

BAB IV

PEMBUATAN ALAT

Proses pembuatan alat *Wire Coiller* otomatis meliputi tahap realisasi perangkat keras (*hardware*) dan perangkat lunak (*software*) yang digunakan untuk mendukung sistem kerja alat secara keseluruhan. Tahapan ini dilakukan untuk memastikan seluruh komponen elektrik dan mekanik dapat terintegrasi dengan baik sesuai dengan fungsi dan rancangan yang telah ditentukan. Pembuatan alat dilakukan melalui beberapa tahapan sistematis, dimulai dari persiapan alat dan bahan, perakitan rangka struktural menggunakan *Aluminium Profile V-Slot* sepanjang 70 cm beserta roda pemandu *Openbuilds*, pemasangan transmisi linier lead screw 70 cm yang dihubungkan ke motor *stepper* Nema 17 menggunakan *Flexible Coupler*, penyusunan rangkaian elektronik sesuai *wiring* diagram, hingga proses pemrograman mikrokontroler Arduino Uno. Seluruh tahapan ini dilakukan secara bertahap agar proses konversi gerak dari rotasi menjadi translasi pada komponen jig pemandu kawat dapat berjalan dengan lancar dan menghasilkan kerapatan lilitan yang seragam.

Selain proses pembuatan, pada bab ini juga dijelaskan proses pengujian dasar terhadap sistem yang telah dirakit untuk memvalidasi performa kendali *Open Loop* yang diterapkan. Pengujian dilakukan untuk memastikan fungsi setiap bagian alat dapat bekerja sesuai dengan perancangan teknis, meliputi akurasi pergerakan Nema 17 melalui perintah *Driver A4988*, ketepatan logika perhitungan konversi nilai resistansi *Input* (rentang 30 ohm hingga 120 ohm dengan kelipatan 5 ohm) menjadi jumlah lilitan fisik, responsivitas navigasi tombol *keypad* 4x4 (tombol A, B, C, # dan *), kestabilan visualisasi data lilitan secara *real-time* pada layar LCD 16x2, hingga efisiensi distribusi daya dari *switching power supply* 12 VDC dan *step down* LM2596. Melalui tahapan realisasi dan uji repetisibilitas mekanik ini, diharapkan alat penggulung kawat otomatis ini mampu menghasilkan koil resistansi nikelin secara mandiri dengan tingkat kepresisian dan standarisasi nilai hambatan yang tinggi.

4.1 Alat dan Bahan

Pada tahap pembuatan alat *Wire Coiller* otomatis ini, dilakukan proses realisasi fisik berdasarkan hasil perancangan sistem yang telah dijelaskan pada Bab III. Proses fabrikasi ini melibatkan penggunaan berbagai alat kerja perakitan serta bahan atau komponen elektronika dan mekanika yang dibutuhkan untuk membangun sistem secara utuh. Alat-alat kerja digunakan untuk mendukung proses manufaktur dan instalasi, seperti penyolderan komponen kontrol, perakitan, pengujian rangkaian, hingga pengujian fungsional rangkaian menggunakan alat ukur elektrik. Sementara itu, bahan merupakan komponen utama yang dirakit menjadi satu kesatuan sistem mekanik sepanjang 70 cm dan sistem elektronik terintegrasi berbasis Arduino Uno sesuai dengan skematik yang telah dirancang sebelumnya.

Pemilihan alat dan bahan ini disesuaikan secara presisi dengan kebutuhan teknis selama proses pembuatan. Hal ini mencakup kebutuhan perakitan mekanis rel *V-Slot Aluminium Profile* dengan roda *Openbuilds V-Wheel Stainless*, penyambungan poros motor *stepper* Nema 17 ke *lead screw* 70 cm menggunakan *Flexible Coupler*, hingga penyusunan perkabelan *wiring* dari matriks *Keypad 4x4*, *step down* LM2596, *Driver A4988*, dan layar *LCD 16x2 I2C*. Seluruh alat bantu kerja dan bahan baku komponen yang digunakan dalam proses realisasi ini disajikan dalam bentuk tabel berikut untuk memudahkan identifikasi serta memberikan gambaran umum mengenai struktur pembentuk alat *Wire Coiller* otomatis.

Daftar komponen perangkat keras dan perangkat lunak yang digunakan dalam pengembangan sistem ini dapat dilihat pada Tabel 4-1 berikut:

Tabel 4- 1 Alat dan Bahan

Nama	Satuan	Jumlah
Arduino Uno R3	Unit	1
Key Pad 4x4	Unit	1
LCD 16x2 I2C	Unit	1
<i>Driver Stepper A4988</i>	Unit	1
Motor Stepper Nema 17	Unit	1
Step Down LM2596	Unit	1
Power Supply 12VDC	Unit	1

Kapasitor Elco 470uF	Unit	1
Power Switch	Unit	1
Kabel Jumper Male to Female	Unit	20
Kabel Serabut 22 AWG	m	1
PCB	cm ²	52
V-Slot Aluminium Profile	Cm	70
THSL T8 Lead Screw 8mm	Cm	70
Flange Coupling 5mm	Unit	1
Bearing 5mm	Unit	4
Flexible Coupling 5mm to 8mm	Unit	1
Copper 10mm x 3mm Hexagonal	Unit	4
Braket Motor Nema 17	Unit	1
Box Kotak Plastik	Unit	1
T Nut T8 Leadsrew	Unit	1
Solder 60W	Unit	1
Timah	m	25

4.2 Perancangan Perangkat Keras (Hardware)

Pada tahap ini dilakukan proses realisasi fisik perangkat keras berdasarkan perancangan mekanik dan elektrik yang telah dibahas BAB III sebelumnya. Perangkat keras pada alat *Wire Coiller* otomatis ini terbagi menjadi dua pilar utama, yaitu sistem mekanik struktural dan sistem elektronik terintegrasi yang saling terhubung untuk mendukung proses pelilitan kawat nikelin secara otomatis. Berbeda dengan metode fabrikasi konvensional yang membutuhkan pencetakan material ekstrusi mentah, proses pembuatan alat ini berfokus pada teknik manufaktur modular dan perakitan presisi. Tahapan pengerjaan dimulai dari pengeboran kerangka *V-Slot Aluminium Profile* sepanjang 70 cm untuk tempat braket khusus, pemasangan roda pemandu *Openbuilds V-Wheel Stainless Steel* padaudukan *jig*, pembuatan *jig* khusus, penyelarasan poros transmisi linier motor *stepper* Nema 17 sampai *lead screw* 70 cm, hingga instalasi seluruh modul elektronik ke dalam panel kontrol utama.

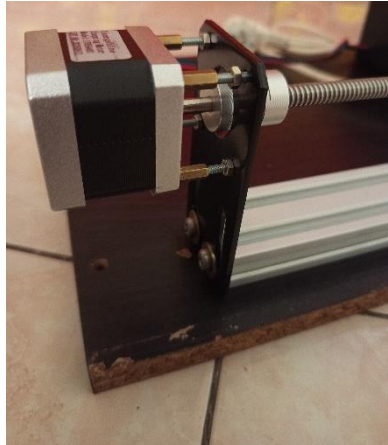
Perancangan dan perakitan perangkat keras ini dilakukan dengan mempertimbangkan faktor rigiditas struktural, kestabilan distribusi massa, kemudahan perawatan (*maintainability*), serta kesesuaian fungsi dinamis setiap komponen terhadap beban puntir saat kawat nikelin ditarik. Seluruh bagian dirancang agar dapat bekerja secara sinkron tanpa umpan balik sensor (*Open*

Loop system). Sinkronisasi mekanis ini mengandalkan konversi gerak dari putaran poros tunggal motor *stepper* Nema 17 yang diteruskan melalui *Flexible Coupler* menuju *lead screw* untuk menggerakkan jig pengarah kawat secara translasional di atas rel profil aluminium. Dengan adanya integrasi yang kokoh antara struktur mekanik linier dan sistem aktuasi elektronik ini, perangkat keras yang direalisasikan diharapkan mampu melakukan proses pelilitan secara otomatis dan konsisten untuk menghasilkan susunan koil nikelin yang rapi serta memenuhi standarisasi nilai resistansi yang presisi.

