DAFTAR PUSTAKA

Pine64.org. (2022, 28 April). Spesifikasi *Single-board computer Rock64*. Diakses pada 4 Mei 2022, dari https://wiki.pine64.org/wiki/ROCK64

Datasheetpdf.com. (2014). Spesifikasi sensor DHT11. Diakses pada 5 Februai 2021, dari https://datasheetspdf.com/pdf-file/785590/D-Robotics/DHT11/1

Innovatorsguru.com. (2021, 31 Januari). Spesifikasi senosr PZEM 004Tv30. Diakses pada 17 Oktober 2022, dari https://innovatorsguru.com/pzem-004t-v3/

Wemos.cc. (2021). Spesifikasi microcontroller Wemos D1 Mini. Diakses pada 9 Oktober 2021, dari https://www.wemos.cc/en/latest/d1/d1_mini.html

Eprintts2.undip.ac.id.(2022, 27 April). Perediksi Tinggi Permukaan Air Laut Menggunakan Model ARIMA Untuk Dearah Pesisir. Diakses pada 20 Juni 2022, dari https://eprints2.undip.ac.id/id/eprint/6022/3/Tesis%20Musbikhin%2030000416410040-BAB%20II.pdf

Repository.polimdo.ac.id((2017, 21 Maret). Prediksi Jumlah Mahasiswa Baru Menggunakan Model Arima. Diakses pada 20 Juni2022, dari http://repository.polimdo.ac.id/827/2/Karya%20Ilmiah%20-%20Prediksi%20Jumlah%20Mahasiswa%20Baru%20menggunakan%20Model%20ARIMA.pdf

Sosesetyo, Ivan & Bendatu, Liam Yenny.(2014). Penjadwalan *Predictive Maintenance* dan Biaya Perawatan Mesin Pellet di PT. Charoen Pokphand Indonesia-sepanjang. Sidoarjo. Jurnal Titra, Vol. 2, No.2.

Putri, Anggia Dasa & Suhendra, Dedy.(2016). Sistem Pakar Untuk Mendeteksi Kerusakan *Air Condiotioner*Menggunakan Metode *Fordward Chaining* Berbasis Web. Batam. Universitas Putera Batam.

Utomo, Anggoro Prasetyo & Wirawan, Nathan Adi. (2018).Perancangan Alat *Monitoring Air Conditioner* Menggunakan Mikrokontroler Wemos. Institut Teknologi Harapan Pangsa. Bandung.

Pratama, Andhika Yodi & Elfizon. (2021).Sistem Pengontrolan Air Conditioner Berbasis Arduino. Universitas Negeri Padang. Padang.

