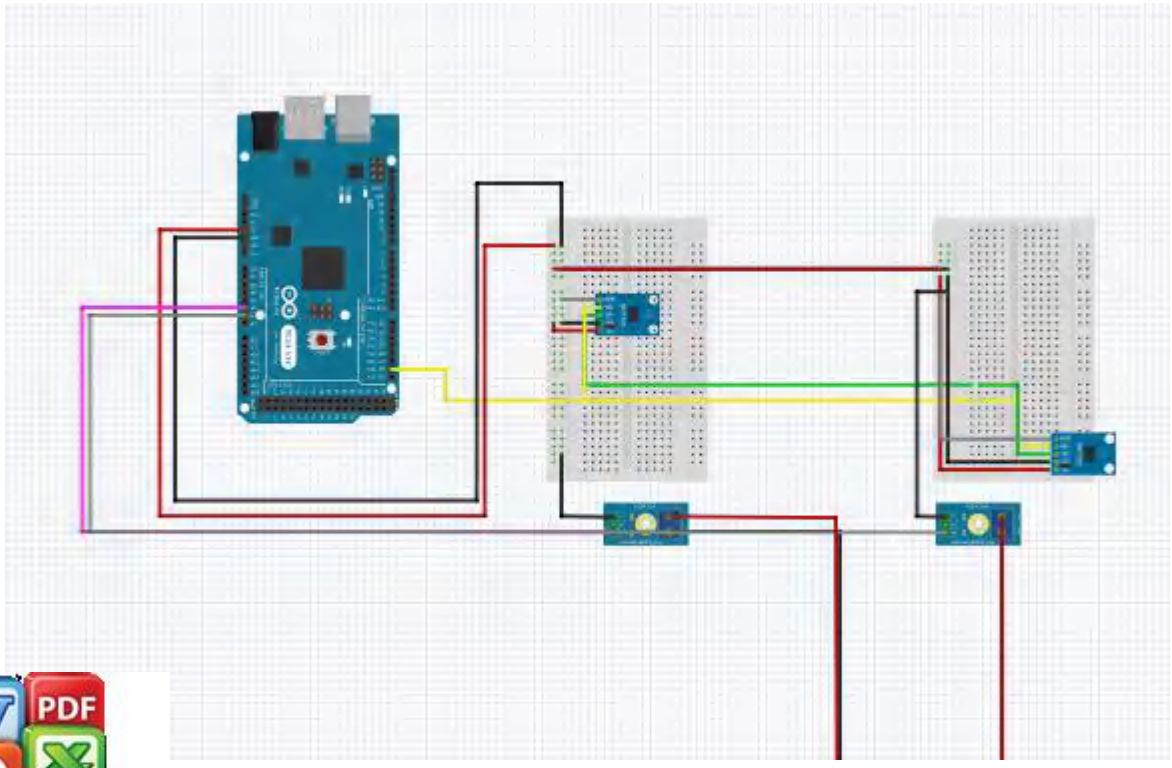
L298N to arduino mega 2560



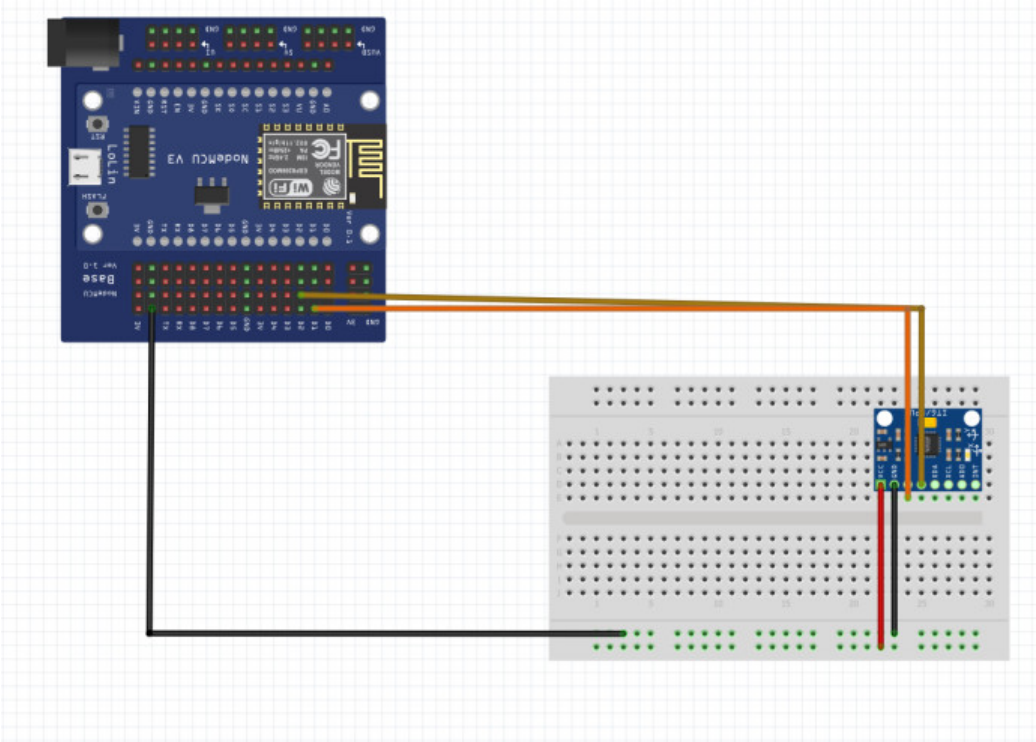
Sensor LDR (Light Dependent Resistor) to arduino mega 2560



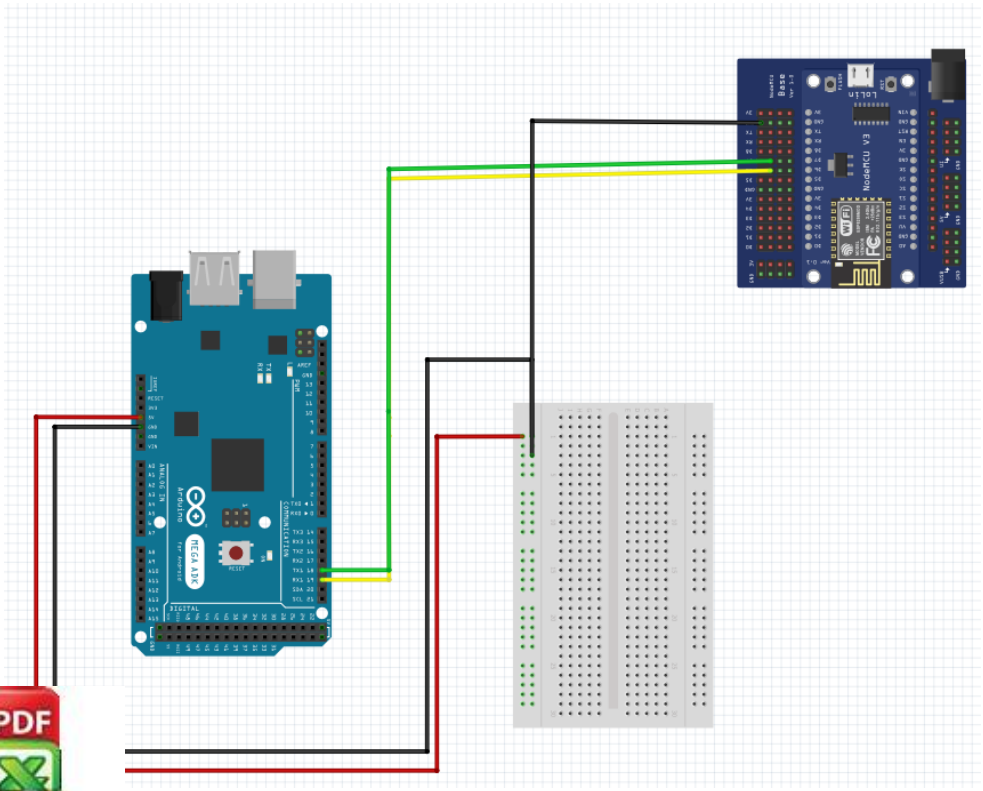
Monitoring Sensor (tegangan, intensitas) to arduino mega 2560



