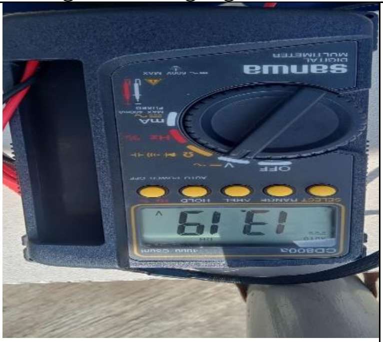
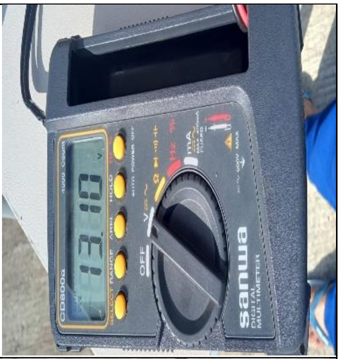
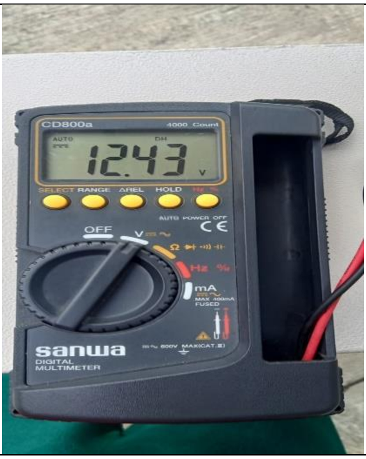
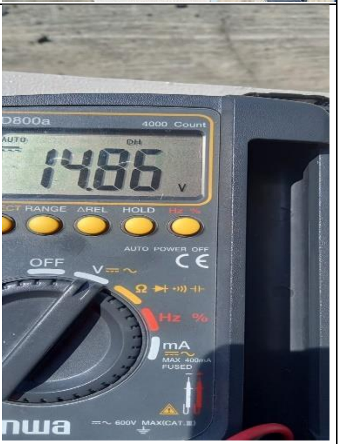
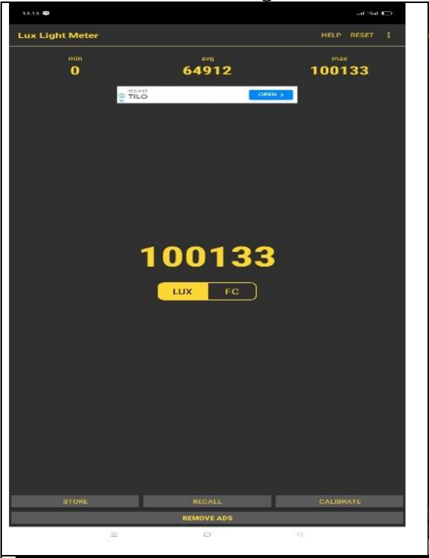
LAMPIRAN A

a. Pembuatan Rangkaian Badan Alat

b. Pengukuran Tegangan dan Arus

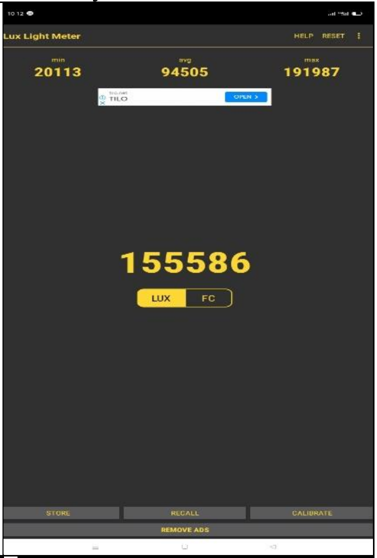
	
	

