BAB IV
PERANCANGAN

Bab ini menyajikan uraian komprehensif mengenai seluruh tahapan perancangan dan implementasi sistem energy meter yang secara khusus dikembangkan untuk diaplikasikan pada *e-cargo bike*. Proses yang dibahas bermula dari analisis mendalam terhadap sistem-sistem terdahulu guna mengidentifikasi kelemahan serta celah penelitian yang ada. Berdasarkan analisis tersebut, dirumuskan serangkaian spesifikasi kebutuhan sistem baru yang fungsional dan teknis. Meskipun perancangan ini melibatkan pemilihan dan integrasi komponen perangkat keras, kontribusi utama dari sisi keilmuan Informatika terletak pada pengembangan algoritma integrasi numerik dan perancangan firmware sistem tertanam yang mampu mengolah data sensor menjadi informasi energi yang akurat dan *real-time*. Sebagai penutup, bab ini mendokumentasikan proses implementasi atau realisasi dari rancangan konseptual tersebut, yang diwujudkan dalam bentuk prototipe fungsional yang sepenuhnya siap untuk melalui tahapan pengujian.

4.1. Analisis Sistem Terdahulu

Sebuah analisis mendalam terhadap lanskap teknologi *energy meter* yang tersedia saat ini mengungkapkan bahwa mayoritas perangkat yang beredar di pasar komersial belum dirancang untuk mengakomodasi kebutuhan unik dan spesifik dari kendaraan listrik, khususnya *e-cargo bike*. Perangkat-perangkat tersebut umumnya ditujukan untuk aplikasi pada lingkungan dengan beban listrik yang cenderung stabil dan dapat diprediksi, seperti pada instalasi rumah tangga (umumnya 220V AC) atau sebagai bagian dari infrastruktur *smart grid* (Rind et al., 2023). Tantangan utama dari pengaplikasian pada kendaraan listrik adalah karakteristik bebannya yang dinamis. Penelitian oleh (Bartolomei et al., 2020) menyoroti bahwa pengujian akurasi *energy meter* standar menjadi sangat unik ketika dihadapkan pada tegangan dan arus yang terdistorsi secara realistis, yang merupakan kondisi umum pada operasional kendaraan listrik. Akibatnya, penggunaan *energy meter* konvensional yang mengandalkan metode perhitungan rata-rata sederhana berisiko menghasilkan

tingkat presisi yang tidak memadai untuk tujuan manajemen daya yang efektif dan akurat.

Seiring dengan kemajuan teknologi sistem tertanam dan *Internt of Things* (IoT), berbagai penelitian akademis telah berfokus pada pengembangan *smart energy meter* berbasis mikrokontroler. Sebuah tren yang signifikan menunjukkan penggunaan mikrokontroler berbiaya rendah seperti *ESP32* untuk sistem monitoring energi yang terhubung dengan internet, seperti yang didemonstrasikan oleh (El-Khozondar et al., 2024) dan (Irsyadi et al., 2024). Namun, fokus utama dari riset-riset ini sering kali adalah pada manajemen energi untuk rumah pintar atau instalasi statis (Saleem et al., 2022). Oleh karena itu, solusi-solusi tersebut belum secara spesifik menyasar rentang pengukuran tegangan tinggi 50—100V dan 0—30A yang menjadi karakteristik utama sistem kelistrikan *e-cargo bike.*

Tantangan teknis yang lebih dalam terletak pada metode perhitungan yang digunakan untuk mengestimasi konsumsi energi dan status baterai. Banyak sistem monitoring baterai bergantung pada metode *Coulomb Counting*. Meskipun populer, pendekatan ini memiliki kelemahan kritis. Seperti yang dianalisis secara mendalam oleh (Movassagh et al., 2021), metode ini rentan terhadap akumulasi galat (*cumulative error*) dari waktu ke waktu, terutama akibat ketidakakuratan sensor dan kondisi operasional yang dinamis, yang pada akhirnya dapat menyebabkan estimasi *State of Charge* (*SoC*) yang tidak andal. Hal ini menggarisbawahi pentingnya memilih atau mengembangkan metode perhitungan yang lebih kuat. Selain itu, riset juga telah berkembang ke arah sistem monitoring yang terintegrasi langsung dengan manajemen baterai (*BMS*) pada robot bergerak (Chellal et al., 2021), yang menunjukkan betapa krusialnya akurasi data energi untuk keamanan dan optimisasi umur baterai.

Berdasarkan tinjauan di atas, dapat disimpulkan adanya sebuah kesenjangan riset yang jelas. Meskipun penelitian mengenai smart meter berbasis *IoT* sangat ekstensif dan telah ada beberapa pengembangan untuk monitoring *e-bike* (Ilahi et al., 2024; Li et al., 2024), masih terdapat kekurangan solusi yang menghadirkan modul *energy meter* yang bersifat mandiri (*standalone*), *non-IoT*, dengan fokus utama pada akurasi perhitungan energi melalui integrasi numerik untuk aplikasi spesifik *e-cargo bike* pada rentang tegangan dan arus 50—100V/0—30A.

Penelitian ini bertujuan untuk mengisi kekosongan dengan mengembangkan sebuah perangkat yang andal, akurat, dan dirancang khusus untuk menghadapi tantangan beban dinamis pada kendaraan listrik ringan.

