RANCANG BANGUN LENGAN ROBOT TIPE ARTIKULASI 3 DERAJAT KEBEBASAN UNTUK SARANA PENDIDIKAN

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Memperoleh gelar Sarjana Terapan
Program Studi Teknologi Rekayasa Mekatronika

Disusun oleh:

1. Pilipus Kevin Putra Anggono
   NIM: 211113015

2. Franchino Abimanyu Annesia
   NIM: 211113023

FAKULTAS VOKASI
PROGRAM STUDI TEKNOLOGI REKAYASA MEKATRONIKA
UNIVERSITAS SANATA DHARMA
YOGYAKARTA
2025

TUGAS AKHIR

RANCANG BANGUN LENGAN ROBOT TIPE ARTIKULASI 3 DERAJAT KEBEBASAN UNTUK SARANA PENDIDIKAN

Disusun oleh:

1. Pilipus Kevin Putra Anggono
   NIM: 211113015
2. Franchino Abimanyu Annesia
   NIM: 211113023

Dosen Pembimbing,

Dr. Eng. Ir. Petrus Sutyasadi

9 Juli 2025

TUGAS AKHIR

RANCANG BANGUN LENGAN ROBOT TIPE ARTIKULASI 3 DERAJAT KEBEBASAN UNTUK SARANA PENDIDIKAN

Disusun oleh:

1. Pilipus Kevin Putra Anggono
   NIM: 211113015
2. Franchino Abimanyu Annesia
   NIM: 211113023

SUSUNAN DEWAN PENGUJI

JABATAN	NAMA LENGKAP	TANDA TANGAN
Ketua	Baskoro Latu Anurogo, S.Sn., M.Ds.	.....
Sekretaris	Ir. Eko Aris Budi Cahyono, M.Eng.	.....
Anggota	Dr. Eng. Ir. Petrus Sutyasadi.	.....

Yogyakarta, 9 Juli 2025
Fakultas Vokasi
Universitas Sanata Dharma
Dekan,

Ir. Bernadinus Sri Widodo, S.T., M.Eng.

PERNYATAAN KEASLIAN KARYA

Kami menyatakan dengan sesungguhnya bahwa skripsi yang kami tulis ini tidak memuat karya atau bagian karya orang lain, kecuali yang telah disebutkan dalam kutipan dan daftar pustaka dengan mengikuti ketentuan sebagaimana layaknya karya ilmiah.

Apabila di kemudian hari ditemukan indikasi plagiarisme dalam naskah ini, kami bersedia menanggung segala sanksi sesuai peraturan perundang-undangan yang berlaku.

Yogyakarta, 9 Juli 2025
Yang membuat pernyataan

Pilipus Kevin Putra Anggono

Franchino Abimanyu Annesia

LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPERLUAN AKADEMIS

Yang bertanda tangan di bawah ini, kami mahasiswa Universitas Sanata Dharma:

1. Nama : Pilipus Kevin Putra Anggono
   NIM : 211113015
2. Nama : Franchino Abimanyu Annesia
   NIM : 211113023

Demi perkembangan ilmu pengetahuan, kami memberikan kepada Perpustakaan Universitas Sanata Dharma karya ilmiah kami yang berjudul:

"RANCANG BANGUN LENGAN ROBOT TIPE ARTIKULASI 3 DERAJAT KEBEBASAN UNTUK SARANA PENDIDIKAN"

beserta perangkat yang diperlukan (bila ada). Dengan demikian kami memberikan hak kepada Perpustakaan Universitas Sanata Dharma baik untuk menyimpan, mengalihkan dalam bentuk media lain, mengolah dalam bentuk pangkalan data, mendistribusikan secara terbatas, dan mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa perlu meminta izin dari kami atau memberikan royalti kepada kami selama tetap mencantumkan nama kami sebagai penulis.

Demikian pernyataan ini kami buat dengan sebenarnya.

Dibuat di Yogyakarta
Pada tanggal: 9 Juli 2025
Yang menyatakan,

Pilipus Kevin Putra Anggono

Franchino Abimanyu Annesia

MOTTO

"Bekerja Secara Serius dan Tidak Serius itu Lelahnya Sama Tetapi Hasilnya Berbeda" – J.Sunyoto

KATA PENGANTAR

Puji dan syukur kami panjatkan kepada Tuhan Yang Maha Esa atas berkat dan rahmat-Nya sehingga kami dapat menyelesaikan tugas akhir yang berjudul "Rancang Bangun Lengan Robot Tipe Artikulasi 3 Derajat Kebebasan Untuk Sarana Pendidikan" dengan baik dan tepat waktu. Tugas akhir ini kami selesaikan sebagai salah satu syarat memperoleh gelar Sarjana Terapan Teknologi Rekayasa Mekatronika., Fakultas Vokasi, Universitas Sanata Dharma Yogyakarta.

Perkembangan teknologi robotika di Indonesia, khususnya dalam bidang pendidikan, masih menghadapi berbagai tantangan terutama terkait dengan keterbatasan akses terhadap peralatan pembelajaran yang memadai dan berbiaya terjangkau. Melalui Tugas Akhir ini, kami berupaya memberikan solusi dengan merancang dan membangun lengan robot artikulasi 3 Derajat Kebebasan yang dapat digunakan sebagai sarana pembelajaran robotika yang efektif dan ekonomis. Dalam penyusunan proposal ini, kami telah menerima banyak bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, pada kesempatan ini kami ingin menyampaikan ucapan terima kasih kepada:

1. Tuhan Yang Maha Esa
2. Orang tua dan keluarga kami yang senantiasa memberikan dukungan baik moral maupun material.
3. Ibu Ir Pippie Arbiyanti, M.Eng. selaku Ketua Program Studi Teknologi Rekayasa Mekatronika yang telah memberikan kesempatan dan dukungan dalam pengembangan proyek ini.
4. Bapak Dr. Eng. Ir. Petrus Sutyasadi selaku Dosen Pembimbing yang telah memberikan arahan, masukan, dan bimbingan selama proses penyusunan proposal ini.
5. Bapak Baskoro Latu Anurogo, S.Sn., M.Ds. selaku ketua penguji skripsi, dan Bapak Ir. Eko Aris Budi Cahyono, M.Eng., selaku dosen penguji skripsi yang telah meluangkan waktunya untuk memberikan arahan dalam penulisan skripsi dan pembuatan tugas akhir.

6. Seluruh dosen dan staf Program Studi Teknologi Rekayasa Mekatronika yang telah memberikan ilmu dan pengetahuan yang sangat berharga selama masa perkuliahan.
7. Teman-teman mahasiswa Program Studi Teknologi Rekayasa Mekatronika yang telah memberikan dukungan moral dan saran konstruktif.

Kami berharap pengembangan tugas akhir yang berupa robot artikulasi 3 derajat kebebasan ini dapat bermanfaat khususnya dalam konteks pendidikan. Segala masukan dan saran yang membangun sangat diharapkan dalam penyempurnaan tugas akhir ini.

Yogyakarta, 9 Juli 2025
Penulis

ABSTRAK

Perkembangan teknologi robotika semakin pesat dan menuntut tersedianya sarana pembelajaran yang praktis dan terjangkau di lingkungan pendidikan tinggi. Namun, keterbatasan fasilitas laboratorium dan mahalnya harga robot komersial menjadi kendala dalam memberikan pengalaman langsung kepada mahasiswa. Penelitian ini bertujuan untuk merancang dan membangun lengan robot artikulasi 3 Derajat Kebebasan dengan biaya yang lebih ekonomis namun tetap mampu memberikan akurasi dan *repeatability* yang memadai untuk kebutuhan pembelajaran robotika dasar.

Robot ini menggunakan motor stepper NEMA 17 sebagai aktuator utama, sensor *Hall effect* A3144 untuk mendeteksi posisi home, serta kerangka berbahan plastik PLA+ hasil 3D printing. Sistem dikendalikan menggunakan Arduino Mega 2560 Pro Mini yang menerima perintah berbasis *G-code* dan *M-code* melalui terminal serial. Hasil pengujian menunjukkan bahwa robot mampu menjalankan gerakan linier dengan baik dan memiliki *repeatability* yang stabil, dimana robot dapat kembali ke posisi yang sama dalam sepuluh kali percobaan dengan deviasi posisi yang sangat kecil.

Pengembangan ini diharapkan dapat menjadi alternatif platform pembelajaran robotika yang terjangkau, mudah dioperasikan, dan memungkinkan mahasiswa memperoleh pengalaman praktis secara langsung. Dengan harga yang ekonomis, diharapkan setiap mahasiswa dapat mengoperasikan robot secara mandiri, sehingga proses pembelajaran menjadi lebih interaktif dan efektif.

Kata Kunci: Lengan robot artikulasi 3 derajat kebebasan, platform pembelajaran robotika.

ABSTRACT

The development of robotics technology was increasingly rapid and demanded the availability of practical and affordable learning facilities in higher education environments. However, limited laboratory facilities and the high price of commercial robots were obstacles in providing direct experience to students. This study aimed to design and build a 3 Degrees of Freedom (3 DoF) articulated robotic arm at a more economical cost but still able to provide adequate accuracy and repeatability for basic robotics learning needs.

This robot used a NEMA 17 stepper motor as the main actuator, an A3144 Hall effect sensor to detect the home position, and a 3D printed PLA+ plastic frame. The system was controlled using an Arduino Mega 2560 that received G-code and M-code based commands via a serial terminal. The test results showed that the robot was able to perform linear movements well and had stable repeatability, where the robot could return to the same position in ten trials with very small position deviations.

This development was expected to be an alternative robotics learning platform that was affordable, easy to operate, and allowed students to gain direct practical experience. With an economical price, it was hoped that each student could operate the robot independently, so that the learning process became more interactive and effective.

Keyword: 3 degrees of freedom (3 DoF) articulated robotic arm, educational robotics platform

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Robot pada awal mulanya berasal dari kata Robota dalam bahasa Czech, yang mempunyai arti pekerja. Mulai menjadi popular ketika seorang penulis berbangsa Czech, Karl Capek, membuat pertunjukan dari lakon komedi yang ditulisnya pada tahun 1921 yang berjudul Rossum‟s Universal Robot [1]. Definisi awal dari robot dikatakan sebagai segala peralatan otomatis yang dibuat untuk menggantikan fungsi yang selama ini dilakukan oleh manusia. Namun dalam perkembangan selanjutnya, robot diartikan sebagai manipulator multi fungsional yang dapat diprogram, yang dengan pemrograman itu ditujukan untuk melakukan sesuatu tugas tertentu [2].

Pengembangan robot manipulator sebagai sarana pembelajaran telah banyak dilakukan dalam lingkungan akademis. Menurut Baso dan Adiputra (2017), penggunaan lengan robot artikulasi dalam pembelajaran dapat meningkatkan pemahaman konsep teoretis mahasiswa sebesar 35% dibandingkan dengan metode konvensional. Penelitian ini juga menggarisbawahi pentingnya pengalaman praktis dalam membangun pemahaman yang kuat tentang prinsip-prinsip robotika.

Hal ini menyebabkan munculnya berbagai upaya untuk mengembangkan alternatif berbiaya rendah dengan tetap menjaga mutu pendidikan [3]. mengidentifikasi beberapa permasalahan utama dalam pengembangan lengan robot berbiaya rendah di Indonesia, antara lain:

1.1.1 Perkembangan Robotika dalam Dunia Pendidikan

Perkembangan teknologi robotika saat ini telah menjadi salah satu pilar penting dalam dunia industri dan pendidikan. Kemampuan untuk memahami dan mengoperasikan robot menjadi kebutuhan yang semakin mendesak, khususnya di bidang teknik mekatronika. Salah satu jenis robot yang banyak

digunakan dalam pembelajaran adalah lengan robotik (robot manipulator) karena mampu merepresentasikan berbagai konsep dasar seperti kinematika, kontrol gerak, dan pemrograman.

Namun demikian, kendala utama dalam pengembangan pembelajaran berbasis robot adalah biaya tinggi dari perangkat robotik komersial yang umumnya digunakan di industri. Di banyak institusi pendidikan, keterbatasan anggaran menyebabkan mahasiswa tidak memiliki akses langsung ke perangkat robotik, sehingga pemahaman konsep menjadi kurang maksimal.

1.1.2 Permasalahan dalam Pengembangan Robot Artikulasi Berbiaya Rendah

Berbagai upaya terus dilakukan untuk mengembangkan lengan robot artikulasi berbiaya rendah tanpa mengorbankan mutu pendidikan. Beribe (2019) mengidentifikasi beberapa permasalahan utama dalam pengembangan robot artikulasi murah di Indonesia, antara lain:

a. Pemilihan Motor yang Tepat

Kesulitan dalam menemukan motor dengan rasio berat terhadap torsi yang optimal. Motor dengan torsi besar biasanya memiliki bobot yang berat sehingga memerlukan struktur pendukung yang kuat, sementara motor yang ringan sering kali memiliki daya torsi yang kurang.

b. Desain Rangka Mekanik

Tantangan dalam merancang rangka yang ringan, kuat, dan stabil**,** khususnya untuk lengan robot dengan jangkauan lebih dari 50 cm. Desain mekanik yang buruk dapat mempengaruhi akurasi dan daya tahan sistem.

c. Sistem Transmisi dan *Backlash*

Sistem transmisi seperti roda gigi sering kali menghadapi masalah *backlash*, yaitu celah gerak bebas yang dapat mengurangi akurasi posisi hingga 15%. *Backlash* ini berpotensi memperburuk *repeatability* lengan robot dalam aplikasi pembelajaran.

d. Kendali Motor dan Integrasi

Kendala dalam menciptakan sistem kendali motor yang presisi dan mudah diintegrasikan ke dalam kurikulum pembelajaran. Motor stepper menjadi solusi yang cukup ideal karena memiliki keseimbangan antara presisi dan kemudahan pemrograman.

1.1.3 Keterbatasan Sarana Praktikum di Perguruan Tinggi

Laboratorium robotika di Program Studi Teknologi Rekayasa Mekatronika menghadapi keterbatasan perangkat praktikum. Ketersediaan robot industri skala besar yang mahal membuat mahasiswa kesulitan untuk mendapatkan pengalaman langsung dalam pengoperasian robot. Harga robot komersial yang berkisar antara Rp 100 juta hingga Rp 500 juta per unit membuat pengadaan dalam jumlah besar menjadi tidak memungkinkan.

1.1.4 Solusi dan Relevansi Penelitian

Sebagai solusi terhadap permasalahan tersebut, penelitian ini mengusulkan pengembangan lengan robot artikulasi 3 Derajat Kebebasan berbasis motor stepper dan struktur hasil 3D *printing*. Pendekatan ini diyakini mampu menekan biaya namun tetap memberikan akurasi dan *repeatability* yang memadai untuk keperluan pembelajaran.

Selain itu, sistem ini didesain agar mudah dipahami dan dioperasikan oleh mahasiswa tanpa harus mempelajari rumus kinematika yang kompleks, sehingga mendorong pembelajaran praktis dan interaktif**.** Dengan biaya produksi yang rendah, diharapkan setiap mahasiswa dapat memiliki akses terhadap satu unit robot secara mandiri, selaras dengan tujuan meningkatkan efektivitas pembelajaran robotika.

Laboratorium robotika Program Studi Teknologi Rekayasa Mekatronika saat ini memiliki keterbatasan dalam hal platform pembelajaran praktis untuk konsep kinematika robot artikulasi, dengan hanya tersedia 2 unit robot industri berukuran besar yang tidak dapat diakses

secara individual oleh mahasiswa dan dengan biaya pengadaan robot artikulasi komersial berkisar antara Rp 100-500 juta per unit, sehingga tidak memungkinkan pengadaan dalam jumlah banyak.

Untuk mengatasi tantangan-tantangan di atas, penelitian ini mengusulkan solusi berupa perancangan lengan robot artikulasi 3 Derajat Kebebasan dengan menggunakan motor stepper sebagai penggerak utama, struktur kerangka berbasis material hasil 3D printing, dan sistem transmisi dengan timing belt untuk meminimalkan backlash. Pendekatan ini, seperti yang direkomendasikan oleh Widyanto dan Kusumah (2023), memungkinkan pengembangan platform pembelajaran yang efektif dengan biaya terjangkau di institusi pendidikan Indonesia.

Penelitian ini memiliki fokus pada pengembangan teknologi robotika. Keselarasan ini penting untuk memastikan keberlanjutan dan pendalaman riset di bidang robotika di lingkungan akademik, sekaligus memberikan kontribusi nyata terhadap *roadmap* penelitian Program Studi Teknologi Rekayasa Mekatronika (TRMK). Sinergi penelitian dengan fokus pengembangan program studi ini menciptakan ekosistem penelitian yang terintegrasi, dimana setiap riset tidak berdiri sendiri melainkan menjadi bagian dari rencana jangka panjang untuk penguatan kapabilitas institusi dalam bidang robotika.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, pengembangan lengan robot artikulasi 3 Derajat untuk sarana pendidikan menghadapi berbagai tantangan teknis yang perlu diselesaikan. Keterbatasan akses terhadap platform pembelajaran robotika yang terjangkau namun tetap memiliki kualitas memadai menjadi permasalahan utama di banyak institusi pendidikan di Indonesia. Melalui pendekatan reverse engineering dan optimasi desain, permasalahan dalam penelitian ini dapat dirumuskan sebagai berikut:

a. Bagaimana merancang dan membangun lengan robot artikulasi 3 Derajat Kebebasan dengan biaya terjangkau namun tetap memiliki tingkat akurasi dan *repeatability* yang memadai untuk pembelajaran robotika?

b. Bagaimana mengimplementasikan lengan robot artikulasi 3 Derajat Kebebasan sebagai platform pembelajaran yang efektif untuk mahasiswa mekatronika?

1.3 Tujuan

Penelitian ini memiliki beberapa tujuan utama yang difokuskan pada pengembangan lengan robot artikulasi *3 DoF* sebagai media pembelajaran yang efektif dan terjangkau. Tujuan tersebut dirumuskan sebagai berikut:

a. Menghasilkan lengan robot artikulasi 3 Derajat Kebebasan dengan biaya terjangkau, akurasi dan *repeatability* yang memadai untuk aplikasi pembelajaran robotika.

b. Mengembangkan sistem lengan robot sederhana dengan perintah *G-code* dan *M-code* sebagai sarana pembelajaran robotika di pendidikan tinggi.

1.4 Manfaat

Hasil dari penelitian ini diharapkan dapat memberikan manfaat langsung, baik untuk mahasiswa maupun institusi pendidikan dalam hal penguasaan teknologi robotika. Manfaat tersebut antara lain:

1.4.1 Bagi Mahasiswa

Memberikan sarana pembelajaran praktik dengan alat yang terjangkau dan berkualitas sehingga diharapkan mahasiswa dapat secara individual memakai alat.

1.4.2 Bagi Universitas

Menyediakan alat pembelajaran dengan kuantitas lebih banyak untuk mempermudah kegiatan belajar mengajar.

1.5 Batasan Masalah

Agar penelitian lebih terfokus dan terarah, maka ruang lingkup penelitian ini dibatasi pada beberapa poin berikut:

a. Lengan robot hanya dirancang memiliki 3 derajat kebebasan .

b. Sistem kendali tidak terhubung ke jaringan internet atau berbasis IoT, melainkan dikendalikan secara lokal via USB

c. Hanya menggunakan akuator motor stepper Nema 17

d. Robot ini tidak ditujukan untuk penggunaan industri, melainkan hanya sebagai alat bantu pembelajaran.

BAB II
TINJAUAN PUSTAKA

2.1 Lengan Robot Artikulasi 3 Derajat Kebebasan

Lengan robot artikulasi 3 derajat kebebasan merupakan sistem mekanik yang dirancang untuk meniru gerakan lengan manusia dengan tiga derajat kebebasan [4]. Sistem ini terdiri dari beberapa segmen yang dihubungkan oleh sendi, memungkinkan gerakan rotasi dan translasi. Menurut Pitowarno dan Endra (2015), struktur artikulasi memberikan keseimbangan optimal antara kompleksitas sistem dan fleksibilitas ruang kerja. Aplikasi lengan robot 3 derajat kebebasan sangat luas, meliputi industri manufaktur hingga pendidikan. [5] menekankan bahwa konfigurasi 3 derajat kebebasan sudah mencukupi untuk sebagian besar aplikasi *pick and place* dasar, sambil tetap menyediakan kompleksitas yang memadai untuk pembelajaran konsep-konsep fundamental robotika di lingkungan pendidikan Indonesia.

2.2 Komponen Penggerak dan Kontrol

Dalam kasus ini komponen penggerak dan kontrol yang dipilih adalah motor stepper sebagai penggerak utama karena kemampuannya memberikan gerakan presisi dan terkontrol [6]. Motor stepper merupakan pilihan optimal untuk aplikasi robotika pendidikan di Indonesia karena menawarkan presisi tinggi dengan harga terjangkau dan ketersediaan yang baik di pasar lokal [7]. Motor stepper memungkinkan lengan robot bergerak dengan akurasi tinggi, yang sangat penting untuk aplikasi yang membutuhkan ketelitian.

