# DAFTAR PUSTAKA

Faleva, M. R., Santoso, D. B., & Nurpulaela, Lela. (2020). Sistem Monitoring Energi Listrik pada Kompor Penghasil Listrik dengan Teknologi Internet of Things (KOLISS-IoT). Karawang: Universitas Singaperbangsa Karawang.

Haris, M. Y., & Putra, A. (2017) Perancangan Sistem Kontrol Lampu Berbasis Mikrokontroler Arduino Uno R3 dengan Sensor Suara. Makassar: Fakultas Teknik Universitas Muhammadiyah Makassar.

Hendranata, Anton. (2003). "ARIMA (Autoregressive Moving Average)". Jakarta: Manajemen Keuangan Sektor Publik FEUI

Junaidy, Benny. (2019). Pendeteksi dan Penetralisir Debu dan Asap pada Udara menggunakan Sensor GP2Y1010AU0F dan MQ-2 Berbasis Arduino Uno R3 AT Mega 328P. Medan: Fakultas MIPA Universitas Sumatera Utara.

Nugroho, A., Asyroh, M. F., Pangestu, A., & Wulandari, B. (2018). Pengatur Suhu dan Kelembaban Kumbung Jamur Otomatis. Yogyakarta: Fakultas Teknik Universitas Negeri Yogyakarta.

Riyanto, Slamet Riky. (2017). Rancang Bangun Alat Kontrol Suhu Dan Kelembaban Pada Fermentasi Tempe Kedelai Berbasis Mikrokontroler. Palembang: Politeknik Negeri Sriwijaya.

Sungkawa, Iwa, & Megasari, Ries. (2011). Penerapan Ukuran Ketepatan Nilai Ramalan Data Deret Waktu dalam Seleksi Model Peramalan Volume Penjualan PT Satriamandiri Citramulia. Jakarta: Departemen Matematika dan Statistika Universitas Bina Nusantara.

Suryana, Taryana. (2021). Measuring Light Intensity Using the BH1750 Sensor. Bandung: Teknik Informatika Universitas Komputer Indonesia.

wikipedia.org. (2022, 10 Maret). Internet untuk Segala. diakses pada 10 Maret 2022, dari https://id.m.wikipedia.org/wiki/Internet_untuk_Segala

aws.amazon.com. (2022, 11 Maret). IoT Analytics. diakses pada 11 Maret 2022, dari https://aws.amazon.com/id/iot-analytics/