4.2. Spesifikasi Kebutuhan Sistem Baru

Berdasarkan analisis permasalahan dan tujuan penelitian yang telah ditetapkan, sistem energy meter yang akan dirancang dan dikembangkan harus memenuhi serangkaian spesifikasi kebutuhan fungsional dan teknis yang terperinci sebagai berikut:

1. Kebutuhan Fungsional

- **Pengukuran Tegangan:** Sistem harus memiliki kemampuan untuk melakukan pengukuran tegangan operasional baterai secara akurat pada rentang kerja 50-100V. Rentang ini ditentukan oleh karakteristik baterai *Li-ion* yang digunakan, mencakup tegangan saat terisi penuh (fully charged), tegangan nominal, hingga tegangan batas bawah (*cut-off voltage*).
- **Pengukuran Arus:** Sistem harus mampu mengukur aliran arus listrik, baik saat pengosongan (*discharging*) maupun pengisian (*charging*), dalam rentang dinamis 0-30A. Rentang ini didasarkan pada karakteristik tarikan arus kontinyu dan puncak dari motor listrik yang digunakan pada *e-cargo bike*.
- **Akumulasi Energi (Wh):** Sistem harus dapat menghitung dan mengakumulasi total energi yang telah dikonsumsi oleh sistem secara kumulatif, yang akan ditampilkan dalam satuan *Watt-hour* (Wh). Ini adalah parameter kunci untuk analisis efisiensi.
- **Akumulasi Kapasitas (Ah):** Sistem harus mampu menghitung dan mengakumulasi total kapasitas baterai yang telah digunakan, yang direpresentasikan dalam satuan *Ampere-hour* (Ah). Parameter ini vital untuk estimasi sisa kapasitas baterai.
- **Tampilan Real-Time:** Sistem wajib menyajikan semua parameter energi esensial—mencakup tegangan (V), arus (A), konsumsi energi (Wh), kapasitas baterai (Ah), dan durasi waktu operasi (h)—secara *real-time* kepada pengguna melalui antarmuka visual berupa layar *OLED*.

2. **Kebutuhan Perangkat Keras**

- ***Mikrokontroler*:** Papan pengembangan Arduino Nano dipilih sebagai unit pemrosesan sentral. Pemilihan ini didasarka pada pertimbangan ukurannya yang sangat ringkas, konsumsi daya yang efisien, serta ketersediaan pin I/O yang memadai untuk kebutuhan proyek sistem tertanam ini.
- **Sensor Arus:** Modul sensor ACS712 digunakan secara spesifik untuk melakukan pengukuran arus listrik. Sensor ini bekerja berdasarkan efek Hall yang memungkinkan pengukuran arus secara non-kontak dan akurat.
- **Sensor Tegangan:** Untuk mengukur teganga baterai yang tinggi, digunakan sebuah rangkaian pembagi tegangan (*votlage divider*) yang dirancang khusus menggunakan kombinasi resistor presisi 33kΩ dan 2.2kΩ untuk menurunkan level tegangan ke rentang yang aman bagi pin ADC *mikrokontroler*.
- **Tampilan:** Sebuah layar *OLED* berukuran 0.96 inci diintegrasikan sebagai antarmuka *display*. Pilihan ini didasarkan pada keunggulannya dalam hal konsumsi daya yang rendah, kontras tinggi, dan keterbacaan yang baik, bahkan di bawah cahaya lingkungan sekitar.

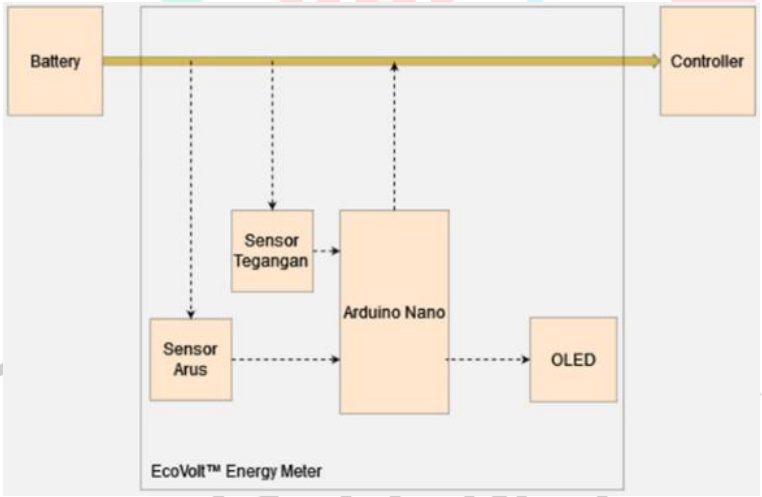
3. **Kebutuhan Perangkat Lunak**

- **Platform dan Bahasa:** *Firmware* atau perangkat lunak sistem dikembangkan menggunakan Bahasa pemrograman *C++* yang diimplementasikan pada *platform Arduino IDE*, yang merupakan standar industry untuk prototipe berbasis *Arduino.*
- **Algoritma Inti:** Jantung dari perangkat lunak ini adalah algoritma yang mengimplementasikan metode integrasi numerik. Algoritma in ibertanggung jawab untuk melakukan kalkulasi akumulatif terhadap nilai energi (Wh) dan kapasitas (Ah) dari data daya sesaat.
- ***Sampling Rate*:** Untuk memastikan sistem dapat merespons perubahan beban dinamis secara efektif, interval waktu pembacaan sensor (*sampling rate*) ditetapkan pada 100 milidetik. Nilai ini dipilih sebagai sebuah kompromi optimal antara akurasi penangkapan fluktuasi data dan beban komputasi pada prosesor *ATmega328.*

4.3. Perancangan Sistem

4.3.1. Perancangan Perangkat Keras

Pemilihan komponen perangkat keras dilakukan dengan tujuan utama untuk membangun sebuah platform akuisisi data yang andal bagi perangkat lunak yang akan dijalankan. Setiap komponen dipilih bukan hanya berdasarkan spesifikasi elektrikalnya, tetapi juga berdasarkan kemampuannya untuk menyediakan data yang stabil dan dapat diolah oleh algoritma yang dirancang. Arduino Nano dipilih sebagai unit pemroses sentral bukan hanya karena ukurannya yang ringkas, tetapi karena kapabilitas mikrokontroler *ATmega328*-nya yang terbukti andal dalam mengeksekusi algoritma pemrosesan sinyal secara *real-time*. Demikian pula, sensor *ACS712* dipilih karena kemampuannya menyediakan data arus dalam bentuk sinyal analog yang stabil dan mudah diolah oleh *Analog-to-Digital Converter* (*ADC*) mikrokontroler. Arsitektur fungsional dan aliran sinyal dari sistem ini diilustrasikan secara konseptual melalui diagram blok pada Gambar 4.1.

