PPNS POLITEKNIK PERKAPALAN NEGERI SURABAYA

TUGAS AKHIR (AE3250)

TRANSAKSI ENERGI PLTS DENGAN BLOCKCHAIN ETHEREUM DISERTAI SARAN PEMBELIAN BERBASIS REGRESI LINIER

Ach. Arif Zuchal Ulya
NRP.0921040023

Dosen Pendamping :
Noorman Rinanto, S.T., M.T., Ph.D.
Ir. Joko Endrasmono, M.T.

PROGRAM STUDI D4-TEKNIK OTOMASI
JURUSAN TEKNIK KELISTRIKAN KAPAL
POLITEKNIK PERKAPALAN NEGERI SURABAYA
SURABAYA
2025

“Halaman ini sengaja dikosongkan”

TUGAS AKHIR (AE3250)

TRANSAKSI ENERGI PLTS DENGAN *BLOCKCHAIN* ETHEREUM DISERTAI SARAN PEMBELIAN BERBASIS REGRESI LINIER

Ach. Arif Zuchal Ulya
NRP. 0921040023

Dosen Pendamping
Noorman Rinanto, S.T., M.T., Ph.D.
Ir. Joko Endrasmono, M.T.

PROGRAM STUDI D4-TEKNIK OTOMASI
JURUSAN TEKNIK KELISTRIKAN KAPAL
POLITEKNIK PERKAPALAN NEGERI SURABAYA
SURABAYA
2025

"Halaman Sengaja Dikosongkan"

LEMBAR PENGESAHAN
TUGAS AKHIR

TRANSAKSI ENERGI PLTS DENGAN *BLOCKCHAIN* ETHEREUM DISERTAI SARAN PEMBELIAN BERBASIS REGRESI LINIER.

Disusun Oleh:
Ach. Arif Zuchal Ulya
0921040023

Diajukan Untuk Memenuhi Salah Satu Syarat Kelulusan
Program Studi D4 Teknik Otomasi
Jurusan Teknik Kelistrikan Kapal
POLITEKNIK PERKAPALAN NEGERI SURABAYA

Disetujui oleh Tim penguji Tugas Akhir Tanggal Ujian : 21 Juli 2025
Periode Wisuda : Oktober 2025

Menyetujui,

Dosen Penguji	NUPTK	Tanda Tangan
1. Ir. Joko Endrasmono, M.T.	(1241742643130090)	(.....)
2. Lilik Subiyanto, S.T, M.T.	(4462747648130070)	(.....)
3. Anggarjuna Puncak Pujiputra, S.T., M.T.	(6535772673130252)	(.....)
4. Mustika Kurnia Mayangsari, S.Kom., M.Tr.Kom.	(0835774675230272)	(.....)

Dosen Pembimbing	NUPTK	Tanda Tangan
1. Noorman Rinanto, S.T., M.T., Ph.D.	(6346754655130083)	(.....)
2. Ir. Joko Endrasmono, M.T.	(1241742643130090)	(.....)

Menyutujui
Ketua Jurusan,

Isa Rachman, S.T., M.T.
NIP. 198008162008121001

Menyutujui
Koordinator Program Studi,

Agus Khumaidi, S.ST., M.T.
NIP. 199308172020121004

PPNS	**PERNYATAAN BEBAS PLAGIAT**	No. : F.WD I. 021 Date : 3 Nopember 2015 Rev. : 01 Page : 1 dari 1

Yang bertandatangan dibawah ini :

Nama	:	Ach. Arif Zuchal Ulya
NRP	:	0921040023
Jurusan/Prodi	:	Teknik Kelistrikan Kapal / D4 Teknik Otomasi

Dengan ini menyatakan dengan sesungguhnya bahwa :

Tugas Akhir yang akan saya kerjakan dengan judul :

TRANSAKSI ENERGI PLTS DENGAN *BLOCKCHAIN* ETHEREUM DISERTAI SARAN PEMBELIAN BERBASIS REGRESI LINIER

Adalah **benar karya saya sendiri** dan **bukan plagiat dari karya orang lain.**

Apabila dikemudian hari terbukti terdapat plagiat dalam karya ilmiah tersebut, maka saya bersedia menerima **sanksi** sesuai ketentuan peraturan yang berlaku.

Demikian surat pernyataan ini saya buat dengan penuh tanggung jawab.

Surabaya, 18 Agustus 2025

Yang Membuat Pernyataan.

(Ach. Arif Zuchal Ulya)
NRP. 0921040023

“Halaman ini sengaja dikosongkan”

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas segala rahmat, taufik, dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul Transaksi Energi PLTS dengan *Blockchain* Ethereum Disertai Saran Pembelian Berbasis Regresi Linier ini sebagai salah satu syarat untuk menyelesaikan pendidikan jenjang Sarjana Terapan pada Program Studi D4 Teknik Otomasi, Jurusan Teknik Kelistrikan Kapal, Politeknik Perkapalan Negeri Surabaya.

Penulisan tugas akhir ini tidak lepas dari bantuan berbagai pihak, oleh karena itu penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Bapak Noorman Rinanto, S.T., M.T., Ph.D. selaku dosen pembimbing I atas bimbingan, arahan, dan motivasi selama proses penyusunan.
2. Bapak Ir. Joko Endrasmono, M.T. selaku dosen pembimbing II yang telah memberikan banyak masukan teknis dan semangat.
3. Seluruh dosen dan staf Program Studi D4 Teknik Otomasi yang telah memberikan ilmu dan dukungan selama masa studi.
4. Orang tua dan keluarga besar atas doa, dukungan moral, dan materi yang tidak ternilai.
5. Teman-teman seperjuangan yang telah menjadi rekan diskusi dan berbagi semangat hingga tugas akhir ini selesai.

Penulis menyadari bahwa tugas akhir ini masih memiliki kekurangan, oleh karena itu kritik dan saran yang membangun sangat diharapkan untuk penyempurnaan di masa mendatang.

Semoga tugas akhir ini dapat memberikan manfaat dan kontribusi nyata di bidang energi terbarukan, khususnya dalam penerapan teknologi blockchain dan sistem prediksi cerdas dalam pengelolaan energi PLTS.

Surabaya, 27 Juli 2025
Penulis,

Ach. Arif Zuchal Ulya

“Halaman ini sengaja dikosongkan”

TRANSAKSI ENERGI PLTS DENGAN BLOCKCHAIN ETHEREUM DISERTAI SARAN PEMBELIAN BERBASIS REGRESI LINIER

Ach. Arif Zuchal Ulya

ABSTRAK

Penelitian ini mengintegrasikan teknologi blockchain Ethereum dengan Pembangkit Listrik Tenaga Surya (PLTS) untuk mencatat transaksi energi yang transparan, aman, dan efektif. *Smart contract* digunakan untuk mengotomatisasi transaksi serta memastikan ketahanan terhadap manipulasi, dengan validasi TxHash melalui server dan Etherscan. Pengujian menunjukkan seluruh transaksi berhasil diproses dengan rata-rata waktu eksekusi sekitar 22 detik, serta nominal transaksi tidak memengaruhi delay. Selain itu, metode regresi linear diterapkan untuk memprediksi kebutuhan energi berdasarkan data historis konsumsi. Hasil menunjukkan bahwa akurasi prediksi sangat dipengaruhi oleh stabilitas data, dengan galat minimum 0,54% pada beban stabil (*charger* laptop) dan maksimum 24,03% pada beban fluktuatif (kipas angin). Rata-rata galat parameter regresi kecil (*slope* 0,19% dan *intercept* 0,39%), menegaskan bahwa metode ini efektif mendukung pengambilan keputusan. Penelitian ini membuktikan bahwa integrasi blockchain dan regresi linear dapat meningkatkan transparansi, keamanan, dan efektivitas pengelolaan energi terbarukan.

Kata Kunci: *Blockchain*, Ethereum, PLTS, *Smart Contract*, Regresi Linier

"Halaman ini sengaja dikosongkan"

SOLAR POWER PLANT ENERGY TRANSACTIONS WITH ETHEREUM BLOCKCHAIN INCLUDING PURCHASE RECOMMENDATIONS USINGLINEAR REGRESSION

Ach. Arif Zuchal Ulya

ABSTRACT

This study integrates Ethereum blockchain with Solar Power Plants (PLTS) to record energy transactions transparently, securely, and effectively. Smart contracts automate transactions and prevent manipulation, with TxHash validation through the server and Etherscan. Experiments showed all transactions were executed successfully with an average processing time of ~22 seconds, while transaction amounts had no effect on delay. Linear regression was applied to predict energy demand using historical consumption data. Prediction accuracy depended on load stability, achieving a minimum error of 0.54% under stable loads (laptop charger) and a maximum error of 24.03% under fluctuating loads (electric fan). The regression parameter errors were minimal (slope: 0.19%; intercept: 0.39%), confirming its reliability for decision-making. Overall, the integration of blockchain and linear regression enhances transparency, security, and efficiency in renewable energy management..

Keywords: Blockchain, Ethereum, Solar Power Plant, Smart Contract, Linear Regression,

“Halaman ini sengaja dikosongkan”

DAFTAR ISI

DAFTAR TABEL

“Halaman ini sengaja dikosongkan”

DAFTAR GAMBAR

DAFTAR LISTING

DAFTAR NOTASI

V : Tegangan listrik (Volt)

I : Arus listrik (Ampere)

E : Energi listrik (kWh)

f : Frekuensi listrik (Hertz)

T : Waktu (detik)

α : Koefisien intersep regresi linear

β : Koefisien slope regresi linear

Y : Variabel dependen dalam analisis regresi (kebutuhan energi)

X : Variabel independen dalam analisis regresi (data historis konsumsi energi)

TxHash : Hash transaksi pada blockchain

ETH : Mata uang kripto Ethereum

n: Jumlah data.

X_i, Y_i*: Data ke-i dari variabel independen dan dependen.*

Y_i*: Nilai aktual.*

$\hat{Y}$ *: Nilai yang diukur oleh sistem.*

n : Jumlah data

$\sum XY$ *: Hasil dari penjumlahan indeks (X) dikali dengan energi (Y)*

$\sum X$ *: Hasil dari penjumlahan indeks atau X*

$\sum Y$ *: Hasil dari penjumlahan nilai energi atau Y*

$\sum X2$ *: Hasil dari penguadratan nilai x atau indeks*

$(\sum X)2$ *: Hasil dari penguadratan yang berasal setelah penjumlahan nilai x atau indeks*

"Halaman ini sengaja dikosongkan"

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi telah mendorong transformasi besar di berbagai sektor industri, termasuk sektor energi yang mengalami perubahan signifikan, seperti adopsi energi terbarukan seperti Pembangkit Listrik Tenaga Surya (PLTS) yang terintegrasi dengan teknologi *blockchain*. PLTS menjadi semakin penting karena perannya dalam mengurangi emisi karbon dan mengatasi dampak perubahan iklim, sehingga menjadi komponen krusial dalam transisi energi global. Selain memberikan solusi ramah lingkungan, PLTS juga membuka peluang untuk menciptakan sistem energi yang lebih berkelanjutan bagi generasi mendatang.

Namun, integrasi sumber energi terbarukan ini menghadapi berbagai tantangan kompleks, seperti ketidakseimbangan distribusi energi karena keterbatasan infrastruktur, kebutuhan akan sistem penyimpanan energi yang efektif untuk mengatasi sifat intermiten energi surya, dan kesulitan dalam sinkronisasi data *real-time* antar penyedia dan konsumen energi. Pertama, distribusi energi yang efisien tetap menjadi masalah utama karena sifat energi surya yang intermiten dan bergantung pada kondisi cuaca. Kedua, diperlukan sistem manajemen jaringan yang mampu menangani fluktuasi pasokan dan permintaan energi secara real-time. Ketiga, pencegahan pencurian energi serta optimasi transaksi masih menjadi masalah mendesak yang memerlukan solusi teknologi yang kuat.

Dalam konteks ini, teknologi *blockchain*, khususnya platform Ethereum, menawarkan solusi inovatif untuk mengatasi tantangan tersebut. *Blockchain*, sebagai teknologi buku besar terdesentralisasi yang aman dan transparan, memiliki potensi besar dalam mengelola dan mendistribusikan energi secara lebih efisien. Misalnya, *blockchain* telah digunakan untuk

mencatat transaksi energi dalam proyek seperti Brooklyn Microgrid, yang menggunakan *blockchain* Ethereum. Dalam proyek ini, konsumen dan produsen energi lokal dapat melakukan perdagangan energi secara langsung dengan transparansi penuh, didukung oleh teknologi smart contracts pada platform Ethereum. Dengan menerapkan smart contracts, berbagai proses dalam sistem energi dapat diotomatisasi dengan tingkat keamanan dan efisiensi yang tinggi (Mehdi Montakhabi et al., 2024).

Keunggulan teknologi *blockchain* dalam sektor energi mencakup kemampuannya menciptakan sistem yang transparan dan dapat diaudit. Setiap transaksi energi dapat dicatat secara permanen dan dilindungi dari manipulasi, sehingga meningkatkan kepercayaan di antara para pemangku kepentingan. Selain itu, *blockchain* memungkinkan optimasi distribusi energi menggunakan data real-time yang akurat dan terverifikasi. Teknologi ini juga memungkinkan terciptanya pasar energi peer-to-peer (P2P), di mana konsumen dapat berperan sebagai prosumer (producer-consumer) dan menjual kelebihan energi mereka ke jaringan. Sistem ini mendorong demokratisasi pasar energi sekaligus menciptakan insentif ekonomi untuk memperluas adopsi energi terbarukan (Mehdi Montakhabi et al., 2024).

Salah satu tantangan utama dalam distribusi energi adalah ketidakseimbangan antara pasokan dan permintaan, terutama karena sifat intermiten PLTS. Menurut penelitian oleh Selsie Anggela Putri (2020), kelebihan kapasitas listrik hingga 42% menjadi masalah serius dalam pengembangan PLTS, karena energi yang dihasilkan tidak selalu sejalan dengan kebutuhan konsumen. *Blockchain* dapat mengatasi masalah ini dengan memfasilitasi distribusi energi yang lebih fleksibel melalui sistem P2P. Dalam model ini, kelebihan energi yang dihasilkan oleh satu rumah tangga dapat langsung dijual ke rumah tangga lain yang membutuhkan, mengurangi ketergantungan pada infrastruktur penyimpanan energi yang mahal dan tidak efisien.

Selain itu, *blockchain* juga dapat meningkatkan efisiensi distribusi energi melalui penggunaan data real-time. Kapengut & Mizrach (2022) dalam

penelitiannya tentang transisi Ethereum ke Proof-of-Stake (PoS) menunjukkan bahwa *blockchain* dapat menangani hingga 1,148,750 transaksi per hari dengan konsumsi energi yang jauh lebih rendah (99,98% lebih efisien dibandingkan Proof-of-Work). Hal ini memungkinkan sistem energi berbasis *blockchain* untuk memproses data pasokan dan permintaan secara cepat dan akurat, sehingga fluktuasi energi dapat dikelola dengan lebih baik.

Metode regresi linier digunakan untuk memberikan rekomendasi pembelian energi berdasarkan data historis transaksi, seperti pola konsumsi dan produksi energi serta harga pasar. Data ini dikumpulkan dari log transaksi *blockchain* dan perangkat IoT yang terintegrasi, kemudian diolah menggunakan algoritma prediksi untuk menghasilkan rekomendasi yang dapat meningkatkan efisiensi dan akurasi. Pendekatan ini membantu memprediksi kebutuhan energi dengan lebih akurat, mendukung pengambilan keputusan yang optimal bagi konsumen maupun penyedia energi. Seperti yang ditunjukkan oleh Lukman Hakim & Utari (2020), metode regresi linier dapat mencapai tingkat akurasi prediksi yang tinggi (MAPE 8,3%), yang dapat diterapkan untuk memprediksi kebutuhan energi dalam sistem berbasis *blockchain*.

Dengan demikian, integrasi teknologi *blockchain* dengan PLTS, disertai penerapan rekomendasi pembelian berbasis regresi linier, dapat diterapkan dalam skenario nyata seperti manajemen distribusi energi lokal di wilayah pedesaan atau perkotaan yang menggunakan PLTS. Teknologi ini memungkinkan konsumen menjual kelebihan energi melalui platform *blockchain*, sementara metode regresi linier membantu merancang strategi pembelian energi berdasarkan kebutuhan aktual yang diprediksi dengan data historis. Meskipun tantangan teknis dan regulasi masih perlu diatasi, kombinasi inovasi ini dapat secara signifikan mendukung transformasi sektor energi.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, disimpulkan pertanyaan penelitan yang akan dibahas dalam penelitian Tugas akhir ini, sebagai berikut :

1. Bagaimana teknologi *blockchain* pada platform Ethereum dapat diintegrasikan dengan transaksi energi dari pasokan sumber energi terbarukan seperti PLTS?
2. Bagaimana penerapan *smart contract*s dalam *blockchain* dapat memastikan transaksi energi yang transparan, aman, dan efektif antara penyedia dan konsumen?
3. Bagaimana penerapan metode regresi linier dapat memberikan rekomendasi pembelian energi berikutnya guna mengoptimalkan transaksi energi pada sistem berbasis *blockchain*?

1.3 Batasan Penelitian

Agar penelitian tugas akhir ini dapat berjalan dan tepat sasaran tanpa mengurangi maksud dan tujuan penelitian tugas akhir, maka diperlukan batasan penelitian. Berikut adalah batasan penelitiannya :

1. Variabel penelitian mencakup jumlah daya yang digunakan dan nilai transaksi menggunakan mata uang Ethereum sepolia,
2. Aktuator yang dikontrol hanya relay pada kontroler
3. Lingkungan jaringan yang digunakan sebatas lokal
4. Penelitian ini akan terbatas pada penerapan platform *blockchain* Ethereum dan tidak akan membahas penggunaan platform *blockchain* lain dalam pengelolaan energi terbarukan
5. Fokus utama penelitian ini adalah pada penerapan *smart contracts* untuk transaksi energi yang aman dan pendistribusian energi berdasarkan nilai ETH yang dibeli
6. Fokus penelitian ini adalah untuk mengimplementasikan metode regresi linear berdasarkan riwayat konsumen guna membuat rekomendasi pembelian berikutnya menggunakan studi kasus sederhana.

7. Pada penelitian ini kita menggunakan skenario tiga konsumer dengan asumsi tiga rumah sebagai konsumer yang tiap satu rumah diwakilkan oleh satu perabotan listrik dan satu rumah lagi sebagai penyedia layanan.
8. Perabotan listrik berupa kipas angin (konsumer 1), *charger* laptop (konsumer 2), dan *charger* telepon genggam (konsumer 3)

1.4 Tujuan Penelitian

Dalam setiap penelitian tentunya tidak lepas dari tujuan yang ingin dicapai. Berkaitan dengan permasalahannya, tujuan yang ingin dicapai oleh penulis dalam penelitian ini adalah untuk :

1. Untuk mengintegrasikan sistem blockchain tersebut dengan data energi yang dihasilkan oleh Pembangkit Listrik Tenaga Surya (PLTS) sebagai sumber energi terbarukan.
2. Penerapan *smart contract*s dalam *blockchain* untuk memastikan transaksi energi yang transparan dan aman antara penyedia energi dan konsumen, serta meningkatkan kepercayaan dalam sistem transaksi energi..
3. Penerapan metode regresi linier untuk memberikan rekomendasi pembelian energi berikutnya guna meningkatkan efektifitas dan mengoptimalkan transaksi energi berbasis *blockchain*.

1.5 Manfaat Penelitian

Dalam penelitian tugas akhir ini memiliki beberapa manfaat. Berikut manfaat dari penelitian tugas akhir ini sebagai berikut.

1. Menawarkan solusi inovatif untuk meningkatkan efektivitas distribusi energi, mengurangi ketergantungan pada sumber energi konvensional, dan mengoptimalkan manajemen jaringan energi dengan memanfaatkan teknologi *blockchain* serta rekomendasi pembelian berbasis regresi linier.
2. Menciptakan sistem transaksi energi yang transparan, aman, dan efektif melalui penerapan *smart contract*s pada *blockchain*, sehingga

meningkatkan kepercayaan antara penyedia energi dan konsumen sekaligus meminimalkan potensi kecurangan dan kesalahan transaksi.

3. Mendukung pengembangan pasar energi *peer-to-peer* yang lebih demokratis serta menciptakan insentif ekonomi untuk memperluas adopsi energi terbarukan secara luas, mendukung keberlanjutan lingkungan.

BAB 2
TINJAUAN PUSTAKA

Pada bab tinjauan pustaka ini mencakup peninjauan literatur yang merujuk pada beberapa referensi atau sumber dari penelitian sebelumnya sebagai panduan untuk metode, penjelasan dasar teori yang mendukung topik permasalahan yang diangkat, serta *software* dan *hardware* yang digunakan dalam menyelesaikan penelitian tugas akhir ini.

2.1 Kajian Penelitian Terdahulu

Berikut rujukan dari beberapa penelitian sebelumnya yang berhubungan dari topik penelitian tugas akhir ini :

1. **Analisis Hambatan Pemanfaatan PLTS di Provinsi Jawa Tengah.**

   Jurnal ini membahas hambatan pengembangan PLTS di Jawa Tengah, termasuk kurangnya infrastruktur penyimpanan energi dan kelebihan kapasitas listrik yang menghambat integrasi energi terbarukan. Fokus utama adalah memanfaatkan potensi radiasi matahari di Jawa Tengah yang cukup tinggi.(Selsie Anggela Putri, 2020)

   Hasil Pengujian:

   1. Kelebihan kapasitas listrik hingga 42% menjadi tantangan pengembangan PLTS.
   2. PLTS mayoritas tidak menggunakan baterai, hanya berfungsi pada siang hari.

   Kelebihan:

   1. Radiasi matahari antara 3,5–4,67 kWh/m²/hari memungkinkan pengembangan PLTS yang signifikan.
   2. PLTS tidak menghasilkan polusi atau emisi karbon.

   Kekurangan:

1. Hanya digunakan untuk urusan rumah tangga.
2. Belum terdapat protokol untuk jual beli energi.

Potensi Pengembangan:

1. Penambahan fitur IoT untuk efisiensi energi.
2. Menggunakan *blockchain* untuk transparansi transaksi sisa energi.

2. **Analisis Peran dan Tantangan Penggunaan *Blockchain* dalam *E-Commerce***

Penelitian ini mengulas penerapan *blockchain* untuk meningkatkan keamanan dan transparansi transaksi *e-commerce*. Studi ini menunjukkan bagaimana *blockchain* dapat meningkatkan kepercayaan konsumen dan mengurangi sengketa transaksi. (Akbar Bathnul Wadi Nst et al., 2025)

Hasil Pengujian:

1. *Blockchain* meningkatkan transparansi transaksi, menurunkan sengketa dari 15% menjadi 5%.
2. Kepercayaan konsumen meningkat 20% dalam 6 bulan.

Kelebihan:

1. Setiap transaksi dicatat dalam buku besar digital yang dapat diakses publik.
2. Teknologi *blockchain* menyediakan enkripsi yang kuat untuk melindungi data.

Kekurangan:

1. Penelitian menggunakan server biasa, bukan *blockchain* publik seperti Ethereum.

Potensi Pengembangan:

1. Mengintegrasikan *blockchain* dengan IoT.
2. Membuat antarmuka pengguna yang lebih ramah bagi konsumen nonteknis.

3. An Event Study *of* the Ethereum Transition to *Proof-of-Stake*

Jurnal ini mengevaluasi dampak transisi Ethereum dari PoW (*Proof-of-Work*) ke PoS (*Proof-of-Stake*). Fokus utamanya pada efisiensi energi, kecepatan transaksi, dan manfaatnya untuk mendukung aplikasi *blockchain*. (Kapengut & Mizrach, 2022)

Hasil Pengujian:

1. Konsumsi energi berkurang 99,98% setelah transisi ke PoS.
2. Kecepatan transaksi meningkat hingga 1,148,750 transaksi/hari.

Kelebihan:

1. Ramah lingkungan dengan pengurangan konsumsi energi.
2. Mendukung integrasi *smart contract* untuk pencatatan data energi.

Kekurangan:

1. Biaya transaksi (*gas fee*) masih menjadi kendala.

Potensi Pengembangan:

1. Pemanfaatan Ethereum untuk mencatat data transaksi energi.

4. ***Leveraging Blockchain for Energy Transition***

Penelitian ini menggunakan kasus Brooklyn Microgrid untuk menunjukkan bagaimana *blockchain* mendukung model energi berbasis komunitas. Fokus pada model P2P (*Peer-to-peer*) dan CSC (Community Self-Consumption) untuk berbagi energi.(Mehdi Montakhabi et al., 2024)

Hasil Pengujian:

1. *Blockchain* digunakan dalam model P2P *(Peer-to-peer)* dan CSC *(Community Self- Consumption)* untuk berbagi surplus energi komunitas.
2. Brooklyn Microgrid menunjukkan transaksi energi yang transparan dan aman.

Kelebihan:

1. *Blockchain* meningkatkan efisiensi dan kepercayaan dalam transaksi energi komunitas.
2. Memotivasi prosumer untuk berkontribusi dalam pasar energi.

Kekurangan:

1. Infrastruktur digital yang kompleks dan mahal.

Potensi Pengembangan:

1. Mengintegrasikan prediksi kebutuhan energi berbasis regresi linear untuk optimasi transaksi.

5. **Prediksi Jumlah Pembelian Sepatu dengan Penerapan Metode Regresi Linear**

Studi ini mengeksplorasi metode regresi linear untuk memprediksi pembelian di *e-commerce*. Fokus pada peningkatan akurasi prediksi untuk membantu pengambilan keputusan. (Lukman Hakim & Utari, 2020)

Hasil Pengujian:

1. Tingkat *error* prediksi (MAPE) sebesar 8,3%.
2. Prediksi pembelian bulan berikutnya berdasarkan data penjualan terakhir dan stok awal.

Kelebihan:

1. Akurasi tinggi dalam prediksi menggunakan regresi linear.

Kekurangan:

1. Belum ada integrasi validasi transaksi.

Potensi Pengembangan:

2. Menggabungkan metode regresi linear dengan *blockchain* untuk prediksi kebutuhan energi PLTS.

2.2 Kajian Pustaka

Dalam melakukan penelitian pada penelitian tugas akhir ini, penulis memerlukan beberapa dasar teori yang dapat mendukung topik

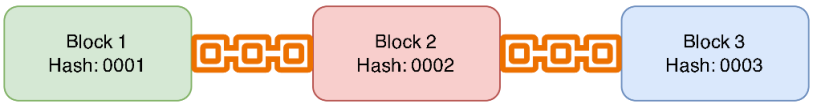
permasalahan yang diambil. Adapun dasar teori dari penelitian tugas akhir ini.

2.2.1 *Blockchain*

Blockchain adalah teknologi yang berfungsi sebagai buku besar digital terdesentralisasi untuk mencatat transaksi secara transparan, aman, dan tidak dapat diubah. Teknologi ini pertama kali diperkenalkan oleh Satoshi Nakamoto pada tahun 2008 sebagai dasar untuk mata uang kripto Bitcoin, namun implementasinya kini telah meluas ke berbagai bidang (Nanda Sari & Gelar, 2024).

Karakteristik Utama

1. Desentralisasi: Sistem *blockchain* tidak bergantung pada satu entitas pusat, melainkan dikelola oleh jaringan node.
2. Transparansi: Transaksi dapat dilihat dan diverifikasi oleh semua node dalam jaringan.
3. *Immutability*: Data yang telah dicatat di *blockchain* tidak dapat diubah, memastikan keandalan data.
4. *Traceability*: Pengguna dapat melacak riwayat transaksi di *blockchain*.
5. *Non-Repudiation*: Transaksi yang telah ditandatangani secara digital tidak dapat disangkal.



Gambar 2.1 Ilustrasi *blockchain*
Sumber: Penulis

Komponen Teknologi *Blockchain*

1. Fungsi *Hash*: *Blockchain* menggunakan fungsi *hash*, seperti SHA-256, Keccak-256, Scrypt, X11, dan berbagai fungsi hashing lainnya yang berguna untuk memastikan integritas data. Fungsi *hash* menghasilkan nilai unik untuk setiap *input*

data sehingga setiap perubahan sekecil apa pun akan menghasilkan *hash* yang berbeda

2. Jaringan *Peer-to-peer* (P2P): Jaringan P2P memungkinkan distribusi data ke semua node dalam jaringan, memastikan tidak ada satu pun titik kegagalan dan meningkatkan keamanan data.

Jenis-Jenis *Blockchain*

1. *Public Blockchain*

   Public *blockchain* adalah jenis *blockchain* yang dapat diakses oleh siapa saja. Jaringan ini bersifat terbuka dan tidak memerlukan izin untuk bergabung atau berpartisipasi, contoh:

   1. **Bitcoin**

      Digunakan untuk mencatat transaksi *cryptocurrency* secara desentralisasi tanpa otoritas pusat.

   2. **Ethereum**

      Sebuah *platform* untuk menjalankan *smart contract* dan aplikasi desentralisasi (dApps), seperti aplikasi keuangan terdesentralisasi (DeFi).

   3. **Solana**

      Public blockchain yang dirancang untuk transaksi cepat dengan biaya rendah, banyak digunakan dalam aplikasi DeFi dan NFT.

2. *Private Blockchain*

   Private blockchain hanya dapat diakses oleh node yang diotorisasi oleh entitas pengelola. Biasanya digunakan oleh organisasi atau perusahaan untuk aplikasi internal, contoh:

   1. **Hyperledger Fabric**

      Blockchain yang dikembangkan oleh Linux Foundation, digunakan oleh perusahaan seperti IBM untuk manajemen rantai pasok.

2. **Corda**

   Platform *blockchain* yang dirancang untuk sektor keuangan, memungkinkan bank untuk mencatat transaksi antar institusi secara aman.

3. **Ripple (XRP)**

   Fokus pada sistem pembayaran antarbank yang cepat dan aman.

3. *Permissioned Blockchain*:

   Permissioned *blockchain* adalah kombinasi antara *public* dan *private blockchain*. Dalam jaringan ini, akses tertentu diberikan kepada pihak-pihak tertentu berdasarkan izin, tetapi data transaksi dapat tetap transparan untuk pihak yang berwenang. contoh:

   1. **IBM Food Trust**

      Platform blockchain untuk meningkatkan transparansi dalam rantai pasok makanan. Digunakan oleh perusahaan seperti Walmart dan Nestlé untuk melacak perjalanan makanan dari produsen hingga konsumen.

   2. **Quorum**

      Blockchain berbasis Ethereum yang digunakan oleh JPMorgan untuk aplikasi keuangan yang membutuhkan privasi data tertentu.

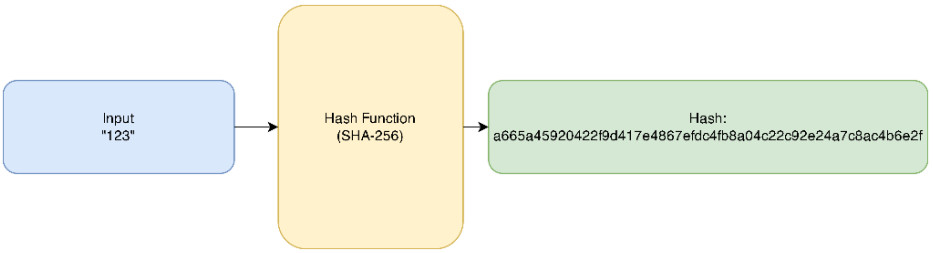
   3. **VeChain**

      Digunakan untuk meningkatkan transparansi dalam logistik dan rantai pasok, misalnya pelacakan produk seperti *wine* dan barang mewah.

2.2.2 Fungsi *Hash*

Fungsi *hash* adalah algoritma yang menerima *input* berupa data dengan panjang yang tidak tetap dan menghasilkan *output* berupa nilai dengan panjang tetap, yang disebut *hash value*, *digital*

fingerprint, atau *message digest*, *Hash value* bertindak sebagai representasi unik dari data *input*. (Abdoun et al., 2020)



Gambar 2.2. Contoh fungsi *hash*.
Sumber: penulis

Sifat Utama *Hash* Function

1. ***Ease of Computation***:
   a. Proses *hashing* harus cepat dan efisien, memungkinkan sistem untuk menghasilkan *hash value* dalam waktu singkat bahkan untuk *input* data yang besar.
   b. Sifat ini penting untuk aplikasi *blockchain*, di mana ratusan transaksi perlu di-*hash* setiap detik.
2. ***Compression***:
   a. *Hash function* mengubah data dengan panjang variabel menjadi *output* dengan panjang tetap.
   b. Contoh:
      Input: "Hello, *Blockchain*!" → *Output* (SHA-256): `e3b0c44298fc1c149afbf4c8996fb924...`
      Panjang *output* selalu tetap (256-bit), terlepas dari ukuran *input*.
3. ***Preimage Resistance* (Sifat Satu Arah)**:
   a. Sulit untuk menemukan *input* asli hanya dengan mengetahui *hash value*-nya.

b. Contoh: Jika diberikan *hash* `e3b0c...`, hampir mustahil untuk menebak bahwa *input* aslinya adalah "Hello, *Blockchain*!".

c. Sifat ini memastikan keamanan data yang di-*hash*, karena data asli tidak dapat diperoleh kembali.

4. ***Second Preimage Resistance***:

a. Tidak mungkin menemukan dua *input* berbeda yang menghasilkan *hash value* identik.

b. Contoh:

Input A: "Data 1" menghasilkan *hash* `h1`.

Sangat sulit menemukan *input* B (berbeda dari A) yang juga menghasilkan `h1`.

c. Hal ini melindungi data dari manipulasi, misalnya dalam tanda tangan digital.

5. ***Collision Resistance****:*

a. Tidak mungkin menemukan dua *input* berbeda yang menghasilkan *hash value* identik.

b. Contoh:

Input A: "*Blockchain*"

Input B: "Block chain"

Hash value dari keduanya harus unik.

c. Sifat ini memastikan integritas data, karena setiap perubahan pada *input* akan menghasilkan *hash value* yang berbeda.

6. ***Avalanche Effect****:*

a. Perubahan kecil pada *input* menyebabkan perubahan besar pada *hash value*.

b. Contoh:

Input A: "Hello, *Blockchain*!"

Hash: `e3b0c44298fc1c149afbf4c8996fb924...`

Input B: "hello, *Blockchain*!" (Hanya huruf pertama berubah kecil)

Hash: `a9f2b8821e382c80e54f302e58f12d20...`

c. Sifat ini membuat *hash* sangat sensitif terhadap perubahan, penting untuk memastikan integritas data.

7. Peran *Hash* dalam *Blockchain*:
   a. Identifikasi Blok: Setiap blok di *blockchain* memiliki *hash* unik berdasarkan data di dalamnya.
   b. Konsistensi Data: Jika data dalam blok diubah, *hash* akan berubah, memutus rantai *blockchain*.
   c. Keamanan: Fungsi *hash* bersifat satu arah, sehingga tidak mungkin memulihkan data asli dari *hash*.

2.2.3 Ethereum *Blockchain*

Ethereum adalah platform *blockchain* terdesentralisasi yang memungkinkan pengembangan dan eksekusi kontrak pintar (*smart contracts*) serta aplikasi terdesentralisasi (*decentralized applications* atau DApps). Diperkenalkan pada tahun 2015, Ethereum memperluas konsep *blockchain* dengan menyediakan mesin virtual turing lengkap, yang dikenal sebagai Ethereum Virtual Machine (EVM), memungkinkan pengembang untuk membuat dan menjalankan aplikasi yang kompleks di jaringan terdistribusi. (Buterin, 2015)

Gambar 2.3. Logo Ethereum
Sumber: coinmarketcap.com

Karakteristik Utama Ethereum:

1. Ethereum *Virtual Machine* (EVM):

a. EVM adalah mesin virtual turing lengkap yang dirancang untuk menjalankan *smart contract*s.
b. EVM memungkinkan pengembang untuk menulis kode dalam bahasa pemrograman seperti Solidity dan menjalankannya di jaringan Ethereum yang terdistribusi.

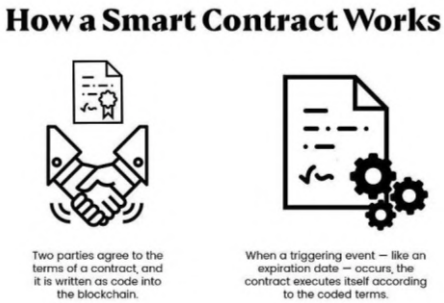
2. Kontrak Pintar (*Smart contract*s):
    a. Ethereum memungkinkan eksekusi kontrak otomatis yang berjalan sesuai dengan kondisi yang telah ditentukan.
    b. Hal ini mengurangi ketergantungan pada perantara, meningkatkan efisiensi, dan menurunkan biaya transaksi.
3. *Proof-of-Stake* (PoS):
    a. Setelah pembaruan besar "*The Merge*" pada tahun 2022, Ethereum beralih dari *Proof-of-Work* (PoW) ke *Proof-of-Stake* (PoS).
    b. Konsensus PoS secara signifikan mengurangi konsumsi energi, membuat Ethereum lebih ramah lingkungan dan efisien. (Kapengut & Mizrach, 2022)
4. Desentralisasi:
    a. Ethereum menggunakan jaringan *peer-to-peer* (P2P) global untuk memastikan tidak ada kontrol terpusat, sehingga meningkatkan keamanan dan transparansi sistem.

2.2.4 Kontrak Pintar (*Smart contract*)

Smart contract adalah program komputer yang berjalan di atas platform *blockchain* Ethereum yang dirancang secara otomatis mengeksekusi dan mengelola perjanjian antara pihak-pihak tanpa memerlukan perantara. Konsep ini pertama kali diperkenalkan oleh Nick Szabo pada tahun 1994 dalam tulisannya berjudul "*Smart contracts: Building Blocks for Digital Markets*". Dalam artikel tersebut, Szabo menjelaskan konsep kontrak pintar (*smart contract*) sebagai protokol transaksi yang secara otomatis mengeksekusi

ketentuan kontrak berdasarkan logika pemrograman tertentu. Gagasan ini menjadi pondasi untuk pengembangan kontrak pintar modern di platform *blockchain* seperti Ethereum.

Ethereum memperluas konsep ini dengan menyediakan Ethereum Virtual Machine (EVM) yang memungkinkan pengembang menjalankan aplikasi kompleks secara terdesentralisasi di jaringan *blockchain* dengan menggunakan bahasa pemrograman solidity. Dalam pembuatannya, *smart contract* di Ethereum sering digunakan untuk berbagai keperluan, seperti *e-voting*, klaim asuransi, dan manajemen aset digital (Dzulfikar & Susanto, 2020)



Gambar 2.4. Ilustrasi sederhana bagaimana kontrak pintar bekerja
Sumber: https://www.verytechnology.com/iot-insights/how-do-ethereum-smart-contracts-*work*-its-deceptively-simple

Fungsi Utama *Smart contract* di Ethereum:

a. Otomatisasi: Menjalankan instruksi yang telah diprogram secara otomatis ketika kondisi tertentu terpenuhi.
b. Transparansi: Semua transaksi dan kode kontrak dapat dilihat oleh publik, memastikan kepercayaan antara pihak-pihak yang terlibat.
c. Keamanan: Setelah diterapkan, *smart contract* tidak dapat diubah, mengurangi risiko manipulasi.
d. Efisiensi: Menghilangkan kebutuhan akan perantara, sehingga mengurangi biaya dan waktu yang diperlukan untuk menyelesaikan transaksi.

2.2.5 *Wallet*

Dompet Digital (*Digital Wallet*) adalah aplikasi perangkat lunak yang memungkinkan pengguna menyimpan, mengelola, dan bertransaksi dengan aset digital, seperti mata uang kripto, melalui perangkat elektronik. Dalam konteks teknologi *blockchain*, dompet digital berfungsi sebagai antarmuka antara pengguna dan jaringan *blockchain*, memungkinkan akses dan pengelolaan aset kripto secara aman (di Angelo & Salzer, 2020).

Klasifikasi Dompet Digital

1. Hot *Wallet*
   a. Dompet digital yang terhubung langsung dengan internet, seperti aplikasi seluler atau desktop.
   b. Contoh: Metamask, Trust *Wallet*.
   c. Keunggulan: Mudah digunakan untuk transaksi cepat.
   d. Kekurangan: Lebih rentan terhadap serangan dunia maya seperti *hacking*.
2. *Cold Wallet*
   a. Dompet digital yang tidak terhubung ke internet, memberikan lapisan keamanan tambahan.
   b. Contoh *cold wallet* Adalah *Hardwallet*, yaitu dompet perangkat keras, seperti Ledger Nano S atau Trezor, yang menyimpan kunci pribadi secara *offline*. Memiliki keunggulan, yaitu tingkat keamanan sangat tinggi karena tidak rentan terhadap malware dan memiliki kekurangan, yaitu membutuhkan perangkat fisik yang dapat hilang atau rusak.
   c. Keamanan dan Penggunaan *Cold Wallet*

Cold *wallet*, khususnya *hardwallet*, dirancang untuk menyimpan aset digital dalam kondisi aman dengan mengisolasi kunci pribadi dari potensi ancaman online (BUCERZAN & BEJAN, 2020). Bejan dan Bucerzan (2020) mencatat bahwa penggunaan

hardwallet adalah solusi ideal bagi investor yang ingin menyimpan aset dalam jangka panjang karena:

a. Perlindungan Tinggi: Hardwallet mengurangi risiko serangan *phishing*, *malware*, atau *hacking* karena tidak selalu terhubung ke internet.
b. Autentikasi Tambahan: Banyak *hardwallet* dilengkapi dengan fitur keamanan tambahan seperti PIN atau autentikasi dua faktor.

Dalam teknologi *blockchain*, dompet digital (baik *hot* maupun *cold wallet*) memainkan peran penting dalam:

a. Menyimpan Kunci Pribadi: Pengguna memerlukan kunci pribadi untuk menandatangani transaksi di jaringan *blockchain*.
b. Mengakses DApps: Banyak dompet *hot wallet*, seperti Metamask, digunakan untuk mengakses aplikasi terdesentralisasi (DApps).
c. Keamanan Aset Digital: *Cold wallet* digunakan untuk menyimpan aset kripto dengan aman tanpa eksposur langsung ke internet.

2.2.6 Metamask

Gambar 2.5. Metamask

Sumber: https://metamask.io/

MetaMask adalah ekstensi peramban dan aplikasi seluler yang berfungsi sebagai dompet digital (*digital wallet*) untuk mengelola aset kripto, khususnya yang berbasis pada jaringan Ethereum (Dr. Poornima G. Naik & Dr. Girish R. Naik, 2023). MetaMask memungkinkan pengguna untuk berinteraksi dengan aplikasi terdesentralisasi (*decentralized applications* atau DApps) langsung

dari peramban mereka, tanpa perlu menjalankan node Ethereum penuh (Winter et al., 2021).

Fungsi dan Fitur Utama MetaMask

a. Manajemen Kunci Pribadi

   MetaMask menyimpan kunci pribadi pengguna secara lokal pada perangkat, memastikan bahwa hanya pengguna yang memiliki kontrol penuh atas aset mereka.

b. Interaksi dengan DApps

   Dengan MetaMask, pengguna dapat dengan mudah terhubung dan berinteraksi dengan berbagai DApps di ekosistem Ethereum, seperti *platform* DeFi, NFT, dan lain-lain.

c. Tampilan Mudah Dipahami

   MetaMask menyediakan antarmuka yang ramah pengguna untuk mengirim dan menerima transaksi, dengan fitur konfirmasi dan penyesuaian biaya gas.

2.2.7 Hardhat

Gambar 2.6. Hard Hat
Sumber: https://hardhat.org/

Hardhat adalah lingkungan pengembangan Ethereum yang dirancang untuk memfasilitasi pembuatan, pengujian, dan penerapan kontrak pintar (*smart contract*s) di jaringan Ethereum. Hardhat menyediakan alat-alat canggih untuk mempermudah pengembangan

dan memastikan keandalan aplikasi terdesentralisasi (Dapps). (Shubham et al., 2024).Hardhat memiliki fungsi, yaitu:

1. *Testing* dan *Debugging*

   Hardhat memungkinkan pengembang untuk melakukan pengujian secara menyeluruh terhadap kontrak pintar, termasuk mengidentifikasi dan memperbaiki kerentanan kode.

2. Deployment

   Hardhat menyediakan alat untuk menyebarkan *smart contract* ke jaringan Ethereum, baik *testnet* maupun *mainnet*, dengan konfigurasi yang fleksibel.

3. Integrasi dengan Alat Lain

   Hardhat mendukung berbagai plugin seperti Ethers.js untuk berinteraksi dengan *blockchain* dan Solidity Coverage untuk mengukur cakupan pengujian. Menurut Dr. Poornima G. Naik, Dr. Girish R. Naik (2023), integrasi ini membuat Hardhat menjadi alat yang sangat serbaguna.

4. Simulasi Jaringan

   Fitur Hardhat Network memungkinkan simulasi jaringan Ethereum secara lokal, mempermudah pengujian transaksi tanpa memerlukan koneksi ke jaringan publik.