Komponen yang digunakan sebagai pengendali mengatur arus dan sinyal yang masuk ke motor adalah driver motor stepper. Mikrokontroler yang digunakan adalah Arduino Mega 2560 Pro mini yang berfungsi sebagai otak yang mengendalikan gerakan lengan robot. Berdasarkan [8], platform Arduino menawarkan keseimbangan optimal antara kemudahan penggunaan untuk tujuan

pendidikan dan kapabilitas yang memadai untuk mengendalikan sistem robotika kompleks dengan biaya rendah.

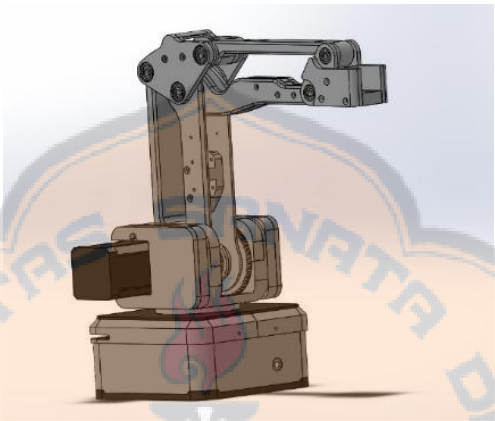
2.3 Optimasi Biaya dan Manufaktur

Salah satu tujuan utama proyek ini adalah meminimalisir biaya produksi. Hal ini dapat dicapai melalui pemilihan komponen yang ekonomis, penggunaan material yang tepat, dan optimasi proses manufaktur [9]. Penelitian Shane (2023) mendemonstrasikan bahwa penggunaan teknologi pencetakan 3D untuk komponen struktural robot dapat menurunkan biaya produksi hingga 70% dibandingkan dengan metode manufaktur konvensional di Indonesia, sambil tetap mempertahankan kekuatan struktural yang memadai. [10] merekomendasikan penggunaan *timing belt* sebagai alternatif ekonomis untuk sistem transmisi dengan *backlash* minimal, memberikan keseimbangan yang baik antara biaya dan performa untuk pengembangan robot pendidikan.

Dalam konteks minimalisir biaya, mengidentifikasi bahwa proses *reverse engineering* perlu mempertimbangkan ketersediaan komponen lokal dan kemampuan manufaktur domestik [11]. Mereka menemukan bahwa adaptasi desain untuk mengakomodasi keterbatasan ini dapat menghasilkan penghematan biaya hingga 45% tanpa mengorbankan fungsionalitas utama.

BAB III
METODE PENELITIAN

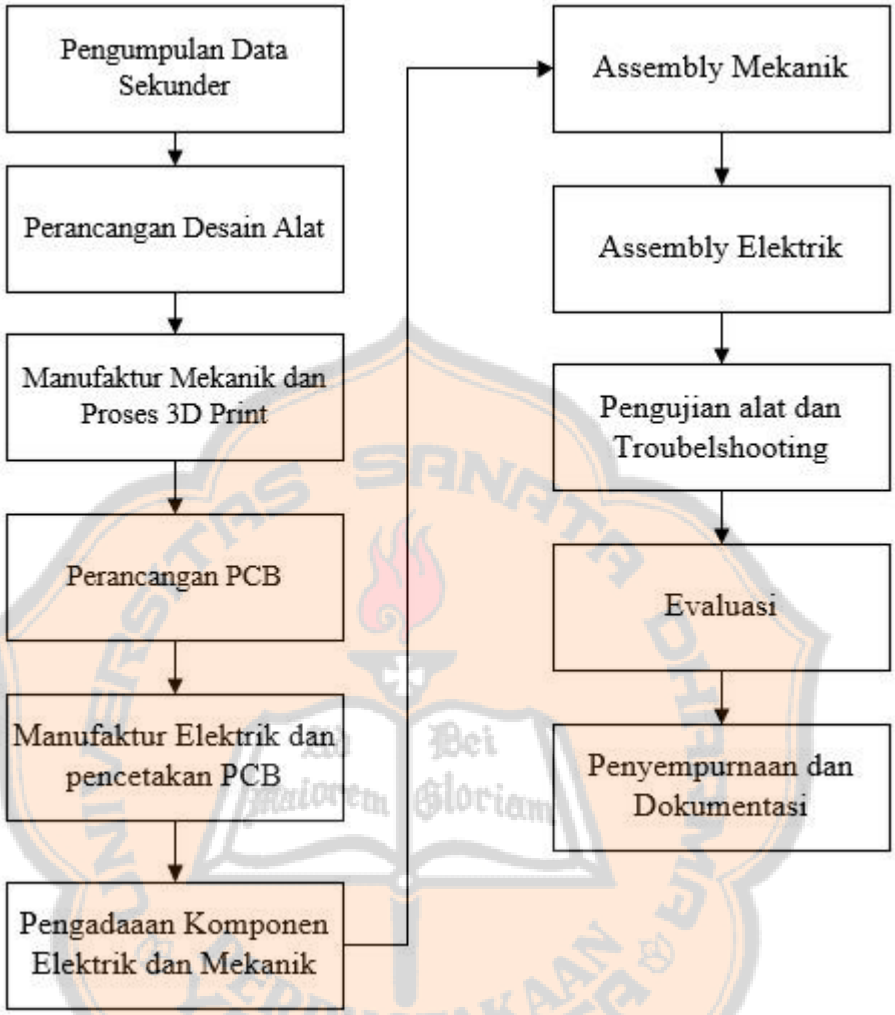
3.1 Deskripsi Alat



Gambar 3.1 Desain Alat Menggunakan *Solidwork*
Sumber: Dokumentasi penulis (2025)

Lengan Robot dengan artikulasi 3 derajat kebebasan yang menggunakan motor stepper adalah sistem robotik yang terdiri dari tiga sendi yang dapat bergerak secara independen, di mana setiap sendi digerakkan oleh motor stepper yang memberikan kontrol posisi presisi. Komponen utamanya meliputi *base* sebagai fondasi, tiga sendi rotasi yang dihubungkan oleh *link* dan digerakkan motor stepper (NEMA 17), *end-effector* di ujung lengan, sistem transmisi berupa timing belt, serta sistem kontrol berbasis mikrokontroler (Arduino Mega 2560 Pro Mini) dengan driver motor stepper DRV8825 yang mengatur gerakan *step-by-step* setiap motor.

3.2 Rencana Pelaksanaan



Gambar 3.2 Metode Rancang Bangun Lengan Robot
Sumber : Dokumentasi penulis (2025)

Secara lebih detail, Rancang Bangun Lengan Robot Tipe Artikulasi 3 Derajat Kebebasan Untuk Sarana Pendidikan adalah sebagai berikut:

A. Pengumpulan Data Sekunder

Tahap pertama dimulai dengan melakukan studi literatur dari berbagai sumber seperti buku, jurnal, dan referensi online yang relevan. Data ini digunakan untuk memahami konsep dasar lengan robot tipe artikulasi, pemilihan motor stepper, driver, serta penerapan robotika dalam dunia pendidikan.

B. Perancangan Desain Alat

Pada tahap ini dilakukan perancangan desain alat, baik desain mekanik maupun desain elektronik. Perangkat lunak seperti *SolidWorks* digunakan untuk membuat model 3D lengan robot dengan 3 derajat kebebasan. Perancangan juga mencakup sistem kontrol berbasis Arduino.

C. Manufaktur Mekanik dan Proses 3D Print

Bagian-bagian mekanik yang telah dirancang kemudian diproduksi menggunakan teknologi 3D printing. Bahan yang digunakan adalah PLA yang memiliki keunggulan ringan namun cukup kuat untuk kebutuhan pembelajaran.

D. Perancangan PCB

Merancang Printed Circuit Board (PCB) agar sistem kelistrikan lebih terintegrasi, rapi, dan memudahkan perakitan. Desain PCB disesuaikan dengan kebutuhan koneksi komponen elektronik. Dalam hal ini menggunakan aplikasi *easyeda.*

E. Manufaktur Elektrik dan Pencetakan PCB

Setelah desain PCB selesai, dilakukan pencetakan dan pemasangan komponen elektronik seperti driver motor, konektor, dan kabel ke dalam PCB sesuai skema yang telah dirancang.

F. Pengadaan Komponen Elektrik dan Mekanik

Melengkapi kebutuhan komponen seperti motor stepper NEMA 17, DRV8825, Arduino, serta komponen mekanik seperti bearing, puli, belt, dan material 3D print untuk mendukung proses perakitan.

G. Assembly Mekanik

Melakukan perakitan rangka mekanik berdasarkan desain 3D yang telah dicetak. Perakitan dilakukan dengan memperhatikan kekuatan struktur dan akurasi gerak.

H. Assembly Elektrik

Merakit sistem elektronik ke dalam struktur mekanik. Meliputi pemasangan

PCB, koneksi motor stepper ke driver, hingga penyambungan ke mikrokontroler.

I. Pengujian Alat dan Troubleshooting

Melakukan pengujian menyeluruh terhadap fungsi dasar lengan robot. Pergerakan tiap sumbu diperiksa apakah sesuai dengan perintah. Jika ditemukan kendala, dilakukan troubleshooting dan perbaikan.

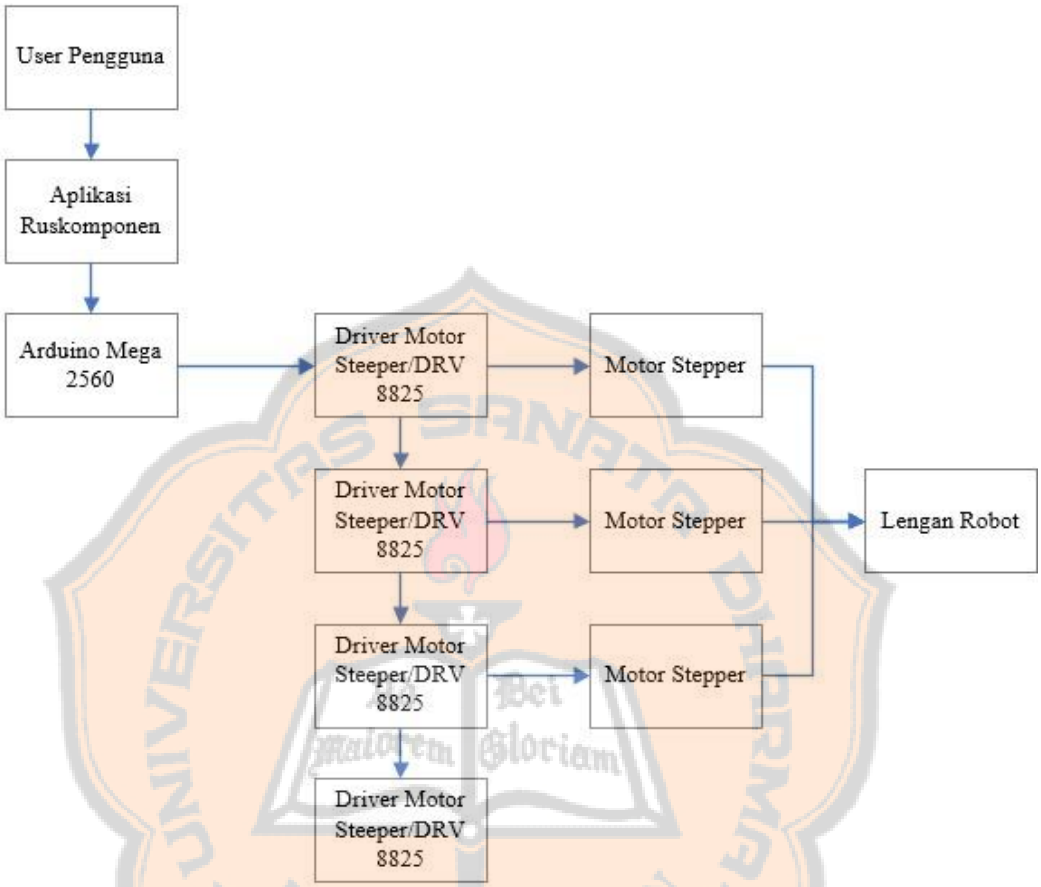
J. Evaluasi

Menilai kinerja robot baik dari segi akurasi, stabilitas, maupun kemudahan operasional. Evaluasi ini bertujuan untuk mengetahui apakah alat layak digunakan sebagai sarana pembelajaran.

K. Penyempurnaan dan Dokumentasi

Tahap terakhir yaitu melakukan perbaikan minor jika diperlukan dan menyusun dokumentasi lengkap mulai dari desain mekanik, diagram kelistrikan, *flowchart*, hingga panduan penggunaan agar alat siap diterapkan dalam lingkungan pendidikan.

3.3 Perancangan Perangkat Keras



Gambar 3.3 Rangkain Blok Diagram Sistem
Sumber: Dokumentasi penulis (2025)

Aplikasi Ruskomponen merupakan terminal komunikasi serial berbasis PC/laptop yang digunakan pengguna untuk mengirimkan perintah gerakan robot secara manual melalui USB.

Gambar 3.8 menunjukkan blok diagram sistem pengoperasian lengan robot dengan 3 derajat kebebasan. Input perintah dikirimkan melalui aplikasi Ruskomponen di komputer, kemudian diteruskan melalui komunikasi serial ke Arduino Mega 2560 Pro Mini. Arduino berperan sebagai pusat kendali yang memproses perintah gerakan dan mengatur sinyal ke masing-masing *driver* motor stepper. *Driver* motor kemudian mengendalikan motor stepper NEMA 17 sebagai

aktuator penggerak utama lengan robot, sehingga menghasilkan gerakan sesuai dengan instruksi pengguna. Fungsi dari blok diagram dijelaskan sebagai berikut :

1. Input

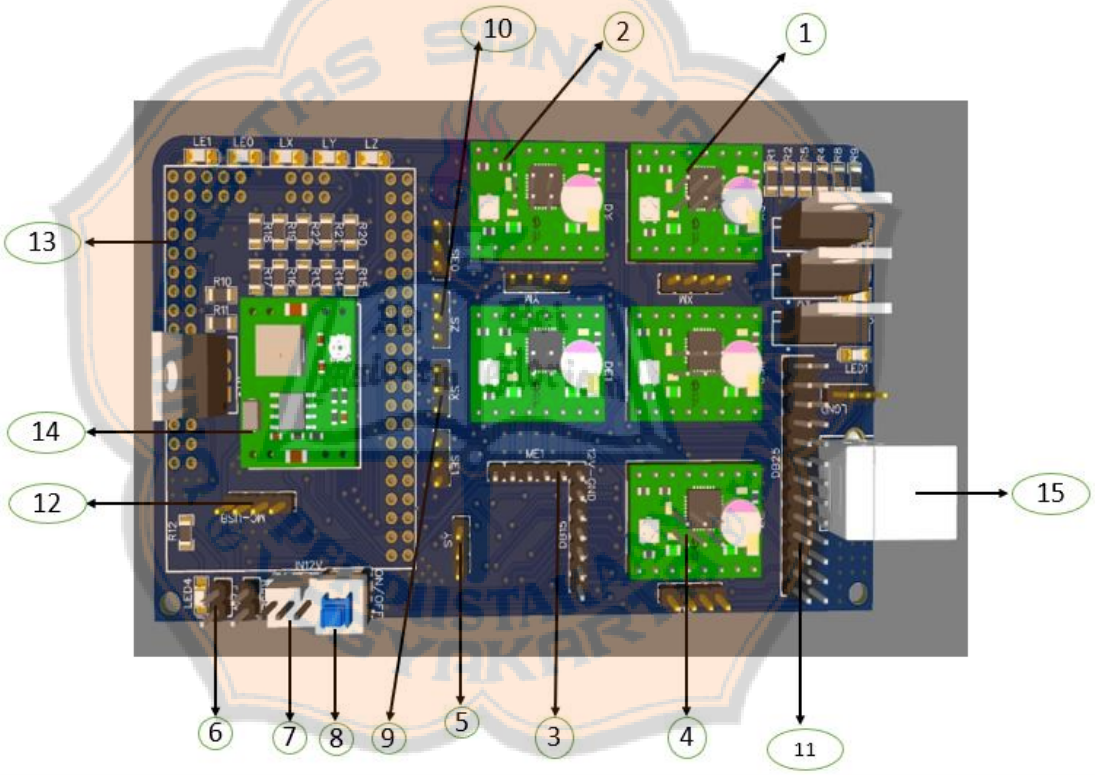
   Aplikasi Ruskomponen Dalam proyek lengan robot artikulasi 3 derajat kebebasan, aplikasi Ruskomponen digunakan sebagai alat komunikasi antara komputer dan Arduino Mega 2560 Pro Mini melalui port serial. Ruskomponen berfungsi untuk mengirim perintah ke robot, seperti mengatur sudut gerakan tiap motor stepper. Contohnya, perintah G0 X0.00 Y216.90 Z 138.00 bisa digunakan untuk mengatur posisi homing. Selain itu, Ruskomponen juga dapat menampilkan respon dari Arduino, sehingga memudahkan proses pengujian dan pemantauan sistem. Dibandingkan Serial Monitor bawaan Arduino, Ruskomponen lebih stabil dan fleksibel, sehingga sangat membantu dalam mengecek dan memperbaiki kesalahan saat pengembangan.

2. Proses
   a. Arduino Mega 2560 Pro Mini merupakan mikrokontroler utama yang menerima perintah dari aplikasi Ruskomponen melalui komunikasi serial. Arduino akan memproses data masukan dan mengatur sinyal kendali ke masing-masing aktuator (motor).
   b. Arduino menghasilkan sinyal STEP dan DIR yang dimaksud sinyal tersebut Pada Arduino, sinyal STEP (langkah) dan DIR (arah) digunakan untuk mengontrol pergerakan motor stepper melalui modul driver. Sinyal STEP digunakan untuk menggerakkan motor satu langkah pada satu waktu, dan sinyal DIR menentukan arah putaran motor, apakah searah jarum jam atau berlawanan arah jarum jam untuk mengatur arah dan jumlah langkah dari motor stepper sesuai dengan perintah yang diterima.

3. Output
   a. *Driver* Motor Stepper (DRV8825) berfungsi sebagai penguat sinyal dari Arduino untuk mengatur arus, arah, dan kecepatan motor stepper secara presisi.

b. Motor Stepper NEMA 17 sebagai penggerak utama pada tiap sumbu lengan robot, menghasilkan rotasi untuk membentuk gerakan pada masing-masing derajat kebebasan.
c. Lengan Robot RNV2 bergerak sebagai hasil akhir dari sistem, menunjukkan posisi dan orientasi sesuai dengan input pengguna melalui aplikasi Ruskomponen.

3.4 Rangkaian Elektrik



Gambar 3.4 Rangkaian Elektrik Menggunakan aplikasi *easyeda*
Sumber: Dokumentasi penulis (2025)

Gambar 3.4 tersebut menunjukkan urutan dan fungsi dari setiap komponen yang terhubung dalam sistem lengan robot artikulasi 3 derajat kebebasan. Masing-masing bagian memiliki peran penting dalam pengoperasian alat, mulai dari pengendalian motor, sensor posisi, hingga sistem koneksi dan pendinginan. Berikut penjelasan dari masing-masing komponen sesuai urutan nomor 1 hingga 11:

1. Pin *Driver* Motor Stepper X

   Menghubungkan sinyal dari Arduino ke *driver* motor stepper untuk menggerakkan sumbu X.

2. Pin *Driver* Motor *Stepper* Y

   Menghubungkan sinyal dari Arduino ke *driver* motor stepper untuk menggerakkan lengan Y.

3. Pin DB 15

   Konektor 15 pin yang digunakan untuk menghubungkan bagian elektronik end effector ke rangka robot secara rapi dan modular.

4. Pin *Driver* Motor *Stepper* Z

   Menghubungkan sinyal dari Arduino ke driver motor stepper untuk menggerakkan lengan Z .

5. Pin *Hall Effect* A3144 Y

   Sensor yang mendeteksi posisi awal motor stepper pada sumbu Y untuk menentukan titik awal gerakan

6. Pin *Fan*

   Kipas yang berfungsi untuk mendinginkan komponen elektronik agar suhu tetap stabil selama pengoperasian.

7. Pin *Power Supply*

   Titik koneksi antara adaptor 12V dengan sistem, untuk menyalurkan daya ke motor dan driver.

8. *Pin Push Button*

   Tombol tekan yang digunakan untuk memberi perintah tertentu, seperti mulai, berhenti, atau reset sistem.

9. Pin *Hall Effect* A3144 X

   Sensor yang mendeteksi posisi awal motor stepper pada lengan X untuk menentukan titik awal gerakan

10. Pin *Hall Effect* A3144 Z

    Sensor yang mendeteksi posisi awal motor stepper pada lengan Z untuk mengatur titik awal gerakan.

11. Pin DB 25

    Konektor 25 pin yang digunakan untuk pengembangan lebih lanjut semisal ingin ditambahkan rail pada robot.

12. Pin MC-USB

    Port USB pada Arduino untuk mengunggah program dan komunikasi data.

13. Pin Arduino Mega 2560 *Pro Mini*

    Papan mikrokontroler yang berfungsi sebagai pusat kendali sistem robot.