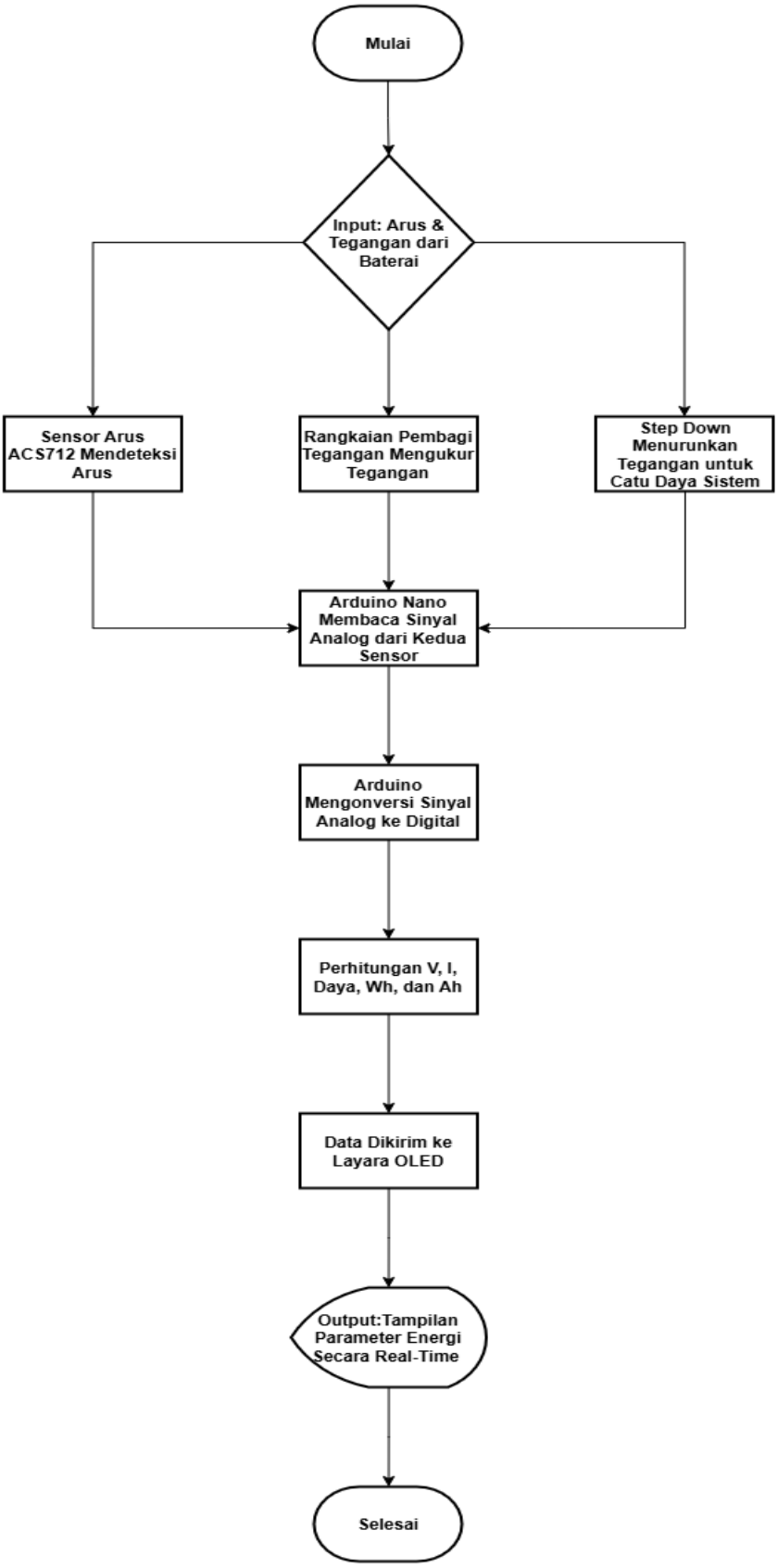


Gambar 4.1. Diagram Bok *Energy Meter*

Alur kerja perangkat keras secara detail adalah sebagai berikut: Sensor arus *ACS712* dipasang pada jalur daya positif untuk mendeteksi besarnya arus yang mengalir, sementara sensor tegangan dihubungkan untuk membaca level tegangan baterai. Rangkaian pembagi tegangan ini memiliki fungsi krusial untuk

mengonversi level tegangan tinggi dari baterai (50-100V) menjadi sinyal analog bertegangan rendah (0-5V) yang aman dan kompatibel dengan rentang input pin *Analog-to-Digital Converter* (*ADC*) pada mikrokontroler. Sinyal analog dari kedua sensor ini kemudian diproses oleh *Arduino Nano*. Di dalam mikrokontroler, sinyal digital hasil konversi diolah lebih lanjut untuk menghitung parameter energi. Hasil akhir dari kalkulasi tersebut kemudian dikirimkan melalui protokol *I2C* ke Layar *OLED* untuk ditampilkan.

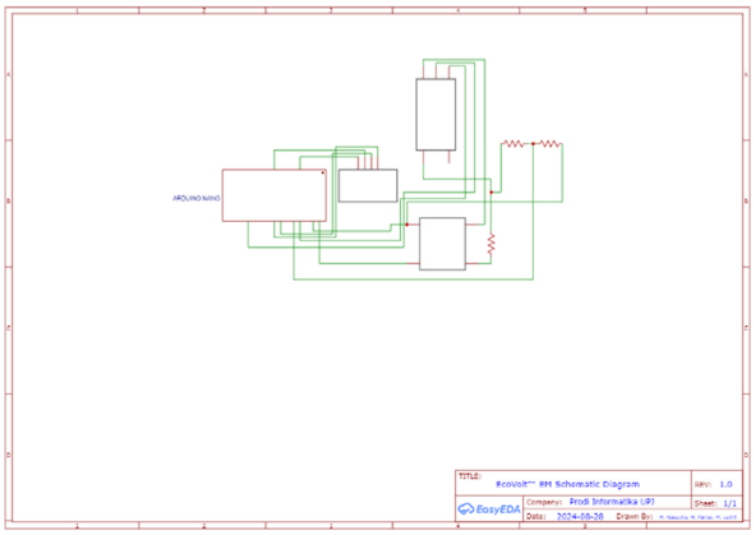
Untuk memperjelas bagaimana komponen-komponen tersebut berinteraksi dan bagaimana data diproses, Gambar 4.2 menunjukkan *flowchart* yang menggambarkan alur kerja sistem.