2.2.8 Solidity

Gambar 2.7. Logo bahasa pemrograman solidity

Sumber: https://www.codepolitan.com/blog/mengenal-solidity-untuk-web-30/

Solidity adalah bahasa pemrograman tingkat tinggi yang dirancang khusus untuk pengembangan kontrak pintar (*smart contracts*) di platform Ethereum. Bahasa ini berparadigma *statically-typed*, mendukung konsep *object-oriented*, dan sintaksnya mirip dengan JavaScript, menjadikannya familiar bagi banyak pengembang. Solidity memungkinkan pengembang menulis logika bisnis yang dieksekusi secara otomatis di *blockchain* Ethereum, memastikan transparansi, keamanan, dan integritas dalam berbagai aplikasi terdesentralisasi (dApps). (Antonino & Roscoe, 2020)

Sejak diperkenalkan, Solidity telah menjadi standar de facto untuk pengembangan kontrak pintar di Ethereum. Bahasa ini terus berkembang dengan penambahan fitur-fitur baru yang meningkatkan kemampuannya dalam menangani berbagai skenario kompleks dalam pengembangan dApps. Beberapa fitur penting yang ditawarkan oleh Solidity meliputi:

a. Tipe Data Kompleks: Mendukung struktur data seperti array, mapping, dan struct yang memungkinkan pengembangan logika kontrak yang kompleks.
b. Pewarisan (*Inheritance*): Memungkinkan kontrak untuk mewarisi properti dan fungsi dari kontrak lain, mendukung pengembangan modular dan reusable.
c. Modifier: Fungsi khusus yang digunakan untuk mengubah perilaku fungsi lain, sering digunakan untuk kontrol akses dan validasi kondisi sebelum eksekusi fungsi.
d. *Event*: Mekanisme untuk memicu log yang dapat didengarkan oleh aplikasi di luar *blockchain*, memungkinkan interaksi antara kontrak pintar dan antarmuka pengguna.

Meskipun menawarkan banyak fitur, pengembangan dengan Solidity tidak lepas dari tantangan, terutama terkait keamanan kontrak pintar.

Kesalahan dalam kode dapat menyebabkan kerentanan yang dieksploitasi, mengakibatkan kerugian finansial yang signifikan.

2.2.9 Sepolia

Gambar 2.8. *Testnet* Sepolia
Sumber: https://developers.moralis.com/sepolia-*testnet*-guide-what-is-the-sepolia-*testnet*/

Sepolia adalah jaringan uji (*testnet*) Ethereum yang dirancang untuk memberikan lingkungan pengujian yang stabil bagi pengembang sebelum meluncurkan aplikasi di jaringan utama (*mainnet*). Sepolia mulai diperkenalkan pada tahun 2021 dan secara bertahap menggantikan *testnet* seperti Ropsten dan Goerli sebagai jaringan uji default. Sepolia menggunakan mekanisme konsensus *Proof-of-Stake* (PoS) untuk mencerminkan kondisi *mainnet* Ethereum, memungkinkan pengembang untuk mensimulasikan berbagai skenario dalam pengembangan aplikasi terdesentralisasi (*DApps*) dan kontrak pintar dengan lebih realistis (Ethereum Foundation, 2024).

Naik Poornima G. Naik & Girish R. Naik (2023) dalam buku *Blockchain Beyond Basics*, *testnet* seperti Sepolia memainkan peran penting dalam ekosistem Ethereum dengan menyediakan lingkungan uji bebas risiko bagi pengembang. Sepolia mendukung pengujian berbagai skenario kompleks, seperti pengujian kontrak pintar, transaksi token, dan integrasi Dapps (Dr. Poornima G. Naik & Dr. Girish R. Naik, 2023). Sepolia memiliki fitur utama, yaitu:

1. Sepolia menggunakan PoS yang mendukung efisiensi energi, memungkinkan pengujian dalam kondisi jaringan yang lebih

mendekati *mainnet* Ethereum. Menurut Naik Poornima G. Naik & Girish R. Naik (2023), fitur ini memastikan pengembang dapat menguji aplikasi mereka dengan performa jaringan yang hampir identik dengan *mainnet*.

2. Sepolia diamankan oleh validator yang terkontrol, memberikan stabilitas tambahan selama proses pengujian dan mengurangi risiko serangan jaringan.
3. Untuk mendukung pengembangan, Sepolia menyediakan token uji yang dapat diakses melalui layanan faucet. Token ini memungkinkan pengembang menguji transaksi dan kontrak pintar tanpa risiko finansial.
4. Sepolia mendukung berbagai alat pengembangan Ethereum seperti MetaMask, Hardhat, dan Remix, sehingga memudahkan integrasi ke dalam alur kerja pengembangan. Menurut Naik Naik Poornima G. Naik & Girish R. Naik (2023), alat-alat ini mempercepat pengujian dan debugging aplikasi sebelum implementasi di *mainnet*.

2.2.10 EtherScan

Gambar 2.9. Etherscan

Sumber: https://etherscan.io/

Etherscan adalah platform eksplorasi *blockchain* Ethereum yang memungkinkan pengguna untuk mencari, melacak, dan menganalisis berbagai transaksi serta data terkait di jaringan Ethereum. Sebagai alat analitik, Etherscan membantu pengembang dan pengguna memahami pola transaksi, kontrak pintar, dan metrik penting lainnya (Dr. Poornima G. Naik & Dr. Girish R. Naik, 2023).

Menurut penelitian Gupta et al. (2023), Etherscan memberikan kemampuan untuk memantau aktivitas transaksi, termasuk transfer ETH, panggilan kontrak, dan pembuatan kontrak. Hal ini sangat relevan untuk analisis transaksi di jaringan Ethereum, terutama untuk memahami pola-pola interaksi antara pengguna dan kontrak pintar. Studi ini mencatat bahwa Etherscan menjadi alat penting dalam mengidentifikasi detail spesifik alamat, seperti jumlah transaksi, jenis kontrak, dan volume Ether yang ditransfer (Gupta et al., 2023).

Manfaat Utama Etherscan:

1. Transparansi Data: Etherscan memfasilitasi akses publik terhadap transaksi yang terjadi di jaringan Ethereum, mendukung prinsip transparansi *blockchain*.
2. Pemantauan Transaksi: Pengguna dapat melacak riwayat transaksi dan memastikan validitas data di jaringan.
3. Analitik Kontrak Pintar: Memberikan informasi tentang jumlah kontrak yang dibuat, transaksi terkait, serta pola penggunaan kontrak pintar.
4. Dukungan untuk Pengembang: Etherscan menyediakan API untuk integrasi ke dalam aplikasi lain, memungkinkan pengembang untuk menggunakan data *blockchain* secara lebih luas.

Dengan kemampuan ini, Etherscan tidak hanya menjadi alat eksplorasi tetapi juga berkontribusi pada riset tentang Ethereum dan teknologi *blockchain* secara umum. Penelitian lebih lanjut dapat memanfaatkan data dari Etherscan untuk mendukung pengembangan aplikasi terdesentralisasi yang efisien dan aman (Poornima Naik Girish R Naik, 2023).

2.2.11 Alchemy

Gambar 2.10. Logo Alchemy

Sumber: https://www.alchemy.com/

Alchemy adalah platform infrastruktur *blockchain* yang dirancang untuk mempermudah pengembangan, pengujian, dan penerapan aplikasi terdesentralisasi (decentralized applications atau DApps). Menurut Dr. Poornima G. Naik, Dr. Girish R. Naik (2023) dalam buku *Blockchain Beyond Basics*, Alchemy adalah salah satu penyedia layanan *Blockchain*-as-a-Service (BaaS) terkemuka yang mempermudah pengembang untuk berinteraksi dengan jaringan *blockchain* tanpa harus menjalankan node Ethereum secara langsung. Sebagai penyedia layanan API dan alat pengembang, Alchemy memungkinkan interaksi yang efektif dengan *blockchain* Ethereum tanpa memerlukan pengelolaan node *blockchain* secara manual. Platform ini sering digunakan dalam ekosistem Ethereum oleh pengembang DApps dan kontrak pintar. Alchemy memiliki fitur utama, yaitu:

1. API *Blockchain* yang Canggih
    a. Alchemy menyediakan API untuk mempermudah akses data *blockchain* secara real-time, termasuk transaksi, log kontrak pintar, dan informasi akun.
    b. API ini dirancang untuk menangani permintaan skala besar dengan kecepatan tinggi, memastikan pengalaman pengembang yang optimal.
2. Alchemy Supernode
    a. Supernode adalah layanan inti yang menyediakan akses ke node Ethereum dengan keandalan tinggi.

b. Layanan ini mencakup caching data, manajemen query, dan optimasi jaringan untuk meningkatkan efisiensi.

3. Dashboard Analitik
    a. Alchemy menawarkan antarmuka visual yang intuitif untuk memantau aktivitas jaringan, memeriksa penggunaan API, dan menganalisis log transaksi.
    b. Fitur ini memungkinkan pengembang untuk mengidentifikasi masalah dan meningkatkan performa aplikasi mereka.
4. Fitur Pengembangan Lanjutan
    a. Error Insights: Alat debugging yang membantu pengembang mengidentifikasi kesalahan dalam kontrak pintar atau transaksi.
    b. Trace API: Memungkinkan analisis lebih mendalam terhadap transaksi *blockchain* untuk debugging tingkat lanjut.
5. Integrasi Multi-Platform
    a. Alchemy mendukung integrasi dengan berbagai alat pengembangan seperti Truffle, Hardhat, dan Remix, menjadikannya pilihan fleksibel untuk pengembang DApps.

2.2.12 NodeJS (Javascript)

Gambar 2.11. NodeJS (Javascript)
Sumber: https://nodejs.org/

Node.js adalah platform berbasis JavaScript runtime yang berjalan di luar browser, memungkinkan pengembang untuk membangun aplikasi *server-side* atau *backend* menggunakan JavaScript. Dikembangkan oleh Ryan Dahl pada tahun 2009, Node.js berbasis pada mesin JavaScript V8 milik Google Chrome, yang membuatnya sangat cepat dan efisien dalam mengeksekusi kode JavaScript.

Node.js dirancang untuk membangun aplikasi yang bersifat *non-blocking*, *event-driven*, dan sangat cocok untuk menangani banyak permintaan secara bersamaan (*concurrent requests*) tanpa mengorbankan performa. Node.js sering digunakan untuk membangun aplikasi web *real-time*, API, dan sistem berbasis jaringan. Dengan ekosistem npm (*Node Package Manager*) yang besar, pengembang dapat dengan mudah mengintegrasikan pustaka dan modul tambahan untuk mempercepat pengembangan aplikasi. Selain itu, Node.js mendukung skala besar, menjadikannya pilihan populer untuk perusahaan dan *startup* dalam membangun aplikasi yang cepat, efisien, dan modern.

Menurut Naik dan Naik (2023), Node.js memainkan peran penting dalam membangun aplikasi yang terintegrasi dengan *blockchain*, khususnya Ethereum (Dr. Poornima G. Naik & Dr. Girish R. Naik, 2023), melalui pustaka-pustaka berikut:

1. Web3.js:

   Digunakan untuk berinteraksi dengan jaringan Ethereum, seperti membuat transaksi, membaca data kontrak pintar, dan memverifikasi blok.

2. Ethers.js:

   Pustaka ringan untuk mempermudah pengembang dalam berinteraksi dengan kontrak pintar dan API Ethereum.

3. Truffle dan Hardhat:

Node.js digunakan sebagai basis untuk menjalankan alat pengembangan seperti Truffle dan Hardhat yang dirancang untuk pengujian dan penyebaran kontrak pintar di Ethereum.

4. Backend untuk dApps:

   Node.js sering digunakan untuk membangun API *backend* yang menjembatani komunikasi antara aplikasi frontend (dApps) dan jaringan *blockchain*.

Node.js juga sering digunakan untuk membangun API *backend* yang menghubungkan aplikasi terdesentralisasi (dApps) dengan jaringan *blockchain*. Kecepatan dan efisiensi Node.js dalam menangani permintaan asinkron membuatnya sangat cocok untuk aplikasi *blockchain* yang membutuhkan komunikasi real-time dengan jaringan.

2.2.13 Ether.js

Ethers.js adalah pustaka JavaScript ringan dan modular yang dirancang untuk berinteraksi dengan jaringan *blockchain* Ethereum. Pustaka ini menyediakan antarmuka yang sederhana namun kuat untuk mengelola transaksi, membaca data dari *blockchain*, dan berinteraksi dengan kontrak pintar (*smart contracts*). **Ethers.js** banyak digunakan dalam pengembangan aplikasi terdesentralisasi (*DApps*) karena fleksibilitas dan kemudahan penggunaannya. (Dr. Poornima G. Naik & Dr. Girish R. Naik, 2023)

Fitur Utama Ethers.js

1. Pengelolaan Kunci Pribadi dan Dompet Digital:
   a. Ethers.js mendukung berbagai format kunci pribadi dan integrasi dengan dompet digital seperti MetaMask.
   b. Pustaka ini memungkinkan pengguna untuk membuat, menyimpan, dan memulihkan dompet digital dengan aman.
2. Interaksi dengan Kontrak Pintar

Dengan Ethers.js, pengembang dapat memanggil fungsi dalam kontrak pintar, mengirim transaksi, dan membaca data dari kontrak pintar menggunakan ABI (Application Binary Interface).

3. Dukungan untuk JSON-RPC

   Pustaka ini memungkinkan komunikasi langsung dengan node Ethereum melalui protokol JSON-RPC, memfasilitasi pengambilan data blok, transaksi, dan log kontrak.

4. Pencatatan dan Pemantauan Transaksi

   Ethers.js menyediakan API yang memungkinkan pengembang untuk melacak status transaksi, memverifikasi blok, dan memantau log yang relevan.

5. Dukungan *Testnet* dan *Mainnet*

   Ethers.js kompatibel dengan jaringan utama (*mainnet*) Ethereum dan *testnet* seperti Sepolia, Goerli, dan lainnya, menjadikannya fleksibel untuk pengembangan dan pengujian aplikasi.

6. Dokumentasi yang Komprehensif

   Ethers.js menawarkan dokumentasi yang jelas dan lengkap, memudahkan pengembang untuk memulai proyek dan menyelesaikan masalah teknis.

Ethers.js memainkan peran penting dalam pengembangan aplikasi berbasis *blockchain* Ethereum karena:

1. Interaksi Kontrak Pintar: Memfasilitasi eksekusi fungsi kontrak pintar dan pemrosesan transaksi.
2. Pemantauan Jaringan: Memberikan alat analitik untuk memantau aktivitas jaringan *blockchain*.
3. Integrasi API: Menghubungkan backend aplikasi dengan node Ethereum untuk pengelolaan data secara efisien.

2.2.14 MongoDB

Gambar 2.12 MongoDB
Sumber: https://www.mongodb.com/

MongoDB adalah sistem basis data NoSQL yang menggunakan model dokumen untuk menyimpan data dalam format fleksibel berbasis BSON (Binary JSON). Sistem ini dirancang untuk mengelola data dalam skala besar dengan fokus pada performa, skalabilitas, dan fleksibilitas. MongoDB memungkinkan pengembang untuk menyimpan data tanpa memerlukan skema tetap, sehingga cocok untuk aplikasi modern yang memerlukan adaptasi cepat terhadap perubahan struktur data (Eyada et al., 2020).

Keunggulan MongoDB:

1. Kinerja Tinggi

   MongoDB memiliki performa lebih baik dibandingkan MySQL dalam beberapa skenario, terutama dalam hal latensi saat menangani volume data yang besar.
2. Kompatibilitas dengan Cloud: MongoDB bekerja dengan baik dalam lingkungan cloud, memungkinkan pengolahan data yang cepat pada berbagai spesifikasi perangkat keras.
3. Kemampuan Real-Time: MongoDB mampu memproses dan menyimpan data IoT secara real-time, yang sangat penting untuk aplikasi modern yang membutuhkan respons cepat (Eyada et al., 2020).

Alasan Pemilihan MongoDB untuk Penelitian:

1. Latensi yang Lebih Rendah:

   Dalam eksperimen, MongoDB menunjukkan latensi yang lebih rendah dibandingkan dengan MySQL, bahkan ketika jumlah sensor atau beban kerja meningkat secara signifikan (Eyada et al.,

2020). Hal ini menjadikan MongoDB pilihan yang efisien untuk aplikasi berbasis IoT.

2. Kemampuan Penanganan Data Heterogen
   MongoDB unggul dalam menyimpan data yang beragam tanpa memerlukan perubahan besar pada struktur basis data.
3. Efisiensi Sumber Daya
   MongoDB tetap dapat memberikan performa yang baik pada perangkat dengan spesifikasi rendah, menjadikannya pilihan yang hemat sumber daya untuk implementasi berbasis IoT (Eyada et al., 2020).

2.2.15 Pembangkit Listrik Tenaga Surya (PLTS)

Pembangkit Listrik Tenaga Surya (PLTS) adalah sistem yang mengonversi energi matahari menjadi energi listrik menggunakan modul fotovoltaik (PV). PLTS dapat diimplementasikan dalam berbagai skala, mulai dari instalasi kecil pada atap bangunan (PLTS atap) hingga pembangkit listrik skala besar.(Wibowo et al., 2024)

Komponen Utama PLTS:

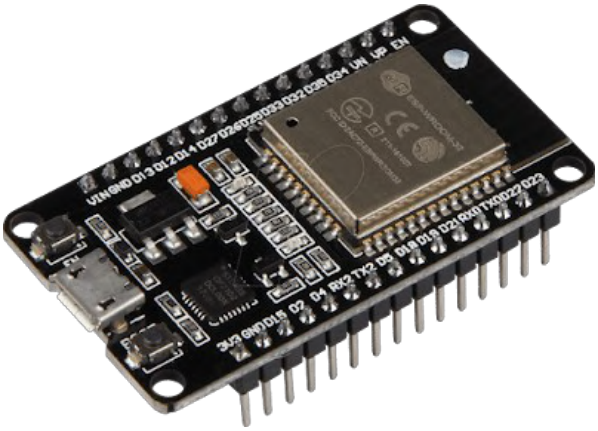
1. Modul Surya (Panel PV): Mengubah sinar matahari menjadi listrik DC.
2. Inverter: Mengonversi listrik DC menjadi AC yang sesuai untuk peralatan listrik.
3. Sistem Mounting: Struktur penyangga panel surya.
4. Sistem Penyimpanan Energi (opsional): Seperti baterai untuk menyimpan energi surplus.

Keuntungan PLTS:

1. Energi Bersih: Mengurangi emisi gas rumah kaca.
2. Sumber Energi Terbarukan: Matahari sebagai sumber energi yang tidak habis.

3. Pengurangan Biaya Listrik: Menghasilkan listrik sendiri dapat mengurangi tagihan listrik.

2.2.16 ESP32



Gambar 2.13. ESP32

Sumber: https://raharja.ac.id/2021/11/16/mikrokontroler-esp32-2/

ESP32 adalah mikrokontroler berbasis IoT yang sering digunakan untuk aplikasi kontrol dan pemantauan data real-time. Dikembangkan oleh Espressif Systems, ESP32 dilengkapi dengan konektivitas Wi-Fi dan Bluetooth yang memungkinkan komunikasi jarak jauh dengan berbagai perangkat pintar. Dalam penelitian ini, ESP32 memainkan peran penting sebagai pengontrol utama dalam sistem yang memonitor dan mengelola energi dari Pembangkit Listrik Tenaga Surya (PLTS) berbasis *blockchain* Ethereum.

Tabel 2.1. Tabel Spesifikasi ESP32

Atribut	Detail
CPU	Tensilica Xtensa LX6 32bit Dual-Core di 160/240MHz
SRAM	520 KB
FLASH	2Mb (max, 64MB)
Tegangan	2,2V – 2,6 V
Arus Kerja	Rata – rata 80mA
Dapat Diprogram	Ya (C,C++, Python, Lua, dll)
Open Sorce	Ya

Konektivitas	
Wi-Fi	802.11 b/g/n
Bluetooth®	4.2BR/EDR+BLU
UART	3
I/O	
GPIO	32
SPI	4
12C	2
PWM	8
ADC	18 (12-bit)
DAC	2 (8-bit)

Keunggulan ESP32:

1. Konektivitas Wi-Fi dan Bluetooth

   Fitur ini memungkinkan ESP32 untuk terhubung ke jaringan lokal dan platform cloud secara efisien, mendukung komunikasi data antara perangkat keras dan *blockchain* Ethereum.

2. Dukungan Multi-Protokol

   ESP32 mendukung protokol seperti UART, SPI, dan I2C, yang digunakan untuk membaca data dari sensor daya seperti PZEM.

3. Efisiensi Energi

   Dengan mode *deep sleep*, ESP32 dirancang untuk konsumsi daya rendah, menjadikannya ideal untuk aplikasi IoT berbasis energi terbarukan.

4. Kemampuan Real-Time

   Prosesor ESP32 memungkinkan pemrosesan data secara real-time, mendukung pengambilan keputusan berbasis data untuk optimasi transaksi energi.

2.2.17 Arduino IDE

Gambar 2.14. Arduino IDE

Sumber: https://www.arduino.cc/en/*software*

Arduino IDE adalah lingkungan pengembangan terintegrasi (*Integrated Development Environment*) yang dirancang untuk memprogram perangkat berbasis mikrokontroler, seperti Arduino Uno dan ESP32. Arduino IDE menyediakan antarmuka yang ramah pengguna untuk menulis, mengedit, dan mengunggah kode ke perangkat keras melalui koneksi USB atau serial. Dalam penelitian oleh Widyatmika et al. (2021), Arduino IDE digunakan untuk memprogram mikrokontroler ESP32 dan Arduino Uno dalam pengukuran tegangan dan arus menggunakan sensor ZMPT101B dan ACS712 (Putu Ardi Wahyu Widyatmika et al., 2021).

Keunggulan Arduino IDE:

1. Sederhana dan ramah pengguna,
2. Ekosistem *open source*,
3. Kompatibilitas dengan berbagai mikrokontroler, dan
4. Kompilasi dan Unggah Cepat

2.2.18 Solid State Relay (SSR)

Gambar 2.15. Relay SSR

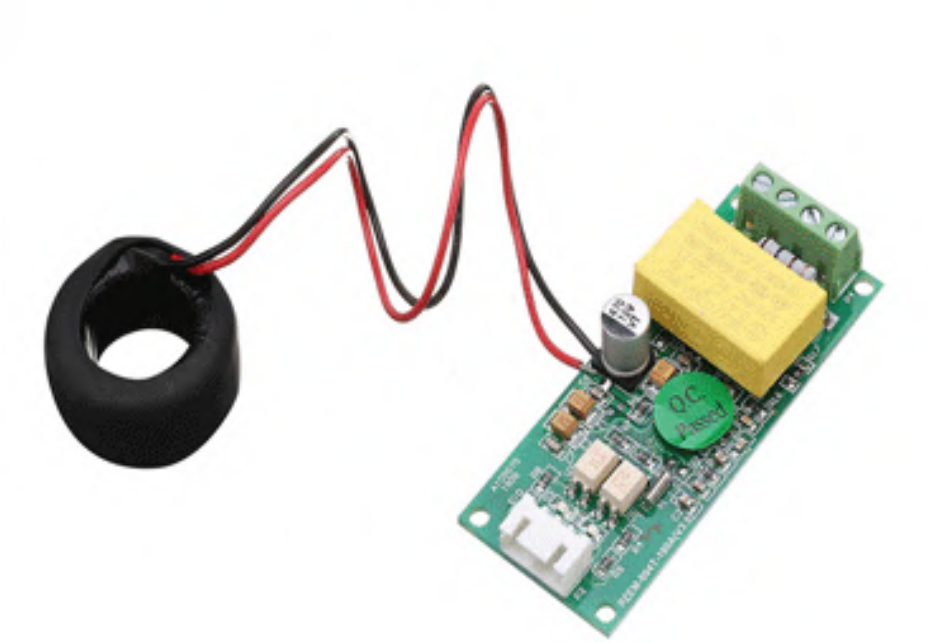
Sumber: https://www.zonemaker.com/product/1546/solid-state-relay-ssr-40dd-40a-5-200vdc-3-32vdc/

Relay SSR (Solid State Relay) adalah perangkat elektronik yang digunakan untuk mengontrol aliran listrik pada beban tanpa melibatkan komponen mekanis. Relay ini menggunakan elemen semikonduktor, seperti thyristor, TRIAC, atau transistor, sebagai pengganti kontak mekanis pada relay tradisional. Dalam penelitian ini, SSR digunakan untuk mengendalikan pemanas dalam tungku listrik dengan mekanisme ON/OFF berbasis sinyal *input* dari mikrokontroler Arduino Uno Atmega 328P(Pratama et al., 2023).

SSR bekerja dengan memanfaatkan isolasi optik antara *input* dan *output* untuk memastikan keamanan dan keandalan operasi. Desain SSR membuat aliran listrik dengan lebih efisien, akurat, dan

minim risiko kerusakan akibat keausan mekanis (Pratama et al., 2023).

2.2.19 PZEM-004t



Gambar 2.16. PZEM-004t
Sumber: https://www.nn-digital.com/blog/2019/07/10/mengenal-pzem-004t-modul-elektronik-untuk-alat-pengukuran-listrik/

PZEM-004T adalah modul sensor multifungsi yang dirancang untuk mengukur berbagai parameter listrik pada sistem AC satu fasa. Modul ini dapat membaca tegangan, arus, daya aktif, faktor daya, energi listrik (kWh), dan frekuensi. Modul ini banyak digunakan dalam aplikasi Internet *of* Things (IoT) untuk pemantauan energi listrik secara kontinu. Modul ini dilengkapi lilitan dengan kumparan trafo arus diameter 3mm yang dapat digunakan untuk mengukur arus maksimal sebesar 100A. Berikut adalah spesifikasi pada sesor PZEM-004 sebagai berikut :

Tabel 2.2. Tabel Spesifikasi sensor PZEM-004T

Parameter	Rentang pembacaan	Resolution	Akurasi
Voltage	80~260V	0.1V	0.5%
Current	0~100A	0.001A	0.5%
Active Power	0~23kW	0.1W	0.5%
Power Factor	0.00~1.00	0.01	1%
Frequency	45Hz~65Hz	0.1Hz	0.5%
Active Energy	0~9999.99kWh	1Wh	0.5%

2.2.20 Regresi Linear

Regresi linear adalah metode analisis statistik yang digunakan untuk memodelkan hubungan antara satu variabel dependen (Y) dengan satu atau lebih variabel independen (X). Dalam penelitian ini, regresi linear digunakan untuk memprediksi kebutuhan energi berdasarkan data historis konsumsi energi (Ida Nabillah & Indra Ranggadara, 2020).

Rumus Umum

$$Y = a + bX \tag{2.1}$$

Y: Variabel dependen (kebutuhan energi).

X: Variabel independen (pola konsumsi energi historis).

$\boldsymbol{a}$: Intersep (nilai Y ketika X = 0).

b: Koefisien regresi (pengaruh X terhadap Y).

Perhitungan Koefisien

1. Koefisien Regresi (b):

$$b = \frac{n\sum X_i Y_i - \sum X_i \sum Y_i}{n\sum X_i^2 - (\sum X_i)^2} \tag{2.2}$$

2. Intersep ($\boldsymbol{a}$)

$$a = \frac{\sum Y_i - b\sum X_i}{n} \tag{2.3}$$

n: Jumlah data.

X_i, Y_i: Data ke-i dari variabel independen dan dependen.

“Halaman ini sengaja dikosongkan”

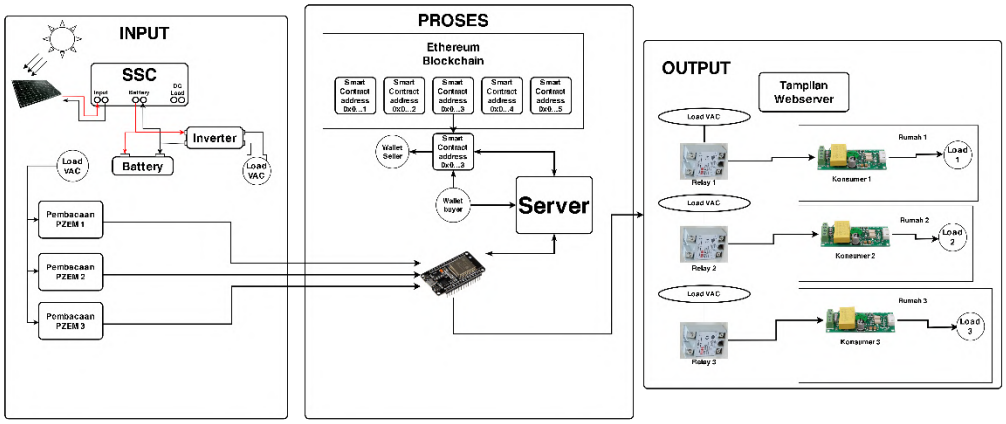
BAB 3
METODOLOGI PENELITIAN

Pada bab 3 ini akan dimulai dengan merumuskan konsep penelitian PLTS terlebih dauhulu kemudian dilakukan serangkaian tahap penelitian yang melibatkan penyusunan flowchart terkait sistem yang akan diimplementasikan pada penelitian tugas akhir ini. Langkah berikutnya melibatkan perencanaan dan desain sub bab yang memiliki tujuan untuk menampilkan rancangan perangkat lunak dan keras yang diterapkan penulis dalam membuat tugas akhir. Setelah itu, disusun jadwal pengerjaan yang menggambarkan *timeline* penulis dalam menyelesaikan penelitian tugas akhir. Selanjutnya, disusun rencana anggaran penelitian dengan tujuan untuk mengidentifikasi estimasi anggaran yang diperlukan oleh penulis dalam pelaksanaan penelitian tugas akhir.

3.1 Konsep Penelitian

Pada Subbab ini akan membahas terkait konsep penelitian yang menjadi landasan utama untuk memahami konteks dan tujuan penelitian.

3.1.1 Diagram Blok Sistem



Gambar 3.1. Diagram blok sistem

Sumber: Penulis

Sistem ini dirancang untuk mengintegrasikan Pembangkit Listrik Tenaga Surya (PLTS) dengan teknologi *blockchain* berbasis Ethereum. Alur sistem terbagi menjadi tiga bagian utama, yaitu *Input*, Proses, dan *Output*.

1. ***Input* (Sumber Energi)**

Pada tahap ini, energi matahari dikonversi menjadi energi listrik oleh modul fotovoltaik (PV). Energi yang dihasilkan merupakan arus searah (DC) dan dikelola oleh *Solar Charge Controller* (SSC) untuk mengoptimalkan proses pengisian daya ke baterai. Selain digunakan secara langsung, energi listrik yang tersimpan di baterai dapat diubah menjadi arus bolak-balik (AC) melalui inverter. Tegangan AC ini selanjutnya digunakan untuk memenuhi kebutuhan listrik beban konsumen.

Setiap konsumen mendapatkan suplai energi yang terhubung melalui jalur relay untuk memastikan kontrol distribusi daya ke masing-masing beban. Sistem ini dirancang untuk melayani hingga tiga konsumen secara simultan, dengan masing-masing jalur beban diawasi oleh sensor daya.

2. Proses (Pengukuran dan *Blockchain*)

a. Pengukuran dan Pengendalian

Data konsumsi daya dari setiap konsumen dikumpulkan menggunakan sensor PZEM. Sensor ini membaca parameter listrik seperti tegangan, arus, dan daya konsumsi masing-masing jalur. Data dari sensor PZEM kemudian dikirimkan ke ESP32, yang bertindak sebagai mikrokontroler utama. ESP32 memproses data dan mengirimkannya melalui koneksi Wi-Fi ke server.

b. Integrasi *Blockchain*

Server bertanggung jawab untuk mengelola data energi yang diterima dan mencatatnya ke dalam *blockchain* Ethereum. Teknologi *blockchain* digunakan untuk mencatat transaksi energi secara transparan, aman, dan tidak dapat diubah. Interaksi dalam *blockchain* dilakukan melalui *smart contract*, yang mengelola transaksi antara *wallet buyer* (pembeli energi) dan *wallet seller* (penjual energi). Setiap transaksi dicatat dengan detail, seperti jumlah energi yang digunakan, waktu transaksi, dan pihak yang terlibat.

Smart contract di *blockchain* Ethereum memastikan bahwa proses pencatatan transaksi energi dilakukan secara otomatis dan efisien. Proses ini menciptakan sistem yang dapat diaudit dan dipercaya oleh semua pihak yang terlibat.

3. *Output* (Konsumen dan Visualisasi)

Energi listrik yang telah dikontrol melalui relay disalurkan ke konsumen. Relay yang digunakan adalah Solid State Relay (SSR), yang memberikan kemampuan pengendalian daya secara presisi untuk setiap jalur beban. Data konsumsi energi setiap konsumen juga dapat dipantau melalui tampilan antarmuka webserver.

Webserver ini dirancang untuk menampilkan data secara real-time, termasuk jumlah energi yang digunakan oleh masing-masing konsumen, status relay, dan log transaksi yang tercatat di *blockchain*. Antarmuka webserver ini memungkinkan pengguna untuk memonitor dan mengelola sistem dengan mudah, memberikan visibilitas penuh terhadap konsumsi energi dan proses transaksinya.

3.1.2 Alur Kerja Sistem

1. Validasi *Wallet* dan Saldo Pembeli
   c. Proses dimulai dengan pengecekan dompet pembeli
   d. Lalu melakukan proses pengecekan saldo Ethereum (ETH) pembeli di dompet mereka.
   e. Jika saldo tidak mencukupi untuk transaksi, sistem akan memberikan error, dan transaksi dihentikan.
2. Konversi Energi ke Ethereum
   a. Setelah saldo pembeli dikonfirmasi mencukupi, jumlah energi yang akan ditransaksikan dikonversi ke nilai Ethereum berdasarkan kurs 1 kWh = 0.000025 ETH. Dengan kurs 1 ETH = Rp 59.584.485,51 saat buku ini ditulis, maka harga per kWh setara Rp 1.489,61. Nilai transaksi akhir ditentukan berdasarkan total energi dikali harga per kWh tersebut.

b. Tahapan ini juga memvalidasi alamat dompet dan private key dari pembeli, jika salah satu informasi tidak benar, transaksi dianggap gagal, dan sistem memberikan error.

3. Pengajuan Transaksi ke Server
   a. Setelah semua informasi valid (saldo cukup, alamat dan private key benar), server mengirimkan permintaan transaksi menggunakan POST Method.

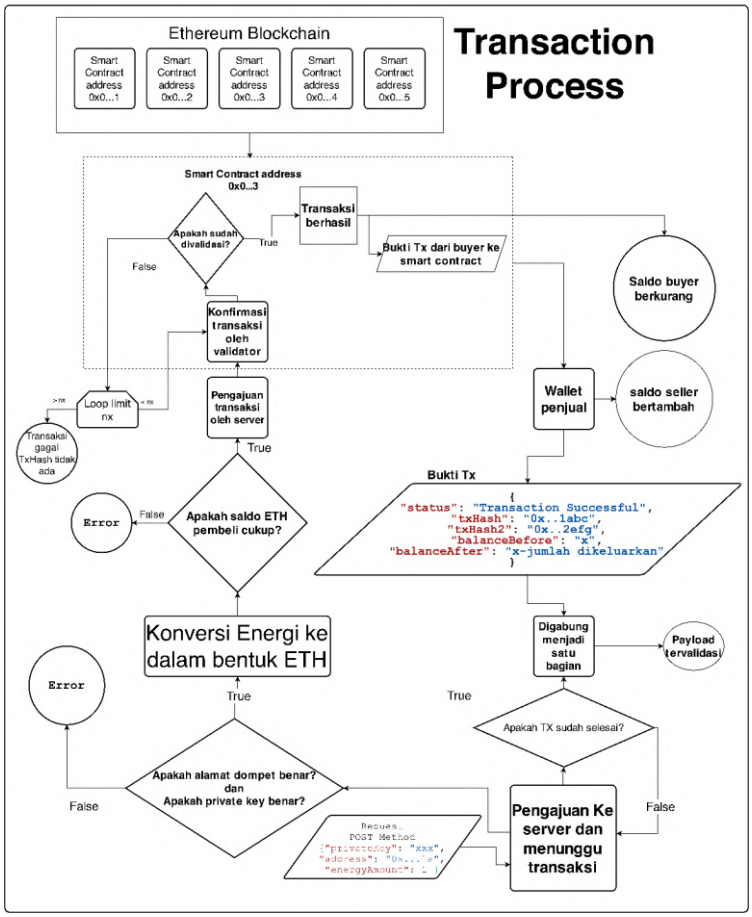
      Payload request meliputi:

      1. privateKey: Kunci pribadi pembeli.
      2. address: Alamat dompet tujuan (penjual).
      3. energyAmount: Jumlah energi yang ditransaksikan dalam Ethereum.

4. Menunggu Validasi dan Eksekusi Transaksi
   b. Setelah transaksi diajukan, server menunggu status konfirmasi dari *blockchain* Ethereum.
   c. Jika transaksi berhasil divalidasi oleh validator, sistem mencatat Transaction *Hash* (TxHash) sebagai bukti transaksi.
      1. TxHash berisi data seperti:
         a. status: Status keberhasilan transaksi.
         b. txHash: Identitas unik transaksi.
         c. balanceBefore: Saldo sebelum transaksi.
         d. balanceAfter: Saldo setelah transaksi.
      2. Jika validasi gagal (TxHash tidak ada atau transaksi tidak diverifikasi), sistem akan mengulang hingga batas maksimum (loop limit).

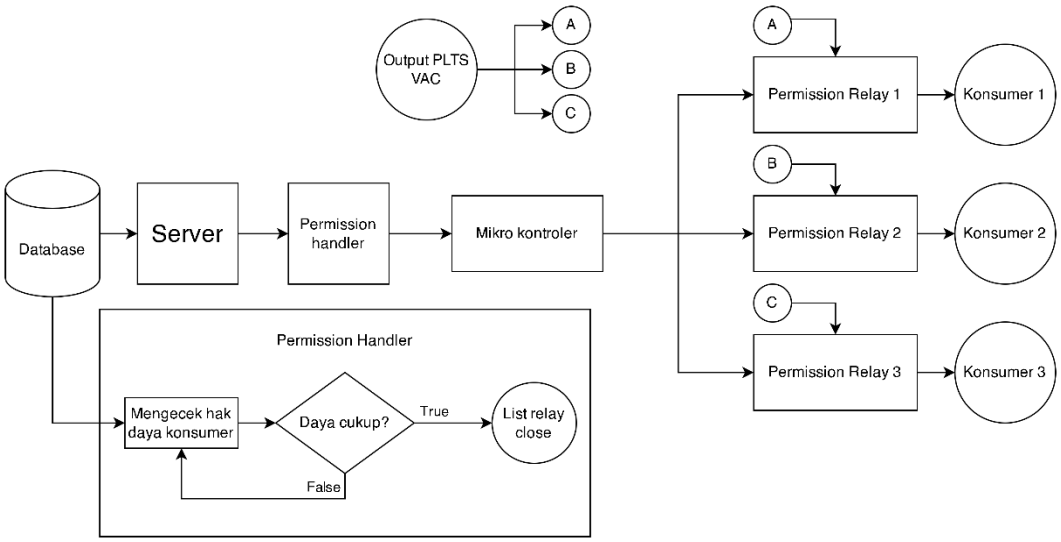
5. Eksekusi *Smart contract*
   a. Ketika transaksi berhasil, *smart contract* pada *blockchain* Ethereum mengatur transfer ETH dari dompet pembeli ke dompet penjual.
   b. Saldo pembeli akan berkurang sesuai jumlah transaksi, sementara saldo penjual bertambah.

6. Penggabungan Payload untuk Validasi Akhir
    a. Semua data transaksi (TxHash, status, dan saldo) digabung menjadi payload yang terverifikasi.
    b. Informasi ini kemudian dicatat pada *blockchain* sebagai log transaksi permanen.
7. Penutup dan Penyelesaian Transaksi
    a. Setelah TxHash diterima dan transaksi diverifikasi, proses dianggap selesai.
    b. Jika ada kesalahan selama transaksi (misalnya, saldo tidak cukup, alamat salah, atau validasi *blockchain* gagal), sistem akan memberikan pemberitahuan error kepada pengguna.
8. Gambar 3.2 merupakan proses transaksi



Gambar 3.2. Diagram proses transaksi
Sumber: Penulis 2025

Proses Distribusi Ke Konsumer



Gambar 3.3. Distribusi dan *permission check* hak daya konsumer

Sumber: Penulis 2025

Gambar 3.3 adalah diagram yang menggambarkan bagaimana sistem distribusi daya PLTS dikelola secara efisien dan terkendali untuk memastikan setiap konsumen menerima daya sesuai hak dan kapasitas yang ditentukan. Berikut adalah penjelasan komponen utama dan alur prosesnya:

1. Permintaan Daya:
    a. Konsumen mengajukan permintaan daya melalui sistem.
    b. Permintaan diverifikasi oleh server berdasarkan data yang tersimpan di database.
2. Pengecekan Hak Akses Daya:
    a. Permission Handler memeriksa apakah daya yang diminta konsumen sesuai dengan hak dan kapasitas daya yang tersedia.
3. Validasi Daya:
    a. Jika daya mencukupi:
        1. Permission Handler menambahkan relay yang bersangkutan ke dalam daftar relay *close*.

2. Relay diaktifkan untuk mengalirkan daya ke konsumen.

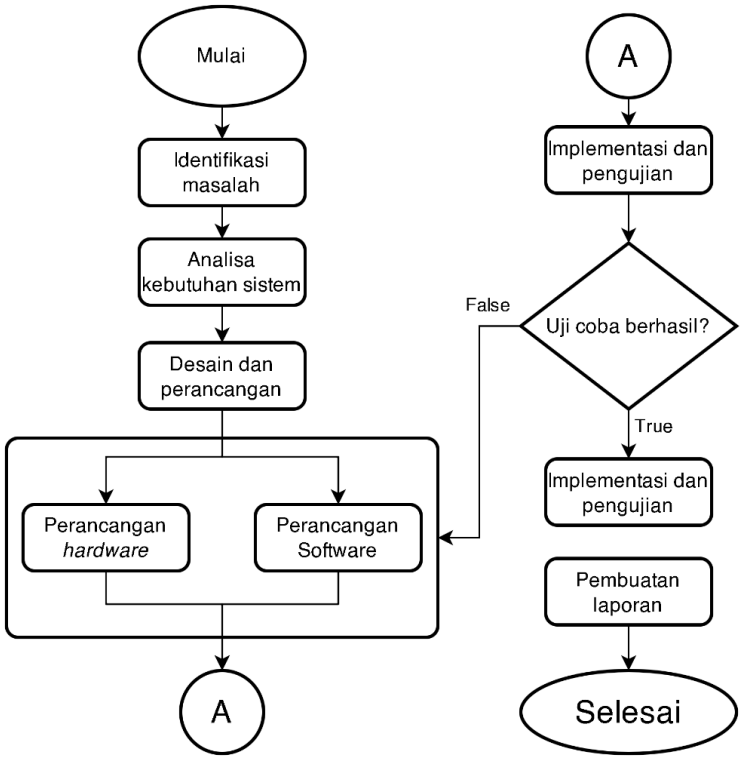
b. Jika daya tidak mencukupi:

1. Permintaan ditolak dan relay tidak aktif.

4. Eksekusi oleh Mikro Kontroler:
   a. Mikro kontroler menerima daftar relay yang harus diaktifkan dan membuka relay sesuai dengan instruksi dari server.
   b. Daya listrik dari PLTS VAC dialirkan ke konsumen melalui relay yang aktif.
5. Distribusi Daya ke Konsumen:
   a. Relay yang aktif mengalirkan daya sesuai dengan jalur masing-masing ke konsumen terkait.

3.2 Tahap Penelitian

Pada tahap penelitian kali ini, membahas tentang alur penelitian yang ditampilkan dalam bentuk diagram alur penelitian.



Gambar 3.4. Diagram Alur Peneliian

Sumber: Penulis 2025

Berdasarkan diagram alur penelitian pada Gambar 3.4 penelitian dimulai dari identifikasi masalah yang akan diteliti, peneliti akan memulai dengan mencari studi literatur untuk mengetahui penelitian-penelitain sebelumnya yang berkaitan dengan topik yang akan diangkat menjadi tugas akhir. Selanjutnya melakukan analisa kebutuhan sistem yang akan dibagun, selanjutnya peneliti akan melakukan perancangan terhadap *hardware* meliputi sensor dan aktuatornya, lalu *software* seperti aplikasi dan *debuggingnya*. Selanjutnya penulis akan mengimplementasikan dan menguji untuk memastikan keberhasilan sistem yang dibuat. Dari hasil uji akan dianalisis secara mendalam kemudian akan disampaikan dalam bentuk laporan tugas akhir.

3.2.1 Tahap Identifikasi Masalah

Pada tahap identifikasi masalah, dilakukan analisis awal untuk memahami permasalahan yang ada di lapangan dan tujuan yang ingin dicapai dalam penelitian ini. Adapun langkah-langkah yang dilakukan meliputi:

1. Identifikasi masalah

Pada tahap ini, peneliti mengidentifikasi masalah terkait integrasi Pembangkit Listrik Tenaga Surya (PLTS) dengan sistem transaksi energi yang transparan, aman, dan efisien. Permasalahan yang diidentifikasi meliputi:

a. Ketidakefektivan dalam distribusi energi dari PLTS ke konsumen, terutama pada pengelolaan kelebihan daya yang tidak terpakai.
b. Tidak adanya sistem otomatis yang mengelola dan mencatat transaksi energi secara transparan dan aman antara prosumer (producer-consumer) dan konsumen.
c. Kurangnya mekanisme untuk memberikan rekomendasi pembelian energi berdasarkan pola konsumsi dan produksi energi.

Dari permasalahan tersebut, peneliti menciptakan sebuah penelitian dengan judul "Transaksi Energi PLTS dengan *Blockchain* Ethereum Disertai Saran Pembelian Berbasis Regresi Linear", yang diharapkan dapat menyelesaikan permasalahan distribusi energi dan transaksi secara efisien dan transparan.