MPU6050 to Nodemcu Esp8266

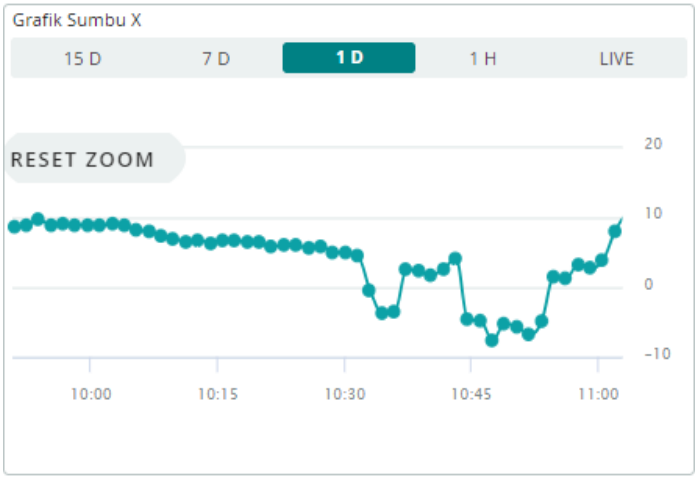


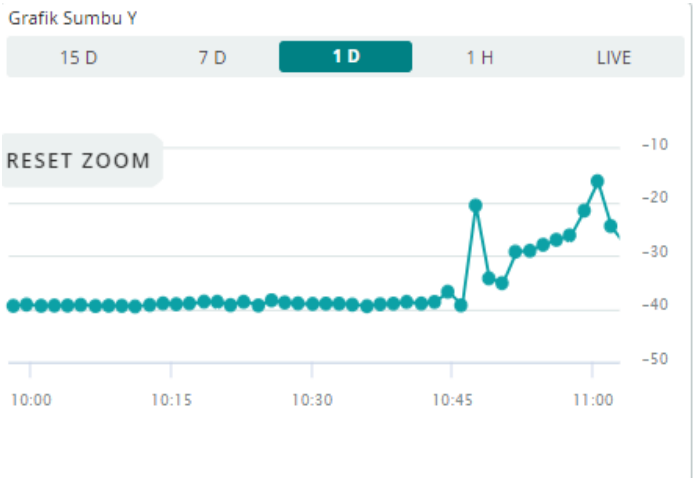
Nodemcu Esp8266 To arduino mega 2560

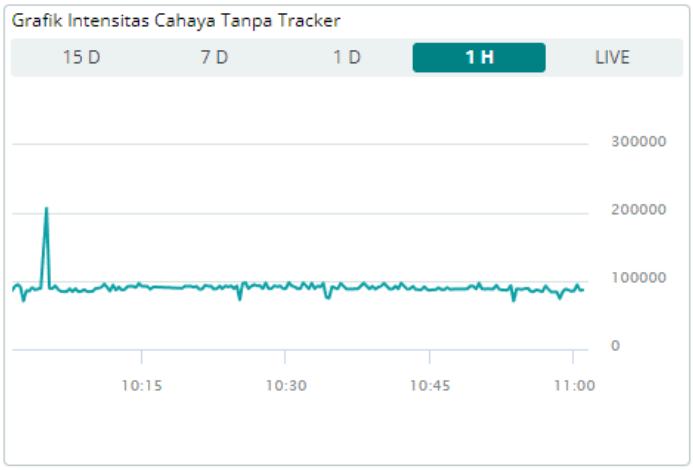


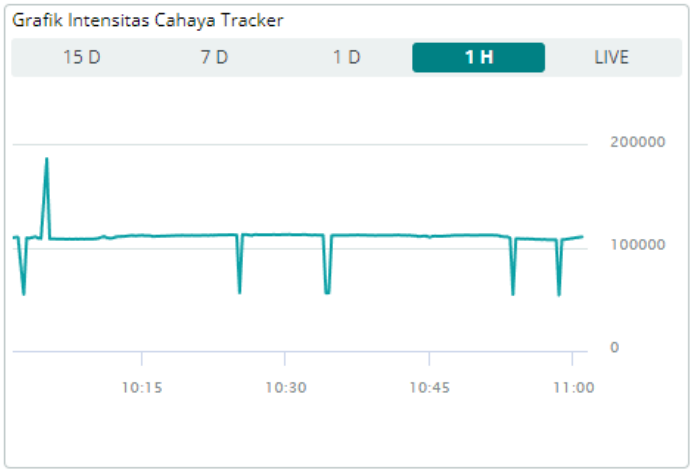
LAMPIRAN PROSES MONITORING DAN HASIL MONITORING