14. *Step-Down* (DC-DC Converter)

    Modul penurun tegangan dari 12V ke 5V untuk menyesuaikan kebutuhan komponen.

15. USB *Type* B

    Port USB untuk koneksi antara Arduino dan komputer saat pemrograman dan pengujian.

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

4.1 Sistem Lengan Robot Tipe Artikulasi 3 Derajat Kebebasan

Gambar 4. 1 Lengan Robot Tipe Artikulasi 3 Derajat Kebebasan
Sumber: Dokumentasi penulis (2025)

Sistem lengan robot yang dirancang merupakan bagian dari mesin berbasis mikrokontroler yang mampu menjalankan perintah berbasis *G-code* dan *M-code* untuk mengontrol pergerakan lengan secara presisi. Lengan robot ini terdiri dari tiga motor stepper NEMA 17 yang masing-masing berfungsi untuk:

1. Motor Sumbu Z (basis) di bagian bawah untuk gerakan rotasi dasar.
2. Motor Sumbu X (lengan bawah) untuk pergerakan maju-mundur atau jangkauan.
3. Motor Sumbu Y (lengan atas) untuk gerakan naik-turun vertikal.

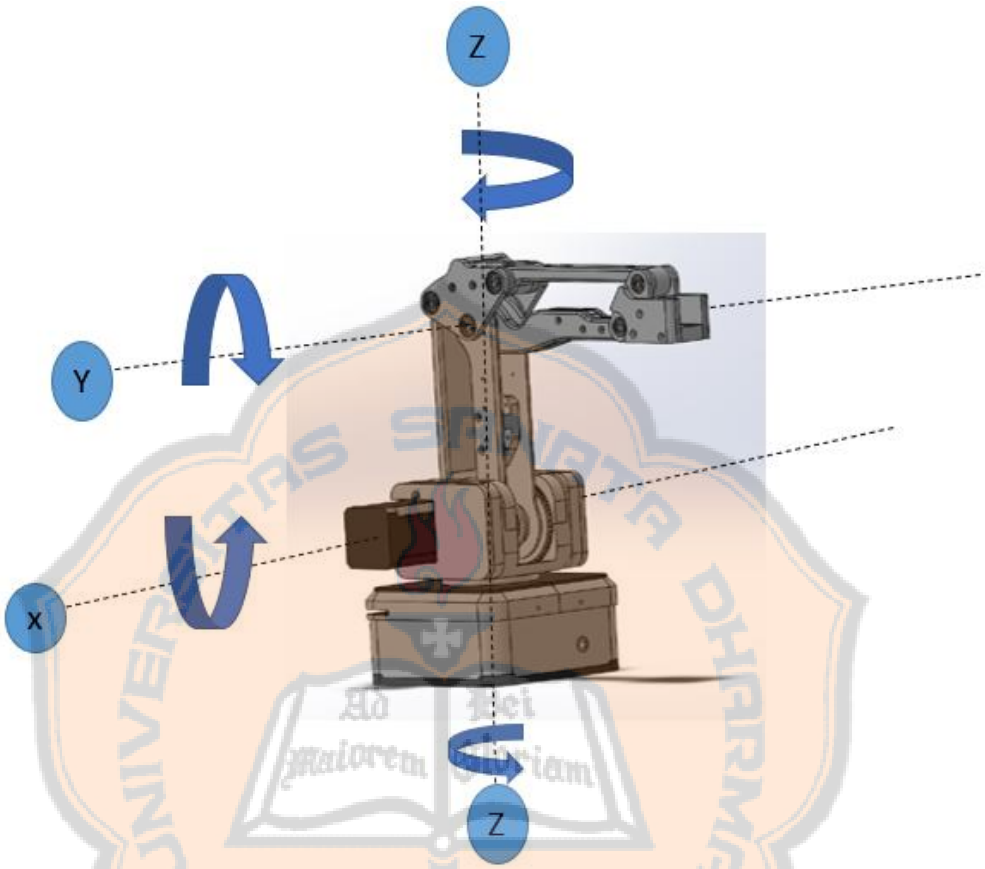
Setiap motor digerakkan menggunakan driver tipe DRV8825, yang disusun di atas PCB hasil rancangan sendiri, dan didinginkan dengan *heatsink* serta sistem

kipas aktif. Motor stepper dikalibrasi menggunakan perintah G28 (*homing*) yang memanfaatkan *sensor hall effect* A3144 untuk mendeteksi posisi nol (*home*). Magnet permanen dipasang pada roda gigi dengan posisi kutub utara yang bersentuhan dengan sensor, dan sensor akan mendeteksi keberadaan magnet tersebut untuk menentukan titik awal referensi.

Seluruh struktur mekanik dibuat menggunakan bahan PLA+ melalui proses pencetakan 3D print, sehingga robot tampak ringkas dan modular. Komponen utama dipasang di atas alas yang kokoh, dan kabel tertata melalui jalur manajemen kabel internal untuk keamanan dan estetika. Mikrokontroler Arduino Mega 2560 Pro Mini menjadi otak sistem, menjalankan *firmware* kustom yang mampu mengenali berbagai perintah standar industri seperti G0, G1, M17, M106, dan sebagainya. *Firmware* ini mendukung sistem koordinat Cartesian dan *homing* otomatis.

Sistem dikendalikan secara lokal melalui sambungan kabel USB ke komputer. Pengguna dapat memberikan perintah menggunakan aplikasi Ruskomponen untuk kontrol terprogram. Protokol komunikasi dilakukan melalui serial dengan *baud rate* 115200.

4.2 Desain Mekanik Lengan Robot



Gambar 4. 2 Sumbu pada robot
Sumber: Dokumentasi penulis (2025)

Desain mekanik dari lengan robot ini dirancang dengan mempertimbangkan kemudahan perakitan, kekokohan struktur, serta efisiensi ruang. Seluruh bagian utama lengan robot dicetak menggunakan material PLA+ dengan teknologi pencetakan 3D (3D *printing*), sehingga menghasilkan struktur yang ringan namun tetap kuat.

Rangka utama terdiri dari tiga bagian utama:

1. Bagian Basis (Sumbu Z)

   Merupakan bagian bawah yang berfungsi sebagai pusat rotasi. Bagian ini menampung motor stepper pertama yang berfungsi memutar seluruh lengan

robot. Di bagian bawah basis juga terdapat sensor Hall effect A3144 dan magnet untuk menentukan posisi awal (home).

2. Lengan Bawah (Sumbu Y)

   Terpasang di atas basis dan digerakkan oleh motor stepper kedua. Lengan ini bergerak secara vertikal (naik-turun) dan bertanggung jawab mengatur tinggi dari lengan atas. Mekanisme sambungan antar lengan robot menggunakan *bearing* (bantalan) yang dipasang pada titik engsel untuk meminimalkan gesekan saat rotasi. *Bearing* ini dipasangkan dengan poros (as) sebagai penghubung antar segmen lengan, memungkinkan gerakan yang halus dan stabil saat motor stepper menggerakkan sumbu-sumbu robot. Pemilihan *bearing* bertujuan agar sambungan kuat namun tetap fleksibel, sehingga mendukung akurasi dan ketahanan sistem dalam jangka panjang..

3. Lengan Atas (Sumbu X)

   Merupakan bagian terluar dari sistem dan bertugas memperluas jangkauan horizontal dari lengan robot. Digerakkan oleh motor stepper ketiga, lengan atas dapat bergerak maju-mundur. Sistem koneksi antar bagian dirancang agar dapat dibongkar pasang dengan mudah.

Untuk memastikan akurasi dan stabilitas gerakan, setiap motor dilengkapi dengan pulley dan roda gigi presisi, serta dipasangkan magnet permanen yang disusun sejajar dengan sensor A3144 untuk mendukung proses *homing* otomatis. Seluruh sistem mekanik dipasang di atas alas datar dan rata, dengan kabel motor dan sensor diatur sedemikian rupa melalui jalur kabel internal yang terintegrasi dalam desain cetakan 3D. Hal ini tidak hanya menjaga kerapihan, namun juga mengurangi potensi gangguan mekanis saat motor bergerak.

4.3 Pengujian dan Kalibrasi Sistem

Setelah proses perakitan lengan robot selesai, dilakukan tahap pengujian dan kalibrasi untuk memastikan bahwa setiap komponen sistem bekerja secara

optimal dan sesuai dengan fungsinya. Tahapan ini penting agar pergerakan robot akurat serta dapat merespon perintah dari pengguna dengan baik.

4.3.1 Pengujian Konektivitas dan Komunikasi Serial

Pengujian pertama dilakukan untuk memastikan bahwa komunikasi antara Arduino Mega 2560 dan komputer berjalan dengan baik. Aplikasi Ruskomponen digunakan untuk mengirimkan perintah berbasis *G-code* ke mikrokontroler. Saat pertama kali terhubung, sistem akan menampilkan pesan status seperti:

Gambar 4. 3 Robot terkoneksi dengan aplikasi ruskomponen
Sumber: Dokumentasi penulis (2025)

Hal ini menunjukkan bahwa sistem telah aktif dan siap menerima perintah kalibrasi.

4.3.2 Kalibrasi Motor Stepper dan Sensor *Hall Effect*

Setiap motor stepper (sumbu X, Y, dan Z) dikalibrasi dengan menempatkan sensor *Hall effect* A3144 di posisi awal (home) masing-masing sumbu. Magnet diletakkan pada bagian roda gigi, dan ketika magnet melewati sensor, sistem akan mengenali posisi home dan menghentikan gerakan motor. Kalibrasi dilakukan dengan perintah yang akan memerintahkan ketiga sumbu untuk bergerak ke posisi awal. Setelah magnet terdeteksi oleh sensor, sistem akan menampilkan:

Gambar 4. 4 Robot posisi *home* dengan aplikasi ruskomponen
Sumber: Dokumentasi penulis (2025)

4.3.3 Pengujian Gerak Linier G-code

Gerakan diuji menggunakan perintah G-code linier, misalnya:

Gambar 4.5 Robot posisi X20, Y216.90 dan Z155 dengan aplikasi ruskomponen
Sumber: Dokumentasi penulis (2025)

Yang memerintahkan lengan robot bergerak ke koordinat tertentu. Pengujian dilakukan bertahap pada masing-masing sumbu, dimulai dari pergerakan sumbu Z (basis), kemudian X (lengan bawah), dan Y (lengan atas).

Selama pengujian ini, beberapa kendala sempat terjadi, antara lain:

1. Salah urutan kabel koil motor stepper yang menyebabkan hanya bergetar namun tidak bergerak.
2. Driver terlalu panas karena setting arus potensio terlalu tinggi.
3. Satu atau lebih sensor *Hall effect* belum tepat posisinya sehingga sistem tidak bisa melakukan homing.

Setelah penyesuaian dilakukan (memastikan arus driver sesuai, pemasangan heatsink, dan kipas pendingin aktif), sistem dapat bergerak dengan lancar.

4.4 Integrasi Sensor *Hall Effect*

Gambar 4.6 Sensor *Hall*
Sumber : https://electricalbro.in/product/a3144-hall-effect-sensor/ (diakses pada 5 Juli 2025)

Sensor *Hall effect* A3144 digunakan pada sistem ini sebagai pendeteksi posisi awal (home) dari setiap sumbu pada lengan robot. Sensor ini bekerja berdasarkan perubahan medan magnet, di mana sensor akan aktif (logika low) saat mendeteksi kutub magnet yang sesuai (biasanya kutub utara).

Setiap sumbu (X, Y, Z) memiliki satu sensor *Hall effect* yang dipasang pada posisi tetap pada rangka, sedangkan sebuah magnet kecil dipasang pada bagian yang bergerak (biasanya roda gigi atau lengan). Saat bagian yang bergerak tersebut mencapai titik tertentu dan magnet berada tepat di depan sensor, maka sistem akan mengenali bahwa posisi tersebut adalah titik nol atau posisi *home.*

Untuk mendapatkan pembacaan yang stabil dan menghindari pembacaan palsu, sensor dikonfigurasi menggunakan *mode pull-up internal*, dimana pin output sensor dihubungkan ke pin input digital pada Arduino Mega 2560 Pro Mini, dan secara default berada pada logika *low*. Saat sensor aktif, output akan menjadi *high*, sehingga mudah dikenali oleh sistem.

Proses integrasi dilakukan melalui langkah-langkah berikut:

1. Penentuan Posisi Sensor dan Magnet

   Posisi sensor Hall diletakkan pada struktur rangka yang tetap, sementara magnet diletakkan pada bagian yang ikut berputar atau bergerak. Posisi ini ditentukan berdasarkan titik awal yang diinginkan sebagai referensi gerakan lengan robot.

2. Pengkabelan Sensor

   Setiap sensor Hall memiliki tiga kabel: VCC (biasanya 5V), GND, dan OUT. Kabel OUT dihubungkan ke input digital Arduino. Semua sensor mendapatkan suplai dari regulator 5V Arduino Mega 2560 Mini Pro.

3. Pembacaan Sinyal Sensor

   Program Arduino akan terus memantau status pin digital untuk mendeteksi apakah magnet sudah berada di depan sensor. Ketika kondisi ini terpenuhi, sistem akan menghentikan motor stepper yang bersangkutan dan menetapkan posisi saat itu sebagai titik nol (*zero position*).

4. Kesesuaian Kutub Magnet

   Magnet harus diletakkan dengan orientasi kutub yang sesuai agar dikenali oleh sisi aktif sensor. Dalam proyek ini, kutub utara magnet diletakkan menghadap langsung ke bagian datar sensor Hall.

Dengan adanya sensor *Hall effect* ini, proses kalibrasi menjadi lebih akurat dan otomatis, karena sistem dapat mengenali titik awal secara mandiri tanpa perlu input manual. Hal ini sangat penting untuk memastikan konsistensi gerakan robot terutama ketika sistem dimatikan dan dinyalakan kembali (*reset*).

4.5 Implementasi Sistem Pendingin

Gambar 4.7 *fan* 12V
Sumber : https://www.lazada.co.id/products/kipasfan-dc-8cm-x-8cm-12v-015a-i442022325.html (diakses pada 5 Juli 2025)

Sistem pendingin merupakan komponen penting dalam proyek lengan robotik ini untuk menjaga suhu kerja komponen elektronik tetap stabil, khususnya driver motor stepper, motor itu sendiri, dan Arduino Mega 2560 Pro Mini yang sensitif terhadap panas. Tanpa pendinginan yang memadai, suhu yang berlebih dapat menyebabkan penurunan performa, *overheat*, bahkan kerusakan permanen pada komponen.

4.5.1 Komponen Pendingin yang Digunakan

Sistem pendingin dalam proyek ini terdiri dari:

1. 3 buah kipas DC 12V:
    1) Dua kipas diletakkan langsung menghadap motor stepper sumbu X dan Y.
    2) Satu kipas diletakkan di dalam bagian base (bagian bawah rangka) untuk mendinginkan area PCB yang terdiri dari driver motor, Arduino, dan jalur distribusi daya.

2. Heatsink Aluminium:

    1) Dipasang pada bagian IC utama driver motor (DRV8825) untuk menyerap panas langsung dari permukaan *chip*.

3. Ventilasi Udara:

    1) Rangka bagian bawah yang tertutup rapat telah dilengkapi lubang ventilasi untuk sirkulasi udara masuk dan keluar agar panas tidak terperangkap.

4.5.2 Mekanisme Pengendalian Kipas

Kipas ada proyek ini awalnya dirancang dikendalikan secara otomatis melalui pin digital Arduino Mega 2560 dengan menggunakan MOSFET IRF540N sebagai saklar elektronik, dikendalikan melalui sebuah *library custom fan control*. Program dapat mengaktifkan kipas saat sistem aktif, dan menunda pematian kipas selama 30 detik setelah sistem dimatikan, untuk memastikan suhu sudah turun.

Namun karena suhu meningkat terlalu cepat terutama di area Arduino, maka kipas akhirnya dihubungkan langsung ke sumber daya 12V agar menyala secara otomatis setiap kali sistem diberi daya. Pendekatan ini lebih efektif untuk sistem sederhana dan mempercepat proses pendinginan saat sistem mulai bekerja.

4.5.3 Evaluasi Pendinginan

Setelah sistem pendingin diaktifkan:

1. Suhu driver motor menurun secara signifikan dibandingkan saat tanpa kipas.
2. Arduino Mega 2560 Mini Pro tidak lagi mengalami overheat dalam waktu singkat.
3. Motor stepper menjadi lebih stabil, terutama pada sumbu Z yang sebagai tumpuan utama.

Kondisi ini menunjukkan bahwa sistem pendingin sangat efektif dan krusial dalam keberhasilan pengoperasian sistem secara jangka panjang.

4.6 Evaluasi dan Kendala Sistem

Setelah seluruh komponen mekanik, elektronik, dan perangkat lunak selesai dirakit dan diuji, dilakukan evaluasi menyeluruh terhadap kinerja sistem lengan robotik. Evaluasi ini meliputi akurasi gerakan, respon terhadap perintah *G-code,* kestabilan motor, serta efektivitas sistem pendingin dan sensor.

4.6.1 Evaluasi Sistem

Beberapa poin positif dari hasil implementasi sistem adalah sebagai berikut:

1. Respons Perintah *G-code* dan *M-code*
   Sistem berhasil merespon perintah berbasis *G-code* dan *M-code* seperti M17 untuk mengaktifkan motor dan G28 untuk kalibrasi (homing) ke posisi awal. Perintah gerak seperti G0 X100 Y100 Z100 dapat dijalankan dengan presisi yang cukup baik setelah kalibrasi.
2. Sensor *Hall Effect* Bekerja Optimal
   Sensor *Hall Effect* A3144 yang dipasang di masing-masing sumbu mampu mendeteksi keberadaan magnet dengan akurat. Hal ini penting dalam proses homing awal agar robot mengetahui posisi nol (home) dari setiap sumbu.
3. Stabilitas dan Presisi Gerak
   Dengan menggunakan motor stepper NEMA 17 dan driver DRV8825, pergerakan robot relatif stabil dan tidak mengalami slip, terutama setelah pemasangan bearing dan penggunaan sistem sambungan berbasis as yang presisi.
4. Pendinginan Efektif
   Sistem pendingin dengan kipas dan heatsink yang dipasang

secara strategis mampu menjaga suhu Arduino dan driver tetap stabil, bahkan saat motor dalam kondisi mengunci (enabled) dalam waktu lama.

4.6.2 Kendala Mekanikal

1) Desain Mekanik dan Pencetakan 3D

Dalam proses perancangan lengan robot, seluruh komponen mekanik dirancang terlebih dahulu menggunakan perangkat lunak desain, yaitu *SolidWorks*. Setiap bagian seperti basis, sambungan lengan, dudukan motor, serta komponen penyangga lainnya dirancang dengan mengacu pada ukuran aktual motor, bearing, dan komponen elektronik yang digunakan. Proses pengukuran ini harus dilakukan secara teliti agar tidak terjadi ketidaksesuaian saat proses perakitan. Namun, dalam praktiknya terdapat beberapa kendala selama proses desain dan pencetakan

a) Ketidaksesuaian Ukuran Desain dan Fisik

Meskipun desain di SolidWorks sudah disesuaikan dengan ukuran datasheet komponen, terkadang hasil cetakan tidak sesuai karena toleransi pencetakan atau adanya perubahan dimensi akibat penyusutan bahan PLA+ setelah pencetakan. Hal ini mengharuskan beberapa bagian untuk dirombak atau disesuaikan ulang di SolidWorks.

b) Konversi Desain ke Format Pencetakan

Setelah desain selesai di *SolidWorks*, file diekspor ke format .STL untuk kemudian diolah menggunakan software *slicer* seperti *Ultimaker Cura*. Dalam proses ini, perlu pengaturan parameter seperti kecepatan, tinggi *layer, infill*, dan penggunaan support yang tepat. Kesalahan pengaturan dapat menyebabkan hasil cetakan cacat atau tidak presisi.

c) Kendala pada Proses 3D Printing

Proses pencetakan 3D juga mengalami berbagai kendala teknis, seperti nozzle yang tersumbat (*clogging*), filament yang tidak keluar stabil, serta kegagalan cetak di tengah proses. Hal ini menyebabkan beberapa komponen harus dicetak ulang hingga mendapatkan hasil yang optimal. Proses cetak yang memakan waktu lama juga menjadi tantangan tersendiri dalam menyelesaikan keseluruhan rangka lengan robot.

d) Kestabilan Permukaan dan Kekuatan Komponen

Beberapa bagian hasil cetakan memiliki permukaan yang kasar atau struktur yang kurang kuat karena arah layer atau infill yang tidak sesuai. Oleh karena itu, perlu dilakukan pengaturan desain ulang agar bagian-bagian penting memiliki ketebalan dan struktur pendukung yang cukup kuat.

Evaluasi dan perbaikan dilakukan secara bertahap hingga seluruh bagian dapat dirakit dan berfungsi sesuai kebutuhan. Proses ini merupakan bagian penting dalam tahapan rekayasa produk berbasis prototyping digital menggunakan teknologi pencetakan 3D.

