PROTOTIPE ALAT PENDETEKSI PENCURIAN LISTRIK BERBASIS LORA DENGAN METODE *SUM OF ABSOLUTE DIFFERENCE* (SAD)

PROYEK AKHIR

Disusun oleh:

ZAHRA ARI MAULIDA
NIT. 301201048

PROGRAM STUDI DIPLOMA 3 TEKNIK LISTRIK BANDARA
POLITEKNIK PENERBANGAN SURABAYA
2024

PROTOTIPE ALAT PENDETEKSI PENCURIAN LISTRIK BERBASIS LORA DENGAN METODE *SUM OF ABSOLUTE DIFFERENCE* (SAD)

PROYEK AKHIR

Diajukan sebagai Syarat untuk Mendapatkan Gelar Ahli Madya (A.Md)
pada Program Studi Diploma 3 Teknik Listrik Bandar Udara

Disusun oleh:

ZAHRA ARI MAULIDA
NIT. 30121048

PROGRAM STUDI DIPLOMA 3 TEKNIK LISTRIK BANDARA
POLITEKNIK PENERBANGAN SURABAYA
2024

HALAMAN PERSETUJUAN

PROTOTIPE ALAT PENDETEKSI PENCURIAN LISTRIK BERBASIS LORA DENGAN METODE *SUM OF ABSOLUTE DIFFERENCE* (SAD)

Oleh:

ZAHRA ARLMAULIDA
NIT. 30121048

Disetujui untuk diujikan pada :
Surabaya, 07 Agustus 2024

Pembimbing I : **RIFDIAN I.S, S.T., M.M., M.T.** : .....
NIP. 19810629 200912 1 002

Pembimbing II : **Dr. LAILA ROCHMAWATI, S.S., M.Pd.** : .....
NIP. 19810723 200502 2 001

HALAMAN PENGESAHAN

PROTOTIPE ALAT PENDETEKSI PENCURIAN LISTRIK BERBASIS LORA DENGAN METODE *SUM OF ABSOLUTE DIFFERENCE* (SAD)

Oleh:

ZAHRA ARI MAULIDA
NIT. 30121048

Diajukan sebagai Syarat Menempuh Mata Kuliah Tugas Akhir pada
Program Studi Diploma 3 Teknik Listrik Bandar Udara
Politeknik Penerbangan Surabaya
Surabaya, 07 Agustus 2024

Panitia Penguji :

Ketua	:	**Dr. SLAMET HARIYADI, S.T., M.M.** NIP. 19630408 198902 1 001
Sekretaris	:	**Dr. LAILA ROCHMAWATI, S.S., M.Pd.** NIP 19810723 200502 2 001
Anggota	:	**RIFDIAN I.S, S.T., M.M., M.T.** NIP 19810629 200912 1 002

Ketua Program Studi
D3 Teknik Listrik Bandar Udara

Dr. GUNAWAN SAKTI, S.T., M.T.
NIP. 19881001 200912 1 003

PERNYATAAN KEASLIAN DAN HAK CIPTA

Saya yang bertanda tangan di bawah ini:

Nama	:	Zahra Ari Maulida
NIT	:	30121048
Program Studi	:	Diploma 3 Teknik Listrik Bandara Angkatan XVI
Judul Proyek Akhir	:	Prototipe Alat Pendeteksi Pencurian Listrik Berbasis Lora dengan Metode *Sum of Absolute Difference* (SAD)

Dengan ini menyatakan bahwa:

1. Proyek Akhir ini merupakan karya asli dan belum pernah diajukan untuk mendapatkan gelar akademik, baik di Politeknik Penerbangan Surabaya maupun Perguruan Tinggi lain, serta dipublikasikan, kecuali secara tertulis dengan jelas dicantumkan dalam daftar pustaka.
2. Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan Hak Bebas Royalti Non Ekslusif *(Non-Exclusive Royalty-Free Right)* kepada Politeknik Penerbangan Surabaya beserta perangkat yang ada (jika diperlukan). Dengan hak ini, Politeknik Penerbangan Surabaya berhak menyimpan, mengalihmediakan/formatkan, mengelola dalam bentuk pangkalan data *(database)*, merawat dan mempublikasikan Proyek Akhir dengan tetap mencantumkan nama saya sebagai pencipta dan sebagai pemilik Hak Cipta.

Pernyataan ini saya buat dengan sebenarnya. Apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh, serta sanksi lainnya sesuai dengan norma yang berlaku di Politeknik Penerbangan Surabaya.

Surabaya, 07 Agustus 2023

Yang membuat pernyataan

ZAHRA ARI MAULIDA
NIT. 30121048

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas rahmat, barokah, dan ridho-Nya sehingga penulis dapat menyelesaikan penyusunan Laporan Tugas Akhir ini. Pada kesempatan ini penulis mengucapkan terima kasih dan rasa hormat atas segala bantuan yang telah diberikan kepada penulis sehingga dapat menyelesaikan Laporan Tugas Akhir ini, yaitu kepada :

1. Tuhan Yang Maha Esa.
2. Orang tua dan keluarga yang senantiasa memberikan doa, kasih sayang, serta dukungan penuh baik berupa moril maupun materi.
3. Bapak Ahmad Bahrawi, S.E., M.T., selaku Direktur Politeknik Penerbangan Surabaya.
4. Bapak Dr. Gunawan Sakti, S.T., M.T., selaku Ketua Program Studi D3 Teknik Listrik Bandara
5. Bapak Rifdian Indrianto Sudjoko, S., S.T., M.M., selaku pembimbing I yang senantiasa membimbing dalam penyusunan Proyek Akhir.
6. Ibu Dr. Laila Rochmawati, S.S., M.Pd selaku pembimbing II yang senantiasa membimbing dalam penyusunan Proyek Akhir.
7. Seluruh dosen dan sivitas akademika Prodi D3 Teknik Listrik Bandara Politeknik Penerbangan Surabaya.
8. Teman-teman sekelas atas kebersamaan dan kerjasamanya.
9. Teman-teman seangkatan, senior alumni TLB XIV dan XV, serta adik-adik TLB XVII yang telah membantu dan memberikan dukungan semangat.

Tentunya karya tulis ini masih jauh dari kata sempurna. Maka dari itu saran, kritik, dan masukan yang membangun penting bagi penulis demi karya yang lebih baik di masa mendatang. Atas segala kesalahan dan kata-kata yang kurang berkenan, penulis memohon maaf.

Surabaya, 07 Agustus 2023
Yang membuat pernyataan

ZAHRA ARI MAULIDA

HALAMAN PERSEMBAHAN

MOTTO

“JANGAN MUDAH KAGUM, JANGAN MUDAH MENYESAL, JANGAN MUDAH DIKEJUTKAN, JANGAN MANJA”

-Semar Badranaya-

PERSEMBAHAN:

Allah SWT yang telah melimpahkan Rahmat dan Hidayah-Nya

Ibu Sri Rahyuni dan Suharni Orang Tua terhebat dan terkuat yang selalu memberikan doa dan semangat untuk kesuksesan putrinya.

Pihak-pihak yang telah membantu dan mendukung dalam

penyusunan Proyek Akhir ini

Teman-teman seperjuangan yang penuh dengan cerita Terimakasih semua

DAFTAR ISI

Halaman

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

Halaman

ABSTRAK

PROTOTIPE ALAT PENDETEKSI PENCURIAN LISTRIK BERBASIS LORA DENGAN METODE *SUM OF ABSOLUTE DIFFERENCE* (SAD)

Oleh :

ZAHRA ARI MAULIDA
30121048

Pencurian listrik merupakan permasalahan yang umum terjadi dibanyak negara, termasuk Indonesia. Pencurian listrik bisa terjadi diberbagai tempat, seperti di rumah, gedung perkantoran, fasilitas industri, dan lain sebagainya. Pencurian listrik dapat berdampak negatif bagi pemilik instalasi listrik dan mengakibatkan kerugian finansial yang signifikan.

Terdapat sebuah alat untuk menghindari bahkan mengurangi pencurian listrik dengan menggunakan sistem monitoring yang akan tersampaikan melalui web. Alat ini akan dipasangkan dimeteran pelanggan untuk memonitor jumlah pemakaian Listrik melalui *web based*. Data yang dimonitor berupa grafik yang menunjukkan ada atau tidaknya indikasi pencurian Listrik dengan membandingkan data konsumsi pada pelanggan dengan data jumlah listrik yang disalurkan oleh gardu penyalur.

Pada peneliti ini menggunakan sensor arus tegangan PZEM 004-T dan mikrokontroller Arduino uno dengan modul LoRa sebagai komunikasi jarak jauh. Juga dilengkapi modul LoRa dan NodeMCU Esp 8266 sebagai modul wifi yang datanya akan ditampilkan melalui *web*.