# LAMPIRAN

**Lampiran 1** Source Code Pengambilan Data

```
#include <AWS_IOT.h>
#include <WiFi.h>
#include <ArduinoJson.h>
#include <Wire.h>
#include <BH1750.h>
#include "DHT.h"
#include "time.h"
#include "config.h"
#include <Adafruit_INA219.h>

AWS_IOT hornbill;

int status = WL_IDLE_STATUS;
int tick=0,msgCount=0,msgReceived = 0;
char payload[512];    //dekl
char rcvdPayload[512];

const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = 25200; //list utcnya +7 jd
7*60*60
const int   daylightOffset_sec = 3600;

float xSuhu = 0;
float xKelembaban = 0;
```

```
int xid = 1;

int measurePin = 35; //Connect dust sensor to esp32 pin
utk data
int ledPower = 2;   //Connect 3 led driver pins of dust
sensor led di pin 2 pd esp32
int samplingTime = 280;
int deltaTime = 40;     //utk led pd dust sensor
int sleepTime = 9680;
float voMeasured = 0;
float calcVoltage = 0;    //inisialisasi nilai awal (pd
sensor debu)
float dustDensity = 0;

#define DHTTYPE DHT11
#define DHTPin 32

BH1750 lightMeter;
DHT dht(DHTPin, DHTTYPE);

Adafruit_INA219 ina219(0x40);

void setup() {
    Serial.begin(9600);

    while (!Serial) {
       delay(1);
    }

    pinMode(ledPower,OUTPUT);
    while (status != WL_CONNECTED)
    {
        Serial.print("Attempting to connect to SSID: ");
        Serial.println(WIFI_SSID);
        // Connect to WPA/WPA2 network. Change this line
if using open or WEP network:
        status = WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
        // wait 5 seconds for connection:
        delay(5000);
    }

    Serial.println("Connected to wifi");

    configTime(gmtOffset_sec, daylightOffset_sec,
ntpServer);

    Wire.begin();

    lightMeter.begin();

    pinMode(DHTPin, INPUT);
    dht.begin();

    if(hornbill.connect(HOST_ADDRESS, CLIENT_ID,
                aws_root_ca_pem, certificate_pem_crt,
private_pem_key)== 0)
    {
        Serial.println("Connected to AWS");
        delay(1000);
    }
    else
    {
        Serial.println("AWS connection failed, Check the
HOST Address");
        while(1);
    }

   uint32_t currentFrequency;
    if (! ina219.begin()) {
      Serial.println("Failed to find INA219 chip");
      while (1) { delay(10); }
    }

    delay(1000);

}

void loop() {

  float shuntvoltage = 0;
  float busvoltage = 0;
  float current_mA = 0;
  float loadvoltage = 0;
  float power_mW = 0;

  shuntvoltage = ina219.getShuntVoltage_mV();
  busvoltage = ina219.getBusVoltage_V();
  current_mA = ina219.getCurrent_mA();
  power_mW = ina219.getPower_mW();
  loadvoltage = busvoltage + (shuntvoltage / 1000);

  Serial.print("Bus Voltage:   ");
Serial.print(busvoltage); Serial.println(" V");
  Serial.print("Shunt Voltage: ");
Serial.print(shuntvoltage); Serial.println(" mV");
  Serial.print("Load Voltage:  ");
Serial.print(loadvoltage); Serial.println(" V");
  Serial.print("Current:       ");
Serial.print(current_mA); Serial.println(" mA");
  Serial.print("Power:         ");
Serial.print(power_mW); Serial.println(" mW");
  Serial.println("");

  float lux = lightMeter.readLightLevel();
  Serial.print("Intensitas Cahaya: ");
  Serial.print(lux);
  Serial.println(" lx");

  xSuhu = dht.readTemperature();
  xKelembaban = dht.readHumidity();
  Serial.print("Suhu Udara: ");
  Serial.println(xSuhu);
  Serial.print("Kelembaban Udara: ");
  Serial.println(xKelembaban);

  digitalWrite(ledPower,LOW);
  delayMicroseconds(samplingTime);

  voMeasured = analogRead(measurePin);
```

```
delayMicroseconds(deltaTime);
digitalWrite(ledPower,HIGH);
delayMicroseconds(sleepTime);


calcVoltage = voMeasured * (3.3 / 4096.0);

dustDensity = 170 * calcVoltage - 0.0;

Serial.print ("density :");
Serial.println(dustDensity); // unit: ug/m3

delay(1000);

time_t rawtime;
struct tm *timeinfo;
char bufferTime[80];

time (&rawtime);
timeinfo = localtime ( &rawtime );
strftime(bufferTime, 80, "%F %X", timeinfo);

Serial.print ("timestamp :");
Serial.println(bufferTime);


  if(tick >= 1)
  {
      tick=0;
```

```
        const int capacity = JSON_OBJECT_SIZE(20);
        StaticJsonBuffer<capacity> jsonBuffer;
        JsonObject& obj = jsonBuffer.createObject();

        obj["id"] = xid;
        obj["debu"] = dustDensity;
        obj["cahaya"] = lux;
        obj["suhu"] = xSuhu;
        obj["kelembaban"] = xKelembaban;
        obj["arus"] = ina219.getCurrent_mA();
        obj["tegangan"] = ina219.getBusVoltage_V();
        obj["timestamp"] = bufferTime;

        obj.printTo(Serial);
        obj.printTo(payload);

        if(hornbill.publish(TOPIC_NAME,payload) == 0)
        {
            Serial.print("Publish Message:");
            Serial.println(payload);
        }
        else
        {
            Serial.println("Publish failed");
        }
    }
    vTaskDelay(1000 / portTICK_RATE_MS);
    tick++;
}
```