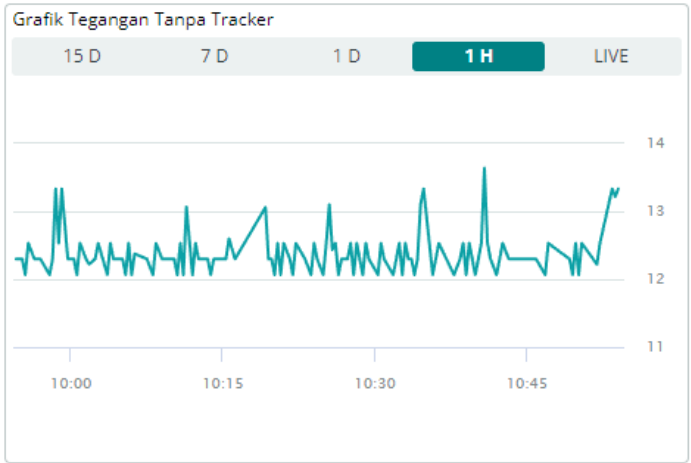
Pukul 10.00 – 11.00

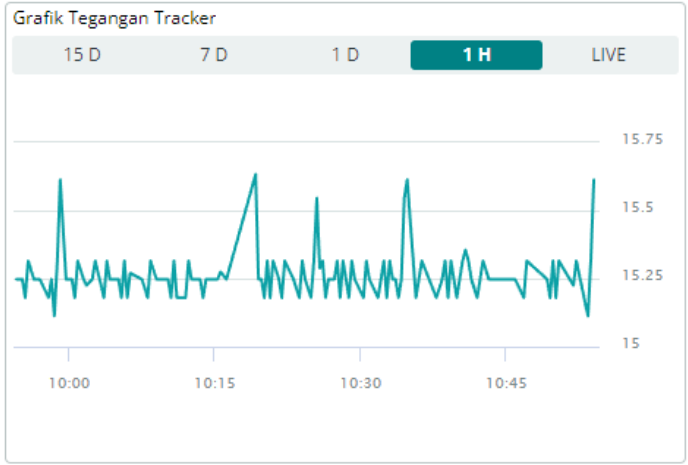




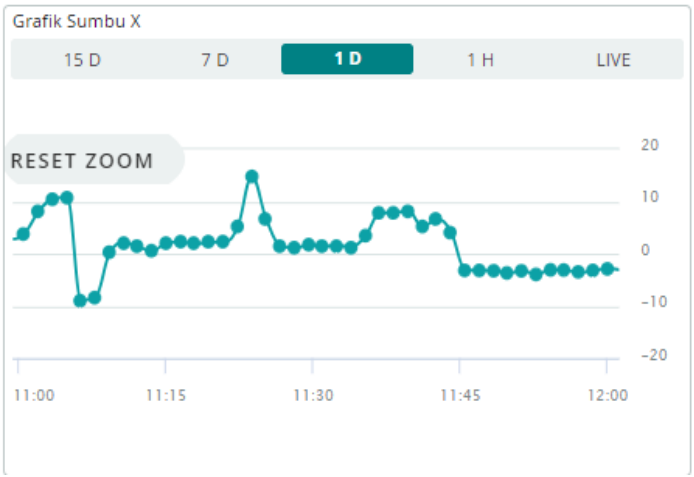


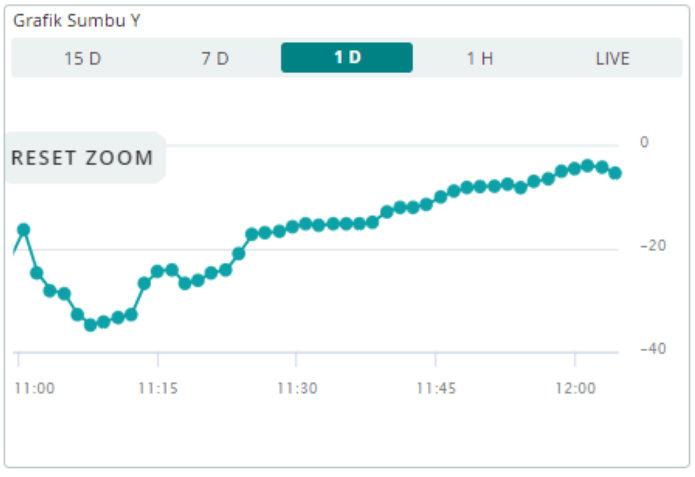


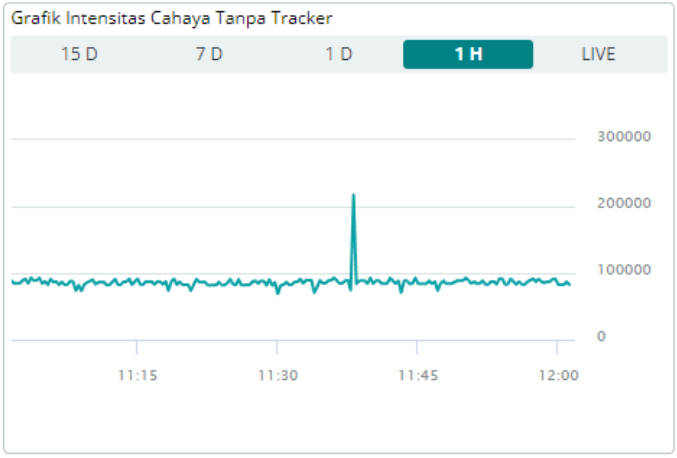


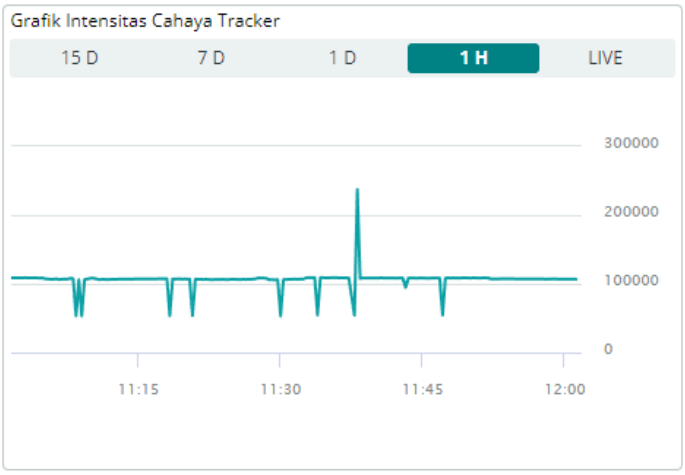


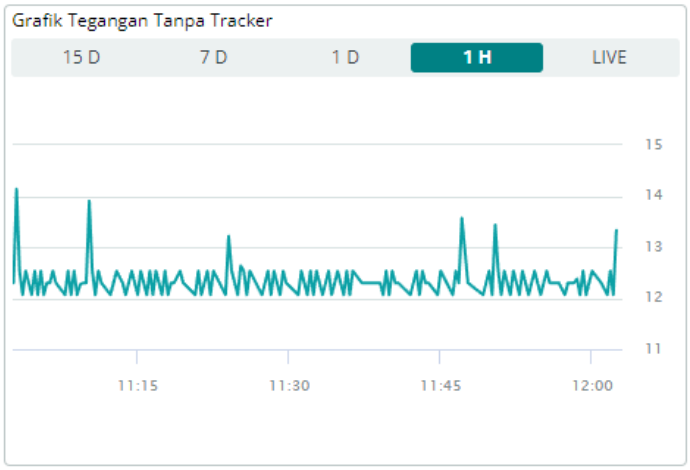
Pukul 11.00 – 12.00

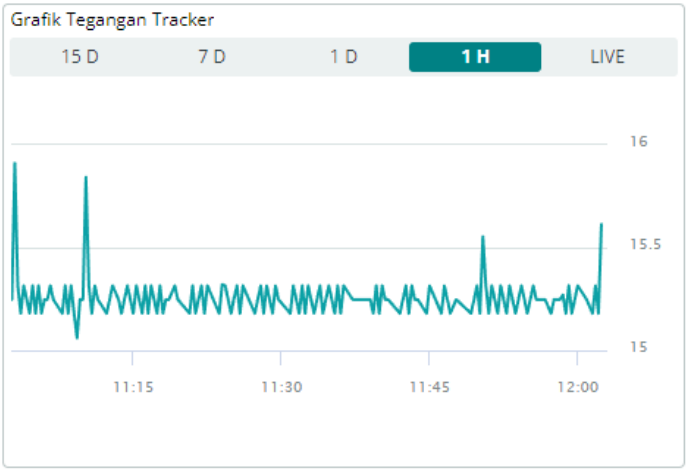




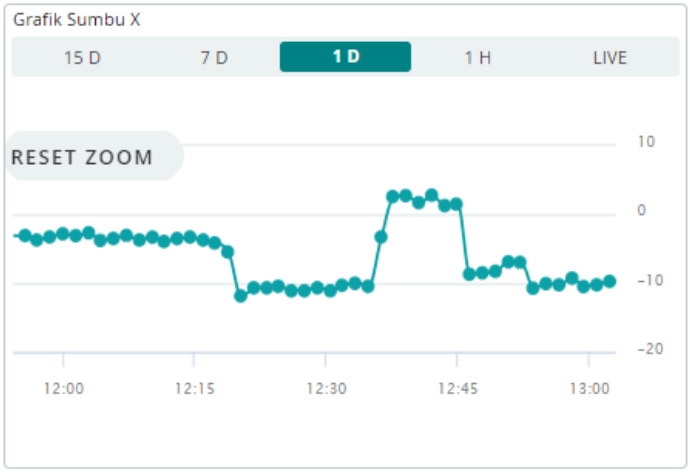


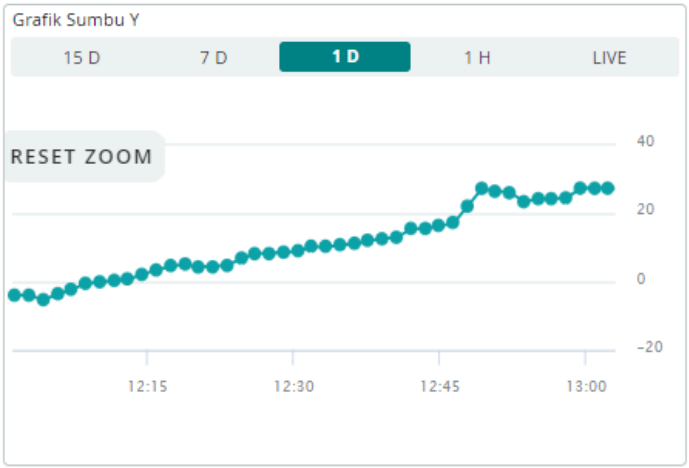


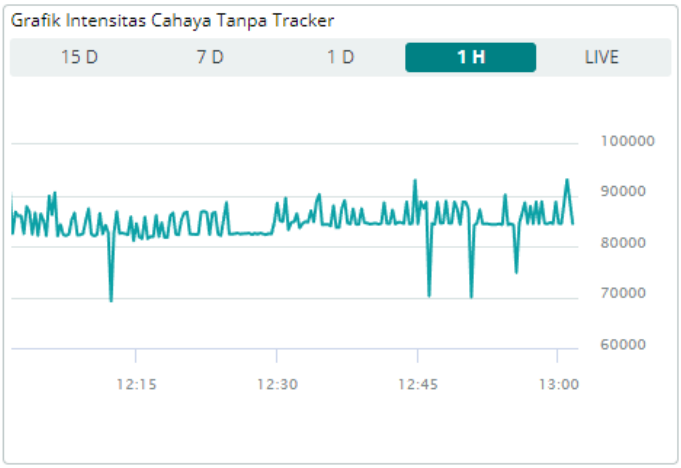


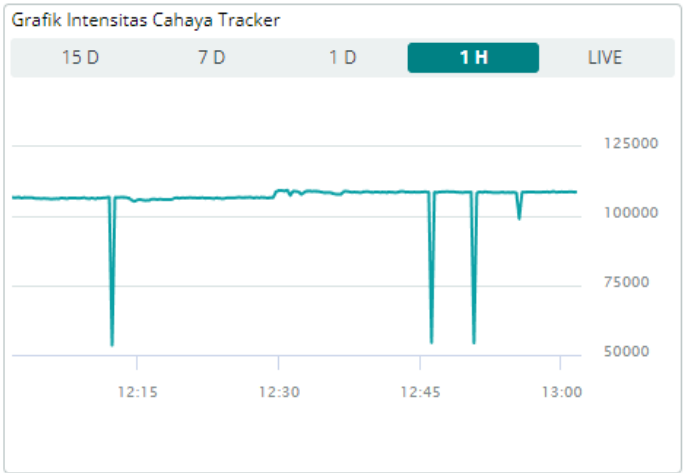


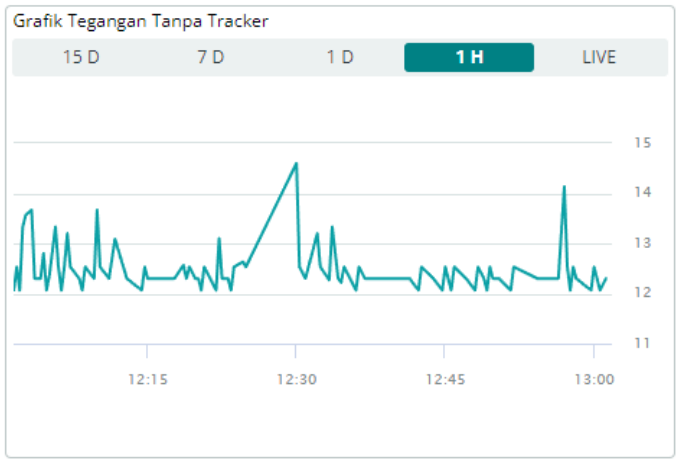
Pukul 12.00 – 13.00

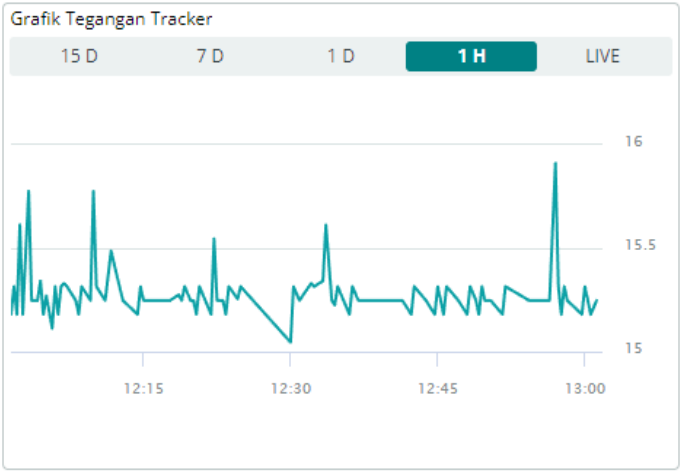




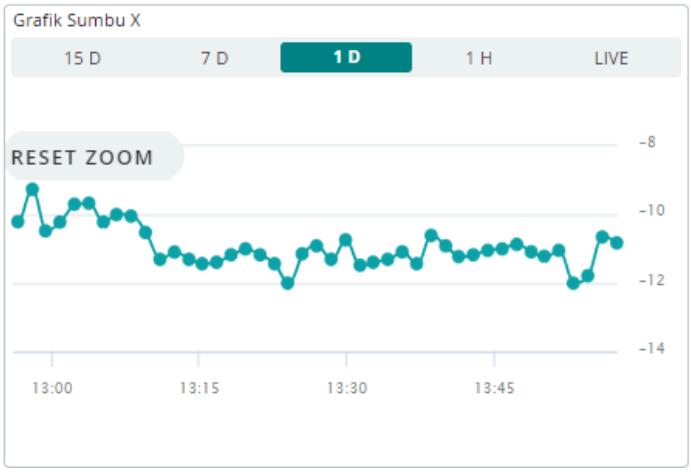


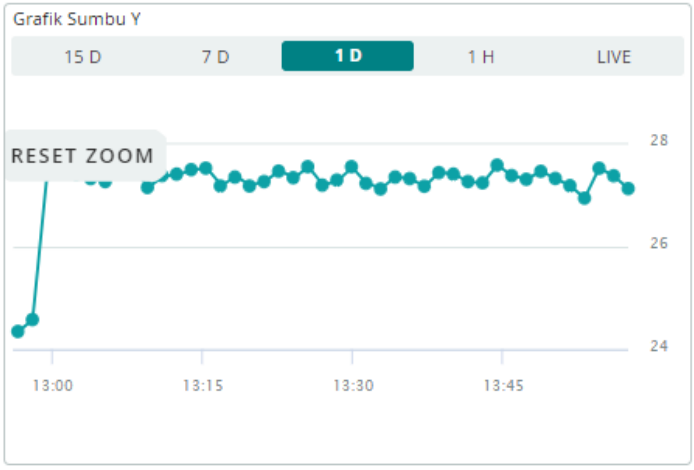


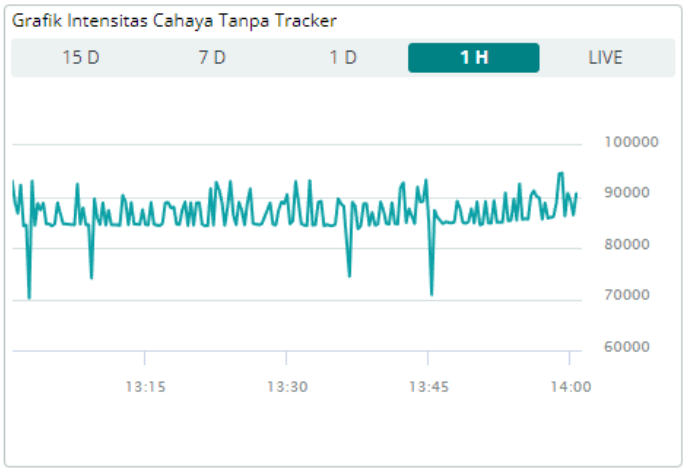


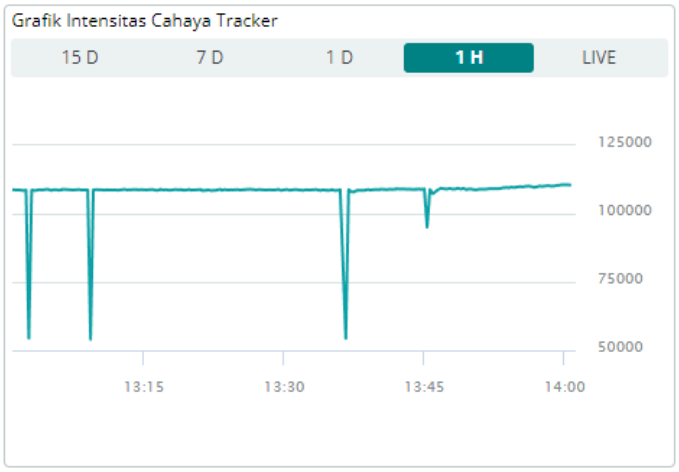


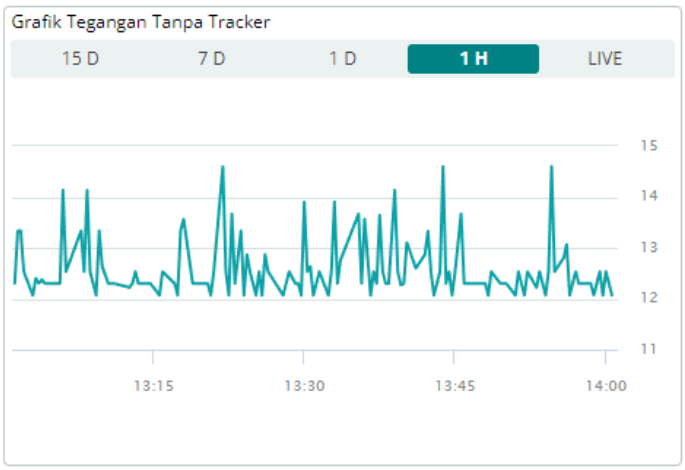
Pukul 13.00 – 14.00

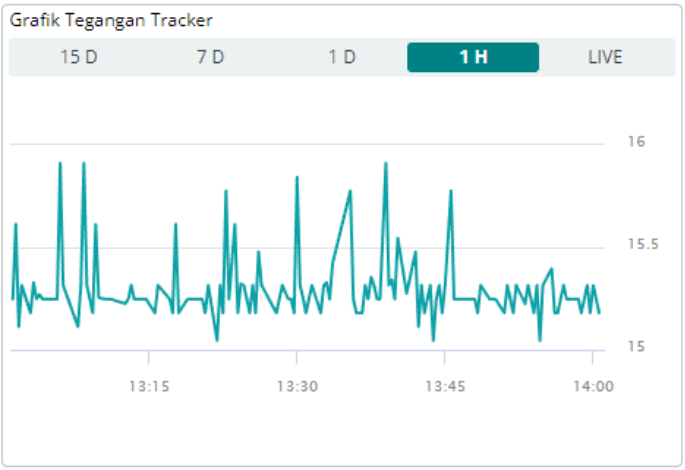












LAMPIRAN KODE PROGRAM ARDUINO

Program Arduino

Tab 1

```
#include <Wire.h>


////////////////////////////////////////////////////////////////////
///////////////PIN Sistem Monitoring Solar Tracker/////////////////
////////////////////////////////////////////////////////////////////

// Untuk Sensor BH1750
// Power
#define BH1750_POWER_DOWN 0x00  // No active state
#define BH1750_POWER_ON 0x01  // Waiting for measurement command
#define BH1750_RESET 0x07  // Reset data register value - not
accepted in POWER_DOWN mode

// Measurement Mode
#define CONTINUOUS_HIGH_RES_MODE 0x10  // Measurement at 1 lux
resolution. Measurement time is approx 120ms
#define CONTINUOUS_HIGH_RES_MODE_2 0x11  // Measurement at 0.5 lux
resolution. Measurement time is approx 120ms
#define CONTINUOUS_LOW_RES_MODE 0x13  // Measurement at 4 lux
resolution. Measurement time is approx 16ms
#define ONE_TIME_HIGH_RES_MODE 0x20  // Measurement at 1 lux
resolution. Measurement time is approx 120ms
#define ONE_TIME_HIGH_RES_MODE_2 0x21  // Measurement at 0.5 lux
resolution. Measurement time is approx 120ms
#define ONE_TIME_LOW_RES_MODE 0x23  // Measurement at 4 lux
resolution. Measurement time is approx 16ms

// I2C Address
#define BH1750_1_ADDRESS 0x23  // Sensor 1 connected to GND
#define BH1750_2_ADDRESS 0x5C  // Sensor 2 connected to VCC

// LED Blink
#define LED_PIN 13

// Definition of Variable
int16_t rawSensorData;
int16_t intensitas_Tanpa_Tracker;
        intensitas_Tracker;
        gangan_Tracker;
        orValue1;
        orValue2;