4.2.1 Perakitan Mekanik Alat

Pada tahap ini, dilakukan proses perakitan seluruh komponen mekanik modular menjadi satu kesatuan sistem alat *Wire Coiller* otomatis. Proses perakitan dilakukan secara presisi berdasarkan desain mekanik yang telah direncanakan sebelumnya, dengan memperhatikan keselarasan posisi (*alignment*) setiap komponen agar sistem dapat bergerak secara stabil selama proses penggulungan kawat nikelin berlangsung.

Perakitan diawali dengan pemasangan sistem mekanik alat *Wire Coiller* otomatis ini secara terintegrasi untuk mentransmisikan gerakan rotasi dari aktuator menjadi gerakan translasi linier sepanjang sumbu horizontal secara stabil. Tahap awal dari perancangan fisik ini bertumpu pada *v-slot Aluminium Profile* sepanjang 70 cm dengan massa jenis linier 15 g/cm, sehingga diperoleh berat total material struktur kurang lebih 1 kg. Komponen ini mengemban fungsi ganda yang sangat krusial, yaitu sebagai fondasi struktural dasar untuk menopang keseluruhan bodi mesin sekaligus bertindak langsung sebagai jalur rel pemandu linier bagi pergerakan mekanis *jig*.



Gambar 4- 1 Nema 17 dan Flexible Coupler yang terpasang

Memasuki tahap pengerjaan berikutnya, di salah satu ujung rel aluminium dipasang Mekanisme Transmisi Tunggal (Nema 17 dan *lead screw* T8) sebagai penggerak utama sistem. Pada langkah ini, motor *stepper* Nema 17 dikunci secara rigid pada kerangka *v-slot* menggunakan komponen braket penopang khusus (Gambar 4- 1). Perancangan dan pemasangan braket ini dispesifikasikan secara kuat untuk menahan beban torsi operasional yang muncul saat proses penarikan kawat nikelin, sekaligus meredam getaran aksial merugikan yang dihasilkan oleh putaran motor agar tidak memengaruhi komponen lainnya.



(a)



(b)

Gambar 4- 2(a) Nema 17 terhubung dengan Lead Screw sampai Jig (b) Lead Screw terhubung dengan Jig menggunakan Nut T8

Setelah motor terpasang stabil pada braketnya, poros keluaran dari motor *stepper* Nema 17 kemudian dihubungkan secara koaksial ke komponen *lead*

screw T8 sepanjang 70 cm (Gambar 4- 2a) dengan diameter 8 mm. Hubungan antar kedua poros ini memanfaatkan komponen *Flexible Coupler* (5mm ke 8mm) yang dikencangkan secara presisi. Penggunaan penghubung fleksibel ini diposisikan sebagai langkah pemutaran mekanis yang bertujuan untuk keselarasan sudut putar mikro antar sumbu poros, sekaligus menyerap beban kejut (*shock load*) secara efektif saat motor melakukan akselerasi putaran awal.

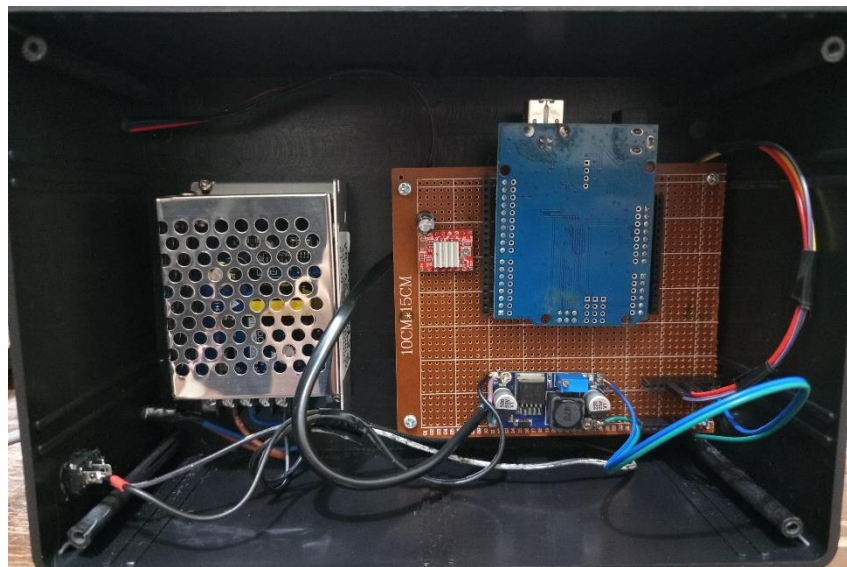


Gambar 4- 3 Sistem Mekanik terhubung dengan Sistem Elektrik

Pada tahapan akhir perancangan mekanik, pergerakan translasi di sepanjang lintasan rel dieksekusi oleh mekanisme rel penggeser (*jig* dan *openbuilds v-wheel stainless*). Komponen *jig* pemandu kawat nikelin dapat meluncur dinamis di atas rel aluminium berkat dukungan roda pemandu CNC *openbuilds v-wheel stainless* yang duduk tepat pada lekukan berbentuk huruf "V" profil aluminium. Integrasi ini memastikan pergerakan linier *jig* berlangsung halus, minim hambatan gesek (*low friction*), dan bebas dari risiko macet (*binding*). Seluruh pergerakan maju-mundur *jig* pengarah kawat ini dikendalikan secara mekanis penuh melalui interaksi dinamis antara ulir putar *lead screw* dengan komponen *brass nut* (mur kuningan) yang tertanam secara presisi pada *body jig*.

4.2.2 Perakitan Sistem Elektronik Alat

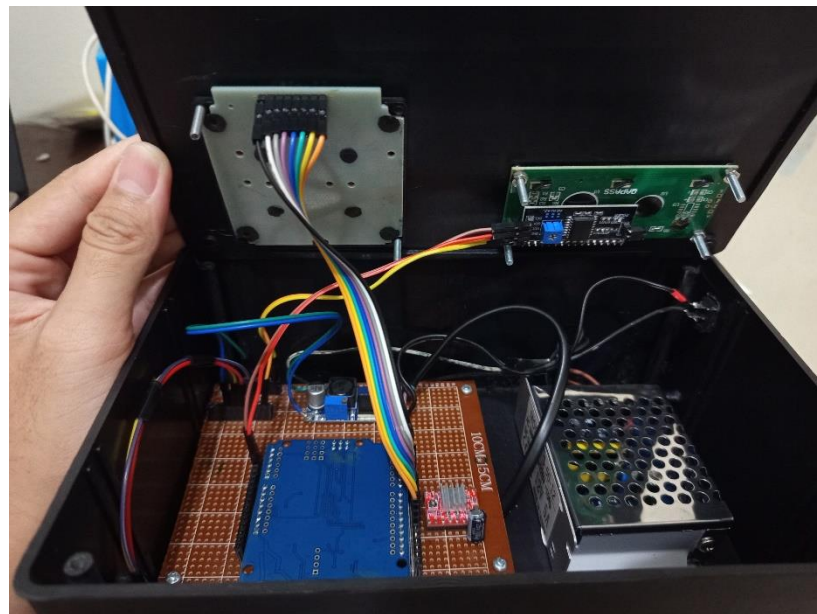
Pada tahap ini, dilakukan proses perakitan dan pemasangan seluruh komponen elektronik yang digunakan pada alat *Wire Coiller* otomatis. Sistem elektronik ini dirancang khusus untuk mengendalikan seluruh proses kerja alat secara *Open Loop*, mulai dari pembacaan *Input* parameter resistansi oleh pengguna, pemrosesan kalkulasi jumlah lilitan, pengendalian arah dan langkah aktuator, hingga visualisasi status sistem pada layar LCD. Seluruh komponen elektrik dihubungkan secara terintegrasi di dalam sebuah box hitam (panel kontrol) berdasarkan *wiring diagram* yang telah dirancang sebelumnya guna memastikan transmisi sinyal data dan daya berjalan dengan stabil.