**Lampiran 2** file config.h

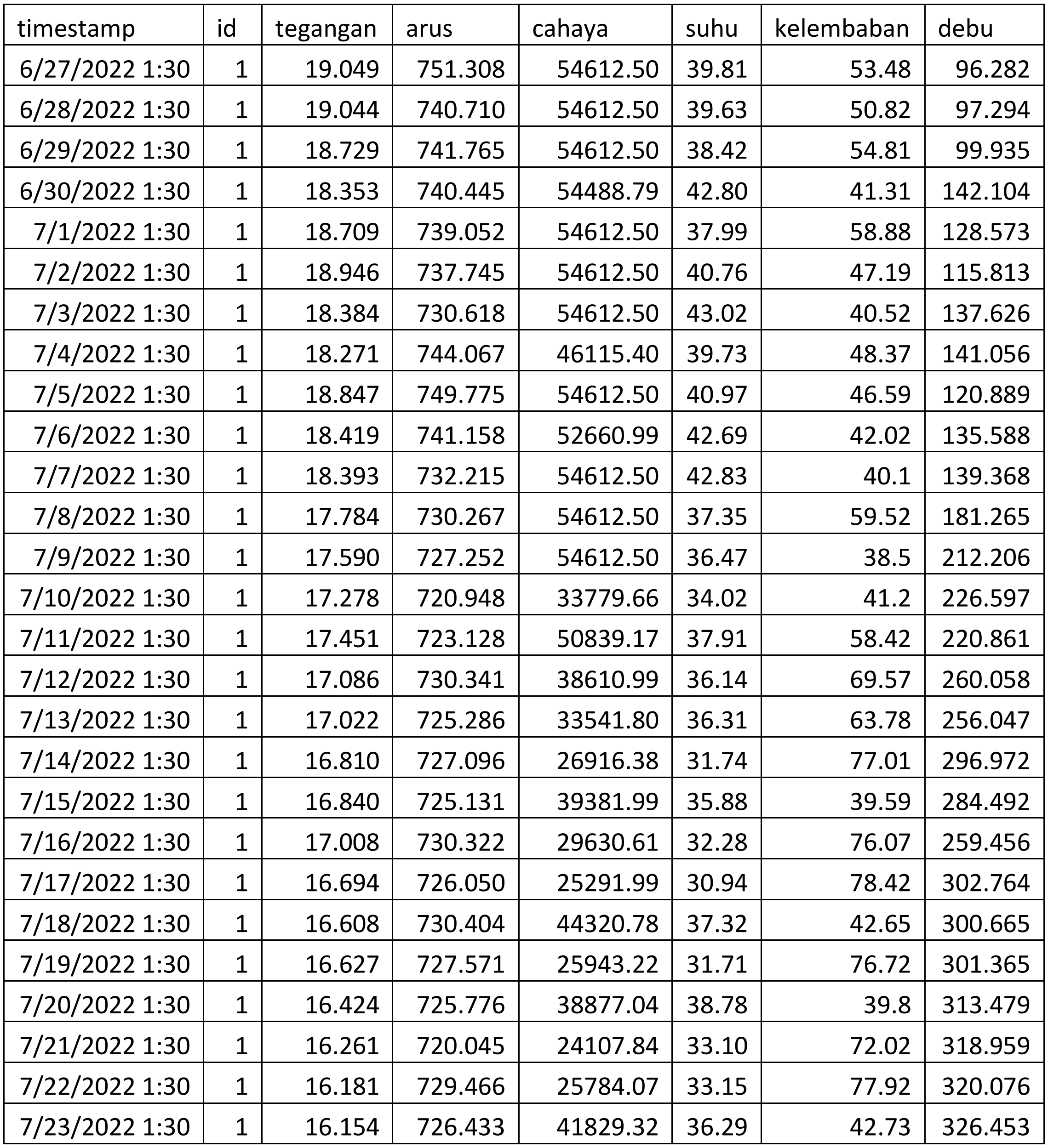
```
#ifndef _CONFIG_H_
#define _CONFIG_H_
// WiFi and MQTT configuration
static auto constexpr WIFI_SSID = "SSID WIFI";
static auto constexpr WIFI_PASSWORD = "PASSWORD WIFI";
static auto constexpr HOST_ADDRESS = "ALAMAT THING";
static auto constexpr CLIENT_ID = "ID";
static auto constexpr TOPIC_NAME = "NAMA TOPIK";
// "AWS root CA1 and C2 (RSA)", see
//
https://docs.aws.amazon.com/iot/latest/developerguide/man
aging-device-certs.html#server-authentication
static auto constexpr aws_root_ca_pem = "-----BEGIN
CERTIFICATE-----\n\
MIIDQTCCAimgAwIBAgITBmyfz5m/jAo54vB4ikPmljZbyjANBgkqhkiG9
w0BAQsF\
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDE
xBBbWF6\
b24gUm9vdCBDQSAxMB4XDTE1MDUyNjAwMDAwMFoXDTM4MDExNzAwMDAwM
FowOTEL\
MAkGA1UEBhMCVVMxDzANBgNVBAoTBkFtYXpvbjEZMBcGA1UEAxMQQW1he
m9uIFJv\
b3QgQ0EgMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALJ4g
HHKeNXj\
ca9HgFB0fW7Y14h29Jlo91ghYPl0hAEvrAIthtOgQ3pOsqTQNroBvo3bS
MgHFzZM\
9O6II8c+6zf1tRn4SWiw3te5djgdYZ6k/oI2peVKVuRF4fn9tBb6dNqcm
zU5L/qw\
IFAGbHrQgLKm+a/sRxmPUDgH3KKHOVj4utWp+UhnMJbulHheb4mjUcAwh
mahRWa6\
VOujw5H5SNz/0egwLX0tdHA114gk957EWW67c4cX8jJGKLhD+rcdqsq08
p8kDi1L\
93FcXmn/6pUCyziKrlA4b9v7LWIbxcceVOF34GfID5yHI9Y/QCB/IIDEg
Ew+OyQm\
jgSubJrIqg0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BA
f8EBAMC\
AYYwHQYDVR0OBBYEFIQYzIU07LwMlJQuCFmcx7IQTgoIMA0GCSqGSIb3D
QEBCwUA\
A4IBAQCY8jdaQZChGsV2USggNiMOruYou6r4lK5IpDB/G/wkjUu0yKGX9
rbxenDI\
U5PMCCjjmCXPI6T53iHTfIUJrU6adTrCC2qJeHZERxhlbI1Bjjt/msv0t
adQ1wUs\
N+gDS63pYaACbvXy8MWy7Vu33PqUXHeeE6V/Uq2V8viTO96LXFvKWlJbY
K8U90vv\
o/ufQJVtMVT8QtPHRh8jrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZF
o6IQ6XU\
5MsI+yMRQ+hDKXJioaldXgjUkK642M4UwtBV8ob2xJNDd2ZhwLnoQdeXe