        blinkState = false;
```

```
// Pin input analog
int analogInPin1 = A5;
int analogInPin2 = A6;

// Definition of Variable
int16_t RawData;
int16_t SensorValue[2];
String str;


//////////////////////////////////////////////////////////////////////
//////////////////////PIN PENGGERAK SOLAR TRACKER//////////////////
//////////////////////////////////////////////////////////////////////

// Pin kontrol motor
const int ENA_PIN = 8;
const int ENB_PIN = 7;
const int IN1_PIN = 6;
const int IN2_PIN = 5;
const int IN3_PIN = 4;
const int IN4_PIN = 3;
const int ldr1 = A0;
const int ldr2 = A1;
const int ldr3 = A2;
const int ldr4 = A3;
const int ldr5 = A4;

int xsensor[] = {0, 25, -25, 0, 0};
int ysensor[] = {0, 0, 0, 25, -25};

// Parameter linear actuator
float sudutX = 0.0;
float sudutY = 0.0;
float SudutDerajatX = 0.0;
float SudutDerajatY = 0.0;
const float pi = 3.14159265;
const float maksPanjangActuator = 10.0; // Panjang maksimum linear
actuator (10 cm)
const unsigned long maxWaktu = 10000;   // Waktu maksimum (10.000
millis)

// Flag inisialisasi solar tracker
    tCompleted = false;


    i setup
    up() {
    .begin(9600);
    e(10, OUTPUT);
```

```
    Serial1.begin(9600);  // untuk komunikasi serial ke ESP8266


  // Inisialisasi library Wire untuk komunikasi I2C
  Wire.begin();

  pinMode(ENA_PIN, OUTPUT);
  pinMode(ENB_PIN, OUTPUT);
  pinMode(IN1_PIN, OUTPUT);
  pinMode(IN2_PIN, OUTPUT);
  pinMode(IN3_PIN, OUTPUT);
  pinMode(IN4_PIN, OUTPUT);

  digitalWrite(ENA_PIN, HIGH);
  digitalWrite(ENB_PIN, HIGH);

  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, HIGH);

  delay(500);

  // Inisialisasi solar tracker
  initSolarTracker();
}

// Fungsi loop
void loop() {
  // Jika inisialisasi sudah selesai, lanjutkan dengan motorControl
  if (initCompleted) {
    sudutmatahari();


    sendData();
  readAndProcessBH1750(BH1750_1_ADDRESS, 0);
  readAndProcessBH1750(BH1750_2_ADDRESS, 1);
  sensorValue1 = analogRead(analogInPin1);
  sensorValue2 = analogRead(analogInPin2);
  tegangan_Tracker = map(float(sensorValue1), 0, 1023, 0, 17);
  tegangan_tanpa_tracker = map(float(sensorValue2), 0, 1023, 0, 17);

Corrected map function

  // Display and send data
  Serial1.print("Intensitas Tanpa Tracker = ");
  Serial1.println(intensitas_Tanpa_Tracker);
  Serial1.print("Intensitas dengan Tracker = ");
  Serial1.println(intensitas_Tracker);
  Serial1.print("Tegangan tracker = ");
  Serial1.println(tegangan_Tracker);
  Serial1.print("Tegangan Tanpa tracker = ");