2) Perakitan Mekanik dan Komponen Fisik

Setelah part hasil cetakan 3D selesai diproduksi, proses perakitan mekanik juga menghadapi sejumlah kendala teknis. Salah satu tantangan utama adalah menyambungkan berbagai komponen seperti baut, mur, dan bearing ke dalam part 3D print yang sudah dicetak. Terdapat beberapa masalah yang muncul:

a) Lubang Tidak Presisi untuk Baut dan Mur

Beberapa lubang yang didesain untuk baut M3 atau M4 ternyata mengalami penyusutan akibat proses pencetakan, sehingga baut sulit masuk atau terlalu longgar. Hal ini memerlukan pembesaran atau pengeboran ulang lubang secara manual.

b) Pemasangan Mur Tidak Tertanam Sempurna

Lubang persegi untuk menanam mur (nut trap) kadang tidak cukup dalam atau sedikit melenceng posisi, menyebabkan mur mudah lepas saat dikencangkan. Solusi yang digunakan adalah menambahkan lem super glue atau memanaskan mur sedikit agar tertanam dengan baik ke dalam plastik.

c) Bearing Sulit Masuk karena Toleransi Ketat

Beberapa lubang untuk bearing tidak cukup besar karena terjadi penyusutan plastik saat mendingin. Proses penyisipan bearing menjadi sulit dan berisiko merusak *part*. Untuk mengatasi hal ini, dilakukan penghalusan manual pada lubang menggunakan cutter atau amplas hingga bearing dapat masuk dengan pas dan tetap kuat.

d) Penyesuaian Poros (As) terhadap Bearing dan Engsel

Setelah bearing terpasang, penyambungan as atau poros juga perlu akurasi tinggi. Jika terlalu longgar, as bisa bergeser saat robot bergerak. Jika terlalu sempit, as tidak bisa dipasang tanpa merusak bearing. Beberapa as harus diasah sedikit menggunakan amplas untuk mendapatkan toleransi yang sesuai.

Dengan berbagai penyesuaian manual ini, perakitan part mekanik dari hasil 3D print akhirnya dapat diselesaikan. Namun, waktu dan tenaga tambahan cukup banyak digunakan dibandingkan jika menggunakan part hasil manufaktur presisi.

4.6.2 Kendala Selama Implementasi

Beberapa kendala dan tantangan teknis yang ditemui selama proses pengembangan dan pengujian antara lain:

1. Driver Panas Berlebih

   Driver DRV8825 untuk motor stepper sumbu Y sempat mengalami overheat meskipun sistem sudah menggunakan pendingin. Hal ini disebabkan oleh pengaturan arus (current

limit) yang terlalu tinggi di potensiometer driver. Solusi: dilakukan penyetelan ulang arus serta penambahan pendingin.

2. Kesalahan *Wiring* Motor

   Pada awalnya beberapa motor tidak bergerak dan hanya bergetar. Setelah ditelusuri, hal ini disebabkan karena urutan kabel koil motor yang terbalik. Solusi: pengecekan ulang dokumentasi motor dan penyambungan kabel sesuai pasangan koil yang benar.

3. *Homing* Tidak Terbaca

   Posisi sensor *Hall Effect* yang tidak tepat menyebabkan proses homing gagal. Penyelesaian dilakukan dengan menyesuaikan posisi magnet agar tepat melewati sensor saat mencapai titik nol masing-masing sumbu.

4. *Fan* Tidak Menyala Saat Awal

   Pada desain awal, *fan* dikontrol oleh Arduino dan kadang tidak aktif saat sistem diberi daya. Solusi akhir adalah menyambungkan langsung *fan* ke supply 12V untuk memastikan menyala setiap sistem aktif.

4.7 Pengujian dan pembahasan

4.7.1 Analisis Akurasi, Presisi, dan Repeatability

Dalam sistem robotika, khususnya pada lengan robot artikulasi 3 derajat kebebasan yang dikembangkan dalam proyek ini, akurasi, presisi, dan *repeatability* merupakan tiga parameter kunci yang menentukan seberapa baik robot dapat menjalankan tugas-tugasnya sesuai dengan perintah yang diberikan. Ketiga aspek ini juga sangat penting dalam dunia industri, terutama ketika robot digunakan untuk aplikasi yang memerlukan ketelitian tinggi seperti perakitan, pemindahan material, atau penempatan komponen.

Repeatability atau dalam bahasa Indonesia dikenal sebagai kemampuan pengulangan, merupakan salah satu parameter penting dalam evaluasi performa sistem robotik. *Repeatability* mengacu pada sejauh mana sebuah lengan robot mampu kembali ke posisi yang sama secara konsisten setelah melakukan gerakan berulang dari titik awal yang identik. Dalam konteks proyek ini, *repeatability* menjadi aspek yang krusial untuk memastikan bahwa sistem lengan robot artikulasi 3 Derajat Kebebasan yang dikembangkan mampu menjalankan tugas-tugas pengulangan dengan tingkat presisi yang memadai. Pengujian *repeatability* dilakukan untuk mengetahui apakah lengan robot dapat bergerak dari posisi awal (home) menuju posisi yang sama sebanyak 10 kali percobaan berturut-turut, dan apakah posisi akhir yang dicapai tetap konsisten. Evaluasi terhadap *repeatability* ini tidak hanya mencerminkan stabilitas gerakan sistem, tetapi juga menjadi salah satu indikator bahwa robot ini layak diterapkan sebagai sarana pembelajaran dalam dunia pendidikan, di mana akurasi dan keandalan sangat diperlukan.

Tabel 4.1 Hasil Pengujian *Repeatability* Lengan Robot Artikulasi 3 DoF

No	Posisi Awal (Home)	Perintah Posisi Tujuan (X Y Z)	Keterangan
1	(X 00 Y 216.90 Z 138)	X-4.00 Y202 Z185	Sesuai
2	(X 00 Y 216.90 Z 138)	X-4.00 Y202 Z185	Sesuai
3	(X 00 Y 216.90 Z 138)	X-4.00 Y202 Z185	Sesuai
4	(X 00 Y 216.90 Z 138)	X-4.00 Y202 Z185	Sesuai
5	(X 00 Y 216.90 Z 138)	X-4.00 Y202 Z185	Sesuai
6	(X 00 Y 216.90 Z 138)	X-4.00 Y202 Z185	Sesuai
7	(X 00 Y 216.90 Z 138)	X-4.00 Y202 Z185	Sesuai
8	(X 00 Y 216.90 Z 138)	X-4.00 Y202 Z185	Sesuai
9	(X 00 Y 216.90 Z 138)	X-4.00 Y202 Z185	Sesuai
10	(X 00 Y 216.90 Z 138)	X-4.00 Y202 Z185	Sesuai

4.7.2 Pembahasan

Pengujian *repeatability* dilakukan untuk mengevaluasi kemampuan lengan robot 3 Derajat Kebebasan dalam mencapai posisi akhir yang sama secara berulang-ulang, dengan titik *home* (X 00 Y 216.90 Z 138) dan perintah gerakan yang identik. Uji coba ini penting untuk menilai kestabilan sistem, akurasi posisi, serta konsistensi gerakan, yang merupakan aspek fundamental dalam aplikasi robotika, khususnya untuk keperluan pembelajaran.

Dalam pengujian ini, robot diperintahkan untuk bergerak menuju posisi X-4.00 Y202 Z185 yang telah ditentukan sebanyak 10 kali percobaan berturut-turut. Hasil pengamatan menunjukkan bahwa lengan robot mampu mencapai posisi akhir yang konsisten dan sesuai dengan perintah *G-code* yang diberikan pada setiap siklus percobaan. Hal ini menunjukkan bahwa sistem memiliki tingkat *repeatability* yang memadai untuk mendukung kegiatan praktikum robotika di lingkungan pendidikan.

Secara keseluruhan, pengujian ini membuktikan bahwa desain lengan robot yang dikembangkan mampu memberikan performa *repeatability* yang baik, stabil, dan cukup akurat untuk memperkenalkan konsep dasar kinematika dan kontrol robot kepada mahasiswa dengan cara yang praktis dan terjangkau.

Selain aspek teknis, salah satu tujuan utama dalam perancangan lengan robot artikulasi 3 Derajat Kebebasan ini adalah menghasilkan platform pembelajaran dengan biaya yang terjangkau namun tetap fungsional dan relevan untuk kebutuhan edukasi. Berdasarkan perhitungan, total biaya yang dikeluarkan untuk pembuatan robot ini adalah Rp 2.835.000. Biaya ini mencakup pembelian komponen elektronik seperti motor stepper NEMA 17 dengan harga Rp 175.000, driver DRV8825, mikrokontroler Arduino Mega 2560 Pro Mini dengan Harga Rp 190.000, sensor Hall Effect, serta bahan mekanik seperti bearing, pulley, belt, dan

material 3D printing. Jika dibandingkan dengan harga lengan robot komersial yang dapat mencapai 100 juta hingga 500 juta rupiah, biaya ini tergolong sangat ekonomis. Dengan biaya tersebut, institusi pendidikan dapat menyediakan lebih banyak unit robot sehingga mahasiswa memiliki kesempatan lebih luas untuk berinteraksi secara langsung dengan perangkat ini. Hal ini sejalan dengan tujuan penelitian untuk menyediakan alternatif sarana pembelajaran robotika yang murah namun tetap efektif di lingkungan pendidikan tinggi.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

1. Lengan robot artikulasi 3 Derajat kebebasan berhasil dirancang dan dibangun dengan biaya yang terjangkau tanpa mengorbankan akurasi dan repeatability. Sistem mampu menjalankan perintah *G-code* dan *M-code* dengan baik, serta menunjukkan kemampuan pengulangan posisi *(repeatabilit*y) yang stabil setelah dilakukan pengujian berulang. Penggunaan motor stepper, sistem transmisi berbasis timing belt, dan material hasil 3D printing memungkinkan tercapainya desain yang ringan, modular, dan presisi.
2. Lengan robot ini telah diimplementasikan sebagai sarana pembelajaran robotika yang efektif dan praktis. Mahasiswa dapat dengan mudah mengoperasikan robot tanpa harus mempelajari rumus kinematika yang kompleks, sehingga mempermudah pemahaman konsep dasar robotika dan kontrol gerak. Harga yang ekonomis memungkinkan setiap mahasiswa untuk dapat berlatih secara mandiri, sehingga menciptakan pengalaman belajar yang lebih interaktif, aplikatif, dan menyenangkan.

5.2 Saran

1 Menambahkan Fitur Kontrol Berbasis Jaringan

Untuk meningkatkan fleksibilitas dan kemudahan dalam pengoperasian, disarankan agar sistem lengan robot ini dikembangkan dengan menambahkan fitur kontrol berbasis jaringan atau IoT. Dengan adanya kontrol jarak jauh melalui jaringan internet atau lokal, pengguna tidak harus terhubung secara langsung melalui kabel ke perangkat komputer. Hal ini akan memperluas ruang lingkup penggunaan robot, terutama dalam lingkungan laboratorium atau pembelajaran berbasis daring.

2. Mengintegrasikan Antarmuka Pengguna Grafis (Graphical User Interface/GUI)

   Saat ini, pengoperasian robot masih dilakukan melalui terminal perintah manual berbasis *G-code* dan *M-code.* Untuk mempermudah pengguna, terutama mahasiswa yang baru mempelajari robotika, sebaiknya sistem dikembangkan dengan menambahkan antarmuka pengguna grafis (GUI) berbasis software seperti Visual Studio, Python, atau berbasis web. GUI akan memudahkan pengguna dalam memberikan perintah, memantau status robot, serta melakukan kalibrasi tanpa memerlukan pemahaman mendalam mengenai kode perintah.

3. Meningkatkan Desain Mekanik Agar Lebih Kuat dan Presisi

   Dari hasil evaluasi, sistem mekanik yang menggunakan material hasil cetak 3D cukup memadai untuk beban ringan dalam pembelajaran dasar. Namun demikian, untuk pengembangan lebih lanjut, perlu dilakukan perbaikan pada kekuatan dan presisi mekanik, seperti penggunaan material yang lebih kokoh atau desain yang meminimalkan kelonggaran sambungan dan backlash. Peningkatan ini memungkinkan robot mampu membawa beban yang lebih besar dan menjalankan tugas-tugas yang lebih kompleks dengan akurasi yang lebih tinggi.

DAFTAR PUSTAKA

[1] Denis, 2023. [Online]. Available: https://profilbaru.com/Robot.

[2] E. Pitowarno, Robotika : Desain, Kontrol dan Kecerdasan Buatan / Endra Pitowarno, Perpustakaan Politeknik ATI Makassar, 2015.

[3] B. S. Beribe, "Ada di Urutan ke-5, Teknologi Robotik Berhasil Gerakkan Industri Indonesia," 9 Desember 2019. [Online]. Available: https://www.akurat.co/infotech/1302156200/Ada-di-Urutan-ke5-Teknologi-Robotik-Berhasil-Gerakkan-Industri-Indonesia?.

[4] Z. Nurkholik, F. A. Fiolana and D. A. Kusumastutie, "Robotik Arm Rancangan Bangun Lengan Robot Arm Untuk Menggambar Menggunakan Invers Kinematik," *Jurnal Ilmiah Sistem Informasi,* 2022.

[5] S. Kariuki, E. Wanjau, I. Muchiri, J. Mugoro, N. Waweru and M. Sasaki, "Pick and Place Control of a 3-DOF Robot Manipulator Based on Image and Pattern Recognition," *University of Technology,* 2024.

[6] U. Singh, "Applications of Stepper Motor," 2018. [Online]. Available: https://mechtex.com/blog/applications-of-stepper-motor.

[7] Testing Indonesia, "Stepper Motor Control," 2020. [Online]. Available: https://testingindonesia.co.id/product/stepper-motor-control/.

[8] N. Alamsyah, H. Arfandy, M. Rahma and A. Darmawansyah, "Rancang Bangun Trainer Kit Berbasis Arduino Sebagai Media Pembelajaran Pada Mata Kuliah Robotika," *Jurnal Teknologi dan Komputer,* 2022.

[9] Universitas Medan Area, "Optimasi Produksi di Era Manufaktur Modern," 2023. [Online]. Available: https://bpmbkm.uma.ac.id/2024/10/23/optimasi-produksi-di-era-manufaktur-modern.

[10] M. Quigley, A. Asbeck and Andrew, "A Low-cost Compliant 7-DOF Robotic Manipulator," *Stanford,* 2021.

[11] Scanm2, "Import Substitution: How Reverse Engineering Solves the Problem of Missing Parts," 2023. [Online]. Available: https://scanm2.com/import-substitution-how-reverse-engineering-solves-the-problem-of-missing-parts.

[12] A. d. A. J. M. Baso, "Pengembangan Prototipe Robot Manipulator," *Politeknik Negri Ujung Pandang,* 2017.

[13] L. Hakim, Rusnaldy and Paryanto, "Reverse Engineering Pada Komponen Otomotif Dengan Metode Photogrammetry," *Jurnal Teknik Mesin S-1,* 2023.

[14] M. Bugday and M. Karali, "Design optimization of industrial robot arm to minimize redundant weight," *Engineering Science and Technology an International Journal,* 2018.

[15] Y. Chen, "Structural Optimization and Lightweight Analysis of Industrial Robot Arm," *Artificial Intelligence Technology Research,* 2019.

[16] W. Kacalak, M. Majewski and Z. Budniak, "Innovative design of non-backlash worm gear drives," *Archives of Civil and Mechanical Engineering,* 2018.

[17] M. Bernal and J. Civera, "LoCoQuad: A Low-Cost Arachnoid Quadruped Robot for Research and Education," *Cornell Universitas,* 2020.

[18] Supriandi, "Desain Adaptif dan Fleksibel pada Robotika Industri : Membuka Jalan Untuk Produksi Berkelanjutan dan Otomatisasi yang Efisien," *Jurnal Multidisiplin West Science,* 2023.

[19] Shane, "Teknologi Cetak 3D: Aplikasi Transformatif," 2023. [Online]. Available: https://www.machinemfg.com/id/3d-print-applications.

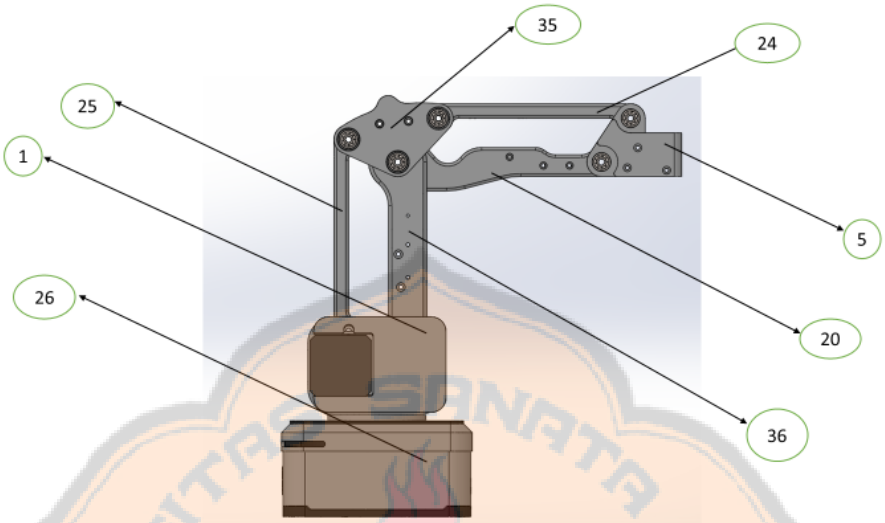
[20] M. N. H. K. Sharin, A. P. P. A. Majeed, N. M. Thamrin and R. Jailani, "DIY 3-DOF Robotic Arm for Teaching and Learning," *Springer Nature Singapore,* 2020.

[21] A. Suarez, D. Garcia Costa, J. Perez, E. López-Iñesta, F. Grimaldo and J. Torres, "Hands-on Learning: Assessing the Impact of a Mobile Robot Platform in Engineering Learning Environments," *Journal Sustainability,* 2023.

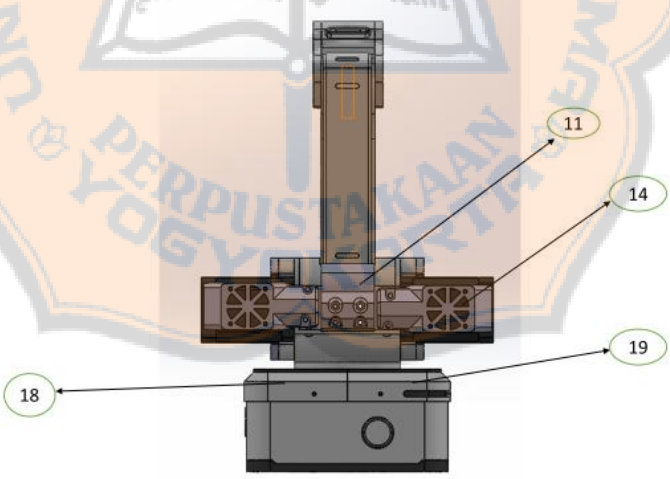
[22] S. Parlak, N. Tokel and Ü. Çakıroğlu, "Reverse Engineering in Robotics Classrooms: Boosting Creative Thinking and Problem Solving," *International Journal of Computer Science Education in Schools,* 2024.