GADbkpy\
rqXRfboQnoZsG4q5WTP468SQvvG5\
-----END CERTIFICATE-----\
-----BEGIN CERTIFICATE-----\
MIIFQTCCAymgAwIBAgITBmyf0pY1hp8KD+WGePhbJruKNzANBgkqhkiG9
w0BAQwF\
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDE
xBBbWF6\
b24gUm9vdCBDQSAyMB4XDTE1MDUyNjAwMDAwMFoXDTQwMDUyNjAwMDAwM
FowOTEL\
MAkGA1UEBhMCVVMxDzANBgNVBAoTBkFtYXpvbjEZMBcGA1UEAxMQQW1he
m9uIFJv\
b3QgQ0EgMjCCAiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBAK2Wn
y2cSkxK\
gXlRmeyKy2tgURO8TW0G/LAIjd0ZEGrHJgw12MBvIITplLGbhQPDW9tK6
Mj4kHbZ\
W0/jTOgGNk3Mmqw9DJArktQGGWCsN0R5hYGCrVo34A3MnaZMUnbqQ523B
NFQ9lXg\
1dKmSYXpN+nKfq5clU1Imj+uIFptiJXZNLhSGkOQsL9sBbm2eLfq0OQ6P
BJTYv9K\
8nu+NQWpEjTj82R0Yiw9AElaKP4yRLuH3WUnAnE72kr3H9rN9yFVkE8P7
K6C4Z9r\
2UXTu/Bfh+08LDmG2j/e7HJV63mjrdvdfLC6HM783k81ds8P+HgfajZRR
idhW+me\
z/CiVX18JYpvL7TFz4QuK/0NURBs+18bvBt+xa47mAExkv8LV/SasrlX6
avvDXbR\
8O70zoan4G7ptGmh32n2M8ZpLpcTnqWHsFcQgTfJU7O7f/aS0ZzQGPSSb
tqDT6Zj\
mUyl+17vIWR6IF9sZIUVyzfpYgwLKhbcAS4y2j5L9Z469hdAlO+ekQiG+
r5jqFoz\
7Mt0Q5X5bGlSNscpb/xVA1wf+5+9R+vnSUeVC06JIglJ4PVhHvG/Lopyb
oBZ/1c6\
+XUyo05f7O0oYtlNc/LMgRdg7c3r3NunysV+Ar3yVAhU/bQtCSwXVEqY0
VThUWcI\
0u1ufm8/0i2BWSlmy5A5lREedCf+3euvAgMBAAGjQjBAMA8GA1UdEwEB/
wQFMAMB\
Af8wDgYDVR0PAQH/BAQDAgGGMB0GA1UdDgQWBBSwDPBMMPQFWAJI/TPlU
q9LhONm\
UjANBgkqhkiG9w0BAQwFAAOCAgEAqqiAjw54o+Ci1M3m9Zh6O+oAA7CXD
pO8Wqj2\
LIxyh6mx/H9z/WNxeKWHWc8w4Q0QshNabYL1auaAn6AFC2jkR2vHat+2/
XcycuUY\
+gn0oJMsXdKMdYV2ZZAMA3m3MSNjrXiDCYZohMr/+c8mmpJ5581Lxedhp
xfL86kS\
k5Nrp+gvU5LEYFiwzAJRGFuFjWJZY7attN6a+yb3ACfAXVU3dJnJUH/jW
S5E4ywl\
7uxMMne0nxrpS10gxdr9HIcWxkPo1LsmmkVwXqkLN1PiRnsn/eBG8om3z
EK2yygm\
btmlyTrIQRNg91CMFa6ybRoVGld45pIq2WWQgj9sAq+uEjonljYE1x2ig
GOpm/Hl\
urR8FLBOybEfdF849lHqm/osohHUqS0nGkWxr7JOcQ3AWEbWaQbLU8uz/
mtBzUF+\