Kata kunci : Pencurian Listrik, Mikrokontroler ESP8266, PZEM 004-T, LoRa

ABSTRACT

PROTOTYPE OF LORA-BASED ELECTRICAL THEFT DETECTION DEVICE WITH SUM OF ABSOLUTE DIFFERENCE METHOD (SAD)

By :

ZAHRA ARI MAULIDA
30121048

Electricity theft is a common problem in many countries, including Indonesia. Electricity theft can occur in various places, such as in homes, office buildings, industrial facilities, and so on. Theft of electricity can negatively impact owners of electrical installations and result in significant financial losses.

There is a tool to avoid and even reduce electricity theft by using a monitoring system that will be delivered via the web. This tool will be installed on the customer's meter to monitor the amount of electricity consumption through web based. The monitored data is in the form of graphs that show whether or not there are indications of electricity theft by comparing consumption data on customers with data on the amount of electricity distributed by distribution substations.

This researcher uses a PZEM 004-T voltage current sensor and an Arduino uno microcontroller with a LoRa module as long-distance communication. Also equipped with LoRa module and Microcontroller Esp 8266 as wifi module whose data will be displayed via the web.

***Keywords**: Electricity Theft, Microcontroler ESP8266, PZEM 004-T, LoRa*

DAFTAR PUSTAKA

Abdallah, Nandi. (2021). Pengaruh Media Sosial Dan Gerakan Massa Terhadap Hakim. Surabaya : Cipta Media Nusantara.

Abdullah, R. (2018). *7 in 1 Pemrograman Web Untuk Pemula.*

Abdurrahman, S. (2017). *Modul Elektroniks dan Mekatronika.* Jakarta: Direktorat Pembinaan SMK.

Ahmad Adhitya Nurhadi, D. D. (2021). *Implementasi Modul Komunikasi LoRa RFM95 Pada Sistem Pemantauan Listrik 3 Fasa Berbasis IoT*.

Arihutomo, M., Rivai, M., & Suwito, S. (2012). Sistem monitoring arus listrik jala-jala menggunakan Power Line Carrier. *Jurnal Teknik ITS, 1*(1), A150-A153.

Blocher, Richard. 2003. Dasar Elektronika. Yogyakarta : Andi Yogyakarta

i, B. A. B., & Pustaka, T. (2012). (Sumber : Frans Romario dan Stevano Augusta, 2012). 6–20.

Budiharto, W. (2018). *Panduan Pemrograman Mikrokontroler AVR ATMega16.* Jakarta: Elex Media Komputindo.

Cakrawala96. (2023, Februari 20). *Lampu Halogen: Pengertian, Prinsip Kerja, Kelebihan, dan Kekurangan*. From https://www.gesainstech.com/

components. (2017, September 30). *Buzzer*. From Active Passive Buzzer: https://components101.com/misc/buzzer-pinout-working-datasheet

Components101. (2021). *Relay module*. From https://components101.com/

D Sharon, d. (1982). *Principles of Analysis Chemistry*.

Datasheet. (n.d.). From alldatasheet.com: https://pdf1.alldatasheet.com/datasheet-pdf/view/16101/PHILIPS/BC547.html

Efendi, Y. (2018). *INTERNET OF THINGS (IOT) SISTEM PENGENDALIAN LAMPU MENGGUNAKAN RASPBERRY PI BERBASIS MOBILE*.

electroncomponents. (2024). *Electron Components*. From electroncomponents.com: https://www.electroncomponents.com/1000uf-25v-Capacitor

Fuada, S., & Pd, S. (2013). Sistem Keamanan Cerdas untuk Menangani Pencurian Listrik pada Smart City. In *E-Indonesia Initiatives forum*.

Gunadie27. (2023, May 9). *Datasheet dan Kode Warna Resistor 1k Ohm*. From solderpanas.com: https://solderpanas.com/17243/datasheet-dan-kode-warna-resistor-1k-ohm.html

Guru, I. (2021, Februari 8). From https://manuals.plus/innovatorsguru/ac-communication-module-pzem-004t-v3-0-manual

Habibi FN, S. S. (2017). *Alat Monitoring Pemakaian Energi Listrik Berbasis Android Menggunakan Modul PZEM-004T*.

Ham, R. (n.d.). *Ketahui Jenis Power Supply Serta Keunggulan dan Kelemahan*. From https://www.griyasis.com/

Indoboot. (2023, Mei 22). *buzzer*. From https://blog.indobot.co.id/

Kurnia, A. (2008). *LANGKAH LANGKAH PENELITIAN*. From https://skripsimahasiswa.blogspot.co.id/2008/10/langkah-langkah-penelitian.html

Kusumah, H. &. (2019). *PENERAPAN TRAINER INTERFACING MIKROKONTROLER DAN INTERNET OF THINGS BERBASIS ESP32 PADA MATA KULIAH INTERFACING*.

Leist, R. (2022, April). *21 of the Best Examples of Mobile Website Design*. From https://blog.hubspot.com/

M Reza Hidayat, C. B. (2018). Perancangan Sistem Keamanan Rumah Berbasis Iot Dengan Nodemcu Esp8266 Menggunakan Sensor Pir Hc-Sr501 Dan Sensor Smoke Detector. *Universitas Jenderal Achmad Yani*, 7 (2).

Malvino, A. P. (2006). *Prinsip Elektronika.* Jakarta: Teknika Salemba.

Mannan, M. (2015). IMPLEMENTASI PENERAPAN INTERNET of THINGS (IoT) PADA MONITORING INFUS MENGGUNAKAN ESP 8266 DAN WEB UNTUK BERBAGI DATA. *Journal Ibrahimy University*.

Mohammad, N., Barua, A., & Arafat, M. A. (2013, February). A smart prepaid energy metering system to control electricity theft. In *2013 international conference on power, energy and control (ICPEC)* (pp. 562-565). IEEE.

Mucheli, N. K. (2019). Smart Power Theft Detection System. *researchgate*.

Muflihah, N. (2013). *Metodologi Penelitian*. From http://nurmuflihah.blogspot.co.id/2013/04/bab-iii-metodologi-penelitian.html

Muslihudin, M., Renvillia, W., Taufiq, Andoyo, A., & Susanto, F. (2018). IMPLEMENTASI APLIKASI RUMAH PINTAR BERBASIS ANDROID DENGAN . *Jurnal Keteknikan dan Sains (JUTEKS)*, 1 (1).

Ningrum, R. F. (2019). Prototype Alat Pembatas Dan Pemutus Arus Listrik Pascabayar Pada Rumah Tangga Berbasis Smartphone. *Jurnal Teknik*, *7*(2).

Nurhadi, D. Darlis, and M. Murti, "Implementasi Modul Komunikasi LoRa RFM95W pada Sistem Pemantuan Listrik 3 Fasa Berbasis IoT," Ultima Computing : Jurnal Sistem

Komputer, vol. 13, no. 1, pp. 17-21, 2021

Panchal, C. S. (2014). Depth Estimation Analysis Using Sum of Absolute Difference Algorithm. *International Journal of Advanced Research in Electrical, Electronics and Instrumentaion Engineering*.

Penfold. (2002). *Dasar dasar elektronika untuk pemula*, 13.

Prastyo, E. A. (2023, April 6). *Pengertian dan Penjelasan tentang Kapasitor Elektrolit*. From https://www.arduino.biz.id/

Riandi, H. (2011, April 25). *Mencuri Listrik PLN* . From https://r-dy-techno.blogspot.com/

Rifky, I. (2021, November 2016). *MIKROKONTROLER ESP32*. From https://raharja.ac.id/

Romario Panjaitan, F. d. (2012). *Pengatur Intensitas Cahaya menggunakan Transistor*.

Sarwono, Jonathan. 2015. Bikin Website Itu Mudah. MediaKita. Jakarta.

Senen, A. (2019). Perancangan prototipe alat ukur arus listrik Ac dan Dc berbasis mikrokontroler arduino dengan sensor arus Acs-712 30 ampere. *Sutet*, *8*(1), 28-33.

Setiawan, V. L. (2022, Maret 24). *Mengapa Transistor Penting Dalam Elektronika?* From https://www.anakteknik.co.id/

Sidharta. (2023). *PENALARAN HUKUM DALAM ARREST PENCURIAN LISTRIK*. From business-law.binus.ac.id: https://business-law.binus.ac.id/2023/09/03/penalaran-hukum-dalam-arrest-pencurian-listrik-putusan-hr-23-mei-1921/.

Sugiyono. (2015). Metode Penelitian Pendidikan Pendekatan Kuantitatif, Kualitatif, dan R & D. Bandung : Alfabeta.

Sulistyo, S. (2016). RUMUSAN METODE DETEKSI PENCURIAN LISTRIK MEMANFAATKAN PERANGKAT WSN. *Jurnal MIPA* .