LAMPIRAN

Source Code Wemos D1 Mini

```
#include "ThingsBoard.h"
#include "DHT.h"
#include <SoftwareSerial.h>
#include <PZEM004Tv30.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#define DHTTYPE DHT11
#define DHTPIN D5
#define WIFI_AP             "Edosaniau"
#define WIFI_PASSWORD       "1sampai0"
PZEM004Tv30 pzem(D3, D4);
DHT dht(DHTPIN, DHTTYPE);
float h;
float t;
float V;
float I;
float I1;
float W;
```

```
float Hz;
float pf;
// See https://thingsboard.io/docs/getting-started-guides/helloworld/
// to understand how to obtain an access token
#define TOKEN               "lHO53hiskAoUw0X7eL1O"
#define THINGSBOARD_SERVER  "192.168.8.100"
// Baud rate for debug serial
#define SERIAL_DEBUG_BAUD   115200
// Initialize ThingsBoard client
WiFiClient espClient;
// Initialize ThingsBoard instance
ThingsBoard tb(espClient);
// the Wifi radio's status
int status = WL_IDLE_STATUS;
void setup() {
  // initialize serial for debugging
  Serial.begin(SERIAL_DEBUG_BAUD);
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  Serial.println("Inisialisasi PZEM004T!");
  Serial.println("Inisialisasi DHT11!");
  dht.begin();
  InitWiFi();
}
void loop() {
  delay(1000);
  if (WiFi.status() != WL_CONNECTED) {
    reconnect();
  }
```

```
  if (!tb.connected()) {
    // Connect to the ThingsBoard
    Serial.print("Connecting to: ");
    Serial.print(THINGSBOARD_SERVER);
    Serial.print(" with token ");
    Serial.println(TOKEN);
    if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
      Serial.println("Failed to connect");
      return;
    }
  }
  sensor();
  pzem004Tv30();
  Serial.println("Sending data...");
  Serial.println("-----------------------------------");
  // Uploads new telemetry to ThingsBoard using MQTT.
  // See https://thingsboard.io/docs/reference/mqtt-api/#telemetry-upload-api
  // for more details
}

void InitWiFi()
{
  Serial.println("Connecting to AP ...");
  // attempt to connect to WiFi network
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
  }
  Serial.println("Connected to AP");
}
void reconnect() {
  // Loop until we're reconnected
  status = WiFi.status();
  if ( status != WL_CONNECTED) {
    WiFi.begin(WIFI_AP, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("Connected to AP");
  }
}
void sensor() {
  // Wait a few seconds between measurements.
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  // Read temperature as Fahrenheit (isFahrenheit = true)
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t)) {
    Serial.println ("Failed to read from DHT sensor!");
    return;
  }
```

```
  Serial.print("Humidity    : ");
  Serial.println(h);
  Serial.print("Tenperature : ");
  Serial.println(t);
  tb.sendTelemetryInt("temperature", t);
  tb.sendTelemetryFloat("humidity", h);
  tb.loop();
}
void pzem004Tv30(){
   float V = pzem.voltage();
    if(V != NAN){
        Serial.print("Voltage: "); Serial.print(V); Serial.println("V");
    } else {
        Serial.println("Error reading voltage");
    }
    float I1 = pzem.current();
    float I = I1;
    if(I != NAN){
        Serial.print("Current: "); Serial.print(I, 3); Serial.println("A");
    } else {
        Serial.println("Error reading current");
    }
    float W = pzem.power();
    if(I != NAN){
        Serial.print("Power: "); Serial.print(W); Serial.println("W");
    } else {
        Serial.println("Error reading power");
    }
```

```
  float kWh = pzem.energy();
  if(I != NAN){
     Serial.print("Energy: "); Serial.print(kWh,3); Serial.println("kWh");
  } else {
     Serial.println("Error reading energy");
  }
  float Hz = pzem.frequency();
  if(I != NAN){
     Serial.print("Frequency: "); Serial.print(Hz, 1); Serial.println("Hz");
  } else {
     Serial.println("Error reading frequency");
  }
  float pf = pzem.pf();
  if(I != NAN){
     Serial.print("PF: "); Serial.println(pf);
  } else {
     Serial.println("Error reading power factor");
  }
tb.sendTelemetryFloat("Volt", V);
tb.sendTelemetryFloat("Ampere", I);
tb.sendTelemetryFloat("Watt", W);
tb.sendTelemetryFloat("WattHours", kWh);
tb.sendTelemetryFloat("Frequency", Hz);
tb.sendTelemetryFloat("Power Factor", pf);
//tb.loop();
delay(1000);
}
```

ARIMA Source Code

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt

df = pd.read_csv('database_dataset950.csv')
df['date'] = pd.to_datetime(df['date'])

df.set_index('date',inplace=True)
df.head()

ls=df['arus']
ls.plot()
ls1=df['temp']
ls1.plot()

timeseries = df['arus']
timeseries.rolling(12).mean().plot(label='arus mean')
timeseries.rolling(12).std().plot(label='arus Std')
timeseries.plot()
plt.legend()

timeseries = df['temp']
timeseries.rolling(12).mean().plot(label='temp mean')
timeseries.rolling(12).std().plot(label='temp Std')
timeseries.plot()
plt.legend()
```

```
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(df['arus'])
figure=plt.figure()
figure=decomposition.plot()
figure.set_size_inches(30,20)

from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(df['temp'])
figure=plt.figure()
figure=decomposition.plot()
figure.set_size_inches(30,20)

from statsmodels.tsa.stattools import adfuller as adf
test_result=adf(df['arus'])
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')

from statsmodels.tsa.stattools import adfuller as adf
test_result=adf(df['temp'])
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['arus1']=df['arus']-df['arus'].shift(1)
df_check=(df['arus1'].dropna())
df['temp1']=df['temp']-df['temp'].shift(1)
df_check=(df['temp1'].dropna())

df['arus1'].plot()
df['temp1'].plot()

from statsmodels.graphics.tsaplots import plot_acf as acf
fig_first = acf(df['arus'].dropna())

from statsmodels.graphics.tsaplots import plot_acf as acf
fig_first = acf(df['temp'].dropna())

model=sm.tsa.statespace.SARIMAX(df['arus'],order=(0,0,1),seasonal_order=(1,1,1,12))
ARIMAresult = model.fit()
print(ARIMAresult.summary())

model1=sm.tsa.statespace.SARIMAX(df['temp'],order=(0,0,1),seasonal_order=(1,1,1,12)
)
ARIMAresult1 = model1.fit()
print(ARIMAresult1.summary())
```