fUwPfHJ5elnNXkoOrJupmHN5fLT0zLm4BwyydFy4x2+IoZCn9Kr5v2c69
BoVYh63\
n749sSmvZ6ES8lgQGVMDMBu4Gon2nL2XA46jCfMdiyHxtN/kHNGfZQIG6
lzWE7OE\
76KlXIx3KadowGuuQNKotOrN8I1LOJwZmhsoVLiJkO/KdYE+HvJkJMcYr
07/R54H\
9jVlpNMKVv/1F2Rs76giJUmTtt8AF9pYfl3uxRuw0dFfIRDH+fO6AgonB
8Xx1sfT\
4PsJYGw=\
-----END CERTIFICATE-----\n";

// "The certificate for this thing"
auto constexpr certificate_pem_crt = "-----BEGIN
CERTIFICATE-----\n\
MIIDWjCCAkKgAwIBAgIVAP39P9tg+7sgScG+zhs/5CW61vkaMA0GCSqGS
Ib3DQEB\
CwUAME0xSzBJBgNVBAsMQkFtYXpvbiBXZWIgU2VydmljZXMgTz1BbWF6b
24uY29t\
IEluYy4gTD1TZWF0dGxlIFNUPVdhc2hpbmd0b24gQz1VUzAeFw0yMTA4M
TMxMzU4\
MzVaFw00OTEyMzEyMzU5NTlaMB4xHDAaBgNVBAMME0FXUyBJb1QgQ2Vyd
GlmaWNh\
dGUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCtauCGB+5Wg
mWe4SZ5\
VJGrZTROMtM0nCEvsvQ8Q1dDADx244+75pPB6SUlo/5jS2LM+AwPwqmiM
0+H/njx\
0uW5co+IvkmW3/J85/tFAp2qR/Q94e5Rd0OXdMSlqdezVDtOdsEVu2ViD
8D/Zir/\
1K6xbRjgmaxISH677HSvgfh2YBlMh65n4Ey0Pw6MhXyaFPANXNUzzeUD9
LJ0RmFi\
HAvkTOaS9mYmksrcgA1qWiGHJNOjSZKq1zLvFl0e2uxtRTnfvkX2sMmSk
Bm7Vxsn\
7g7tozAiwerfm7Q98wqo3rAXUSL+ujiLixP9NJze1NHamBsffsWbSGz0Q
xsw+wbj\
O3SZAgMBAAGjYDBeMB8GA1UdIwQYMBaAFDKcPx/InoWhs+QenmtfnN2jx
jmEMB0G\
A1UdDgQWBBQkpjYv2pZfehyyjihIebclam6zjTAMBgNVHRMBAf8EAjAAM
A4GA1Ud\
DwEB/wQEAwIHgDANBgkqhkiG9w0BAQsFAAOCAQEAFn/kUHBddD6Dmg091
Ycq0gKA\
hCwfrEbowAoNdXiTMPMhdYiEIK08Y29AmIWdd6FzBQFbQzHpPXdOq23D3
+tllk1N\
ZVagURng/zs3c4amlc8vkih3wEWJp6Kc1WahOooTIZVsDmbc89RwkXRi3
zkemWJ/\
c7S0trhg9JnjzNWICmapI/q88G5kNxB9iwlQPcS8w3qj3c69TtwiTEx3q
ZMCCeHm\
zg7jNjUnJoHPbzrlIXIYmHUfyCCJfqdgyG812qFvJHP43fbZ4Jhc18SgD
ioqh/G5\
gbdWNGD9FomKXqDDC0XL4NrGqvfIHhHYrx/r+HSHqSHndkJF2qzTpLoBs
yniYw==\
-----END CERTIFICATE-----\n";

// "The private key of the thing". The public key is not
needed here.
auto constexpr private_pem_key = "-----BEGIN RSA PRIVATE