Gambar 4.2. Flowchart Alur Kerja Sistem

Berdasarkan *flowchart* di atas, alur kerja sistem dimulai dengan pembacaan sinyal analog dari sensor arus *ACS712* dan rangkaian pembagi tegangan. Sinyal ini kemudian dibaca dan diolah oleh *Arduino Nano*, yang mengonversinya menjadi data digital. Selanjutnya, *mikrokontroler* melakukan kalkulasi parameter energi sebelum akhirnya mengirimkan hasilnya untuk ditampilkan secara *real-time* pada layar *OLED*. Alur kerja inilah yang menjadi dasar utama dalam perancangan perangkat lunak (*firmware*) sistem.

Seluruh konektivitas antar komponen, termasuk pinout dan jalur catu daya, diimplementasikan secara detail dalam sebuah rangkaian elektronik seperti yang diperlihatkan pada diagram skematik di Gambar 4.3.



Gambar 4.3. Diagram Skematik *Energy Meter*

4.3.2. Perancangan Perangkat Lunak

Perancangan perangkat lunak atau *firmware* bertujuan untuk menciptakan sebuah sistem operasi *real-time* yang mampu menjalankan fungsi pengukuran, perhitungan, dan penampilan data secara kontinu dan efisien. Inti dari perancangan perangkat lunak terletak pada implementasi algoritma integrasi numerik, yang berfungsi untuk mengakumulasi nilai energi (Wh) dan kapasitas (Ah) secara akurat dari data daya sesaat yang diukur.

1. **Fungsi Inisialisasi (setup()):** Fungsi in dieksekusi sekali saat perangkat dinyalakan. Tugasnya meliputi inisialisasi komunikasi serial untuk keperluan *debugging*, mengaktifkan dan membersihkan layar *OLED*, menampilkan logo pembuka, serta mengatur semua variable kumulatif (seperti total_Wh dan total_Ah) ke nilai awal nol untuk memulai sesi pengukuran yang baru.
2. **Loop Utama (loop()):** Fungsi ini berjalan secara berulang tanpa henti selama perangkat menyala. Setiap siklus loop merepresentasikan satu interval pengukuran ($\Delta t = 100ms$).
   - **Akuisisi Data:** Arduino membaca nilai analog dari pin yang terhubung ke sensor tegangan dan sensor arus.
   - **Konversi dan Kalibrasi:** Nilai analog mentah yang terbaca (0—1023) dikonversi menjadi nilai rekayasa dalam satuan fisik (Volt dan Ampere) dengan mengaplikasikan faktor kalibrasi yang telah ditentukan pada tahap pengujian awal.
   - **Kalkulasi Daya Sesaat (P):** Daya listrik sesaat dihitung menggunakan rumus:

     $$P = V\ x\ I$$

   - **Implementasi Integrasi Numerik:** Pada setiap interval waktu (Δt), program menghitung porsi kecil energi dan kapasitas yang digunakan pada interval tersebut. Nilai Δt (100 ms) dikonversi ke dalam satuan jam (0. 0000278 jam) untuk kalkulasi Wh dan Ah.

     $$dWh = P\ x\ \Delta t$$
     $$dAh = I\ x\ \Delta t$$

   - **Akumulasi Nilai:** Nilai dWh dan dAh yang didapat kemudian ditambahkan secara kumulatif ke variable total $Wh = Wh + dWh$ dan $Ah = Ah + dAh$. Proses penjumlahan berulang inilah yang secara esensial merupakah implementasi dari integrasi numerik.
   - **Pembaruan Tampilan:** Nilai-nilai terbaru (V, I, Wh, Ah, h) dikirm ke layar OLED untuk memperbarui informasi yang dilihat oleh pengguna.

4.3.3. Kode Program

Implementasi dari logika akuisisi, perhitungan, dan akumulasi tersebut bisa dilihat pada Gambar 4.4 sampai 4.6 berikut. Kode ini menunjukkan bagaimana nilai tegangan dan arus dibaca, dikalibrasi, dan kemudian digunakan dalam *loop* akumulasi untuk menghitung total energi dan kapasitas yang terpakai.

```
#Program Komputer untuk Energy Meter 50-100v / 0-30A
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
float k_V_in = 0.9836;
float k_V_out = 0.9836;
float vn = 2.49;
float k_I = 0.8803;
float v_per_a = 0.066;
float dt = 100;
float k_dt = 1.0;
int n = 36;
float t = (n*dt/1000)/3600;
float Threshold_Charging_atas = 81.0;
float Threshold_Charging_bawah = 78.0;
//Setup Pin-pin Analog dan Digital
int VinPin = A6;
int VoutPin = A7;
int amperePin = A0;
int led = LED_BUILTIN;
int status_LED = HIGH;
// Setup OLED
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1    // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); //Buat objek display dari class adafruit_ssd1306

// Bitmap untuk Logo
const unsigned char upj [] PROGMEM = {
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7c, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

Gambar 4.4. Potongan Kode Program (1)

```
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x03, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x0f, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1c, 0x1e, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xfc, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
void setup() { // put your setup code here, to run once:
 Serial.begin(9600);
  // Initialize with the I2C addr 0x3C (for the 128x64)
 if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
  Serial.println(F("SSD1306 allocation failed"));
  for(;;); // Don't proceed, loop forever
}
  //Booting logo
 display.clearDisplay();
 display.drawBitmap(0, 0, upj, 128, 32, 1);
 display.display();
 delay(1000);
 pinMode(led, OUTPUT);
 pinMode(relay, OUTPUT);
 digitalWrite(led, status_LED);
 digitalWrite(relay, status_Relay);
}
void loop() {  // put your main code here, to run repeatedly:
  //Sampling V_in dan V_out untuk Tiap Iterasi
 rawValVolt_in = analogRead(VinPin);
 rawValVolt_out = analogRead(VoutPin);
 V_sampling_in = k_V_in * 80.0/1023.0 * rawValVolt_in;
 V_sampling_out = k_V_out * 80.0/1023.0 * rawValVolt_out;
 V_in = V_in + V_sampling_in;
 V_out = V_out + V_sampling_out;
```

Gambar 4.5. Potongan Kode Program (2)