LAMPIRAN

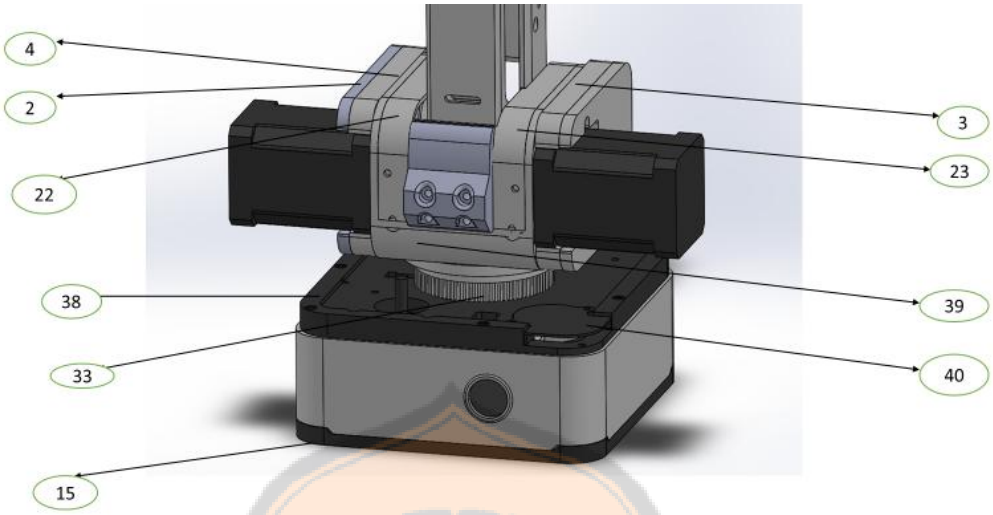
Lampiran 1 Rangkaian Mekanik



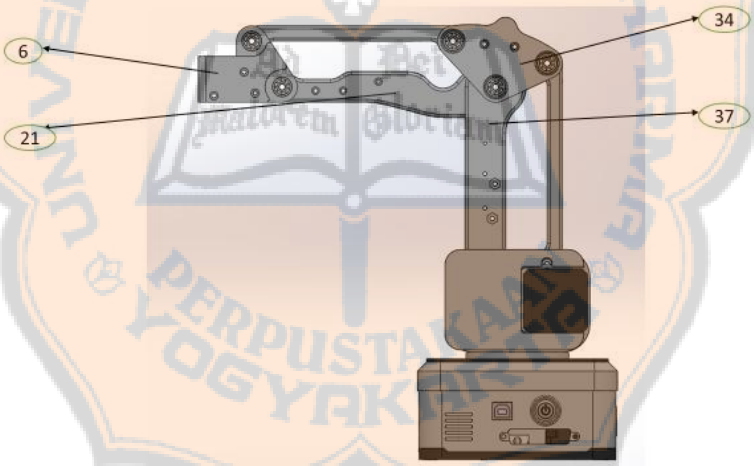
Gambar L1 Robot Tampak Samping



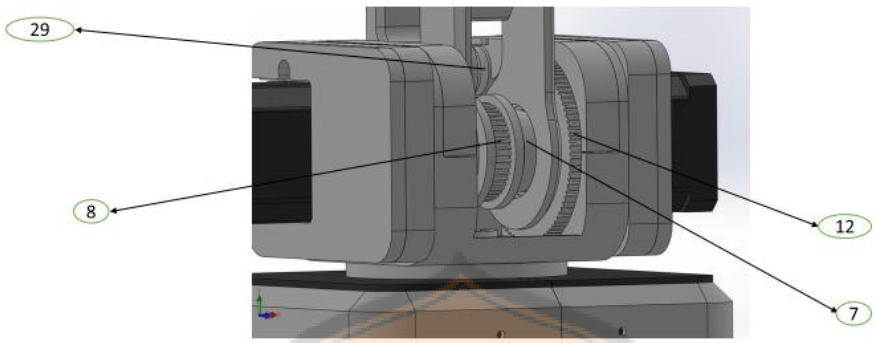
Gambar L2 Robot Tampak Belakang



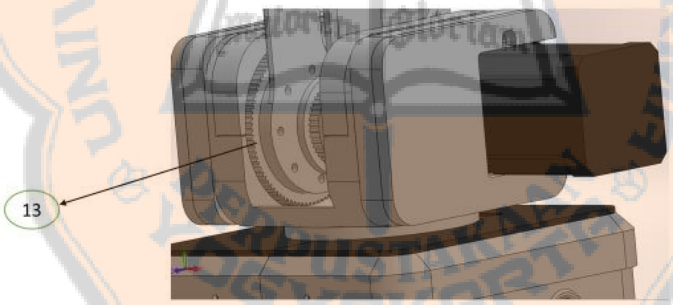
Gambar L3 Robot Tampak Belakang



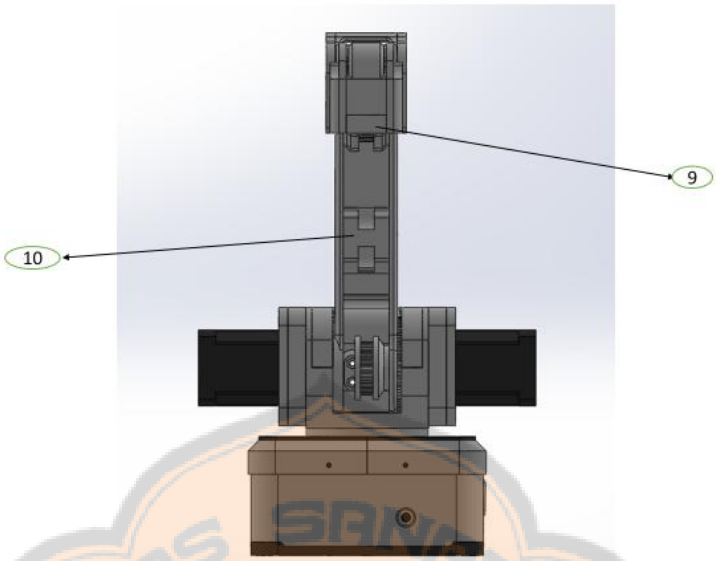
Gambar L4 Robot Tampak Samping



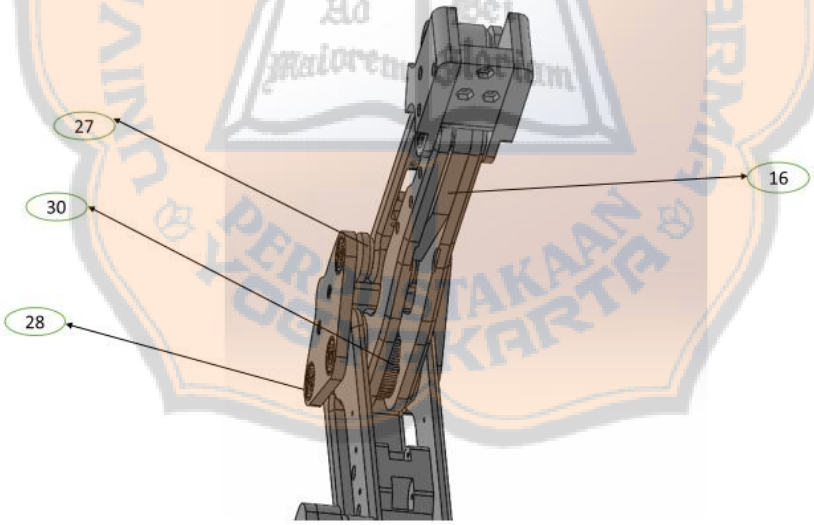
Gambar L5 bagian Roda Gigi Robot



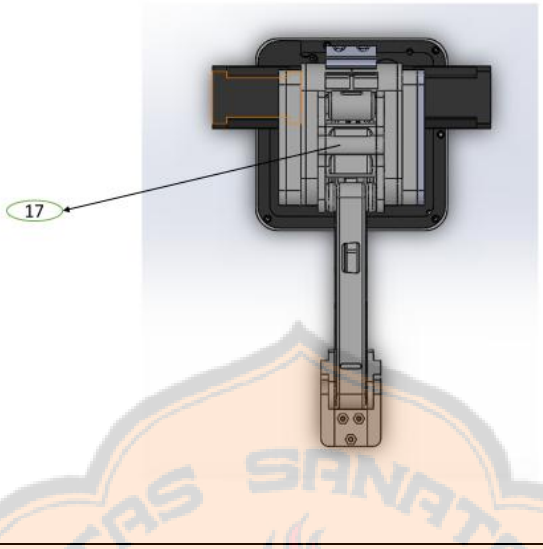
Gambar L6 Bagian Roda Gigi Robot



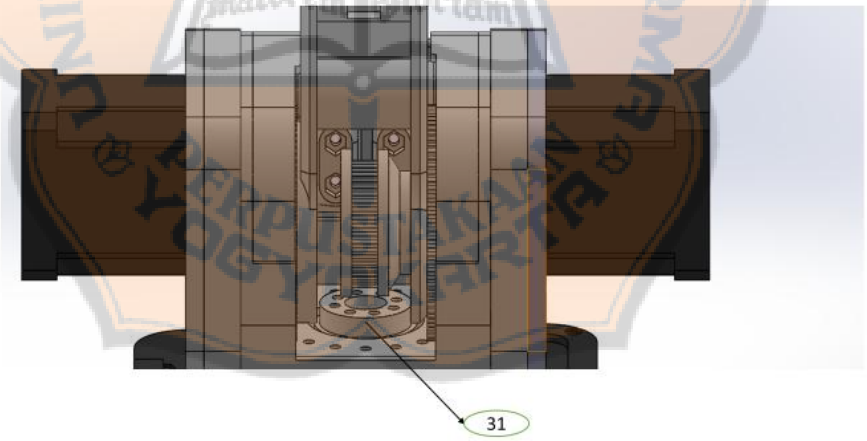
Gambar L7 Robot Tampak Depan



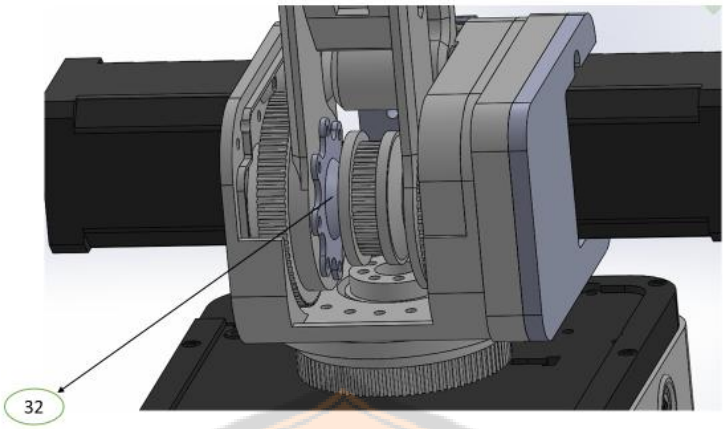
Gambar L8 Lengan Robot Y



Gambar L9 Robot Tampak Atas

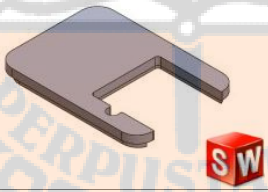


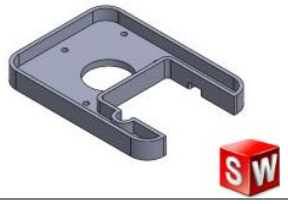
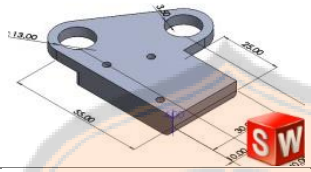
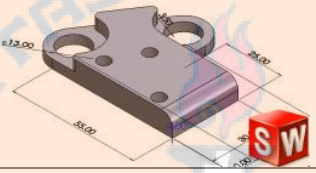
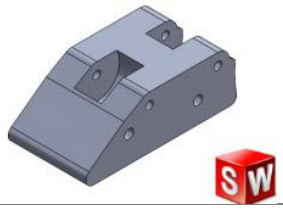
Gambar L10 Bagian Roda Gigi Robot

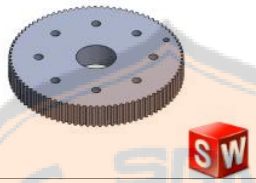


Gambar L 11 Bagian Roda Gigi Robot

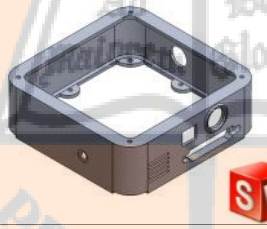
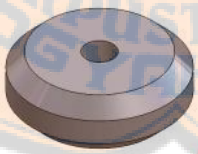
Tabel M1 . Penjelasan daftar *Part* 3D *print* untuk rangkaian mekanik (lampiran 1)

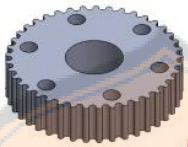
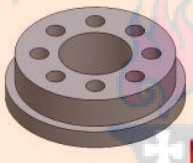
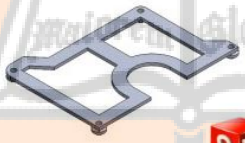
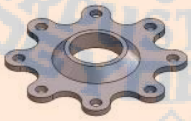
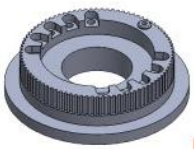
PART NUMBER	**GAMBAR**	**JUMLAH PRINT**
PART 1 R		1
PART 2 L		1
PART 3 R		1

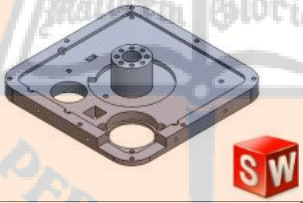
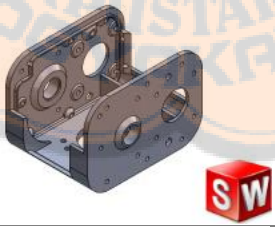
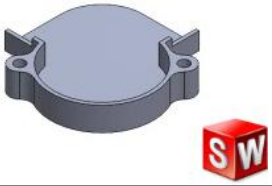
PART NUMBER	GAMBAR	JUMLAH PRINT
PART 4 L		1
PART 5 R		1
PART 6 L		1
PART 7		1
PART 8		1
PART 9		1
PART 10		1

PART NUMBER	GAMBAR	JUMLAH PRINT
PART 11		1
PART 12		1
PART 13		1
PART 14		2
PART 15		1
PART 16		1

PART NUMBER	GAMBAR	JUMLAH PRINT
PART 17		1
PART 18		1
PART 19		1
PART 20		1
PART 21		1
PART 22		1

PART NUMBER	GAMBAR	JUMLAH PRINT
PART 23		1
PART 24		1
PART 25		1
PART 26		1
PART 27		4
PART 28		10

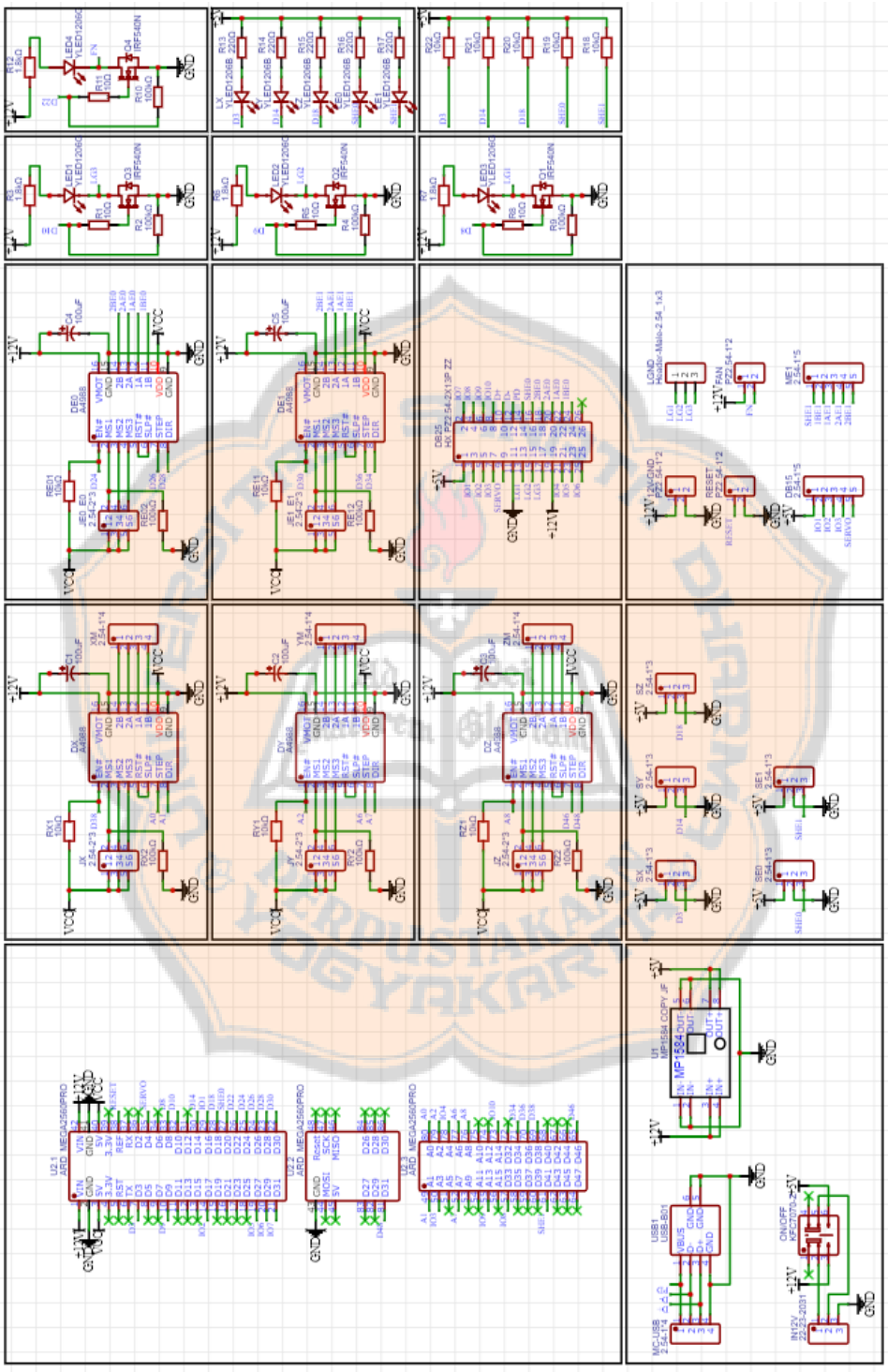
PART NUMBER	GAMBAR	JUMLAH PRINT
PART 29		2
PART 30		1
PART 31		1
PART 32		1
PART 33		1
PART 34		1
PART 35 L		1

PART NUMBER	GAMBAR	JUMLAH PRINT
PART 36 R		1
PART 37		1
PART 38		1
PART 39		1
PART 40		1
PART 41		1

Tabel M2. Komponen Mekanik

No	Komponen	JUMLAH
1	*Idler* GT2 IGT b3w6	4
2	AS 4mm	60
3	*Bearing* F62422	26
4	*Ball bearing* 6705 zz 25x32x4	3
5	F624zzn 4x15x5 *bearing laker ball*	10
6	PTFB 4X2	30
7	*Belt* GT2 200x6	3
8	*Belt* GT2 220x6mm	1
9	GT2 *timing belt close loop gigi*	1
10	*Pulli* 16T B5 W6	1
11	GT2 *timing pulley* 20 *teeth gear* 3	3
12	Pab 4x1/2	7
13	Baut 3x20	30
14	Baut 3x6	20
15	Baut 3x12	10
16	Baut 3x15	10
17	Baut 3x10	15
18	Baut 3x30	10
19	Baut 3x40	15
20	Pab 4x3/8	30
21	*Ring* nat 3x8x0,8 kuning	10
22	Baut R M3x10	2
23	Baut M3x10mm *Philips pan head* 304 *stainless steel*	5
24	Baut M3x16mm *Philips pan head* 304 *stainless steel*	12
25	Baut M3x10MM *round head*	10
26	JP M3x25	10
27	Mur M3 pth	10
28	*Nozle* MK 0,4MM	2
29	Cetak 3D *print* bahan PLA+	337
30	Magnet No 3x1	12

LAMPIRAN 2. SKEMATIK RANGKAIAN



Gambar L 12 Skematik Rangkaian

Tabel M3. Komponen Elektrik

No	Komponen	JUMLAH
1	Fan 30w 12V	4
2	Wiremesh	1
3	ASB 51105	1
4	CNC Sheild V3 a4988 driver expans	1
5	Stepstick a4988 stepper driver mo	4
6	DRV 8825	4
7	3D Printer Stepstick DRV8825 Step	5
8	DB25 Connector Male cable type	1
9	DB25 Connector Female cable type	1
10	Konektor DB12 VGA MALE	1
11	Konektor DB12 VGA FEMALE	1
12	Kabel konektor motor stepper XH2	2
13	Kabel 1x0,6mm pejal single wire	20
14	Kabel 26AWG UL 1007 26 AWG ELECTRO	18
15	42MM NEMA 17 2 Phase	1
16	NEMA 17 stepper motor 17HS6401 1.7A High torque	2
17	Heat shrinkable tubing 1,5mm	1
18	Heat shrinkable tubing 2mm	2
19	Heat shrinkable tubing 1mm	1
20	MP1584EN Small 3A DC to DC Step	1
21	IRI 540 MOSFET IRF 540 100V 33A N Channel power Mos FE	8
22	IRF540N N Channel Mosfet T	4
23	Green button waterproof 16mm 5A metal self lock flat with power	2
24	Socket power DC dengan baut untuk box	2
25	Socket XH 54 Connector 2.54mm 2P Putih 2 pin nale female lurus 1 set	3
26	Socket header male 2,54mm lurus 40P 40 2,54mm connector merah	6
27	Arduino Mega 2560 pro CH340	1
28	Arduino Mega 2560 PRO Mega2560 PRO CH340	1
29	10k ohm SMD 1206 Resistor	10
30	100k ohm SMD 1206 Resistor	10
31	220 ohm SMD 1206 Resistor	5
32	1K8 ohm SMD 1206 Resistor	4
33	10 ohm SMD 1206 resistor	6
34	10K Ohm 1/4 watt metal film resi	3
35	LED SMD 1206 Putih	10
36	10UF/25V ELCO 6x11mm	4
37	220UF/25V ELCO 8x11mm nichicon	1
38	100UF 35V 6,3x7,7 capacitor elco kapasitas 100uF	5
39	Konektor *micro USB male 5P*	1

No	Komponen	JUMLAH
40	2 *pin jumper* hitam	1
41	1x40 *pin female header single row*	3
42	Konektor *USB type B printer female*	1
43	A3144 A3144E 0H3144 Y3144 *Hall Effect sensor magnet for arduino*	8
44	Arduino *stackable header* 8 pin	10
45	*Black housing* 1 pin	80
46	*Black housing* 4 pin	3
47	*Stepper motor drive heat sink* TMC	2