c. Pembacaan Pengukuran Intensitas Cahaya



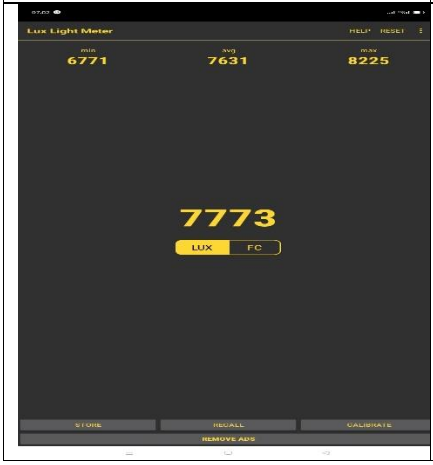
min	avg	max
0	64912	100133

100133



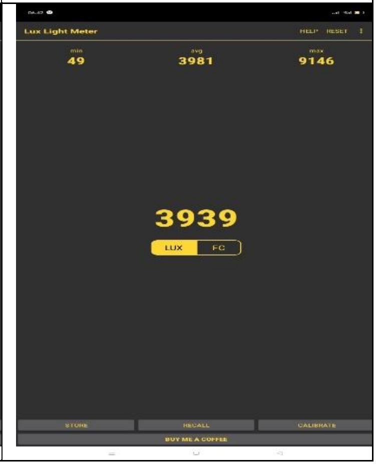
min	avg	max
20113	94505	191987

155586



min	avg	max
6771	7631	8225

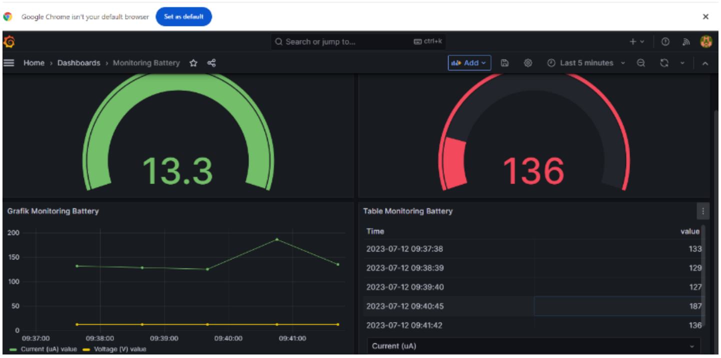
7773



min	avg	max
49	3981	9146

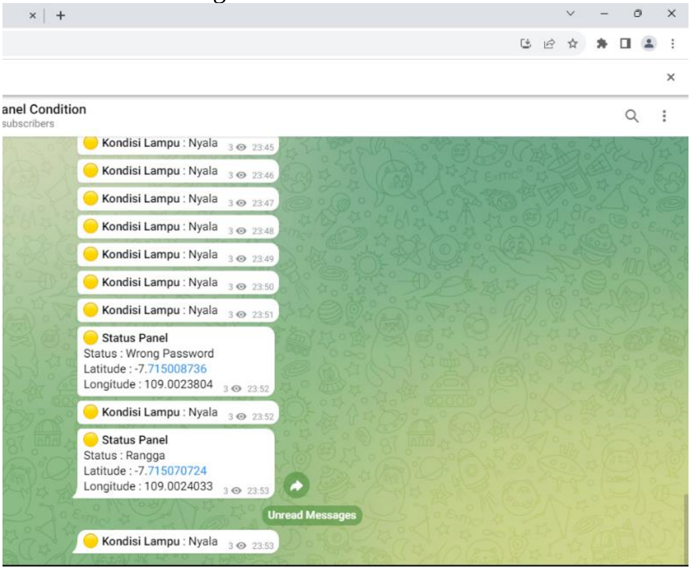
3939

d. Notifikasi Grafana



Time	value
2023-07-12 09:37:38	133
2023-07-12 09:38:39	129
2023-07-12 09:39:40	127
2023-07-12 09:40:45	187
2023-07-12 09:41:42	136

e. Notifikasi Telegram



Kondisi Lampu : Nyala 3 23:45

Kondisi Lampu : Nyala 3 23:46

Kondisi Lampu : Nyala 3 23:47

Kondisi Lampu : Nyala 3 23:48

Kondisi Lampu : Nyala 3 23:49

Kondisi Lampu : Nyala 3 23:50

Kondisi Lampu : Nyala 3 23:51

Status Panel
Status : Wrong Password
Latitude : -7.715008736
Longitude : 109.0023804 3 23:52