KEY-----\n\
MIIEowIBAAKCAQEArWrghgfuVoJlnuEmeVSRq2U0TjLTNJwhL7L0PENXQ
wA8duOP\
u+aTweklJaP+Y0tizPgMD8KpojNPh/548dLluXKPiL5Jlt/yfOf7RQKdq
kf0PeHu\
UXdDl3TEpanXs1Q7TnbBFbtlYg/A/2Yq/9SusW0Y4JmsSEh+u+x0r4H4d
mAZTIeu\
Z+BMtD8OjIV8mhTwDVzVM83lA/SydEZhYhwL5EzmkvZmJpLK3IANalohh
yTTo0mS\
qtcy7xZdHtrsbUU5375F9rDJkpAZu1cbJ+4O7aMwIsHq35u0PfMKqN6wF
1Ei/ro4\
i4sT/TSc3tTR2pgbH37Fm0hs9EMbMPsG4zt0mQIDAQABAoIBAQCllcSXF
iGUhuPj\
ZdanyYFiJ7te45bZyC7D7drZY++BcUtKthS7dPonEKhAnas8QnIvhrV1o
wj4cIgD\
MY9oUFi7z1CQYCnCH0nZzGQXS90iB5yGLiGxtfBbQMq36Emm2MncLPZG+
u1eSe6d\
r39lUj4P+jzj0Lg5OMeSRsBQCQkmdR7fG8Q+bPTRMPIxuZWndDk//Ykdf
JXWUOuA\
qHtWnH5sUJVER97Z1HsgThemkxBtbuf7E/9mf1/HxQbYDj8uVNi8xMgOK
qY2kPah\
vOw3qorRwrX81ThFN0VAwvZMxX/AJfOQyCq2/1lv9gqr0UV6Cj6ibsnoW
CmPJ47t\
VAq2TPgBAoGBAOCnTpi8t4WXKXYfutYKSxns2ZGPA3UHQhqPbqYotvhgh
9WuXzyX\
5pJNDVj2rA/3Pd8z+sjNTbNrFrSPHPFgKbv3TxYFTzazKQuPN07+Uhklw
oNewI3v\
MpLcS0sEw+ItaJFDtTUMHAbf9gLyDEw7KAZdVPel9QdQvePItGbmedUBA
oGBAMWd\
Zw2mRygA2G4NWbSzKuQ+eMMmkcv6EXFnX35z3htta7zMujNr/gg5/blcx
ZErXY7o\
1Xs/VA17ziWqs3+oROqmvVRVCs+jjyyyPGKkWe0Idqq2vuSIPYbrQPq2J
H9vJe5E\
5wM7MCnITEImXi0fLd7HkNNmE5DwtUfh0Hg8+CeZAoGAGe92xW9UCJow8
TAsk7VQ\
gkezvngyP/p8iTBx58DfjsEmLWcaG5a1ycIXfmPL7XPvsqr/ktsQ01vTn
TjAcKVX\
bBnbCYHHaVs5q9n22qnFtJ2FXUE9HaW1wvqnvptoOd5gKhUVAHDLc4e7k
Q98l/hk\
YIe53XVNWnDZ20g6e360JgECgYAxQuCanXWlejh5jqhrDDLH/O67rxSvJ
WJNw7gh\
XJ7kdW1i/iH4iZYk1Nfz6jVxz76dzlrRQ2MfjVpBN4FRdE/hsrkFmcF9K
0eYm8HD\
0TZ9iRuTyMRjUbcMt/5VwSvt/HYXJGP2AT4rXDvuTYQyeVBE9bibihmci
vSVu6vt\
HT27GQKBgHS+dr5P1XetBm9cnB5DnP/JeiBesIZ5EwkmkEOZokW8qifY6
G+7G9vx\
TAzOUVBXYiFdLjZ1kiZiFWP3e+q0Td1uYuG3a29u9gqsI6Y2gkz6jT4mi
kBebNOd\
BfTZzCs/bzPm9YBAoGkbad6T0qr1fVF2OsIW2BYHvRjfB76YGxQ5\
-----END RSA PRIVATE KEY-----\n";

#endif
```