Lampiran 3. Program Utama

```
#include <Arduino.h>
//GENERAL CONFIG SETTINGS
#include "config.h"

#include "robotGeometry.h"
#include "interpolation.h"
#include "RampsStepper.h"
#include "queue.h"
#include "command.h"
#include "equipment.h"
#include "endstop.h"
#include "logger.h"
#include "fanControl.h"
//INCLUDE CORRESPONDING GRIPPER MOTOR CLASS
#if GRIPPER == SERVO
  #include "servo_gripper.h"
#elif GRIPPER == BYJ
  #include "byj_gripper.h"
#endif
  #include "pinout/pinout.h"

//STEPPER OBJECTS
RampsStepper stepperHigher(Z_STEP_PIN, Z_DIR_PIN, Z_ENABLE_PIN,
INVERSE_Z_STEPPER, MAIN_GEAR_TEETH, MOTOR_GEAR_TEETH, MICROSTEPS,
STEPS_PER_REV);
RampsStepper stepperLower(Y_STEP_PIN, Y_DIR_PIN, Y_ENABLE_PIN,
INVERSE_Y_STEPPER, MAIN_GEAR_TEETH, MOTOR_GEAR_TEETH, MICROSTEPS,
STEPS_PER_REV);
RampsStepper stepperRotate(X_STEP_PIN, X_DIR_PIN, X_ENABLE_PIN,
INVERSE_X_STEPPER, MAIN_GEAR_TEETH, MOTOR_GEAR_TEETH, MICROSTEPS,
STEPS_PER_REV);

//RAIL OBJECTS
#if RAIL
  RampsStepper stepperRail(E0_STEP_PIN, E0_DIR_PIN, E0_ENABLE_PIN,
INVERSE_E0_STEPPER, MAIN_GEAR_TEETH, MOTOR_GEAR_TEETH, MICROSTEPS,
STEPS_PER_REV);
  #if BOARD_CHOICE == WEMOSD1R32 //PINSWAP REQIURED ON D1R32 DUE
TO INSUFFICIENT DIGIAL PINS
    #define SERVO_PIN 23 // REDEFINE SERVO_PIN FOR RAIL // SHARE
WITH Z_MIN_PIN
    Endstop endstopE0(E0_MIN_PIN, E0_DIR_PIN, E0_STEP_PIN,
E0_ENABLE_PIN, E0_MIN_INPUT, E0_HOME_STEPS, HOME_DWELL, true);
  #else
    Endstop endstopE0(E0_MIN_PIN, E0_DIR_PIN, E0_STEP_PIN,
E0_ENABLE_PIN, E0_MIN_INPUT, E0_HOME_STEPS, HOME_DWELL, false);
  #endif
#endif

//ENDSTOP OBJECTS
Endstop endstopX(X_MIN_PIN, X_DIR_PIN, X_STEP_PIN, X_ENABLE_PIN,
X_MIN_INPUT, X_HOME_STEPS, HOME_DWELL, false);
```

```
Endstop endstopY(Y_MIN_PIN, Y_DIR_PIN, Y_STEP_PIN, Y_ENABLE_PIN,
Y_MIN_INPUT, Y_HOME_STEPS, HOME_DWELL, false);
#if BOARD_CHOICE == WEMOSD1R32
  Endstop endstopZ(Z_MIN_PIN, Z_DIR_PIN, Z_STEP_PIN, Z_ENABLE_PIN,
Z_MIN_INPUT, Z_HOME_STEPS, HOME_DWELL, true);
#else
  Endstop endstopZ(Z_MIN_PIN, Z_DIR_PIN, Z_STEP_PIN, Z_ENABLE_PIN,
Z_MIN_INPUT, Z_HOME_STEPS, HOME_DWELL, false);
#endif

//EQUIPMENT OBJECTS
#if GRIPPER == SERVO
  Servo_Gripper servo_gripper(SERVO_PIN, SERVO_GRIP_DEGREE,
SERVO_UNGRIP_DEGREE);
#elif GRIPPER == BYJ
  BYJ_Gripper byj_gripper(BYJ_PIN_0, BYJ_PIN_1, BYJ_PIN_2,
BYJ_PIN_3, BYJ_GRIP_STEPS);
#endif
Equipment lg1(LG1_PIN);
Equipment lg2(LG2_PIN);
Equipment lg3(LG3_PIN);
Equipment led(LED_PIN);
FanControl fan(FAN_PIN, FAN_DELAY);

//EXECUTION & COMMAND OBJECTS
RobotGeometry geometry(END_EFFECTOR_OFFSET, LOW_SHANK_LENGTH,
HIGH_SHANK_LENGTH);
Interpolation interpolator;
Queue<Cmd> queue(QUEUE_SIZE);
Command command;

//PS4 CONTROLLER OBJECT FOR ESP32
#if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == DUALSHOCK4
  #include "controller_ps4.h"
  Controller_PS4 controller_ps4(PS4_MAC);
#endif
#if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == WIIMOTE
  #include "controller_wiimote.h"
  Controller_Wiimote controller_wiimote;
#endif

//--------------------------------IO-----------------------------
--------------
int IO1Before = LOW;
int IO2Before = LOW;
int IO3Before = LOW;
//---------------------------------------------------------------
--------------

void setup()
{
  Serial.begin(BAUD);
  #if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == DUALSHOCK4
    controller_ps4.setup();
  #endif
  #if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == WIIMOTE
```

```
    controller_wiimote.setup();
  #endif
  stepperHigher.setPositionRad(PI / 2.0); // 90°
  stepperLower.setPositionRad(0);         // 0°
  stepperRotate.setPositionRad(0);        // 0°
  #if RAIL
  stepperRail.setPosition(0);
  #endif
  if (HOME_ON_BOOT) { //HOME DURING SETUP() IF HOME_ON_BOOT
ENABLED
    homeSequence();
    Logger::logINFO("ROBOT ONLINE");
  } else {
    setStepperEnable(false); //ROBOT ADJUSTABLE BY HAND AFTER
TURNING ON
    if (HOME_X_STEPPER && HOME_Y_STEPPER && !HOME_Z_STEPPER){
      Logger::logINFO("ROBOT ONLINE");
      Logger::logINFO("ROTATE ROBOT TO FACE FRONT CENTRE & SEND
G28 TO CALIBRATE");
    }
    if (HOME_X_STEPPER && HOME_Y_STEPPER && HOME_Z_STEPPER){
      Logger::logINFO("ROBOT ONLINE");
      Logger::logINFO("READY CALIBRATION");
      Logger::logINFO("ok");
    }
    if (!HOME_X_STEPPER && !HOME_Y_STEPPER){
      Logger::logINFO("ROBOT ONLINE");
      Logger::logINFO("HOME ROBOT MANUALLY & SEND G28 TO
CALIBRATE");
    }
  }
  interpolator.setInterpolation(INITIAL_X, INITIAL_Y, INITIAL_Z,
INITIAL_E0, INITIAL_X, INITIAL_Y, INITIAL_Z, INITIAL_E0);

//-------------------------------------IO-------------------------
-----------------------
  pinMode(IO1_PIN, INPUT);
  pinMode(IO2_PIN, INPUT);
  pinMode(IO3_PIN, INPUT);
//------------------------------------------------------------------
-------------------------

}

void loop() {
  interpolator.updateActualPosition();
  geometry.set(interpolator.getXPosmm(), interpolator.getYPosmm(),
interpolator.getZPosmm());
  stepperRotate.stepToPositionRad(geometry.getRotRad());
  stepperLower.stepToPositionRad(geometry.getLowRad());
  stepperHigher.stepToPositionRad(geometry.getHighRad());
  #if RAIL
    stepperRail.stepToPositionMM(interpolator.getEPosmm(),
STEPS_PER_MM_RAIL);
  #endif
  stepperRotate.update();
```

```
  stepperLower.update();
  stepperHigher.update();
  #if RAIL
    stepperRail.update();
  #endif
  fan.update();

  if (!queue.isFull()) {
    if (command.handleGcode()) {
    Cmd cmd = command.getCmd();
    if (cmd.id == 'M' && cmd.num == 119) {
      Logger::logINFO("ENDSTOP STATES: [X:" +
String(endstopX.state()) +
                      " Y:" + String(endstopY.state()) +
                      " Z:" + String(endstopZ.state()) + "]");
      return;
    }

      queue.push(command.getCmd());
    }
  }
  if ((!queue.isEmpty()) && interpolator.isFinished()) {
    executeCommand(queue.pop());
    if (PRINT_REPLY) {
      Serial.println(PRINT_REPLY_MSG);
    }
  }

  if (millis() % 500 < 250) {
    led.cmdOn();
  }
  else {
    led.cmdOff();
  }

  #if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == DUALSHOCK4
    ps4_controller_loop();
  #endif
  #if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == WIIMOTE
    wiimote_controller_loop();
  #endif


 //============== SENSOR
=========================================================
  if (digitalRead(IO1_PIN) == HIGH && IO1Before == LOW) {
      Logger::logINFO("S1 ON");
      IO1Before = HIGH;
  } else if (digitalRead(IO1_PIN) == LOW && IO1Before == HIGH) {
      Logger::logINFO("S1 OFF");
      IO1Before = LOW;
  }

  if (digitalRead(IO2_PIN) == HIGH && IO2Before == LOW) {
      Logger::logINFO("S2 ON");
      IO2Before = HIGH;
```

```
  } else if (digitalRead(IO2_PIN) == LOW && IO2Before == HIGH) {
      Logger::logINFO("S2 OFF");
      IO2Before = LOW;
  }

  if (digitalRead(IO3_PIN) == HIGH && IO3Before == LOW) {
      Logger::logINFO("S3 ON");
      IO3Before = HIGH;
  } else if (digitalRead(IO3_PIN) == LOW && IO3Before == HIGH) {
      Logger::logINFO("S3 OFF");
      IO3Before = LOW;
  }
}

void executeCommand(Cmd cmd) {

  if (cmd.id == -1) {
    printErr();
    return;
  }
  if (cmd.id == 'G') {
    switch (cmd.num) {
    case 0:
    case 1:
      fan.enable(true);
      Point posoffset;
      posoffset = interpolator.getPosOffset();
      cmdMove(cmd, interpolator.getPosmm(), posoffset,
command.isRelativeCoord);
      interpolator.setInterpolation(cmd.valueX, cmd.valueY,
cmd.valueZ, cmd.valueE, cmd.valueF);
      Logger::logINFO("LINEAR MOVE: [X:" + String(cmd.valueX-
posoffset.xmm) + " Y:" + String(cmd.valueY-posoffset.ymm) + " Z:"
+ String(cmd.valueZ-posoffset.zmm) + " E:" + String(cmd.valueE-
posoffset.emm)+"]");
    case 4: cmdDwell(cmd); break;
    case 28:
      if (BOARD_CHOICE == UNO || BOARD_CHOICE == WEMOSD1R32){
        homeSequence_UNO();
        break;
      } else {
        homeSequence();
        break;
      }
    case 90: command.cmdToAbsolute(); break; // ABSOLUTE
COORDINATE MODE
    case 91: command.cmdToRelative(); break; // RELATIVE
COORDINATE MODE
    case 92:
      interpolator.resetPosOffset();
      cmdMove(cmd, interpolator.getPosmm(),
interpolator.getPosOffset(), false);
      interpolator.setPosOffset(cmd.valueX, cmd.valueY,
cmd.valueZ, cmd.valueE);
      break;
    default: printErr();
```

```
    }
  }
  else if (cmd.id == 'M') {
    switch (cmd.num) {
    case 1: lg1.cmdOn(); break;
    case 2: lg1.cmdOff(); break;
    case 3:
      #if GRIPPER == BYJ
        byj_gripper.cmdOn(); break;
      #elif GRIPPER == SERVO
        servo_gripper.cmdOn(); break;
      #endif
    case 5:
      #if GRIPPER == BYJ
        byj_gripper.cmdOff(); break;
      #elif GRIPPER == SERVO
        servo_gripper.cmdOff(); break;
      #endif
    case 6: lg2.cmdOn(); break;
    case 7: lg2.cmdOff(); break;
    case 17: setStepperEnable(true); break;
    case 18: setStepperEnable(false); break;
    case 106: fan.enable(true); break;
    case 107: fan.enable(false); break;
    case 114: command.cmdGetPosition(interpolator.getPosmm(),
interpolator.getPosOffset(), stepperHigher.getPosition(),
stepperLower.getPosition(), stepperRotate.getPosition()); break;//
Return the current positions of all axis
    case 119: {
      String endstopMsg = "ENDSTOP: [X:";
      endstopMsg += String(endstopX.state());
      endstopMsg += " Y:";
      endstopMsg += String(endstopY.state());
      endstopMsg += " Z:";
      endstopMsg += String(endstopZ.state());
      #if RAIL
        endstopMsg += " E:";
        endstopMsg += String(endstopE0.state());
      #endif
      endstopMsg += "]";
      //ORIGINAL LOG STRING UNDESIRABLE FOR UNO PROCESSING
      //Logger::logINFO("ENDSTOP STATE:
[UPPER_SHANK(X):"+String(endstopX.state())+"
LOWER_SHANK(Y):"+String(endstopY.state())+"
ROTATE_GEAR(Z):"+String(endstopZ.state())+"]");
      Logger::logINFO(endstopMsg);
      break;}
    case 205:
      interpolator.setSpeedProfile(cmd.valueS);
      Logger::logINFO("SPEED PROFILE: [" +
String(interpolator.speed_profile) + "]");
      break;
    case 206: lg3.cmdOn(); break;
    case 207: lg3.cmdOff(); break;
    default: printErr();
    }
```

```
  }
  else {
    printErr();
  }
}

void setStepperEnable(bool enable){
  stepperRotate.enable(enable);
  stepperLower.enable(enable);
  stepperHigher.enable(enable);
  #if RAIL
    stepperRail.enable(enable);
  #endif
  fan.enable(enable);
}

void homeSequence(){
  setStepperEnable(false);
  fan.enable(true);
  if (HOME_Y_STEPPER && HOME_X_STEPPER){
    endstopY.home(!INVERSE_Y_STEPPER);
    endstopX.home(!INVERSE_X_STEPPER);
  } else {
    setStepperEnable(true);
    endstopY.homeOffset(!INVERSE_Y_STEPPER);
    endstopX.homeOffset(!INVERSE_X_STEPPER);
  }
  if (HOME_Z_STEPPER){
    endstopZ.home(INVERSE_Z_STEPPER);
  }
  #if RAIL
    if (HOME_E0_STEPPER){
      endstopE0.home(!INVERSE_E0_STEPPER);
    }
  #endif
  interpolator.setInterpolation(INITIAL_X, INITIAL_Y, INITIAL_Z,
INITIAL_E0, INITIAL_X, INITIAL_Y, INITIAL_Z, INITIAL_E0);
  Logger::logINFO("HOMING COMPLETE");
}

//DUE TO UNO CNC SHIELD LIMIT, 1 EN PIN SERVES 3 MOTORS, HENCE
DIFFERENT HOMESEQUENCE IS REQUIRED
void homeSequence_UNO(){
  #if GRIPPER == SERVO
  if (servo_gripper.readDegree() != SERVO_UNGRIP_DEGREE){
    servo_gripper.cmdOff();
  }
  #endif
  if (HOME_Y_STEPPER && HOME_X_STEPPER){
    while (!endstopY.state() || !endstopX.state()){
      endstopY.oneStepToEndstop(!INVERSE_Y_STEPPER);
      endstopX.oneStepToEndstop(!INVERSE_X_STEPPER);
    }
    endstopY.homeOffset(!INVERSE_Y_STEPPER);
    endstopX.homeOffset(!INVERSE_X_STEPPER);
  } else {
```

```
    setStepperEnable(true);
    endstopY.homeOffset(!INVERSE_Y_STEPPER);
    endstopX.homeOffset(!INVERSE_X_STEPPER);
  }
  if (HOME_Z_STEPPER){
    endstopZ.home(INVERSE_Z_STEPPER); //INDICATE STEPPER HOMING
DIRECDTION
  }
  #if RAIL
    if (HOME_E0_STEPPER){
      endstopE0.home(!INVERSE_E0_STEPPER);
    }
  #endif
  interpolator.setInterpolation(INITIAL_X, INITIAL_Y, INITIAL_Z,
INITIAL_E0, INITIAL_X, INITIAL_Y, INITIAL_Z, INITIAL_E0);
  Logger::logINFO("HOMING COMPLETE");
}

#if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == DUALSHOCK4
void ps4_controller_loop(){
  controller_ps4.update();
  interpolator.speed_profile = 0;
  if (controller_ps4.buttons[PS4_CROSS]){if (GRIPPER ==
SERVO){servo_gripper.cmdOn();}}
  if (controller_ps4.buttons[PS4_CIRCLE]){if (GRIPPER ==
SERVO){servo_gripper.cmdOff();}}
  if
(controller_ps4.buttons[PS4_OPTIONS]){setStepperEnable(true);}
  if (controller_ps4.buttons[PS4_SHARE]){setStepperEnable(false);}
  if (controller_ps4.buttons[PS4_TOUCHPAD]){homeSequence_UNO();}
  float x_distance = 0.0;
  float y_distance = 0.0;
  float z_distance = 0.0;
  float e_distance = 0.0;
  if (abs(controller_ps4.buttons[PS4_RSTICKY]) > 10){
//    z_distance = float(controller_ps4.buttons[PS4_RSTICKY]) /
2500.0;
    z_distance =
float(controller_ps4.buttons[PS4_RSTICKY])*JOYSTICK_SPEED_MULTIPLI
ER / 2500.0;
  }
  if (abs(controller_ps4.buttons[PS4_LSTICKX]) > 10){
    float turn_rad =
float(controller_ps4.buttons[PS4_LSTICKX])*JOYSTICK_SPEED_MULTIPLI
ER / 300000.0;
    x_distance += sin(geometry.getRotRad() + turn_rad) *
geometry.getHypot() - interpolator.getXPosmm();
    y_distance += cos(geometry.getRotRad() + turn_rad) *
geometry.getHypot() - interpolator.getYPosmm();
  }
  if (abs(controller_ps4.buttons[PS4_LSTICKY]) > 10){
    float hp_distance =
float(controller_ps4.buttons[PS4_LSTICKY])*JOYSTICK_SPEED_MULTIPLI
ER / 2500.0;
    float hp_ratio = hp_distance / geometry.getHypot();
    x_distance += interpolator.getXPosmm() * hp_ratio;
```

```
    y_distance += interpolator.getYPosmm() * hp_ratio;
  }
  if (abs(controller_ps4.buttons[PS4_L2VALUE]) > 10){
    e_distance -=
float(controller_ps4.buttons[PS4_L2VALUE])*JOYSTICK_SPEED_MULTIPLI
ER/ 10000.0;
  }
  if (abs(controller_ps4.buttons[PS4_R2VALUE]) > 10){
    e_distance +=
float(controller_ps4.buttons[PS4_R2VALUE])*JOYSTICK_SPEED_MULTIPLI
ER/ 10000.0;
  }
  if (x_distance || y_distance || z_distance || e_distance){
    interpolator.speed_profile = 0;

interpolator.setInterpolation(interpolator.getXPosmm()+x_distance,
interpolator.getYPosmm()+y_distance,
interpolator.getZPosmm()+z_distance,
interpolator.getEPosmm()+e_distance, 5);
  }
}
#endif

#if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == WIIMOTE
void wiimote_controller_loop(){
  controller_wiimote.update();
  interpolator.speed_profile = 0;
  if (controller_wiimote.button == WII_A && GRIPPER == SERVO){
    if (!servo_gripper.isOn()){
      servo_gripper.cmdOn();
    } else {
      servo_gripper.cmdOff();
    }
  }
  if (controller_wiimote.button ==
WII_PLUS){setStepperEnable(true);}
  if (controller_wiimote.button ==
WII_MINUS){setStepperEnable(false);}
  if (controller_wiimote.button == WII_HOME){homeSequence_UNO();}

  float x_distance = 0.0;
  float y_distance = 0.0;
  float z_distance = 0.0;
  float e_distance = 0.0;
  float hp_distance = 0.0;
  float turn_rad = 0.0;

  if (controller_wiimote.button == WII_LEFT ||
controller_wiimote.button == WII_LEFT+WII_DOWN ||
controller_wiimote.button == WII_LEFT+WII_UP){
    hp_distance = float(0.015 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (controller_wiimote.button == WII_RIGHT ||
controller_wiimote.button == WII_RIGHT+WII_DOWN ||
controller_wiimote.button == WII_RIGHT+WII_UP){
    hp_distance = float(-0.015 * JOYSTICK_SPEED_MULTIPLIER);
```

```
  }
  if (controller_wiimote.button == WII_DOWN ||
controller_wiimote.button == WII_LEFT+WII_DOWN ||
controller_wiimote.button == WII_RIGHT+WII_DOWN){
    turn_rad = float(-0.0001 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (controller_wiimote.button == WII_UP ||
controller_wiimote.button == WII_LEFT+WII_UP ||
controller_wiimote.button == WII_RIGHT+WII_UP){
    turn_rad = float(0.0001 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (controller_wiimote.button == WII_LEFT+WII_B){
    z_distance = float (0.015 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (controller_wiimote.button == WII_RIGHT+WII_B){
    z_distance = float (-0.015 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (controller_wiimote.button == WII_DOWN+WII_B){
    e_distance = float (-0.008 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (controller_wiimote.button == WII_UP+WII_B){
    e_distance = float (0.008 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (abs(hp_distance) > 0.0){
    float hp_ratio = hp_distance / geometry.getHypot();
    x_distance += interpolator.getXPosmm() * hp_ratio;
    y_distance += interpolator.getYPosmm() * hp_ratio;
  }
  if (abs(turn_rad) > 0.0){
    x_distance += sin(geometry.getRotRad() + turn_rad) *
geometry.getHypot() - interpolator.getXPosmm();
    y_distance += cos(geometry.getRotRad() + turn_rad) *
geometry.getHypot() - interpolator.getYPosmm();
  }

  if (x_distance || y_distance || z_distance || e_distance){
    interpolator.speed_profile = 0;

interpolator.setInterpolation(interpolator.getXPosmm()+x_distance,
interpolator.getYPosmm()+y_distance,
interpolator.getZPosmm()+z_distance,
interpolator.getEPosmm()+e_distance, 5);
    //SOLUTION FOR WIIMOTE Y<0.05 FREEZE ISSUE
    if (interpolator.getYPosmm() < 0.5) {
      interpolator.setInterpolation(interpolator.getXPosmm(), 0.5,
interpolator.getZPosmm(), interpolator.getEPosmm(), 5);
    }
  }
}
#endif
```

Lampiran 4. Deskripsi Program *Firmware* Robot Lengan

Program ini merupakan firmware utama untuk sistem robot lengan berbasis Arduino Mega 2560 Pro Mini yang dirancang untuk menerima perintah G-code. Program ini memiliki struktur modular yang mengintegrasikan berbagai komponen seperti motor stepper, *sensor hall effect* (sebagai endstop), *gripper* (dapat menggunakan servo), dan sistem *rail* tambahan (sumbu E).