```

```
  Serial.println(tegangan_tanpa_tracker);

  blinkState = !blinkState;
  digitalWrite(LED_PIN, blinkState);

  // Send data to NodeMCU using Serial1
  Serial1.print(intensitas_Tanpa_Tracker);
  Serial1.print(",");
  Serial1.print(intensitas_Tracker);
  Serial1.print(",");
  Serial1.println(tegangan_Tracker);
  Serial1.print("Tegangan tanpa tracker = ");
  Serial1.println(tegangan_tanpa_tracker);
  delay(5000);  // delay untuk ke nodemcu

    // Kondisi untuk menjalankan motor
    if (abs(sudutX) > 0.1 || abs(sudutY) > 0.1) {
      motorControl();
    } // Kondisi dimana ketika sudut x maupun sudut y > 0.1 maka
eksekusi fungsi motorcontrol();
    delay(1000);
  }
}
```

Tab 2

```
////////////////////////////////////////////////////////////////////
//////////////////INISIASI MATAHARI (START)////////////////////////
////////////////////////////////////////////////////////////////////

void initSolarTracker() {
  Serial.println("Solar tracker initialization...");

  // Indikator pembacaan LDR untuk menandakan adanya cahaya matahari
  int maxSunlightReading = 950;

  // Gerakkan motor untuk mencari posisi matahari
  while (analogRead(ldr1) > maxSunlightReading) {
    // Gerakkan motor ke arah pertama
    digitalWrite(IN1_PIN, HIGH);
    digitalWrite(IN2_PIN, LOW);
    digitalWrite(IN3_PIN, HIGH);
    digitalWrite(IN4_PIN, LOW);
    delay(10000); // Anggap bahwa gerakan motor ini mencari posisi


    lentikan motor
    talWrite(IN1_PIN, LOW);
    talWrite(IN2_PIN, LOW);
    talWrite(IN3_PIN, LOW);
```

```
    digitalWrite(IN4_PIN, LOW);

    // Cek apakah sudah menemukan matahari
    if (analogRead(ldr1) <= maxSunlightReading) {
      Serial.println("Sunlight detected!");
      initCompleted = true; // Setel flag bahwa inisiasi sudah
selesai
      return;               // Keluar dari fungsi jika matahari
sudah ditemukan
    }

    // Tunggu sejenak sebelum memulai gerakan ke arah kedua
    delay(1000);

    // Gerakkan motor ke arah kedua
    digitalWrite(IN1_PIN, LOW);
    digitalWrite(IN2_PIN, HIGH);
    digitalWrite(IN3_PIN, HIGH);
    digitalWrite(IN4_PIN, LOW);
    delay(10000); // Anggap bahwa gerakan motor ini mencari posisi
matahari

    // Hentikan motor
    digitalWrite(IN1_PIN, LOW);
    digitalWrite(IN2_PIN, LOW);
    digitalWrite(IN3_PIN, LOW);
    digitalWrite(IN4_PIN, LOW);

    // Cek apakah sudah menemukan matahari
    if (analogRead(ldr1) <= maxSunlightReading) {
      Serial.println("Sunlight detected!");