2. Penetapan Tujuan dan Manfaat Penelitian

Setelah permasalahan diidentifikasi, langkah berikutnya adalah menetapkan tujuan yang ingin dicapai melalui penelitian ini. Adapun tujuan dari penelitian ini adalah

1. Mengintegrasikan teknologi *blockchain* Ethereum ke dalam sistem energi terbarukan untuk menciptakan transparansi, keamanan, dan efisiensi dalam distribusi energi.
2. Menerapkan *smart contract*s pada *blockchain* untuk mengotomatisasi dan mengamankan transaksi energi antara prosumer dan konsumen.
3. Menggunakan metode regresi linier untuk memberikan rekomendasi pembelian energi berdasarkan data transaksi historis.

Setelah tujuan ditetapkan, maka manfaat-manfaat dari penelitian ini akan dapat diperoleh, berikut manfaat penelitian yang akan didapat jika penelitian ini berhasil:

1. Menawarkan solusi inovatif yang dapat mendukung efisiensi distribusi energi terbarukan.
2. Membantu meningkatkan kepercayaan antara penyedia energi dan konsumen melalui sistem transaksi yang transparan dan aman.
3. Mendukung pengembangan pasar energi *peer-to-peer* yang lebih demokratis dan berkelanjutan.
4. Fitur regresi linear yang memungkinkan konsumer dapat bijak dalam membeli energi listrik.

3. Studi Literatur

Langkah berikutnya adalah melakukan studi literatur untuk memahami teknologi, metode, dan sistem yang relevan dengan penelitian ini. Fokus studi literatur meliputi:

1. Teknologi *Blockchain* Ethereum

   Termasuk mekanisme transaksi pada *blockchain* Ethereum dan penerapan *smart contract*s untuk mendukung transaksi energi.

2. *Software Development Kit* (SDK)

   SDK yang digunakan dalam pembuatan server yang terhubung dengan *blockchain* Ethereum yang meliputi hardhat, NodeJS, Etherscan, Alchemy, dan EtherJS.

3. PLTS: Analisis distribusi energi, pengelolaan kelebihan energi, dan tantangan dalam integrasi ke sistem energi.
4. Metode Regresi Linier: Implementasi regresi linier untuk menghasilkan rekomendasi pembelian energi berdasarkan pola konsumsi dan produksi energi.
5. Perangkat IoT dan Komponen Pendukung: Penggunaan perangkat seperti ESP32, sensor PZEM-004T, dan relay dalam mendukung pengumpulan dan pengelolaan data energi.

Dengan pendekatan ini, penelitian ini bertujuan untuk menciptakan sistem distribusi energi berbasis PLTS yang efisien, transparan, dan dapat diandalkan.

3.2.2 Analisa Kebutuhan Sistem

Menganalisis kebutuhan sistem merupakan langkah untuk memahami kebutuhan sistem yang diperlukan dalam suatu penelitian. Berdasarkan diagram sistem yang telah disusun, beberapa perangkat diperlukan. Sementara itu, teknologi yang diterapkan mencakup *hardware* dan *software* yang disajikan pada tabel 3.10 sebagai berikut :

Tabel 3.1. Kebutuhan *Hardware* dan *Software*

HARDWARE	***SOFTWARE***
ESP32	Arduino IDE
Step Down	NodeJS
Relay SSR	MongoDB
Sensor PZEM-004T	Visual Studio Code
HARDWARE	***SOFTWARE***
SCC (*Solar Charge Control*)	Hardhat
Panel surya 550 Wp	Metamask
Baterai aki	Sepolia
Inverter	Etherscan
Laptop	Alchemy
WiFi	Solidity

3.2.3 Desain Perancangan Sistem

Pada tahap perancangan ini terdapat proses perancangan komponen fisik suatu sistem atau alat. Seperti rangkaian elektrik dan sensor. Meliputi penentuan spesifikasi, pemilihan komponen, dan pembutan desain fisik untuk memenuhi kebutuhan sistem.

Gambar 3.5 merupakan gambaran arsitektur dari sistem. manajemen daya yang langsung menggunakan mikrokontroler ESP32 dan akan dibuat pada penelitian ini. Berikut merupakan penjelasan dari masing-masing proses yang terdapat pada Gambar 3.5.

1. *Input* (PLTS)
    a. Energi matahari ditangkap oleh panel surya dan diolah oleh Solar Charge Controller (SSC) untuk diubah menjadi energi listrik DC yang kemudian disimpan di baterai.
    b. Inverter mengubah energi DC menjadi energi AC yang dapat digunakan untuk beban Listrik

2. Sensor PZEM-004T
    a. PZEM 004T digunakan untuk mengukur tegangan, arus, daya, dan energi pada masing-masing beban listrik (Load 1, Load 2, dan Load 3).
    b. Data dari masing-masing PZEM dikirimkan ke mikrokontroler ESP32 untuk diolah.
3. Mikrokontroler ESP32
    a. ESP32 menerima data dari PZEM dan memprosesnya untuk menentukan distribusi daya ke beban melalui relay.
    b. ESP32 juga berfungsi untuk mengirimkan data hasil pengukuran dan status distribusi daya ke server untuk ditampilkan melalui antarmuka web.
4. Relay
    a. Relay digunakan untuk memutus dan menyambung arus listrik ke beban sesuai dengan perintah yang diterima dari ESP32.
    b. Masing-masing relay terhubung ke satu beban listrik (Load 1, Load 2, dan Load 3).
5. *Output* (Tampilan Webserver)
    a. Data hasil pengukuran daya dan status relay dikirim ke server dan ditampilkan pada tampilan webserver.
    b. Pengguna dapat memantau konsumsi daya, status distribusi listrik, dan melakukan kontrol terhadap beban melalui antarmuka ini.

    Berikut adalah penjelasan dari hubungan antar perangkat.

    1. SSC ke PZEM

        SSC mengalirkan energi listrik yang diukur oleh PZEM sebelum diteruskan ke relay dan beban listrik.
    2. PZEM ke ESP32

        Data pengukuran daya dari setiap PZEM diterima oleh ESP32 untuk dianalisis.

3. ESP32 ke Relay

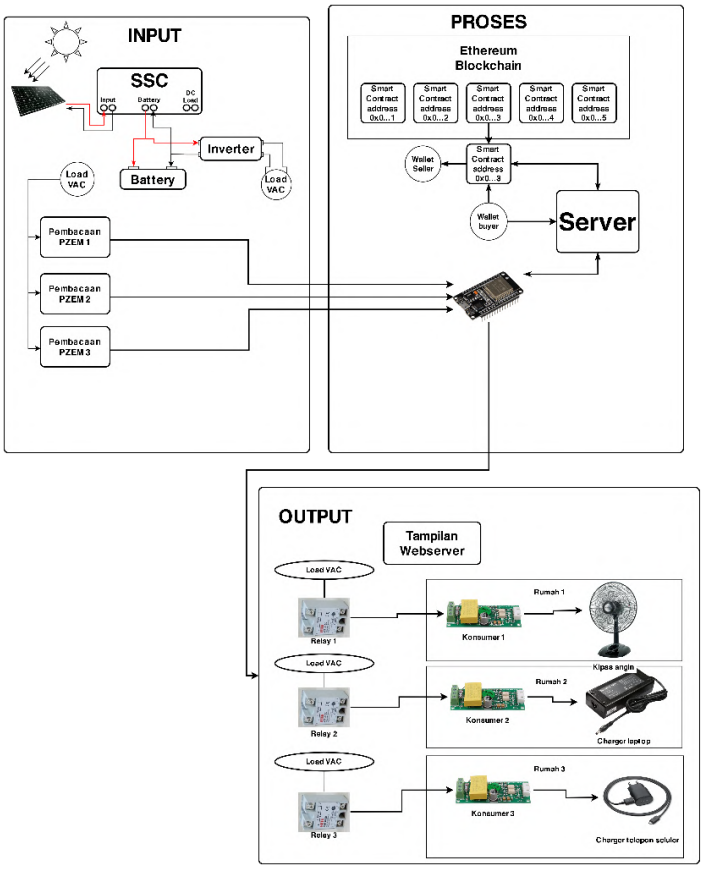
   Berdasarkan hasil analisis, ESP32 memberikan perintah kepada relay untuk mengatur aliran daya ke masing-masing beban listrik.

4. ESP32 ke Server

   ESP32 mengirim data ke server menggunakan protokol komunikasi untuk menampilkan status distribusi daya dan hasil pengukuran pada antarmuka web.

5. Server ke Webserver

   Server menerima data dari ESP32 dan menyajikannya dalam format yang mudah dipahami pengguna melalui tampilan webserver.



Gambar 3.5. Perancangan distribusi energi
Sumber: Penulis

3.2.4 Perancangan *Hardware*

Pada tahap perancangan *hardware* peneliti akan membutuhkan komponen termasuk sensor, aktuator, dan yang paling penting adalah mikrokontroler yaitu ESP32. Sensor kali ini menggunakan 3 PZEM-004T untuk mengukur tegangan,arus, dan daya listrik AC yang digunakan konsumer. selanjutnya, aktuatornya yaitu relay untuk *ON/OFF,* berdasarkan *permission list* yang dibuat oleh server ke mikro kontroler.

3.2.5 Perancangan *Software*

Perancangan perangkat lunak dalam penelitian ini bertujuan untuk memastikan semua komponen sistem dapat bekerja secara terintegrasi, baik perangkat keras, perangkat lunak, maupun interaksi dengan *blockchain* Ethereum. Komponen *software* dirancang untuk mendukung pengolahan data dari perangkat, pengendalian perangkat beban, pengelolaan transaksi berbasis *blockchain*, dan visualisasi data melalui antarmuka web. Berikut adalah tahap-tahap perancangan *software*:

Arsitektur *Software*

Perangkat lunak dirancang dengan menggunakan pendekatan arsitektur modular yang terdiri dari beberapa komponen utama:

1. Mikrokontroler ESP32:
    a. Mengelola komunikasi antara perangkat keras (sensor dan relay) dengan server.
    b. Mengirimkan data sensor (PZEM-004T, relay status) ke server dan melakukan komunikasi dengan protokol HTTP.
2. Server:
    a. Dibangun dengan Node.js untuk menangani permintaan dari ESP32 dan komunikasi dengan *blockchain* Ethereum.
    b. Mengelola transaksi energi, pengolahan data, dan pengiriman data ke *blockchain* menggunakan API seperti Ethers.js.

c. Mengolah data transaksi dan energi pada MongoDB.
d. Analisis regresi linear diimplementasikan langsung di server
e. Server menghasilkan rekomendasi pembelian energi berdasarkan analisis regresi linear terhadap data historis.

3. Antarmuka Web:
   a. Menampilkan informasi data real-time seperti penggunaan energi, transaksi, rekomendasi pembelian, dan status perangkat.
   b. Menggunakan framework HTML-CSS untuk antarmuka yang dinamis.
4. *Blockchain* Ethereum

   Menggunakan *smart contract* untuk mengelola transaksi energi antara pengguna (prosumer dan konsumen) dengan keamanan dan transparansi tinggi.
5. *Smart contract*

   Smart contract ditulis dalam Solidity untuk mengelola transaksi energi dan memastikan keamanan serta transparansi dalam transaksi.
6. Database

   Digunakan untuk menyimpan beberapa data, yaitu
   a. Log data energi dari ESP32.
   b. Transaksi *blockchain* dari *smart contract*.
   c. Hasil analisis regresi linear yang digunakan untuk rekomendasi pembelian energi.

Pemilihan Tools dan Teknologi

Beberapa tools dan SDK digunakan untuk mendukung pengembangan *software*, antara lain:

1. Hardhat:

   Mengelola dan menguji *smart contract* sebelum diterapkan ke *blockchain* Ethereum.

2. Solidity:

   Bahasa pemrograman untuk membuat *smart contract*.
3. Node.js:

   Menangani komunikasi server dan integrasi dengan perangkat IoT serta *blockchain*.
4. Ethers.js:

   Untuk berinteraksi dengan *blockchain* Ethereum, seperti membaca data transaksi, saldo dompet, dan mengirim transaksi.
5. MongoDB:

   Database untuk mencatat log transaksi dan penggunaan energi.
6. Arduino IDE:

   Digunakan untuk memprogram ESP32 dalam mengelola komunikasi dengan sensor, relay, dan server.
7. API Alchemy dan Etherscan:

   Memantau transaksi dan interaksi dengan *blockchain* Ethereum.

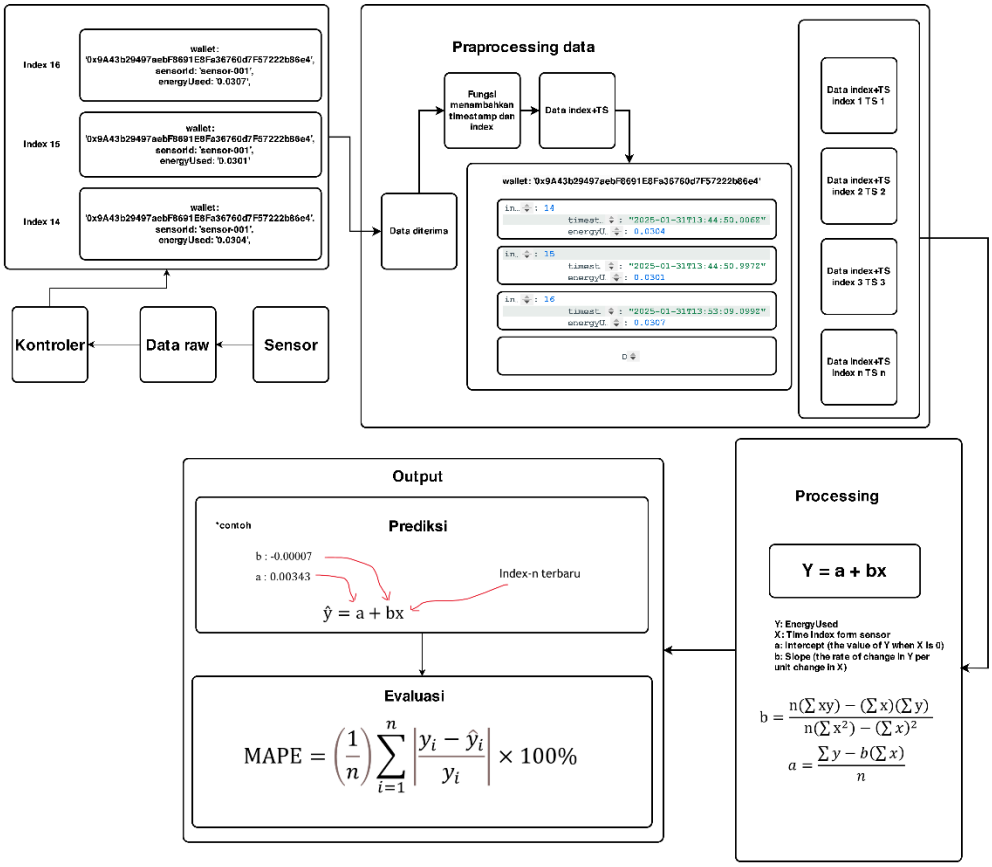
Alur Perangkat Lunak

1. Komunikasi Sensor dan ESP32:
   a. Data dari sensor PZEM-004T diterima oleh ESP32 melalui antarmuka serial.
   b. Data dikirimkan ke server menggunakan protokol MQTT atau HTTP.
2. Server dan *Blockchain*:
   a. Data dari ESP32 diterima oleh server.
   b. Server memproses data untuk transaksi *blockchain* menggunakan API Ethers.js.
   c. *Smart contract* pada *blockchain* Ethereum memvalidasi transaksi dan mencatat log transaksi.
3. Tampilan Web:
   a. Data real-time dari server dikirim ke antarmuka web untuk memantau penggunaan energi dan status perangkat.

b. Informasi transaksi dan histori penggunaan energi ditampilkan untuk pengguna.

Proses Regresi Linear di Server

Proses regresi linear diterapkan untuk memprediksi kebutuhan energi berbasis data historis yang dikumpulkan dari sistem. Tahapan proses ini memiliki gambaran seperti gambar 3.6.



Gambar 3.6. Diagram alur regresi linear

1. *Input* Data:
   a. Pengumpulan Data

      Data konsumsi energi diperoleh dari ESP32 yang mengukur tegangan, arus, daya, dan energi menggunakan sensor PZEM. Data ini kemudian dikirimkan ke server untuk disimpan dalam database MongoDB.

b. Jenis Data yang Digunakan

   Data yang mencakup pola penggunaan energi historis seperti konsumsi energi dan timestamp atau waktu menjadi *input* utama untuk model regresi linear.

2. Pemrosesan Data:
   a. Pengambilan Data

      Data yang disimpan dalam MongoDB diambil oleh server untuk dimasukkan ke dalam algoritma regresi linear.

   b. Model Regresi Linear

      Model regresi linear yang berjalan di server menghitung prediksi kebutuhan energi di masa depan berdasarkan pola penggunaan historis. Perhitungan ini mencakup identifikasi hubungan antara variabel independen (waktu, pola penggunaan) dan variabel dependen (kebutuhan energi).

3. Hasil Analisis:
   a. Rekomendasi Energi

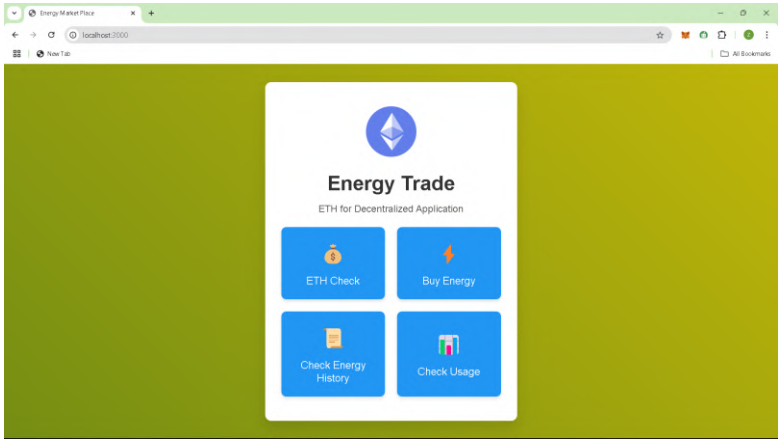
      Model regresi linear menghasilkan jumlah energi yang direkomendasikan untuk dibeli dalam satuan kWh. Rekomendasi ini bertujuan untuk mendukung pengguna dalam mengelola konsumsi energi mereka secara efisien.

   b. Integrasi dengan Antarmuka

      Hasil prediksi dikirimkan kembali ke antarmuka web yang memungkinkan pengguna melihat rekomendasi energi secara langsung melalui tampilan yang ramah pengguna.

Dengan perancangan perangkat lunak ini, diharapkan sistem dapat berjalan secara optimal dan mendukung integrasi antara PLTS, server, dan *blockchain* Ethereum untuk transaksi energi berbasis *smart contract*.

Perencanaan *User Interface* dalam bentuk website untruk konsumer

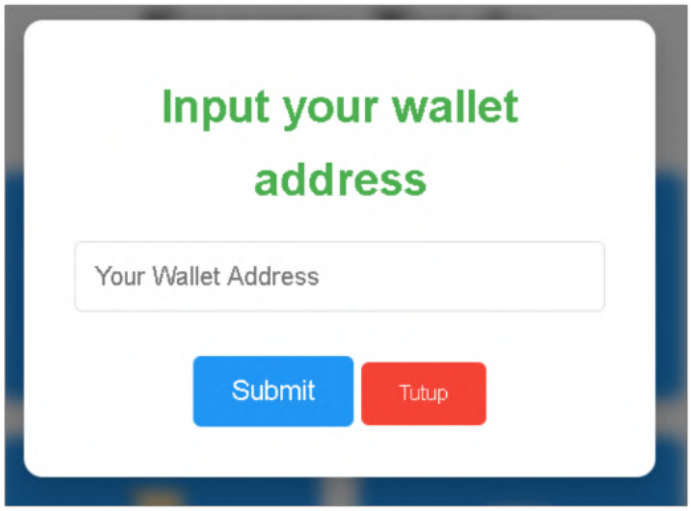


Gambar 3.7. Laman Pembeli

Laman *dashboard* pembeli untuk melakukan beberapa permintaaan

1. Fitur cek ETH pada pilihan "*ETH Check*"
2. Fitur pembelian hak energi pada fitur "*Buy Energy*"
3. Fitur cek Riwayat Pembelian "*Check Energy History*"
4. Fitur riwayat penggunaan energi, sisa energi, dan saran pembelian pada pilihan "*Check Usage*"

Gambar 3.8 adalah fitur untuk mengecek jumlah ETH dengan memasukkan alamat wallet

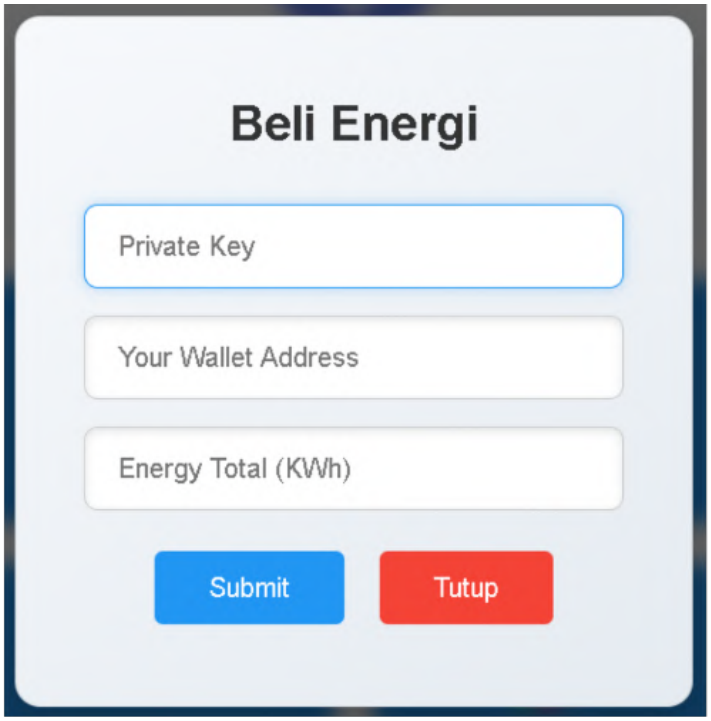


Gambar 3.8 Form cek jumlah ETH pada ETH *Check*

Gambar 3.9 adalah hasil pengecekan jumlah ETH pada saldo.

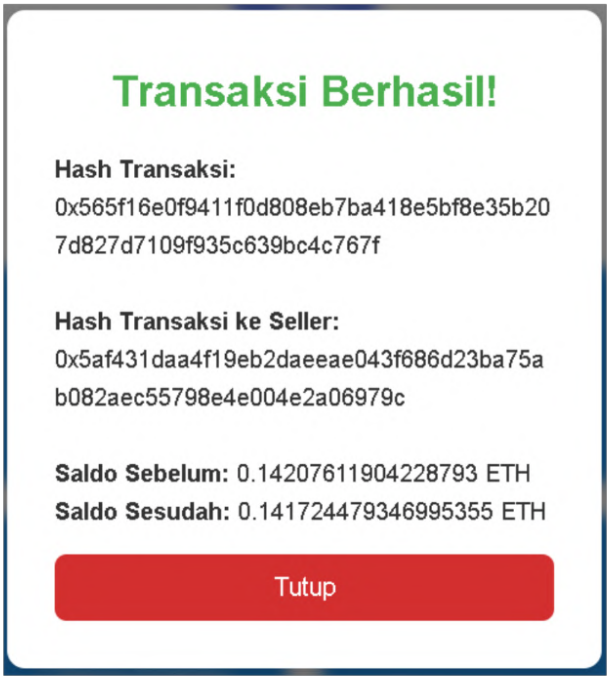
Gambar 3.9 Hasil pengecekan

Gambar 3.10 adalah fitur untuk membeli energi dengan ETH pembeli



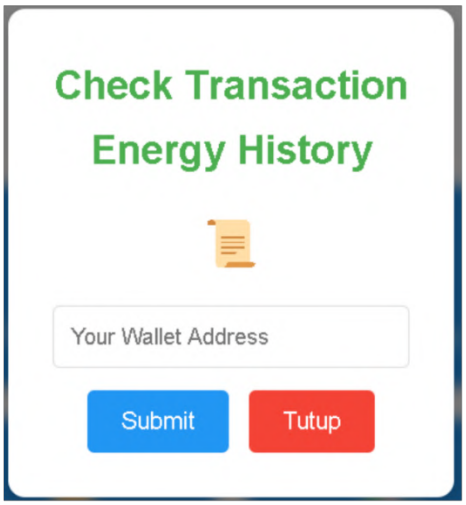
Gambar 3.10 *Form* pembelian energi

Gambar 3.11 adalah hasil setelah memasukkan data dengan benar dan bukti transaksi



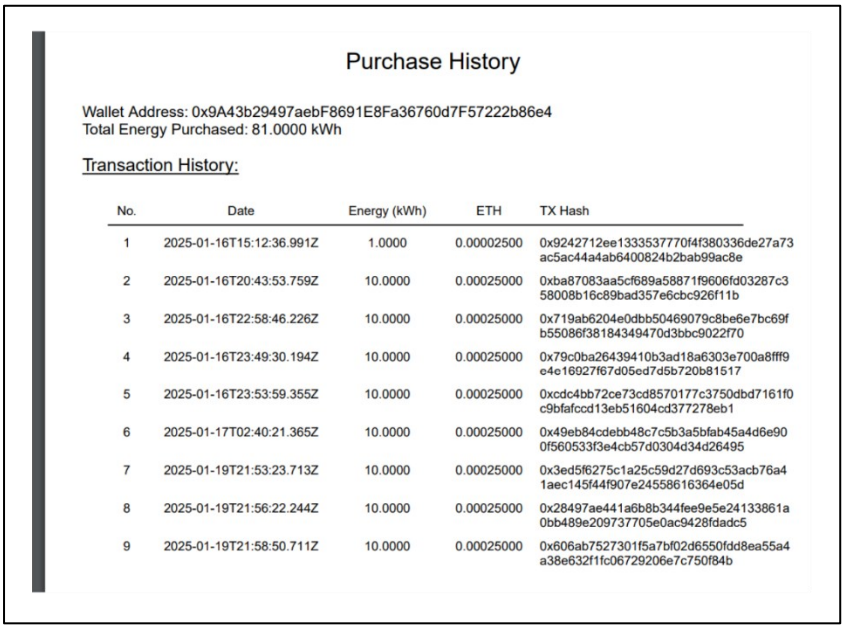
Gambar 3.11 Hasil bukti pembayaran

Gambar 3.11 adalah fitur untuk mengecek riwayat pembelian



Gambar 3.12 Form untuk mengisi riwayat pembelian

Hasil yang akan didapat setelah melakukan *input* dan *submit* adalah file dalam bentuk format PDF seperti gabar 3.12



Purchase History

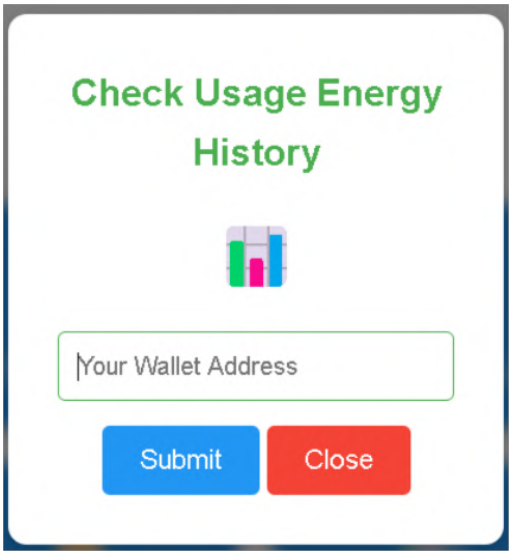
Wallet Address: 0x9A43b29497aebF8691E8Fa36760d7F57222b86e4
Total Energy Purchased: 81.0000 kWh

Transaction History:

No.	Date	Energy (kWh)	ETH	TX Hash
1	2025-01-16T15:12:36.991Z	1.0000	0.00002500	0x9242712ee1333537770f4f380336de27a73ac5ac44a4ab6400824b2bab99ac8e
2	2025-01-16T20:43:53.759Z	10.0000	0.00025000	0xba87083aa5cf689a58871f9606fd03287c358008b16c89bad357e6cbc926f11b
3	2025-01-16T22:58:46.226Z	10.0000	0.00025000	0x719ab6204e0dbb50469079c8be6e7bc69fb55086f38184349470d3bbc9022f70
4	2025-01-16T23:49:30.194Z	10.0000	0.00025000	0x79c0ba26439410b3ad18a6303e700a8fff9e4e16927f67d05ed7d5b720b81517
5	2025-01-16T23:53:59.355Z	10.0000	0.00025000	0xcdc4bb72ce73cd8570177c3750dbd7161f0c9bfafccd13eb51604cd377278eb1
6	2025-01-17T02:40:21.365Z	10.0000	0.00025000	0x49eb84cdebb48c7c5b3a5bfab45a4d6e900f560533f3e4cb57d0304d34d26495
7	2025-01-19T21:53:23.713Z	10.0000	0.00025000	0x3ed5f6275c1a25c59d27d693c53acb76a41aec145f44f907e24558616364e05d
8	2025-01-19T21:56:22.244Z	10.0000	0.00025000	0x28497ae441a6b8b344fee9e5e24133861a0bb489e209737705e0ac9428fdadc5
9	2025-01-19T21:58:50.711Z	10.0000	0.00025000	0x606ab7527301f5a7bf02d6550fdd8ea55a4a38e632f1fc06729206e7c750f84b

Gambar 3.13 Hasil riwayat format PDF

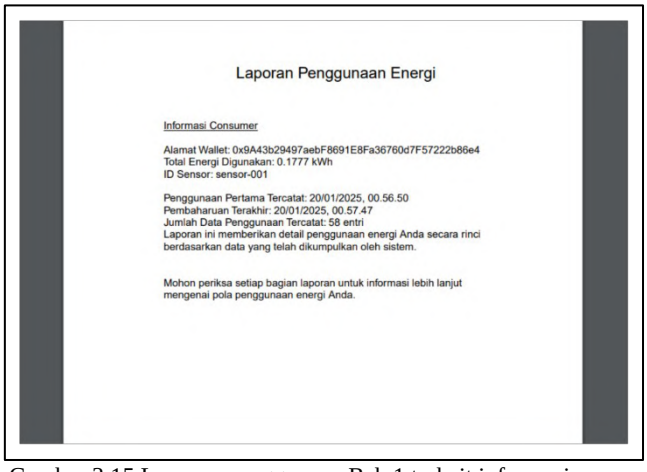
Hasil dalam bentuk format PDFGambar 3.13 adalah gambar fitur untuk mengecek riwayat penggunaan energi consumer menggunakan *form.*



Gambar 3.14 *Form* untuk melihat riwayat penggunaan

Hasil yang akan didapat setelah melakukan *input* dan *submit* adalah file dalam bentuk format PDF seperti pada gambar 3.14, 3,15, 3.16, dan 3.17. Laporan dibagi menjadi 3 bab, yaitu Informasi Umum, Detail Penggunaan Energi, dan Hasil (*Summary*).

Gambar 3.15 merupakan laporan pada bab 1 yang mencakup Informasi Umum


Laporan Penggunaan Energi

Informasi Consumer

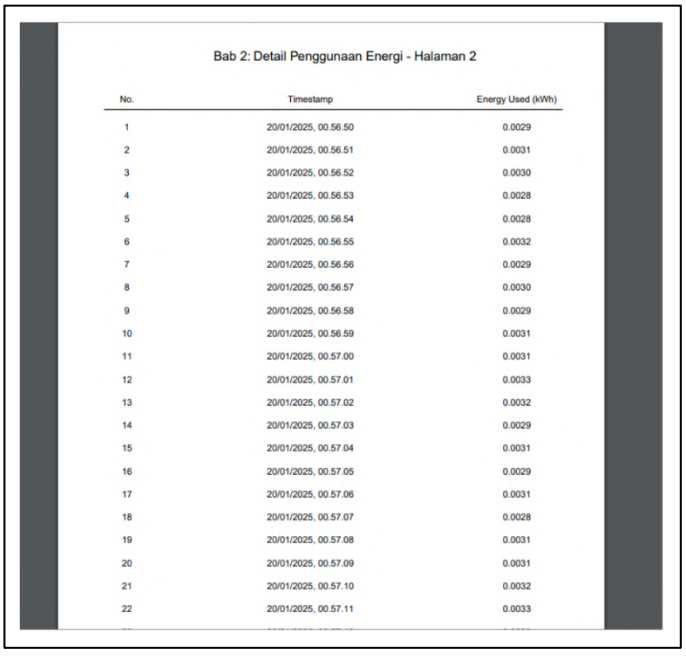
Alamat Wallet: 0x9A43b29497aebF8691E8Fa36760d7F57222b86e4
Total Energi Digunakan: 0.1777 kWh
ID Sensor: sensor-001

Penggunaan Pertama Tercatat: 20/01/2025, 00.56.50
Pembaharuan Terakhir: 20/01/2025, 00.57.47
Jumlah Data Penggunaan Tercatat: 58 entri
Laporan ini memberikan detail penggunaan energi Anda secara rinci berdasarkan data yang telah dikumpulkan oleh sistem.

Mohon periksa setiap bagian laporan untuk informasi lebih lanjut mengenai pola penggunaan energi Anda.

Gambar 3.15 Laporan penggunaan Bab 1 terkait informasi umum

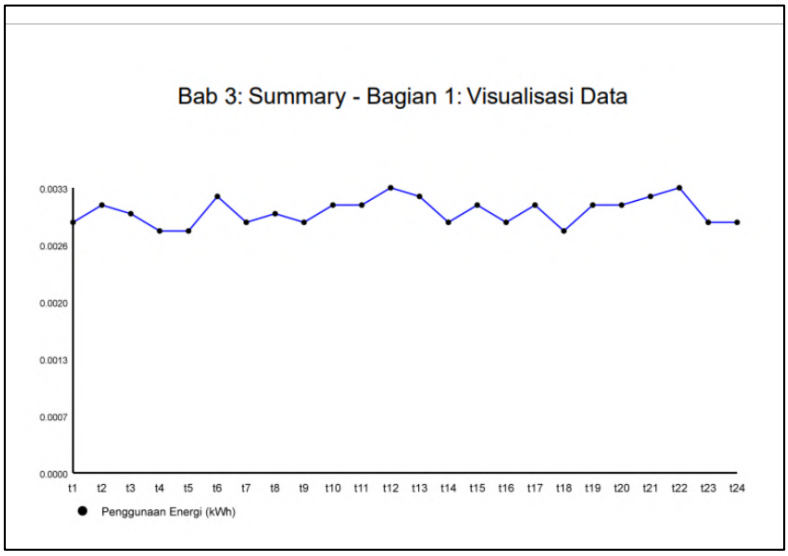
Gambar 3.16 merupakan rencana laporan pada bab 2, Detail Penggunaan Energi


Bab 2: Detail Penggunaan Energi - Halaman 2

No.	Timestamp	Energy Used (kWh)
1	20/01/2025, 00.56.50	0.0029
2	20/01/2025, 00.56.51	0.0031
3	20/01/2025, 00.56.52	0.0030
4	20/01/2025, 00.56.53	0.0028
5	20/01/2025, 00.56.54	0.0028
6	20/01/2025, 00.56.55	0.0032
7	20/01/2025, 00.56.56	0.0029
8	20/01/2025, 00.56.57	0.0030
9	20/01/2025, 00.56.58	0.0029
10	20/01/2025, 00.56.59	0.0031
11	20/01/2025, 00.57.00	0.0031
12	20/01/2025, 00.57.01	0.0033
13	20/01/2025, 00.57.02	0.0032
14	20/01/2025, 00.57.03	0.0029
15	20/01/2025, 00.57.04	0.0031
16	20/01/2025, 00.57.05	0.0029
17	20/01/2025, 00.57.06	0.0031
18	20/01/2025, 00.57.07	0.0028
19	20/01/2025, 00.57.08	0.0031
20	20/01/2025, 00.57.09	0.0031
21	20/01/2025, 00.57.10	0.0032
22	20/01/2025, 00.57.11	0.0033

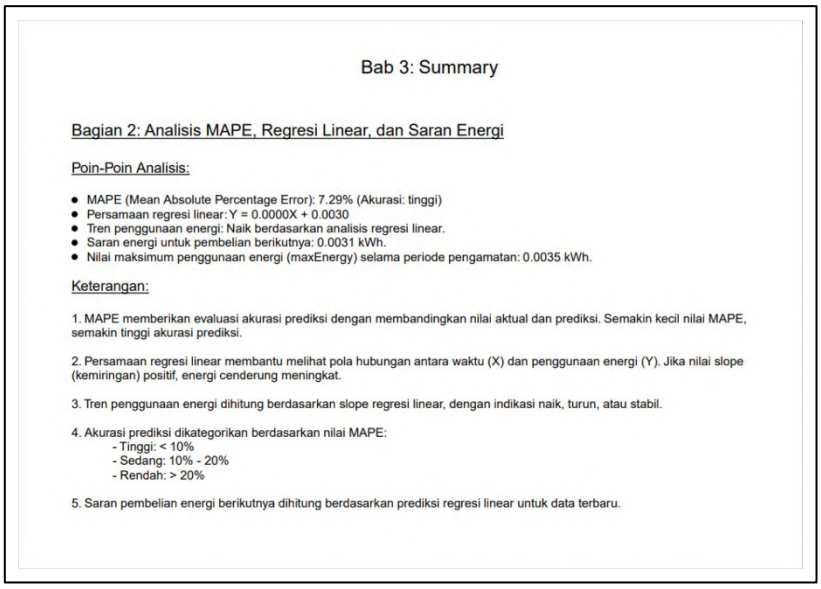
Gambar 3.16 bab 2, Detail Penggunaan Energi

Gambar 3.17 merupakan rencana laporan pada Bab 3, yang berisi visualisasi data



Gambar 3.17 Bab 2, Ringkasan bagian 1, visualisasi data

Gambar 3.18 merupakan rencana laporan pada bab 3, ringkasan (*summary*) bagian 2



Gambar 3.18 Bab 2, Ringkasan bagian 2, Ringkasan

3.2.6 Implementasi dan Pengujian

Pada tahap ini, sistem diuji untuk memastikan integrasi yang baik antara perangkat keras, perangkat lunak, dan *blockchain* Ethereum. Pengujian dilakukan dengan fokus pada:

- Sensor PZEM-004T untuk mengukur tegangan, arus, daya, dan energi.
- ESP32 untuk mengontrol relay dan berkomunikasi dengan server.
- Server Backend untuk mengolah data energi, melakukan analisis regresi linear, dan mencatat transaksi energi di *blockchain*.
- Antarmuka Web untuk menampilkan data energi, transaksi, dan rekomendasi pembelian energi.

3.2.7 Analisa Data

Data yang dikumpulkan selama pengujian dianalisis untuk mengevaluasi kinerja sistem, terutama dalam memberikan saran pembelian energi berdasarkan data historis pelanggan. Analisis difokuskan pada keakuratan prediksi model regresi linear dalam mendukung transaksi energi.

Evaluasi Model Prediksi

Evaluasi model regresi linear dilakukan menggunakan *Mean Absolute Percentage Error* (MAPE) sebagai indikator utama. MAPE digunakan untuk mengukur tingkat kesalahan prediksi dalam bentuk persentase. Nilai MAPE yang lebih kecil menunjukkan bahwa model memiliki tingkat akurasi yang tinggi dalam memberikan saran pembelian energi.

$$\mathrm{MAPE} = \frac{1}{\mathrm{n}} \sum \left| \frac{\mathrm{Y_i} - \hat{\mathrm{Y}}_\mathrm{i}}{\mathrm{Y_i}} \right| \times 100\% \qquad (3.1)$$

Dimana:

Y_i: Nilai aktual.

$\hat{Y}$: Nilai yang diukur oleh sistem.

n : Jumlah data

Berikut tujuan analisis data

1. Memastikan Akurasi Saran Pembelian Energi

   Nilai MAPE memberikan gambaran seberapa dekat prediksi model dengan kebutuhan energi aktual pelanggan.

2. Mengoptimalkan Efisiensi Sistem

   Menggunakan hasil analisis untuk menyempurnakan algoritma prediksi, sehingga saran pembelian menjadi lebih akurat dan sesuai dengan kebutuhan aktual pelanggan.

3. Mendukung Pengambilan Keputusan

   Saran pembelian energi yang didasarkan pada hasil evaluasi MAPE membantu pelanggan mengelola konsumsi energi mereka secara lebih efisien.

Melalui analisa ini, sistem diharapkan dapat memberikan rekomendasi pembelian energi dengan akurasi tinggi, memungkinkan efisiensi dalam transaksi energi berbasis *blockchain*.

BAB 4
HASIL DAN PEMBAHASAN

Dalam bab ini, penulis menjelaskan secara rinci hasil desain perangkat keras dan perangkat lunak serta serangkaian pengujian yang telah dilaksanakan untuk menyelesaikan Tugas Akhir. Pembahasan mencakup evaluasi menyeluruh terhadap kinerja sensor dan aktuator, dengan tujuan memverifikasi bahwa seluruh komponen sistem dapat bekerja sesuai dengan spesifikasi dan rancangan awal yang telah ditetapkan. Analisis mendalam terhadap data hasil pengujian juga disajikan untuk menilai kesesuaian sistem dengan tujuan penelitian.

4.1 Pengujian Parsial

PZEM-004t

Pengujian sensor PZEM-004t dilakukan dengan cara menggunakan pembanding antara PZEM-004t dan lampu 100W, seperti gambar 4.1

Gambar 4.1 Pengujian sensor PZEM-004t dengan wattmeter

Tabel 4.1 Pengujian daya (watt) dengan sensor PZEM-004t dan multimeter

Daya (watt)				
Lampu	Multimeter	PZEM	Error	Persentase error
1	99	103.3	4.3	4.34%
2	196	204.6	8.6	4.39%
3	295	309.1	14.1	4.78%
4	395	414.2	19.2	4.86%
5	491	514.7	23.7	4.83%
6	587	615	28	4.77%
Rata-rata %error				4.66%

Rata-rata persentase error daya dari tabel 4.1 pengujian daya antara PZEM dan multimeter menunjukkan nilai persen error 4.66%.

Tabel 4.2 Pengujian tegangan (V) dengan sensor PZEM-004t dan multimeter

Tegangan (V)				
1	227	235	8	3.52%
2	227	234.8	7.8	3.44%
3	227	234.5	7.5	3.30%
4	226	234.1	8.1	3.58%
5	226	233.7	7.7	3.41%
6	226	232.7	6.7	2.96%
Rata-rata %error				3.37%

Rata-rata persentase error daya dari tabel 4.2 pengujian daya antara PZEM dan multimeter menunjukkan nilai persen error 3.37%.

Tabel 4.3 Pengujian arus (A) dengan sensor PZEM-004t dan multimeter

Arus (A)				
1	0.43	0.44	0.01	2.33%
2	0.87	0.88	0.01	1.15%
3	1.3	1.323	0.023	1.77%
4	1.74	1.768	0.028	1.61%
5	2.17	2.212	0.042	1.94%
6	2.6	2.643	0.043	1.65%
Rata-rata %error				1.74%

Rata-rata persentase error daya dari tabel 4.3 pengujian daya antara PZEM dan multimeter menunjukkan nilai persen error 1.74%.

Berdasarkan hasil pengukuran tabel 4.1 – 4.3 antara multimeter dan PZEM terhadap tiga parameter listrik—daya, tegangan, dan arus—dapat disimpulkan bahwa PZEM secara konsisten menunjukkan nilai yang lebih tinggi dibandingkan multimeter. Pada pengukuran daya, PZEM memiliki

rata-rata error sebesar 4.66% dengan pola error yang meningkat seiring bertambahnya beban, menunjukkan kecenderungan *overestimate* yang stabil dan proporsional. Untuk tegangan, rata-rata error sebesar 3.37% menunjukkan deviasi yang kecil dan relatif stabil, namun tetap konsisten lebih tinggi dari multimeter. Sementara itu, pengukuran arus menunjukkan performa paling akurat, dengan rata-rata error hanya 1.74% dan tanpa pola deviasi signifikan.