Gambar 4- 4 Pemasangan Arduino Uno, A4988, LM2596 dan Power Supply 12 VDC

Proses instalasi elektronika di dalam box hitam diawali dengan pemasangan Arduino Uno R3 sebagai unit pemroses pusat sistem. Arduino Uno R3 berfungsi menerima enkripsi data dari *keypad*, mengolah konversi nilai resistansi menjadi parameter putaran mekanis, serta mengirimkan instruksi logika digital menuju unit penggerak dan modul tampilan. Langkah berikutnya adalah pemasangan satu buah modul *Driver motor stepper* A4988. *Driver* ini dihubungkan ke Arduino Uno melalui pin digital

D3 untuk sinyal pulsa langkah (STEP), pin D2 untuk kendali arah putaran (DIR), dan pin *Enable* (EN) terhubung ke GND untuk mengaktifkan *Driver* atau digunakan untuk mematikan motor secara cepat. Guna menjaga *Driver* selalu dalam kondisi siap beroperasi (*always awake*), pin *RESET* dan *SLEEP* pada modul A4988 disambungkan secara langsung melalui jalur *jumper*. Jalur daya utama 12 VDC dari *power supply* ke *Driver* ini juga dipasang sebuah kapasitor elco 470 μ F secara paralel untuk meredam *voltage spike* induktif.



Gambar 4- 5 Pemasangan rangkaian akhir pada Box Hitam

Konektor fasa dari motor *stepper* Nema 17 kemudian dihubungkan menuju pin *Output Driver* A4988 (pasangan pin 1A-1B untuk coil 1 dan pin 2A-2B untuk coil 2). Motor *stepper* Nema 17 ini bertindak sebagai aktuator tunggal yang memutar poros *lead screw* untuk menggerakkan *body jig* di atas rel aluminium. Sebagai media interaksi operator, komponen *keypad* 4x4 dan LCD 16x2 I2C dipasang terintegrasi pada bagian permukaan luar penutup box hitam. *Keypad* 4x4 dihubungkan secara matriks berurutan ke pin digital D4 hingga D11 Arduino Uno untuk menangkap *Input* nilai resistansi kelipatan 5 ohm serta eksekusi tombol navigasi tombol '#' untuk *start*, '*' untuk menghapus nilai target yang salah, 'A' untuk melihat data

koil yang telah terbentuk sebelumnya, 'B' untuk *jogging* maju manual, 'C' untuk *Unwinding* dan 'D' untuk *instant stop*. Sementara itu, LCD 16x2 dihubungkan menggunakan modul I2C yang jalurnya dipetakan langsung menuju pin analog A4 (SDA) dan A5 (SCL) Arduino Uno. Implementasi komunikasi serial I2C ini secara signifikan mampu memangkas kebutuhan jumlah kabel sehingga tata letak di dalam box (panel kontrol) menjadi jauh lebih ringkas dan rapi.



Gambar 4- 6 Box Rangkaian Wire Coiller

Sistem distribusi daya total pada alat ini disuplai oleh *switching power supply* dengan *Input* tegangan 220 VAC yang dikonversi menjadi *Output* 12 VDC 5A. Tegangan 12 VDC ini dialirkan langsung untuk memasok daya motor melalui pin VMOT pada *Driver* A4988. Untuk menyuplai kebutuhan daya sirkuit logika, jalur 12 VDC tersebut dicabangkan masuk menuju *Input* komponen *step down* LM2596 DC-DC. Modul LM2596 ini bertugas menurunkan tegangan secara konstan menjadi 5 VDC, yang kemudian dihubungkan langsung untuk memberikan catu daya aman bagi pin 5V Arduino Uno, pin VDD *Driver* A4988, serta terminal VCC pada LCD 16x2 I2C. Seluruh terminal *Ground* (GND) dari *power supply*, LM2596, Arduino Uno, LCD, hingga *Driver* disatukan ke dalam satu titik referensi yang sama (*common ground*) demi mencegah terjadinya interferensi sinyal digital listrik. Setelah seluruh interkoneksi kabel selesai dirakit, dilakukan pengujian kontinuitas jalur dan pengecekan polaritas tegangan

menggunakan multimeter digital untuk memastikan tidak adanya hubungan arus pendek sebelum sistem memasuki tahapan integrasi perangkat lunak (*software coding*).

4.3 Perancangan Perangkat Lunak

Pada tahap ini, dilakukan proses perancangan dan implementasi perangkat lunak (*software*) yang digunakan untuk mengendalikan seluruh sistem kerja pada alat *Wire Coiller* otomatis. Perangkat lunak ini berfungsi sebagai jembatan instruksi logika antara komponen perangkat keras (*hardware*) dengan mikrokontroler Arduino Uno sebagai pusat kontrol utama, sehingga setiap komponen dapat mengeksekusi aksi mekanis dan visual sesuai dengan fungsi yang telah dirancang. Program kendali ini dibangun menggunakan bahasa pemrograman Arduino (berbasis C/C++) di dalam lingkungan *software* Arduino IDE, yang kemudian diunggah ke papan Arduino Uno R3 melalui komunikasi serial via kabel USB.

Perancangan perangkat lunak dilakukan dengan menyusun logika algoritma terstruktur agar mampu mengatur proses kerja alat secara otomatis tanpa adanya umpan balik dari sensor (*Open Loop control system*). Alur kode program dirancang untuk mengoordinasikan komunikasi data dan pengiriman pulsa digital dari Arduino Uno menuju berbagai komponen terintegrasi, seperti modul tampilan LCD 16x2 I2C, matriks tombol *keypad* 4x4, serta pin kontrol (STEP, DIR, EN) pada modul *Driver* A4988 untuk menggerakkan aktuator motor *stepper* Nema 17.

Pada alat *Wire Coiller* ini, perangkat lunak memegang beberapa fungsi utama yang sangat krusial. Fungsi tersebut meliputi proses inisialisasi seluruh pin I/O (*Input/Output*) saat alat ini pertama kali dinyalakan, membaca parameter nilai hambatan variabel (30 ohm hingga 120 ohm dengan sistem *Input* kelipatan 5 ohm) yang dimasukkan operator melalui *keypad*, mengonversi parameter resistansi tersebut secara matematis menjadi akumulasi jumlah lilitan, serta mengendalikan pergerakan presisi dari motor *stepper* Nema 17.

Selama motor *stepper* Nema 17 beroperasi, perangkat lunak dikonfigurasi agar mampu memperbarui data dan menampilkan informasi status penggulungan secara *real-time* ke layar LCD 16x2 I2C, sehingga operator dapat memantau jalannya proses pelilitan koil nikelin dengan mudah. Sub-bab ini akan menjelaskan secara terperinci mengenai tahapan persiapan pemrograman pada Arduino IDE, struktur logika diagram alir program, cara kerja kode eksekusi pada setiap komponen elektrik yang digunakan, serta implementasi akhir dari perangkat lunak terhadap keseluruhan sistem alat *Wire Coiller* otomatis

4.3.1 Persiapan Pemrograman

Sebelum proses penyusunan kode program dilakukan pada alat *Wire Coiller* otomatis, diperlukan beberapa tahapan persiapan terstruktur agar fase pemrograman dan pengunggahan (*uploading*) data ke papan mikrokontroler dapat berjalan dengan lancar. Tahap persiapan ini meliputi instalasi perangkat lunak lingkungan pengembangan (IDE), pengintegrasian pustaka (*library*) eksternal penunjang komponen, konfigurasi jenis papan pengembang (*board*), serta penyesuaian port komunikasi serial antara mikrokontroler dengan komputer perangkat keras.