```
V_out = V_out + V_sampling_out;

rawValAmp = analogRead(amperePin);
v_I = 5.0 / 1023.0 * rawValAmp;
I_sampling = k_I * (v_I - vn) / v_per_a;
I = I + I_sampling;
if(counter == n){
 counter = 0;
 status_LED = !status_LED;  //Membuat LED berkedip setiap n sampling
 digitalWrite(led, status_LED);
 V_in = V_in/n; V_out = V_out/n; I = I/n;
   //Ampere-hour Calculation
 dAh = I * t;
 Ah = Ah + dAh;
   //Watt-hour Calculation
 dWh = (V_out * I) * t;
 Wh = Wh + dWh;
   //Timer dalam Jam
 timer = timer + t;
   //Cetak Hasil ke OLED
 display.clearDisplay(); display.setTextSize(1); display.setTextColor(1);
 display.setCursor(0, 0); display.print(V_in,2); display.print("v");
 display.setCursor(0, 16); display.print(timer,3); display.println("h");
 display.setCursor(42, 0); display.print(I,2); display.print("A");
 display.setCursor(42,16); display.print(Ah,2); display.println("Ah");
 display.setCursor(84, 0); display.print(V_out,2); display.print("v");
 display.setCursor(84, 16); display.print(Wh,2); display.print("Wh");
 display.display();
 //Cetak Hasil ke PC
 Serial.print(V_in,2); Serial.print("v  "); Serial.print(timer,3); Serial.println("h");
 Serial.print(I,2); Serial.print("A  "); Serial.print(Ah,2); Serial.println("Ah");
 Serial.print(V_out,2); Serial.print("v  "); Serial.print(Wh,2); Serial.println("Wh");
 Serial.println();
   //Kembalikan V dan I ke nol namun tidak untuk Ah dan Wh.
 V_in = 0; V_out = 0; I = 0;
 }
 counter = counter + 1;
 delay(k_dt * dt);
}
```

Gambar 4.6. Potongan Kode Program (3)

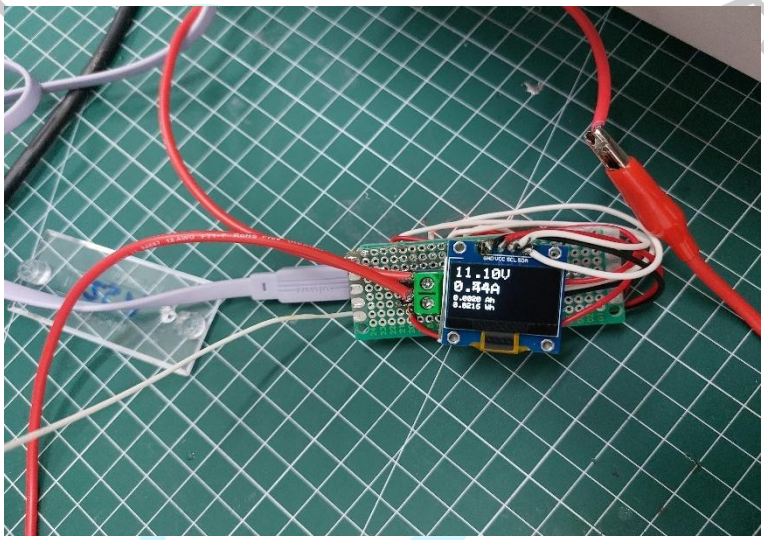
Kode di atas menunjukkan bagaimana nilai rata-rata dari tegangan dan arus digunakan untuk menghitung penambahan energi (*dWh*) dan kapasitas (*dAh*), yang kemudian diakumulasikan. Proses ini secara efektif mengimplementasikan metode integrasi numerik untuk mendapatkan total konsumsi energi secara akurat.

4.4. Implementasi Prototipe

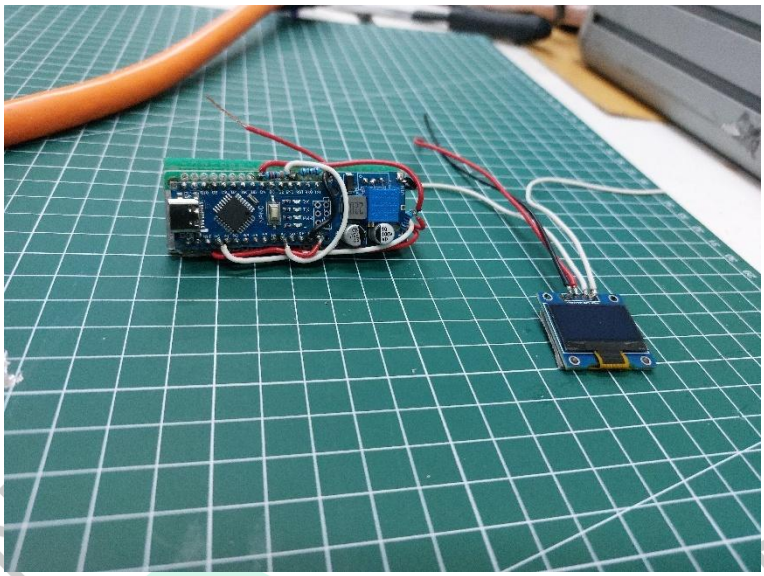
Proses implementasi merupakan jembatan antara rancangan konseptual dengan produk fisik. Tahap ini mengubah diagram skematik dan alur logika menjadi sebuah prototipe fungsional melalui serangkaian langkah perakitan dan perakitan yang terdokumentasi.