Panel Surya (Photo Voltaic)

Gambar 4.2 Uji panel surya

Pengujian panel surya dilakukan dengan menggunakan *Solar Charge Controller* (SCC) dan *multimeter*, seperti pada gambar 4.2 dan hasil yang diperoleh dapat dilihat pada tabel 4.2

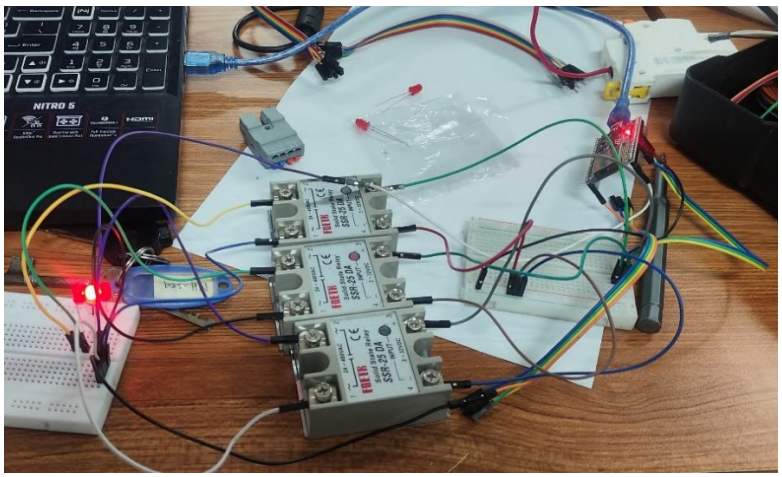
Tabel 4.4 Pengukuran PV dengan multimeter dan SCC (MPPT)

index	Waktu	Multimeter	SCC (MPPT)	Selisih error	Persentase error
1	10.05	41.7	41.2	0.50	1.20%
2	10.06	41.7	41.2	0.50	1.20%
3	10.07	41.6	41.1	0.50	1.20%

index	Waktu	Multimeter	SCC (MPPT)	Selisih error	Persentase error
4	10.08	41.4	40.9	0.50	1.20%
5	10.09	41.6	41.1	0.50	1.20%
6	10.10	41.6	41.1	0.50	1.20%
7	10.11	41.9	41.4	0.50	1.20%
8	10.12	41.7	41.2	0.50	1.20%
9	10.13	41.6	41.1	0.50	1.20%
10	10.14	41.8	41.3	0.50	1.20%
Rata-rata					1.20%

Berdasarkan data pengukuran antara multimeter dan SCC (MPPT) pada rentang waktu pukul 10.05 hingga 10.14, dapat disimpulkan bahwa nilai yang diukur oleh SCC secara konsisten lebih rendah sebesar 0.50 V dibandingkan dengan multimeter. Selisih ini menghasilkan rata-rata persentase error sekitar 1.20%, yang menunjukkan bahwa perbedaan tersebut sangat kecil dan stabil di seluruh waktu pengamatan. Nilai tegangan multimeter berada di kisaran 41.4–41.9 V, sementara SCC menunjukkan 40.9–41.4 V.

Relay SSR (Solid State Relay)



Gambar 4.3 Perobaan Relay

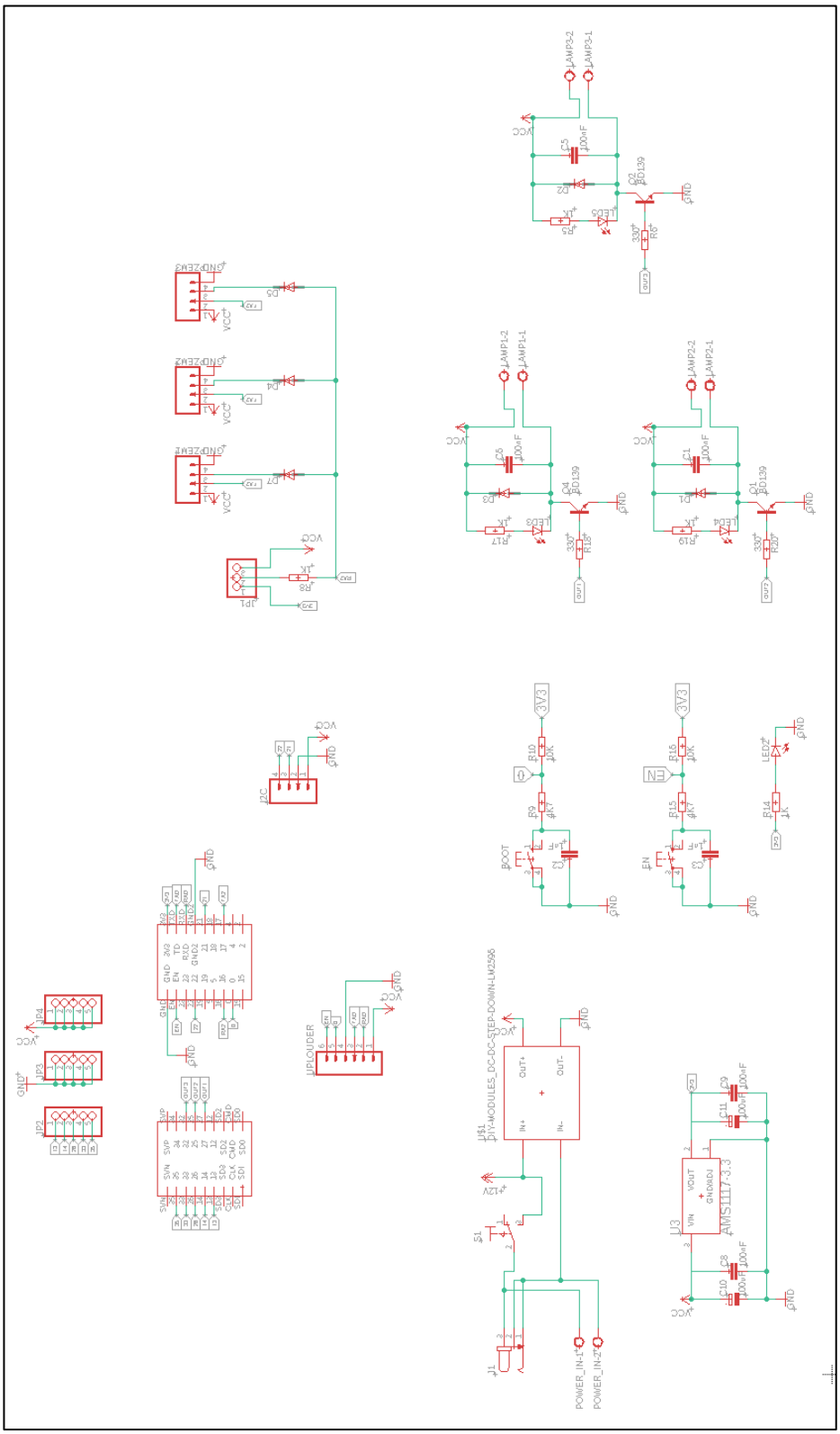
Pengujian Relay SSR dilakukan dengan cara melakukan pengujian menggunakan 3 lampu LED dan 3 SSR. Hasil pengujian dilakukan sesuai gambar 4.3 dan hasil dapat dilihat pada tabel 4.5.

Tabel 4.5 Percobaan Relay SSR

No	Relay			Lampu			% Error
	1	2	3	1	2	3	
1	ON	ON	ON	ON	ON	ON	0%
2	ON	ON	OFF	ON	ON	OFF	0%
3	ON	OFF	ON	ON	OFF	ON	0%
4	ON	OFF	OFF	ON	OFF	OFF	0%
5	OFF	ON	ON	OFF	ON	ON	0%
6	OFF	ON	OFF	OFF	ON	OFF	0%
7	OFF	OFF	ON	OFF	OFF	ON	0%
8	OFF	OFF	OFF	OFF	OFF	OFF	0%

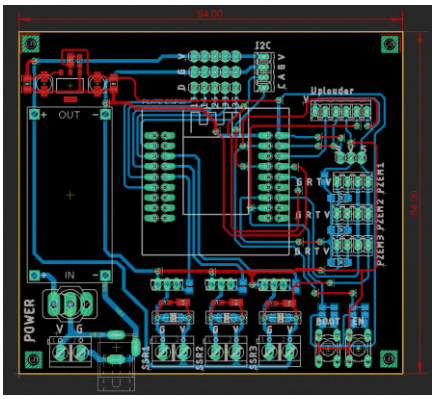
Berdasarkan tabel pengujian 4.5 status relay dan lampu, didapatkan bahwa terdapat korelasi langsung antara kondisi relay dan kondisi lampu, di mana relay 1 mengendalikan lampu 1, relay 2 mengendalikan lampu 2, dan relay 3 mengendalikan lampu 3. Pada baris pertama, ketika ketiga relay dalam kondisi ON, ketiga lampu juga menyala (ON), dan seiring dilakukannya kombinasi pemadaman relay, hasilnya menunjukkan bahwa hanya lampu yang relainya aktif yang menyala. Hal ini terus konsisten hingga kondisi terakhir (baris 8), di mana seluruh relay berada dalam posisi OFF, menyebabkan semua lampu dalam keadaan mati (OFF). Hasil ini menunjukkan bahwa sistem kontrol bekerja dengan baik dan responsif, dengan pengendalian individual yang akurat antara masing-masing relay dan lampu, tanpa adanya error atau ketidaksesuaian logika ON/OFF dalam pengujian ini.

4.2 Pembuatan mikrokontroler



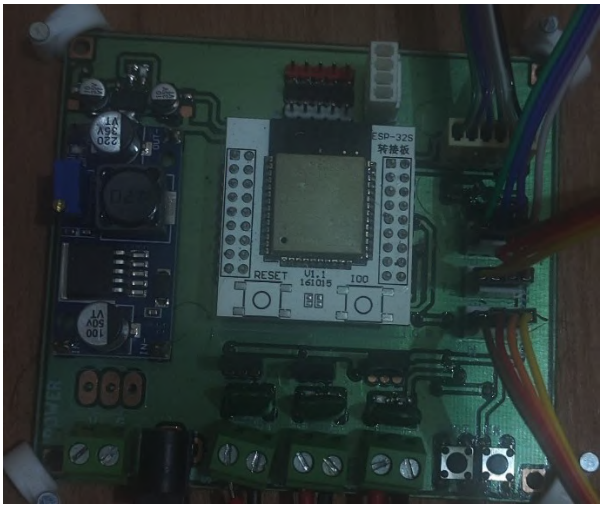
Gambar 4.4 Schematic mikro kontroler

Chip yang digunakan untuk mikro kontroler adalah esp32wroom32. Pin dapat dilihat seperti gambar 4.4 yaitu pin 27 untuk relay 1, pin 25 untuk relay 2, dan pin 32 untuk relay 3 serta pin untuk pembacaan modbus PZEM-004t menggunakan pin 16 (RX2) dan 17 (TX2). Terdapat juga pelindung arus balik pada pin relay (27, 25, 32).



Gambar 4.5 Board yang dicetak

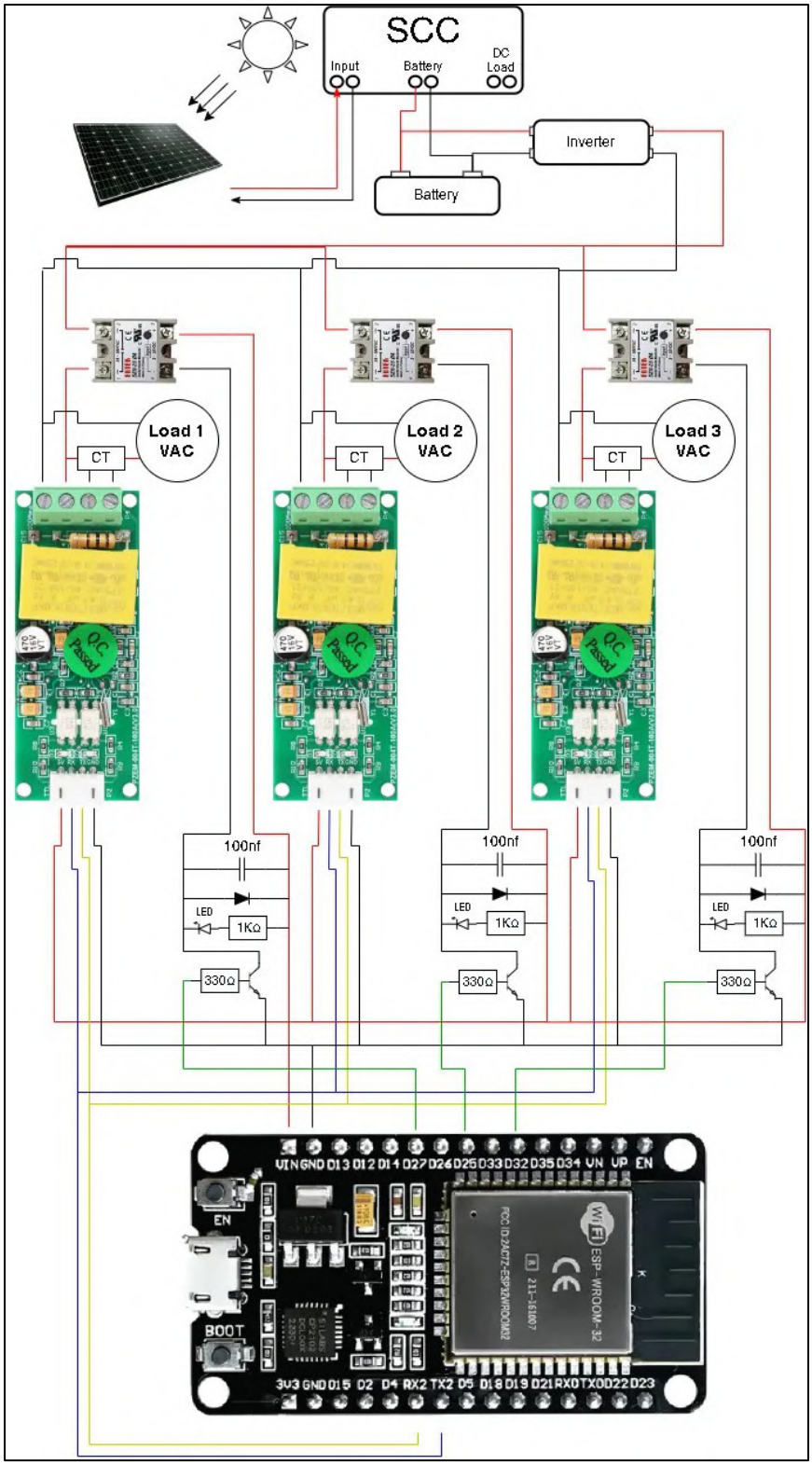
Rancangan pada gambar 4.5 merupakan final board yang diguanakan pada penelitian ini dan dicetak seperti pada gambar 4.6



Gambar 4.6 Hasil final dari board yang telah dirakit dan dicetak

Pada gamabr 4.6 adalah board yang telah dirakit dan telah disusun sesuai perencanaan di schematic dan desain board.

4.3 Diagram Wiring

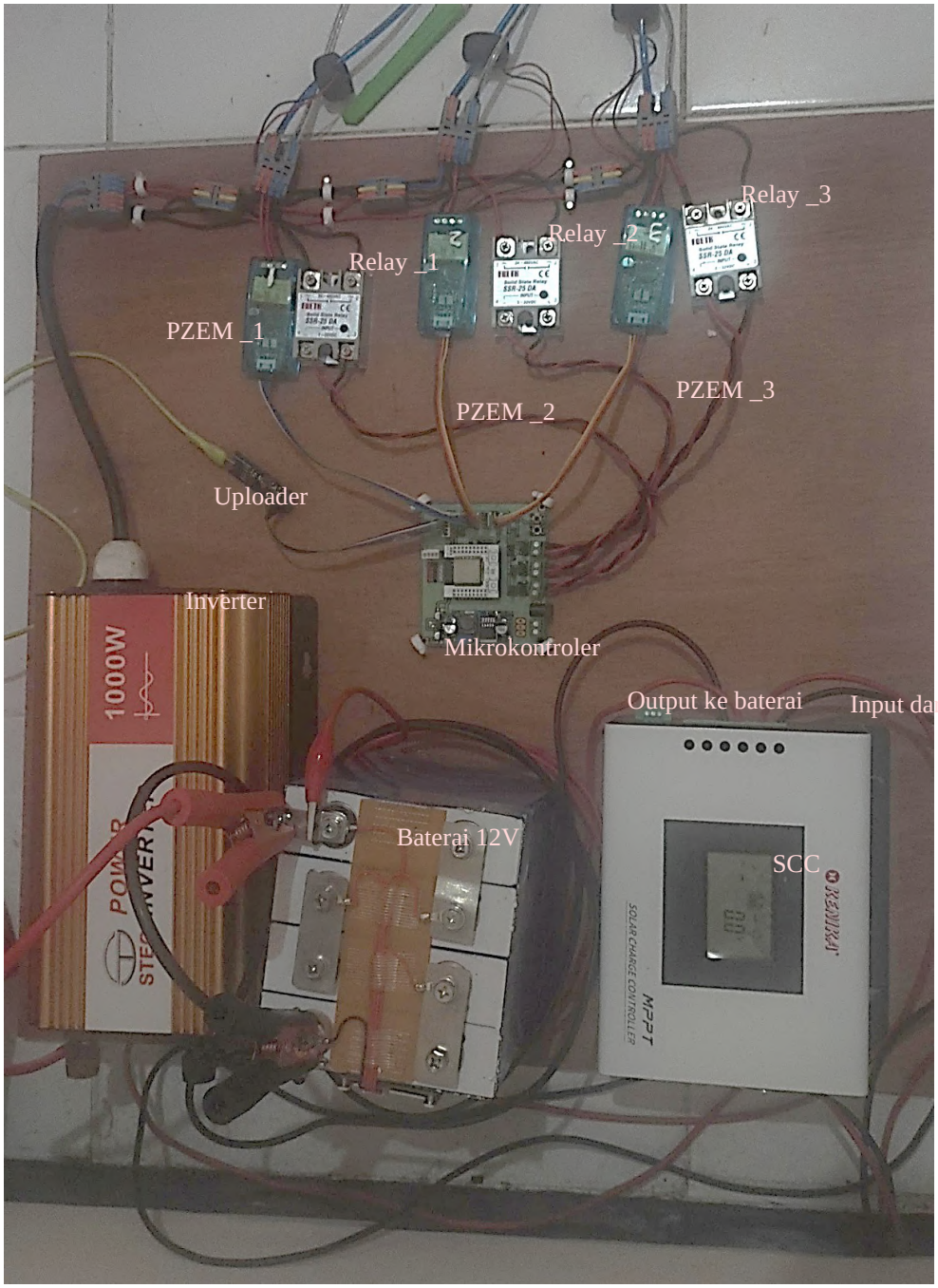


Gambar 4.7 Diagram Pengkabelan (*wiring*)

Diagram pengkabelan seperti gambar 4.7 merupakan hasil pengkabelan sesuai dengan perencanaan di mana terdapat:

1. Panel Surya yang akan dijadikan sumber utama penghasil listrik
2. Keluaran dari panel surya akan diteruskan ke SCC (Solar Charge Controller)
3. Pin *battery* dari SCC akan dihubungkan ke baterai
4. Kutub positif baterai akan dihubungkan ke kutub positif inverter
5. Kutub negatif baterai akan dihubungkan ke kutub negatif inverter
6. Output dari inverter akan berupa Listrik AC (Alternative Current)
7. Listrik AC tersebut akan dibagi dan dikontrol menjadi tiga untuk tiap konsumer menggunakan relay SSR serta dikontrol menggunakan PZEM
8. PZEM digunakan untuk membaca arus, tegangan, daya, faktor daya, dan frekuensi dengan menggunakan mikrokontroler
9. CT dipasang di salah satu kabel, yaitu L (line) daripada N (netral) untuk tiap-tiap konsumer
10. Komunikasi masing-masing PZEM ke ESP32 menggunakan RX dan TX yang dihubungkan ke ESP32 – RX2 di pin 16 dan TX2 di pin 17
11. Kontrol relay yang digunakan memiliki rangkaian pengaman yang terdiri dari LED, dioda, transistor NPN, kapasitor, dan resistor guna menangani dampak lonjakan tegangan serta memastikan kontrol yang stabil dan aman untuk mikrokontroler.
12. Kontrol relay menggunakan pin 27 untuk konsumer 1, pin 25 untuk konsumer 2, dan pin 32 untuk konsumer 3
13. Vin (5V) atau input tegangan diberikan ke tiga rangkaian komunikasi PZEM dan tiga rangkaian pengaman relay – ESP32

4.4 Hasil Perangkangkat keras



Gambar 4.8 Hasil rangkaian perangkat keras

Pada gambar 4.8 merupakan rangkaian perangkat keras yang telah disusun agar dapat bekerja sesuai dengan perencanaan meliputi:

1. Energi yang masuk dari sel surya akan dikontrol oleh Solar Charge Control (SCC) melewati output khusus bernama *battery*
2. Keluaran pin dari SCC, yaitu *battery* dihubungkan ke baterai lewat proses charge
3. Setelah baterai terisi, energi dalam baterai akan langsung di-*discharge* ke inverter untuk dijadikan listrik AC
4. Listrik AC akan dikontrol oleh mikrokontroler melalui relay dan PZEM di sini berfungsi untuk memonitoring listrik yang dididistribusi.

4.5 Pembangunan kontrak pintar (*smart contract*)

4.5.1 Pembuatan Lingkungan Tertutp Hardhat

Kontrak pintar dibuat di lingkungan pengembang yang terisolasi menggunakan hardhat yang telah disediakan oleh nodeJS dengan menggunakan CLI (*Command Line Interface*) "`npm init -y`" di direktori yang akan digunakan.

Setelah itu akan muncul, seperti di listing 4.1

```
C:\Users\ACER NITRO\Documents\belajar\ujicobanuklir>npx hardhat init
888    888                      888 888               888
888    888                      888 888               888
888    888                      888 888               888
8888888888  8888b.  888d888 .d88888 88888b.   8888b.  888888
888    888     "88b 888P"  d88" 888 888 "88b     "88b 888
888    888 .d888888 888    888  888 888  888 .d888888 888
888    888 888  888 888    Y88b 888 888  888 888  888 Y88b.
888    888 "Y888888 888     "Y88888 888  888 "Y888888  "Y888

Welcome to Hardhat v2.22.17

√ What do you want to do? · Create a JavaScript project
√ Hardhat project root: · C:\Users\ACER NITRO\Documents\belajar\ujicobanuklir
√ Do you want to add a .gitignore? (Y/n) · n
√ Do you want to install this sample project's dependencies with npm (hardhat @nomicfoundation/hardhat-toolbox)? (Y/n) · n

You need to install these dependencies to run the sample project:
  npm install --save-dev "hardhat@^2.22.17" "@nomicfoundation/hardhat-toolbox@^5.0.0"

Project created

See the README.md file for some example tasks you can run

Give Hardhat a star on Github if you're enjoying it!

     https://github.com/NomicFoundation/hardhat

C:\Users\ACER NITRO\Documents\belajar\ujicobanuklir>
```

Listing 4.1 Pembuatan lingkungan terisolasi

Setelah semua selesai maka lingkungan tertutup ini dapat digunakan sebagai tempat kerja untuk *development*.

4.5.2 *Script* kontrak pintar

```
// contracts/MessageStore.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimplePayment {
    address public seller; // Alamat penjual

    // Constructor untuk menentukan alamat penjual
    constructor(address _seller) {
        seller = _seller;
    }

    // Fungsi untuk menerima ETH
    receive() external payable {}

    // Fungsi untuk mengirim ETH ke penjual
    function transferToSeller() public {
        require(address(this).balance > 0, "No funds available");

        // Mengirimkan semua ETH yang ada di contract ke penjual
        (bool success, ) = seller.call{value: address(this).balance}("");
        require(success, "Transfer to seller failed.");
    }

    // Fungsi untuk mendapatkan saldo contract
    function getBalance() public view returns (uint256) {
        return address(this).balance;
    }
}
```

Listing 4.2 *Script* kontrak pintar

Penulisan *smart contract* seperti listing 4.2 bernama "`SimplePayment`" ini menggunakan satu variable yang harus didefinisikan atau diisi saat akan melakukan deploy smart contract oleh server dengan parameter "`address _seller`". Pada kontrak pintar ini mempunyai beberapa fungsi, meliput:

1. Inisialisasi:

   Menentukan seller (alamat penjual) via constructor saat deploy.

2. Terima ETH
   1. Fungsi "receive()" aktif saat ETH dikirim ke kontrak.
   2. Menyimpan ETH ke saldo kontrak.
3. Cek Saldo
   1. Memanggil `getBalance()` untuk lihat jumlah ETH di kontrak.
   2. Mengembalikan nilai "`address(this).balance`"
4. Kirim ETH ke Penjual dengan:
   1. Memanggil "`transferToSeller()`"

2. Mengecek saldo > 0, jika tidak, gagal ("*No funds available*").
3. Mengirim semua ETH ke seller via call.
4. Verifikasi transfer berhasil, jika gagal, maka ("*Transfer to seller failed*.").

4.5.3 *Deploy Smart Contract*

```
require("@nomiclabs/hardhat-waffle");
require("@nomiclabs/hardhat-ethers");
require("@nomiclabs/hardhat-etherscan");
require('dotenv').config({ path: __dirname + '/zzz.env' })

const mnemonic = process.env.MNEMONIC; // Ensure MNEMONIC is defined in your zzz.env file
const alchemyKey = process.env.ALCHEMY_KEY; // Ensure ALCHEMY_KEY is defined in your zzz.env file

/** @type import('hardhat/config').HardhatUserConfig */
module.exports = {
  solidity: "0.8.28",
  networks: {
    sepolia: {
      url: `https://eth-sepolia.g.alchemy.com/v2/${alchemyKey}`,
      accounts: { mnemonic: mnemonic },
    },
  },
  etherscan: {
    apiKey: process.env.ETHERSCAN_API_KEY, // Ensure ETHERSCAN_API_KEY is defined in your zzz.env file
  },
};
```

Listing 4.3 *Config deploy* kontrak pintar

```
MNEMONIC=start [redacted]
ALCHEMY_KEY=3F[redacted]
ETHERSCAN_API_KEY=J5SHIZ[redacted]
PRIVATE_KEY=df2b48[redacted]
WALLET=0xG73ddD0D4a28309D1fD9321A8Fde5A0bac78179E
SELLER=j2n3p0vusm98rs976z9fds87z87vlIUIBKHBskljLUIH89689jbskPlkkLoubiyigjLjlkhlhifyi1
SELLER_PASS=dfg098af97knj402kbUGKBJHAS8968KJkBJAujkhzpdtrdsgftfljhglkjsldfjnkhsbdlfsbkjLKJBgkhcKJGVRTS54w76JKHVMJ540934
```

Listing 4.4 File zzz.env

Berikut ini penjelasan *deploy* kontrak pintar sesuai listing 4.5.3 menggunakan parameter MNEMONIC, ALCHEMY_KEY, dan ETHERSCAN_API_KEY dari zzz.env seperti listing 4.5.4 dan berikut adalah penjelasan *config*-nya:

1. Menggunakan pustaka (Waffle, Ethers.js, Etherscan) untuk kompilasi, testing, deploy, dan verifikasi kontrak.
2. Menghubungkan ke *testnet* Sepolia lewat Alchemy dengan kunci API.
3. Menggunakan mnemonic untuk membuat akun Ethereum untuk deploy.
4. Memungkinkan verifikasi kontrak di Etherscan dengan API key.
5. Menyimpan informasi yang sensitif (mnemonic, kunci API) di file zzz.env.

Berikut penjelasan *script* eksekusi *deploy* kontrak pintar seperti listing 4.5

1. Nilai alamat penjual 0x673dd... diberikan ke variabel sellerAddress di skrip deploy JavaScript.
2. Nilai alamat 0x673dd... diteruskan sebagai argumen saat fungsi "SimplePayment.deploy(sellerAddress)" dipanggil.
3. Nilai alamat 0x673dd... diterima oleh parameter "`address _seller`" di constructor kontrak Solidity di listing 4.4.1.1.
4. Nilai alamat 0x673dd... dari "`_seller`" disalin ke variabel state address public "seller" di dalam constructor.
5. Nilai alamat 0x673dd... disimpan secara permanen di storage blockchain sebagai bagian dari state seller setelah deploy selesai.

```
async function main() {
  const [deployer] = await ethers.getSigners();

  // Ganti dengan alamat wallet yang ingin Anda gunakan sebagai alamat penjual
  const sellerAddress = "0x673ddD0D4a28309D1fD9321A8Fde5A0bac78179E";

  const SimplePayment = await ethers.getContractFactory("SimplePayment");
  const simplePayment = await SimplePayment.deploy(sellerAddress);
  await simplePayment.deployed();
  console.log("SimplePayment deployed to:", simplePayment.address);
}

main()
.then(() => process.exit(0))
.catch((error) => {
    console.error(error);
    process.exit(1);
});
```

Listing 4.5 Script eksekusi *deploy* kontrak pintar

Untuk melakukan ekseskusi deploy kontrak pintar dapat menggunakan CLI (*Command Line Interface*) di direktori awal pendeployan.

```
PS C:\Users\ACER NITRO\Documents\belajar\hardhatETH> npx hardhat run scripts/deploy.js --network sepolia
SimplePayment deployed to: 0x747af05F8CEDCe8Bf59232b6Be032a1b78cD62AD
PS C:\Users\ACER NITRO\Documents\belajar\hardhatETH>
```

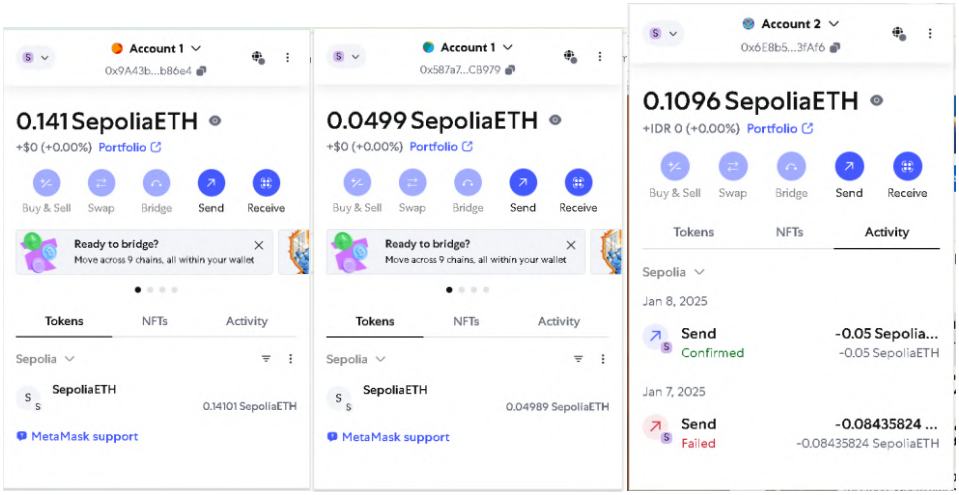
Listing 4.6 CLI eksekusi *deploy* kontrak pintar

CLI yang dipakai adalah "`npx hardhat run (direktori script)/(program deploy.js) --network sepolia`".

Setelah itu alamat kontrak dihasilkan dengan alamat seperti listing 4.6, akan tetapi alamat kontrak pada kasus ini sudah dibuat sebelumnya, yaitu ada pada alamat ”0x8254a986319461bf29ae35940a96786e507ad9ac”.

4.6 Pembuatan Dompet Digital

Pembuatan dompet digital menggunakan metamask yang dapat diunduh di ekstensi *browser* chrome.



Gambar 4.9 Daftar dompet digital pembeli

Dompet digital yang telah dibuat seperti gambar 4.9 ini akan digunakan sebagai pembeli, berikut *mapping* daftar dompet pembeli:

1. Pembeli 1 (kiri)
   “0x9A43b29497aebF8691E8Fa36760d7F57222b86e4”
2. Pembeli 2 (tengah)
   “0x587a768aa01bd43f8Aa720A4539DAF8CB96CB979”
3. Pembeli 3 (kanan)
   “0x6E8b56414035d7F8C6635A911beDb1044203fAf6”

4.7 Server *Handler Transaction*

Server di sini berperan sebagai perantara antara hasil transaksi yang melalui kontrak pintar. Server juga berfungsi untuk melakukan validasi *request* dari pembeli dengan nilai ETH yang diberikan apakah sesuai atau tidak.

```
// API Key Alchemy untuk Sepolia
const provider = new ethers.providers.JsonRpcProvider("https://eth-sepolia.g.alchemy.com/v2/(API_xxx)");

const contractAddress = "0xB9067eA7df746A653Ac86D31b8361FE34048e655"; // Alamat smart contract
```

Listing 4.7. Penggunaan API dari Alchemy dan kontrak pintar

Server akan menggunakan API dari Alchemy seperti pada listing 4.7 untuk berkomunikasi dengan *blockchain* sepolia sebagai node Ethereum, Ether.js untuk mendapatkan hasil *request* transaksi, seperti hasil transaksi dari kontrak pintar, dan jika diperlukan menggunakan Etherscan (eksternal) untuk memvalidasi status transaksi secara manual apakah benar-benar terjadi atau tidak.

```
app.post('/sendTransaction', async (req, res) => {
    const buyerPrivateKey = req.body.privateKey;
    const buyerAddress = req.body.address;
    const energyAmount = req.body.energyAmount;
```

Listing 4.8. Request dari server yang dipicu oleh pembeli

Pada listing 4.8 pembeli bisa melakukan *request* dari server lalu server akan memproses transaksi dalam bentuk *request* pembeli menggunakan parameter *body* (buyerPrivateKey, buyer*Address*, energyAmount) dengan metode *post* dan dicatat oleh server.

```
// Validasi input
if (!buyerPrivateKey || !buyerAddress || !energyAmount || isNaN(parseFloat(energyAmount))) {
    return res.status(400).json({ status: 'Error', message: 'Missing or invalid required parameters' });
}

try {
    // Validasi wallet
    let buyerWallet;
    try {
        buyerWallet = new ethers.Wallet(buyerPrivateKey);
    } catch (error) {
        return res.status(400).json({ status: 'Error', message: 'Wrong on wallet address or private key' });
    }

    // Validasi apakah address dari wallet cocok dengan address yang diberikan
    if (buyerWallet.address.toLowerCase() !== buyerAddress.toLowerCase()) {
        return res.status(400).json({ status: 'Error', message: 'Wrong on wallet address or private key' });
    }
```

Listing 4.9. Sistem validasi wallet

Pada listing 4.9 server melakukan beberapa pemeriksaan dan validasi untuk memastikan masukan yang diterima oleh API sudah benar dan aman sebelum diproses lebih lanjut. Berikut penjelasannya:

1. Memeriksa apakah *body* pada semua parameter wajib (buyerPrivateKey, buyer*Address*, energyAmount) ada
2. Memastikan energyAmount adalah angka yang valid dengan `isNaN(parseFloat(energyAmount))`
3. Jika ada yang tidak valid, mengembalikan response error HTTP 400 (Bad *Request*)
4. Mencoba membuat objek wallet Ethereum menggunakan *private key* yang diberikan dengan memanfaatkan pustaka Ether.JS
5. Jika gagal (misalnya format private key salah), menangkap error dan mengembalikan *response* error
6. Memeriksa apakah *address* yang dihasilkan dari private key cocok dengan *address* yang diberikan
7. Menggunakan `toLowerCase()` untuk memastikan perbandingan *case-insensitive*
8. Jika tidak cocok, berarti ada ketidaksesuaian antara *private key* dan *address* yang diberikan

```
// Validasi saldo
const balanceBefore = await provider.getBalance(buyerWallet.address);
console.log(`Balance before transaction: ${ethers.utils.formatEther(balanceBefore)} ETH`);
```

Listing 4.10. Verifikasi saldo pembeli

9. Pada listing 4.10, server akan melakukan validasi saldo pembeli

Selanjutnya, jika server telah selesai melakukan validasi maka proses transaksi dapat berlangsung seperti listing 4.11 yang akan memberikan proses transaksi.

```
// Validasi saldo
const balanceBefore = await provider.getBalance(buyerWallet.address);
console.log(`Balance before transaction: ${ethers.utils.formatEther(balanceBefore)} ETH`);

// Kalkulasi jumlah ETH berdasarkan energi dalam kWh
const energyInKWh = parseFloat(energyAmount); // Energi dalam kWh
const ethRatePerKWh = 0.000000025; // Tarif ETH per Wh
const rawAmount = energyInKWh * ethRatePerKWh; // Hitung nilai mentah

// Bulatkan nilai ke 18 desimal sebelum digunakan
const roundedAmount = rawAmount.toFixed(18);
const amount = ethers.utils.parseUnits(roundedAmount, "ether");
```

Listing 4.11. Program konversi dan perhitungan gasfee dan konversi nilai produk

1. Server melakukan konversi terhadap harga energi yang akan diperjual belikan dengan ETH
2. *Request* dikirim *user* akan diproses dan dikonversi ke ETH
3. Mengestimasi nilai biaya gas untuk transaksi.
4. Mengkonversi nilai 1 ETH menjadi 10^{10} wei.
5. Contoh:

   0.0025 ETH → "0.002500000000000000" (string)

   lalu dengan menggunakan fungsi dari ether.js, yaitu `ethers.utils.parseUnits()` akan mengubahnya menjadi dalam bentuk satuan wei (2500000000000000 wei).
6. `provider.estimateGas()`

   Mengecek berapa *gas limit* yang dibutuhkan untuk mengirim ETH ke *contractAddress*.
7. `provider.getGasPrice()`

   Mendapatkan harga gas per unit (dalam wei) saat ini di jaringan.
8. `totalGasCost`

   Biaya total gas = gasLimit * gasPrice (dalam wei)

   Contoh:

   a. Gas limit: 21000 unit

   b. Gas price: 20 Gwei (20×10^9 wei)

   c. Total: 21000 * 20 Gwei = 0.00042 ETH.

9. Lalu server akan melakukan validasi ulang saldo pengguna dengan fungsi
   a. `balanceBefore`: Saldo pengguna sebelum transaksi (dalam wei).
   b. `totalCost`: Jumlah ETH yang dikirim (amount) + biaya gas (totalGasCost).
   c. `balanceBefore.lt(totalCost)`:Mengecek apakah saldo mencukupi. Jika tidak, kembalikan error dan transaksi batal.

```
// Simpan data request
const requestData = {
    address: buyerAddress,
    energyAmount: energyAmount,
    hash_txid: null,
    timestamp: getCurrentTime()
};
const requestResult = await requestsCollection.insertOne(requestData);
console.log('Request data inserted with id:', requestResult.insertedId);

// Kirim ETH ke smart contract
const tx = await buyerWallet.connect(provider).sendTransaction({
    to: contractAddress,
    value: amount
});

console.log("Transaction sent:", tx);
const receipt = await tx.wait();

// Perbarui request dengan hash transaksi pertama
await requestsCollection.updateOne(
    { _id: requestResult.insertedId },
    { $set: { hash_txid: tx.hash } }
);
```

Listing 4.12. Integrasi request dan validasi transaksi

10. Setelah itu fungsi *request* dari server di listing 4.12 dipicu oleh pembeli akan disimpan sementara ke dalam *database* dan menunggu hasil Txhash muncul dan tervalidasi

```
Received request: {
  address: '0x6E8b56414035d7F8C6635A911beDb1044203fAf6',
  energyAmount: '1'
}
Balance before transaction: 0.110009257019608365 ETH
Energy in kWh: 1
Amount to send (ETH): 0.000025000000000000
Estimated gas cost (buyer transaction): 0.00000156551487924 ETH
Connected to database
Request data inserted with id: new ObjectId('681d01e04504397d37ced32b')
Transaction sent: {
  type: 2,
  chainId: 11155111,
  nonce: 63,
  maxPriorityFeePerGas: BigNumber { _hex: '0x59682f00', _isBigNumber: true },
  maxFeePerGas: BigNumber { _hex: '0x62151d24', _isBigNumber: true },
  gasPrice: null,
  gasLimit: BigNumber { _hex: '0x52e4', _isBigNumber: true },
  to: '0xB9067eA7df746A653Ac86D31b8361FE34048e655',
  value: BigNumber { _hex: '0x16bcc41e9000', _isBigNumber: true },
  data: '0x',
  accessList: [],
  hash: '0x620152240553caaf051d47b110a99bd2717199ee2615df670a5a17d67e8e3cb8',
  v: 0,
  r: '0x4861d80001fbc2a32ca980625c08a7b61f828599aa23b92143e2b808815b5d30',
  s: '0x66826496beb734a40b41e36d403c138152993d36cadbade3dc32f428fc9ce6a0',
  from: '0x6E8b56414035d7F8C6635A911beDb1044203fAf6',
  confirmations: 0,
  wait: [Function (anonymous)]
}
```

Listing 4.13 Validasi transaksi pembeli ke kontrak pintar

11. Lalu server akan mengirimkan transaksi dengan menggunakan *provider* alchemy dan status transaksi akan ditampilkan pada console.log, seperti pada listing 4.7.7
12. Setelah didapatkan Txhash dari bukti transaksi, Txhash akan dijadikan acuan dalam bukti transaksi dan *request* pada listing 4.13 akan ditambahkan struktur Txhash sebagai validasi.

```
const contractABI = [
    "function transferToSeller() public",
    "function getBalance() public view returns (uint256)"
];
```

Listing 4.14. ABI server untuk berkomunikasi dengan kontrak pintar

```
// Memanggil fungsi transferToSeller
const contract = new ethers.Contract(contractAddress, contractABI, buyerWallet.connect(provider));
```

Listing 4.15. Fungsi untuk memanggil ABI dari kontrak pintar melalui provider alchemy

13. Setelah aset masuk ke kontrak pintar, server akan langsung memanggil fungsi `transferToSeller` seperti pada program kontrak pintar di listing 4.15 yang menggunakan ABI (Application Binary Interface) di listing 4.14 untuk berkomunikasi dengan kontrak pintar. Fungsi `getBalance()`pada di sini digunakan untuk memantau apakah ada

saldo pada kontrak yang tertahan dan dimanfaatkan untuk proses pemantauan.

```
Gas cost for seller transaction: 0.000002523016567146 ETH
Transfer to seller transaction: {
  type: 2,
  chainId: 11155111,
  nonce: 64,
  maxPriorityFeePerGas: BigNumber { _hex: '0x59682f00', _isBigNumber: true },
  maxFeePerGas: BigNumber { _hex: '0x61c2265c', _isBigNumber: true },
  gasPrice: null,
  gasLimit: BigNumber { _hex: '0x8ab3', _isBigNumber: true },
  to: '0xB9067eA7df746A653Ac86D31b8361FE34048e655',
  value: BigNumber { _hex: '0x00', _isBigNumber: true },
  data: '0xeac42867',
  accessList: [],
  hash: '0xe273d9d1565ddc166c375e0e8db615d88d50a80181d022ae3447c2ae3b655799',
  v: 0,
  r: '0xd662cb92ea1ab6632be10ec1cc9a0f9a860a0402def068ff213210e0bb5be243',
  s: '0x44af7964be971fa515b93c67f51859c99f10c16541ef4a77818b97a8f07bc774',
  from: '0x6E8b56414035d7F8C6635A911beDb1044203fAf6',
  confirmations: 0,
  wait: [Function (anonymous)]
}
```

Listing 4.16 Output kontrak pintar ke dompet penjual

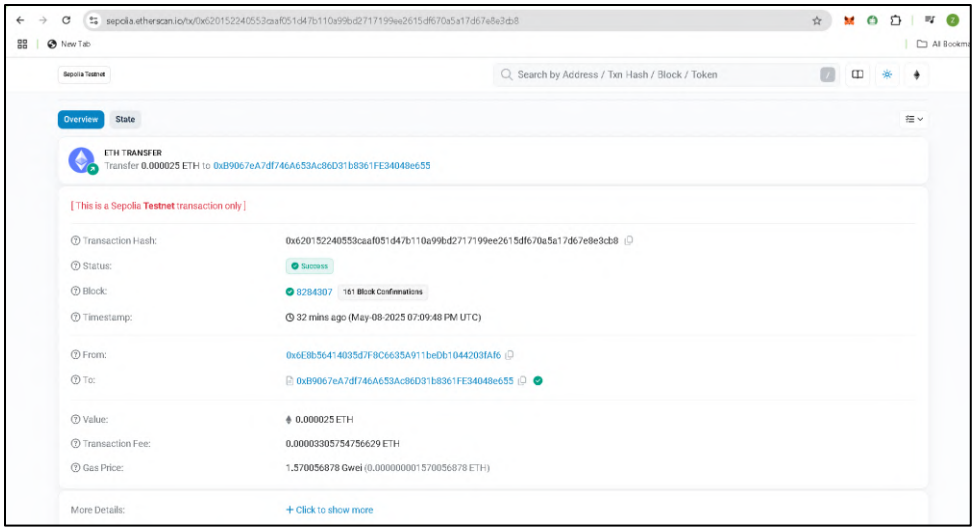
14. *Output* untuk transaksi kedua atau kontrak pintar ke penjual akan ditampilkan oleh server ke dalam `console.log` seperti pada listing 4.16
15. Jika transaksi sudah sampai pada listing 4.16, maka transaksi dinyatakan berhasil oleh server

```
_id: ObjectId('681d01e04504397d37ced32b')
address : "0x6E8b56414035d7F8C6635A911beDb1044203fAf6"
energyAmount : "1"
hash_txid : "0x620152240553caaf051d47b110a99bd2717199ee2615df670a5a17d67e8e3cb8"
timestamp : "2025-05-09T02:11:28.778Z"
```

Listing 4.17. Payload tervalidasi

16. Kemudian server akan menggabungkan semuanya dan disimpan ke dalam *database* mongodb dan server akan memberikan payload tervalidasi seperti listing 4.17 yang memiliki atribut penting berupa alamat pembeli, jumlah produk (contoh jumlah energi), Txhash, dan waktu terjadinya transaksi

4.8 Validasi Transaksi Eksternal



Gambar 4.10 Validasi dari Etherscan

Validasi secara *eksternal* dapat dilakukan menggunakan website di Etherscan sesuai gambar 4.10. Pada *website* Etherscan pihak developer telah menyediakan fasilitas untuk melihat Txhash yang tervalidasi dan pending. Jika sudah terbaca oleh Etherscan, transaksi akan dinyatakan sah dan tervalidasi valid. Pada contoh penelelitian ini TxHash memiliki isi:

“0x620152240553caaf051d47b110a99bd2717199ee2615df670a5a17d67e8e3cb8”

Isi TxHash ini sesuai dengan TxHash yang dihasilkan oleh server setelah proses *request* dan sesuai dengan Etherscan di *ledger* atau buku besar transaksi sepolia.