Sumodiharjo, P. (2019). Aksi Pencurian Listrik Marak di Pacitan, Ini Upaya PLN. *detikNews*.

Surabaya, Politeknik Elektronika Negeri. (n.d.). *Modul Praktikum Komunikasi LoRa Node dan Gateway*. Surabaya: zenhadi.

Walukow, S., & Doringin, F. (2020, September). Sistem Pendeteksi dan Penginformasi Terjadinya Pencurian Listrik Berbasis SMS Gateway dan Arduino. In Prosiding Industrial Research Workshop and National Seminar (Vol. 11, No. 1, pp. 188-192).

Yulistiani, T. (2023). ALAT PEMBATAS ARUS ADJUSTABLE LIMITER BERBASIS MIKROKONTROLER. *Sarjana thesis, Universitas Siliwangi.*, 2-4.

LAMPIRAN

LAMPIRAN A *Standard Operational Procedure* (SOP)

"PROTOTIPE ALAT PENDETEKSI PENCURIAN LISTRIK BERBASIS LORA DENGAN METODE *SUM OF ABSOLUTE DIFFERENCE* (SAD)"

Oleh:
ZAHRA ARI MAULIDA
NIT. 30121048

Standard Operational Procedure (SOP) dalam pengoperasian alat adalah panduan yang merinci mengenai langkah-langkah yang harus diikuti untuk menggunakan alat dengan benar dan aman. SOP ini mencakup instruksi rinci tentang cara menghidupkan dan mematikan alat. Dimana dengan menggunakan *Standar Operational Procedure* (SOP) yang telah dibuat bertujuan untuk menjaga alat Proyek Akhir terhindar dari kerusakan atau salah prosedur, setiap pengguna alat akan mengikuti prosedur yang sama, sehingga dapat menjamin konsistensi dan akurasi hasil. Para pengguna harus memahami dan mengikuti SOP ini dengan cermat untuk memastikan operasi yang aman, efisien, dan tepat sesuai dengan tujuan penggunaan alat tersebut.

Berikut merupakan *Standar Operational Procedure* (SOP) untuk menghidupkan dan mematikan alat Proyek Akhir dengan judul "Prototipe Alat Pendeteksi Pencurian Listrik Berbasis Lora dengan Metode *Sum of Absolute Difference* (SAD)" sebagai berikut:

1. Mengoperasikan Alat Sesuai dengan SOP
    a. Sambungkan kabel power untuk menghidupkan power supply 5V dan sambungkan juga kabel untuk menghidupkan ESP8266 dan LCD.
    b. Pastikan semua komponen dapat menyala sesuai fungsinya masing-masing.
    c. Kemudian nyalakan *hotspot* sehingga nantinya dapat termonitoring

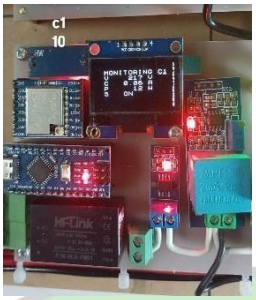
oleh LCD untuk menampilkan data sensor PZEM 004T, AC712, dan juga ZMPT101B untuk memantau kondisi arus, tegangan, dan daya yang akan termonitoring pada aplikasi android.

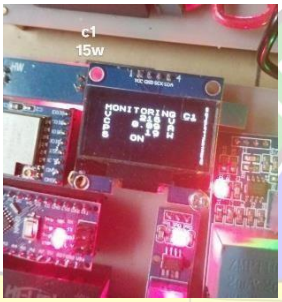
d. Jika alat sudah siap untuk digunakan, maka alat akan bekerja sesuai apa yang telah diperintahkan, kemudian hasil pengukuran dapat secara langsung termonitoring pada LCD dan aplikasi *Ethief Detector* secara *real-time*.

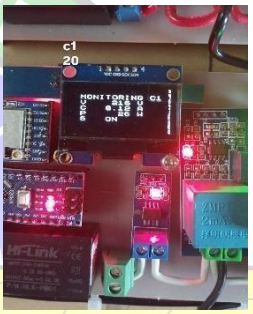
2. Mematikan Alat Sesuai SOP
   a. Matikan *hotspot* untuk memutus koneksi antara alat dan program yang telah ter*setting*, serta memutuskan pemantauan pada aplikasi.
   b. Kemudian lepas kabel power dan kabel *micro usb* untuk memutuskan aliran listrik yang menuju komponen.
   c. Setelah semua langkah telah dilakukan, maka alat sudah aman dan alat akan mati tidak bekerj