4.4.1. Perakitan Prototipe Awal

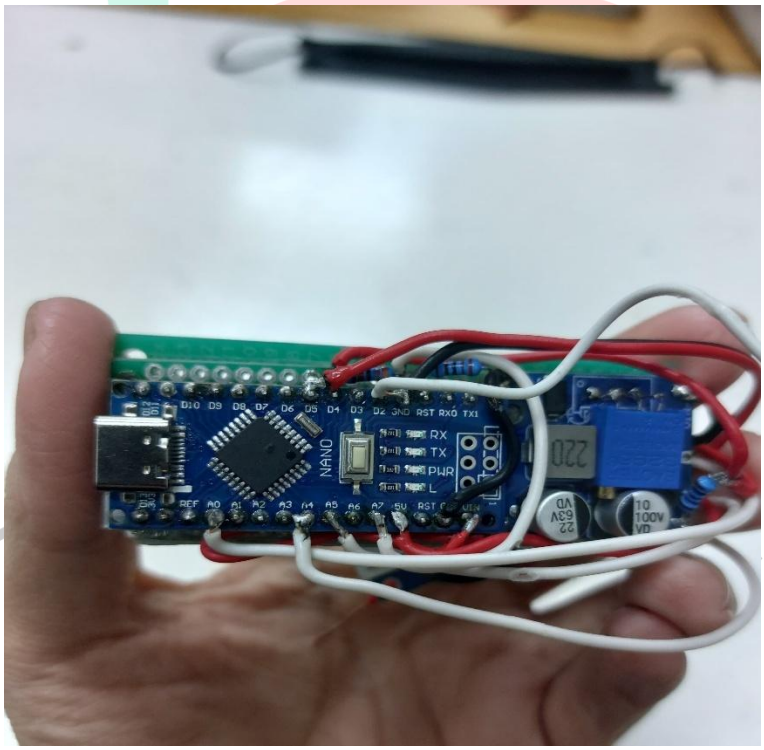
Tahap paling awal dari implementasi adalah perakitan komponen-komponen utama pada papan rangkaian. Tujuan dari fase ini adalah untuk melakukan validasi fungsional awal terhadap rangkaian inti, memastikan setiap sensor memberikan pembacaan yang masuk akal, serta menguji logika fundamental dari program yang dikembangkan. Prototipe pada tahap ini sengaja dirakit tanpa casing untuk memberikan aksesibilitas maksmila saat melakukan *debugging*, pengukuran, dan iterasi perbaikan, sebagaimana yang ditampilkan pada Gambar 4.7 sampai 4.10.



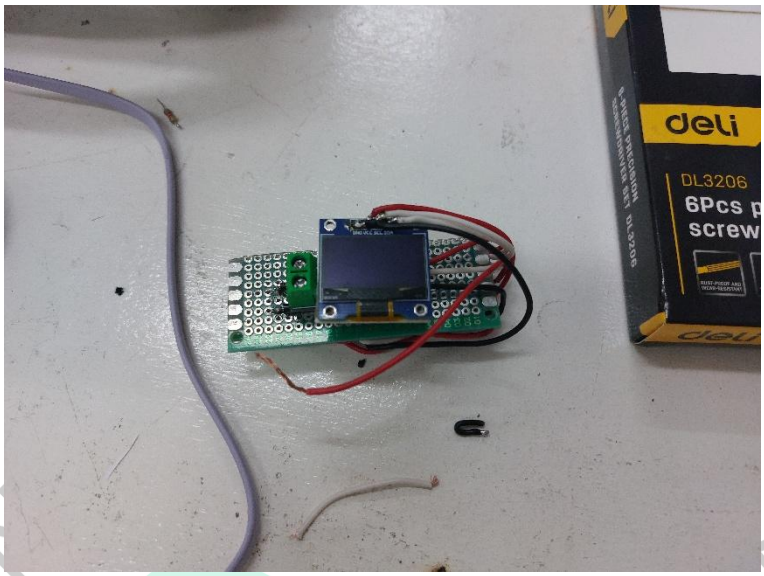
Gambar 4.7. Foto Tahap Awal Pengembangan (1)



Gambar 4.8. Foto Tahap Awal Pengembangan (2)



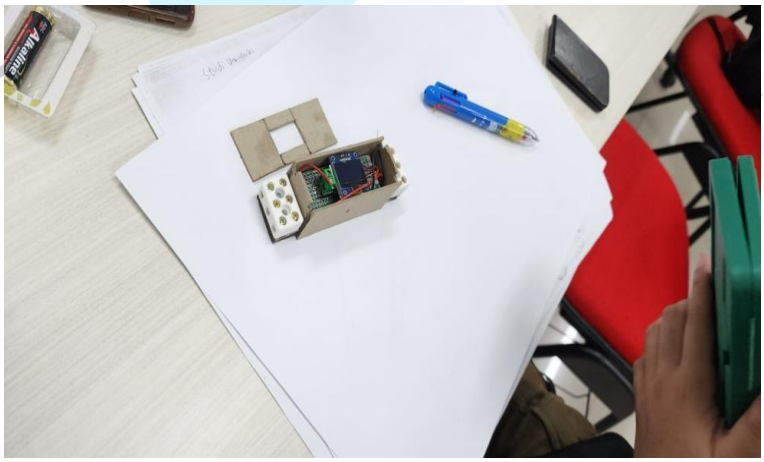
Gambar 4.9. Foto Tahap Awal Pengembangan (3)



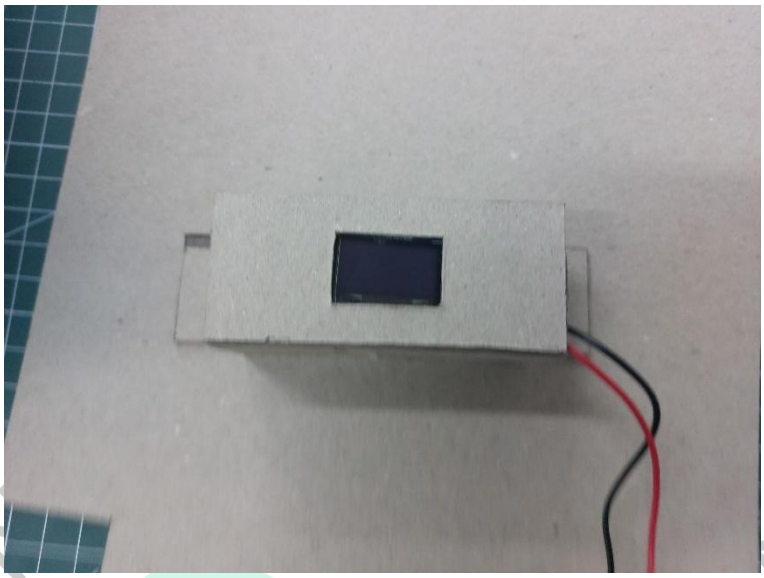
Gambar 4.10. Foto Tahap Awal Pengembangan (4)

4.4.2. Desain dan Pembuatan Casing

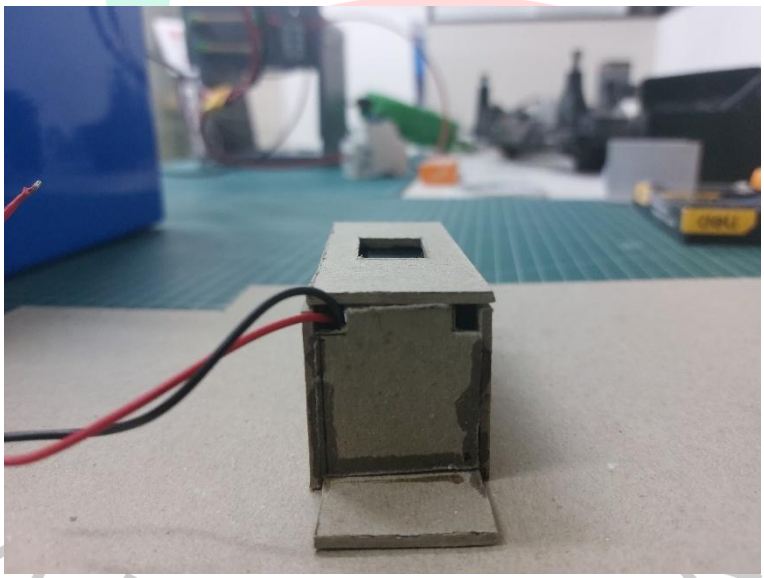
Untuk melindungi sirkuit elektronik yang rentan dan memberikan wujud produk yang lebih ringkas dan terintegrasi, sebuah casing pelindung dirancang secara khusus. Proses ini dimulai dengan pembuatan desain awal menggunakan bahan *hardboard* yang mudah dibentuk. Gambar 4.11 hingga 4.14 menunjukkan bahwa iterasi awal sangat penting untuk memastikan presisi dimensi dan tata letak internal komponen.



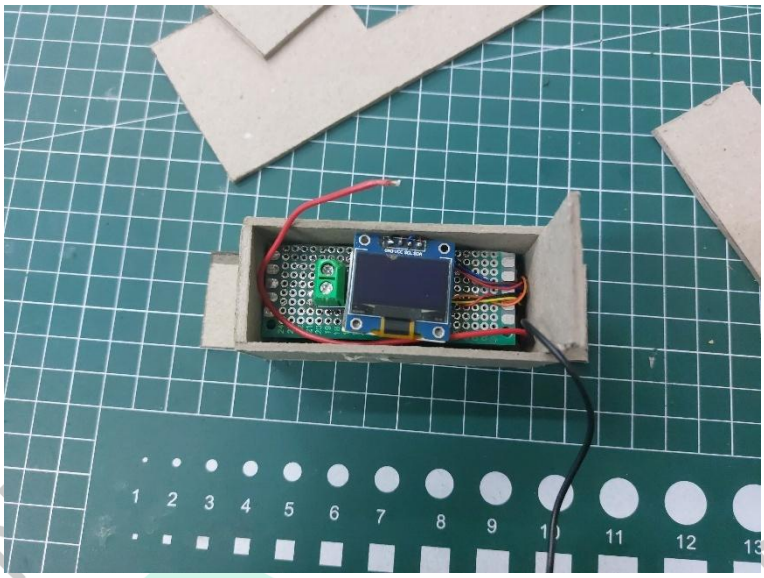
Gambar 4.11. Foto Desain *Casing* Awal menggunakan *Hardboard* (1)



Gambar 4.12. Foto Desain *Casing* Awal menggunakan *Hardboard* (2)

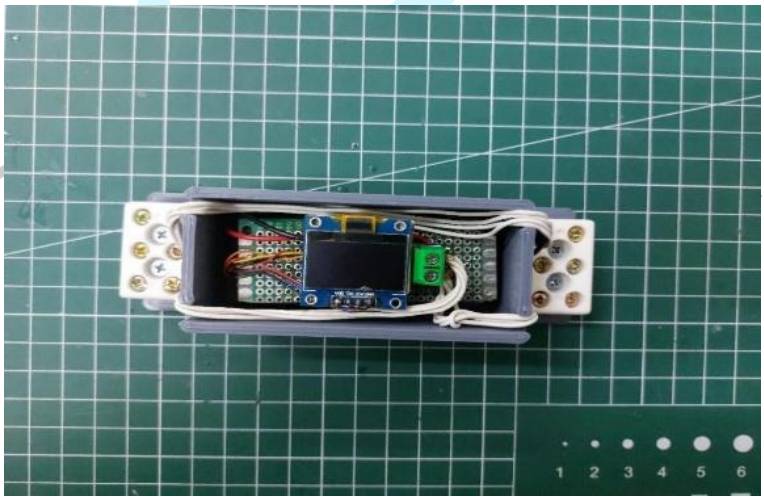


Gambar 4.13. Foto Desain *Casing* Awal menggunakan *Hardboard* (3)



Gambar 4.14. Foto Desain *Casing* Awal menggunakan *Hardboard* (4)

Setelah desain terbukti pas dan fungsional, *casing* final diproduksi menggunakan teknologi 3D *printing.* Metod ini dipilih untuk menghasilkan *casing* yang tidak hanya presisi secara dimensional tetapi juga memiliki durabilitas dan estetika yang lebih profesional, sehingga layak untuk dipasang pada unit kendaraan Gambar 4.15.