Kondisi Lampu : Nyala 3 23:52

Status Panel
Status : Rangga
Latitude : -7.715070724
Longitude : 109.0024033 3 23:53

Kondisi Lampu : Nyala 3 23:53

LAMPIRAN B

Berikut kode pemrograman :

a. ESP 1

```
#include <Arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>

// const char* ssid = "JMBA";
// const char* password = "iya bentar";
const char* ssid = "4G-UFI-17AD";
const char* password = "1234567890";
const char* mqtt_server = "broker.hivemq.com";


WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;
String Data;
String Data2;

////////////////////INA219//////////////////
#include "Wire.h"
#include "Adafruit_INA219.h"

Adafruit_INA219 ina219;
  float shuntvoltage = 0;
  float busvoltage = 0;
  float current_mA = 0;
  float loadvoltage = 0;
  float power_mW = 0;
////////////////////EPOCHTIME///////////////////
#include <NTPClient.h>
// change next line to use with another
board/shield
//#include <WiFi.h> // for WiFi shield
//#include <WiFi101.h> // for WiFi 101 shield or
MKR1000
#include <WiFiUdp.h>

WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);
unsigned long epochTime;
unsigned long getTime() {
  timeClient.update();
  unsigned long now = timeClient.getEpochTime();
  return now;
}

/////////////////////VOLT//////////////////////
int voltPin = 35; // pin arduino yang terhubung
dengan pin S modul sensor tegangan

float Vmodul = 0.0;
float hasil = 0.0;
float R1 = 30000.0; //30k
float R2 = 7500.0; //7500 ohm resistor
int Voltvalue = 0;
```

```
//////////////////////LDR////////////////////////
#define LIGHT_SENSOR_PIN1 32
#define LIGHT_SENSOR_PIN2 33
#define OUT3 19
bool conditionLamp;
//////////////////////GPS////////////////////////
#include <TinyGPSPlus.h>
#include <SoftwareSerial.h>

static const int RXPin = 25, TXPin = 23;
static const uint32_t GPSBaud = 9600;

// The TinyGPS++ object
TinyGPSPlus gps;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);
float longitude;
float latitude;
bool pindah;

////////////////////MQTT//////////////////////
void setup_WiFi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA);
 WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
```

```
   {
    delay(500);
    Serial.print(".");
   }
  randomSeed(micros());
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
void callback(char* topic, byte* payload, unsigned
int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
  // Switch on the LED if an 1 was received as
first character
  // if ((char)payload[0] == '1') {
  //   digitalWrite(BUILTIN_LED, LOW);   // Turn
the LED on (Note that LOW is the voltage level
  //   // but actually the LED is on; this is
because
  //   // it is active low on the ESP-01)
  // } else {
  //   digitalWrite(BUILTIN_LED, HIGH);  // Turn
the LED off by making the voltage HIGH
```

```
  // }
}
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      // Once connected, publish an
announcement...
      client.publish("outTopic", "hello world");
      // ... and resubscribe
      client.subscribe("inTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void GPS(){
  while (ss.available() > 0){
    gps.encode(ss.read());
    if (gps.location.isUpdated()){
      Serial.print("Latitude= ");
      Serial.print(gps.location.lat(), 6);
      latitude = gps.location.lat();
      Serial.print(" Longitude= ");
      Serial.println(gps.location.lng(), 6);
      longitude = gps.location.lng();
    }
  }
}

uint16_t LDR(int Pin_LDR)
{
  int analogValue = analogRead(Pin_LDR);

  // Serial.print("Analog Value = ");
  // Serial.print(analogValue);   // the raw
analog reading

  // We'll have a few threshholds, qualitatively