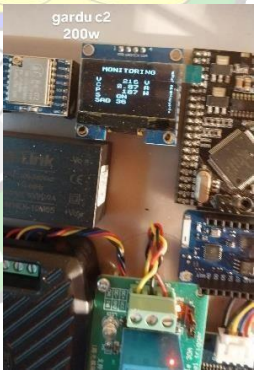
LAMPIRAN B Dokumentasi Alat

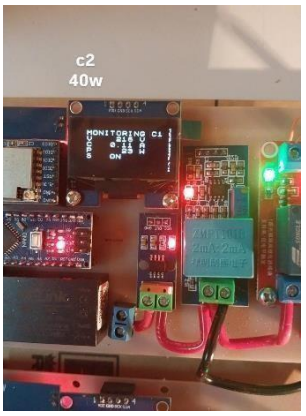
Gambar pengujian alat

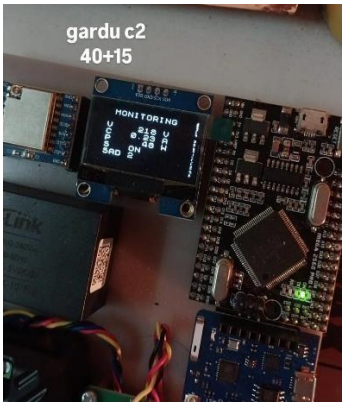


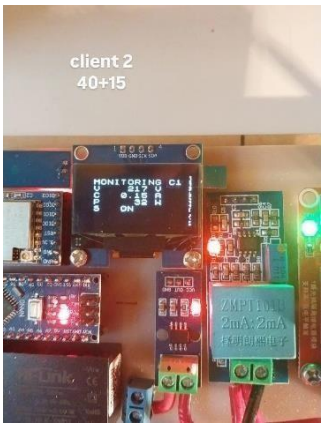


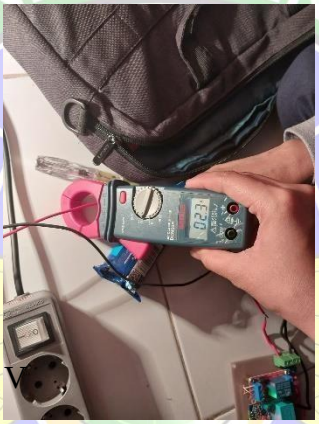


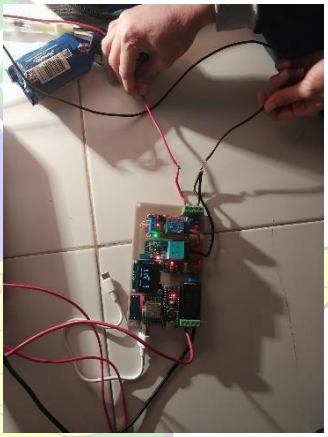



c2
40w


gardu c2
40+15

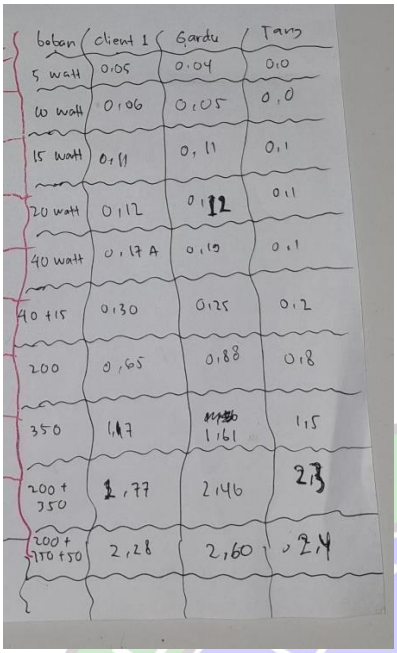

client 2
40+15



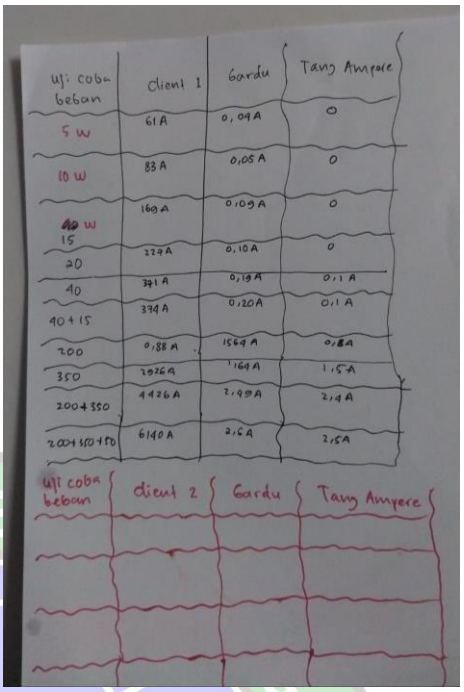



TRUE RMS
HOLD
OFF
AC CLAMP METER
DCM60R
sanwa


TRUE RMS
HOLD
OFF
AC CLAMP METER
DCM60R
sanwa



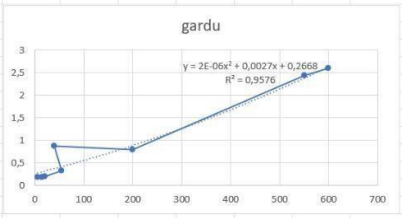
beban	client 1	Gardu	Tang
5 watt	0,05	0,04	0,0
10 watt	0,06	0,05	0,0
15 watt	0,11	0,11	0,1
20 watt	0,112	0,12	0,1
40 watt	0,17 A	0,19	0,1
40 + 15	0,130	0,125	0,2
200	0,165	0,188	0,18
350	1,17	1,161	1,15
200 + 350	1,77	2,146	2,3
200 + 350 + 50	2,28	2,60	2,4

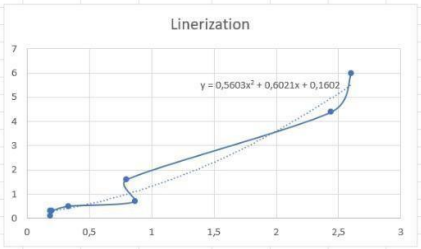


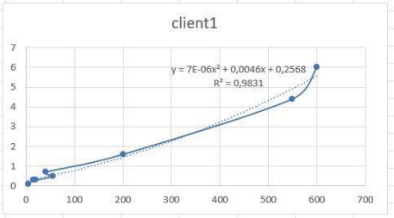
uji coba beban	client 1	Gardu	Tang Ampere
5 w	61 A	0,04 A	0
10 w	83 A	0,05 A	0
15	169 A	0,09 A	0
20	227 A	0,10 A	0
40	371 A	0,19 A	0,1 A
40 + 15	394 A	0,20 A	0,1 A
200	0,88 A	1564 A	0,8A
350	2026 A	1,69 A	1,5 A
200 + 350	4426 A	2,19 A	2,1 A
200 + 350 + 50	6140 A	2,6 A	2,5 A

uji coba beban	client 2	Gardu	Tang Ampere

Beban	gardu	client1	tang ampere
5	0,19	0,1	0.1
15	0,19	0,3	0.1
20	0,2	0,3	0.1
55	0,33	0,5	0.3
40	0,87	0,7	0.8
200	0,8	1,6	0.8
550	2,44	4,4	2.3
600	2,6	6	4.1