      initCompleted = true; // Setel flag bahwa inisiasi sudah
selesai
      return;               // Keluar dari fungsi jika matahari
sudah ditemukan
    }

    // Tunggu sejenak sebelum memulai gerakan ke arah pertama lagi
    delay(1000);

   // Gerakkan motor ke arah ketiga
    digitalWrite(IN1_PIN, LOW);
    digitalWrite(IN2_PIN, HIGH);
    talWrite(IN3_PIN, LOW);
    talWrite(IN4_PIN, HIGH);
    y(10000); // Anggap bahwa gerakan motor ini mencari posisi

    Hentikan motor
```

```
    digitalWrite(IN1_PIN, LOW);
    digitalWrite(IN2_PIN, LOW);
    digitalWrite(IN3_PIN, LOW);
    digitalWrite(IN4_PIN, LOW);


    // Cek apakah sudah menemukan matahari
    if (analogRead(ldr1) <= maxSunlightReading) {
      Serial.println("Sunlight detected!");
      initCompleted = true; // Setel flag bahwa inisiasi sudah
selesai
      return;               // Keluar dari fungsi jika matahari
sudah ditemukan
    }

   // Gerakkan motor ke arah keempat
    digitalWrite(IN1_PIN, HIGH);
    digitalWrite(IN2_PIN, LOW);
    digitalWrite(IN3_PIN, LOW);
    digitalWrite(IN4_PIN, HIGH);
    delay(10000); // Anggap bahwa gerakan motor ini mencari posisi
matahari

     // Hentikan motor
    digitalWrite(IN1_PIN, LOW);
    digitalWrite(IN2_PIN, LOW);
    digitalWrite(IN3_PIN, LOW);
    digitalWrite(IN4_PIN, LOW);

     // Cek apakah sudah menemukan matahari
    if (analogRead(ldr1) <= maxSunlightReading) {
      Serial.println("Sunlight detected!");
      initCompleted = true; // Setel flag bahwa inisiasi sudah
selesai
      return;               // Keluar dari fungsi jika matahari
sudah ditemukan
    }

  // Jika tidak menemukan matahari setelah kondisi tertentu
  Serial.println("Sunlight not detected yet.");
}
}
```



```
dData() {
need for a separate function, sending data in the loop now.

dAndProcessBH1750(int address, int index) {
```

```
  initBH1750(address, CONTINUOUS_HIGH_RES_MODE);
  delay(120);
  rawSensorData = readRawDataBH1750(address);

  // Assign values based on index
  if (index == 0) {
    intensitas_Tanpa_Tracker = rawSensorData / 1.2;
  } else {
    intensitas_Tracker = rawSensorData / 1.2;
  }
}

void initBH1750(int address, int mode) {
  Wire.beginTransmission(address);
  Wire.write(mode);
  Wire.endTransmission(true);
}

int16_t readRawDataBH1750(int address) {
  Wire.beginTransmission(address);
  Wire.requestFrom(address, 2, true);
  int16_t rawData = Wire.read() << 8 | Wire.read();
  Wire.endTransmission(true);
  return rawData;
}
```