determined
  // if (analogValue < 40) {
  //   Serial.println(" => Dark");
  // } else if (analogValue < 800) {
  //   Serial.println(" => Dim");
  // } else if (analogValue < 2000) {
  //   Serial.println(" => Light");
  // } else if (analogValue < 3200) {
  //   Serial.println(" => Bright");
  // } else {
  //   Serial.println(" => Very bright");
  // }
  return analogValue;
}

void lampu(){
  int valueLDR1 = LDR(LIGHT_SENSOR_PIN1);
  int valueLDR2 = LDR(LIGHT_SENSOR_PIN2);
  Serial.print ("LDR1 = ");
  Serial.println (valueLDR1);
  Serial.print ("LDR2 = ");
  Serial.println (valueLDR2);
  if (valueLDR1 >= 1000){
    digitalWrite(OUT3, LOW);
  }
  else if (valueLDR1 <= 1000){
    digitalWrite(OUT3, HIGH);
  }
  if (valueLDR2 <= 1000){
      Serial.println ("Lampu nyala");
      conditionLamp = 1;
  }
  else if (valueLDR2 >= 1000){
      Serial.println ("Lampu mati");
      conditionLamp = 0;
  }
}

void INA219(){
  shuntvoltage = ina219.getShuntVoltage_mV();
  busvoltage = ina219.getBusVoltage_V();
  current_mA = ina219.getCurrent_mA();
  power_mW = ina219.getPower_mW();
```

```
loadvoltage = busvoltage + (shuntvoltage / 1000);

  Serial.print(busvoltage); Serial.print("\t");
  Serial.print(shuntvoltage); Serial.print("\t");
  Serial.print(loadvoltage); Serial.print("\t");
  Serial.print(current_mA); Serial.print("\t");
  Serial.println(power_mW);

  delay(1000);
}

void volt(){
   Voltvalue = analogRead(voltPin);
   Vmodul = (Voltvalue * 4.3) / 1024.0;
   hasil = Vmodul / (R2/(R1+R2));

  Serial.print("Tegangan keluaran modul = ");
  Serial.print(Vmodul,2);
  Serial.print("volt");
  Serial.print(", Hasil pengukuran = ");
  Serial.print(hasil,2);
  Serial.println("volt");
  // delay(500);
}


void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
////////////////////Volt////////////////////
  pinMode(voltPin, INPUT);
```

```
/////////////////////LDR/////////////////////
  pinMode(LIGHT_SENSOR_PIN1, INPUT);
  pinMode(LIGHT_SENSOR_PIN2, INPUT);
  pinMode(OUT3, OUTPUT);
  digitalWrite (OUT3, HIGH);
/////////////////////INA219////////////////////
    if (! ina219.begin()) {
    Serial.println("Failed to find INA219 chip");
    while (1) { delay(10); }
  }
/////////////////////GPS/////////////////////
  ss.begin(GPSBaud);
////////////////////MQTT/////////////////////
  setup_WiFi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
///////////////////EPOCHTIME//////////////////////
/
  while ( WiFi.status() != WL_CONNECTED ) {
    delay ( 500 );
    Serial.print ( "." );
  }

  timeClient.begin();
}

void loop() {
  // put your main code here, to run repeatedly:
  // GPS();
  INA219();
  volt();
  lampu();
  // epochTime = getTime();
  // Data =
Data+epochTime+";"+Vmodul+";"+nilaiarus;
  DynamicJsonDocument doc(1024);

  doc["time"]    = epochTime;
  doc["data"][0] = Vmodul;
  doc["data"][1] = current_mA;
  // // doc["data"][2] = pindah;
  // doc["data"][2] = longitude;
  // doc["data"][3] = latitude;
  doc["data"][2] = conditionLamp;



  serializeJson(doc, Data);
  // DynamicJsonDocument doc2(1024);

  // doc2["time"]    = epochTime;
  // // doc2["data"][0] = Vmodul;
  // // doc2["data"][1] = nilaiarus;
  // // doc["data"][2] = pindah;
  // doc2["data"][1] = longitude;
  // doc2["data"][2] = latitude;
  // doc2["data"][3] = ID;


  // serializeJson(doc2, Data2);

  if (!client.connected()) {
```

```
reconnect();
  }
  client.loop();
  unsigned long now = millis();
  if (now - lastMsg > 60000) {
    lastMsg = now;
    ++value;
    client.publish("TEST", String(Data).c_str());



    // delay(10);

  }
  Data = "";
}

// put function definitions here