LAMPIRAN C Program Koding Alat

```
#include <Arduino.h>

/*==================LoRa=============================================
==========================================================================
====================================
 *
 */
#include <SPI.h>          // include libraries
#include <LoRa.h>

#define LORA_MOSI_Pin 11
#define LORA_MISO_Pin 12
#define LORA_SCK_Pin  13

#define CS_Pin 10         // LoRa radio chip select
#define RESET_Pin 9       // LoRa radio reset
#define IRQ_Pin 3         // change for your board; must be a
hardware interrupt pin

String outgoing;              // outgoing message
String LoRa_incoming;             // incoming message

uint8_t msgCount = 0;             // count of outgoing messages
const uint8_t client1Address = 0xAA;     // address of this device
const uint8_t client2Address = 0xAB;     // address of another
device
const uint8_t masterAddress = 0xFF;      // destination to send to
int incomingAddress = 0;
long lastSendTime = 0;        // last send time
const int interval = 2500;          // interval between sends
bool LoRa_updateState = false;

void LoRa_setup(void);
void LoRa_txMode(void);
void LoRa_rxMode(void);
void LoRa_sendMessage(String message);
void LoRa_onReceive(int packetSize);
void LoRa_onTxDone(void);
void LoRa_parsingData(void);
/*==================Oled
Display=================================================================
```

```
==========================================================================
====================
 *
 */
#include <Wire.h>
#include <U8x8lib.h>

#define I2C_SCL_Pin 21
#define I2C_SDA_Pin 20

U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/ U8X8_PIN_NONE);

const int OLED_refreshPeriod = 2000;
unsigned long OLED_millis = 0;

void OLED_setup(void);
void OLED_print(void);
/*====================ESP8266=============================================
==========================================================================
=======================================
 *
 */
#include <SoftwareSerial.h>

#define SER1_TX_Pin 18
#define SER1_RX_Pin 19

const int ESP_streamPeriod = 4000;
unsigned long ESP_millis = 0;

SoftwareSerial SoftSerial(SER1_RX_Pin, SER1_TX_Pin);

void Serial_setup(void);
void ESP_setupStream(void);
void ESP_stream(void);
/*==================RS485 PZEM-014
Sensor===================================================================
==========================================================================
====================
  RegAddr Description                Resolution
  0x0000  Voltage value              1LSB correspond to 0.1V
  0x0001  Current value low 16 bits  1LSB correspond to 0.001A
  0x0002  Current value high 16 bits
  0x0003  Power value low 16 bits    1LSB correspond to 0.1W
  0x0004  Power value high 16 bits
```

```
  0x0005  Energy value low 16 bits     1LSB correspond to 1Wh
  0x0006  Energy value high 16 bits
  0x0007  Frequency value              1LSB correspond to 0.1Hz
  0x0008  Power factor value           1LSB correspond to 0.01
  0x0009  Alarm status  0xFFFF is alarm, 0x0000is not alarm
 */
#include <ModbusMaster.h>

#define SER2_TX_Pin 16
#define SER2_RX_Pin 17

// SoftwareSerial pzemSerial(10,11); //rx, tx
ModbusMaster node;
static uint8_t pzemSlaveAddr = 0x01;

float voltage = 0;
float current = 0;
float power = 0;

const int PZEM_refreshPeriod = 3000;
unsigned long PZEM_millis = 0;

void PZEM_setup(void);
void PZEM_readData(void);
void PZEM_resetEnergy(uint8_t slaveAddr);
void PZEM_changeAddress(uint8_t OldslaveAddr, uint8_t NewslaveAddr);
/*==================Relay=============================================
==========================================================
=======================================
 *
 */
#define RELAY_Pin 25
#define ON HIGH
#define OFF LOW

bool relayState = 1;

void RELAY_setup(void);
void RELAY_write(bool);

/*==================System and
SAD================================================================
==========================================================
================
```

```
 *
 */
#include <movingAvg.h>

movingAvg master_Power(10);                 // define the moving
average object
movingAvg client1_Power(10);                 // define the moving
average object
movingAvg client2_Power(10);                 // define the moving
average object

int SAD_array[10][2] = {
  //master power , client1 + client2 power
  {0,0},
  {0,0},
  {0,0},
  {0,0},
  {0,0},
  {0,0},
  {0,0},
  {0,0},
  {0,0},
  {0,0}
};

const unsigned long SAD_refreshPeriod = 180000;
unsigned long SAD_millis = 0;
const int SAD_maxTolerance = 1000;

float master_Current = 0;
float master_Voltage = 0;
float master_freq = 50.0f;
int master_power = 0;
bool master_relayState = ON;

float client1_Current = 0;
float client1_Voltage = 0;
float client1_freq = 50.0f;
int client1_power = 0;
bool client1_relayState = ON;

float client2_Current = 0;
float client2_Voltage = 0;
float client2_freq = 50.0f;
int client2_power = 0;
```

```
bool client2_relayState = ON;

const float MASTER_MAX_CURRENT = 5.0f;
const float CLIENT1_MAX_CURRENT = 2.0f;
const float CLIENT2_MAX_CURRENT = 2.0f;
const float TOLERANT_ERROR_CURRENT = 0.5f;

unsigned long startTime;
const unsigned long timeout = 300000; // Timeout 5 menit dalam
milidetik
bool timeoutState = false;

void SAD_calculation(void);

void setup() {
  // put your setup code here, to run once:
  Serial_setup();
  ESP_setupStream();
  PZEM_setup();
  OLED_setup();
  LoRa_setup();
  RELAY_setup();
  RELAY_write(relayState);
  delay(2000);

  u8x8.drawString(2, 0, "SAD SYSTEM");
  u8x8.drawString(3,  0,  "  GARDU  ");
  delay(3000);
  u8x8.clearDisplay();
}

void loop() {
  // put your main code here, to run repeatedly:
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    LoRa_incoming = "";
    while (LoRa.available()) {
      LoRa_incoming = LoRa.readString();
    }
    Serial.println("Gateway Receive: ");
    LoRa_updateState = true;
  }
  PZEM_readData();
  // parse for a packet
  LoRa_parsingData();
```

```
  SAD_calculation();
  OLED_print();
  RELAY_write(relayState);
  ESP_stream();
  delay(1);
}

void Serial_setup(void){
  // Open serial communications and wait for port to open:
  Serial.begin(115200);
  Serial.println("Hello Master");
}
void  ESP_setupStream(void){
  SoftSerial.begin(115200);
  SoftSerial.println("Hello Streamer");
  Serial.println("Software Serial Init Success ...");
}
void ESP_stream(void){
  if(millis() - ESP_millis >= ESP_streamPeriod){
    ESP_millis = millis();

    Serial.print(F("stream send: "));
    Serial.print(F("/")); Serial.print(master_Voltage, 2);
Serial.print(F("/")); Serial.print(master_Current, 2);
Serial.print(F("/")); Serial.print(master_freq);
Serial.print(F("/")); Serial.print(master_relayState);
    Serial.print(F("/1/")); Serial.print(client1_Voltage, 2);
Serial.print(F("/")); Serial.print(client1_Current, 2);
Serial.print(F("/")); Serial.print(client1_freq);
Serial.print(F("/")); Serial.print(client1_relayState);
    Serial.print(F("/")); Serial.print(client2_Voltage, 2);
Serial.print(F("/")); Serial.print(client2_Current, 2);
Serial.print(F("/")); Serial.print(client2_freq);
Serial.print(F("/")); Serial.print(client2_relayState);
    Serial.println();

    SoftSerial.begin(115200);
    delay(10);
    SoftSerial.print(F("/")); SoftSerial.print(master_Voltage, 2);
SoftSerial.print(F("/")); SoftSerial.print(master_Current, 2);
SoftSerial.print(F("/")); SoftSerial.print(master_freq);
SoftSerial.print(F("/")); SoftSerial.print(master_relayState);
    SoftSerial.print(F("/1/")); SoftSerial.print(client1_Voltage,
2); SoftSerial.print(F("/")); SoftSerial.print(client1_Current, 2);
```

```
SoftSerial.print(F("/")); SoftSerial.print(client1_freq);
SoftSerial.print(F("/")); SoftSerial.print(client1_relayState);
    SoftSerial.print(F("/")); SoftSerial.print(client2_Voltage, 2);
SoftSerial.print(F("/")); SoftSerial.print(client2_Current, 2);
SoftSerial.print(F("/")); SoftSerial.print(client2_freq);
SoftSerial.print(F("/")); SoftSerial.print(client2_relayState);
    SoftSerial.end();
  }
}
void LoRa_setup(void){
  // override the default CS, reset, and IRQ pins (optional)
  SPI.begin();
  // LoRa.setPins(CS_Pin, RESET_Pin, IRQ_Pin);// set CS, reset, IRQ
pin
  if (!LoRa.begin(433E6)) {             // initialize ratio at 915
MHz
    Serial.println("LoRa init failed. Check your connections.");
    while (true);                       // if failed, do nothing
  }
  Serial.println("LoRa init succeeded ...");

  // LoRa.onReceive(LoRa_onReceive);
  // LoRa.onTxDone(LoRa_onTxDone);
  LoRa_rxMode();
}
void LoRa_rxMode(){
  LoRa.disableInvertIQ();               // normal mode
  LoRa.receive();                       // set receive mode
}
void LoRa_txMode(){
  LoRa.idle();                          // set standby mode
  LoRa.enableInvertIQ();                // active invert I and Q
signals
}
void LoRa_sendMessage(String message) {
  LoRa_txMode();                        // set tx mode
  LoRa.beginPacket();                   // start packet
  LoRa.print(message);                  // add payload
  LoRa.endPacket(true);                 // finish packet and send it
}
void LoRa_onReceive(int packetSize) {
  String message = "";

  while (LoRa.available()) {
    message = LoRa.readString();
```

```
  }

  Serial.println("Gateway Receive: ");
  // Serial.println(message);

  // String receiveMessage = LoRa_incoming;
  String dataStr;
  int firstBracket = message.indexOf('/');
  int secondBracket = message.indexOf('/' , firstBracket+1);
  int thirdBracket = message.indexOf('/' , secondBracket+1);
  int fourthBracket = message.indexOf('/' , thirdBracket+1);
  int fifthBracket = message.indexOf('/' , fourthBracket+1);

  dataStr = message.substring(firstBracket + 1, secondBracket);
  int recipient = dataStr.toInt();          // recipient address
  dataStr = message.substring(secondBracket + 1, thirdBracket);
  int sender = dataStr.toInt();             // sender address

  if(sender == client1Address && recipient == masterAddress){
    dataStr = message.substring(thirdBracket + 1, fourthBracket);
    client1_Voltage = dataStr.toInt();

    dataStr = message.substring(fourthBracket + 1, fifthBracket);
    client1_Current = dataStr.toInt()/1000.0;

    // Serial.print("client1:");
    // Serial.print(client1_Voltage); Serial.print("\t");
    // Serial.println(client1_Current);
  }
  if(sender == client2Address && recipient == masterAddress){
    dataStr = message.substring(thirdBracket + 1, fourthBracket);
    client2_Voltage = dataStr.toInt();

    dataStr = message.substring(fourthBracket + 1, fifthBracket);
    client2_Current = dataStr.toInt()/1000.0;

    // Serial.print("client2:");
    // Serial.print(client2_Voltage); Serial.print("\t");
    // Serial.println(client2_Current);
  }
}
void LoRa_onTxDone() {
  // Serial.println("TxDone");
  LoRa_rxMode();
}
```

```
void LoRa_parsingData(void){
  if(LoRa_updateState){
    LoRa_updateState = false;

    String receiveMessage = LoRa_incoming;
    String dataStr;
    uint8_t firstBracket = receiveMessage.indexOf('/');
    uint8_t secondBracket = receiveMessage.indexOf('/' ,
firstBracket+1);
    uint8_t thirdBracket = receiveMessage.indexOf('/' ,
secondBracket+1);
    uint8_t fourthBracket = receiveMessage.indexOf('/' ,
thirdBracket+1);
    uint8_t fifthBracket = receiveMessage.indexOf('/' ,
fourthBracket+1);

    dataStr = receiveMessage.substring(firstBracket + 1,
secondBracket);
    uint8_t recipient = dataStr.toInt();          // recipient
address
    dataStr = receiveMessage.substring(secondBracket + 1,
thirdBracket);
    uint8_t sender = dataStr.toInt();            // sender address

    if(sender == client1Address && recipient == masterAddress){
      dataStr = receiveMessage.substring(thirdBracket + 1,
fourthBracket);
      client1_Voltage = dataStr.toInt();

      dataStr = receiveMessage.substring(fourthBracket + 1,
fifthBracket);
      client1_Current = dataStr.toInt()/1000.0;

      Serial.print("client1: ");
      Serial.print(client1_Voltage); Serial.print("\t");
      Serial.println(client1_Current);
    }
    if(sender == client2Address && recipient == masterAddress){
      dataStr = receiveMessage.substring(thirdBracket + 1,
fourthBracket);
      client2_Voltage = dataStr.toInt();

      dataStr = receiveMessage.substring(fourthBracket + 1,
fifthBracket);
      client2_Current = dataStr.toInt()/1000.0;
```

```
      Serial.print("client2: ");
      Serial.print(client2_Voltage); Serial.print("\t");
      Serial.println(client2_Current);
    }
  }
}
void OLED_setup(void){
  // Setup the OLED display.
  delay(250); // wait for the OLED to power up
  u8x8.begin();
  u8x8.setPowerSave(0);
  u8x8.setFont(u8x8_font_chroma48medium8_r);
  Serial.println("OLED Init success ...");
}
void OLED_print(void){
  if(millis() - OLED_millis >= OLED_refreshPeriod){
    OLED_millis = millis();

    // Update display.
    char bufPrint[50];
    char bufVolt[10];
    char bufCurrent[10];
    char bufPower[10];

    u8x8.clearDisplay();

    u8x8.drawString(3, 0,"MONITORING");
    memset(bufVolt, 0, sizeof(bufVolt)); dtostrf(voltage, 3, 0,
bufVolt);
    memset(bufPrint, 0, sizeof(bufPrint)); sprintf(bufPrint,"V %5s
V", bufVolt);
    u8x8.drawString(2, 2, bufPrint);

    memset(bufCurrent, 0, sizeof(bufCurrent)); dtostrf(current, 3,
1, bufCurrent);
    memset(bufPrint, 0, sizeof(bufPrint)); sprintf(bufPrint,"C %5s
A", bufCurrent);
    u8x8.drawString(2, 3, bufPrint);

    memset(bufPower, 0, sizeof(bufPower)); dtostrf(power, 3, 0,
bufPower);
    memset(bufPrint, 0, sizeof(bufPrint)); sprintf(bufPrint,"P %5s
W", bufPower);
    u8x8.drawString(2, 4, bufPrint);
```

```
    memset(bufPrint, 0, sizeof(bufPrint));
sprintf(bufPrint,"S  %3s", relayState ? " ON" : "OFF");
    u8x8.drawString(2, 5, bufPrint);
  }
}
void RELAY_setup(void){
  pinMode(RELAY_Pin, OUTPUT);
  RELAY_write(OFF);
  Serial.println("Relay init succeeded ...");
}
void RELAY_write(bool state){
  digitalWrite(RELAY_Pin, state);
}
void PZEM_setup(void){
  Serial2.begin(9600);
  Serial.println("Serial 2 Init Success ...");

  node.begin(pzemSlaveAddr, Serial2);
  PZEM_resetEnergy(pzemSlaveAddr);
}
void PZEM_readData(void){
  if(millis() - PZEM_millis >= PZEM_refreshPeriod){
    PZEM_millis = millis();

    Serial2.begin(9600);
    // node.begin(pzemSlaveAddr, Serial2);
    // PZEM_resetEnergy(pzemSlaveAddr);

    uint8_t result;
    result = node.readInputRegisters(0x0000, 9); //read the 9
registers of the PZEM-014 / 016
    if (result == node.ku8MBSuccess){
      float _voltage = node.getResponseBuffer(0x0000) / 10.0;

      uint32_t tempdouble = 0x00000000;

      float _power;
      tempdouble |= node.getResponseBuffer(0x0003);        //LowByte
      tempdouble |= node.getResponseBuffer(0x0004) << 8;  //highByte
      _power = tempdouble / 10.0;


      float _current;
      tempdouble = node.getResponseBuffer(0x0001);        //LowByte
```

```
        tempdouble |= node.getResponseBuffer(0x0002) << 8;  //highByte
        _current = tempdouble / 1000.0;

        uint16_t _energy;
        tempdouble = node.getResponseBuffer(0x0005);       //LowByte
        tempdouble |= node.getResponseBuffer(0x0006) << 8;  //highByte
        _energy = tempdouble;

        voltage = _voltage;
        current  =  _current;
        power = _power;
        master_power = power;
        master_Current = current;
        master_Voltage = voltage;

        // Serial.print(voltage);
        // Serial.print("V    ");

        // Serial.print(current);
        // Serial.print("A    ");

        // Serial.print(power);
        // Serial.print("W  ");

        // Serial.print(node.getResponseBuffer(0x0008));
        // Serial.print("pf   ");

        // Serial.print(energy);
        // Serial.print("Wh   ");
        // Serial.println();

      } else {
        Serial.println("Failed to read modbus");
      }

      Serial2.end();
    }
}
void PZEM_resetEnergy(uint8_t slaveAddr){
  //The command to reset the slave's energy is (total 4 bytes):
  //Slave address + 0x42 + CRC check high byte + CRC check low byte.
  uint16_t u16CRC = 0xFFFF;
  static uint8_t resetCommand = 0x42;
  u16CRC = crc16_update(u16CRC, slaveAddr);
  u16CRC = crc16_update(u16CRC, resetCommand);
```

```
  Serial.println("Resetting Energy");
  Serial2.write(slaveAddr);
  Serial2.write(resetCommand);
  Serial2.write(lowByte(u16CRC));
  Serial2.write(highByte(u16CRC));
  delay(1000);
}
void PZEM_changeAddress(uint8_t OldslaveAddr, uint8_t NewslaveAddr)
{
  static uint8_t SlaveParameter = 0x06;
  static uint16_t registerAddress = 0x0002; // Register address to
be changed
  uint16_t u16CRC = 0xFFFF;
  u16CRC = crc16_update(u16CRC, OldslaveAddr);
  u16CRC = crc16_update(u16CRC, SlaveParameter);
  u16CRC = crc16_update(u16CRC, highByte(registerAddress));
  u16CRC = crc16_update(u16CRC, lowByte(registerAddress));
  u16CRC = crc16_update(u16CRC, highByte(NewslaveAddr));
  u16CRC = crc16_update(u16CRC, lowByte(NewslaveAddr));

  Serial.println("Changing Slave Address");

  Serial2.write(OldslaveAddr);
  Serial2.write(SlaveParameter);
  Serial2.write(highByte(registerAddress));
  Serial2.write(lowByte(registerAddress));
  Serial2.write(highByte(NewslaveAddr));
  Serial2.write(lowByte(NewslaveAddr));
  Serial2.write(lowByte(u16CRC));
  Serial2.write(highByte(u16CRC));
  delay(1000);
}
void SAD_calculation(void){
  relayState = master_relayState;

  client1_power = client1_Current * client1_Voltage;
  client2_power = client2_Current * client2_Voltage;
  // int avg_master_Power = master_Power.reading(master_power);
  // int avg_client1_Power = client1_Power.reading(client1_power);
  // int avg_client2_Power = client2_Power.reading(client2_power);

  if(((millis() - SAD_millis) >= SAD_refreshPeriod) && relayState){
    SAD_millis = millis();
```

```
    // Asumsikan elemen terakhir akan dihapus atau diganti dengan
nilai baru
    for (int i = 10 - 1; i > 0; i--) {
      SAD_array[i][0] = SAD_array[i - 1][0]; // Geser elemen ke
posisi berikutnya
      SAD_array[i][1] = SAD_array[i - 1][1]; // Geser elemen ke
posisi berikutnya
    }

    // Setelah loop, Anda bisa menetapkan nilai baru ke arr[0] jika
diperlukan
    SAD_array[0][0] = master_power;
    SAD_array[0][1] = client1_power + client2_power;

    for (int i=0; i<10; i++){
      Serial.print(SAD_array[i][0]); Serial.print("\t");
Serial.println(SAD_array[i][1]);
    }

    int sadValue = 0;
    for (int i = 0; i < 10; i++) {
      sadValue += abs(SAD_array[i][0] - SAD_array[i][1]); //
Tambahkan perbedaan absolut ke SAD
    }
    Serial.print("SAD Value: "); Serial.println(sadValue);

    if(sadValue >= SAD_maxTolerance){
      master_relayState = OFF;
      timeoutState = true;
      startTime = millis();
    }

    if(timeoutState){
      if (millis() - startTime >= timeout) {
        startTime = millis();

        master_relayState = ON;
        timeoutState = false;
      }
    }
  }
}
```