**Lampiran 3** kode ARIMA

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
df = pd.read_csv('datasetname.csv')
df
df['timestamp'] = pd.to_datetime(df['timestamp'])
df
df.set_index('timestamp',inplace=True)
df.head()
ls=df[‘variabel’]
ls.plot()
timeseries = df[‘variabel’]
timeseries.rolling(12).mean().plot(label=‘variabel mean')
timeseries.rolling(12).std().plot(label=‘variabel Std')
timeseries.plot()
plt.legend()
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(df[‘variabel’])
figure=plt.figure()
figure=decomposition.plot()
figure.set_size_inches(15,7)
from statsmodels.tsa.stattools import adfuller as adf
test_result=adf(df[‘variabel’])
print('ADF Test:')
labels = ['ADF Statistic','p-Value','No.of lags
used','Number of observations used']
for value,label in zip(test_result, labels):
    print(label+':'+str(value))
```

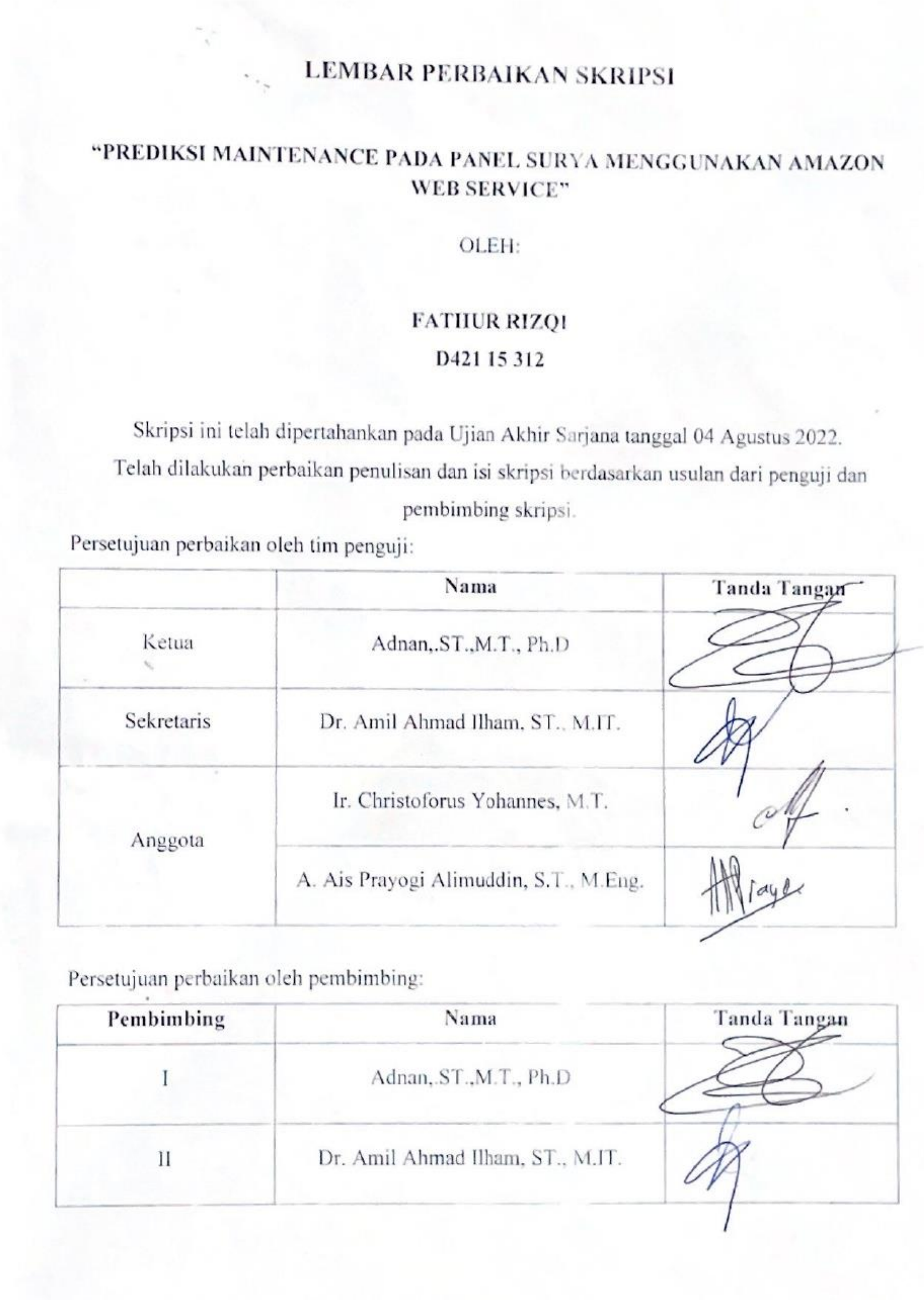
```
if test_result[1]<=0.05:
    print('Reject, null hypotesis an data is stationary')
else:
    print('fail')
df[‘variabel’1]=df[‘variabel’]-df[‘variabel’].shift(5)
df_check=(df[‘variabel’1].dropna())
df[‘variabel’1].plot()
from statsmodels.graphics.tsaplots import plot_acf as acf
fig_first = acf(df[‘variabel’].dropna())
model=sm.tsa.statespace.SARIMAX(df[‘variabel’],order=(0,1
,0),seasonal_order=(1,1,1,12))
ARIMAresult = model.fit()
print(ARIMAresult.summary())
df['forecast_data']=ARIMAresult.predict(start=20,end=30,d
ynamic=False)
df[['tegangan','forecast_data']].plot(figsize=(20,8))
prediction=ARIMAresult.predict(start=29,end=39,
model='additive')
df['tegangan'].plot(legend=True, label='train',
figsize=(10,6))
prediction.plot(legend=True, label='prediction', )
fcast =
ARIMAresult.predict(start=29,end=39,dynamic=False)
fcast.to_csv("namaprediksi.csv")
```

**Lampiran 4** Dataset



| timestamp | id | tegangan | arus | cahaya | suhu | kelembaban | debu |
|---|---|---|---|---|---|---|---|
| 6/27/2022 1:30 | 1 | 19.049 | 751.308 | 54612.50 | 39.81 | 53.48 | 96.282 |
| 6/28/2022 1:30 | 1 | 19.044 | 740.710 | 54612.50 | 39.63 | 50.82 | 97.294 |
| 6/29/2022 1:30 | 1 | 18.729 | 741.765 | 54612.50 | 38.42 | 54.81 | 99.935 |
| 6/30/2022 1:30 | 1 | 18.353 | 740.445 | 54488.79 | 42.80 | 41.31 | 142.104 |
| 7/1/2022 1:30 | 1 | 18.709 | 739.052 | 54612.50 | 37.99 | 58.88 | 128.573 |
| 7/2/2022 1:30 | 1 | 18.946 | 737.745 | 54612.50 | 40.76 | 47.19 | 115.813 |