4.9 Transaksi yang Tercatat Server

Berikut ini adalah transaksi yang telah tercatat oleh sistem dan tervalidasi baik secara *on-chain* ataupun *off-chain.* Tabel 4.6 adalah daftar dompet dan riwayat transaksi yang sah dan tercatat sesuai listing 4.18 hingga 4.20

Listing 4.18 Konsumer 1

Listing 4.19 Konsumer 2

Listing 4.20 Konsumer 3

Tabel 4.6 Tabel daftar transaksi

Konsumer	Dompet	Transaksi	Jumlah Energi (Wh)	ETH
1	0x9A43b29497aebF8691E8Fa36760d7F57222b86e4	0x3231ba353edfbd824f8b3034206c85c95d048124bd50ee70f739cda58c7c0124	10000	0.00025
2	0x587a768aa01bd43f8Aa720A4539DAF8CB96CB979	0xed84c4384d0ea101e8a7ff321d58e767ea9f9f406b517f45c98488ebeb8002ac	1000	0.000025
3	0x6E8b56414035d7F8C6635A911beDb1044203fAf6	0x7a5d60ca5a3928739270dfcb933fdaefb34b0fa50972c30f587501c6cf88aa6e	2	0.00000005

Transaksi yang tercatat oleh server seperti tabel 4.6 merupakan transaksi yang sah dan tervalidasi oleh *validator* Ethereum dan telah ditulis

pada buku besar atau *ledger* di Ethereum pada jaringan Ethereum Sepolia dan tidak dapat diganti ataupun dimanipulasi.

4.10 Pengujian Beberapa Transaksi

Pengujian yang ditampilkan pada Tabel 4.7 bertujuan untuk mengukur waktu tunda (delay) komunikasi antara sistem *on-chain* dan *off-chain* dalam proses transaksi berbasis *smart contract* serta menilai tingkat keberhasilan transaksi. Setiap transaksi diuji menggunakan wallet dari konsumen satu dengan 10 variasi jumlah transfer (ETH) yang berbeda.

Tabel 4.7 Pengujian transaksi dan delay transaksi

No	Jumlah (ETH)	Tx1 (Buyer to Smart Contract)		Tx2 (Smart Contract to Seller)		Total delay (ms)
		Hash1	Delay (ms)	Hash2	Delay (ms)	
1	0.00002	0xb0952f2c67dfd36a590ea59622459b84bb063614c55349352706dadd6ae9f424	5217	0x6dd83ef5a69fe1431495ee50f6a29412b9c19d50bfb84c3510fa77e2624265a3	12863	18080
2	0.0000125	0xe06f0e40cf3a13706ae5eadc098baa394ae5d9b42a7f1047a21a9c88341248bc	9017	0xf1ccfc01e569b9a466db086cdd418bcd69b5dc73696b9b9dc4228be642aedee7	12827	21844
3	0.000009775	0xf646bec88d297ae16abadfa3a06e722db63d5599c8f2946e8c48fcc3c90a41d4	13800	0xd98f0205d4f823e65c07f8d1cb00e6c9974737fc3fa0349731b483ee1f458b8f	13004	26804
4	0.00003	0xfa59f9023394c4902c73e214b117d5d4c21c9fc8b467df14694cf71a2ecad335	5489	0x514172d4ab4a69bce69f0132e2b65c56ca8d6474600f2220935a600e5192af08	13451	18940
5	0.00025	0x6be7754c7a4c1f49cd6f1f7712bd2c964ec3822a9e8953df4d0d1c9edee3516c	9612	0x2123c325ef5e878bf05f232f2b1d11022654ed524a2a865ad9f5fddc3318c700	13058	22670
6	0.000525	0xcf988bb3884cee3b69ba7ef6107b7e6ccbfb8f252f1097a89ebdfaaf2eb382dd	13959	0xbb6b8213237f1b2dc62ba66a2a0977f2f0477ddb94ddcaab5ce12da76dc283cd	9622	23581
7	0.000075	0xb2546332d8c1fbde69655483cc37734fe9c48d8abc07735830d197bc50c1c083	9063	0xf20df1f142274926e3011bae94798f6b3f0f7f14a535acfb7f7b5d3bd1bc8eac	13179	22242
8	0.0001375	0x113ee3e1114fccf0f1425c39945411ccb6066f9f465936a824fa1250f521a140	5097	0xb1929ddefc976396f29ae79793980a4cfbe52f6352bf4097134c3bcde10d916d	13093	18190
9	0.000000125	0xf825b949ca3c3665a011905449a4234f4d82a756d20f31eacacfd9c73390a2f7	17141	0x5bed05bc93e4a9cad0b091a975a5000ecf58293ff394ae3f0a991fc633d12c69	9049	26190
10	0.001356175	0x8a9492c67a99373f1c40766dd84dd3c7b872ac1ee9d18240f8010b5888aa463b	17346	0x2c4173394c1a7a8135c8afcb4bffb8f144679a4e122f8f75739d09edcca2592d	8872	26218

Berdasarkan hasil pengujian, seluruh transaksi yang dilakukan berhasil diproses, sebagaimana ditunjukkan oleh adanya hash pada setiap transaksi (Tx1 dan Tx2). Hal ini menandakan bahwa tingkat keberhasilan transaksi mencapai 100%.

Secara lebih rinci, delay transaksi dapat dianalisis sebagai berikut:

1. ***Delay* Tx1 (*Buyer → Smart Contract*)**

   Waktu tunda pada tahap Tx1 menunjukkan variasi dengan rentang antara 5.097 ms hingga 17.346 ms dan memiliki rata-rata 10.574 ms.

2. ***Delay* Tx2 (*Smart Contract → Seller*)**

   Pada tahap Tx2, delay yang terjadi relatif lebih konsisten dibanding Tx1, yaitu berkisar antara 8.872 ms hingga 13.863 ms dan memiliki rata-rata 11.902 ms.

3. **Total *Delay* Transaksi**

   Total waktu yang dibutuhkan untuk menyelesaikan satu transaksi, mulai dari konsumen melakukan pembayaran hingga penjual menerima konfirmasi, berada pada kisaran 18.080 ms hingga 26.218 ms. Rata-rata waktu eksekusi transaksi adalah sekitar 22.476 ms.

4. **Pengaruh Jumlah Transaksi**

   Berdasarkan hasil pengujian, jumlah ETH yang ditransaksikan tidak menunjukkan pengaruh signifikan terhadap waktu delay. Sebagai contoh, transaksi dengan nilai sangat kecil (0.000000125 ETH) membutuhkan total waktu 26.190 ms, sedangkan transaksi dengan nilai terbesar (0.001356175 ETH) membutuhkan waktu yang hampir sama, yaitu 26.218 ms. Dengan demikian, dapat disimpulkan bahwa besar kecilnya nominal transaksi tidak menjadi faktor penentu waktu delay.

4.11 Pembacaan Modbus PZEM-004t

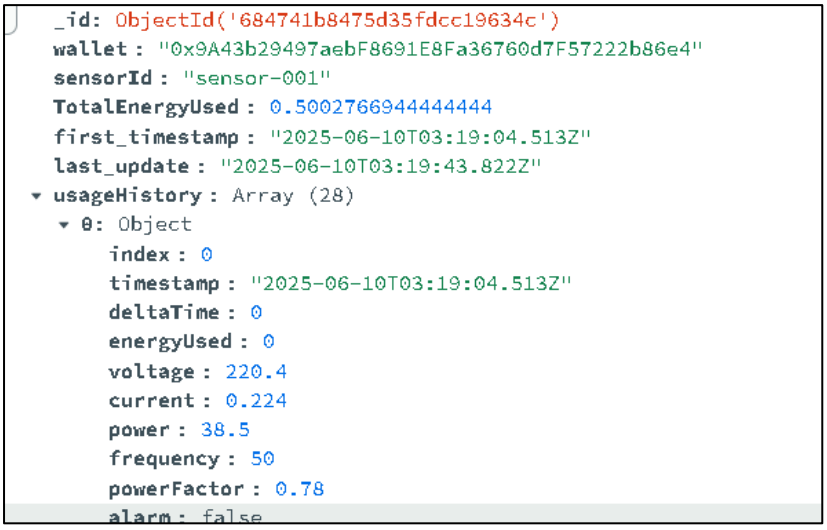
Pembacaan modbus PZEM-004t dilakukan dengan menggunakan komunikasi serial pada RX TX di pin 16 dan 17 pada mikrokontroler ESP32.

4.12 Data Handling Sensor

Data Handling Sensor atau penanganan pembacaan sensor akan dilakukan oleh server. Data sensor PZEM akan dikirim melalui ESP32 melalui protocol komunikasi serial dan langsung diteruskan oleh program *handling* ke server di *endpoint* "/*logEnergyConsumption*" dalam bentuk *payload* yang terformat, contoh

"sensorId": "sensor-001",
"wallet": "0x9A43b29497aebF8691E8Fa36760d7F57222b86e4",
"voltage": 220.4,
"current": 0.224,
"power": 38.5,
"Energy": 2,
"frequency": 50,
"powerFactor": 0.76
… *payload* dompet 2, dan *payload* dompet 3

Wallet atau dompet dan sensorId yang ada pada *payload* digunakan untuk pengenalan oleh server dan akan tercatat di *database* sesuai dengan wallet. Seperi pada list 4.21.

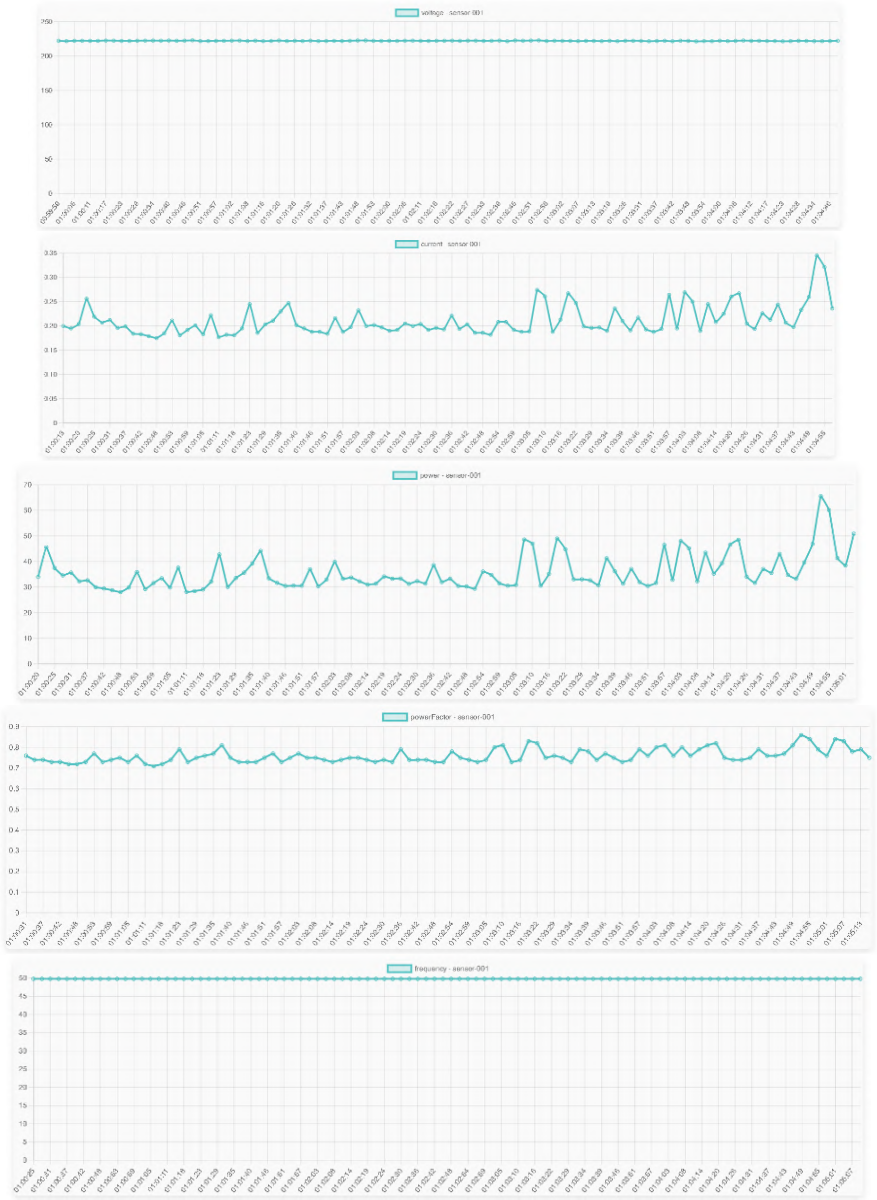


Listing 4.21 Isi general payload

Setelah itu beberapa isi dari *payload* yang akan diambil untuk melakukan monitoring adalah sebagai berikut

"voltage": 220.4,
"current": 0.224,
"power": 38.5,
"frequency": 50,
"powerFactor": 0.76

Payload tersebut akan digunakan untuk melakukan monitoring seperti pada contoh gambar 4.11



Gambar 4.11 Monitoring

Monitoring tersebut menggunakan 100 data terakhir yang telah dikirimkan oleh sensor PZEM dan hanya dapat diakses oleh seller dengan login terlebih dahulu.

4.13 Manajemen Energi dan Hak Energi Konsumen

Manajemen energi menggunakan payload yang telah dikirim yaitu menggunakan *power* atau daya yang dikalikan dengan waktu sehingga menghasilkan energi

$$\mathrm{E\ (Joule)} = \mathrm{P\ (W)} \times \mathrm{t\ (s)} \tag{4.1}$$

Dimana:

E = Energi (Wh)

P = Power atau daya (W)

t = Waktu (s)

Power = 73

E(Joule) = 73 w

E(Joule) = 73 Joule

1 Wh = 3600 Joule

$$\mathrm{Energi\ (Wh)} = \frac{\mathrm{Energi\ (J)}}{3600} \tag{4.2}$$

$$\mathrm{Energi\ (Wh)} = \frac{73}{3600} \tag{4.3}$$

$$\mathrm{Energi\ (Wh)} \approx 0.020277 \tag{4.4}$$

```
index : 13
timestamp : "2025-06-16T17:20:20.896Z"
deltaTime : 0.998
energyUsed : 0.020237222222222222
voltage : 221.9
current : 0.376
power : 73
frequency : 49.8
powerFactor : 0.87
alarm : false
```


Listing 4.22 Data energyUsed

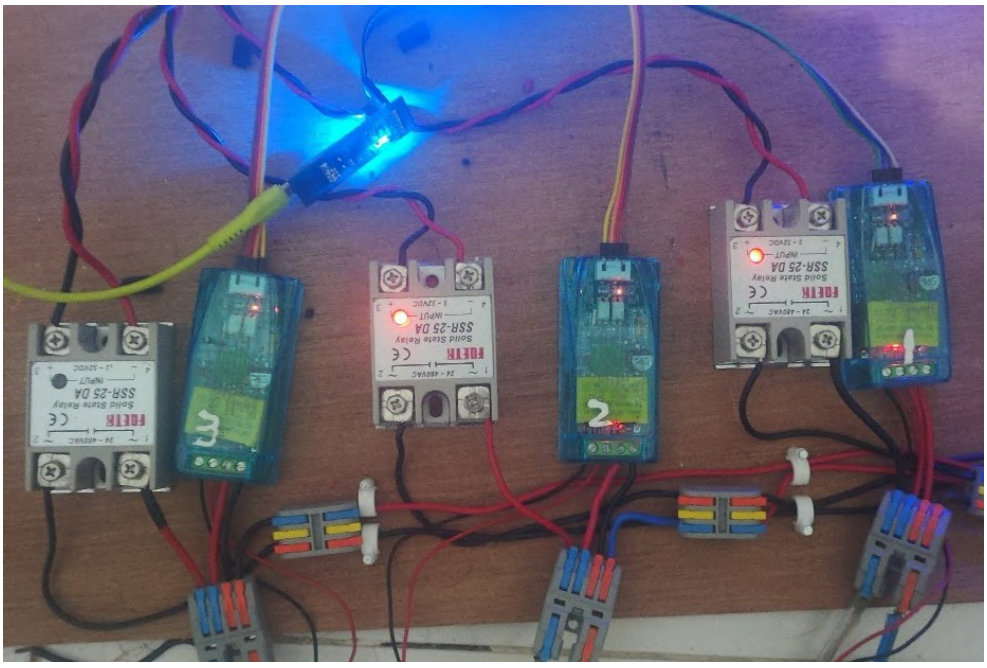
Jika dilihat dan dibandingkan dengan *energyUsed* seperti listing 4.22 dan hasil perhitungan manual, maka memiliki nilai yang mirip. Hal ini dapat digunakan dalam monitoring secara presisi dan dapat digunakan untuk pencatatan.


```
_id: ObjectId('6822082d112977eac9221226')
wallet : "0x9A43b29497aebF8691E8Fa36760d7F57222b86e4"
RemainingEnergy : 9979.605526166666
lastUpdated : "2025-06-10T03:45:08.198Z"
relayStatus : "open"

_id: ObjectId('68220bd9112977eac9221227')
wallet : "0x587a768aa01bd43f8Aa720A4539DAF8CB96CB979"
RemainingEnergy : 988.3294258888889
lastUpdated : "2025-06-10T03:45:03.877Z"
relayStatus : "open"

_id: ObjectId('68471d9ce10e99492f3c92c2')
wallet : "0x6E8b56414035d7F8C6635A911beDb1044203fAf6"
RemainingEnergy : 0
lastUpdated : "2025-06-10T03:30:16.052Z"
relayStatus : "closed"
```


Listing 4.23 Daftar Hak konsumer

Listing 4.23 adalah daftar yang merinci hak-hak konsumen terkait dengan pengelolaan energi dan status relay dari dompet digital (wallet) mereka. Setiap entri dalam daftar ini menyajikan informasi penting seperti alamat dompet, sisa energi yang tersedia, waktu terakhir pembaruan data, dan status relay (terbuka atau tertutup). Informasi ini memberikan gambaran transparan mengenai kepemilikan dan kondisi operasional dompet konsumen, yang esensial untuk memastikan hak mereka dalam penggunaan dan akses terhadap layanan terkait.

Jika konsumer masih memiliki sejumlah *RemainingEnergy,* seperti pada pada dompet 1 (0x9A43b29497aebF8691E8Fa36760d7F57222b86e4) dan dompet 2 (0x587a768aa01bd43f8Aa720A4539DAF8CB96CB979) maka konsumer masih berhak mendapatkan energi, sedangkan jika konsumer tidak memiliki hak energi seperti yang terlihat pada dompet konsumer 3 di listing 4.23 (0xb173e4E236d596C6D78D6183f28Bc75599c65cd8), maka konsumer tidak memiliki hak energi.



Gambar 4.12 Relay aktif dan mati sesuai daftar hak energi

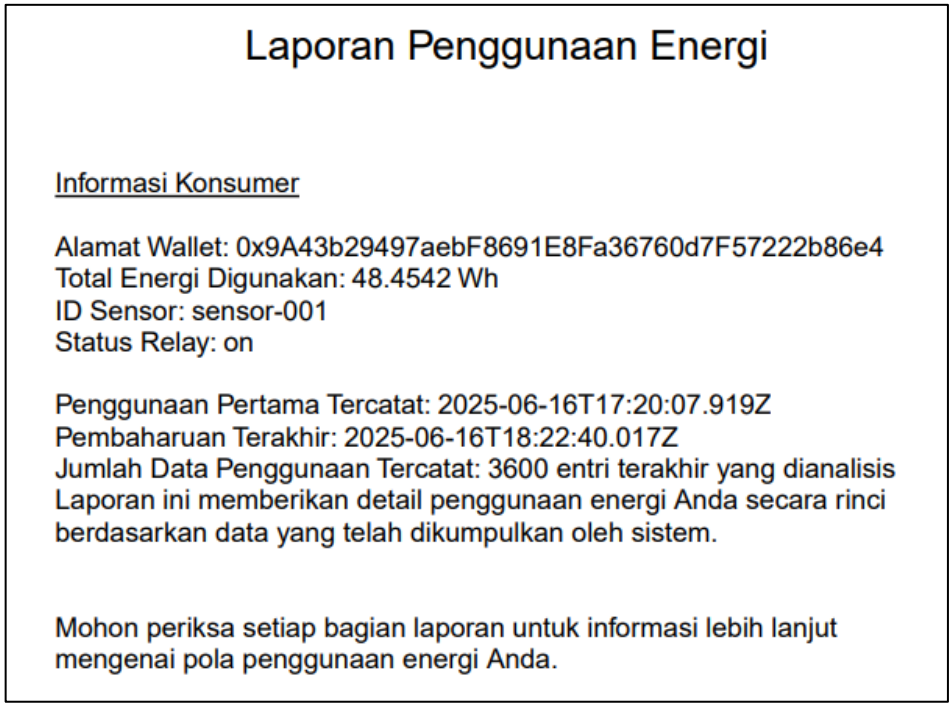
Status relay akan diperbaharui melalaui *endpoint* "/*relayStatus*" oleh server yang menggunakan metode "*get*" lalu akan dipantau statusnya oleh ESP32 di *endpoint* tersebut untuk memperbaharui relay konsumer sesuai dompet konsumer seperti gambar 4.12 yang dimana relay konsumer 1 masih menyala, relay konsumer 2 masih menyala, dan relay konsumer 3 tidak menyala. Hal ini sesuai dengan listing 4.23.

4.14 Laporan Untuk Konsumer dan Regresi Linear

Laporan yang digunakan pada bagian ini merupakan laporan dari alamat dompet "0x9A43b29497aebF8691E8Fa36760d7F57222b86e4" lalu laporan diolah dan dapat diunduh oleh konsumer, yaiut perihal laporan tentang informasi dan laporan penting yang terdiri dari 3 bab, yaitu

1. Bab 1 tentang Laporan Penggunaan Energi
2. Bab 2 tentang Detail Penggunaan Energi
3. Bab 3 tentang *Summary* atau hasil

4.15 Bab 1 Laporan Penggunaan Energi



Gambar 4.13 Bab 1

Pada bab seperti gambar 4.13 ini mengandung informasi umum yang disertakan untuk mengetahui:

1. Alamat Wallet0x9A43b29497aebF8691E8Fa36760d7F57222b86e4
2. Total Energi Digunakan: 48.4542 (Wh)
3. ID Sensor: sensor-001
4. Status relay: on
5. Penggunaan Pertama Tercatat: 2025-06-16T17:20:07.919Z
6. Pembaharuan Terakhir: 2025-06-16T18:22:40.017Z
7. Jumlah Data Penggunaan Tercatat: 3600 entri

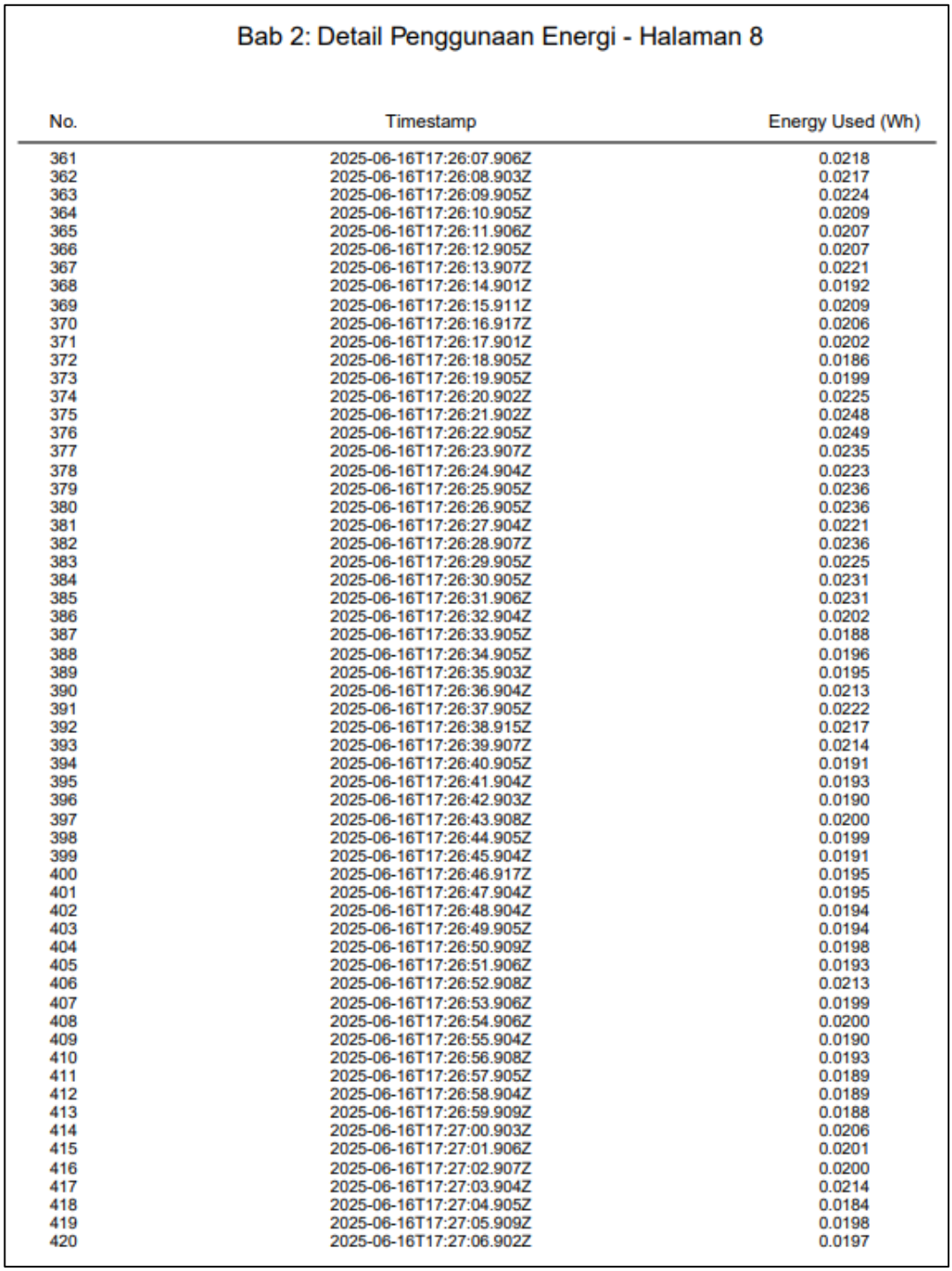
4.16 Bab 2 Detail Penggunaan Energi

Pada bab 2 seperti gambar 4.14 ini mengandung informasi umum yang disertakan untuk mengetahui detail penggunaan energi yang digunakan oleh konsumer dengan rentang data maksimal 3600 entri data terakhir. 3600 data terakhir ini sudah ditentukan dari server untuk membatasi kerja server agar tidak berlebih.

Gambar 4.14 mengandung informasi pembacaan server meliputi:

1. Nomer pembacaan
2. Waktu
3. Jumlah energi yang dikonsumsi

Hal ini ditujukan agar konsumer dapat melihat secara detail penggunaan energi mereka dalam kurun waktu yang telah ditentukan oleh server atau kurang lebih dalam 1 jam.



Bab 2: Detail Penggunaan Energi - Halaman 8

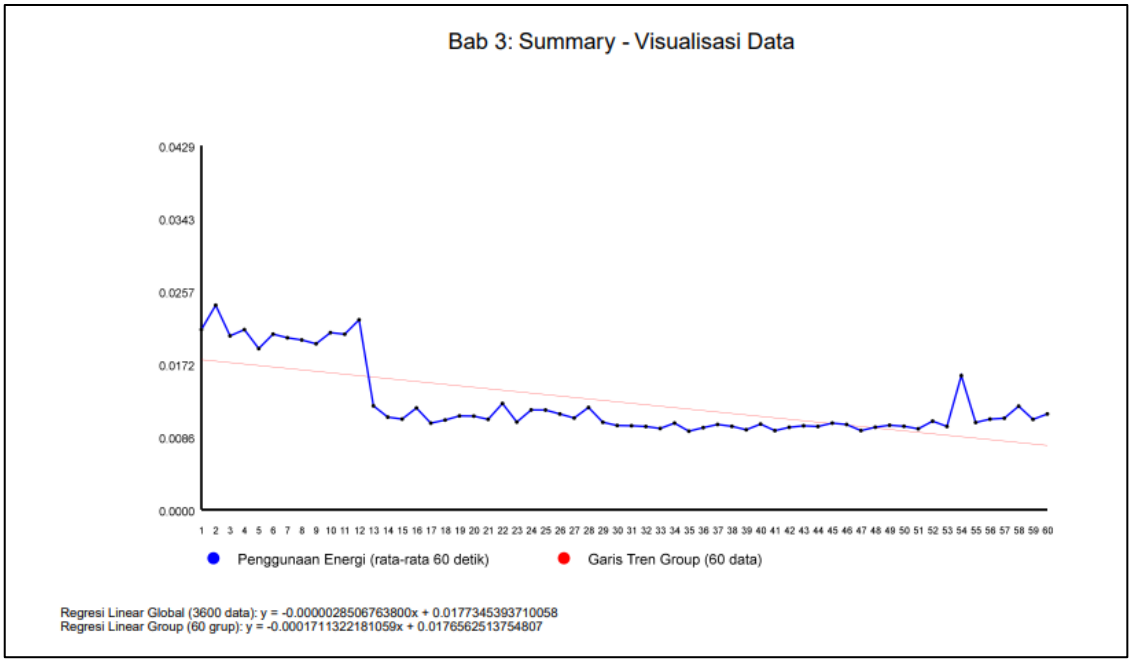
No.	Timestamp	Energy Used (Wh)
361	2025-06-16T17:26:07.906Z	0.0218
362	2025-06-16T17:26:08.903Z	0.0217
363	2025-06-16T17:26:09.905Z	0.0224
364	2025-06-16T17:26:10.905Z	0.0209
365	2025-06-16T17:26:11.906Z	0.0207
366	2025-06-16T17:26:12.905Z	0.0207
367	2025-06-16T17:26:13.907Z	0.0221
368	2025-06-16T17:26:14.901Z	0.0192
369	2025-06-16T17:26:15.911Z	0.0209
370	2025-06-16T17:26:16.917Z	0.0206
371	2025-06-16T17:26:17.901Z	0.0202
372	2025-06-16T17:26:18.905Z	0.0186
373	2025-06-16T17:26:19.905Z	0.0199
374	2025-06-16T17:26:20.902Z	0.0225
375	2025-06-16T17:26:21.902Z	0.0248
376	2025-06-16T17:26:22.905Z	0.0249
377	2025-06-16T17:26:23.907Z	0.0235
378	2025-06-16T17:26:24.904Z	0.0223
379	2025-06-16T17:26:25.905Z	0.0236
380	2025-06-16T17:26:26.905Z	0.0236
381	2025-06-16T17:26:27.904Z	0.0221
382	2025-06-16T17:26:28.907Z	0.0236
383	2025-06-16T17:26:29.905Z	0.0225
384	2025-06-16T17:26:30.905Z	0.0231
385	2025-06-16T17:26:31.906Z	0.0231
386	2025-06-16T17:26:32.904Z	0.0202
387	2025-06-16T17:26:33.905Z	0.0188
388	2025-06-16T17:26:34.905Z	0.0196
389	2025-06-16T17:26:35.903Z	0.0195
390	2025-06-16T17:26:36.904Z	0.0213
391	2025-06-16T17:26:37.905Z	0.0222
392	2025-06-16T17:26:38.915Z	0.0217
393	2025-06-16T17:26:39.907Z	0.0214
394	2025-06-16T17:26:40.905Z	0.0191
395	2025-06-16T17:26:41.904Z	0.0193
396	2025-06-16T17:26:42.903Z	0.0190
397	2025-06-16T17:26:43.908Z	0.0200
398	2025-06-16T17:26:44.905Z	0.0199
399	2025-06-16T17:26:45.904Z	0.0191
400	2025-06-16T17:26:46.917Z	0.0195
401	2025-06-16T17:26:47.904Z	0.0195
402	2025-06-16T17:26:48.904Z	0.0194
403	2025-06-16T17:26:49.905Z	0.0194
404	2025-06-16T17:26:50.909Z	0.0198
405	2025-06-16T17:26:51.906Z	0.0193
406	2025-06-16T17:26:52.908Z	0.0213
407	2025-06-16T17:26:53.906Z	0.0199
408	2025-06-16T17:26:54.906Z	0.0200
409	2025-06-16T17:26:55.904Z	0.0190
410	2025-06-16T17:26:56.908Z	0.0193
411	2025-06-16T17:26:57.905Z	0.0189
412	2025-06-16T17:26:58.904Z	0.0189
413	2025-06-16T17:26:59.909Z	0.0188
414	2025-06-16T17:27:00.903Z	0.0206
415	2025-06-16T17:27:01.906Z	0.0201
416	2025-06-16T17:27:02.907Z	0.0200
417	2025-06-16T17:27:03.904Z	0.0214
418	2025-06-16T17:27:04.905Z	0.0184
419	2025-06-16T17:27:05.909Z	0.0198
420	2025-06-16T17:27:06.902Z	0.0197

Gambar 4.14 Detail Penggunaan Energi

4.17 Bab 3 Summary (Hasil Akhir)

Pada Bab 3 laporan terdapat 3 bagian, yaitu Visualisasi Data, Summary – Hasil, dan Rata-rata Tiap Grup.

4.17.1 Bab 3 Summary – Visualisasi Data



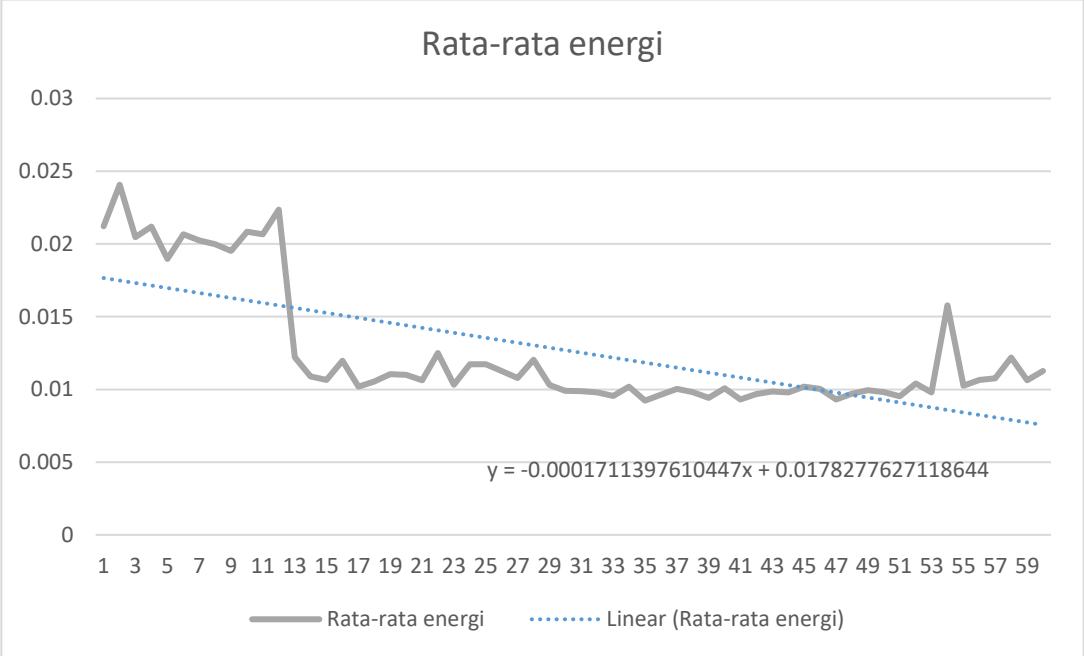
Gambar 4.15 Visualisasi Data

Gambar 4.15 merupakan bentuk visualisasi sebuah data yang telah dikelompokkan dengan cara menggunakan pembagian tiap 60 data dan memiliki jumlah maksimal 60 data pada grafik.

Terdapat 2 jenis regresi linear yang ditulisakan dan digambarkan, yaitu persamaan regresi linear global dan kelompok. Regresi linear kelompok menggunakan hasil dari perhitungan 60 kelompok sebuah data dan regresi linear global menggunakan perhitungan dari data global atau keseluruhan sebanyak 3.600 data.

Pendekatan ini diperlukan untuk mengatasi beban server. Oleh karena itu, 3.600 data yang diterima tidak merepresentasikan 3.600 detik atau 1 jam secara langsung, melainkan merupakan 3.600 entri data yang dikirim oleh mikrokontroler dengan total waktu dihitung berdasarkan penjumlahan Δt dari setiap pembacaan yang berhasil diterima oleh server.

Jika dilakukan perhitungan data kelompok secara manual pada perangkat lunak excel akan menghasilkan grafik seperti pada gambar 4.16



Gambar 4.16 Grafik energi dari perangkat lunak excel

Grafik pada gambar 4.16 menggunakan data excel yang dimasukkan secara manual dari pembacaan energi oleh PZEM di *database*. Grafik menunjukkan nilai regresi kelompok yang mirip yaitu:

y = -0.0001711322181059x + 0.0176562513754807 (server)

y = -0.0001711397610447x + 0.0178277627118644 (excel)

jika dihitung persen errornya adalah

Slope server = -0.0001711322181059

Slope excel = -0.0001711397610447

Intercept server = 0.0176562513754807

Intercept excel = 0.0178277627118644

$$\%\text{Error} = \frac{|\text{server} - \text{excel}|}{\text{excel}} * 100\% \qquad (4.5)$$

1. Persen Error Slope

$$\%\text{Error} = \frac{|-0.000171132-(-0.000171139)|}{-0.0001711397610447} * 100\% \quad (4.6)$$

$$\%\text{Error} = 0.44\% \quad (4.7)$$

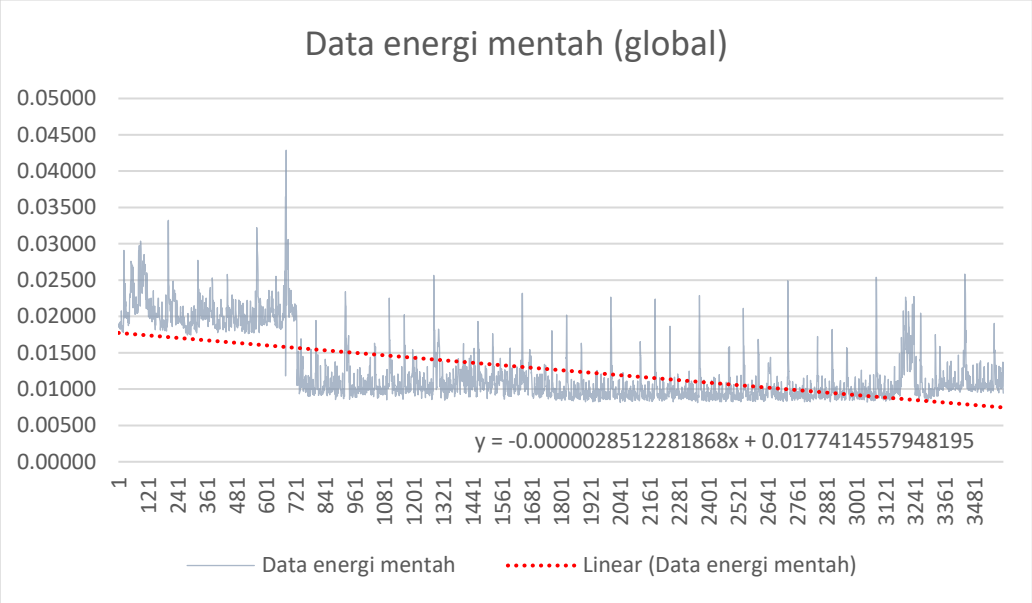
2. Persen Error Intercept

$$\%\text{Error} = \frac{|0.017656251-0.017827762|}{0.0178277627118644} * 100\% \quad (4.8)$$

$$\%\text{Error} = 0.96\% \quad (4.9)$$

Dengan demikian error yang diperoleh antara perhitungan kelompok server dan perangkat lunak excel adalah 0.44% untuk slope (m) dan 0.96% untuk intercept

Selanjutnya adalah persamaan regresi linear secara global atau keseluruhan. Jika dilakukan perhitungan data global secara manual pada perangkat lunak excel akan menghasilkan grafik seperti pada gambar 4.17



Gambar 4.17 Data energi mentah (global)

Grafik pada gambar 4.17 menggunakan data excel yang dimasukkan secara manual dari pembacaan energi oleh PZEM di *database*. Grafik menunjukkan nilai regresi yang mirip yaitu:

y = -0.0000028506763800x + 0.0177345393710058 (server)

y = -0.0000028512281868x + 0.0177414557948195 (excel)

jika dihitung % errornya adalah

Slope server = -0.0000028506763800

Slope excel = -0.0000028512281868

Intercept server = 0.0177345393710058

Intercept excel = 0.0177414557948195

$$\%\text{Error} = \frac{|\text{server} - \text{excel}|}{\text{excel}} * 100\% \quad (4.10)$$

3. Persen Error Slope

$$\%\text{Error} = \frac{|-0.000002850 - (-0.000002851)|}{-0.0000028512281868} * 100\% \quad (4.11)$$

$$\%\text{Error} = 0.19\% \quad (4.12)$$

4. Persen Error Intercept

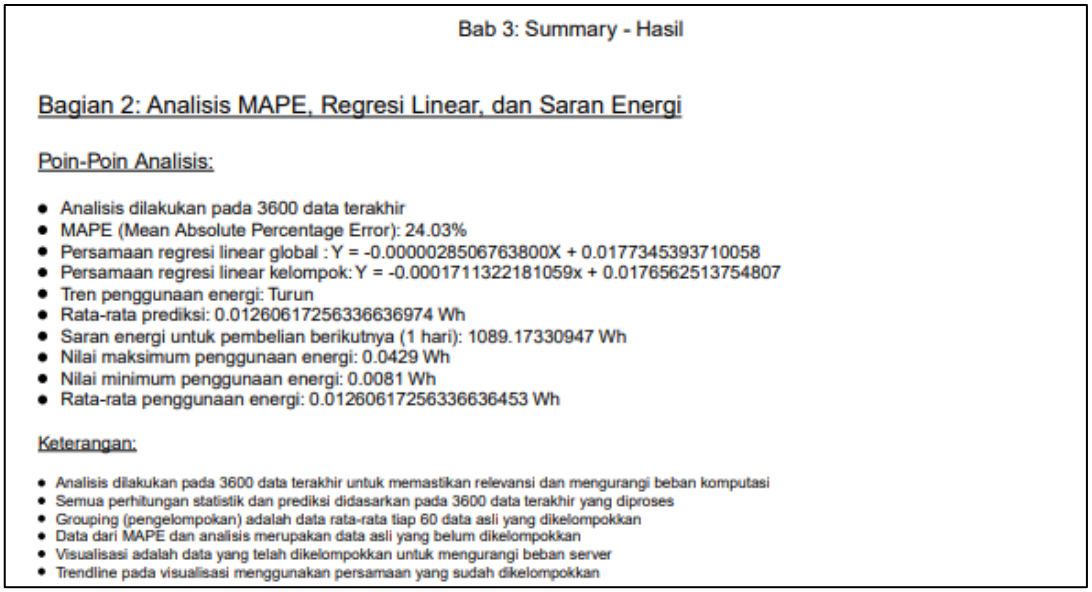
$$\%\text{Error} = \frac{|0.017656251 - 0.017827762|}{0.0178277627118644} * 100\% \quad (4.13)$$

$$\%\text{Error} = 0.39\% \quad (4.14)$$

Dengan demikian % error yang diperoleh antara perhitungan global server dan perangkat lunak excel adalah 0.19% untuk slope (m) dan 0.39% untuk intercept

Setelah dilakukan perhitungan, maka dapat disimpulkan bahwa persamaan regresi liniear antara perhitungan manual dengan excel dan server telah sama karena memiliki persentase error yang sangat kecil, yaitu di bawah 1% baik data kelompok maupun global.

4.17.2 Bab 3 Summary – Hasil



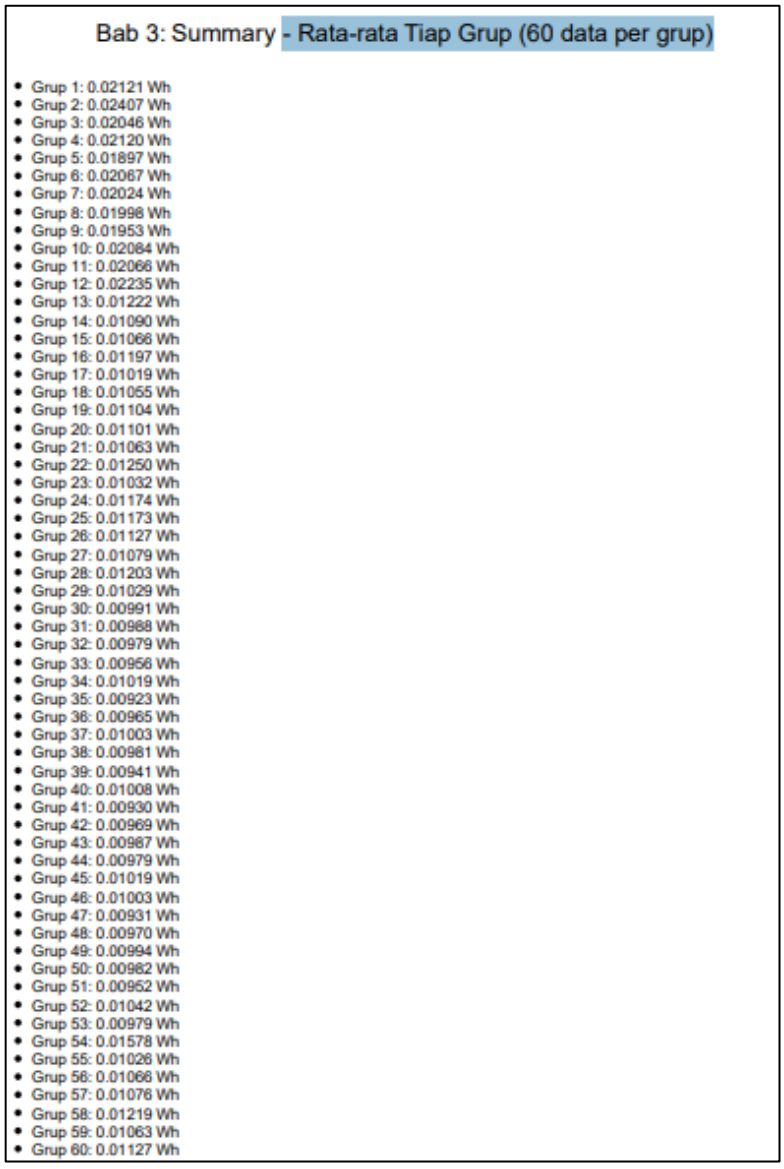
Gambar 4.18 Bab 3: Summary – Hasil

Gambar 4.18 merupakan hasil yang dibuat oleh server yang berisi:

1. Jumlah data entri
2. MAPE (Mean Absolute Percentage Error)
3. Persamaan regresi linear global
4. Persamaan regresi linear kelompok
5. Tren penggunaan energi
6. Rata-rata prediksi
7. Saran energi untuk pembelian berikutnya (1 hari)
8. Nilai maksimum penggunaan energi
9. Nilai minimum penggunaan energi
10. Rata-rata penggunaan energi

4.17.3 Bab 3 Summary – Rata-rata Tiap Grup (60 data per grup)

Pada gambar 4.19 adalah nilai rata-rata tiap grup berisi 60 data dan server membatasi sebanyak 60 grup sehingga total yang di *handle* oleh server adalah 3.600 data.