Program coding streamer/wifi

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <SoftwareSerial.h>

#define SSER_TX_Pin D2
#define SSER_RX_Pin D1

// software serial #1: RX = digital pin D7, TX = digital pin D8/D4
SoftwareSerial SoftSerial(SSER_RX_Pin, SSER_TX_Pin);

const char* ssid = "createme.id";
const char* password = "ccaaddok";

// 'https://thiefdetector.cloud/api/data'
//
'gardu_voltac','gardu_current','gardu_freq','gardu_relay','gardu_ala
rm',
// 'a_voltac','a_current','a_freq','a_relay',
// 'b_voltac','b_current','b_freq','b_relay',
//
https://thiefdetector.cloud/api/data/220/5/50/1/1/220/5/50/1/220/5/5
0/1

String urlBase = "https://thiefdetector.cloud/api/data";
String url = "";

unsigned long time_now = 0;
const long streamPeriod = 2000;

String dataSerial;

void constructURL(void);
void sendData(void);
void streamData(void);
void checkSoftSerial(void);

WiFiClient client;
```

```
// constants won't change. Used here to set a pin number:
const int ledPin = LED_BUILTIN;  // the number of the LED pin

// Variables will change:
int ledState = LOW;  // ledState used to set the LED

// Generally, you should use "unsigned long" for variables that hold
time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0;  // will store last time LED was
updated

// constants won't change:
const long interval = 500; // interval at which to blink
(milliseconds)

//
bool uploadState = false;

void setup() {
  // put your setup code here, to run once:
  // Open serial communications and wait for port to open:
  Serial.begin(115200);
  Serial.println("Hello Streamer");
  delay(100);

  // set the data rate for the SoftwareSerial port
  SoftSerial.begin(115200);
  // SoftSerial.println("Hello Master");
  Serial.println("Init SoftSerial One Success...");
  delay(100);