| 7/3/2022 1:30 | 1 | 18.384 | 730.618 | 54612.50 | 43.02 | 40.52 | 137.626 |
| 7/4/2022 1:30 | 1 | 18.271 | 744.067 | 46115.40 | 39.73 | 48.37 | 141.056 |
| 7/5/2022 1:30 | 1 | 18.847 | 749.775 | 54612.50 | 40.97 | 46.59 | 120.889 |
| 7/6/2022 1:30 | 1 | 18.419 | 741.158 | 52660.99 | 42.69 | 42.02 | 135.588 |
| 7/7/2022 1:30 | 1 | 18.393 | 732.215 | 54612.50 | 42.83 | 40.1 | 139.368 |
| 7/8/2022 1:30 | 1 | 17.784 | 730.267 | 54612.50 | 37.35 | 59.52 | 181.265 |
| 7/9/2022 1:30 | 1 | 17.590 | 727.252 | 54612.50 | 36.47 | 38.5 | 212.206 |
| 7/10/2022 1:30 | 1 | 17.278 | 720.948 | 33779.66 | 34.02 | 41.2 | 226.597 |
| 7/11/2022 1:30 | 1 | 17.451 | 723.128 | 50839.17 | 37.91 | 58.42 | 220.861 |
| 7/12/2022 1:30 | 1 | 17.086 | 730.341 | 38610.99 | 36.14 | 69.57 | 260.058 |
| 7/13/2022 1:30 | 1 | 17.022 | 725.286 | 33541.80 | 36.31 | 63.78 | 256.047 |
| 7/14/2022 1:30 | 1 | 16.810 | 727.096 | 26916.38 | 31.74 | 77.01 | 296.972 |
| 7/15/2022 1:30 | 1 | 16.840 | 725.131 | 39381.99 | 35.88 | 39.59 | 284.492 |
| 7/16/2022 1:30 | 1 | 17.008 | 730.322 | 29630.61 | 32.28 | 76.07 | 259.456 |
| 7/17/2022 1:30 | 1 | 16.694 | 726.050 | 25291.99 | 30.94 | 78.42 | 302.764 |
| 7/18/2022 1:30 | 1 | 16.608 | 730.404 | 44320.78 | 37.32 | 42.65 | 300.665 |
| 7/19/2022 1:30 | 1 | 16.627 | 727.571 | 25943.22 | 31.71 | 76.72 | 301.365 |
| 7/20/2022 1:30 | 1 | 16.424 | 725.776 | 38877.04 | 38.78 | 39.8 | 313.479 |
| 7/21/2022 1:30 | 1 | 16.261 | 720.045 | 24107.84 | 33.10 | 72.02 | 318.959 |
| 7/22/2022 1:30 | 1 | 16.181 | 729.466 | 25784.07 | 33.15 | 77.92 | 320.076 |
| 7/23/2022 1:30 | 1 | 16.154 | 726.433 | 41829.32 | 36.29 | 42.73 | 326.453 |

## Lampiran 5



### LEMBAR PERBAIKAN SKRIPSI

**"PREDIKSI MAINTENANCE PADA PANEL SURYA MENGGUNAKAN AMAZON WEB SERVICE"**

OLEH:

**FATHUR RIZQI**
**D421 15 312**

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 04 Agustus 2022. Telah dilakukan perbaikan penulisan dan isi skripsi berdasarkan usulan dari penguji dan pembimbing skripsi.

Persetujuan perbaikan oleh tim penguji:

| | Nama | Tanda Tangan |
|---|---|---|
| Ketua | Adnan,.ST.,M.T., Ph.D | |
| Sekretaris | Dr. Amil Ahmad Ilham, ST., M.IT. | |
| Anggota | Ir. Christoforus Yohannes, M.T. | |
| | A. Ais Prayogi Alimuddin, S.T., M.Eng. | |

Persetujuan perbaikan oleh pembimbing:

| Pembimbing | Nama | Tanda Tangan |
|---|---|---|
| I | Adnan,.ST.,M.T., Ph.D | |
| II | Dr. Amil Ahmad Ilham, ST., M.IT. | |