Tahap pertama yang dilakukan yaitu menginstal perangkat lunak Arduino IDE pada komputer. *Software* Arduino IDE ini bertindak sebagai media utama untuk menulis teks kode program (sketsa), mengedit, memverifikasi (*compiling*) sintaks bahasa C/C++, serta mengunggah logika kontrol ke dalam memori mikrokontroler. Setelah proses instalasi selesai, perangkat lunak dijalankan dan dilakukan konfigurasi awal dengan memilih jenis papan pengembang yang digunakan, yaitu Arduino Uno, serta menentukan alokasi *Port COM* komunikasi serial yang terdeteksi saat papan sirkuit dihubungkan via kabel *USB*.

Setelah Arduino IDE siap, langkah krusial berikutnya adalah menambahkan dan memasang *library* spesifik yang diperlukan ke dalam program. Keberadaan *library* ini sangat penting untuk menyederhanakan penulisan kode sintaks serta mempermudah komunikasi data antarmuka.

Pada sistem kendali alat penggulung kawat otomatis ini, digunakan tiga *library* pendukung utama, yaitu *Wire.h* yang digunakan untuk komunikasi I2C, *LiquidCrystal_I2C.h* yang digunakan untuk mengendalikan LCD I2C dan *Keypad.h* untuk membaca pemetaan matriks baris dan kolom pada komponen *Keypad 4x4*.

Proses pemasangan *library* dapat dilakukan dengan langkah berikut:

1. Membuka aplikasi Arduino IDE.
2. Memilih menu Sketch → Include Library → Manage Libraries.
3. Pada kolom pencarian, masukkan nama *library* yang dibutuhkan.
4. Pilih *library* yang sesuai kemudian klik tombol Install.
5. Tunggu hingga proses instalasi selesai.

Setelah proses instalasi *library* selesai, langkah berikutnya yaitu melakukan konfigurasi board yang digunakan. Pada penelitian ini digunakan Arduino Uno R3 sebagai mikrokontroler utama, sehingga Arduino IDE perlu diatur agar sesuai dengan board yang digunakan.

Langkah konfigurasi board dilakukan sebagai berikut:

1. Hubungkan Arduino Uno R3 ke komputer menggunakan kabel USB.
2. Pilih menu Tools → Board → Arduino AVR Boards → Arduino Uno.
3. Pilih menu Tools → Port, kemudian pilih port Arduino Uno yang terdeteksi pada komputer.

Setelah konfigurasi *board* selesai dilakukan, kode program dapat ditulis pada *text editor* Arduino IDE sesuai dengan kebutuhan algoritma sistem *Wire Coiller* otomatis yang dirancang. Sebelum program diunggah ke dalam memori mikrokontroler, dilakukan proses kompilasi dan verifikasi terlebih dahulu menggunakan tombol *Verify* (ikon centang) pada Arduino IDE. Proses ini bertujuan untuk memeriksa struktur sintaksis bahasa C/C++ serta memastikan tidak terdapat kesalahan penulisan program (*syntax error*) maupun alat pada pemanggilan *library* seperti *Keypad.h* , *Wire.h* dan *LiquidCrystal_I2C.h*.

Jika proses verifikasi berhasil dilakukan tanpa adanya pesan kesalahan (*Done compiling*), program dapat langsung diunggah ke memori *flash* Arduino Uno dengan menekan tombol *Upload* (ikon panah ke kanan) pada Arduino IDE. Selama proses pengunggahan berlangsung, data digital ditransfer dari komputer melalui kabel USB menuju chip ATmega328P. Setelah status pada perangkat lunak menunjukkan "*Done uploading*", Arduino Uno R3 akan secara otomatis melakukan *reset* internal dan langsung mengeksekusi logika program kontrol *Open Loop* yang telah dibuat, sehingga sistem dapat langsung bekerja menerima *Input* nilai resistansi dari operator.

Tahapan persiapan hingga pengunggahan pemrograman ini dilakukan secara sistematis untuk memastikan bahwa jalur komunikasi data antara perangkat lunak dan perangkat keras telah terhubung dengan sempurna. Dengan demikian, proses pengembangan, pengujian logika matematika kelipatan 5 ohm, hingga proses kalibrasi pergerakan mekanis rel penggeser dapat berjalan dengan lancar sekaligus meminimalkan risiko kegagalan sistem saat alat dioperasikan.

4.3.2 Pembuatan Program Arduino

Pada tahap ini, dilakukan proses pembuatan skrip program utama (*firmware*) yang berfungsi untuk mengendalikan seluruh sistem otomasi pergerakan *Open Loop* pada alat *Wire Coiller* otomatis. Kode program dibangun menggunakan perangkat lunak Arduino IDE dengan bahasa pemrograman Arduino berbasis C/C++ yang kemudian diunggah ke mikrokontroler Arduino Uno sebagai pusat kendali utama sistem. Pembuatan program ini dieksekusi secara terstruktur berdasarkan alur kerja diagram alir (*flowchart*) yang telah dirancang sebelumnya agar seluruh komponen elektrik dapat bekerja secara berurutan, responsif, dan terintegrasi dengan baik.

Arduino Uno bertindak sebagai otak mekanisasi yang bertugas membaca setiap enkripsi penekanan tombol dari pengguna, mengolah kalkulasi data yang diterima, serta memberikan instruksi biner kepada komponen *Output*.

Input pada sistem ini sepenuhnya berasal dari komponen matriks Keypad 4x4, sedangkan terminal keluaran (*Output*) terhubung pada LCD 16x2 I2C sebagai media monitoring visual dan modul *Driver stepper* A4988 yang bertugas mengalirkan pulsa daya ke kumparan motor *stepper* Nema 17.

Tahap awal penyusunan skrip program dilakukan dengan mendefinisikan pustaka (*library*) eksternal yang dibutuhkan untuk mendukung protokol komunikasi dan pemetaan data antar-komponen. Sintaks diawali dengan pemanggilan `#include <Wire.h>` untuk mengaktifkan jalur bus serial I2C, `#include <LiquidCrystal_I2C.h>` untuk menyederhanakan konfigurasi karakter pada layar LCD dengan alamat heksadesimal 0x27, serta `#include <Keypad.h>` untuk mengaktifkan pemetaan matriks karakter tombol tanpa membebani penggunaan pin digital mikrokontroler.

Setelah proses pemanggilan *library* dilakukan, tahap berikutnya adalah menentukan alokasi konfigurasi pin *Input* dan *Output* pada Arduino Uno R3. Jalur keluaran digital ditetapkan pada pin DIR (D2) dan STEP (D3) untuk mengendalikan arah putaran serta pulsa langkah pada *Driver* A4988, sementara pin D4 hingga D11 dikonfigurasi secara matriks berurutan (rowPins dan colPins) sebagai jalur pembacaan baris dan kolom pada *Keypad* 4x4.

Tahap selanjutnya yaitu melakukan deklarasi variabel kontrol dan memori internal program. Variabel digunakan untuk menyimpan parameter sistem yang dinamis selama proses berlangsung, seperti *Input* resistansi untuk menampung ketikan angka target dari operator, konstanta *Steps per revolution* = 3200 untuk acuan mode *microstepping* 1/16, serta variabel memori data terakhir, terakhir rotasi, dan boolean ada data terakhir yang berfungsi untuk menyimpan sekaligus menampilkan kembali riwayat spesifikasi koil nikelin yang sukses tergulung pada siklus sebelumnya.

Setelah proses deklarasi selesai dilakukan, program kemudian dibagi menjadi beberapa fungsi (*sub-routine*) spesifik untuk mempermudah pengelolaan program dan meningkatkan keterbacaan kode. Fungsi-fungsi tersebut didefinisikan secara terpisah di luar *loop* utama, yang meliputi

fungsi tampilanUtama() dan updateLCD() untuk manajemen visual, fungsi prosesResistensi() untuk pencocokan logika *switch-case*, fungsi jalankanMotorTerukur() untuk proses penggulangan otomatis dengan kurva kecepatan, fungsi putarTerusBerlawanan() untuk pengembalian posisi (*Unwinding/reset*), fungsi putarManualMaju() untuk pergerakan manual (*jogging*), fungsi tampilkanDataTerakhir() untuk memanggil memori, serta fungsi aksiStop() sebagai interupsi darurat.