```

b. Esp 2

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <Arduino.h>
#include <ArduinoJson.h>
// Update these with values suitable for your
network.

const char* ssid = "4G-UFI-17AD";
const char* password = "1234567890";
const char* mqtt_server = "broker.hivemq.com";
```

```
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
int value = 0;
String Data;
String Data2;
////////////////////EPOCHTIME////////////////////
#include <NTPClient.h>
// change next line to use with another
board/shield
//#include <WiFi.h> // for WiFi shield
//#include <WiFi101.h> // for WiFi 101 shield or
MKR1000
#include <WiFiUdp.h>

WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);
unsigned long epochTime;
unsigned long getTime() {
  timeClient.update();
  unsigned long now = timeClient.getEpochTime();
  return now;
}

//////////////////////RELAY////////////////////
#define OUT1 32
#define OUT2 33
#define OUT3 19
unsigned long waktu1 = 0;
unsigned long waktu2 = 0;
unsigned long waktu3 = 0;

///////////////////////KEYPAD////////////////////
// #define ROW_NUM     4 // four rows
// #define COLUMN_NUM  4 // four columns
#include <Keypad.h>
const byte ROW_NUM = 4;
const byte COLUMN_NUM = 4;
char keys[ROW_NUM][COLUMN_NUM] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte pin_rows[ROW_NUM]      = {13, 18, 5, 17}; //
GIOP19, GIOP18, GIOP5, GIOP17 connect to the row
pins
byte pin_column[COLUMN_NUM] = {16, 4, 0, 2};   //
GIOP16, GIOP4, GIOP0, GIOP2 connect to the column
pins

Keypad Keypad = Keypad( makeKeymap(keys),
pin_rows, pin_column, ROW_NUM, COLUMN_NUM );

const String Rangga = "123A"; // change your
password here
const String budi = "789C"; // change your
password here
```

```
const String Dita = "ACCD";
String input_password;
String ID;

/////////////////////LCD I2C/////////////////////
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
int cursorColumn = 0;

/////////////////////PIR///////////////////////
#define s_pir 26
bool PIR;
int Maintenance = 0;
/////////////////////GPS///////////////////////
#include <TinyGPSPlus.h>
#include <SoftwareSerial.h>

static const int RXPin = 12, TXPin = 14;
static const uint32_t GPSBaud = 9600;

// The TinyGPS++ object
TinyGPSPlus gps;

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);
float longitude;
float latitude;
bool pindah;

/////////////////////DOORR/////////////////////
#define DOORR_SENSOR_PIN 25
```

```
int DoorrState;

void setup_WiFi() {

  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned
int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  // // Switch on the LED if an 1 was received as
first character
  // if ((char)payload[0] == '1') {
  //   digitalWrite(BUILTIN_LED, LOW);   // Turn
the LED on (Note that LOW is the voltage level
  //   // but actually the LED is on; this is
because
  //   // it is active low on the ESP-01)
  // } else {
  //   digitalWrite(BUILTIN_LED, HIGH);  // Turn
the LED off by making the voltage HIGH
  // }

}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
```

```
if (client.connect(clientId.c_str())) {
      Serial.println("connected");
      // Once connected, publish an
announcement...
      // client.publish("outTopic", "hello
world");
      // ... and resubscribe
      client.subscribe("inTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}

void kirimData(){
  DynamicJsonDocument doc(1024);