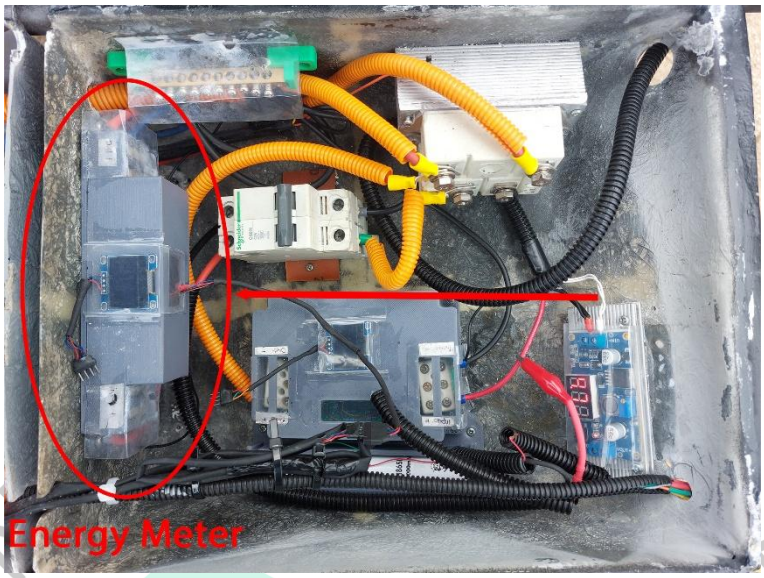


Gambar 4.15. Foto *Energy Meter* dengan *Casing 3D Print*

4.4.3. Instalasi pada *E-Cargo Bike*

Tahap inti dari proses implementasi adalah integrasi fisik prototipe final ke dalam unit *e-cargo bike* yang sesungguhnya. Pemilihan lokasi pemasangan dilakukan secara strategis dengan mempertimbangkan kemudahan aksesnya, serta perlindungan maksimal dari potensi guncangan, getaran, dan paparan cuaca selama operasional. Proses instalasi melibatkan pengeboran pada kerangka kendaraan untuk menciptakan titik pemasangan yang kokoh, diikuti dengan pemasangan modul menggunakan mur dan baut, sebagaimana terlihat pada gambar 4.16 dan 4.17.

Gambar 4.16. Foto Instalasi *Energy Meter* pada *e-cargo bike* (1)



Gambar 4.17. Foto Instalasi *Energy Meter* pada *e-cargo bike* (2)

4.4.4. Fitting Solar Panel

Pada Gambar 4.18 dilakukan pemasangan *solar panel* ke *bracket*, proses pemasangan *solar panel* dilakukan dengan mempertimbangkan sudut terhadap matahari untuk mendapatkan efisiensi energi yang maksimal.

Gambar 4.18. Foto *Fitting Solar Panel*

4.4.5. Fitting Baterai

Pada Gambar 4.19 dilakukan *fitting* baterai. Langkah ini mencakup pengaturan posisi baterai dan penguncian menggunakan *bracket* untuk menghindari pergeseran saat kendaraan digunakan.

Gambar 4.19. Foto *Fitting* Baterai

4.5. Perancangan Pengujian

Tahap perancangan pengujian merupakan fase untuk memvalidasi bahwa prototipe energy meter yang dikembangkan telah memenuhi seluruh spesifikasi kebutuhan fungsional dan non-fungsional yang telah ditetapkan. Untuk mendapatkan evaluasi yang menyeluruh, proses pengujian akan menggunakan dua metode utama: *black box* dan *white box*.

4.5.1. Perancangan Black Box

Pengujian Black Box difokuskan untuk memverifikasi fungsionalitas sistem dari sudut pandang pengguna, tanpa perlu mengetahui detail implementasi kode di dalamnya. Tujuan dari pengujian ini adalah untuk memastikan bahwa setiap fitur bekerja sesuai dengan harapan. Tabel 4.1 berikut menunjukkan skenario pengujian yang dirancang:

Tabel 4.1. *Tabel Perancangan Black Box*