Untuk mempermudah pemahaman terhadap struktur program yang dibuat, potongan program utama ditampilkan pada beberapa Gambar 4-7 berikut:

```

1  #include <Wire.h>
2  #include <LiquidCrystal_I2C.h>
3  #include <Keypad.h>
4
5  // =====
6  // 1. INISIASI & KONFIGURASI KOMPONEN
7  // =====
8
9  // --- LCD ---
10 LiquidCrystal_I2C lcd(0x27, 16, 2);
11
12 // --- Keypad ---
13 const byte ROWS = 4;
14 const byte COLS = 4;
15 char keys[ROWS][COLS] = {
16   {'1','2','3','A'},
17   {'4','5','6','B'},
18   {'7','8','9','C'},
19   {'*','0','#','D'}
20 };
21 byte rowPins[ROWS] = {8, 9, 10, 11};
22 byte colPins[COLS] = {4, 5, 6, 7};
23
24 Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
25 ...
26 // =====
27 // 2. PENGATURAN PIN & PARAMETER HARDWARE
28 // =====
29 const int dirPin = 2;
30 const int stepPin = 3;
31 const int stepsPerRevolution = 3200; // 1/16 Microstepping
32
33 // =====
34 // 3. VARIABEL KONTROL & MEMORI DATA
35 // =====
36 String inputOhm = "";
37 int terakhirOhm = 0;
38 int terakhirRotasi = 0;
39 bool adaDataTerakhir = false;

```

(a)

(b)

Gambar 4- 7 (a) Inisiasi dan Konfigurasi (b) Pengaturan pin dan Variabel Kontrol

Selanjutnya dilakukan penjelasan mengenai cara kerja program pada masing masing komponen yang digunakan pada sistem *Wire Coiller* otomatis.

4.3.3 Cara Kerja Program

Pada tahap ini, dijelaskan secara terperinci mengenai cara kerja program yang ditanamkan pada alat *Wire Coiller* otomatis berdasarkan fungsi-fungsi (*sub-routine*) yang telah disusun pada lingkungan Arduino IDE. Setiap fungsi memiliki tugas, algoritma, dan peran spesifik yang berbeda dalam mengendalikan rangkaian elektrik, sehingga seluruh komponen dapat mengeksekusi aksi mekanis maupun visual secara terintegrasi. Pembagian struktur program ke dalam beberapa fungsi terpisah ini dilakukan untuk

mempermudah proses pengelolaan memori mikro, meningkatkan keterbacaan kode (*code readability*), serta efisiensi pelacakan galat (*debugging*) apabila terjadi kesalahan logika pemrograman atau slip langkah pada sistem mekanik.

Program pada alat ini dirancang dengan prinsip kendali *Open Loop* untuk menjalankan instruksi secara runut dan sistematis. Alur kerja dimulai dari proses inisialisasi register I/O, penayangan menu siaga awal pada layar LCD, pembacaan masukan nilai resistansi kawat nikelin melalui *keypad*, pencocokan konstanta jumlah putaran, eksekusi penggulangan otomatis dengan profil kecepatan variabel, hingga fitur interaktif pengembalian posisi *jig* pengatur kawat ke titik awal (*reset position*). Berbeda dengan mesin penggulang konvensional yang mengandalkan sensor pembatas mekanis, program pada alat ini sepenuhnya mengandalkan perhitungan matematis jumlah pulsa langkah (*step count*) pada motor *stepper* Nema 17 untuk menentukan batas gerak translasi, sehingga sistem bekerja lebih ringkas.

Selain kendali otomatis, logika program juga dikonfigurasi untuk mengontrol pergerakan manual (*jogging* maju) yang responsif serta sistem pemantauan visual secara *real-time* melalui komunikasi bus I2C. Pada sub-bab ini, akan dibedah secara mendalam mengenai mekanisme kerja dari setiap fungsi utama yang tertanam di dalam *firmware* Arduino Uno, lengkap dengan potongan kode (*code snippet*) yang berkaitan langsung dengan fungsi tersebut. Penjelasan terstruktur ini disajikan untuk memberikan gambaran komprehensif mengenai arsitektur logika perangkat lunak, parameter waktu tunda pulsa mikrosekond (μs), serta korelasi fungsional antar-fungsi dalam mengendalikan keseluruhan ekosistem kerja alat *Wire Coiller* otomatis.

a. Fungsi Inisialisasi Sistem

Fungsi ini merupakan fungsi dasar yang dieksekusi tepat satu kali oleh mikrokontroler saat papan Arduino Uno R3 pertama kali menerima suplai daya elektrik atau sesaat setelah tombol *reset* fisik ditekan. Fungsi ini

mengemban peran vital untuk melakukan inisialisasi register seluruh komponen elektronik yang terintegrasi pada sistem *Open Loop*, sehingga setiap perangkat siap beroperasi sesuai dengan fungsi penugasan masing-masing. Di dalam blok kode ini, Arduino Uno R3 mengonfigurasi parameter komunikasi serial, mengaktifkan jalur protokol bus I2C, serta menetapkan mode kerja pin digital sebagai jalur keluaran (*Output*) maupun penentu kondisi awal aktuator sebelum mesin *Wire Coiller* otomatis digunakan secara penuh oleh operator.

Potongan kode program (*code snippet*) yang merealisasikan proses inisialisasi awal sistem ini seperti gambar berikut:

```

44 void setup() {
45   Serial.begin(9600);
46   lcd.init();
47   lcd.backlight();
48
49   pinMode(stepPin, OUTPUT);
50   pinMode(dirPin, OUTPUT);
51
52   // Inisialisasi awal arah putaran (default LOW)
53   digitalWrite(dirPin, LOW);
54
55   tampilanUtama();
56 }

```

Gambar 4- 8 Inisialisasi Sistem

Berdasarkan struktur kode di atas, perintah `Serial.begin(9600)` dijalankan untuk membuka jalur komunikasi serial data pada kecepatan *baud rate* 9600 bps antara mikrokontroler dengan komputer guna keperluan pemantauan data (*debugging*). Selanjutnya, fungsi `lcd.init()` dan `lcd.backlight()` dipanggil untuk mengaktifkan sirkuit kontroler internal LCD 16x2 melalui bus I2C sekaligus menyalakan lampu latar layar agar informasi visual dapat langsung terbaca dengan jelas oleh operator.

Tahapan inisialisasi kemudian dilanjutkan dengan menetapkan mode pin digital melalui sintaks `pinMode(stepPin, OUTPUT)` pada pin D3 untuk jalur pulsa langkah dan `pinMode(dirPin, OUTPUT)` pada pin D2 untuk kendali arah putaran *Driver A4988*. Sebagai langkah pengaman posisi awal (*default state*), perintah `digitalWrite(dirPin, LOW)` dieksekusi di akhir penyiapan pin guna memastikan arah putaran mekanis awal motor *stepper* Nema 17

langsung terkunci pada orientasi maju atau searah dengan penggulungan kawat nikelin (arah kanan). Setelah seluruh konfigurasi sirkuit selesai dilakukan, fungsi `setup()` akan menutup fasenya dengan memanggil fungsi `tampilanUtama()` untuk menampilkan menu siaga target resistansi pada layar LCD.

b. Fungsi Standby dan Navigasi

Fungsi ini merupakan kombinasi fungsi yang mengatur tata letak visual (*user interface*) pada layar LCD 16x2 berbasis I2C selama alat berada dalam kondisi siaga (*standby*) maupun saat menerima *Input* parameter. Kedua fungsi ini memiliki peran interaktif untuk menyajikan petunjuk operasional mesin serta memperbarui karakter angka yang ditekan oleh operator melalui *keypad* secara langsung (*real-time*). Pemisahan ke dalam dua fungsi visual ini bertujuan agar mikrokontroler Arduino Uno R3 tidak perlu terus-menerus melakukan pembersihan total layar (*clear screen*) yang dapat menyebabkan efek berkedip (*flickering*) pada layar LCD.