Tab 4

```
//////////////////////////////////////////////////////////////////
//////////////////////MENCARI NILAI SUDUT MATAHARI /////////////////
//////////////////////////////////////////////////////////////////

void sudutmatahari() {
  int nilaiSensor[] = {analogRead(ldr1), analogRead(ldr2),
analogRead(ldr3), analogRead(ldr4), analogRead(ldr5)};

  float xrata = 0.0, yrata = 0.0;
  float srata = 0.0;

  // Menghitung nilai rata-rata untuk xrata dan srata
   for (int i = 0; i < 5; i++) {
      xrata += nilaiSensor[i] * xsensor[i];
       ata += nilaiSensor[i];


    nghitung nilai rata-rata untuk yrata
     int i = 0; i < 5; i++) {
    ata += nilaiSensor[i] * ysensor[i];
```

```
      srata += nilaiSensor[i];
    }

  if (srata != 0) {
    xrata = xrata / srata;
    yrata = yrata / srata;


  SudutDerajatX = abs (atan(xrata / 6.5) * 180.0 / pi);
  SudutDerajatY = abs (atan(yrata / 6.5) * 180.0 / pi);

    sudutX = atan(xrata / 6.5);
    sudutY = atan(yrata / 6.5);

  }
}
```

Tab 5

```
////////////////////////////////////////////////////////////////////
/////////////MENCARI NILAI EXTEND (+) ATAU RETRACT (-) /////////////
////////////////////////////////////////////////////////////////////
float hitungPanjangExtendX(float sudutX) {
  sudutX = abs(sudutX);
  float HitungExtendX = sq(16 * sin(sudutX) + 21) + sq(16 *
cos(sudutX)) ;
  float panjangExtendX = sqrt(HitungExtendX) - 26;

  return panjangExtendX;
}

float hitungPanjangExtendY(float sudutY) {
  sudutY = abs(sudutY);
  float HitungExtendY = sq(16 *sin(sudutY)) - sq(16 * (1 -
cos(sudutY)));
  float panjangExtendY = sqrt(HitungExtendY);

  return panjangExtendY;
}


////////////////////////////////////////////////////////////////////
///////////////KONVERSI NILAI EXTEND MENJADI WAKTU ///////////////
////////////////////////////////////////////////////////////////////

:orControl() {
 panjangExtendX = hitungPanjangExtendX(sudutX);
 panjangExtendY = hitungPanjangExtendY(sudutY);
```

```
  unsigned long waktuExtendX = map(panjangExtendX, 0,
maksPanjangActuator, 0, maxWaktu);
  unsigned long waktuExtendY = map(panjangExtendY, 0,
maksPanjangActuator, 0, maxWaktu);

  kontrolMotor(waktuExtendX, waktuExtendY);
}

////////////////////////////////////////////////////////////////////
//////////////////////FUNGSI KONTROL MOTOR ////////////////////////
////////////////////////////////////////////////////////////////////

void kontrolMotor(unsigned long waktuExtendX, unsigned long waktuExtendY) {
  unsigned long startTimeX = millis();
  unsigned long startTimeY = millis();

  while ((millis() - startTimeX) < waktuExtendX || (millis() - startTimeY) <
waktuExtendY) {
    if (sudutX > 0) {
      digitalWrite(IN1_PIN, HIGH);
      digitalWrite(IN2_PIN, LOW);
      Serial.print("Linear Actuator X Sedang Bergerak: ");
      Serial.println(waktuExtendX);

    } else if (sudutX < 0) {
      digitalWrite(IN1_PIN, LOW);
      digitalWrite(IN2_PIN, HIGH);
      Serial.print("Linear Actuator X Sedang Bergerak: ");
      Serial.println(waktuExtendX);

    }

    if (sudutY > 0) {
      digitalWrite(IN3_PIN, HIGH);
      digitalWrite(IN4_PIN, LOW);
      Serial.print("Linear Actuator Y Sedang Bergerak: ");
      Serial.println(waktuExtendY);

    } else if (sudutY < 0) {
      digitalWrite(IN3_PIN, LOW);
      digitalWrite(IN4_PIN, HIGH);
      Serial.print("Linear Actuator Y Sedang Bergerak: ");
      Serial.println(waktuExtendY);
    }
  }

  digitalWrite(IN1_PIN, LOW);
  digitalWrite(IN2_PIN, LOW);
  digitalWrite(IN3_PIN, LOW);
       Write(IN4_PIN, LOW);

       println("Linear Actuator Berhenti Bergerak");
```