Fungsi utama dari program ini adalah untuk:

1. Mengatur interpolasi gerakan lengan robot secara halus dan presisi melalui objek Interpolation
2. Mengontrol masing-masing motor melalui objek *RampsStepper*
3. Mengatur eksekusi perintah G-code melalui sistem antrean
4. Menyediakan fitur homing otomatis dengan deteksi sensor hall effect

Dalam pengembangan sistem kendali lengan robot ini, program disusun secara modular untuk memisahkan fungsi-fungsi utama ke dalam beberapa file dengan tanggung jawab spesifik. File utama (.ino) berfungsi sebagai pusat kendali yang menjalankan loop utama dan mengatur jalannya seluruh sistem. Sementara itu, berbagai file pendukung dibagi dalam bentuk modul untuk memudahkan pemeliharaan dan pengembangan.

Struktur File Program

1. config.h

   File konfigurasi global yang berisi berbagai definisi parameter sistem, pengaturan kecepatan, logika arah motor, dan lainnya. File ini memungkinkan pengaturan dilakukan secara terpusat tanpa perlu mengubah kode logika utama.
2. RampsStepper.h dan RampsStepper.cpp

Berisi deklarasi dan implementasi kelas RampsStepper yang mengelola pengendalian motor stepper menggunakan Arduino Mega 2560 Pro Mini. Modul ini menangani fungsi-fungsi penting seperti:

I. Pengaturan posisi motor

II. Perhitungan konversi dari sudut ke langkah (step)

III. Pengaktifan motor

3. endstop.h dan endstop.cpp

   Bertugas mengatur proses homing motor dengan membaca sensor hall effect sebagai batas gerakan. Kelas Endstop pada file ini memastikan motor berhenti saat mencapai titik referensi.

4. interpolation.h dan interpolation.cpp

   Berfungsi untuk menghitung lintasan interpolasi antar titik, memastikan pergerakan end-effector lengan robot berlangsung secara halus dan teratur. Modul ini juga menangani sistem koordinat dan pergeseran posisi.

5. pinout.h

   Berisi konfigurasi pin input/output sesuai dengan pin bawaan Arduino Mega 2560 Pro Mini. File ini menjadi referensi utama yang digunakan seluruh modul untuk menentukan pin kontrol motor, sensor, kipas, dan aktuator lainnya.

Lampiran 5. Spesifikasi Motor Stepper

SPESIFIKASI MOTOR STEPPER NEMA 17 HS6401

Parameter	Nilai	Satuan
Model	17HS6401	-
Tipe Motor	Hybrid Stepper Motor	-
Standar	NEMA 17	-
Holding Torque	70	N.cm
Holding Torque	7	Kg.cm
Holding Torque	0.7	N.M
Step Angle	1.8	degree
Steps per Revolution	200	steps
Rated Current	1.7	A
Dimensi Body	42 x 42 x 60	mm
Diameter Shaft	5	mm
Length Shaft	23.5±1	mm
Inductance	2.8	mH
Wire	4	pin
Length Wire	±170	cm
Motor Weight	450	gram
Type	Hybrid	-

SPESIFIKASI MOTOR STEPPER NEMA 17 HS3401

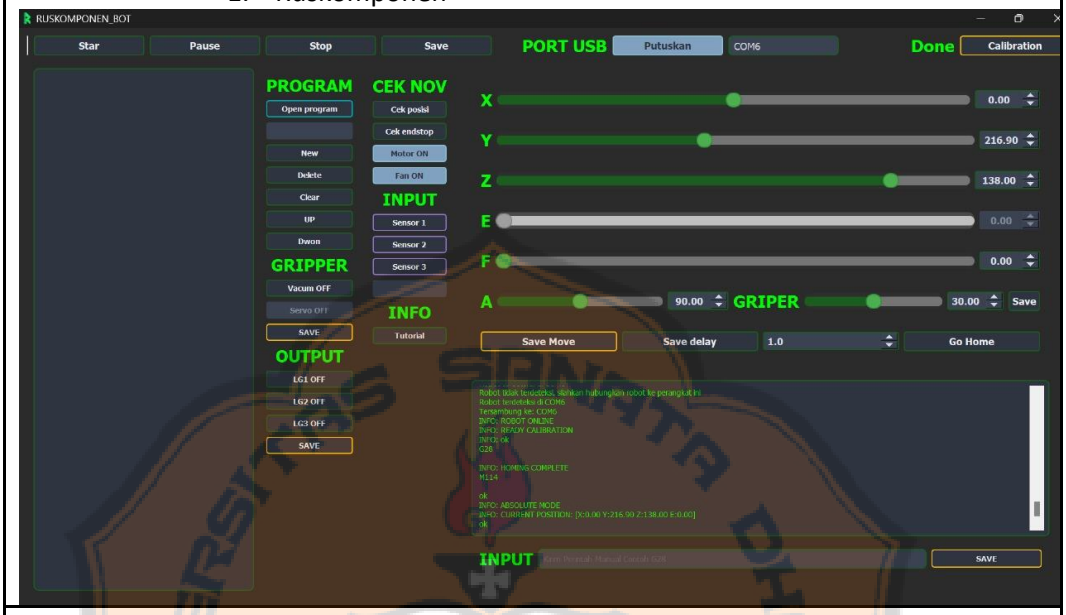
Parameter	Nilai	Satuan
Model	17HS3401	-
Features	Low noise, Low fever, Smooth operation, Good acceleration	-
Step Angle	1.8°	degree
Steps per Revolution	200	steps
Phase Current	1.3	A
Phase Resistance	2.4	ohm
Phase Inductance	2.8	mH
Holding Torque	28	N.cm
Braking Torque	1.6	N.cm
Rotor Inertia	34	g.cm²
Number of Leads	4	-
Motor Weight	220	g
Body Length	34	mm

Parameter	Nilai	Satuan
Frame Size	42 x 42	mm
Step Angle Accuracy	±5%	full step, no load
Resistance Accuracy	±10%	-
Inductance Accuracy	±20%	-
Temperature Rise	80	°C Max
Operating Temperature	-20 to +50	°C
Insulation Resistance	100M	ohm Min
Electric Strength	500VDC, 820VAC	1 minute
Radial Clearance	0.02	Max (450g load)
Axial Clearance	0.08	Max (450g load)
Adapted Driver	Two-phase stepper driver	A4988, DRV8825, TB6600

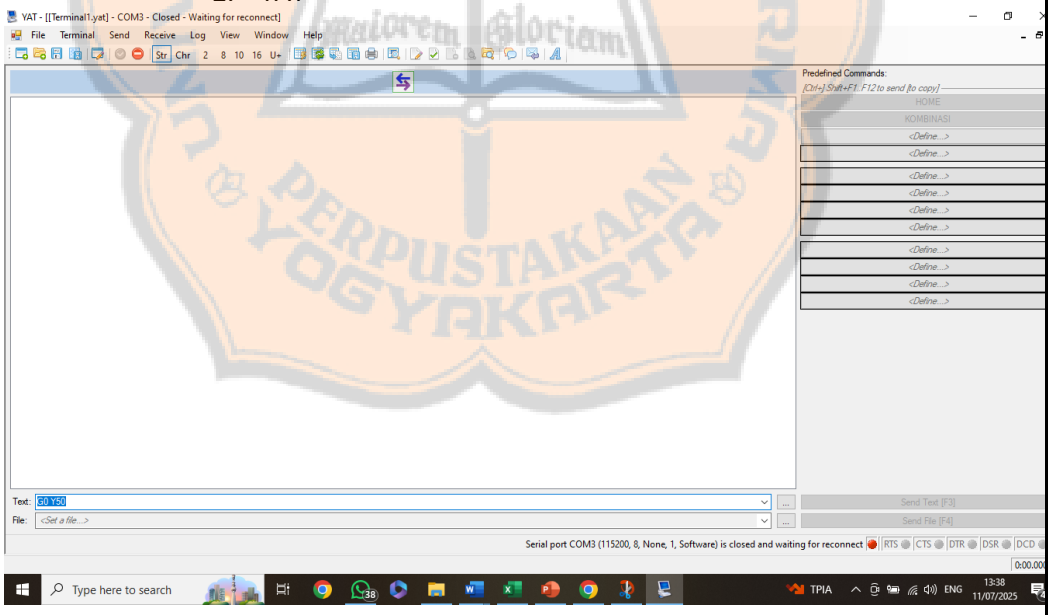
Lampiran 6. List Software Yang Digunakan

SoftWare Yang Digunakan

1. Ruskomponen



2. YAT



Lampiran 7. *List* Program *Arduino* Keseluruhan

Main Program	*CPP Files*	*H Files*
robotArm_v0.83	*byj_gripper.cpp*	*byj_gripper.h*
	command.cpp	*command.h*
	controller_ps4.cpp	*config.h*
	controller_wiimote.cpp	*config_esp32.h*
	endstop.cpp	*controller_ps4.h*
	equipment.cpp	*controller_wiimote.h*
	fanControl.cpp	*endstop.h*
	interpolation.cpp	*equipment.h*
	logger.cpp	*fanControl.h*
	RampsStepper.cpp	*interpolation.h*
	robotGeometry.cpp	*logger.h*
	servo_gripper.cpp	*pinout.h*
		queue.h
		RampsStepper.h
		robotGeometry.h
		servo_gripper.h