Potongan kode program yang merealisasikan logika penampilan menu siaga dan pembaruan karakter *Input* seperti Gambar 4-9 berikut:

```

108 void tampilanUtama() {
109     lcd.clear();
110     lcd.setCursor(0, 0);
111     lcd.print("Set Target:");
112     lcd.setCursor(0, 1);
113     lcd.print("30-120 Ohm");
114 }
115
116 void updateLCD() {
117     lcd.setCursor(0, 1);
118     lcd.print(" ");
119     lcd.setCursor(0, 1);
120     lcd.print("Target: ");
121     lcd.print(inputOhm);
122     lcd.print(" Ohm");
123 }

```

Gambar 4- 9 Standby dan Navigasi

Berdasarkan struktur kode di atas, fungsi `tampilanUtama()` bertindak sebagai tampilan menu awal (*default standby menu*). Melalui perintah `lcd.clear()`, seluruh karakter sebelumnya pada layar akan dibersihkan. Sintaks `lcd.setCursor(0, 0)` mengarahkan posisi penulisan ke baris pertama kolom pertama untuk menampilkan teks “Set Target:”, sedangkan

lcd.setCursor(0, 1) memosisikan teks “30-120 Ohm” pada baris kedua sebagai petunjuk batasan rentang nilai hambatan variabel yang diizinkan oleh sistem.

Sementara itu, fungsi updateLCD() akan dipanggil secara dinamis di dalam fungsi loop() utama setiap kali mikrokontroler mendeteksi adanya penekanan tombol angka pada *keypad*. Sesaat setelah karakter angka baru masuk ke dalam variabel *Input ohm*, fungsi ini terlebih dahulu mengirimkan karakter spasi kosong “ ” pada baris kedua untuk menghapus visualisasi petunjuk rentang lama tanpa perlu membersihkan seluruh layar. Selanjutnya, nilai angka yang sedang diketik oleh operator akan ditampilkan langsung melalui kombinasi perintah lcd.print(*InputOhm*). Mekanisme ini memastikan interaksi antara operator dan alat *Wire Coiller* otomatis ini dapat berjalan secara komunikatif, informatif, serta meminimalisir kesalahan *Input* parameter sebelum proses lilitan dimulai.

c. Fungsi Logika Konversi

Fungsi proses ini merupakan sub-rutin logika yang memegang peranan krusial dalam sistem kendali *Open Loop* alat ini, yaitu mentransformasikan parameter elektrik yang di *Input* oleh operator menjadi besaran mekanis. Fungsi ini bertugas memetakan nilai resistansi target kawat nikelin berdiameter 0,3 mm (dalam satuan ohm) menjadi jumlah rotasi fisik (*total revolution*) yang presisi pada poros motor *stepper* Nema 17 dan *lead screw* T8. Konversi ini didasarkan pada struktur pencabangan kondisi *switch-case* yang memuat pustaka data empiris, di mana setiap tingkat nilai hambatan kelipatan 5 ohm dalam rentang 30 ohm hingga 120 ohm telah dikorelasikan secara matematis dengan jumlah putaran yang dibutuhkan untuk menghasilkan panjang kumparan koil yang sesuai.

Potongan kode program (*code snippet*) yang merealisasikan proses konversi logis seperti Gambar 4-10 berikut:

<pre> 125 // --- Fungsi Pemrosesan & Logika Motor --- 126 void prosesResistansi(int ohm) { 127 int putaranMotor = 0; 128 129 switch (ohm) { 130 case 30: putaranMotor = 76; break; 131 case 35: putaranMotor = 89; break; 132 case 40: putaranMotor = 101; break; 133 case 45: putaranMotor = 114; break; 134 case 50: putaranMotor = 127; break; 135 case 55: putaranMotor = 139; break; 136 case 60: putaranMotor = 152; break; 137 case 65: putaranMotor = 165; break; 138 case 70: putaranMotor = 177; break; 139 case 75: putaranMotor = 190; break; 140 case 80: putaranMotor = 203; break; 141 case 85: putaranMotor = 215; break; 142 case 90: putaranMotor = 228; break; 143 case 95: putaranMotor = 241; break; 144 case 100: putaranMotor = 254; break; 145 case 105: putaranMotor = 266; break; 146 case 110: putaranMotor = 279; break; 147 case 115: putaranMotor = 292; break; 148 case 120: putaranMotor = 304; break; </pre>	<pre> 152 if (putaranMotor == -1) { 153 lcd.clear(); 154 lcd.setCursor(0, 0); 155 lcd.print("Error: Nilai Ohm"); 156 lcd.setCursor(0, 1); 157 lcd.print("Tidak Valid!"); 158 delay(2000); 159 return; 160 } 161 162 lcd.clear(); 163 lcd.setCursor(0, 0); 164 lcd.print("Target: "); lcd.print(ohm); lcd.print(" Ohm"); 165 lcd.setCursor(0, 1); 166 lcd.print("Motor: "); lcd.print(putaranMotor); lcd.print(" Rot"); 167 delay(1500); 168 169 jalankanMotorTerukur(putaranMotor, ohm); 170 } </pre>
---	---

(a)
(b)

Gambar 4- 10 (a) Input Resistansi (b) Logika Konversi

Berdasarkan struktur kode di atas, variabel integer ohm yang diterima dari *keypad* akan dievaluasi melalui pernyataan *switch* (ohm). Sebagai contoh, apabila operator memasukkan target nilai hambatan sebesar 30 ohm, maka program secara otomatis memilih kondisi case 30 dan memberikan nilai konstanta variabel putaranMotor sebanyak 76 rotasi. Apabila data *Input* tidak sesuai dengan salah satu daftar kelipatan 5 ohm yang tersedia di dalam pustaka tabel, alur kode program akan diarahkan menuju blok instruksi *default* yang memberikan nilai luaran putaranMotor = -1.

Kondisi *error* tersebut kemudian dideteksi oleh percabangan logika *if* (putaranMotor == -1), yang seketika menghentikan proses kelanjutan sirkuit kontrol dan menampilkan pesan peringatan interupsi berupa tulisan “*Error: Nilai Ohm Tidak Valid!*” pada panel LCD selama 2 detik sebelum kembali ke *standby*. Sebaliknya, jika data *Input* dinyatakan *valid*, layar LCD 16x2 akan menampilkan informasi target nilai hambatan beserta kalkulasi jumlah putaran mekanisnya selama 1,5 detik sebagai langkah konfirmasi visual. Pada tahap akhir, fungsi ini akan meneruskan parameter data putaranMotor dan ohm menuju fungsi inti jalankanMotorTerukur() untuk segera

mengeksekusi pulsa translasi mekanis penggulangan koil nikelin di atas rel penggeser.

d. Fungsi *Winding* Otomatis

Fungsi *winding* otomatis merupakan sub-rutin utama yang mengendalikan proses penggulangan kawat nikelin secara otomatis berdasar pada akumulasi pulsa terhitung (*step count*). Fungsi ini bekerja dengan mengonversi parameter jumlah rotasi total (*totalPutaran*) dan nilai resistansi (*nilaiOhm*) yang telah lolos dari proses validasi menjadi serangkaian pulsa digital berfrekuensi tinggi pada mode kendali *Open Loop*. Guna meminimalisir terjadinya gejala slip langkah (*lost Steps*) atau beban kejut (*shock load*) mekanis akibat kelembaman inersia saat motor mulai bergerak maupun saat akan berhenti, fungsi ini menerapkan algoritma kurva kecepatan trapesium (*trapezoidal speed profile*) dengan mengatur durasi waktu tunda mikrosekon (*delayMicroseconds*) secara dinamis.