  WiFi.begin(ssid, password, 6);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(100);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected! IP address: ");
  Serial.println(WiFi.localIP());

  pinMode(ledPin, OUTPUT);
}
```

```
void loop() {
  // put your main code here, to run repeatedly:
  streamData();
  checkSoftSerial();
  blink();
  delay(100);
}


void sendData() {
  if (client.connect("solartracker.cloud", 80)) {
    client.print("GET ");
    client.print(url);
    client.println(" HTTP/1.1");
    client.print("Host: ");
    client.println("solartracker.cloud");
    client.println("Connection: close");
    client.println();
    client.stop();
    Serial.println("Send.");
  } else {
    Serial.println("Connection failed.");
  }
}

void constructURL() {
  // Construct the URL based on the variables
  url = urlBase + dataSerial;
  Serial.print("url:  ");
  Serial.println(url);
}

void checkSoftSerial(){
  String data="";
  bool serialState = false;
  if (SoftSerial.available() > 0) {

    data = SoftSerial.readString();
    // dataSerial = data;

    serialState = true;

    // Serial.print("serial: ");
    // Serial.println(dataSerial);
  }
```

```
  if (Serial.available() > 0) {

    data = Serial.readString();
    // dataSerial = data;

    serialState = true;

    // Serial.print("serial: ");
    // Serial.println(dataSerial);
  }

  if(serialState == true){
    serialState = false;

    dataSerial = data;
    Serial.println(dataSerial);

    uploadState = true;
  }
}

void streamData(){
  if (((millis() - time_now) >= streamPeriod) && uploadState) {
    time_now = millis();

    // Reconstruct URL with updated values
    constructURL();
    // Send data
    sendData();

    uploadState = false;
  }
}

void blink(){
  // check to see if it's time to blink the LED; that is, if the
difference
  // between the current time and last time you blinked the LED is
bigger than
  // the interval at which you want to blink the LED.
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;
```

```
    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW) {
      ledState = HIGH;
    } else {
      ledState = LOW;
    }

    // set the LED with the ledState of the variable:
    digitalWrite(ledPin, ledState);
  }
}
```

Program coding *client* 1

```
#include <Arduino.h>
#include <SPI.h>              // include libraries
#include <Wire.h>
#include <LoRa.h>
#include <U8x8lib.h>
#include <ACS712.h>
#include <movingAvg.h>
#include <ZMPT101B.h>

#define RELAY_Pin 8
#define I2C_SCL_Pin A5
#define I2C_SDA_Pin A4
#define CURENT_Pin A0
#define VOLTAGE_Pin A1
#define CS_Pin 10          // LoRa radio chip select
#define RESET_Pin 9        // LoRa radio reset
#define IRQ_Pin 2          // change for your board; must be a
hardware interrupt pin
#define SENSITIVITY 520.0f
#define FREQUENCY 50.0f

U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/ U8X8_PIN_NONE);
ACS712 ACS(CURENT_Pin, 5.0, 1023, 185);
movingAvg currentSensor(200);               // define the moving
average object

// ZMPT101B sensor output connected to analog pin A0
// and the voltage source frequency is 50 Hz.
ZMPT101B voltageSensor(VOLTAGE_Pin, FREQUENCY);
```

```
String outgoing;              // outgoing message

byte msgCount = 0;            // count of outgoing messages
byte client1Address = 0xAA;     // address of this device
byte client2Address = 0xAB;     // address of another device
byte masterAddress = 0xFF;      // destination to send to
long lastSendTime = 0;        // last send time
int interval = 2000;          // interval between sends

int period  =  2000;
unsigned long time_now = 0;

bool relayState = 1;

void setup() {
  // put your setup code here, to run once:
  // Open serial communications and wait for port to open:
  Serial.begin(115200);
  Serial.println("Hello Client");

  // Setup the OLED display.
  delay(250); // wait for the OLED to power up
  u8x8.begin();
  u8x8.setPowerSave(0);
  u8x8.setFont(u8x8_font_chroma48medium8_r);
  Serial.println("OLED Init success ...");

  // override the default CS, reset, and IRQ pins (optional)
  LoRa.setPins(CS_Pin, RESET_Pin, IRQ_Pin);// set CS, reset, IRQ pin
  if (!LoRa.begin(433E6)) {             // initialize ratio at 915
MHz
    Serial.println("LoRa init failed. Check your connections.");
    while (true);                       // if failed, do nothing
  }
  // LoRa.onReceive(onReceive);
  LoRa.onTxDone(onTxDone);
  // LoRa_rxMode();
  Serial.println("LoRa init succeeded.");

  ACS.autoMidPoint();
  // Serial.println(ACS.getMidPoint());
  currentSensor.begin();
  Serial.println("Measuring current with ACS712 ...");
```

```
  voltageSensor.setSensitivity(SENSITIVITY);
  Serial.println("Measuring voltage with ZMPT101B ...");

  pinMode(RELAY_Pin, OUTPUT);
  digitalWrite(RELAY_Pin, relayState);
  Serial.println("Set Relay Pin As OUTPUT ...");
}

void loop() {
  // put your main code here, to run repeatedly:
  int current_mA = 0;
  int avg_current_mA = 0;
  int voltage = 0;
  int power = 0;

  voltage = voltageSensor.getRmsVoltage();

  //=-0,000000015 *( F32*F32*F32) + 0,0001 *(F32*F32) + 0,3848 * F32
+ 23,265

  current_mA = ACS.mA_AC();
  avg_current_mA = currentSensor.reading(current_mA);

  float firstOrder = avg_current_mA * 0.4 ;
  double secondOrder = avg_current_mA * avg_current_mA * 0.0001 ;
  double thirdOrder = avg_current_mA * avg_current_mA *
avg_current_mA * (-0.000000015 );
  const int constValue = 23;

  double buff_current = abs(constValue + firstOrder + secondOrder +
thirdOrder );

  if(buff_current < 35) buff_current = 0;
  if((buff_current >= 165) && (buff_current<=190)) buff_current =
buff_current+20;
  if((buff_current >= 270) && (buff_current<=320)) buff_current =
buff_current-50;
  if((buff_current >= 630) && (buff_current<=670)) buff_current =
buff_current+200;
  if((buff_current >= 1100) && (buff_current<=1300)) buff_current =
buff_current+440;
  if((buff_current >= 1650) && (buff_current<=1850)) buff_current =
buff_current+690;
```

```
  if((buff_current >= 2150) && (buff_current<=2400)) buff_current =
buff_current+320;

  double current = buff_current / 1000;

  power = voltage * current;

  digitalWrite(RELAY_Pin, relayState);

  if(millis() - time_now >= period){
    time_now = millis();

    // Update display.
    char bufPrint[50];
    char bufVolt[10];
    char bufCurrent[10];
    char bufPower[10];
    // u8x8.clearDisplay();

    u8x8.drawString(2, 1, "MONITORING C1");

    memset(bufVolt, 0, sizeof(bufVolt)); dtostrf(voltage, 3, 0,
bufVolt);
    memset(bufPrint, 0, sizeof(bufPrint)); sprintf(bufPrint,"V %5s
V", bufVolt);
    u8x8.drawString(2, 2, bufPrint);

    memset(bufCurrent, 0, sizeof(bufCurrent)); dtostrf(current, 4,
2, bufCurrent);
    memset(bufPrint, 0, sizeof(bufPrint)); sprintf(bufPrint,"C %5s
A", bufCurrent);
    u8x8.drawString(2, 3, bufPrint);

    memset(bufPower, 0, sizeof(bufPower)); dtostrf(power, 3, 0,
bufPower);
    memset(bufPrint, 0, sizeof(bufPrint)); sprintf(bufPrint,"P %5s
W", bufPower);
    u8x8.drawString(2, 4, bufPrint);

    memset(bufPrint, 0, sizeof(bufPrint));
sprintf(bufPrint,"S  %3s", relayState ? " ON" : "OFF");
    u8x8.drawString(2, 5, bufPrint);
  }

  if (runEvery(random(500) + 4000)) { // repeat every 1000 millis
```

```
    String message = "/" + String(masterAddress) + "/" +
String(client2Address) + "/" + String(voltage) + "/" +
String(buff_current) + "/" + String(power);
    LoRa_sendMessage(message); // send a message
    Serial.print("Node Send: ");
    Serial.println(message);
}
}
void LoRa_rxMode(){
  LoRa.enableInvertIQ();                // active invert I and Q
signals
  LoRa.receive();                       // set receive mode
}

void LoRa_txMode(){
  LoRa.idle();                          // set standby mode
  LoRa.disableInvertIQ();               // normal mode
}

void LoRa_sendMessage(String message) {
  LoRa_txMode();                        // set tx mode
  LoRa.beginPacket();                   // start packet
  LoRa.print(message);                  // add payload
  LoRa.endPacket(true);                 // finish packet and send it
}

void onReceive(int packetSize) {
  String message = "";

  while (LoRa.available()) {
    message += (char)LoRa.read();
  }

  Serial.print("Node Receive: ");
  Serial.println(message);
}

void onTxDone() {
  // Serial.println("TxDone");
  LoRa_rxMode();
}

boolean runEvery(unsigned long interval)
{
```

```
  static unsigned long previousMillis = 0;
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval)
  {
    previousMillis = currentMillis;
    return true;
  }
  return false;
}
```

Program coding *client* 2