```
df['forecast_data']=ARIMAresult.predict(start=650,end=950)
df[['arus','forecast_data']].plot(figsize=(20,8))

df['forecast_data1']=ARIMAresult1.predict(start=650,end=950)
df[['temp','forecast_data1']].plot(figsize=(20,8))

prediction=ARIMAresult.predict(start=950,end=999, model='additive')
df['arus'].plot(legend=True, label='train', figsize=(10,6))
prediction.plot(legend=True, label='prediction', )

prediction=ARIMAresult1.predict(start=950,end=999, model='additive')
df['temp'].plot(legend=True, label='train', figsize=(10,6))
prediction.plot(legend=True, label='prediction', )

fcast = ARIMAresult.predict(start=950,end=999,dynamic=False)
fcast1 = ARIMAresult1.predict(start=950,end=999,dynamic=False)

fcast.to_csv("data_prediksi_Arus_50.csv")
fcast1.to_csv("data_prediksi_Temp_50.csv")

data_arus = pd.read_csv("data_prediksi_Arus_50.csv")
data_aktual = pd.read_csv("database_actual.csv")
data_temp = pd.read_csv("data_prediksi_Temp_50.csv")

data_a = data_arus['predicted_mean']
dataAktual = data_aktual['arus']
dataAktual1 = data_aktual['temp']
```

```
data_t = data_temp['predicted_mean']
import sklearn.metrics as metrics
mae = metrics.mean_absolute_error(dataAktual, data_a)
mae1 = metrics.mean_absolute_error(dataAktual1, data_t)
mse = metrics.mean_squared_error(dataAktual, data_a)
mse1 = metrics.mean_squared_error(dataAktual1, data_t)
rmse = np.sqrt(mse) # or mse**(0.5)
rmse1 = np.sqrt(mse1) # or mse**(0.5)
r2 = metrics.r2_score(dataAktual, data_a)
r21 = metrics.r2_score(dataAktual1, data_t)
def mape(dataAktual, data_a):
    dataAktual, data_a = np.array(dataAktual), np.array(data_a)
    return np.mean(np.abs((dataAktual - data_a) / dataAktual)) * 100
m = mape(dataAktual,data_a)
def mape1(dataAktual1, data_t):
    dataAktual1, data_t = np.array(dataAktual1), np.array(data_t)
    return np.mean(np.abs((dataAktual1 - data_t) / dataAktual1)) * 100
m1 = mape(dataAktual1,data_t)
print("Results of sklearn.metrics :")
print("MAE Arus :",mae)
print("MAE Temp :",mae1)
print("MSE Arus :", mse)
print("MSE Temp :", mse1)
print("RMSE Arus :", rmse)
print("RMSE Temp :", rmse1)
print("R-Squared Arus :", r2)
print("R-Squared Temp :", r21)
print("MAPE Arus :", m)
```