Gambar 4.19 Nilai rata-rata group

4.18 Saran Pembelian Energi Dalam Satu Hari Kedepan

Saran pembelian pada bagian ini menggunakan saran pemberlian dari alamat dompet “0x9A43b29497aebF8691E8Fa36760d7F57222b86e4”

```
// Saran pembelian energi = total prediksi untuk 1 hari (dengan tren yang sama)
const nextDayPrediction = slope + intercept * (60*60*24);
```

Listing 4.24 Rumus saran satu hari kedepan

Proses pemberian saran untuk satu hari kedepan menggunakan regresi linear sesuai dengan listing 4.24.

```
const usageHistory = data.usageHistory.map((entry) => ({
    timestamp: entry.timestamp,
    energyUsed: entry.energyUsed,
}));
```

Listing 4.25 Pengambilan Data

Proses ini diawali dengan pengambilan data dari listing 4.25 yang mengambil data database “data.usageHistory”. Jadi hanya dua properti penting yang diambil, yaitu timestamp dan energyUsed.

```
const recentData = usageHistory.slice(-config.maxDataPoints);
const analysisData = recentData;
const actual = analysisData.slice(1).map((entry) => entry.energyUsed);
```

Listing 4.26 Pengambilan data terbaru

```
const config = {
    maxDataPoints: 3600,
    groupSize: 60,
    canvasWidth: 800,
    canvasHeight: 400,
    scaleFactor: 5,
```

Listing 4.27 Config yang dibuat di server

Listing 4.26 ini mengambil maxDataPoints terakhir dari usageHistory, artinya hanya fokus pada data terbaru saja, yaitu 3600 data terakhir jika config.maxDataPoints = 3600 sesuai listing 4.27

Baris selanjutnya tentang analysisData, menggunakan fungsi dari recentData dan akan dilanjut dengan fungsi *actual* yang bertugas memulai dari indeks pertama, bukan nol.

```
// Regresi Linear untuk semua data
const n = actual.length;
const sumX = Array.from({ length: n }, (_, i) => i).reduce((a, b) => a + b, 0);
const sumY = actual.reduce((a, b) => a + b, 0);
const sumXY = actual.reduce((sum, y, i) => sum + i * y, 0);
const sumX2 = Array.from({ length: n }, (_, i) => i ** 2).reduce((a, b) => a + b, 0);

const slope = ((n * sumXY - sumX * sumY) / (n * sumX2 - sumX ** 2));
const intercept = (sumY - slope * sumX) / n;
```

Listing 4.28 Parameter Regresi Linear

Berikut penjelasan listing 4.28

1. **Tujuan Regresi Linear**
   a. Regresi linear sederhana digunakan untuk mencari garis lurus terbaik yang merepresentasikan hubungan antara waktu (X) dan penggunaan energi (Y).
   b. Garis ini digunakan untuk mengetahui apakah energi cenderung meningkat, menurun, atau stabil dari waktu ke waktu.
2. **Menentukan Jumlah Data (n)**
   - Variabel n menyimpan jumlah data yang akan dianalisis, yaitu banyaknya nilai actual (energi digunakan).
   - Nilai n ini penting sebagai dasar semua perhitungan statistik selanjutnya.
3. **Menghitung Total Waktu dan Energi**
   - sumX: jumlah dari semua indeks waktu (1, 2, 3, ..., n-1), menunjukkan total "waktu virtual".
   - sumY: jumlah seluruh data actual, yaitu total energi yang digunakan selama periode data.
4. **Menghitung Hubungan Antara Waktu dan Energi**
   - sumXY: menjumlahkan hasil kali setiap waktu (indeks i) dan energi pada saat itu (actual[i]).
   - sumX2: menjumlahkan kuadrat dari setiap waktu (i^2), digunakan dalam rumus regresi untuk menghitung ketepatan kemiringan.
5. **Menghitung Kemiringan Garis (Slope)**
   Rumus:

$$\text{slope} = \frac{n * \sum XY - \sum X * \sum Y}{n * \sum X^2 - (\sum X)^2} \quad (4.15)$$

   Di mana:

   n adalah jumlah indeks

   ∑XY adalah hasil dari penjumlahan indeks (X) dikali dengan energi (Y)

   ∑X adalah hasil dari penjumlahan indeks atau X

   ∑Y adalah hasil dari penjumlahan nilai energi atau Y

$\sum X^2$ adalah hasil dari penguadratan nilai x atau indeks

$(\sum X)^2$ adalah hasil dari penguadratan yang berasal setelah penjumlahan nilai x atau indeks

6. Menghitung Titik Potong Y (Intercept)

   Rumus:

$$\text{intercept} = \frac{\sum Y - \text{slope} * \sum X}{n} \tag{4.16}$$

   Di mana:

   n adalah jumlah indeks

   $\sum Y$ adalah hasil dari penjumlahan nilai energi atau Y

   $\sum X$ adalah hasil dari penjumlahan indeks atau X

```
// Saran pembelian energi = total prediksi untuk 1 hari (dengan tren yang sama)
let totalPrediksi = 0;
for (let i = 0; i < n; i++) {
    const prediksiY = slope * i + intercept;
    totalPrediksi += prediksiY;
}
const rataRataPrediksi = totalPrediksi / n;

// Hitung saran pembelian untuk 1 hari berdasarkan rata-rata prediksi
let nextDayPrediction = rataRataPrediksi * 86400;
```

Listing 4.29 Penyaranan pembelian energi dalam satu hari berikutnya

Pada bagian Listing 4.29 ditunjukkan proses perhitungan saran pembelian energi berdasarkan pendekatan regresi linear. Data konsumsi energi selama satu jam terakhir digunakan sebagai dasar untuk membentuk model regresi dengan parameter slope dan intercept. Selanjutnya, dilakukan perhitungan rata-rata dari seluruh hasil prediksi regresi tersebut. Nilai rata-rata kemudian dikalikan dengan 86400 detik untuk memperoleh estimasi kebutuhan energi dalam satu hari ke depan. Berikut penjelasannya:

1. Perhitungan Nilai Prediksi per Indeks

   Sistem menghitung nilai prediksi konsumsi energi untuk setiap indeks waktu i dari 0 hingga n−1, di mana:

   1. y_i adalah konsumsi energi prediksi pada detik ke-i,
   2. m adalah slope dari regresi linear,
   3. c adalah intercept dari regresi linear.

2. Penjumlahan Semua Hasil Prediksi:
    a. Nilai prediksi dari semua indeks y_0, y_1, ... y_{n-1} dijumlahkan menggunakan perulangan *for*.
    b. Total ini disimpan dalam variabel totalPrediksi.
3. Penghitungan Rata-rata Prediksi:
    a. Rata-rata hasil prediksi dihitung dengan membagi totalPrediksi dengan jumlah data nnn.
    b. Nilai ini menggambarkan estimasi rata-rata konsumsi energi per detik berdasarkan tren data historis.
4. Perkalian dengan 86400 Detik:
    a. Rata-rata prediksi dikalikan dengan 86400 detik (jumlah detik dalam satu hari).
    b. Hasil perkalian ini menjadi estimasi total kebutuhan energi selama satu hari penuh.

Berikut jumlah saran pembelian dalam 1 hari berikutnya menggunakan perangkat lunak excel.

Tabel 4.8 Nilai mentah sensor

No	Timestamp	Energi	Prediksi	%error
1	2025-06-16T17:22:40.903Z	0.01865	0.017738605	4.89%
2	2025-06-16T17:22:41.900Z	0.01892	0.017735753	6.26%
...	...	...	...	...
3600	2025-06-16T18:22:40.017Z	0.00943	0.007477034	20.71%

Tabel 4.8 adalah tabel yang berisi nilai mentah dari sensor dalam satuan detik. Nilai ini selnajutnya akan diolah menggunakan regresi linear dengan cara mencari dua parameter terlebih dahulu, yaitu slope dan intercept.

Untuk slope dapat menggunakan "=SLOPE(C2:C3601;A2:A3601)" dan intercept menggunakan "=INTERCEPT(C2:C3601;A2:A3601)", dimana kolom A adalah indeks, nomor, atau x dan kolom C adalah energi atau y. Lalu diperoleh nilai seperti pada bab 4.13 'Bab 3 Summary – Visualisasi Data' yang sudah memiliki nilai intercept dan slope, yaitu

a. Slope server = -0.0000028506763800

b. Slope excel = -0.0000028512281868
c. Intercept server = 0.0177345393710058
d. Intercept excel = 0.0177414557948195.

Parameter-parameter tersebut lalu digunakan untuk melakukan perhitungan semua prediksi dengan "=intercept + slope * indeks". Lalu semua prediksi pada kolom D atau 'prediksi' akan dirata-rata dan akan menghasilkan rata-rata prediksi menggunakan formulasi "=AVERAGE(D2:D3601)". Hasil yang diperoleh adalah 0.0126078 Wh. Nilai tersebut akan dikalikan dengan penggunaan satu hari kedepan dengan persamaan

$$\frac{1}{n}\sum_{i=0}^{n-1}(slope * i + intercept) * 60 * 60 * 24 \quad (4.17)$$

$$\frac{1}{3600}\sum_{i=0}^{n-1}(-0.00000285 * i + 0.01774145) * 60 * 60 * 24 \quad (4.18)$$

Maka hasil yang diperoleh dari persamaan tersebut menggunakan excel adalah 1089.3156 Wh.

Berikut nilai perbandingan nilai saran pembelian dalam satu hari berikutnya menggunakan perangkat lunak excel dan server.

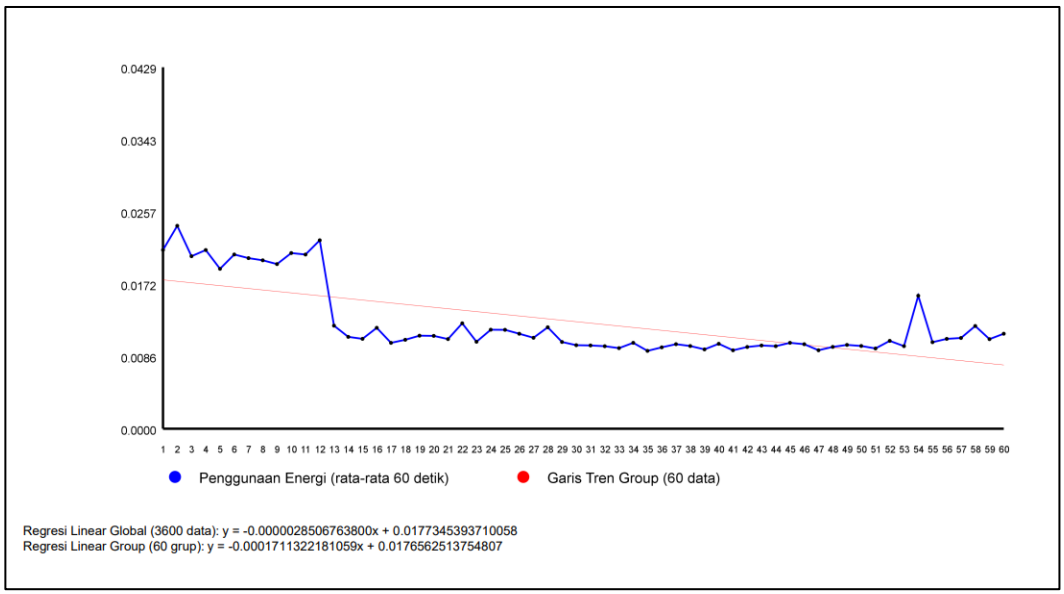
a. Server : 1089.1733
b. Excel : 1089.3156

Error yang dimiliki adalah 0.0131%

4.19 Perbandingan Prediksi

Perbandingan grafik pada bagian ini mencakup tiga konsumer, yaitu konsumer satu (kipas angin), konsumer dua (*charger* laptop), dan konsumer tiga (*charger handphone*). Hasil analisa grafik yang digunakan adalah hasil dari server. Berikut hasil yang dikeluarkan dari server

1. Kipas Angin (Konsumer Satu)



Gambar 4.20 grafik data konsumer satu

Gambar grafik 4.20 memiliki garis trend yang turun ke bawah. Untuk lebih detail terdapat pada gambar 4.21

Bagian 2: Analisis MAPE, Regresi Linear, dan Saran Energi

Poin-Poin Analisis:

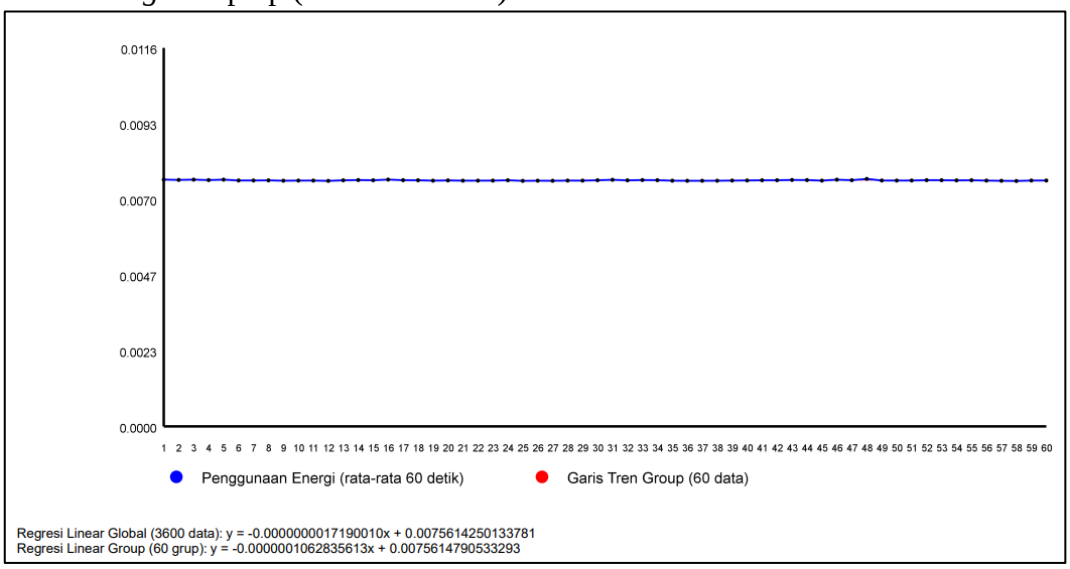
- Analisis dilakukan pada 3600 data terakhir
- MAPE (Mean Absolute Percentage Error): 24.03%
- Persamaan regresi linear global : Y = -0.0000028506763800X + 0.0177345393710058
- Persamaan regresi linear kelompok: Y = -0.0001711322181059x + 0.0176562513754807
- Tren penggunaan energi: Turun
- Rata-rata prediksi: 0.012606172563366369 74 Wh
- Saran energi untuk pembelian berikutnya (1 hari): 1089.17330947 Wh
- Nilai maksimum penggunaan energi: 0.0429 Wh
- Nilai minimum penggunaan energi: 0.0081 Wh
- Rata-rata penggunaan energi: 0.012606172563366364 53 Wh

Keterangan:

- Analisis dilakukan pada 3600 data terakhir untuk memastikan relevansi dan mengurangi beban komputasi
- Semua perhitungan statistik dan prediksi didasarkan pada 3600 data terakhir yang diproses
- Grouping (pengelompokan) adalah data rata-rata tiap 60 data asli yang dikelompokkan
- Data dari MAPE dan analisis merupakan data asli yang belum dikelompokkan
- Visualisasi adalah data yang telah dikelompokkan untuk mengurangi beban server
- Trendline pada visualisasi menggunakan persamaan yang sudah dikelompokkan

Gambar 4.21 Analisa pada bab 3 tentang konsumer satu

2. *Charger* Laptop (Konsumer Dua)



Gambar 4.22 grafik data konsumer dua

Gambar grafik 4.22 memiliki garis tren yang turun ke bawah. Untuk lebih detail terdapat pada gambar 4.23

Bagian 2: Analisis MAPE, Regresi Linear, dan Saran Energi

Poin-Poin Analisis:

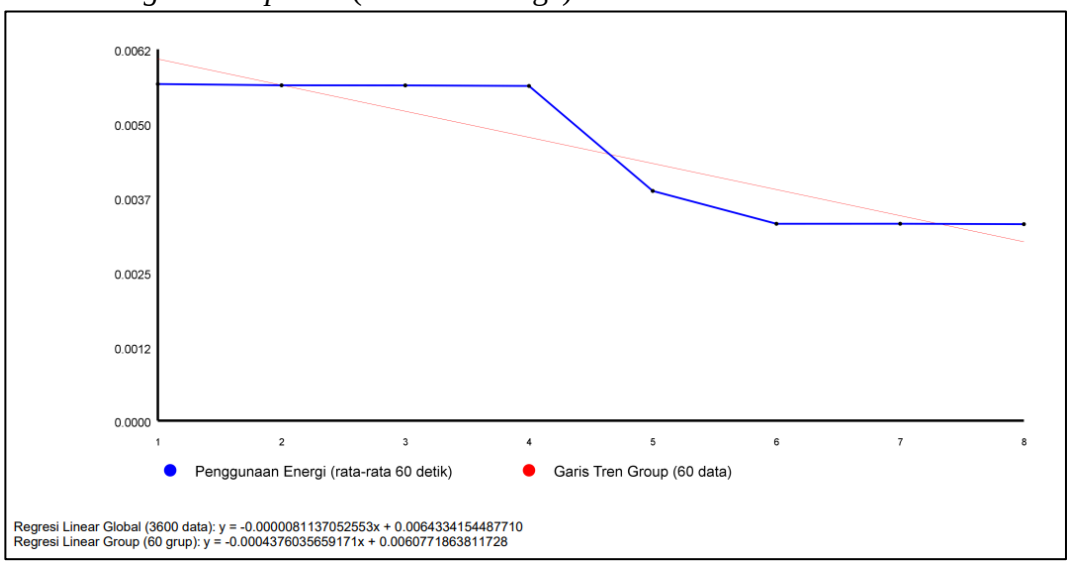
- Analisis dilakukan pada 3600 data terakhir
- MAPE (Mean Absolute Percentage Error): 0.54%
- Persamaan regresi linear global : Y = -0.0000000017190010X + 0.0075614250133781
- Persamaan regresi linear kelompok: Y = -0.0000001062835613x + 0.0075614790533293
- Tren penggunaan energi: Turun
- Rata-rata prediksi: 0.007558833253064120813 Wh
- Saran energi untuk pembelian berikutnya (1 hari): 653.03993065 Wh
- Nilai maksimum penggunaan energi: 0.0116 Wh
- Nilai minimum penggunaan energi: 0.0039 Wh
- Rata-rata penggunaan energi: 0.007558833253064120726 Wh

Keterangan:

- Analisis dilakukan pada 3600 data terakhir untuk memastikan relevansi dan mengurangi beban komputasi
- Semua perhitungan statistik dan prediksi didasarkan pada 3600 data terakhir yang diproses
- Grouping (pengelompokan) adalah data rata-rata tiap 60 data asli yang dikelompokkan
- Data dari MAPE dan analisis merupakan data asli yang belum dikelompokkan
- Visualisasi adalah data yang telah dikelompokkan untuk mengurangi beban server
- Trendline pada visualisasi menggunakan persamaan yang sudah dikelompokkan

Gambar 4.23 Analisa pada bab 3 tentang konsumer dua

3. *Charger Handphone* (Konsumer Tiga)



Gambar 4.24 grafik data konsumer tiga

Gambar grafik 4.24 memiliki garis trend yang turun ke bawah. Untuk lebih detail terdapat pada gambar 4.25

Bagian 2: Analisis MAPE, Regresi Linear, dan Saran Energi

Poin-Poin Analisis:

- Analisis dilakukan pada 3600 data terakhir
- MAPE (Mean Absolute Percentage Error): 10.77%
- Persamaan regresi linear global : Y = -0.0000081137052553X + 0.0064334154487710
- Persamaan regresi linear kelompok: Y = -0.0004376035659171x + 0.0060771863811728
- Tren penggunaan energi: Turun
- Rata-rata prediksi: 0.00471330993464051575 Wh
- Saran energi untuk pembelian berikutnya (1 hari): 407.22997835 Wh
- Nilai maksimum penggunaan energi: 0.0062 Wh
- Nilai minimum penggunaan energi: 0.0033 Wh
- Rata-rata penggunaan energi: 0.00471330993464051575 Wh

Keterangan:

- Analisis dilakukan pada 3600 data terakhir untuk memastikan relevansi dan mengurangi beban komputasi
- Semua perhitungan statistik dan prediksi didasarkan pada 3600 data terakhir yang diproses
- Grouping (pengelompokan) adalah data rata-rata tiap 60 data asli yang dikelompokkan
- Data dari MAPE dan analisis merupakan data asli yang belum dikelompokkan
- Visualisasi adalah data yang telah dikelompokkan untuk mengurangi beban server
- Trendline pada visualisasi menggunakan persamaan yang sudah dikelompokkan

Gambar 4.25 Analisa pada bab 3 tentang konsumer tiga

4.20 Perbandingan Hasil Prediksi atau Saran dan Persen Error dari Masing-Masing Konsumer

Tabel 4.9 Perbandingan saran dan persen error saran

Konsumer	MAPE (%)	Saran Energi (Wh)
1	24.03	1089.173309
2	0.54	653.0399307
3	10.77	407.2299784

Tabel 4.9 merupakan hasil perbandingan dari saran energi yang diberikan oleh keluaran server. Berikut hasil penjelasannya

1. Konsumer satu memiliki nilai persen error sebanyak 24.03% mengingat data dari konsumer satu, seperti pada gambar 4.20 yang cenderung tidak stabil
2. Konsumer dua memiliki nilai persen error sebanyak 0.54% mengingat data dari konsumer dua, seperti pada gambar 4.22 yang cenderung stabil
3. Konsumer dua memiliki nilai persen error sebanyak 10.77% mengingat data dari konsumer dua, seperti pada gambar 4.22 yang kurang stabil dan kurang lengkap untuk dikaji dikarenakan nilai energi yang digunakan sudah melebihi batas sehingga relay menutup terlebih dahulu ditunjukkan seperti pada gambar 4.23 dan listing 4.12 sehingga menyebabkan proses Analisa server sudah keluar, akan tetapi kurang optimal.

Hasil perbandingan pada Tabel 4.8 menunjukkan bahwa tingkat akurasi saran energi yang diberikan oleh keluaran server dipengaruhi oleh kestabilan dan kelengkapan data konsumer. Konsumer satu memiliki tingkat error yang cukup tinggi (24,03%) karena data yang tidak stabil. Konsumer dua menunjukkan tingkat error yang sangat rendah (0,54%) karena datanya cenderung stabil. Namun, pada kondisi tertentu konsumer dua juga memiliki tingkat error lebih tinggi (10,77%) akibat data yang kurang stabil dan tidak lengkap, terutama ketika energi yang digunakan melebihi batas sehingga relay menutup lebih awal, yang menyebabkan proses analisa server kurang optimal.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

1. Hasil penelitian menunjukkan bahwa teknologi blockchain pada platform Ethereum dapat diintegrasikan dengan sistem transaksi energi dari PLTS melalui mekanisme *smart contract*. Integrasi ini memungkinkan pencatatan transaksi dilakukan secara otomatis, transparan, dan terverifikasi oleh jaringan blockchain tanpa memerlukan pihak ketiga. Seluruh transaksi yang diuji berhasil diproses 100%, ditunjukkan oleh hash valid pada setiap Tx1 (*Buyer to Smart Contract*) dan Tx2 (*Smart Contract to Seller*). Hal ini membuktikan bahwa *platform* Ethereum dapat berfungsi sebagai media pencatatan dan eksekusi transaksi secara aman dan terpercaya.
2. Berdasarkan pengujian, analisis *delay* menunjukkan bahwa rata-rata waktu eksekusi total transaksi adalah sekitar ±22 detik. Selain itu, besar kecilnya nominal ETH yang ditransaksikan tidak berpengaruh signifikan terhadap waktu eksekusi sehingga mekanisme *smart contract* dapat dianggap stabil dan konsisten dalam memproses transaksi.
3. Selain itu, metode regresi linier yang diterapkan dalam penelitian ini mampu memberikan rekomendasi pembelian energi untuk periode selanjutnya berdasarkan data historis konsumsi energi. Hasil pengujian menunjukkan bahwa tingkat akurasi prediksi bergantung pada stabilitas data konsumsi, di mana konsumer dua atau beban *charger* laptop dengan data stabil memiliki galat sangat rendah (0,54%), sementara konsumer satu atau beban kipas angin memiliki data fluktuatif dengan menghasilkan galat lebih tinggi (24,03%). Secara galat parameter regresi linier, parameter *slope* dan *intercept* memiliki galat yang kecil, yaitu *slope* 0,19% dan *intercept* 0,39% sehingga dapat disimpulkan bahwa metode ini memiliki galat *slope* dan *intercept* yang kecil dan

galat prediksi bergantung pada beban yang terpasang pada tiap konsumer.

5.2 Saran

Untuk pengembangan lebih lanjut, sistem ini disarankan diuji pada skala pengguna yang lebih luas dan lingkungan nyata dengan fluktuasi konsumsi energi yang kompleks serta mempertimbangkan penggunaan metode prediksi yang lebih canggih dan jaringan *blockchain* lain untuk meningkatkan efisiensi, mengurangi biaya transaksi, dan kecepatan transaksi dan tidak lupa juga diperlukan peningkatan sistem keamanan *off-chain* yang lebih ketat mengingat kebanyakan endpoint pada program server masih belum dikuatkan sehingga rentan terhadap serangan siber.

DAFTAR PUSTAKA

Abdoun, N., El Assad, S., Manh Hoang, T., Déforges, O., Assaf, R., Khalil, M., & Deforges, O. (2020). Designing Two Secure Keyed Hash Functions Based on Sponge Construction and the Chaotic Neural Network. *Entropy*, *2020*(9). https://doi.org/10.3390/e22091012ï

Akbar Bathnul Wadi Nst, M., Abdillah Syam, M., Zacky Luthfi, R., Aditya Abdi, F., & Tambak, S. (2025). Analisis Peran dan Tantangan Penggunaan Blockchain untuk Meningkatkan Pertumbuhan Penjualan dan Kepercayaan Konsumen di E-Commerce. *JSI (Jurnal Sistem Informasi) Universitas Dirgantara Marsekal Suryadarma*.

Antonino, P., & Roscoe, A. W. (2020). *Formalising and verifying smart contracts with Solidifier: a bounded model checker for Solidity*. http://arxiv.org/abs/2002.02710

BUCERZAN, D., & BEJAN, C. A. (2020). WALLET KEY MANAGEMENT IN BLOCKCHAIN TECHNOLOGY. *Proceedings of the 19th International Conference on INFORMATICS in ECONOMY Education, Research and Business Technologies*, *2020*, 83–88. https://doi.org/10.24818/ie2020.02.05

Buterin, V. (2015). *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.* https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf

di Angelo, M., & Salzer, G. (2020). *Wallet Contracts on Ethereum -- Identification, Types, Usage, and Profiles*. http://arxiv.org/abs/2001.06909

Dr. Poornima G. Naik, & Dr. Girish R. Naik. (2023). *Blockchain Beyond Basics*. www.shashwatpublication.com

Dzulfikar, F., & Susanto, A. (2020). Implementation of Smart Contracts Ethereum Blockchain in Web-Based Electronic Voting (e-voting). *TRANSFORMTIKA*, *18*(1), 56–62.

Ethereum Foundation. (2024). *Sepolia & Holesky: Dencun Update*.

Eyada, M. M., Saber, W., El Genidy, M. M., & Amer, F. (2020). Performance Evaluation of IoT Data Management Using MongoDB Versus MySQL Databases in Different Cloud Environments. *IEEE Access*, *8*, 110656–110668. https://doi.org/10.1109/ACCESS.2020.3002164

Gupta, A., Gupta, J., Bhardwaj, P., Chandra, Y., & Sagar, D. (2023). EXPLORING THE ETHEREUM BLOCKCHAIN: ANALYTICS. *International Journal of Research and Analytical Reviews*. www.ijrar.org

Ida Nabillah, & Indra Ranggadara. (2020). Rancang Bangun Panel Automatic Load Shedding pada Genset Satu Fasa Arfinaldi 1 , Aslimeri 2 Universitas Negeri Padang. *Journal of Information System*, *5*, 250–255. https://doi.org/DOI:10.33633/joins.v5i2.3900

Kapengut, E., & Mizrach, B. (2022). *An Event Study of the Ethereum Transition to Proof-of-Stake*. http://arxiv.org/abs/2210.13655

Lukman Hakim, D., & Utari, L. (2020). *Prediksi Jumlah Pembelian Sepatu Dengan Penerapan Metode Regresi Linear*. *10*, 71–80. https://doi.org/10.36350/jbs.v10i2

Mehdi Montakhabi, Esteve Almirall, Akash Madhusudan, Mustafa A Mustafa, Wim Vanhaverbeke, & Shenja van der Graaf. (2024). Corrigendum to: Leveraging blockchain for energy transition in urban contexts. *Big Data and Society*, *11*(1). https://doi.org/10.1177/20539517231223863

Nanda Sari, A., & Gelar, T. (2024). *BLOCKCHAIN: TEKNOLOGI DAN IMPLEMENTASINYA* (Vol. 7, Issue 1).

Pratama, B. E., Pauzi, G. A., Suciyati, W., & Supriyanto, A. (2023). Heating Rate Control in Electric Furnace with SSR and K-Type Thermocouple Using Arduino Uno Atmega 328P. *Journal of Energy, Material, and Instrumentation Technology*, *4*(1). https://jemit.fmipa.unila.ac.id/

Putu Ardi Wahyu Widyatmika, Ni Putu Ayu Widyanata Indrawati, Wayan Wahyu Adi Prastya, Ketut Darminta, Gde Nyoman Sangka, & Anak Agung Ngurah Gde Sapteka. (2021). Perbandingan Kinerja Arduino Uno dan ESP32 Terhadap Pengukuran Arus dan Tegangan. *Jurnal Otomasi, Kontrol & Instrumentasi* , *13*.

Selsie Anggela Putri, L. K. A. N. M. (2020). *Analisis Hambatan Pemanfaatan Plts Di Provinsi Jawa Tengah Dalam Upaya Meningkatkan Peran Energi Baru Dan Energi Tebarukan Pada Tahun 2020-2022*.

Shubham, P. R., Wali, C., & Samarth, A. G. C. (2024). *Crowd Funding Marketplace On Ethereum Using NextJs & Solidity* (Vol. 12, Issue 6). www.ijcrt.org

Wibowo, H., Bandri, S., & Anugrah, A. (2024). Studi Perencanaan Pembangkit Listrik Tenaga Surya Atap On-Grid Skala Rumah Tinggal Daya 1300 VA Menggunakan Software Homer. *Anggun Anugrah INNOVATIVE: Journal Of Social Science Research*, *4*, 9524–9544.

Winter, P., Lorimer, A. H., Snyder, P., & Livshits, B. (2021). *Security, Privacy, and Decentralization in Web3*. http://arxiv.org/abs/2109.06836

“Halaman ini sengaja dikosongkan”

LAMPIRAN

LAMPIRAN A SMART CONTRACT

Lampiran A1 - Program Smart Contract

```
// contracts/MessageStore.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimplePayment {
    address public seller; // Alamat penjual

    // Constructor untuk menentukan alamat penjual
    constructor(address _seller) {
        seller = _seller;
    }

    // Fungsi untuk menerima ETH
    receive() external payable {}

    // Fungsi untuk mengirim ETH ke penjual
    function transferToSeller() public {
        require(address(this).balance > 0, "No funds available");

        // Mengirimkan semua ETH yang ada di contract ke penjual
        (bool success, ) = seller.call{value:
address(this).balance}("");
        require(success, "Transfer to seller failed.");
    }

    // Fungsi untuk mendapatkan saldo contract
    function getBalance() public view returns (uint256) {
        return address(this).balance;
    }
}
```

Lampiran A2 - Config Deploy Smart Contract

```
require("@nomiclabs/hardhat-waffle");
require("@nomiclabs/hardhat-ethers");
require("@nomiclabs/hardhat-etherscan");
require('dotenv').config({ path: __dirname + '/zzz.env' })

const mnemonic = process.env.MNEMONIC; // Ensure MNEMONIC is defined
in your zzz.env file
const alchemyKey = process.env.ALCHEMY_KEY; // Ensure ALCHEMY_KEY is
defined in your zzz.env file

/** @type import('hardhat/config').HardhatUserConfig */
module.exports = {
  solidity: "0.8.28",
  networks: {
    sepolia: {
      url: `https://eth-sepolia.g.alchemy.com/v2/${alchemyKey}`,
      accounts: { mnemonic: mnemonic },
    },
  },
  etherscan: {
          apiKey:    process.env.ETHERSCAN_API_KEY,    //    Ensure
ETHERSCAN_API_KEY is defined in your zzz.env file
  },
};
```

Lampiran A3 - Program Deploy Smart Contract

```
async function main() {
  const [deployer] = await ethers.getSigners();

  // Ganti dengan alamat wallet yang ingin Anda gunakan sebagai
alamat penjual
  const                        sellerAddress                        =
"0x673ddD0D4a28309D1fD9321A8Fde5A0bac78179E";

  const              SimplePayment              =              await
ethers.getContractFactory("SimplePayment");
  const simplePayment = await SimplePayment.deploy(sellerAddress);
  await simplePayment.deployed();
  console.log("SimplePayment deployed to:", simplePayment.address);
}

main()
.then(() => process.exit(0))
.catch((error) => {
    console.error(error);
    process.exit(1);
});
```

LAMPIRAN B SERVER

Lampiran B1 – Program pewaktuan server menggunakan C

```
#include <stdio.h>
#include <windows.h>

void get_local_time_iso8601(char *buffer, size_t buffer_size) {
    SYSTEMTIME st;

    // Mendapatkan waktu lokal
    GetLocalTime(&st);

    // Format waktu menjadi ISO 8601
                snprintf(buffer,       buffer_size,       "%04d-%02d-
%02dT%02d:%02d:%02d.%03dZ",
             st.wYear,
             st.wMonth,
             st.wDay,
             st.wHour,
             st.wMinute,
             st.wSecond,
             st.wMilliseconds);
}

int main() {
    char time_buffer[30];

    // Memanggil fungsi untuk mendapatkan waktu lokal dalam format
ISO 8601
    get_local_time_iso8601(time_buffer, sizeof(time_buffer));

    // Menampilkan hasil
    printf("%s\n", time_buffer);

    return 0;
}
```