  // doc["time"]   = epochTime;
  doc["ID"][0] = ID;
  doc["latitude"] = latitude;
  doc["longitude"] = longitude;
  // // doc["data"][2] = pindah;

  // doc["data"][3] = latitude;
  // doc["data"][4] = ID;
serializeJson(doc, Data);
  client.publish("TEST1", String(Data).c_str());
```

```
delay(10);
  Data = "";
}


void Keypad()
{
  char key = Keypad.getKey();
  // delay (300);
  if (key) {
    Serial.println(key);
    lcd.clear();
    lcd.setCursor(cursorColumn, 0);
    lcd.print(key);
    cursorColumn++;                 // move cursor
to next position
    if(cursorColumn == 16) {        // if reaching
limit, clear LCD
      lcd.clear();
      cursorColumn = 0;
    }
    if (key == '*') {
      input_password = ""; // clear input password
      lcd.clear();
      Maintenance = 1;
    } else if (key == '#') {

      if (Rangga == input_password) {
        Serial.println("The password is correct,
ACCESS GRANTED!");
        // lcd.setCursor(0, 1);
// lcd.print("ID = Rangga");
        ID = "Rangga";
        kirimData();
        Maintenance = 1;
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Maintenance");
        lcd.setCursor(0, 1);
        lcd.print("Rangga");
        digitalWrite (OUT2, LOW);
        delay (20000);
        Maintenance = 0;
        digitalWrite (OUT2, HIGH);
      }
      else if (budi == input_password){
        Serial.println("The password is correct,
ACCESS GRANTED!");
        ID = "budi";
        kirimData();
        Maintenance = 1;
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Maintenance");
        lcd.setCursor(0, 1);
        lcd.print("budi");
        digitalWrite (OUT2, LOW);
        delay (20000); Maintenance = 0;
        digitalWrite (OUT2, HIGH);
      }
```

```
else if (Dita == input_password){
        Serial.println("The password is correct,
ACCESS GRANTED!");
        ID = "Dita";
        kirimData();
        Maintenance = 1;
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Maintenance");
        lcd.setCursor(0, 1);
        lcd.print("Dita");
        digitalWrite (OUT2, LOW);
        delay (20000);
        Maintenance = 0;
        digitalWrite (OUT2, HIGH);
        }



       else {
        Serial.println("The password is incorrect,
ACCESS DENIED!");
        Maintenance = 2;
      }
      input_password = ""; // clear input password
    } else {
      input_password += key; // append new
character to input password string
    }
  }
}
```

```
void GPS(){
  while (ss.available() > 0){
    gps.encode(ss.read());
    if (gps.location.isUpdated()){
      Serial.print("Latitude= ");
      Serial.print(gps.location.lat(), 6);
      latitude = gps.location.lat();
      Serial.print(" Longitude= ");
      Serial.println(gps.location.lng(), 6);
      longitude = gps.location.lng();
    }
  }
}
void pir()
{
  bool state_pir = digitalRead(s_pir);

  if (state_pir == 0){
  // Serial.println("Tidak Terdeteksi Makhluk
Hidup");
  PIR = 0;
  // digitalWrite(p_relay, HIGH); // off karena
aktif low
  }
  else {
  // Serial.println("Terdeteksi Makhluk
Hidup!");
  PIR = 1;
// digitalWrite(p_relay, LOW); // on karena aktif
low
```