```
print("MAPE temp :", m1)
data_arus['date'] = pd.to_datetime(data_arus['date'])
data_arus.set_index('date',inplace=True)
data_aktual['date'] = pd.to_datetime(data_aktual['date'])
data_aktual.set_index('date',inplace=True)
data_temp['date'] = pd.to_datetime(data_temp['date'])
data_temp.set_index('date',inplace=True)
data_aktual['arus'].plot(legend=True, label='Data Aktual', figsize=(10,6))
data_arus['predicted_mean'].plot(legend=True, label='Data Prediksi', )
data_aktual['temp'].plot(legend=True, label='Data Aktual', figsize=(10,6))
data_temp['predicted_mean'].plot(legend=True, label='Data Prediksi', )
```

Data Aktual dan Data Predriksi

date	T_Aktual	T_Prediksi	A_Aktual	A_Prediksi
2022-04-24	25.9	25.976919205721998	1.55	1.5427528048315649
2022-04-25	25.9	26.018960709299304	1.55	1.5447829581879868
2022-04-26	26	25.976096783177937	1.55	1.5468222137041618
2022-04-27	26	25.940771584112216	1.56	1.5418377198769682
2022-04-28	26.1	25.94201111534256	1.56	1.5456286745553887
2022-04-29	26	25.98785463460188	1.56	1.5505385461671068
2022-04-30	26	25.983629884036358	1.55	1.5485721624822446
2022-05-01	25.9	26.02148952135474	1.55	1.5460537296565993
2022-05-02	25.9	26.0585439682571	1.56	1.5398865892517288
2022-05-03	26	26.015939652002206	1.56	1.539805685762417
2022-05-04	26	25.984401495102517	1.56	1.5416115391456888
2022-05-05	26.1	25.984531227378707	1.55	1.54434088314127
2022-05-06	26.2	25.975515754337152	1.55	1.544426209505234
2022-05-07	26.1	25.990395558408487	1.55	1.5448035549367896
2022-05-08	26	25.967671252927186	1.55	1.5468347595238892
2022-05-09	26	25.95514296424342	1.55	1.5418304646054868
2022-05-10	25.9	25.956819411891995	1.55	1.5456064526909516
2022-05-11	25.9	25.983573573005444	1.55	1.5505364200004932

2022-05-12	26	25.977859660376225	1.56	1.548577799548564
2022-05-13	26	25.99381573929642	1.56	1.5460693094318625
2022-05-14	26.1	26.00868281009278	1.56	1.5398870369944877
2022-05-15	26	25.986309623114618	1.56	1.5398064529101003
2022-05-16	26	25.97890325292622	1.56	1.54160517683
2022-05-17	25.9	25.97907871391021	1.56	1.5443237454446854
2022-05-18	25.9	25.975021058489872	1.56	1.544419602946009
2022-05-19	25.8	25.98032676537785	1.56	1.5448034736213474
2022-05-20	25.9	25.96470137818042	1.56	1.5468347099933109
2022-05-21	25.9	25.960208663096505	1.55	1.5418304932491147
2022-05-22	26	25.96203911731776	1.55	1.545606540422309
2022-05-23	26	25.982064562136546	1.56	1.550536428394545
2022-05-24	26	25.975825741913955	1.56	1.548577777293569
2022-05-25	25.9	25.98406114059028	1.56	1.5460692479233047
2022-05-26	25.9	25.991107489600218	1.56	1.5398870352268106
2022-05-27	26	25.975865476346364	1.56	1.5398064498814208
2022-05-28	25.9	25.976965203912684	1.56	1.5416052019482611
2022-05-29	25.9	25.977156783589557	1.56	1.544323813103877
2022-05-30	25.8	25.97484668552337	1.56	1.5444196290285375
2022-05-31	25.9	25.976777664815373	1.56	1.5448034739423786
2022-06-01	25.9	25.963654541275325	1.55	1.5468347101888562
2022-06-02	26	25.961994246988645	1.55	1.5418304931360305
2022-06-03	26	25.963878986248048	1.55	1.5456065400759478
2022-06-04	26.1	25.981532658133027	1.56	1.5505364283614056
2022-06-05	26	25.97510881575046	1.56	1.548577777381431
2022-06-06	26	25.98062278887828	1.56	1.5460692481661389
2022-06-07	25.9	25.9849124491762	1.56	1.5398870352337892
2022-06-08	25.9	25.97218406914522	1.57	1.539806449893378
2022-06-09	26	25.976282070311356	1.57	1.5416052018490949
2022-06-10	25.9	25.976479331588973	1.56	1.5443238128367602
2022-06-11	26	25.974785221632754	1.56	1.5444196289255643
2022-06-12	26.1	25.975526659380634	1.55	1.5448034739411112

LEMBAR PERBAIKAN SKRIPSI

"EVALUASI KINERJA SISTEM PREDIKSI MAINTENANCE AC MENGGUNAKAN SINGLE-BOARD ROCK64"

OLEH:

SAID SYAMIL AMAS

D421 15 002

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 08 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.	
	Anugrayani Bustamin, S.T., M.T.	

Persetujuan perbaikan oleh pembimbing:

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