Lampiran B2 – Program server

```
const express = require('express');
const path = require('path');
const XLSX  = require('xlsx');
const { ethers } = require('ethers');
const { MongoClient } = require('mongodb');
const PDFDocument = require('pdfkit');
const { v4: uuidv4 } = require('uuid'); // Import uuidv4
const { execSync } = require('child_process');
const { createCanvas } = require('canvas');
const fs = require('fs');
require('dotenv').config({ path: __dirname + '/zzz.env' });
const bodyParser = require('body-parser');
const axios = require("axios");

const appSeller = express();
const app = express();
app.use(express.json());
app.use(express.static(path.join(__dirname, 'public')));

const cors = require('cors');
app.use(cors());
const port = 3000;
const SELLER_PORT = 4000;
appSeller.use(express.json());
appSeller.use(bodyParser.urlencoded({ extended: true })); // Untuk
menangani form HTML
appSeller.use(express.static(path.join(__dirname, 'private')));

const SELLER = process.env.SELLER;
const SELLER_PASS = process.env.SELLER_PASS;
const APIalchemy = process.env.ALCHEMY_KEY;

////////////////// Fungsi Pewaktuan Server ////////////////////
function getCurrentTime() {
    try {
        const output = execSync('get_time.exe').toString().trim();
        return output;
    } catch (error) {
        console.error('Error executing get_time.c:', error);
       throw new Error('Failed to get current time from C program');
    }
}
////////////////// Fungsi Pewaktuan Server END /////////////////

// URI dan Nama Database
const uri = 'mongodb://localhost:27017';
const dbName = 'transactionDB';

const provider = new ethers.providers.JsonRpcProvider("https://eth-
sepolia.g.alchemy.com/v2/3FwxeLzcyAJOBL2Gx77kpgj9EVLGz4h_"); // API
Key Alchemy untuk Sepolia
```

```
const                        contractAddress                        =
"0xB9067eA7df746A653Ac86D31b8361FE34048e655";   //   Alamat   smart
contract

const contractABI = [
    "function transferToSeller() public",
    "function getBalance() public view returns (uint256)"
];

/////////////////Koneksi mongo DB - Start//////////////////
const client = new MongoClient(uri);
let db;
let requestsCollection;
let transactionsCollection;
let consumerRelayCollection;
let energyRight;
let energylogSensor

async function connectDB() {
    if (!client.topology || !client.topology.isConnected()) {
        await client.connect();
        const db = client.db(dbName);
        requestsCollection = db.collection('requests');
        transactionsCollection = db.collection('transactions');
                                    consumeEnergyCollection        =
db.collection('consumer_energy_history');
        consumerRelayCollection = db.collection('consumerRelay');
        energyRight = db.collection('energy_right');
        energylogSensor = db.collection('consumed_energy_sensor')
        console.log('Connected to database');
    }
}
/////////////////Koneksi mongo DB - End//////////////////

// Redirect ke halaman login jika diakses di port 4000

app.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, 'public', 'index.html'));
});

// Output waktu sebelum server berjalan
console.log("Server time at startup:", getCurrentTime());

app.post('/sendTransaction', async (req, res) => {
    const buyerPrivateKey = req.body.privateKey;
    const buyerAddress = req.body.address;
    const energyAmount = req.body.energyAmount;

    // Log request
    console.log("Received request:", {
        address: buyerAddress,
        energyAmount: energyAmount
    });
```

```
    // Validasi input
     if (!buyerPrivateKey || !buyerAddress || !energyAmount ||
isNaN(parseFloat(energyAmount))) {
          return res.status(400).json({ status: 'Error', message:
'Missing or invalid required parameters' });
    }

    try {
        // Validasi wallet
        let buyerWallet;
        const startRequestTime = Date.now(); // waktu awal request

        try {
            buyerWallet = new ethers.Wallet(buyerPrivateKey);
        } catch (error) {
            return res.status(400).json({ status: 'Error', message:
'Wrong on wallet address or private key' });
        }

        // Validasi apakah address dari wallet cocok dengan address
yang diberikan
                    if    (buyerWallet.address.toLowerCase()    !==
buyerAddress.toLowerCase()) {
            return res.status(400).json({ status: 'Error', message:
'Wrong on wallet address or private key' });
        }

        // Validasi saldo
                           const     balanceBefore     =     await
provider.getBalance(buyerWallet.address);
                     console.log(`Balance    before    transaction:
${ethers.utils.formatEther(balanceBefore)} ETH`);

        // Kalkulasi jumlah ETH berdasarkan energi dalam kWh
       const energyInKWh = parseFloat(energyAmount); // Energi dalam
kWh
        const ethRatePerKWh = 0.000000025; // Tarif ETH per Wh
         const rawAmount = energyInKWh * ethRatePerKWh; // Hitung
nilai mentah

        // Bulatkan nilai ke 18 desimal sebelum digunakan
        const roundedAmount = rawAmount.toFixed(18);
           const  amount  =  ethers.utils.parseUnits(roundedAmount,
"ether");

        console.log(`Energy in kWh: ${energyInKWh}`);
        console.log(`Amount to send (ETH): ${roundedAmount}`);

        // Perkirakan biaya gas untuk transaksi pertama
        const gasEstimate = await provider.estimateGas({
            to: contractAddress,
            value: amount
        });
        const gasPrice = await provider.getGasPrice();
```

```
        const totalGasCost = gasEstimate.mul(gasPrice);
           console.log(`Estimated gas cost (buyer transaction):
${ethers.utils.formatEther(totalGasCost)} ETH`);

        // Periksa apakah saldo mencukupi
        const totalCost = amount.add(totalGasCost);
        if (balanceBefore.lt(totalCost)) {
            return res.status(400).json({
                status: 'Error',
              message: 'Insufficient funds for intrinsic transaction
cost'
            });
        }

        await connectDB();

        // Simpan data request
        const requestData = {
            address: buyerAddress,
            energyAmount: energyAmount,
            hash_txid: null,
            timestamp: getCurrentTime()
        };
                            const     requestResult     =     await
requestsCollection.insertOne(requestData);
               console.log('Request   data   inserted   with   id:',
requestResult.insertedId);

        // Kirim ETH ke smart contract
        const tx1Start = Date.now();
                                  const       tx       =       await
buyerWallet.connect(provider).sendTransaction({
            to: contractAddress,
            value: amount
        });

        console.log("Transaction sent:", tx);
        const receipt = await tx.wait();
        const tx1End = Date.now();
        const tx1Duration = (tx1End - tx1Start); // ms
        console.log(`Tx1 duration: ${tx1Duration} ms`);

        // Perbarui request dengan hash transaksi pertama
        await requestsCollection.updateOne(
            { _id: requestResult.insertedId },
            { $set: { hash_txid: tx.hash } }
        );

        // Memanggil fungsi transferToSeller
           const  contract  =  new  ethers.Contract(contractAddress,
contractABI, buyerWallet.connect(provider));

        // Perkirakan gas untuk transaksi kedua
```

```
                            const      gasEstimate2      =      await
contract.estimateGas.transferToSeller();
        const gasPrice2 = await provider.getGasPrice();
                                  const       sellerGasCost       =
ethers.utils.formatEther(gasEstimate2.mul(gasPrice2));
               console.log(`Gas   cost   for   seller   transaction:
${sellerGasCost} ETH`);

        const tx2Start = Date.now();
        const tx2 = await contract.transferToSeller();
        console.log("Transfer to seller transaction:", tx2);
        const receipt2 = await tx2.wait();
        const tx2End = Date.now();
        const tx2Duration = (tx2End - tx2Start); // ms
        console.log(`Tx2 duration: ${tx2Duration} ms`);

        const endRequestTime = Date.now();
        const totalDuration = (endRequestTime - startRequestTime);
        console.log(`Total duration: ${totalDuration} ms`);

                            const      balanceAfter      =      await
provider.getBalance(buyerWallet.address);
                     console.log(`Balance    after    transaction:
${ethers.utils.formatEther(balanceAfter)} ETH`);

        // Simpan data transaksi
        const transactionData = {
            requestId: requestResult.insertedId,
            balanceBefore: ethers.utils.formatEther(balanceBefore),
            amountToSend: ethers.utils.formatEther(amount),
            transactionSent: {
                type: tx.type,
                chainId: tx.chainId,
                nonce: tx.nonce,
                   maxPriorityFeePerGas: tx.maxPriorityFeePerGas ?
tx.maxPriorityFeePerGas.toString() : null,
                                  maxFeePerGas:   tx.maxFeePerGas   ?
tx.maxFeePerGas.toString() : null,
                  gasPrice: tx.gasPrice ? tx.gasPrice.toString() :
null,
                gasLimit: tx.gasLimit.toString(),
                to: tx.to,
                from: tx.from,
                value: tx.value.toString(),
                accessList: tx.accessList,
                hash: tx.hash,
                confirmations: receipt.confirmations,
                gasCost: ethers.utils.formatEther(totalGasCost)
            },
            transferToSellerTransaction: {
                type: tx2.type,
                chainId: tx2.chainId,
                nonce: tx2.nonce,
```

```
                    maxPriorityFeePerGas: tx2.maxPriorityFeePerGas ?
tx2.maxPriorityFeePerGas.toString() : null,
                                maxFeePerGas:   tx2.maxFeePerGas   ?
tx2.maxFeePerGas.toString() : null,
                 gasPrice: tx2.gasPrice ? tx2.gasPrice.toString() :
null,
                gasLimit: tx2.gasLimit.toString(),
                to: tx2.to,
                from: tx2.from,
                value: tx2.value.toString(),
                accessList: tx2.accessList,
                hash: tx2.hash,
                confirmations: receipt2.confirmations,
                gasCost: sellerGasCost
            },
            balanceAfter: ethers.utils.formatEther(balanceAfter),
            timestamp: getCurrentTime(),
            durations: {
                tx1: tx1Duration,
                tx2: tx2Duration,
                total: totalDuration
            }
        };

                        const     transactionResult     =     await
transactionsCollection.insertOne(transactionData);
             console.log('Transaction  data  inserted  with  id:',
transactionResult.insertedId);

        // Panggil fungsi untuk memperbarui consume_energy
        await updateConsumeEnergy(buyerAddress, energyInKWh);

        // Respons ke buyer
        res.json({
            status: 'Transaction Successful',
            txHash: tx.hash,
            txHash2: tx2.hash,
            balanceBefore: ethers.utils.formatEther(balanceBefore),
            balanceAfter: ethers.utils.formatEther(balanceAfter),
            buyerGasCost: ethers.utils.formatEther(totalGasCost),
            durations: {
                tx1: tx1Duration,
                tx2: tx2Duration,
                total: totalDuration
            }
        });

    } catch (error) {
        console.error("Error during transaction:", error);
        res.status(500).json({ status: 'Error', error: error.message
});
    }
});
```

```
async function updateConsumeEnergy() {
    try {
        console.log("Starting consumer_energy_history update...");
        await connectDB();

        // Ambil semua data request yang memiliki hash_txid
        const requests = await requestsCollection.find({ hash_txid:
{ $ne: null } }).toArray();
        console.log(`Fetched ${requests.length} requests with valid
hash_txid.`);

         // Map untuk menyimpan total energi dan riwayat transaksi
per wallet
        const walletDataMap = {};

        // Iterasi melalui semua request
        for (const request of requests) {
            const wallet = request.address;
            const energyAmount = parseFloat(request.energyAmount);

            // Ambil transaksi terkait dari koleksi transactions
                                const    transaction    =    await
transactionsCollection.findOne({            "transactionSent.hash":
request.hash_txid });
                           const   ethAmount   =   transaction   ?
parseFloat(ethers.utils.formatEther(transaction.transactionSent.val
ue)) : 0;

            const transactionEntry = {
                hash_tx: request.hash_txid,
                timestamp: request.timestamp,
                energyAmount,
                ethAmount, // Tambahkan jumlah ETH
            };

            if (!walletDataMap[wallet]) {
                walletDataMap[wallet] = {
                    totalEnergy: 0,
                    transactionHistory: [],
                };
            }

            // Tambahkan energi ke total dan riwayat transaksi
            walletDataMap[wallet].totalEnergy += energyAmount;
            walletDataMap[wallet].transactionHistory.push(transacti
onEntry);
        }

        // Perbarui koleksi consume_energy
        for (const [wallet, data] of Object.entries(walletDataMap))
{
                                const    existingEntry    =    await
consumeEnergyCollection.findOne({ wallet });
```

```
            if (existingEntry) {
                // Filter transaksi baru yang belum ada di riwayat
                                        const    newTransactions    =
data.transactionHistory.filter(
                                                                (tx)     =>
!existingEntry.transactionHistory.some((existingTx)              =>
existingTx.hash_tx === tx.hash_tx)
                );

               // Perbarui nilai totalEnergy dan tambahkan transaksi
baru ke riwayat
                                         const    updatedHistory    =
[...existingEntry.transactionHistory, ...newTransactions];

                 const newTotalEnergy = existingEntry.totalEnergy +
newTransactions.reduce((sum, tx) => sum + tx.energyAmount, 0);

                await consumeEnergyCollection.updateOne(
                    { wallet },
                    {
                        $set: {
                            totalEnergy: newTotalEnergy,
                            transactionHistory: updatedHistory,
                            lastUpdated: getCurrentTime(),
                        },
                    }
                );
                  console.log(`Updated consumer_energy_history for
wallet ${wallet}: ${newTotalEnergy} kWh`);
            } else {
                // Jika wallet belum ada, tambahkan entri baru
                await consumeEnergyCollection.insertOne({
                    wallet,
                    totalEnergy: data.totalEnergy,
                    transactionHistory: data.transactionHistory,
                    firstPurchase: getCurrentTime(),
                    lastUpdated: getCurrentTime(),
                });
                 console.log(`Added new entry for wallet ${wallet}:
${data.totalEnergy} kWh`);
            }
        }

        console.log("consumer_energy_history update completed.");
    } catch (error) {
          console.error("Error  updating  consumer_energy_history:",
error.message);
    }
}

app.post('/updateConsumeEnergy', async (req, res) => {
    try {
        await updateConsumeEnergy();
```

```
                    res.json({    status:    "Success",    message:
"consumer_energy_history collection updated successfully." });
    } catch (error) {
              res.status(500).json({   status:   "Error",   message:
error.message });
    }
});

// Fungsi untuk memvalidasi request
async function validateRequest(request) {
    try {
        console.log("Validating request...");

        await connectDB();

        // Ambil transaksi terkait berdasarkan hash_txid
        const transaction = await transactionsCollection.findOne({
            "transactionSent.hash": request.hash_txid
        });

        if (!transaction) {
            return { isValid: false, reason: "Transaction not found
for the given hash_txid." };
        }

        if (transaction.transactionSent.from !== request.address) {
             return { isValid: false, reason: "Wallet address does
not match the transaction." };
        }

                                const      requestedValueInWei      =
ethers.utils.parseUnits((request.energyAmount                       *
0.000000025).toString(), "ether");
                    if    (transaction.transactionSent.value    !==
requestedValueInWei.toString()) {
            return {
                isValid: false,
                   reason: `Transaction value mismatch. Requested:
${ethers.utils.formatEther(requestedValueInWei)}    ETH,    Found:
${ethers.utils.formatEther(transaction.transactionSent.value)}
ETH.`
            };
        }

        return { isValid: true, reason: "Request is valid." };
    } catch (error) {
        console.error("Error during validation:", error.message);
          return { isValid: false, reason: "Internal error during
validation." };
    }
}

// Endpoint untuk memvalidasi request
app.post('/validateRequest', async (req, res) => {
```

```
    const request = req.body; // Mengambil data request dari body

          if    (!request.address    ||    !request.hash_txid    ||
!request.energyAmount) {
        return res.status(400).json({
            status: "Error",
            message: "Missing required parameters in the request."
        });
    }

    const validationResult = await validateRequest(request);

    if (validationResult.isValid) {
        res.json({
            status: "Success",
            message: validationResult.reason
        });
    } else {
        res.status(400).json({
            status: "Error",
            message: validationResult.reason
        });
    }
});

app.post('/assignConsumer', async (req, res) => {
    const { consumer, wallet, relay, pin } = req.body;

    // Validasi input
    if (!consumer || !wallet || !relay || !pin) {
        return res.status(400).json({
            status: "Error",
              message: "Missing required fields: consumer, wallet,
relay, or pin."
        });
    }

    // Validasi tipe data relay dan pin
    if (isNaN(relay) || isNaN(pin)) {
        return res.status(400).json({
            status: "Error",
            message: "Relay and pin must be numeric."
        });
    }

    try {
        await connectDB();

          // Periksa apakah consumer atau wallet sudah ada dalam
database
                         const     existingConsumer     =     await
consumerRelayCollection.findOne({ consumer });
                          const     existingWallet     =     await
consumerRelayCollection.findOne({ wallet });
```

```
        if (existingConsumer) {
            return res.status(400).json({
                status: "Error",
              message: `Consumer "${consumer}" is already assigned.`
            });
        }

        if (existingWallet) {
            return res.status(400).json({
                status: "Error",
                message: `Wallet "${wallet}" is already assigned.`
            });
        }

        // Masukkan data ke koleksi
        const newMapping = {
            consumer,
            wallet,
            relay: Number(relay), // Konversi relay ke angka
            pin: Number(pin),     // Konversi pin ke angka
        };

                                const      result      =      await
consumerRelayCollection.insertOne(newMapping);

        res.json({
            status: "Success",
            message: "Mapping assigned successfully.",
            data: newMapping
        });

       console.log(`Mapping added: ${JSON.stringify(newMapping)}`);
    } catch (error) {
        console.error("Error assigning consumer:", error.message);
        res.status(500).json({
            status: "Error",
            message: "Internal server error."
        });
    }
});

// Pilihan 1: Cek saldo
app.get('/checkBalance', async (req, res) => {
    const { address } = req.query;
    if (!address || !ethers.utils.isAddress(address)) {
          return res.status(400).json({ status: "Error", message:
"Invalid wallet address." });
    }

    try {
        const balance = await provider.getBalance(address);
        const formattedBalance = ethers.utils.formatEther(balance);
        res.json({
```

```
            status: "Success",
            address: address,
            balance: formattedBalance
        });
    } catch (error) {
          res.status(500).json({ status: "Error", message: "Error
retrieving wallet balance." });
    }
});

app.post('/getPurchaseHistory', async (req, res) => {
    const { wallet, format } = req.body;

    if (!wallet || !ethers.utils.isAddress(wallet)) {
        return res.status(400).json({
            status: 'Error',
            message: 'Invalid or missing wallet address.',
        });
    }

    await connectDB();
     const consumerData = await consumeEnergyCollection.findOne({
wallet });

    if (!consumerData) {
        return res.status(404).json({
            status: 'Error',
                      message: 'Wallet address not found in
consumer_energy_history collection.',
        });
    }

    if (format === 'pdf') {
        const doc = new PDFDocument({ margin: 30 });
        const uniqueId = uuidv4(); // Generate UUID
                                        const fileName =
`${wallet}_${uniqueId}_purchase_history.pdf`; // Generate unique
file name

        res.setHeader('Content-Type', 'application/pdf');
                res.setHeader('Content-Disposition', `attachment;
filename=${fileName}`);
        doc.pipe(res);

        // Header
        doc.fontSize(18).text('Purchase History', { align: 'center'
});
        doc.moveDown();
                          doc.fontSize(12).text(`Wallet Address:
${consumerData.wallet}`);
                           doc.text(`Total Energy Purchased:
${consumerData.totalEnergy.toFixed(4)} kWh`);
        doc.moveDown();
```

```
        // Tabel Header
         doc.fontSize(14).text('Transaction History:', { underline:
true });
        doc.moveDown();

        // Header tabel
        const columnWidths = [30, 150, 80, 80, 200];
        const tableTop = doc.y;

         const headers = ['No.', 'Date', 'Energy (kWh)', 'ETH', 'TX
Hash'];
        headers.forEach((header, i) => {
           doc.fontSize(10).text(header, 50 + columnWidths.slice(0,
i).reduce((a, b) => a + b, 0), tableTop + 5, {
                width: columnWidths[i],
                align: i === 4 ? 'left' : 'center',
            });
        });

        // Garis setelah header
        doc.moveTo(50, tableTop + 20)
            .lineTo(550, tableTop + 20)
            .stroke();

        // Isi tabel
       consumerData.transactionHistory.forEach((transaction, index)
=> {
            const rowY = tableTop + 30 + index * 30; // Tambahkan
jarak 10px

            const row = [
                index + 1,
                transaction.timestamp,
                transaction.energyAmount.toFixed(4),
                transaction.ethAmount.toFixed(8),
                transaction.hash_tx,
            ];

            row.forEach((cell, i) => {
                                doc.fontSize(10).text(cell,   50   +
columnWidths.slice(0, i).reduce((a, b) => a + b, 0), rowY, {
                    width: columnWidths[i],
                    align: i === 4 ? 'left' : 'center',
                });
            });
        });

        doc.end();
    } else {
        res.status(400).json({
            status: 'Error',
            message: 'Unsupported format.',
        });
    }
```

```
});

app.post('/logEnergyConsumption', async (req, res) => {
    const {
        wallet,
        sensorId,
        energyUsed,
        power,
        voltage,
        current,
        frequency,
        powerFactor,
        alarm,
        elapsedTime
    } = req.body;

    if (!wallet || !sensorId) {
        return res.status(400).json({
            status: 'Error',
            message: 'Missing required fields: wallet, sensorId',
        });
    }

    if (!ethers.utils.isAddress(wallet)) {
        return res.status(400).json({
            status: 'Error',
            message: 'Invalid wallet address',
        });
    }

    try {
        await connectDB();

        const timestamp = getCurrentTime();
       const existingEntry = await energylogSensor.findOne({ wallet,
sensorId });
        let deltaTime = 0;

        let energyUsedValue = 0;
        if (power !== undefined) {
            const powerFloat = parseFloat(power);
                                const    previousTimestampStr    =
existingEntry?.usageHistory?.slice(-1)[0]?.timestamp;

            if (previousTimestampStr) {
                                const   previousTimestamp   =   new
Date(previousTimestampStr);
               deltaTime = (new Date(timestamp) - previousTimestamp)
/ 1000;

                // Validasi deltaTime
                if (deltaTime > 0 && deltaTime < 300) {
                    energyUsedValue = (powerFloat * 1) / 3600;
                } else {
```

```
                        console.warn("Delta waktu tidak valid:",
deltaTime);
                    deltaTime = 0;
                }
            }
        }

        // Validasi energyUsed
        if (isNaN(energyUsedValue) || energyUsedValue < 0) {
            energyUsedValue = 0;
        }

             const { RemainingEnergy, relayStatus } = await
syncEnergyRight(wallet);
        if (relayStatus === "closed") {
            return res.status(400).json({
                status: 'Error',
                  message: 'Relay is already closed. No remaining
energy available.',
            });
        }

         const newRemainingEnergy = Math.max(0, RemainingEnergy -
energyUsedValue);
         const newRelayStatus = newRemainingEnergy > 0 ? "open" :
"closed";

        const usageData = {
           index: existingEntry ? existingEntry.usageHistory.length
: 0,
            timestamp,
            deltaTime, // 🆕 Ditambahkan di sini
            energyUsed: energyUsedValue,
            voltage: parseFloat(voltage),
            current: parseFloat(current),
            power: parseFloat(power),
            frequency: parseFloat(frequency),
            powerFactor: parseFloat(powerFactor),
            alarm: !!alarm,
        };

        if (existingEntry) {
                                const    updatedUsageHistory    =
[...existingEntry.usageHistory, usageData];
                               const    updatedTotalEnergyUsed    =
updatedUsageHistory.reduce(
                (sum, entry) => sum + entry.energyUsed, 0
            );

            await energylogSensor.updateOne(
                { wallet, sensorId },
                {
                    $set: {
                        TotalEnergyUsed: updatedTotalEnergyUsed,
```

```
                        last_update: timestamp,
                        usageHistory: updatedUsageHistory,
                    },
                }
            );
        } else {
            const newEntry = {
                wallet,
                sensorId,
                TotalEnergyUsed: energyUsedValue,
                first_timestamp: timestamp,
                last_update: timestamp,
                usageHistory: [usageData],
            };

            await energylogSensor.insertOne(newEntry);
        }

        await energyRight.updateOne(
            { wallet },
            {
                $set: {
                    RemainingEnergy: newRemainingEnergy,
                    relayStatus: newRelayStatus,
                    lastUpdated: timestamp,
                },
            }
        );

        res.json({
            status: 'Success',
            message: 'Energy consumption logged successfully',
            data: {
                wallet,
                sensorId,
                energyUsed: energyUsedValue,
                RemainingEnergy: newRemainingEnergy,
                relayStatus: newRelayStatus,
                timestamp,
                sensorData: usageData,
            },
        });

    } catch (error) {
             console.error('Error  logging  energy  consumption:',
error.message);
        res.status(500).json({
            status: 'Error',
            message: 'Internal server error',
        });
    }
});

async function syncEnergyRight(wallet) {
```

```
    try {
        // Validasi awal untuk wallet
        if (!wallet || typeof wallet !== "string") {
            throw new Error("Invalid or missing wallet address");
        }

        // Ambil data dari consumer_energy_history
                        const     consumerHistory     =     await
consumeEnergyCollection.findOne({ wallet });
        if (!consumerHistory) {
                throw new Error(`Wallet ${wallet} not found in
consumer_energy_history`);
        }

        // Ambil total konsumsi energi dari consumed_energy_sensor
        const sensorConsumption = await energylogSensor.aggregate([
            { $match: { wallet } },
                { $group: { _id: null, totalConsumed: { $sum:
"$TotalEnergyUsed" } } }
        ]).toArray();

          const totalEnergyUsed = sensorConsumption.length > 0 ?
sensorConsumption[0].totalConsumed : 0;

        // Hitung RemainingEnergy dengan validasi batas
                    const    remainingEnergy    =    Math.max(0,
consumerHistory.totalEnergy - totalEnergyUsed);

        // Tentukan status relay berdasarkan RemainingEnergy
        const relayStatus = remainingEnergy > 0 ? "open" : "closed";

        // Ambil data existing dari energy_right
          const existingEnergyRight = await energyRight.findOne({
wallet });

         // Jika RemainingEnergy dan relayStatus sama dengan data
yang ada, jangan perbarui timestamp
        if (
            existingEnergyRight &&
            existingEnergyRight.RemainingEnergy === remainingEnergy
&&
            existingEnergyRight.relayStatus === relayStatus
        ) {
            return {
                wallet,
              RemainingEnergy: existingEnergyRight.RemainingEnergy,
                relayStatus: existingEnergyRight.relayStatus,
                lastUpdated: existingEnergyRight.lastUpdated,
            };
        }

        // Ambil waktu saat ini hanya jika terjadi perubahan
        const currentTime = getCurrentTime();
```

```
        // Perbarui koleksi energy_right
        await energyRight.updateOne(
            { wallet },
            {
                $set: {
                    RemainingEnergy: remainingEnergy,
                    relayStatus: relayStatus,
                    lastUpdated: currentTime,
                },
            },
             { upsert: true } // Tambahkan dokumen baru jika tidak
ada
        );

        console.log(
              `Synced energy_right for wallet ${wallet}. Remaining
Energy: ${remainingEnergy}, Relay Status: ${relayStatus}`
        );

        return {
            wallet,
            RemainingEnergy: remainingEnergy,
            relayStatus: relayStatus,
            lastUpdated: currentTime,
        };
    } catch (error) {
            console.error(`Error  syncing  energy_right  for  wallet
${wallet}:`, error.message);
        throw error;
    }
}

app.post('/syncEnergyRight', async (req, res) => {
    const { wallet } = req.body;

    if (!wallet) {
        return res.status(400).json({
            status: 'Error',
            message: 'Missing wallet address',
        });
    }

    try {
        const result = await syncEnergyRight(wallet);
        res.json({
            status: 'Success',
             message: `Energy right synced successfully for wallet
${wallet}`,
        });
    } catch (error) {
        res.status(500).json({
            status: 'Error',
            message: error.message,
        });
```

```
    }
});

////////////////////// RelayStart //////////////////////

app.get("/relayStatus", async (req, res) => {
    const client = new MongoClient(uri);
    try {
      await client.connect();
      const db = client.db(dbName);

                        const       consumerRelay       =       await
db.collection("consumerRelay").find().toArray();
                         const       energyRight       =       await
db.collection("energy_right").find().toArray();

      const result = consumerRelay.map(relay => {
        const match = energyRight.find(er => er.wallet.toLowerCase()
=== relay.wallet.toLowerCase());
        const status = match?.relayStatus === "open" ? "on" : "off";

        return {
          relay: relay.relay,
          pin: relay.pin,
          consumer: relay.consumer,
          status,
        };
      });

      res.json(result);
    } catch (error) {
      console.error("Error fetching relay status:", error);
      res.status(500).send("Server Error");
    } finally {
      await client.close();
    }
  });

////////////////////// RelayEnd //////////////////////

let walletTarget = "";

////////////////////////// Histosy //////////////////////
app.post('/usageHistory', async (req, res) => {
    const { wallet } = req.body;

    if (!wallet || !ethers.utils.isAddress(wallet)) {
        return res.status(400).json({
            status: 'Error',
            message: 'Invalid or missing wallet address.',
        });
    }

    try {
```

```
        await connectDB();

        const data = await energylogSensor.findOne({ wallet });
        if (!data) {
            return res.status(404).json({
                status: 'Error',
                                message:   'Wallet   not   found   in
consumed_energy_sensor.',
            });
        }

        const usageHistory = data.usageHistory.map((entry) => ({
            timestamp: entry.timestamp,
            energyUsed: entry.energyUsed,
        }));

        if (usageHistory.length < 2) {
            return res.status(400).json({
                status: 'Error',
                message: 'Insufficient data for analysis.',
            });
        }

        // 2. Ambil nama konsumen yang terhubung dengan wallet dari
database
       const consumerEntry = await consumerRelayCollection.findOne({
         wallet: { $regex: new RegExp(`^${wallet}$`, "i") } // case-
insensitive
        });
        if (!consumerEntry) {
            return res.status(404).json({
              status: 'Error',
              message: 'Consumer not found for this wallet.',
            });
          }
          const consumerName = consumerEntry.consumer;

          //  🔹 ambil data relay dari endpoint /relayStatus
                            const     responseRelay     =     await
axios.get('http://localhost:3000/relayStatus');
          const dataRelay = responseRelay.data;

          //  🔹 cari relay yang sesuai dengan nama konsumen
          const relayTarget = dataRelay.find(r =>
            r.consumer.toLowerCase() === consumerName.toLowerCase()
          );

          const relayStatus = relayTarget?.status || 'unknown';

         const maxEnergy = Math.max(...usageHistory.map((entry) =>
entry.energyUsed));
        const doc = new PDFDocument({
            size: 'A4',
            margins: { top: 85, bottom: 85, left: 115, right: 85 },
```

```
        });
        const uniqueId = uuidv4();
        const fileName = `${wallet}_${uniqueId}_usage_history.pdf`;

        res.setHeader('Content-Type', 'application/pdf');
                res.setHeader('Content-Disposition',   `attachment;
filename=${fileName}`);
        doc.pipe(res);

        // Bab 1: Informasi Sensor dan Wallet
        doc.fontSize(18).text('Laporan Penggunaan Energi', { align:
'center' });
        doc.moveDown(2);

        // Informasi umum
          doc.fontSize(12).text('Informasi  Konsumer',  {  underline:
true });
        doc.moveDown();
        doc.text(`Alamat Wallet: ${wallet}`);
                            doc.text(`Total     Energi     Digunakan:
${data.TotalEnergyUsed.toFixed(4)} Wh`);
        doc.text(`ID Sensor: ${data.sensorId}`);
        doc.text(`Status Relay: ${relayStatus}`);
        doc.moveDown();

        // Tambahan detail terkait consumer
        const firstUsageDate = data.first_timestamp;
        const lastUpdateDate = data.last_update;

        doc.text(`Penggunaan Pertama Tercatat: ${firstUsageDate}`);
        doc.text(`Pembaharuan Terakhir: ${lastUpdateDate}`);
                  doc.text(`Jumlah   Data   Penggunaan   Tercatat:
${Math.min(data.usageHistory.length,  3600)}  entri  terakhir  yang
dianalisis`);
        doc.text(
             'Laporan ini memberikan detail penggunaan energi Anda
secara rinci berdasarkan data yang telah dikumpulkan oleh sistem.'
        );
        doc.moveDown(2);
        doc.text(
              'Mohon periksa setiap bagian laporan untuk informasi
lebih lanjut mengenai pola penggunaan energi Anda.'
        );
        doc.addPage();

// Konfigurasi untuk tabel Bab 2
const configBab2 = {
    rowsPerPage: 60,
    columnWidths: [50, 350, 100],
    headers: ['No.', 'Timestamp', 'Energy Used (Wh)'],
    fontSize: 9,
    headerFontSize: 10,
    rowSpacing: 10,              // jarak antar baris data
    headerToLineSpacing: 5,      // jarak header dengan garis bawah
```

```
    lineToDataSpacing: 5,       // jarak garis dengan data
    margin: {                    // margin kertas
        top: 50,
        bottom: 10,
        left: 50,
        right: 50
    }
};

const limitedUsageHistory = usageHistory.slice(0, 3600);
let currentPage = 2;

for (let i = 0; i < limitedUsageHistory.length; i += 
configBab2.rowsPerPage) {
    if (i > 0) doc.addPage();
    doc.fontSize(14).text(`Bab 2: Detail Penggunaan Energi - Halaman 
${currentPage}`, { align: 'center' });
    doc.moveDown(2);
    currentPage++;

        const chunk = limitedUsageHistory.slice(i, i + 
configBab2.rowsPerPage);

    // Posisi vertikal awal tabel
       const tableTop = doc.y < configBab2.margin.top ? 
configBab2.margin.top : doc.y;

    // Header tabel
    configBab2.headers.forEach((header, idx) => {
                const xPos = configBab2.margin.left + 
configBab2.columnWidths.slice(0, idx).reduce((a, b) => a + b, 0);
        doc.fontSize(configBab2.headerFontSize).text(header, xPos, 
tableTop, {
            width: configBab2.columnWidths[idx],
            align: 'center',
        });
    });

    // Garis bawah header
    const headerLineY = tableTop + configBab2.headerToLineSpacing + 
configBab2.headerFontSize;
    doc.moveTo(configBab2.margin.left, headerLineY)
                           .lineTo(configBab2.margin.left + 
configBab2.columnWidths.reduce((a,b) => a + b, 0), headerLineY)
       .stroke();

    // Isi tabel
    const dataStartY = headerLineY + configBab2.lineToDataSpacing;
    chunk.forEach((entry, idx) => {
        const rowY = dataStartY + idx * configBab2.rowSpacing;
        const row = [
            i + idx + 1,
            entry.timestamp,
            entry.energyUsed.toFixed(4),
```

```
        ];
        row.forEach((cell, colIdx) => {
                        const   xPos   =   configBab2.margin.left   +
configBab2.columnWidths.slice(0, colIdx).reduce((a, b) => a + b, 0);
            doc.fontSize(configBab2.fontSize).text(cell, xPos, rowY,
{
                width: configBab2.columnWidths[colIdx],
                align: 'center',
             });
        });
    });
}

////////////////////////////// Bab 3: Summary - visualisasi data -
Start ////////////////////////////////////////
// Konfigurasi dasar
const config = {
    maxDataPoints: 3600,
    groupSize: 60,
    canvasWidth: 800,
    canvasHeight: 400,
    scaleFactor: 5,
    colors: {
        background: 'white',
        axis: 'black',
        dataLine: 'blue',
        trendLineGroup: 'red',
        trendLineGlobal: 'green',
        dataPoint: 'black'
    },
    fontSize: {
        title: 14,
        axisLabel: 8,
        legend: 12
    },
    showTrendLineGroup: true,   // true = tampilkan garis tren group,
false = sembunyikan
     showTrendLineGlobal: false    // true = tampilkan garis tren
global, false = sembunyikan
};

// Ambil 3600 data terakhir dari usageHistory
const recentData = usageHistory.slice(-config.maxDataPoints);
const analysisData = recentData;
const     actual     =     analysisData.slice(1).map((entry)     =>
entry.energyUsed);

// Regresi Linear untuk semua data
const n = actual.length;
const sumX = Array.from({ length: n }, (_, i) => i).reduce((a, b) =>
a + b, 0);
const sumY = actual.reduce((a, b) => a + b, 0);
const sumXY = actual.reduce((sum, y, i) => sum + i * y, 0);
```

```
const sumX2 = Array.from({ length: n }, (_, i) => i ** 2).reduce((a,
b) => a + b, 0);

const slope = ((n * sumXY - sumX * sumY) / (n * sumX2 - sumX ** 2));
const intercept = (sumY - slope * sumX) / n;

// Hitung rata-rata per 60 data (menit) -> menghasilkan 60 data
const averagedData = [];
for (let i = 0; i < recentData.length; i += config.groupSize) {
    const chunk = recentData.slice(i, i + config.groupSize);
    const avgEnergy = chunk.reduce((sum, d) => sum + d.energyUsed,
0) / chunk.length;
    averagedData.push({
        energyUsed: avgEnergy,
        timestamp: `${chunk[0].timestamp} - ${chunk[chunk.length -
1].timestamp}`
    });
}

// Skala untuk data global (3600 data)
const maxEnergyGlobal = Math.max(...actual);
const    maxEnergyGroup    =    Math.max(...averagedData.map(d    =>
d.energyUsed));
const    scaleXGroup    =    (config.canvasWidth    -    100)    /
(averagedData.length - 1);
const maxY = Math.max(maxEnergyGlobal, maxEnergyGroup);
const scaleY = (config.canvasHeight - 100) / maxY;
const predicted = actual.map((_, i) => slope * i + intercept);
const scaleXGlobal = (config.canvasWidth - 100) / (actual.length -
1);

// Canvas dan context
const canvasWidth = config.canvasWidth * config.scaleFactor;
const canvasHeight = config.canvasHeight * config.scaleFactor;
const canvas = createCanvas(canvasWidth, canvasHeight);
const ctx = canvas.getContext('2d');
ctx.scale(config.scaleFactor, config.scaleFactor);

// Background
ctx.fillStyle = config.colors.background;
ctx.fillRect(0, 0, config.canvasWidth, config.canvasHeight);

// Gambar axis
ctx.strokeStyle = config.colors.axis;
ctx.lineWidth = 2;
// X axis
ctx.beginPath();
ctx.moveTo(50, config.canvasHeight - 50);
ctx.lineTo(config.canvasWidth - 50, config.canvasHeight - 50);
ctx.stroke();
// Y axis
ctx.beginPath();
ctx.moveTo(50, 50);
ctx.lineTo(50, config.canvasHeight - 50);
```

```
ctx.stroke();


// Fungsi regresi linear (input data array, output {slope,
intercept})
function linearRegression(data) {
    const n = data.length;
    const xVals = data.map((_, i) => i);
    const yVals = data.map(d => d.energyUsed);
    const sumX = xVals.reduce((a, b) => a + b, 0);
    const sumY = yVals.reduce((a, b) => a + b, 0);
    const sumXY = data.reduce((sum, d, i) => sum + i * d.energyUsed,
0);
    const sumX2 = xVals.reduce((sum, x) => sum + x * x, 0);

    const slope = (n * sumXY - sumX * sumY) / (n * sumX2 - sumX *
sumX);
    const intercept = (sumY - slope * sumX) / n;
    return { slope, intercept };
}

// Hitung regresi linear untuk group dan global data
const regGroup = linearRegression(averagedData);

// Gambar trendline group (merah)
if (config.showTrendLineGroup) {
    ctx.strokeStyle = config.colors.trendLineGroup;
    ctx.lineWidth = 0.25;
    ctx.beginPath();
    ctx.moveTo(
        50,
            config.canvasHeight - 50 - (regGroup.intercept +
regGroup.slope * 0) * scaleY
    );
    ctx.lineTo(
        50 + (averagedData.length - 1) * scaleXGroup,
            config.canvasHeight - 50 - (regGroup.intercept +
regGroup.slope * (averagedData.length - 1)) * scaleY
    );
    ctx.stroke();
}

// Gambar trendline global (hijau)
if (config.showTrendLineGlobal) {
    ctx.strokeStyle = config.colors.trendLineGlobal;
    ctx.lineWidth = 0.25;
    ctx.beginPath();
    ctx.moveTo(
        50,
        config.canvasHeight - 50 - (intercept + slope * 0) * scaleY
    );
    ctx.lineTo(
        50 + (n - 1) * scaleXGlobal,
```

```
        config.canvasHeight - 50 - (intercept + slope * (n - 1)) *
scaleY
    );
    ctx.stroke();
}

// Gambar line chart data group (biru)
ctx.strokeStyle = config.colors.dataLine;
ctx.lineWidth = 1.5;
ctx.beginPath();
averagedData.forEach((d, i) => {
    const x = 50 + i * scaleXGroup;
    const y = config.canvasHeight - 50 - d.energyUsed * scaleY;
    if (i === 0) ctx.moveTo(x, y);
    else ctx.lineTo(x, y);
});
ctx.stroke();

// Gambar titik data group (hitam)
ctx.fillStyle = config.colors.dataPoint;
averagedData.forEach((d, i) => {
    const x = 50 + i * scaleXGroup;
    const y = config.canvasHeight - 50 - d.energyUsed * scaleY;
    ctx.beginPath();
    ctx.arc(x, y, 8 / config.scaleFactor, 0, Math.PI * 2);
    ctx.fill();

    // Label X (index menit)
    ctx.font = `${config.fontSize.axisLabel}px Arial`;
    ctx.textAlign = 'center';
    ctx.fillText(i + 1, x, config.canvasHeight - 30);
});

// Label Y axis
const numYLabels = 5;
ctx.fillStyle = config.colors.dataPoint;
ctx.font = `${48 / config.scaleFactor}px Arial`;
ctx.textAlign = 'right';
for (let i = 0; i <= numYLabels; i++) {
    const value = (maxY / numYLabels) * i;
    const y = config.canvasHeight - 50 - value * scaleY;
    ctx.fillText(value.toFixed(4), 45, y + 5);
}

// Legend
ctx.font = `${config.fontSize.legend}px Arial`;
ctx.textAlign = 'left';

// Legend: data line (biru)
ctx.fillStyle = config.colors.dataLine;
ctx.beginPath();
ctx.arc(50 + 10, config.canvasHeight - 10, 5, 0, Math.PI * 2);
ctx.fill();
ctx.fillStyle = config.colors.dataPoint;
```

```
ctx.fillText('Penggunaan Energi (rata-rata 60 detik)', 50 + 30,
config.canvasHeight - 5);

// Legend: trendline group (merah)
if (config.showTrendLineGroup) {
ctx.fillStyle = config.colors.trendLineGroup;
ctx.beginPath();
ctx.arc(50 + 300, config.canvasHeight - 10, 5, 0, Math.PI * 2);
ctx.fill();
ctx.fillStyle = config.colors.dataPoint;
ctx.fillText('Garis Tren Group (60 data)', 50 + 320,
config.canvasHeight - 5);}

// Legend: trendline global (hijau)
if (config.showTrendLineGlobal) {
ctx.fillStyle = config.colors.trendLineGlobal;
ctx.beginPath();
ctx.arc(50 + 520, config.canvasHeight - 10, 5, 0, Math.PI * 2);
ctx.fill();
ctx.fillStyle = config.colors.dataPoint;
ctx.fillText('Garis Tren Global (3600 data)', 50 + 540,
config.canvasHeight - 5);}

// Tambahkan judul ke PDF
doc.addPage({ size: 'A4', layout: 'landscape' });
doc.fontSize(config.fontSize.title).text('Bab 3: Summary -
Visualisasi Data', { align: 'center' });
doc.moveDown(1);

// Simpan canvas ke image dan masukkan ke PDF
const buffer = canvas.toBuffer('image/png');
const imagePath = 'chart_summary.png';
fs.writeFileSync(imagePath, buffer);
doc.image(imagePath, { fit: [700, 300], align: 'center' });

// Tambahkan keterangan slope dan intercept
doc.moveDown(20); // jarak sekitar 12px pada fontSize default
(sekitar 14pt)
doc.fontSize(8).text(`Regresi Linear Global (3600 data): y =
${slope.toFixed(16)}x + ${intercept.toFixed(16)}`);
doc.text(`Regresi Linear Group (60 grup): y =
${regGroup.slope.toFixed(16)}x +
${regGroup.intercept.toFixed(16)}`);

fs.unlinkSync(imagePath);

////////////////////////////// Bab 3: Summary - visualisasi data -
End ////////////////////////////////////////////

////////////////////////////////// Bab 3: Summary - Summary of data -
Start ////////////////////////////////////////////
// Bab 3: Summary - Section 2
doc.addPage({ size: 'A4', layout: 'landscape' });
```

```
doc.fontSize(config.fontSize.title).text('Bab 3: Summary - Hasil',
{ align: 'center' });
doc.moveDown(2);

// Section 2: Analisis MAPE, Regresi Linear, dan Saran Energi
doc.fontSize(16).text('Bagian 2: Analisis MAPE, Regresi Linear, dan
Saran Energi', { underline: true });
doc.moveDown();

// Evaluasi MAPE
const mapeValues = actual.map((y, i) => {
    const yPred = predicted[i];
    if (y === 0) return null;
    return Math.abs((y - yPred) / y);
}).filter(v => v !== null);

const mape = (mapeValues.reduce((a, b) => a + b, 0) /
mapeValues.length * 100).toFixed(2);

// Saran pembelian energi = total prediksi untuk 1 hari (dengan tren
yang sama)
let totalPrediksi = 0;
for (let i = 0; i < n; i++) {
    const prediksiY = slope * i + intercept;
    totalPrediksi += prediksiY;
}
const rataRataPrediksi = totalPrediksi / n;

// Hitung saran pembelian untuk 1 hari berdasarkan rata-rata prediksi
let nextDayPrediction = rataRataPrediksi * 86400;

// Poin-Poin Analisis
doc.fontSize(14).text('Poin-Poin Analisis:', { underline: true });
doc.moveDown();

doc.fontSize(12).list([
    `Analisis dilakukan pada ${config.maxDataPoints} data terakhir`,
    `MAPE (Mean Absolute Percentage Error): ${mape}%`,
    `Persamaan regresi linear global : Y = ${slope.toFixed(16)}X +
${intercept.toFixed(16)}`,
          `Persamaan    regresi    linear    kelompok:    Y    =
${regGroup.slope.toFixed(16)}x                                    +
${regGroup.intercept.toFixed(16)}`,
     `Tren penggunaan energi: ${slope > 0 ? 'Naik' : slope < 0 ?
'Turun' : 'Stabil'}`,
    `Rata-rata prediksi: ${rataRataPrediksi.toFixed(20)} Wh`,
       `Saran   energi   untuk   pembelian   berikutnya   (1   hari):
${nextDayPrediction.toFixed(8)} Wh`,
               `Nilai       maksimum       penggunaan       energi:
${Math.max(...actual).toFixed(4)} Wh`,
               `Nilai       minimum       penggunaan       energi:
${Math.min(...actual).toFixed(4)} Wh`,
    `Rata-rata penggunaan energi: ${(sumY / n).toFixed(20)} Wh`
]);
```

```
// Keterangan
doc.moveDown(1);
doc.fontSize(12).text('Keterangan:', { underline: true });
doc.moveDown();

doc.fontSize(10).list([
     'Analisis dilakukan pada 3600 data terakhir untuk memastikan
relevansi dan mengurangi beban komputasi',
    'Semua perhitungan statistik dan prediksi didasarkan pada 3600
data terakhir yang diproses',
     'Grouping (pengelompokan) adalah data rata-rata tiap 60 data
asli yang dikelompokkan',
     'Data dari MAPE dan analisis merupakan data asli yang belum
dikelompokkan',
      'Visualisasi adalah data yang telah dikelompokkan untuk
mengurangi beban server',
     'Trendline pada visualisasi menggunakan persamaan yang sudah
dikelompokkan'
], { numbered: true });

// Rata-rata tiap grup
doc.addPage({ size: 'A4', layout: 'portrait' });
doc.fontSize(config.fontSize.title).text('Bab 3: Summary - Rata-rata
Tiap Grup (60 data per grup)', {align: 'center',});
doc.moveDown();

const groupAverages = averagedData.map((d, i) => `Grup ${i + 1}:
${d.energyUsed.toFixed(5)} Wh`);
doc.fontSize(9).list(groupAverages);

doc.end();
////////////////////////////// Bab 3: Summary - Summary of data -
End ////////////////////////////////////////

            } catch (error) {
                     console.error('Error generating usage history
report:', error.message);
                res.status(500).json({
                    status: 'Error',
                    message: 'Internal server error.',
                });
            }
        });

//////////////////////ADMIN//////////////////////////
let activeSessions = {}; // Penyimpanan sesi aktif
const userSessions = {};

// Rute untuk halaman login
appSeller.get('/', (req, res) => {
    res.sendFile(path.join(__dirname, 'private', 'login.html'));
});
```

```
// Login endpoint
appSeller.post('/private/login', (req, res) => {
    const { username, password } = req.body;

    if (username === SELLER && password === SELLER_PASS) {
        // Cek apakah user sudah ada session aktif
        const oldUUID = userSessions[username];
        if (oldUUID) {
            // Hapus session lama
            delete activeSessions[oldUUID];
              console.log(`Session lama untuk ${username} dihapus:
${oldUUID}`);
        }

        // Buat session baru
        const uuid = uuidv4();
        activeSessions[uuid] = {
            username,
            active: true
        };
        userSessions[username] = uuid;  // update UUID aktif untuk
user

         console.log(`Login berhasil. UUID baru untuk ${username}:
${uuid}`);
        return res.json({
            message: 'Login successful',
            uuid,
            redirectUrl: `/private_index.html?uuid=${uuid}`
        });
    }

    res.status(401).json({ message: 'Invalid credentials' });
});

// Middleware validasi sederhana
function validateSellerUUID(req, res, next) {
    const uuid = req.query.uuid || req.body.uuid;

    if (!uuid || !activeSessions[uuid]?.active) {
        console.log('Akses ditolak untuk UUID:', uuid);
       return res.status(403).json({ message: 'Session invalid' });
    }

    next();
}

appSeller.get('/energyRight', validateSellerUUID, async (req, res)
=> {
    try {
        const uuid = req.query.uuid;
        console.log(`Accessing energyData with UUID: ${uuid}`);

        await connectDB();
```

```
        const energyData = await energyRight.find({}).toArray();
        res.json(energyData);
    } catch (error) {
        console.error('Error:', error);
        res.status(500).json({ message: 'Server error' });
    }
});

appSeller.post('/CheckSellerETH', validateSellerUUID, async (req,
res) => {
    const { walletAddress, uuid } = req.body;

    try {
        const balance = await provider.getBalance(walletAddress);
        res.json({
            status: "Success",
            balance: ethers.utils.formatEther(balance),
            walletAddress,
            uuid
        });
    } catch (error) {
        console.error('ETH Check Error:', error);
        res.status(500).json({
            status: "Error",
            message: error.message
        });
    }
});

appSeller.post('/private/logout', validateSellerUUID, (req, res) =>
{
    const uuid = req.query.uuid || req.body.uuid;
    delete activeSessions[uuid];
    console.log(`Logout successful for UUID: ${uuid}`);
    res.json({ message: 'Logged out' });
});

appSeller.get('/monitoring', validateSellerUUID, async (req, res) =>
{
    try {
        const { uuid } = req.query;
       console.log(`Accessing monitoring data with UUID: ${uuid}`);

        await connectDB();

         // Ambil semua data monitoring (atau filter by UUID jika
diperlukan)
        const rawData = await energylogSensor.find({}).toArray();

        // Batasi usageHistory untuk setiap dokumen
        const limitedData = rawData.map(doc => {
            const usageHistory = doc.usageHistory || [];
           const limitedUsage = usageHistory.slice(-3600); // ambil
3600 data terakhir
```

```
            return {
                ...doc,
                usageHistory: limitedUsage
            };
        });

        res.json({
            status: 'Success',
            message: 'Monitoring data retrieved successfully',
            data: limitedData,
        });
    } catch (error) {
        console.error('Error retrieving monitoring data:', error);
        res.status(500).json({
            status: 'Error',
            message: 'Server error',
        });
    }
});

//////////////////////ADMIN END//////////////////////////////

async function syncAllWalletsEverySecond() {
    try {
        await connectDB(); // Pastikan koneksi hanya dibuka sekali

                            const      allMappings      =      await
consumerRelayCollection.find({}).toArray();
        const wallets = allMappings.map(mapping => mapping.wallet);

        async function loop() {
            for (const wallet of wallets) {
                try {
                    const result = await syncEnergyRight(wallet);
                  console.log(`[SYNC] Wallet: ${wallet} | Remaining:
${result.RemainingEnergy} | Status: ${result.relayStatus}`);
                } catch (err) {
                       console.warn(`[ERROR] Failed syncing wallet
${wallet}: ${err.message}`);
                }
            }

            setTimeout(loop, 1000); // Delay 1 detik sebelum ulangi
loop
        }

        loop(); // Mulai loop pertama
    } catch (err) {
        console.error("[INIT SYNC LOOP ERROR]", err.message);
    }
}

////////////////////// tambahan //////////////////
app.post('/post', async (req, res) => {
```

```
    const { wallet } = req.body;
    if (!wallet) {
          return res.status(400).json({ status: 'Error', message:
'Wallet is required.' });
    }

    try {
        await connectDB();

        const data = await energylogSensor.findOne({ wallet });
        if (!data || !data.usageHistory) {
            return res.status(404).json({ status: 'Error', message:
'Data not found.' });
        }

        const recentData = data.usageHistory.slice(-3600);

        // Sheet 1: rata-rata per grup
        const groupSize = 60;
        const averagedData = [];
        for (let i = 0; i < recentData.length; i += groupSize) {
            const chunk = recentData.slice(i, i + groupSize);
               const avgEnergy = chunk.reduce((sum, d) => sum +
d.energyUsed, 0) / chunk.length;
            averagedData.push({
                Grup: `Grup ${i / groupSize + 1}`,
                RataRataEnergi: parseFloat(avgEnergy.toFixed(5)),
                          RentangWaktu: `${chunk[0].timestamp} -
${chunk[chunk.length - 1].timestamp}`
            });
        }

        // Sheet 2: data mentah
        const rawData = recentData.map((entry, index) => ({
            No: index + 1,
            Timestamp: entry.timestamp,
            Energi: parseFloat(entry.energyUsed.toFixed(5))
        }));

        const wb = XLSX.utils.book_new();
        const ws1 = XLSX.utils.json_to_sheet(averagedData);
        const ws2 = XLSX.utils.json_to_sheet(rawData);
        XLSX.utils.book_append_sheet(wb, ws1, 'Rata-rata Grup');
        XLSX.utils.book_append_sheet(wb, ws2, 'Data Mentah');

        // Simpan ke file sementara
       const filePath = path.join(__dirname, 'data_rata_rata.xlsx');
        XLSX.writeFile(wb, filePath);

        // Kirim file ke client
        res.download(filePath, 'data_rata_rata.xlsx', (err) => {
            if (err) {
                console.error('Download error:', err);
```

```
                res.status(500).json({ status: 'Error', message:
'Gagal mengunduh file.' });
            }

            // Hapus file setelah dikirim (opsional)
            fs.unlink(filePath, () => {});
        });
    } catch (err) {
        console.error(err);
         res.status(500).json({ status: 'Error', message: 'Server
error.' });
    }
});

appSeller.listen(SELLER_PORT, () => {
            console.log(`Seller     server     is     running     at
http://localhost:${SELLER_PORT}`);
});

app.listen(port, () => {
    console.log(`Server berjalan di http://localhost:${port}`);
});

syncAllWalletsEverySecond();
```