```
  }
}

void DoorrValue()
{
  DoorrState = digitalRead(DOORR_SENSOR_PIN); //
read state

  // if (DoorrState == HIGH) {
  //   Serial.println("The Doorr is open");
  // } else {
  //   Serial.println("The Doorr is closed");
  // }
}


void logicPintu (){
  // GPS();
  pir();
  Keypad();
    if (PIR == 1 && Maintenance == 1)
    {
      // GPS();
      // Serial.println("Sedang Maintenance");
      // digitalWrite (OUT2, HIGH);
      // delay (5000);
      // Maintenance = 0;
      // digitalWrite (OUT2, HIGH);
      // lcd.clear();
      // lcd.setCursor(0, 0);
      // lcd.print("Maintenance");
      // delay (2000);
      // lcd.clear();
    }
    else if (PIR == 0 && Maintenance == 1)
    {
      // GPS();
      // Serial.println("Sedang Maintenance");
      // digitalWrite (OUT2, HIGH);
      // lcd.clear();
      // lcd.setCursor(0, 0);
      // lcd.print("Maintenance");
      // delay (2000);
      // lcd.clear();
    }
    else if (PIR == 1 && Maintenance == 2)
    {
      // GPS();
      // Serial.println("PEMBOBOLAN");
      ID = "PEMBOBOLAN";
      kirimData();
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("PEMBOBOLAN");
      digitalWrite (OUT1, LOW);
      delay (20000);
      lcd.clear();
      Maintenance = 0;
      digitalWrite (OUT1, HIGH);
    }
    else if (PIR == 0 && Maintenance == 0)
{
```

```
// GPS();
      // Serial.println("No Problem");
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("No Problem");
      ID = "No Problem";
      DynamicJsonDocument doc(1024);
      // doc["time"]   = epochTime;
      doc["ID"][0] = ID;
      doc["latitude"] = latitude;
      doc["longitude"] = longitude;
      // // doc["data"][2] = pindah;

      // doc["data"][3] = latitude;
      // doc["data"][4] = ID;

      serializeJson(doc, Data);
      unsigned long now = millis();
      if (now - lastMsg > 60000) {
        lastMsg = now;
        ++value;
        client.publish("TEST1", 
String(Data).c_str());
        // delay(10);
      }
      Data = "";
      delay (2000);
      lcd.clear();
    }
    else if (PIR == 1 && Maintenance == 0) 
{
```

```
// GPS();
      // Serial.println("ADA ORANG");
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("    Welcome");
      lcd.setCursor(0, 1);
      lcd.print(" Enter Password");
    }
}


void setup() {
  lcd.init(); // initialize the lcd
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("    Welcome");
  lcd.setCursor(0, 1);
  lcd.print(" Enter Password");
  Serial.begin(9600);
  input_password.reserve(32); // maximum input
characters is 33, change if needed
/////////////////////PIR//////////////////////
  pinMode(s_pir, INPUT);
/////////////////////GPS/////////////////////
  ss.begin(GPSBaud);
/////////////////////DOORR//////////////////////
  pinMode(DOORR_SENSOR_PIN, INPUT_PULLUP);
/////////////////////RELAY/////////////////////
  pinMode(OUT1, OUTPUT);
  pinMode(OUT2, OUTPUT);
pinMode(OUT3, OUTPUT);
  digitalWrite (OUT1, HIGH);
  digitalWrite (OUT2, HIGH);
  digitalWrite (OUT3, HIGH);

  // pinMode(BUILTIN_LED, OUTPUT);     //
Initialize the BUILTIN_LED pin as an output
  Serial.begin(9600);
  // setup_WiFi();
  // client.setServer(mqtt_server, 1883);
  // client.setCallback(callback);
/////////////////MQTT/////////////////////
  setup_WiFi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
/////////////////EPOCHTIME//////////////////////
  // while ( WiFi.status() != WL_CONNECTED ) {
  //   delay ( 500 );
  //   Serial.print ( "." );
  // }

  // timeClient.begin();
}

void loop() {
  GPS();
  // Keypad();
  logicPintu();
  DoorrValue();

  // epochTime = getTime();
  // Data =
Data+epochTime+";"+Vmodul+";"+nilaiarus;


  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  // unsigned long now = millis();
  // if (now - lastMsg > 60000) {
  //   lastMsg = now;
  //   ++value;
  //   client.publish("TEST1",
String(Data).c_str());
  //   // delay(10);
  // }
  // Data = "";
}
```