```
#include <Arduino.h>
#include <SPI.h>              // include libraries
#include <Wire.h>
#include <LoRa.h>
#include <U8x8lib.h>
#include <ACS712.h>
#include <movingAvg.h>
#include <ZMPT101B.h>

#define RELAY_Pin 8
#define I2C_SCL_Pin A5
#define I2C_SDA_Pin A4
#define CURENT_Pin A0
#define VOLTAGE_Pin A1
#define CS_Pin 10             // LoRa radio chip select
#define RESET_Pin 9           // LoRa radio reset
#define IRQ_Pin 2             // change for your board; must be a
hardware interrupt pin
#define SENSITIVITY 520.0f
#define FREQUENCY 50.0f

U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/ U8X8_PIN_NONE);
ACS712 ACS(CURENT_Pin, 5.0, 1023, 185);
movingAvg currentSensor(200);                // define the moving
average object

// ZMPT101B sensor output connected to analog pin A0
// and the voltage source frequency is 50 Hz.
ZMPT101B voltageSensor(VOLTAGE_Pin, FREQUENCY);


String outgoing;              // outgoing message
```

```
byte msgCount = 0;            // count of outgoing messages
byte client1Address = 0xAA;     // address of this device
byte client2Address = 0xAB;     // address of another device
byte masterAddress = 0xFF;      // destination to send to
long lastSendTime = 0;        // last send time
int interval = 2000;          // interval between sends

int period = 500;
unsigned long time_now = 0;

bool relayState = 1;

void setup() {
  // put your setup code here, to run once:
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  Serial.println("Hello Client");

  // Setup the OLED display.
  delay(250); // wait for the OLED to power up
  u8x8.begin();
  u8x8.setPowerSave(0);
  u8x8.setFont(u8x8_font_chroma48medium8_r);
  Serial.println("OLED Init success ...");

  // override the default CS, reset, and IRQ pins (optional)
  LoRa.setPins(CS_Pin, RESET_Pin, IRQ_Pin);// set CS, reset, IRQ pin
  if (!LoRa.begin(433E6)) {             // initialize ratio at 915
MHz
    Serial.println("LoRa init failed. Check your connections.");
    while (true);                       // if failed, do nothing
  }
  LoRa.onReceive(onReceive);
  LoRa.onTxDone(onTxDone);
  LoRa_rxMode();
  Serial.println("LoRa init succeeded.");

  ACS.autoMidPoint();
  // Serial.println(ACS.getMidPoint());
  currentSensor.begin();
  Serial.println("Measuring current with ACS712 ...");

  voltageSensor.setSensitivity(SENSITIVITY);
  Serial.println("Measuring voltage with ZMPT101B ...");
```

```
  pinMode(RELAY_Pin, OUTPUT);
  digitalWrite(RELAY_Pin, relayState);
  Serial.println("Set Relay Pin As OUTPUT ...");
}

void loop() {
  // put your main code here, to run repeatedly:
  int current_mA = 0;
  int avg_current_mA = 0;
  int voltage = 0;
  int power = 0;

  voltage = voltageSensor.getRmsVoltage();

  //=-0,000000015 *( F32*F32*F32) + 0,0001 *(F32*F32) + 0,3848 * F32
+ 23,265

  current_mA = ACS.mA_AC();
  avg_current_mA = currentSensor.reading(current_mA);

  float firstOrder = avg_current_mA * 0.4 ;
  double secondOrder = avg_current_mA * avg_current_mA * 0.0001 ;
  double thirdOrder = avg_current_mA * avg_current_mA *
avg_current_mA * (-0.000000015 );
  const int constValue = 23;

  double buff_current = abs(constValue + firstOrder + secondOrder +
thirdOrder );

  if(buff_current < 35) buff_current = 0;
  if((buff_current >= 165) && (buff_current<=190)) buff_current =
buff_current+20;
  if((buff_current >= 270) && (buff_current<=320)) buff_current =
buff_current-50;
  if((buff_current >= 630) && (buff_current<=670)) buff_current =
buff_current+200;
  if((buff_current >= 1100) && (buff_current<=1300)) buff_current =
buff_current+440;
  if((buff_current >= 1650) && (buff_current<=1850)) buff_current =
buff_current+690;
  if((buff_current >= 2150) && (buff_current<=2400)) buff_current =
buff_current+320;

  double current = buff_current / 1000;
```

```
  power = voltage * current;

  digitalWrite(RELAY_Pin, relayState);

  if(millis() - time_now >= period){
    time_now = millis();

    // Update display.
    char bufPrint[50];
    char bufVolt[10];
    char bufCurrent[10];
    char bufPower[10];
    // u8x8.clearDisplay();

    u8x8.drawString(2, 1, "MONITORING C1");

    memset(bufVolt, 0, sizeof(bufVolt)); dtostrf(voltage, 3, 0,
bufVolt);
    memset(bufPrint, 0, sizeof(bufPrint)); sprintf(bufPrint,"V %5s
V", bufVolt);
    u8x8.drawString(2, 2, bufPrint);

    memset(bufCurrent, 0, sizeof(bufCurrent)); dtostrf(current, 4,
2, bufCurrent);
    memset(bufPrint, 0, sizeof(bufPrint)); sprintf(bufPrint,"C %5s
A", bufCurrent);
    u8x8.drawString(2, 3, bufPrint);

    memset(bufPower, 0, sizeof(bufPower)); dtostrf(power, 3, 0,
bufPower);
    memset(bufPrint, 0, sizeof(bufPrint)); sprintf(bufPrint,"P %5s
W", bufPower);
    u8x8.drawString(2, 4, bufPrint);

    memset(bufPrint, 0, sizeof(bufPrint));
sprintf(bufPrint,"S  %3s", relayState ? " ON" : "OFF");
    u8x8.drawString(2, 5, bufPrint);
  }

  if (runEvery(random(500) + 4000)) { // repeat every 1000 millis

    String message = "/" + String(masterAddress) + "/" +
String(client2Address) + "/" + String(voltage) + "/" +
String(buff_current) + "/" + String(power);
```

```
    LoRa_sendMessage(message); // send a message
    Serial.print("Node Send: ");
    Serial.println(message);
}
}
void LoRa_rxMode(){
  LoRa.enableInvertIQ();                // active invert I and Q
signals
  LoRa.receive();                       // set receive mode
}

void LoRa_txMode(){
  LoRa.idle();                          // set standby mode
  LoRa.disableInvertIQ();               // normal mode
}

void LoRa_sendMessage(String message) {
  LoRa_txMode();                        // set tx mode
  LoRa.beginPacket();                   // start packet
  LoRa.print(message);                  // add payload
  LoRa.endPacket(true);                 // finish packet and send it
}

void onReceive(int packetSize) {
  String message = "";

  while (LoRa.available()) {
    message += (char)LoRa.read();
  }

  Serial.print("Node Receive: ");
  Serial.println(message);
}

void onTxDone() {
  // Serial.println("TxDone");
  LoRa_rxMode();
}

boolean runEvery(unsigned long interval)
{
  static unsigned long previousMillis = 0;
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval)
  {
```

```
    previousMillis = currentMillis;
    return true;
  }
  return false;
}
```

LAMPIRAN D Daftar Riwayat Hidup

DAFTAR RIWAYAT HIDUP

Zahra Ari Maulida, Lahir di Wonogiri, 13 Mei 2003. Anak pertama dari dua bersaudara, dari pasangan Bapak Budi Arianto dan Ibu Sri Rahyuni. Mempunyai 1saudara kandung adik Aliyah Calista Ariagustine dan Beragama Islam. Bertempat tinggal di Gubeng Klingsingan 4/35, Kecamatan Gubeng, KelurahanGubeng, Kota Surabaya, Jawa Timur.

.

Dengan pendidikan formal yang pernah diikuti sebagai berikut :

1. SDN Gubeng 1 Surabaya (lulus pada tahun 2015)
2. SMPN 6 Surabaya (lulus pada tahun 2018)
3. SMAN 6 Surabaya (lulus pada tahun 2021)

Pada tahun 2021 diterima sebagai Taruna di Politeknik Penerbangan Surabaya Program Studi Diploma 3 Teknik Listrik Bandar Udara Angkatan ke-16. Melaksanakan *On the Job Training* pertama di Bandar Udara Aji Pangeran Tumenggung Pranoto Samarinda pada tanggal 08 Mei 2023 sampai 12 September 2023 dan *On the Job Training* kedua di Bandar Jenderal Ahmad Yani Semarang pada tanggal 02 Oktober 2023 sampai 29 Februari 2