Lampiran B3 – Program Penerima Serial ESP32 ke Server

```
const { SerialPort } = require('serialport'); // <- destruktur dari
module
const { ReadlineParser } = require('@serialport/parser-readline');
const axios = require('axios');
const endpointURL = 'http://localhost:3000/logEnergyConsumption'; //
Ganti dengan IP jika server beda

// Ganti sesuai portmu
const portName = 'COM15';
const baudRate = 115200;

const port = new SerialPort({ path: portName, baudRate: baudRate });
const parser = port.pipe(new ReadlineParser({ delimiter: '\n' }));

port.on('open', () => {
   console.log(`Serial port ${portName} terbuka dengan baudrate
${baudRate}`);
});

parser.on('data', (data) => {
  data = data.trim();

  if (data.startsWith('{') && data.endsWith('}')) {
    try {
      const json = JSON.parse(data);
      console.log("✅ JSON diterima dari ESP32:");
      console.log(json);

      // Kirim ke endpoint
      axios.post(endpointURL, json)
        .then(response => {
         console.log(`📤 Terkirim ke server (${response.status}):`,
response.data);
        })
        .catch(error => {
         console.error(`❌ Gagal kirim ke server:`, error.message);
        });

    } catch (err) {
      console.error("❌ Gagal parsing JSON:", err.message);
    }
  }
});

port.on('error', (err) => {
  console.error('Terjadi kesalahan pada port serial:', err.message);
});
```

LAMPIRAN C ESP32

Lampiran C1 – Program ESP32

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ModbusMaster.h>
#include <ArduinoJson.h>

// --- WiFi ---
const char* ssid = "mboh";
const char* password = "12345678";
const                char*                serverUrl                =
"http://192.168.1.2:3000/logEnergyConsumption";
const char* relayStatusUrl = "http://192.168.1.2:3000/relayStatus";

struct SensorInfo {
  uint8_t id;
  const char* wallet;
};

SensorInfo sensorMap[] = {
  {1, "0x9A43b29497aebF8691E8Fa36760d7F57222b86e4"},
  {2, "0x587a768aa01bd43f8Aa720A4539DAF8CB96CB979"},
  {3, "0x6E8b56414035d7F8C6635A911beDb1044203fAf6"},
};

// --- Relay Control ---
const int relayPins[] = {27, 25, 32};
const int numRelays = 3;

// --- Modbus RTU ---
#define RXD2 16
#define TXD2 17
#define NUM_REGISTERS 10
#define SCAN_RATE 1000  // 5 detik

ModbusMaster node1, node2, node3;
HardwareSerial ModbusSerial(1);
uint32_t lastScan = 0;

// Task handle
TaskHandle_t TaskRelayHandle;
TaskHandle_t TaskModbusHandle;
TaskHandle_t TaskHTTPHandle;
// --- Deklarasi global ---
SemaphoreHandle_t modbusMutex;
void TaskHTTPStatus(void *pvParameters);  // Deklarasi awal
void checkWiFi();                         // Deklarasi awal
bool relayStatus[numRelays] = {false, false, false};  // Default ON

WiFiServer server(80);

void setup() {
  Serial.begin(115200);
```

```
  modbusMutex = xSemaphoreCreateMutex();
  // Relay
  for (int i = 0; i < numRelays; i++) {
    pinMode(relayPins[i], OUTPUT);
    digitalWrite(relayPins[i], LOW);
  }

  // WiFi
  WiFi.begin(ssid, password);
  Serial.print("Menghubungkan WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi      Tersambung,      IP:      "      +
WiFi.localIP().toString());
  server.begin();

  // Modbus
  ModbusSerial.begin(9600, SERIAL_8N1, RXD2, TXD2);
  node1.begin(1, ModbusSerial);
  node2.begin(2, ModbusSerial);
  node3.begin(3, ModbusSerial);

  // Multitasking
  xTaskCreatePinnedToCore(TaskRelay,  "TaskRelay",  3000,  NULL,  1,
&TaskRelayHandle, 1);
  xTaskCreatePinnedToCore(TaskModbus, "TaskModbus", 5000, NULL, 1,
&TaskModbusHandle, 0);
  xTaskCreatePinnedToCore(TaskHTTPStatus,  "TaskHTTPStatus",  4000,
NULL, 1, &TaskHTTPHandle, 1);
}

void loop() {
  WiFiClient client = server.available();
  if (client) {
    String request = client.readStringUntil('\r');
    client.flush();

    // Parse GET relay1/relay2/relay3
    if (request.indexOf("GET /relay") >= 0) {
      int relayNum = request.charAt(10) - '1';  // '1'->0
      if (relayNum >= 0 && relayNum < numRelays) {
        digitalWrite(relayPins[relayNum],
!digitalRead(relayPins[relayNum]));
        client.print("HTTP/1.1       200       OK\r\nContent-Type:
text/plain\r\n\r\nRelay ");
        client.print(relayNum + 1);
        client.print(" -> ");
        client.print(digitalRead(relayPins[relayNum])   ?   "ON"   :
"OFF");
      } else {
        client.print("HTTP/1.1   400   Bad   Request\r\n\r\nInvalid
Relay");
```

```
      }
    } els
      client.print("HTTP/1.1 404 Not Found\r\n\r\n");
    }

    client.stop();
  }
}

// -------- Task Relay (Serial manual control) --------
void TaskRelay(void *pvParameters) {
  while (1) {
    if (Serial.available() > 0) {
      char command = Serial.read();
      if (command >= '1' && command <= '3') {
        int relayNum = command - '1';
        digitalWrite(relayPins[relayNum],
!digitalRead(relayPins[relayNum]));
        Serial.printf("Relay %d sekarang: %s\n", relayNum + 1,
digitalRead(relayPins[relayNum]) ? "AKTIF" : "NON-AKTIF");
      }
    }
    vTaskDelay(10 / portTICK_PERIOD_MS);
  }
}

// -------- Task Modbus --------
void TaskModbus(void *pvParameters) {
  while (1) {
    if (millis() - lastScan >= SCAN_RATE) {
      lastScan = millis();
      if (relayStatus[0]) readAndSend(node1, 1);
      if (relayStatus[1]) readAndSend(node2, 2);
      if (relayStatus[2]) readAndSend(node3, 3);
    }
  }
}

// -------- Fungsi Baca & Kirim Data --------
void readAndSend(ModbusMaster &node, uint8_t sensorId) {
  StaticJsonDocument<300> doc;

  // Identifikasi sensor
  const char* walletAddr = "";
  char sensorIdStr[12];
  snprintf(sensorIdStr,     sizeof(sensorIdStr),     "sensor-%03d",
sensorId);

  for (SensorInfo s : sensorMap) {
    if (s.id == sensorId) {
      walletAddr = s.wallet;
      break;
    }
  }
```

```
  doc["sensorId"] = sensorIdStr;
  doc["wallet"] = walletAddr;

  // Baca Modbus
  if (xSemaphoreTake(modbusMutex, pdMS_TO_TICKS(1000)) == pdTRUE) {
    uint8_t result = node.readInputRegisters(0x0000, NUM_REGISTERS);
    if (result == node.ku8MBSuccess) {
      uint16_t reg[NUM_REGISTERS];
      for (int i = 0; i < NUM_REGISTERS; i++) reg[i] =
node.getResponseBuffer(i);

      doc["voltage"] = reg[0] * 0.1;
      doc["current"] = ((uint32_t)reg[2] << 16 | reg[1]) * 0.001;
      doc["power"] = ((uint32_t)reg[4] << 16 | reg[3]) * 0.1;
      doc["energy"] = ((uint32_t)reg[6] << 16 | reg[5]) * 1.0;
      doc["frequency"] = reg[7] * 0.1;
      doc["powerFactor"] = reg[8] * 0.01;
    } else {
      Serial.printf("[Modbus] Gagal baca ID %d | Error: 0x%02X\n",
sensorId, result);
    }
    xSemaphoreGive(modbusMutex);
  } else {
    Serial.printf("[Modbus] Timeout akses Modbus ID %d\n",
sensorId);
  }

  // Kirim ke Serial, baik berhasil atau gagal
  String payload;
  serializeJson(doc, payload);
  Serial.println(payload);
}

// -------- Task HTTP Status --------
void TaskHTTPStatus(void *pvParameters) {
  while (1) {
    checkWiFi();
    if (WiFi.status() == WL_CONNECTED) {
      HTTPClient http;
      http.begin(relayStatusUrl);
      int httpCode = http.GET();

      if (httpCode == 200) {
        String payload = http.getString();
        StaticJsonDocument<512> doc;
        DeserializationError error = deserializeJson(doc, payload);

        if (!error && doc.is<JsonArray>()) {
          for (JsonObject obj : doc.as<JsonArray>()) {
            int relayNum = obj["relay"]; // 1-based
            String status = obj["status"];
            int pin = obj["pin"];
```

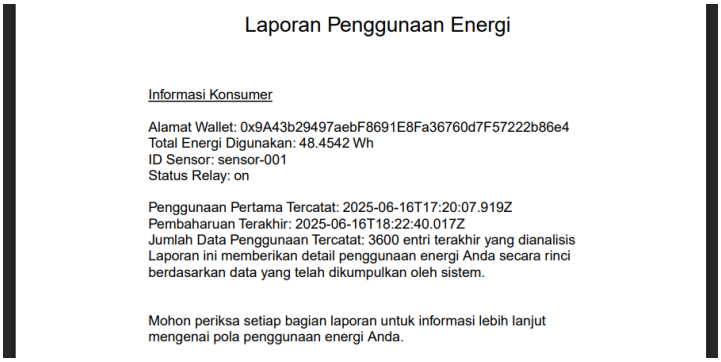
```
          if (relayNum >= 1 && relayNum <= numRelays) {
            bool state = (status == "on");
            digitalWrite(pin, state ? HIGH : LOW);
            relayStatus[relayNum - 1] = state;   // Simpan status
untuk pembacaan
            Serial.printf("[UPDATE] Relay %d (Pin %d) => %s\n",
relayNum, pin, state ? "ON" : "OFF");
          }

          }
        } else {
          Serial.println("Gagal parsing JSON array dari server!");
        }
      } else {
        Serial.printf("Gagal GET status relay. Kode: %d\n",
httpCode);
      }

      http.end();
    }

    vTaskDelay(1000 / portTICK_PERIOD_MS);  // Cek status setiap 5
detik
  }
}

void checkWiFi() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("[WiFi]     Terputus!     Mencoba     menyambung
kembali...");
    WiFi.disconnect();
    WiFi.begin(ssid, password);

    int retry = 0;
    while (WiFi.status() != WL_CONNECTED && retry < 20) {
      delay(500);
      Serial.print(".");
      retry++;
    }

    if (WiFi.status() == WL_CONNECTED) {
      Serial.println("\n[WiFi]   Tersambung   ulang!   IP:   "   +
WiFi.localIP().toString());
    } else {
      Serial.println("\n[WiFi] Gagal reconnect setelah 10 detik.");
    }
  }
}
```

LAMPIRAN D LAPORAN PENGGUNAAN ENERGI (PDF)

Lampiran ini merupakan cuplikan dari laporan PDF penggunaan energi yang dihasilkan secara otomatis oleh sistem.

Lampiran D1 – BAB 1 Halaman 1 Laporan Penggunaan Energi

Wallet konumer 1 “0x9A43b29497aebF8691E8Fa36760d7F57222b86e4”



Laporan Penggunaan Energi

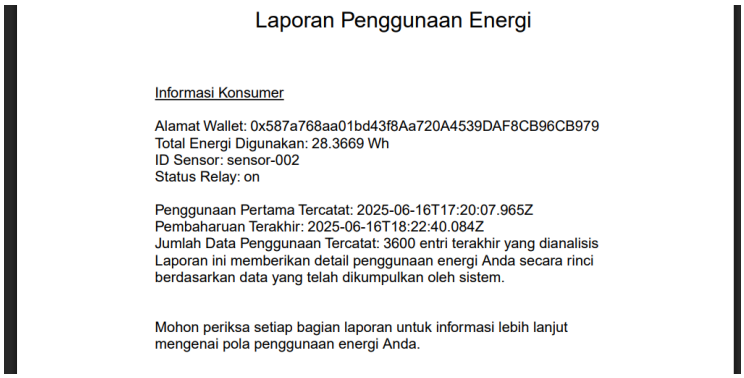
Informasi Konsumer

Alamat Wallet: 0x9A43b29497aebF8691E8Fa36760d7F57222b86e4
Total Energi Digunakan: 48.4542 Wh
ID Sensor: sensor-001
Status Relay: on

Penggunaan Pertama Tercatat: 2025-06-16T17:20:07.919Z
Pembaharuan Terakhir: 2025-06-16T18:22:40.017Z
Jumlah Data Penggunaan Tercatat: 3600 entri terakhir yang dianalisis
Laporan ini memberikan detail penggunaan energi Anda secara rinci berdasarkan data yang telah dikumpulkan oleh sistem.

Mohon periksa setiap bagian laporan untuk informasi lebih lanjut mengenai pola penggunaan energi Anda.

Wallet konumer 2 “0x587a768aa01bd43f8Aa720A4539DAF8CB96CB979”



Laporan Penggunaan Energi

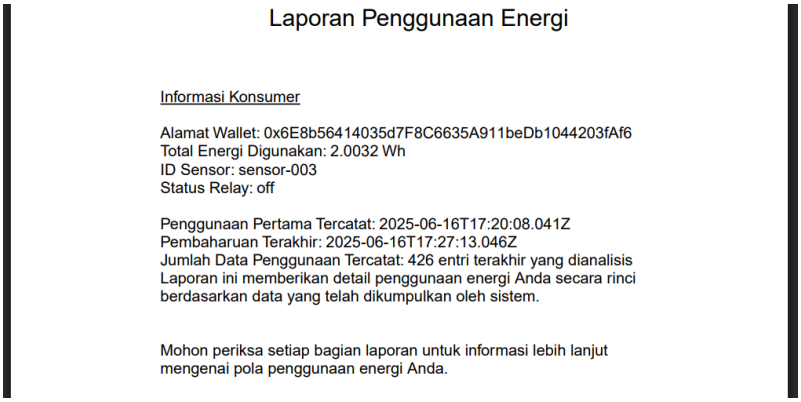
Informasi Konsumer

Alamat Wallet: 0x587a768aa01bd43f8Aa720A4539DAF8CB96CB979
Total Energi Digunakan: 28.3669 Wh
ID Sensor: sensor-002
Status Relay: on

Penggunaan Pertama Tercatat: 2025-06-16T17:20:07.965Z
Pembaharuan Terakhir: 2025-06-16T18:22:40.084Z
Jumlah Data Penggunaan Tercatat: 3600 entri terakhir yang dianalisis
Laporan ini memberikan detail penggunaan energi Anda secara rinci berdasarkan data yang telah dikumpulkan oleh sistem.

Mohon periksa setiap bagian laporan untuk informasi lebih lanjut mengenai pola penggunaan energi Anda.

Wallet konumer 3 “0x6E8b56414035d7F8C6635A911beDb1044203fAf6”



Laporan Penggunaan Energi

Informasi Konsumer

Alamat Wallet: 0x6E8b56414035d7F8C6635A911beDb1044203fAf6
Total Energi Digunakan: 2.0032 Wh
ID Sensor: sensor-003
Status Relay: off

Penggunaan Pertama Tercatat: 2025-06-16T17:20:08.041Z
Pembaharuan Terakhir: 2025-06-16T17:27:13.046Z
Jumlah Data Penggunaan Tercatat: 426 entri terakhir yang dianalisis
Laporan ini memberikan detail penggunaan energi Anda secara rinci berdasarkan data yang telah dikumpulkan oleh sistem.

Mohon periksa setiap bagian laporan untuk informasi lebih lanjut mengenai pola penggunaan energi Anda.

Lampiran D2 –BAB 2 Halaman 2 Detail Penggunaan Energi

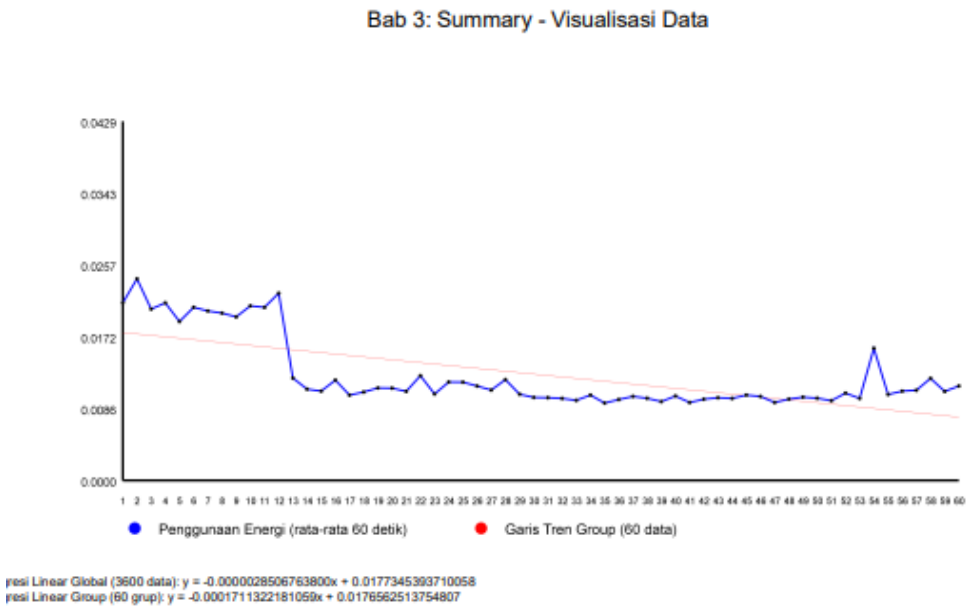
Untuk Lampiran D2 – BAB 2 berisi value atau nilai dari sensor pada masing-masing dompet digital. Dalam lampiran ini hanya ditampilkan salah satu contoh log sensor karena seluruh log memiliki format yang serupa sebagai hasil dari proses pencatatan (logging) sensor secara berkala.

Bab 2: Detail Penggunaan Energi - Halaman 2

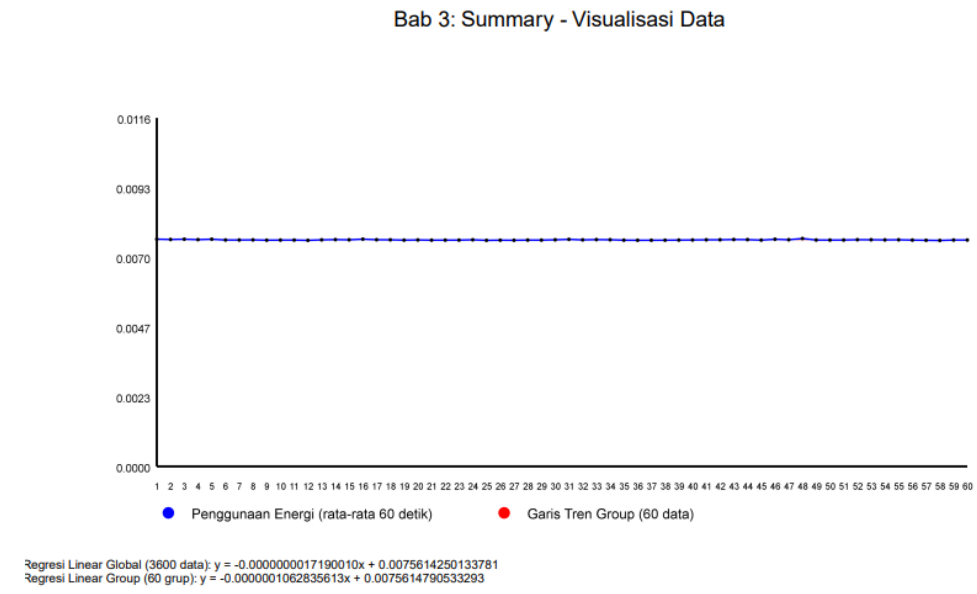
No.	Timestamp	Energy Used (Wh)
1	2025-06-16T17:20:07.919Z	0.0000
2	2025-06-16T17:20:08.892Z	0.0219
3	2025-06-16T17:20:09.893Z	0.0208
4	2025-06-16T17:20:10.895Z	0.0243
5	2025-06-16T17:20:11.893Z	0.0242
6	2025-06-16T17:20:12.893Z	0.0203
7	2025-06-16T17:20:13.900Z	0.0215
8	2025-06-16T17:20:14.893Z	0.0219
9	2025-06-16T17:20:15.899Z	0.0222
10	2025-06-16T17:20:16.896Z	0.0191
11	2025-06-16T17:20:17.891Z	0.0218
12	2025-06-16T17:20:18.898Z	0.0207
13	2025-06-16T17:20:19.898Z	0.0198
14	2025-06-16T17:20:20.896Z	0.0202
15	2025-06-16T17:20:21.905Z	0.0205
16	2025-06-16T17:20:22.896Z	0.0227
17	2025-06-16T17:20:23.895Z	0.0210
18	2025-06-16T17:20:24.897Z	0.0210
19	2025-06-16T17:20:25.898Z	0.0210
20	2025-06-16T17:20:26.895Z	0.0204
21	2025-06-16T17:20:27.897Z	0.0215
22	2025-06-16T17:20:28.895Z	0.0201
23	2025-06-16T17:20:29.891Z	0.0201
24	2025-06-16T17:20:30.892Z	0.0207
25	2025-06-16T17:20:31.898Z	0.0204
26	2025-06-16T17:20:32.893Z	0.0209
27	2025-06-16T17:20:33.895Z	0.0196
28	2025-06-16T17:20:34.896Z	0.0196
29	2025-06-16T17:20:35.894Z	0.0219
30	2025-06-16T17:20:36.897Z	0.0210
31	2025-06-16T17:20:37.897Z	0.0219
32	2025-06-16T17:20:38.893Z	0.0219
33	2025-06-16T17:20:39.893Z	0.0209
34	2025-06-16T17:20:40.898Z	0.0200
35	2025-06-16T17:20:41.893Z	0.0199
36	2025-06-16T17:20:42.893Z	0.0204
37	2025-06-16T17:20:43.897Z	0.0205
38	2025-06-16T17:20:44.902Z	0.0245
39	2025-06-16T17:20:45.894Z	0.0241
40	2025-06-16T17:20:46.897Z	0.0244
41	2025-06-16T17:20:47.893Z	0.0229
42	2025-06-16T17:20:48.893Z	0.0230
43	2025-06-16T17:20:49.897Z	0.0234
44	2025-06-16T17:20:50.898Z	0.0233
45	2025-06-16T17:20:51.893Z	0.0245
46	2025-06-16T17:20:52.895Z	0.0247
47	2025-06-16T17:20:53.897Z	0.0246
48	2025-06-16T17:20:54.893Z	0.0235
49	2025-06-16T17:20:55.897Z	0.0219
50	2025-06-16T17:20:56.897Z	0.0201
51	2025-06-16T17:20:57.892Z	0.0200
52	2025-06-16T17:20:58.897Z	0.0202
53	2025-06-16T17:20:59.897Z	0.0195
54	2025-06-16T17:21:00.895Z	0.0197
55	2025-06-16T17:21:01.897Z	0.0198
56	2025-06-16T17:21:02.898Z	0.0210
57	2025-06-16T17:21:03.895Z	0.0208
58	2025-06-16T17:21:04.898Z	0.0207
59	2025-06-16T17:21:05.897Z	0.0192
60	2025-06-16T17:21:06.893Z	0.0191

Lampiran D3 –BAB 3 Visualisasi Data

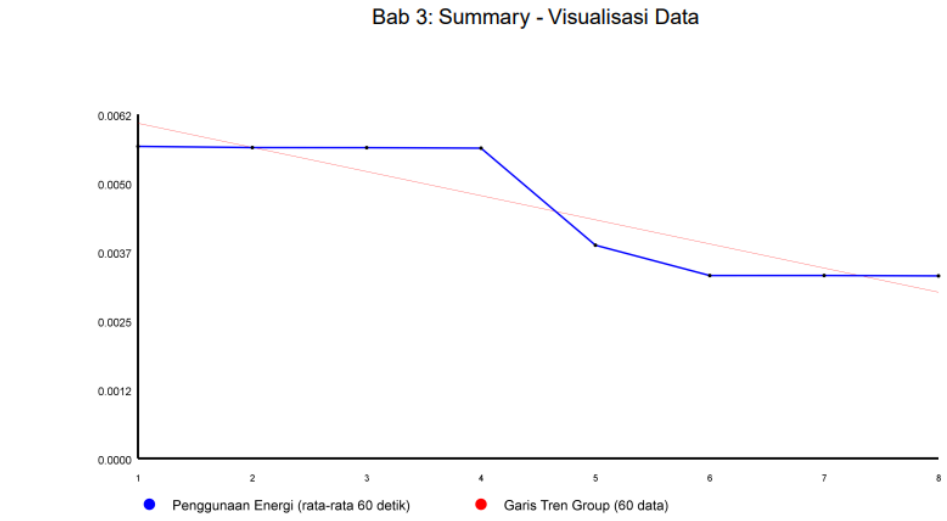
Wallet konumer 1 “0x9A43b29497aebF8691E8Fa36760d7F57222b86e4”



Wallet konumer 2 “0x587a768aa01bd43f8Aa720A4539DAF8CB96CB979”



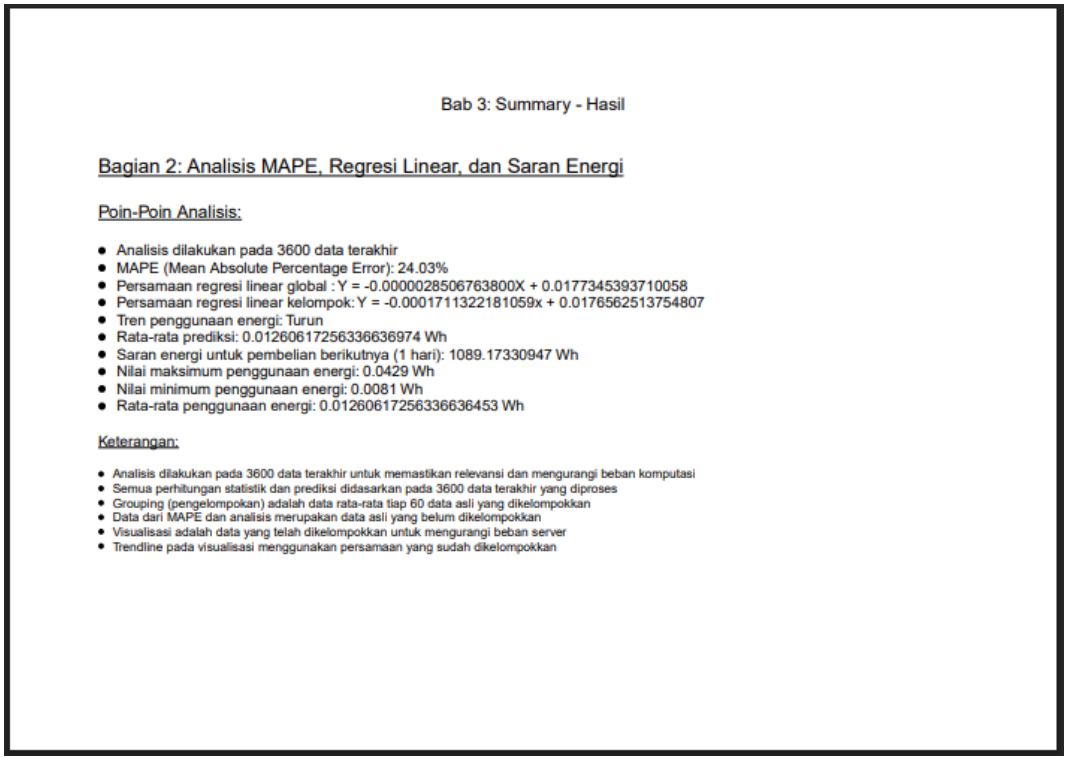
Wallet konumer 3 “0x6E8b56414035d7F8C6635A911beDb1044203fAf6”



Regresi Linear Global (3600 data): y = -0.0000081137052553x + 0.0064334154487710
Regresi Linear Group (60 grup): y = -0.0004376035659171x + 0.0060771863811728

Lampiran D4 – BAB 3 Hasil

Wallet konumer 1 “0x9A43b29497aebF8691E8Fa36760d7F57222b86e4”



Bab 3: Summary - Hasil

Bagian 2: Analisis MAPE, Regresi Linear, dan Saran Energi

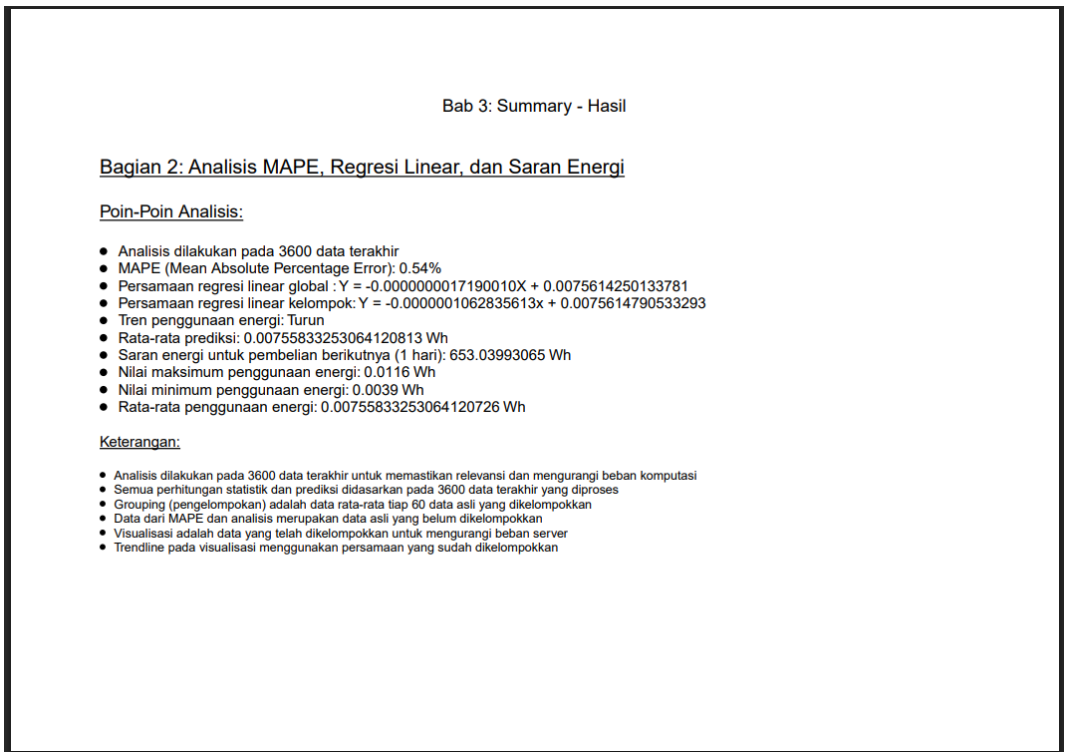
Poin-Poin Analisis:

- Analisis dilakukan pada 3600 data terakhir
- MAPE (Mean Absolute Percentage Error): 24.03%
- Persamaan regresi linear global : Y = -0.0000028506763800X + 0.0177345393710058
- Persamaan regresi linear kelompok: Y = -0.000171322181059x + 0.0176562513754807
- Tren penggunaan energi: Turun
- Rata-rata prediksi: 0.012606172563366369 74 Wh
- Saran energi untuk pembelian berikutnya (1 hari): 1089.17330947 Wh
- Nilai maksimum penggunaan energi: 0.0429 Wh
- Nilai minimum penggunaan energi: 0.0081 Wh
- Rata-rata penggunaan energi: 0.012606172563366364 53 Wh

Keterangan:

- Analisis dilakukan pada 3600 data terakhir untuk memastikan relevansi dan mengurangi beban komputasi
- Semua perhitungan statistik dan prediksi didasarkan pada 3600 data terakhir yang diproses
- Grouping (pengelompokan) adalah data rata-rata tiap 60 data asli yang dikelompokkan
- Data dari MAPE dan analisis merupakan data asli yang belum dikelompokkan
- Visualisasi adalah data yang telah dikelompokkan untuk mengurangi beban server
- Trendline pada visualisasi menggunakan persamaan yang sudah dikelompokkan

Wallet konumer 2 “0x587a768aa01bd43f8Aa720A4539DAF8CB96CB979”



Bab 3: Summary - Hasil

Bagian 2: Analisis MAPE, Regresi Linear, dan Saran Energi

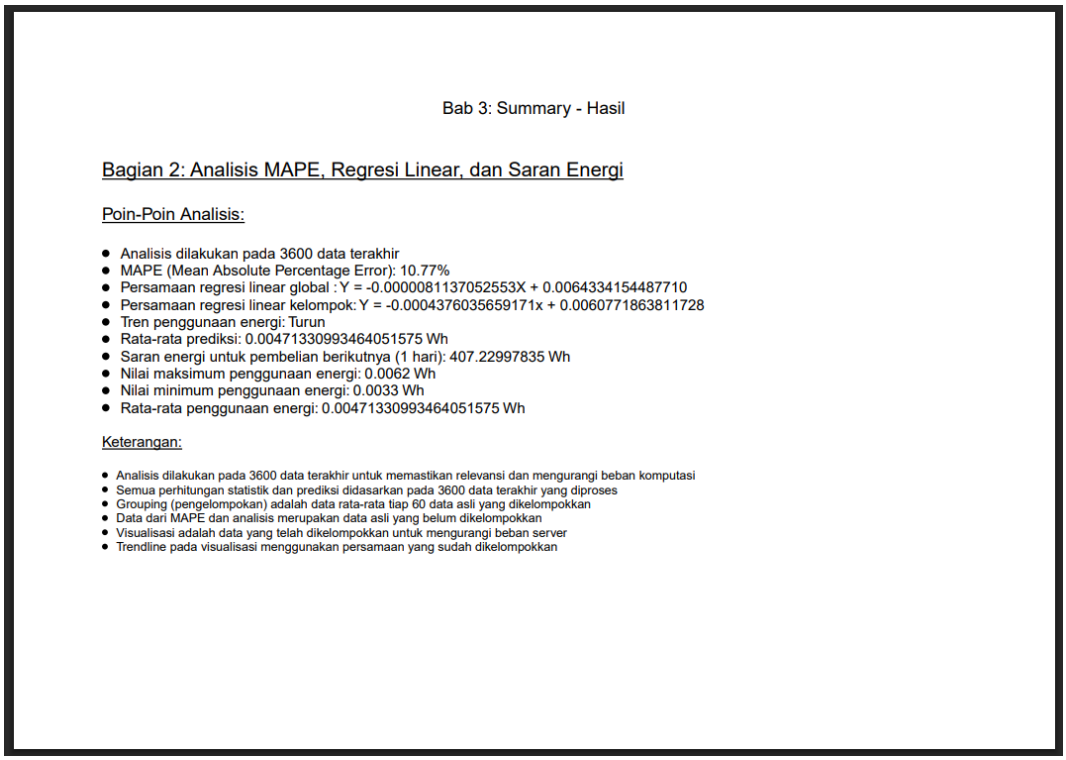
Poin-Poin Analisis:

- Analisis dilakukan pada 3600 data terakhir
- MAPE (Mean Absolute Percentage Error): 0.54%
- Persamaan regresi linear global : Y = -0.0000000017190010X + 0.0075614250133781
- Persamaan regresi linear kelompok: Y = -0.0000001062835613x + 0.0075614790533293
- Tren penggunaan energi: Turun
- Rata-rata prediksi: 0.007558332530641208 13 Wh
- Saran energi untuk pembelian berikutnya (1 hari): 653.03993065 Wh
- Nilai maksimum penggunaan energi: 0.0116 Wh
- Nilai minimum penggunaan energi: 0.0039 Wh
- Rata-rata penggunaan energi: 0.007558332530641207 26 Wh

Keterangan:

- Analisis dilakukan pada 3600 data terakhir untuk memastikan relevansi dan mengurangi beban komputasi
- Semua perhitungan statistik dan prediksi didasarkan pada 3600 data terakhir yang diproses
- Grouping (pengelompokan) adalah data rata-rata tiap 60 data asli yang dikelompokkan
- Data dari MAPE dan analisis merupakan data asli yang belum dikelompokkan
- Visualisasi adalah data yang telah dikelompokkan untuk mengurangi beban server
- Trendline pada visualisasi menggunakan persamaan yang sudah dikelompokkan

Wallet konumer 3 “0x6E8b56414035d7F8C6635A911beDb1044203fAf6”



Bab 3: Summary - Hasil

Bagian 2: Analisis MAPE, Regresi Linear, dan Saran Energi

Poin-Poin Analisis:

- Analisis dilakukan pada 3600 data terakhir
- MAPE (Mean Absolute Percentage Error): 10.77%
- Persamaan regresi linear global : Y = -0.000008113705255 3X + 0.0064334154487710
- Persamaan regresi linear kelompok: Y = -0.0004376035659171x + 0.0060771863811728
- Tren penggunaan energi: Turun
- Rata-rata prediksi: 0.00471330993464051575 Wh
- Saran energi untuk pembelian berikutnya (1 hari): 407.22997835 Wh
- Nilai maksimum penggunaan energi: 0.0062 Wh
- Nilai minimum penggunaan energi: 0.0033 Wh
- Rata-rata penggunaan energi: 0.00471330993464051575 Wh

Keterangan:

- Analisis dilakukan pada 3600 data terakhir untuk memastikan relevansi dan mengurangi beban komputasi
- Semua perhitungan statistik dan prediksi didasarkan pada 3600 data terakhir yang diproses
- Grouping (pengelompokan) adalah data rata-rata tiap 60 data asli yang dikelompokkan
- Data dari MAPE dan analisis merupakan data asli yang belum dikelompokkan
- Visualisasi adalah data yang telah dikelompokkan untuk mengurangi beban server
- Trendline pada visualisasi menggunakan persamaan yang sudah dikelompokkan

Lampiran D4 –BAB 3 Rata-rata Tiap Grup (60 data per grup)

Wallet konumer 1 “0x9A43b29497aebF8691E8Fa36760d7F57222b86e4”

Bab 3: Summary - Rata-rata Tiap Grup (60 data per grup)

- Grup 1: 0.02121 Wh
- Grup 2: 0.02407 Wh
- Grup 3: 0.02046 Wh
- Grup 4: 0.02120 Wh
- Grup 5: 0.01897 Wh
- Grup 6: 0.02067 Wh
- Grup 7: 0.02024 Wh
- Grup 8: 0.01998 Wh
- Grup 9: 0.01953 Wh
- Grup 10: 0.02084 Wh
- Grup 11: 0.02066 Wh
- Grup 12: 0.02235 Wh
- Grup 13: 0.01222 Wh
- Grup 14: 0.01090 Wh
- Grup 15: 0.01066 Wh
- Grup 16: 0.01197 Wh
- Grup 17: 0.01019 Wh
- Grup 18: 0.01055 Wh
- Grup 19: 0.01104 Wh
- Grup 20: 0.01101 Wh
- Grup 21: 0.01063 Wh
- Grup 22: 0.01250 Wh
- Grup 23: 0.01032 Wh
- Grup 24: 0.01174 Wh
- Grup 25: 0.01173 Wh
- Grup 26: 0.01127 Wh
- Grup 27: 0.01079 Wh
- Grup 28: 0.01203 Wh
- Grup 29: 0.01029 Wh
- Grup 30: 0.00991 Wh
- Grup 31: 0.00988 Wh
- Grup 32: 0.00979 Wh
- Grup 33: 0.00956 Wh
- Grup 34: 0.01019 Wh
- Grup 35: 0.00923 Wh
- Grup 36: 0.00965 Wh
- Grup 37: 0.01003 Wh
- Grup 38: 0.00981 Wh
- Grup 39: 0.00941 Wh
- Grup 40: 0.01008 Wh
- Grup 41: 0.00930 Wh
- Grup 42: 0.00969 Wh
- Grup 43: 0.00987 Wh
- Grup 44: 0.00979 Wh
- Grup 45: 0.01019 Wh
- Grup 46: 0.01003 Wh
- Grup 47: 0.00931 Wh
- Grup 48: 0.00970 Wh
- Grup 49: 0.00994 Wh
- Grup 50: 0.00982 Wh
- Grup 51: 0.00952 Wh
- Grup 52: 0.01042 Wh
- Grup 53: 0.00979 Wh
- Grup 54: 0.01578 Wh
- Grup 55: 0.01026 Wh
- Grup 56: 0.01066 Wh
- Grup 57: 0.01076 Wh
- Grup 58: 0.01219 Wh
- Grup 59: 0.01063 Wh

Wallet konumer 2 “0x587a768aa01bd43f8Aa720A4539DAF8CB96CB979”

Bab 3: Summary - Rata-rata Tiap Grup (60 data per grup)

- Grup 1: 0.00758 Wh
- Grup 2: 0.00757 Wh
- Grup 3: 0.00758 Wh
- Grup 4: 0.00757 Wh
- Grup 5: 0.00758 Wh
- Grup 6: 0.00755 Wh
- Grup 7: 0.00755 Wh
- Grup 8: 0.00756 Wh
- Grup 9: 0.00755 Wh
- Grup 10: 0.00755 Wh
- Grup 11: 0.00755 Wh
- Grup 12: 0.00754 Wh
- Grup 13: 0.00756 Wh
- Grup 14: 0.00757 Wh
- Grup 15: 0.00756 Wh
- Grup 16: 0.00758 Wh
- Grup 17: 0.00756 Wh
- Grup 18: 0.00756 Wh
- Grup 19: 0.00755 Wh
- Grup 20: 0.00756 Wh
- Grup 21: 0.00755 Wh
- Grup 22: 0.00755 Wh
- Grup 23: 0.00755 Wh
- Grup 24: 0.00756 Wh
- Grup 25: 0.00754 Wh
- Grup 26: 0.00755 Wh
- Grup 27: 0.00755 Wh
- Grup 28: 0.00755 Wh
- Grup 29: 0.00755 Wh
- Grup 30: 0.00756 Wh
- Grup 31: 0.00758 Wh
- Grup 32: 0.00756 Wh
- Grup 33: 0.00757 Wh
- Grup 34: 0.00756 Wh
- Grup 35: 0.00755 Wh
- Grup 36: 0.00754 Wh
- Grup 37: 0.00755 Wh
- Grup 38: 0.00755 Wh
- Grup 39: 0.00755 Wh
- Grup 40: 0.00756 Wh
- Grup 41: 0.00756 Wh
- Grup 42: 0.00756 Wh
- Grup 43: 0.00757 Wh
- Grup 44: 0.00757 Wh
- Grup 45: 0.00755 Wh
- Grup 46: 0.00758 Wh
- Grup 47: 0.00756 Wh
- Grup 48: 0.00760 Wh
- Grup 49: 0.00755 Wh
- Grup 50: 0.00755 Wh
- Grup 51: 0.00755 Wh
- Grup 52: 0.00756 Wh
- Grup 53: 0.00756 Wh
- Grup 54: 0.00756 Wh
- Grup 55: 0.00756 Wh
- Grup 56: 0.00755 Wh
- Grup 57: 0.00754 Wh
- Grup 58: 0.00754 Wh
- Grup 59: 0.00755 Wh
- Grup 60: 0.00755 Wh

Wallet konumer 3 “0x6E8b56414035d7F8C6635A911beDb1044203fAf6”

Bab 3: Summary - Rata-rata Tiap Grup (60 data per grup)

- Grup 1: 0.00566 Wh
- Grup 2: 0.00564 Wh
- Grup 3: 0.00564 Wh
- Grup 4: 0.00563 Wh
- Grup 5: 0.00387 Wh
- Grup 6: 0.00332 Wh
- Grup 7: 0.00332 Wh
- Grup 8: 0.00331 Wh

LAMPIRAN BIODATA

BIODATA MAHASISWA

Nama	: Ach. Arif Zuchal Ulya
NRP	: 0921040023
Program Studi	: D4 Teknik Otomasi
Agama	: Islam
Status	: Belum Menikah
Alamat Asal	: Perumahan Canda Bhirawa T-15, Katang, Ngasem, Kabupaten Kediri
Nomor Telepon	: 082338422877
Jenis Kelamin	: Laki-laki
Email	: zuchal9k@gmail.com
Tempat, Tanggal Lahir	: Kediri, 18 September 2002
Nama Orang Tua/ Wali	: Salam Supriyanto
Alamat Orang Tua / Wali	: Perumahan Canda Bhirawa T-15, Katang, Ngasem, Kabupaten Kediri
Telepon Orang Tua / Wali	: 082334938694
Riwayat Pendidikan	

PENDIDIKAN FORMAL			
Pendidikan	Tahun	Tempat Pendidikan	Jurusan
Diploma 4	2021-Sekarang	Politeknik Perkapalan Negeri Surabaya	Teknik Otomasi
SMA	2017-2020	SMAN 2 Pare	MIPA
SMP	2014-2017	MTsN 2 Kota Kediri	-
SD	2008-2014	SDI Al-huda	-

"Halaman Sengaja Dikosongkan"