LAMPIRAN KODE PROGRAM ESP8266

```
#include "arduino_secrets.h"
/*
  Sketch generated by the Arduino IoT Cloud Thing "Untitled"
  https://create.arduino.cc/cloud/things/5dd0cf54-494b-4432-85e8-
7ec47cac8b52

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated
when changes are made to the Thing

  float intensitas_Tanpa_Tracker;
  float intensitas_Tracker;
  float rollX;
  float tegangan_tanpa_tracker;
  float tegangan_Tracker;

  Variables which are marked as READ/WRITE in the Cloud Thing will
also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end
of this sketch.
*/

#include "thingProperties.h"
#include <SoftwareSerial.h>
#include <Wire.h>
#include "MPU6050.h"

SoftwareSerial mySerial(D6, D7); // Assuming that your TX is
connected to D5 (GPIO5) and RX is connected to D53 (GPIO53)

float pitchY;
float intensitasTanpaTracker;
float lastIntensitasTanpaTracker;
float intensitasTracker;

////////////////////////////////////////////////////////////////
//////////////////////BUAT MPU6050//////////////////////////////
////////////////////////////////////////////////////////////////


// variabel buat mpu yang lain

:celOffsetX = 0.0, accelOffsetY = 0.0, accelOffsetZ = 0.0; //
ıccel (buat koreksi)
'roOffsetX = 55675.0, gyroOffsetY = 1.0, gyroOffsetZ = 0.0;
: gyro (buat koreksi)
'evRollX, prevPitchY; // buat [pengurangan]
AccelX, AccelY, AccelZ, Temperature, GyroX, GyroY, GyroZ;
```

```
////////////////////////////////////////////////////////////////
//////////////////////mpu6050/////////////////////////////////
////////////////////////////////////////////////////////////////

// Konfigurasi dan alamat register MPU6050
const uint8_t MPU6050SlaveAddress = 0x68;
const uint16_t AccelScaleFactor = 16384;
const uint16_t GyroScaleFactor = 131;

const uint8_t MPU6050_REGISTER_SMPLRT_DIV = 0x19;
const uint8_t MPU6050_REGISTER_PWR_MGMT_1 = 0x6B;
const uint8_t MPU6050_REGISTER_PWR_MGMT_2 = 0x6C;
const uint8_t MPU6050_REGISTER_CONFIG = 0x1A;
const uint8_t MPU6050_REGISTER_GYRO_CONFIG = 0x1B;
const uint8_t MPU6050_REGISTER_ACCEL_CONFIG = 0x1C;
const uint8_t MPU6050_REGISTER_FIFO_EN = 0x23;
const uint8_t MPU6050_REGISTER_INT_ENABLE = 0x38;
const uint8_t MPU6050_REGISTER_ACCEL_XOUT_H = 0x3B;
const uint8_t MPU6050_REGISTER_SIGNAL_PATH_RESET = 0x68;

bool initializationComplete = false;

void setup() {
  Serial.begin(9600);
  mySerial.begin(9600);
  Wire.begin();
  delay(1500);
  MPU6050_Init();
  initializationComplete = true;

  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();
  if (initializationComplete && mySerial.available()) {
    String receivedData = mySerial.readStringUntil('\n');
    processData(receivedData);
  }

cessData(String data) {
cess MPU6050 data
awValue(MPU6050SlaveAddress, 0x3B);  // Assuming starting
 address for sensor data
```

```
  double Ax = ((double)AccelX - accelOffsetX) / 16384.0 * 100;
  double Ay = ((double)AccelY - accelOffsetY) / 16384.0 * 100;
  rollX = Ax;
  pitchY = Ay;

  // Split the received data using ',' as the delimiter
  String values[4]; // Increased the array size to include all four
values
  int index = 0;
  int start = 0;

  for (int i = 0; i < data.length(); i++) {
    if (data.charAt(i) == ',') {
      values[index] = data.substring(start, i);
      start = i + 1;
      index++;
    }
  }
  values[index] = data.substring(start);

   // Convert string values to float
  intensitas_Tanpa_Tracker = values[0].toFloat();
  intensitas_Tracker = values[1].toFloat();
  tegangan_Tracker = values[2].toFloat();
  tegangan_tanpa_tracker = values[3].toFloat();

  // Map the intensitas values
  float mappedIntensitas_Tanpa_Tracker =
mapfloat(intensitas_Tanpa_Tracker, 0, 300, 80000, 95000);
  float mappedIntensitas_Tracker = mapfloat(intensitas_Tracker, 0,
300, 80000, 110000);

  float randomSubtractValue = random(15000, 25001);
  intensitas_Tanpa_Tracker = mappedIntensitas_Tracker -
randomSubtractValue;

  float randomTegangan_Tracker = random(15000, 16001) / 1000.0; //
Random value between 15 and 16
  float randomTegangan_Tanpa_Tracker = random(12000, 15001) /
1000.0; // Random value between 13 and 15

  // Adjust the voltage parameters
  tegangan_Tracker = randomTegangan_Tracker;
      gan_tanpa_tracker = randomTegangan_Tanpa_Tracker;

      splay the values only if they are not zero
      tensitas_Tanpa_Tracker != 0) {
      al.print("Intensitas Tanpa Tracker: ");
      al.println(intensitas_Tanpa_Tracker);
```

```
  }

  if (intensitas_Tracker != 0) {
    Serial.print("Intensitas dengan Tracker: ");
    Serial.println(mappedIntensitas_Tracker);

    // Update the cloud variable only if the value is not zero
    intensitas_Tracker = mappedIntensitas_Tracker;
  }

  if (tegangan_Tracker != 0) {
    Serial.print("Tegangan Tracker: ");
    Serial.println(tegangan_Tracker);
  }

  if (tegangan_tanpa_tracker != 0) {
    Serial.print("Tegangan Tanpa Tracker: ");
    Serial.println(tegangan_tanpa_tracker);
  }

  delay(5000);
}

void I2C_Write(uint8_t deviceAddress, uint8_t regAddress, uint8_t
data) {
  Wire.beginTransmission(deviceAddress);
  Wire.write(regAddress);
  Wire.write(data);
  Wire.endTransmission();
}

void Read_RawValue(uint8_t deviceAddress, uint8_t regAddress) {
  Wire.beginTransmission(deviceAddress);
  Wire.write(regAddress);
  Wire.endTransmission();
  Wire.requestFrom(deviceAddress, (uint8_t)14);
  AccelX = (Wire.read() << 8 | Wire.read());
  AccelY = (Wire.read() << 8 | Wire.read());
  AccelZ = (Wire.read() << 8 | Wire.read());
  Temperature = (Wire.read() << 8 | Wire.read());
  GyroX = (Wire.read() << 8 | Wire.read());
  GyroY = (Wire.read() << 8 | Wire.read());
  GyroZ = (Wire.read() << 8 | Wire.read());


I6050_Init() {
 150);
ite(MPU6050SlaveAddress, 0x19, 0x07);
ite(MPU6050SlaveAddress, 0x6B, 0x01);
```

```
  I2C_Write(MPU6050SlaveAddress, 0x6C, 0x00);
  I2C_Write(MPU6050SlaveAddress, 0x1A, 0x00);
  I2C_Write(MPU6050SlaveAddress, 0x1B, 0x00);
  I2C_Write(MPU6050SlaveAddress, 0x1C, 0x00);
  I2C_Write(MPU6050SlaveAddress, 0x23, 0x00);
  I2C_Write(MPU6050SlaveAddress, 0x38, 0x01);
}

float mapfloat(float x, float in_min, float in_max, float out_min,
float out_max) {
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) +
out_min;
}
```