Potongan kode program (*code snippet*) yang mengeksekusi algoritma kurva kecepatan dan monitoring proses penggulangan seperti Gambar 4-11 berikut:

<pre>// --- Fungsi Winding Otomatis (Akselerasi & Deselerasi) --- void jalankanMotorTerukur(int totalPutaran, int nilaiOhm) { digitalWrite(dirPin, LOW); // Arah Winding (Kanan) lcd.clear(); lcd.setCursor(0, 0); lcd.print("Target: "); lcd.print(nilaiOhm); lcd.print(" Ohm"); lcd.setCursor(0, 1); lcd.print("Rot: 0/"); lcd.print(totalPutaran); long totalStepsTarget = (long)totalPutaran * stepsPerRevolution; int putaranSaatIni = 0; // --- KONFIGURASI KECEPATAN --- const int startDelay = 1000; // Kecepatan awal lambat (delay bes const int targetDelay = 150; // Kecepatan konstan penuh (delay k int delayKecepatan = startDelay; // Batas langkah berdasarkan permintaan (3 putaran = 9600 step) long batasAkselerasi = 9600; long titikDeselerasi = totalStepsTarget - 9600; // Pengaman: Jika total putaran ternyata sangat pendek (kurang dari if (totalStepsTarget <= 19200) { batasAkselerasi = totalStepsTarget / 2; } }</pre>	<pre>// 1. FASE AKSELERASI (3 Putaran Pertama) if (x < batasAkselerasi) { if (x % 11 == 0 && delayKecepatan > targetDelay) { delayKecepatan--; } } // 2. FASE DESELERASI (3 Putaran Terakhir) else if (x >= titikDeselerasi) { if (x % 11 == 0 && delayKecepatan < startDelay) { delayKecepatan++; } } // Eksekusi Pulsa Stepper digitalWrite(stepPin, HIGH); delayMicroseconds(delayKecepatan); digitalWrite(stepPin, LOW); delayMicroseconds(delayKecepatan); // Update progress LCD setiap kelipatan 5 rotasi if ((x + 1) % (stepsPerRevolution * 5) == 0) { putaranSaatIni += 5; lcd.setCursor(5, 1); lcd.print(putaranSaatIni); lcd.print("/"); lcd.print(totalPutaran); lcd.print(" "); }</pre>
(a)	(b)

Gambar 4- 11 (a) Widing Otomatis (b) Akselerasi Deselerasi

Untuk menjamin kelancaran mekanis rel penggeser dan menghindari fenomena slip langkah (*lost Steps*) akibat inersia beban, fungsi ini menerapkan algoritma kurva kecepatan trapesium (*trapezoidal velocity profile*). Profil kecepatan ini dibagi menjadi tiga fase logis, yaitu fase akselerasi (peningkatan kecepatan), fase konstan, dan fase deselerasi (penurunan kecepatan). Parameter batas akselerasi dan titik deselerasi dikunci sejauh 3 putaran mekanis di awal dan akhir lintasan, yang dikonversikan menjadi nilai 9.600 langkah pulsa ($\text{StepsPerRevolution} * 3$). Pada fase awal akselerasi ($x < \text{batasAkselerasi}$), waktu tunda pulsa (delayKecepatan) dipangkas secara linier satu demi satu setiap kelipatan 11 langkah dari kondisi lambat startDelay (1.000 μs) menuju batas kecepatan puncak targetDelay (150 μs). Sebaliknya, ketika pergerakan melampaui titik deselerasi, nilai tunda pulsa dinaikkan kembali secara bertahap menuju 1.000 μs guna menghentikan motor *stepper* Nema 17 dengan halus tanpa hentakan. Sebagai langkah antisipasi pengaman, jika target lilitan yang dimasukkan bernilai sangat pendek (≤ 6 putaran total atau 19.200 step), sistem secara otomatis membagi dua batas trapesium secara simetris melalui instruksi pengkondisian *if*.

Selain pengaturan kurva kecepatan, fungsi ini mengintegrasikan fitur interupsi darurat (*software emergency interrupt*) serta manajemen pembaruan data layar LCD secara berkala selama perulangan *for* berlangsung. Guna meminimalkan jeda waktu pembacaan (*scanning delay*), program memanfaatkan sirkuit pembuat sisa bagi modulo $x \% 200 = 0$ untuk memeriksa penekanan karakter pada *keypad* setiap kelipatan 200 langkah pulsa *stepper*. Jika pada sela-sela proses penggulangan operator menekan tombol 'D', fungsi akan segera menangkap instruksi tersebut sebagai sinyal berhenti darurat (*Instant Stop*), keluar dari perulangan secara paksa menggunakan perintah *return*, dan mengalihkan program ke fungsi *aksiStop()*. Untuk keperluan pemantauan (*monitoring*), sirkuit program juga dikonfigurasi memperbarui karakter *progress* putaran pada baris kedua LCD setiap kelipatan 5 rotasi penuh ($\text{StepsPerRevolution} * 5$). Di akhir

sirkuit kode, jika proses penggulangan berhasil terselesaikan secara utuh tanpa interupsi, data nilai resistansi dan jumlah rotasi tersebut akan disimpan ke dalam variabel memori internal terakhir ohm serta data terakhir rotasi sebagai arsip riwayat data produksi koil nikelin sebelumnya.

e. Fungsi *Unwinding*

Fungsi *Unwinding* merupakan sebuah fungsi khusus (*sub-routine*) yang dirancang untuk mengembalikan posisi mekanis *jig* pemandu kawat ke titik awal lintasan secara manual melalui perintah pembalikan arah putaran motor (*Unwinding/reset position*). Fungsi ini dieksekusi oleh mikrokontroler Arduino Uno R3 ketika pengguna menekan tombol karakter 'C' pada komponen *Keypad* 4x4 saat berada di menu utama. Karena sistem mekanik alat ini dirancang sebagai sistem kendali *Open Loop* yang tidak mengandalkan *sensor limit switch* sebagai pembatas fisik, fungsi penggulangan balik ini menjadi komponen krusial untuk melakukan pengaturan ulang lintasan secara fleksibel dan aman berdasarkan interupsi dari operator.

Potongan kode program yang merealisasikan fungsi penggulangan balik (*Unwinding*) seperti Gambar 4-12 berikut:

```
// --- Fungsi Unwinding ---
void putarTerusBerlawanan() {
    digitalWrite(dirPin, HIGH); // Arah Unwinding (Kiri)

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Unwinding Coil");
    lcd.setCursor(0, 1);
    lcd.print("D = STOP");

    int delayKecepatan = 1000;
    const int targetDelay = 150;
    long stepCount = 0;

    while (true) {
        if (stepCount % 200 == 0) {
            char stopKey = keypad.getKey();
            if (stopKey == 'D') {
                aksiStop();
                break;
            }
        }
    }
}
```

Gambar 4- 12 Arah Unwinding

Berdasarkan struktur kode di atas, langkah pertama yang dieksekusi oleh Arduino Uno R3 saat fungsi ini aktif adalah mengubah kondisi logika dirPin (pin D2) menjadi *HIGH*. Perubahan status logika ini menginstruksikan *Driver A4988* untuk membalikkan fasa polaritas arus pada kumparan motor *stepper* Nema 17, sehingga poros motor dan *lead screw* T8 berputar berlawanan arah dari proses penggulungan otomatis. Secara bersamaan, layar LCD 16x2 I2C dibersihkan untuk menampilkan visualisasi "*Unwinding Coil*" di baris pertama dan petunjuk instruksi "*D = STOP*" di baris kedua sebagai umpan balik interaktif kepada operator.

Untuk menjaga kestabilan rel penggeser dari beban hentakan akibat pembalikan arah secara mendadak, fungsi ini menerapkan algoritma akselerasi halus (*soft start*) berbasis pelonjakan waktu tunda pulsa digital. Saat perulangan konstan *while (true)* dimulai, variabel *delay* kecepatan diinisialisasi pada angka lambat sebesar 1.000 μ s. Selama akumulasi variabel *step count* belum mencapai batas 3 putaran awal (setara 9.600 langkah pulsa), nilai tunda mikrosekon tersebut akan dikurangi satu unit secara bertahap setiap kelipatan 11 *step* hingga mencapai batas kecepatan konstan penuh (*targetDelay*) pada angka 150 μ s.