No.	Pengujian	Hasil yang Diharapkan
1.	Inisialisasi & Tampilan Awal	1. Layar *OLED* menyala. 2. Logo pembuka ditampilkan. 3. Semua parameter (V, A, Wh, Ah, h) muncul dengan nilai awal 0 atau sesuai kondisi awal.
2.	Akurasi Pengukuran Tegangan	Nilai tegangan yang ditampilkan di OLED memiliki *error* yang kecil ketika dibandingkan dengan multimeter.
3.	Akurasi Pengukuran Arus	Nilai Arus yang ditampilkan di OLED memiliki *error* yang kecil ketika dibandingkan dengan multimeter.
4.	Akurasi Akumulasi Ah dan Wh	Hasil *error* pada Ah dan Wh yang yang dilakukan dengan perhitungan manual memiliki nilai yang rendah.
5.	Uji Fungsional pada *e-cargo bike*	Modul tetap menyala dan menampilkan semua parameter secara *real-time* tanpa ada gangguan atau mati mendadak saat kendaraan beroperasi.

4.5.2. Perancangan White Box

Pengujian White Box dirancang untuk menganalisis dan memvalidasi kode program yang telah diimplementasikan pada *mikrokontroler Arduino Nano*. Tujuannya adalah untuk menemukan potensi *bug* atau ketidakefisienan dalam kode. Tabel 4.2 menunjukkan skenario pengujian yang akan dilakukan:

Tabel 4.2. *Tabel Perancangan White Box*

No.	Komponen Diuji	Hasil yang Diharapkan
1.	Fungsi setup()	Semua proses inisialisasi (komunikasi serial, layar *OLED*, *pin I/O*) berhasil dijalankan sekali tanpa *error* saat perangkat pertama kali dinyalakan.
2.	Fungsi loop() dan Interval Waktu	*Loop* berjalan tanpa henti dan jeda waktu antar iterasi sesuai dengan nilai yang ditetapkan (100 ms), memastikan *sampling rate* yang konsisten.
3.	Konversi Data Sensor dan Kalibrasi	Perhitungan matematis dalam kode untuk konversi data analog ke digital dan kalibrasi sesuai dengan rumus yang telah dirancang.
4.	Logika Integrasi Numerik	Proses penjumlahan kumulatif pada setiap interval waktu berjalan dengan benar, sesuai dengan implementasi metode integrasi numerik untuk mengakumulasi energi dan kapasitas.
5.	Reset Variabel Perhitungan	Variabel V dan I kembali ke 0 setelah data ditampilkan, sedangkan Ah dan Wh tidak di-reset, memastikan perhitungan rata-rata yang benar tanpa kehilangan data akumulasi.