Karena sistem ini bekerja secara *Open Loop* tanpa menggunakan sensor *limit switch*, motor akan dipaksa berputar terus-menerus memutar *lead screw* untuk memundurkan *jig*. Oleh sebab itu, di dalam struktur perulangan ditanamkan pemeriksaan kondisi tombol interupsi darurat secara berkala. Setiap kelipatan 200 *step* pulsa berjalan, Arduino akan membaca kondisi sirkuit *keypad* melalui perintah *keypad.getKey()*. Apabila operator menekan tombol fungsi 'D', program akan langsung memicu fungsi interupsi *aksiStop()* dan mengeksekusi perintah *break* untuk keluar dari perulangan, sehingga putaran motor *stepper* Nema 17 dapat dihentikan seketika demi keamanan mekanis mesin.

f. Fungsi *Jogging* Manual

Fungsi ini merupakan sebuah sub-rutin (*sub-routine*) interaktif yang dirancang untuk memfasilitasi operator dalam melakukan penyesuaian

posisi mekanis rel penggeser secara manual (jogging maju/ke arah kanan). Fitur ini sangat krusial dalam sistem kontrol *Open Loop* untuk memosisikan jig pemandu kawat nikelin pada titik awal penggulangan yang ideal sebelum proses fabrikasi koil otomatis dieksekusi. Fungsi ini bekerja dengan memanfaatkan metode interupsi penahanan tombol melalui pustaka Keypad.h, di mana pergerakan aktuator tunggal motor *stepper* Nema 17 akan sepenuhnya bergantung pada durasi penekanan tombol 'B' oleh operator.

Potongan kode program (*code snippet*) yang mengatur mekanisme kerja *jogging* manual seperti Gambar 4-13 berikut:

```
void putarManualMaju() {
    digitalWrite(dirPin, LOW); // Arah Maju / Winding (Kanan)

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Jogging Maju...");
    lcd.setCursor(0, 1);
    lcd.print("Lepas B = STOP");

    int delayKecepatan = 1000;
    const int targetDelay = 250;
    long stepCount = 0;

    while (keypad.getState() == HOLD) {
        if (stepCount < 9600) {
            if (stepCount % 11 == 0 && delayKecepatan > targetDelay) {
                delayKecepatan--;
            }
            stepCount++;
        }
    }

    digitalWrite(stepPin, HIGH);
    delayMicroseconds(delayKecepatan);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(delayKecepatan);
}
```

Gambar 4- 13 Jogging Manual (Maju)

Berdasarkan struktur kode di atas, ketika fungsi ini dipanggil melalui penahanan tombol 'B', mikrokontroler pertama-tama akan mengunci arah putaran motor dengan memberikan logika *LOW* pada dirPin (D2) untuk mengarahkan pergerakan ke sisi kanan lintasan rel aluminium. Layar LCD 16x2 I2C kemudian akan memperbarui tampilannya secara *real-time* untuk memberikan umpan balik visual kepada pengguna berupa status “Jogging Maju... Lepas B = *STOP*”.

Inti dari pengendalian manual ini terletak pada penggunaan struktur perulangan *while* (keypad.getState() == HOLD). Selama operator terus menahan tombol 'B', program akan terjebak di dalam kalang (*loop*) tersebut

untuk terus membangkitkan pulsa digital *HIGH* dan *LOW* secara bergantian pada stepPin (D3). Di dalam perulangan ini juga disematkan algoritma akselerasi halus bersyarat if ($\text{stepCount} < 9600$) yang berfungsi menurunkan nilai waktu tunda mikrosekond (delayMicroseconds) dari $1.000\ \mu\text{s}$ ke $250\ \mu\text{s}$ secara bertahap. Pembatasan nilai kecepatan $\text{targetDelay} = 250\ \mu\text{s}$ ini sengaja dibuat lebih lambat daripada mode otomatis demi menjaga faktor keamanan mekanis saat penyesuaian manual dilakukan.

Sintaks `keypad.getKeys()` ditempatkan secara strategis di dalam perulangan *while* untuk terus memperbarui status pemetaan matriks *keypad*. Sesaat setelah operator melepaskan tekanan pada tombol 'B', kondisi syarat *HOLD* menjadi tidak terpenuhi, sehingga program secara instan keluar dari perulangan dan menghentikan pengiriman pulsa ke *Driver A4988* seketika. Alur fungsi ini diakhiri dengan pembersihan layar LCD untuk menampilkan pesan “*Jogging Berhenti*” selama 1 detik sebelum sistem kembali dialihkan ke menu siaga utama.

g. Fungsi Penyimpanan Riwayat

Fungsi ini merupakan sebuah fungsi interaktif yang dirancang untuk memanggil kembali (*recall memory*) data hasil fabrikasi penggulangan koil nikelin yang sukses terbentuk pada siklus kerja sebelumnya. Fungsi ini berperan penting sebagai fitur efisiensi operasional, sehingga operator tidak perlu mengetik ulang parameter nilai resistansi jika ingin memeriksa atau memastikan spesifikasi koil terakhir yang telah diselesaikan oleh alat. Di dalam alur program utama, fungsi ini akan terpicu dan dieksekusi secara instan oleh mikrokontroler Arduino Uno R3 ketika operator menekan tombol karakter 'A' pada komponen *Keypad 4x4* saat sistem berada dalam menu siaga (*standby*).

Potongan kode program (*code snippet*) yang merealisasikan proses pembacaan memori data terakhir seperti Gambar 4-14 berikut:

```

void tampilkanDataTerakhir() {
    lcd.clear();
    if (adaDataTerakhir) {
        lcd.setCursor(0, 0);
        lcd.print("Koil Terakhir:");
        lcd.setCursor(0, 1);
        lcd.print(terakhirOhm); lcd.print("Ohm | ");
        lcd.print(terakhirRotasi); lcd.print("Rot");
    } else {
        lcd.setCursor(0, 0);
        lcd.print("Belum Ada Data");
        lcd.setCursor(0, 1);
        lcd.print("Koil Sebelumnya");
    }
    delay(4000);
}

```

Gambar 4- 14 Riwayat Data Koil Resistansi yang telah terbentuk

Berdasarkan struktur kode di atas, mekanisme kerja fungsi ini diawali dengan mengosongkan layar LCD menggunakan perintah `lcd.clear()`, yang kemudian dilanjutkan dengan pemeriksaan kondisi melalui instruksi percabangan `if (adaDataTerakhir)`. Variabel boolean `ada data terakhir` bertindak sebagai penanda logis; nilai variabel ini hanya akan berubah menjadi `true` apabila proses penggulangan otomatis pada fungsi `jalankanMotorTerukur()` telah berhasil diselesaikan secara penuh (100% tuntas tanpa interupsi tombol *stop*).

Jika kondisi terpenuhi (`true`), mikrokontroler akan membaca nilai yang tersimpan di dalam variabel memori terakhir ohm serta terakhir rotasi, lalu memvisualisasikannya ke layar LCD 16x2 dengan format informasi nilai hambatan (Ohm) dan akumulasi jumlah putaran mekanis (Rotasi). Sebaliknya, jika alat baru saja dinyalakan dan belum pernah melakukan proses penggulangan sama sekali, kondisi *default* variabel bernilai `false`, sehingga program akan mengalihkan instruksi ke blok `else` untuk menampilkan pesan “Belum Ada Data Koil Sebelumnya”. Di akhir fungsi, diberikan waktu tunda (*delay*) selama 4.000 milisekon (4 detik) agar operator dapat membaca informasi riwayat tersebut dengan jelas sebelum layar LCD secara otomatis dibersihkan kembali dan dialihkan menuju menu utama siaga.