robotArm_v0.83

```
#include <Arduino.h>
//GENERAL CONFIG SETTINGS
#include "config.h"

#include "robotGeometry.h"
#include "interpolation.h"
#include "RampsStepper.h"
#include "queue.h"
#include "command.h"
#include "equipment.h"
#include "endstop.h"
#include "logger.h"
#include "fanControl.h"
//INCLUDE CORRESPONDING GRIPPER MOTOR CLASS
#if GRIPPER == SERVO
  #include "servo_gripper.h"
#elif GRIPPER == BYJ
  #include "byj_gripper.h"
#endif
  #include "pinout/pinout.h"

//STEPPER OBJECTS
RampsStepper stepperHigher(Z_STEP_PIN, Z_DIR_PIN, Z_ENABLE_PIN,
INVERSE_Z_STEPPER, MAIN_GEAR_TEETH, MOTOR_GEAR_TEETH, MICROSTEPS,
STEPS_PER_REV);
RampsStepper stepperLower(Y_STEP_PIN, Y_DIR_PIN, Y_ENABLE_PIN,
INVERSE_Y_STEPPER, MAIN_GEAR_TEETH, MOTOR_GEAR_TEETH, MICROSTEPS,
STEPS_PER_REV);
```

```
RampsStepper stepperRotate(X_STEP_PIN, X_DIR_PIN, X_ENABLE_PIN,
INVERSE_X_STEPPER, MAIN_GEAR_TEETH, MOTOR_GEAR_TEETH, MICROSTEPS,
STEPS_PER_REV);

//RAIL OBJECTS
#if RAIL
  RampsStepper stepperRail(E0_STEP_PIN, E0_DIR_PIN, E0_ENABLE_PIN,
INVERSE_E0_STEPPER, MAIN_GEAR_TEETH, MOTOR_GEAR_TEETH, MICROSTEPS,
STEPS_PER_REV);
  #if BOARD_CHOICE == WEMOSD1R32 //PINSWAP REQIURED ON D1R32 DUE
TO INSUFFICIENT DIGIAL PINS
    #define SERVO_PIN 23 // REDEFINE SERVO_PIN FOR RAIL // SHARE
WITH Z_MIN_PIN
    Endstop endstopE0(E0_MIN_PIN, E0_DIR_PIN, E0_STEP_PIN,
E0_ENABLE_PIN, E0_MIN_INPUT, E0_HOME_STEPS, HOME_DWELL, true);
  #else
    Endstop endstopE0(E0_MIN_PIN, E0_DIR_PIN, E0_STEP_PIN,
E0_ENABLE_PIN, E0_MIN_INPUT, E0_HOME_STEPS, HOME_DWELL, false);
  #endif
#endif

//ENDSTOP OBJECTS
Endstop endstopX(X_MIN_PIN, X_DIR_PIN, X_STEP_PIN, X_ENABLE_PIN,
X_MIN_INPUT, X_HOME_STEPS, HOME_DWELL, false);
Endstop endstopY(Y_MIN_PIN, Y_DIR_PIN, Y_STEP_PIN, Y_ENABLE_PIN,
Y_MIN_INPUT, Y_HOME_STEPS, HOME_DWELL, false);
#if BOARD_CHOICE == WEMOSD1R32
  Endstop endstopZ(Z_MIN_PIN, Z_DIR_PIN, Z_STEP_PIN, Z_ENABLE_PIN,
Z_MIN_INPUT, Z_HOME_STEPS, HOME_DWELL, true);
#else
  Endstop endstopZ(Z_MIN_PIN, Z_DIR_PIN, Z_STEP_PIN, Z_ENABLE_PIN,
Z_MIN_INPUT, Z_HOME_STEPS, HOME_DWELL, false);
#endif

//EQUIPMENT OBJECTS
#if GRIPPER == SERVO
  Servo_Gripper servo_gripper(SERVO_PIN, SERVO_GRIP_DEGREE,
SERVO_UNGRIP_DEGREE);
#elif GRIPPER == BYJ
  BYJ_Gripper byj_gripper(BYJ_PIN_0, BYJ_PIN_1, BYJ_PIN_2,
BYJ_PIN_3, BYJ_GRIP_STEPS);
#endif
Equipment lg1(LG1_PIN);
Equipment lg2(LG2_PIN);
Equipment lg3(LG3_PIN);
Equipment led(LED_PIN);
FanControl fan(FAN_PIN, FAN_DELAY);

//EXECUTION & COMMAND OBJECTS
RobotGeometry geometry(END_EFFECTOR_OFFSET, LOW_SHANK_LENGTH,
HIGH_SHANK_LENGTH);
Interpolation interpolator;
Queue<Cmd> queue(QUEUE_SIZE);
Command command;

//PS4 CONTROLLER OBJECT FOR ESP32
```

```
#if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == DUALSHOCK4
  #include "controller_ps4.h"
  Controller_PS4 controller_ps4(PS4_MAC);
#endif
#if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == WIIMOTE
  #include "controller_wiimote.h"
  Controller_Wiimote controller_wiimote;
#endif

//--------------------------------IO-----------------------------
--------------
int IO1Before = LOW;
int IO2Before = LOW;
int IO3Before = LOW;
//-----------------------------------------------------------------
--------------

void setup()
{
  Serial.begin(BAUD);
  #if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == DUALSHOCK4
    controller_ps4.setup();
  #endif
  #if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == WIIMOTE
    controller_wiimote.setup();
  #endif
  stepperHigher.setPositionRad(PI / 2.0); // 90°
  stepperLower.setPositionRad(0);         // 0°
  stepperRotate.setPositionRad(0);        // 0°
  #if RAIL
  stepperRail.setPosition(0);
  #endif
  if (HOME_ON_BOOT) { //HOME DURING SETUP() IF HOME_ON_BOOT
ENABLED
    homeSequence();
    Logger::logINFO("ROBOT ONLINE");
  } else {
    setStepperEnable(false); //ROBOT ADJUSTABLE BY HAND AFTER
TURNING ON
    if (HOME_X_STEPPER && HOME_Y_STEPPER && !HOME_Z_STEPPER){
      Logger::logINFO("ROBOT ONLINE");
      Logger::logINFO("ROTATE ROBOT TO FACE FRONT CENTRE & SEND
G28 TO CALIBRATE");
    }
    if (HOME_X_STEPPER && HOME_Y_STEPPER && HOME_Z_STEPPER){
      Logger::logINFO("ROBOT ONLINE");
      Logger::logINFO("READY CALIBRATION");
      Logger::logINFO("ok");
    }
    if (!HOME_X_STEPPER && !HOME_Y_STEPPER){
      Logger::logINFO("ROBOT ONLINE");
      Logger::logINFO("HOME ROBOT MANUALLY & SEND G28 TO
CALIBRATE");
    }
  }
```

```
  interpolator.setInterpolation(INITIAL_X, INITIAL_Y, INITIAL_Z,
INITIAL_E0, INITIAL_X, INITIAL_Y, INITIAL_Z, INITIAL_E0);

//------------------------------------IO-------------------------
-----------------------
  pinMode(IO1_PIN, INPUT);
  pinMode(IO2_PIN, INPUT);
  pinMode(IO3_PIN, INPUT);
//----------------------------------------------------------------
-------------------------

}

void loop() {
  interpolator.updateActualPosition();
  geometry.set(interpolator.getXPosmm(), interpolator.getYPosmm(),
interpolator.getZPosmm());
  stepperRotate.stepToPositionRad(geometry.getRotRad());
  stepperLower.stepToPositionRad(geometry.getLowRad());
  stepperHigher.stepToPositionRad(geometry.getHighRad());
  #if RAIL
    stepperRail.stepToPositionMM(interpolator.getEPosmm(),
STEPS_PER_MM_RAIL);
  #endif
  stepperRotate.update();
  stepperLower.update();
  stepperHigher.update();
  #if RAIL
    stepperRail.update();
  #endif
  fan.update();

  if (!queue.isFull()) {
    if (command.handleGcode()) {
    Cmd cmd = command.getCmd();
    if (cmd.id == 'M' && cmd.num == 119) {
      Logger::logINFO("ENDSTOP STATES: [X:" +
String(endstopX.state()) +
                      " Y:" + String(endstopY.state()) +
                      " Z:" + String(endstopZ.state()) + "]");
      return;
    }

      queue.push(command.getCmd());
    }
  }
  if ((!queue.isEmpty()) && interpolator.isFinished()) {
    executeCommand(queue.pop());
    if (PRINT_REPLY) {
      Serial.println(PRINT_REPLY_MSG);
    }
  }

  if (millis() % 500 < 250) {
    led.cmdOn();
  }
```

```
  else {
    led.cmdOff();
  }

  #if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == DUALSHOCK4
    ps4_controller_loop();
  #endif
  #if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == WIIMOTE
    wiimote_controller_loop();
  #endif

 //============== SENSOR
============================================================
  if (digitalRead(IO1_PIN) == HIGH && IO1Before == LOW) {
      Logger::logINFO("S1 ON");
      IO1Before = HIGH;
  } else if (digitalRead(IO1_PIN) == LOW && IO1Before == HIGH) {
      Logger::logINFO("S1 OFF");
      IO1Before = LOW;
  }

  if (digitalRead(IO2_PIN) == HIGH && IO2Before == LOW) {
      Logger::logINFO("S2 ON");
      IO2Before = HIGH;
  } else if (digitalRead(IO2_PIN) == LOW && IO2Before == HIGH) {
      Logger::logINFO("S2 OFF");
      IO2Before = LOW;
  }

  if (digitalRead(IO3_PIN) == HIGH && IO3Before == LOW) {
      Logger::logINFO("S3 ON");
      IO3Before = HIGH;
  } else if (digitalRead(IO3_PIN) == LOW && IO3Before == HIGH) {
      Logger::logINFO("S3 OFF");
      IO3Before = LOW;
  }
}

void executeCommand(Cmd cmd) {

  if (cmd.id == -1) {
    printErr();
    return;
  }
  if (cmd.id == 'G') {
    switch (cmd.num) {
    case 0:
    case 1:
      fan.enable(true);
      Point posoffset;
      posoffset = interpolator.getPosOffset();
      cmdMove(cmd, interpolator.getPosmm(), posoffset,
command.isRelativeCoord);
      interpolator.setInterpolation(cmd.valueX, cmd.valueY,
cmd.valueZ, cmd.valueE, cmd.valueF);
```

```
      Logger::logINFO("LINEAR MOVE: [X:" + String(cmd.valueX-
posoffset.xmm) + " Y:" + String(cmd.valueY-posoffset.ymm) + " Z:"
+ String(cmd.valueZ-posoffset.zmm) + " E:" + String(cmd.valueE-
posoffset.emm)+"]");
    case 4: cmdDwell(cmd); break;
    case 28:
      if (BOARD_CHOICE == UNO || BOARD_CHOICE == WEMOSD1R32){
        homeSequence_UNO();
        break;
      } else {
        homeSequence();
        break;
      }
    case 90: command.cmdToAbsolute(); break; // ABSOLUTE
COORDINATE MODE
    case 91: command.cmdToRelative(); break; // RELATIVE
COORDINATE MODE
    case 92:
      interpolator.resetPosOffset();
      cmdMove(cmd, interpolator.getPosmm(),
interpolator.getPosOffset(), false);
      interpolator.setPosOffset(cmd.valueX, cmd.valueY,
cmd.valueZ, cmd.valueE);
      break;
    default: printErr();
    }
  }
  else if (cmd.id == 'M') {
    switch (cmd.num) {
    case 1: lg1.cmdOn(); break;
    case 2: lg1.cmdOff(); break;
    case 3:
      #if GRIPPER == BYJ
        byj_gripper.cmdOn(); break;
      #elif GRIPPER == SERVO
        servo_gripper.cmdOn(); break;
      #endif
    case 5:
      #if GRIPPER == BYJ
        byj_gripper.cmdOff(); break;
      #elif GRIPPER == SERVO
        servo_gripper.cmdOff(); break;
      #endif
    case 6: lg2.cmdOn(); break;
    case 7: lg2.cmdOff(); break;
    case 17: setStepperEnable(true); break;
    case 18: setStepperEnable(false); break;
    case 106: fan.enable(true); break;
    case 107: fan.enable(false); break;
    case 114: command.cmdGetPosition(interpolator.getPosmm(),
interpolator.getPosOffset(), stepperHigher.getPosition(),
stepperLower.getPosition(), stepperRotate.getPosition()); break;//
Return the current positions of all axis
    case 119: {
      String endstopMsg = "ENDSTOP: [X:";
      endstopMsg += String(endstopX.state());
```

```
      endstopMsg += " Y:";
      endstopMsg += String(endstopY.state());
      endstopMsg += " Z:";
      endstopMsg += String(endstopZ.state());
      #if RAIL
        endstopMsg += " E:";
        endstopMsg += String(endstopE0.state());
      #endif
      endstopMsg += "]";
      //ORIGINAL LOG STRING UNDESIRABLE FOR UNO PROCESSING
      //Logger::logINFO("ENDSTOP STATE:
[UPPER_SHANK(X):"+String(endstopX.state())+"
LOWER_SHANK(Y):"+String(endstopY.state())+"
ROTATE_GEAR(Z):"+String(endstopZ.state())+"]");
      Logger::logINFO(endstopMsg);
      break;}
    case 205:
      interpolator.setSpeedProfile(cmd.valueS);
      Logger::logINFO("SPEED PROFILE: [" +
String(interpolator.speed_profile) + "]");
      break;
    case 206: lg3.cmdOn(); break;
    case 207: lg3.cmdOff(); break;
    default: printErr();
    }
  }
  else {
    printErr();
  }
}

void setStepperEnable(bool enable){
  stepperRotate.enable(enable);
  stepperLower.enable(enable);
  stepperHigher.enable(enable);
  #if RAIL
    stepperRail.enable(enable);
  #endif
  fan.enable(enable);
}

void homeSequence(){
  setStepperEnable(false);
  fan.enable(true);
  if (HOME_Y_STEPPER && HOME_X_STEPPER){
    endstopY.home(!INVERSE_Y_STEPPER);
    endstopX.home(!INVERSE_X_STEPPER);
  } else {
    setStepperEnable(true);
    endstopY.homeOffset(!INVERSE_Y_STEPPER);
    endstopX.homeOffset(!INVERSE_X_STEPPER);
  }
  if (HOME_Z_STEPPER){
    endstopZ.home(INVERSE_Z_STEPPER);
  }
  #if RAIL
```

```
    if (HOME_E0_STEPPER){
      endstopE0.home(!INVERSE_E0_STEPPER);
    }
  #endif
  interpolator.setInterpolation(INITIAL_X, INITIAL_Y, INITIAL_Z,
INITIAL_E0, INITIAL_X, INITIAL_Y, INITIAL_Z, INITIAL_E0);
  Logger::logINFO("HOMING COMPLETE");
}

//DUE TO UNO CNC SHIELD LIMIT, 1 EN PIN SERVES 3 MOTORS, HENCE
DIFFERENT HOMESEQUENCE IS REQUIRED
void homeSequence_UNO(){
  #if GRIPPER == SERVO
  if (servo_gripper.readDegree() != SERVO_UNGRIP_DEGREE){
    servo_gripper.cmdOff();
  }
  #endif
  if (HOME_Y_STEPPER && HOME_X_STEPPER){
    while (!endstopY.state() || !endstopX.state()){
      endstopY.oneStepToEndstop(!INVERSE_Y_STEPPER);
      endstopX.oneStepToEndstop(!INVERSE_X_STEPPER);
    }
    endstopY.homeOffset(!INVERSE_Y_STEPPER);
    endstopX.homeOffset(!INVERSE_X_STEPPER);
  } else {
    setStepperEnable(true);
    endstopY.homeOffset(!INVERSE_Y_STEPPER);
    endstopX.homeOffset(!INVERSE_X_STEPPER);
  }
  if (HOME_Z_STEPPER){
    endstopZ.home(INVERSE_Z_STEPPER); //INDICATE STEPPER HOMING
DIRECDTION
  }
  #if RAIL
    if (HOME_E0_STEPPER){
      endstopE0.home(!INVERSE_E0_STEPPER);
    }
  #endif
  interpolator.setInterpolation(INITIAL_X, INITIAL_Y, INITIAL_Z,
INITIAL_E0, INITIAL_X, INITIAL_Y, INITIAL_Z, INITIAL_E0);
  Logger::logINFO("HOMING COMPLETE");
}

#if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == DUALSHOCK4
void ps4_controller_loop(){
  controller_ps4.update();
  interpolator.speed_profile = 0;
  if (controller_ps4.buttons[PS4_CROSS]){if (GRIPPER ==
SERVO){servo_gripper.cmdOn();}}
  if (controller_ps4.buttons[PS4_CIRCLE]){if (GRIPPER ==
SERVO){servo_gripper.cmdOff();}}
  if
(controller_ps4.buttons[PS4_OPTIONS]){setStepperEnable(true);}
  if (controller_ps4.buttons[PS4_SHARE]){setStepperEnable(false);}
  if (controller_ps4.buttons[PS4_TOUCHPAD]){homeSequence_UNO();}
  float x_distance = 0.0;
```

```
  float y_distance = 0.0;
  float z_distance = 0.0;
  float e_distance = 0.0;
  if (abs(controller_ps4.buttons[PS4_RSTICKY]) > 10){
//    z_distance = float(controller_ps4.buttons[PS4_RSTICKY]) /
2500.0;
    z_distance =
float(controller_ps4.buttons[PS4_RSTICKY])*JOYSTICK_SPEED_MULTIPLI
ER / 2500.0;
  }
  if (abs(controller_ps4.buttons[PS4_LSTICKX]) > 10){
    float turn_rad =
float(controller_ps4.buttons[PS4_LSTICKX])*JOYSTICK_SPEED_MULTIPLI
ER / 300000.0;
    x_distance += sin(geometry.getRotRad() + turn_rad) *
geometry.getHypot() - interpolator.getXPosmm();
    y_distance += cos(geometry.getRotRad() + turn_rad) *
geometry.getHypot() - interpolator.getYPosmm();
  }
  if (abs(controller_ps4.buttons[PS4_LSTICKY]) > 10){
    float hp_distance =
float(controller_ps4.buttons[PS4_LSTICKY])*JOYSTICK_SPEED_MULTIPLI
ER / 2500.0;
    float hp_ratio = hp_distance / geometry.getHypot();
    x_distance += interpolator.getXPosmm() * hp_ratio;
    y_distance += interpolator.getYPosmm() * hp_ratio;
  }
  if (abs(controller_ps4.buttons[PS4_L2VALUE]) > 10){
    e_distance -=
float(controller_ps4.buttons[PS4_L2VALUE])*JOYSTICK_SPEED_MULTIPLI
ER/ 10000.0;
  }
  if (abs(controller_ps4.buttons[PS4_R2VALUE]) > 10){
    e_distance +=
float(controller_ps4.buttons[PS4_R2VALUE])*JOYSTICK_SPEED_MULTIPLI
ER/ 10000.0;
  }
  if (x_distance || y_distance || z_distance || e_distance){
    interpolator.speed_profile = 0;
    interpolator.setInterpolation(interpolator.getXPosmm()+x_dista
nce, interpolator.getYPosmm()+y_distance,
interpolator.getZPosmm()+z_distance,
interpolator.getEPosmm()+e_distance, 5);
  }
}
#endif

#if BOARD_CHOICE == WEMOSD1R32 && ESP32_JOYSTICK == WIIMOTE
void wiimote_controller_loop(){
  controller_wiimote.update();
  interpolator.speed_profile = 0;
  if (controller_wiimote.button == WII_A && GRIPPER == SERVO){
    if (!servo_gripper.isOn()){
      servo_gripper.cmdOn();
    } else {
      servo_gripper.cmdOff();
```

```
    }
  }
  if (controller_wiimote.button ==
WII_PLUS){setStepperEnable(true);}
  if (controller_wiimote.button ==
WII_MINUS){setStepperEnable(false);}
  if (controller_wiimote.button == WII_HOME){homeSequence_UNO();}

  float x_distance = 0.0;
  float y_distance = 0.0;
  float z_distance = 0.0;
  float e_distance = 0.0;
  float hp_distance = 0.0;
  float turn_rad = 0.0;

  if (controller_wiimote.button == WII_LEFT ||
controller_wiimote.button == WII_LEFT+WII_DOWN ||
controller_wiimote.button == WII_LEFT+WII_UP){
    hp_distance = float(0.015 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (controller_wiimote.button == WII_RIGHT ||
controller_wiimote.button == WII_RIGHT+WII_DOWN ||
controller_wiimote.button == WII_RIGHT+WII_UP){
    hp_distance = float(-0.015 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (controller_wiimote.button == WII_DOWN ||
controller_wiimote.button == WII_LEFT+WII_DOWN ||
controller_wiimote.button == WII_RIGHT+WII_DOWN){
    turn_rad = float(-0.0001 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (controller_wiimote.button == WII_UP ||
controller_wiimote.button == WII_LEFT+WII_UP ||
controller_wiimote.button == WII_RIGHT+WII_UP){
    turn_rad = float(0.0001 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (controller_wiimote.button == WII_LEFT+WII_B){
    z_distance = float (0.015 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (controller_wiimote.button == WII_RIGHT+WII_B){
    z_distance = float (-0.015 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (controller_wiimote.button == WII_DOWN+WII_B){
    e_distance = float (-0.008 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (controller_wiimote.button == WII_UP+WII_B){
    e_distance = float (0.008 * JOYSTICK_SPEED_MULTIPLIER);
  }
  if (abs(hp_distance) > 0.0){
    float hp_ratio = hp_distance / geometry.getHypot();
    x_distance += interpolator.getXPosmm() * hp_ratio;
    y_distance += interpolator.getYPosmm() * hp_ratio;
  }
  if (abs(turn_rad) > 0.0){
    x_distance += sin(geometry.getRotRad() + turn_rad) *
geometry.getHypot() - interpolator.getXPosmm();
```

```
    y_distance += cos(geometry.getRotRad() + turn_rad) *
geometry.getHypot() - interpolator.getYPosmm();
  }

  if (x_distance || y_distance || z_distance || e_distance){
    interpolator.speed_profile = 0;
    interpolator.setInterpolation(interpolator.getXPosmm()+x_dista
nce, interpolator.getYPosmm()+y_distance,
interpolator.getZPosmm()+z_distance,
interpolator.getEPosmm()+e_distance, 5);
    //SOLUTION FOR WIIMOTE Y<0.05 FREEZE ISSUE
    if (interpolator.getYPosmm() < 0.5) {
      interpolator.setInterpolation(interpolator.getXPosmm(), 0.5,
interpolator.getZPosmm(), interpolator.getEPosmm(), 5);
    }
  }
}
#endif
```

RampsStepper.cpp

```
#include <Arduino.h>
#include "RampsStepper.h"

RampsStepper::RampsStepper(int aStepPin, int aDirPin, int
aEnablePin, bool aInverse, float main_gear_teeth, float
motor_gear_teeth, int microsteps, int steps_per_rev) {
  setReductionRatio(main_gear_teeth / motor_gear_teeth, microsteps
* steps_per_rev);
  stepPin = aStepPin;
  dirPin = aDirPin;
  enablePin = aEnablePin;
  inverse = aInverse;
  stepperStepPosition = 0;
  stepperStepTargetPosition;
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
  pinMode(enablePin, OUTPUT);
  enable(false);
}

void RampsStepper::enable(bool value) {
  digitalWrite(enablePin, !value);
}

bool RampsStepper::isOnPosition() const {
  return stepperStepPosition == stepperStepTargetPosition;
}

int RampsStepper::getPosition() const {
  return stepperStepPosition;
}

void RampsStepper::setPosition(int value) {
  stepperStepPosition = value;
  stepperStepTargetPosition = value;
}
```

```
void RampsStepper::stepToPosition(int value) {
  stepperStepTargetPosition = value;
}

void RampsStepper::stepToPositionMM(float mm, float steps_per_mm)
{
  stepperStepTargetPosition = mm * steps_per_mm;
}

void RampsStepper::stepRelative(int value) {
  value += stepperStepPosition;
  stepToPosition(value);
}

float RampsStepper::getPositionRad() const {
  return stepperStepPosition / radToStepFactor;
}

void RampsStepper::setPositionRad(float rad) {
  setPosition(rad * radToStepFactor);
}

void RampsStepper::stepToPositionRad(float rad) {
  stepperStepTargetPosition = rad * radToStepFactor;
}

void RampsStepper::stepRelativeRad(float rad) {
  stepRelative(rad * radToStepFactor);
}

void RampsStepper::update() {
  while (stepperStepTargetPosition < stepperStepPosition) {
    digitalWrite(dirPin, !inverse);
    digitalWrite(stepPin, HIGH);
    digitalWrite(stepPin, LOW);
    stepperStepPosition--;
  }

  while (stepperStepTargetPosition > stepperStepPosition) {
    digitalWrite(dirPin, inverse);
    digitalWrite(stepPin, HIGH);
    digitalWrite(stepPin, LOW);
    stepperStepPosition++;
  }
}

// void RampsStepper::update() {
//   while (stepperStepTargetPosition < stepperStepPosition) {
//     digitalWrite(dirPin, !inverse);
//     digitalWrite(stepPin, HIGH);
//     delayMicroseconds(800);  // ← Tambahkan delay mikro
//     digitalWrite(stepPin, LOW);
//     delayMicroseconds(800);  // ← Tambahkan delay mikro
//     stepperStepPosition--;
//   }
```

```
//   while (stepperStepTargetPosition > stepperStepPosition) {
//     digitalWrite(dirPin, inverse);
//     digitalWrite(stepPin, HIGH);
//     delayMicroseconds(800);  // ← Tambahkan delay mikro
//     digitalWrite(stepPin, LOW);
//     delayMicroseconds(800);  // ← Tambahkan delay mikro
//     stepperStepPosition++;
//   }
// }

void RampsStepper::setReductionRatio(float gearRatio, int
stepsPerRev) {
  radToStepFactor = gearRatio * stepsPerRev / 2 / PI;
};
```

RampsStepper.h

```
#ifndef RAMPSSTEPPER_H_
#define RAMPSSTEPPER_H_

class RampsStepper {
public:
  RampsStepper(int aStepPin, int aDirPin, int aEnablePin, bool
aInverse, float main_gear_teeth, float motor_gear_teeth, int
microsteps, int steps_per_rev);
  void enable(bool value = true);

  bool isOnPosition() const;
  int getPosition() const;
  void setPosition(int value);
  void stepToPosition(int value);
  void stepToPositionMM(float mm, float steps_per_mm);
  void stepRelative(int value);
  float getPositionRad() const;
  void setPositionRad(float rad);
  void stepToPositionRad(float rad);
  void stepRelativeRad(float rad);

  void update();

  void setReductionRatio(float gearRatio, int stepsPerRev);
private:
  int stepperStepTargetPosition;
  int stepperStepPosition;
  int stepPin;
  int dirPin;
  int enablePin;
  bool inverse;
  float radToStepFactor;
};

#endif
```

byj_gripper.cpp

```
#include "byj_gripper.h"
#include <Arduino.h>
```

```
BYJ_Gripper::BYJ_Gripper(int pin0, int pin1, int pin2, int pin3,
int steps){
  grip_steps = steps;
  byj_pin_0 = pin0;
  byj_pin_1 = pin1;
  byj_pin_2 = pin2;
  byj_pin_3 = pin3;
  step_cycle = 0;
  pinMode(byj_pin_0, OUTPUT);
  pinMode(byj_pin_1, OUTPUT);
  pinMode(byj_pin_2, OUTPUT);
  pinMode(byj_pin_3, OUTPUT);
}

void BYJ_Gripper::cmdOn() {
  direction = true;
  for (int i = 1; i <= grip_steps; i++) {
    moveSteps();
    delay(1);
  }
}

void BYJ_Gripper::cmdOff() {
  direction = false;
  for (int i = 1; i <= grip_steps; i++) {
    moveSteps();
    delay(1);
  }
}

void BYJ_Gripper::setDirection(){
  if (direction == true) {
    step_cycle++;
  }
  if (direction == false) {
    step_cycle--;
  }
  if (step_cycle > 7) {
    step_cycle = 0;
  }
  if (step_cycle < 0) {
    step_cycle = 7;
  }
}

void BYJ_Gripper::moveSteps() {
  switch (step_cycle) {
  case 0:
    digitalWrite(byj_pin_0, LOW);
    digitalWrite(byj_pin_1, LOW);
    digitalWrite(byj_pin_2, LOW);
    digitalWrite(byj_pin_3, HIGH);
    break;
  case 1:
    digitalWrite(byj_pin_0, LOW);
```

```
    digitalWrite(byj_pin_1, LOW);
    digitalWrite(byj_pin_2, HIGH);
    digitalWrite(byj_pin_3, HIGH);
    break;
  case 2:
    digitalWrite(byj_pin_0, LOW);
    digitalWrite(byj_pin_1, LOW);
    digitalWrite(byj_pin_2, HIGH);
    digitalWrite(byj_pin_3, LOW);
    break;
  case 3:
    digitalWrite(byj_pin_0, LOW);
    digitalWrite(byj_pin_1, HIGH);
    digitalWrite(byj_pin_2, HIGH);
    digitalWrite(byj_pin_3, LOW);
    break;
  case 4:
    digitalWrite(byj_pin_0, LOW);
    digitalWrite(byj_pin_1, HIGH);
    digitalWrite(byj_pin_2, LOW);
    digitalWrite(byj_pin_3, LOW);
    break;
  case 5:
    digitalWrite(byj_pin_0, HIGH);
    digitalWrite(byj_pin_1, HIGH);
    digitalWrite(byj_pin_2, LOW);
    digitalWrite(byj_pin_3, LOW);
    break;
  case 6:
    digitalWrite(byj_pin_0, HIGH);
    digitalWrite(byj_pin_1, LOW);
    digitalWrite(byj_pin_2, LOW);
    digitalWrite(byj_pin_3, LOW);
    break;
  case 7:
    digitalWrite(byj_pin_0, HIGH);
    digitalWrite(byj_pin_1, LOW);
    digitalWrite(byj_pin_2, LOW);
    digitalWrite(byj_pin_3, HIGH);
    break;
  default:
    digitalWrite(byj_pin_0, LOW);
    digitalWrite(byj_pin_1, LOW);
    digitalWrite(byj_pin_2, LOW);
    digitalWrite(byj_pin_3, LOW);
    break;
  }
  setDirection();
}
```

byj_gripper.h

```
#ifndef BYJ_GRIPPER_H_
#define BYJ_GRIPPER_H_

class BYJ_Gripper {
public:
```

```
  BYJ_Gripper(int pin0, int pin1, int pin2, int pin3, int steps);
  void cmdOn();
  void cmdOff();
private:
  bool direction;
  void moveSteps();
  void setDirection();
  int byj_pin_0;
  int byj_pin_1;
  int byj_pin_2;
  int byj_pin_3;
  int grip_steps;
  int step_cycle;
};
#endif
```

command.cpp

```
#include "command.h"
#include "logger.h"
#include <Arduino.h>

Command::Command() {
  //initialize Command to a zero-move value;
  new_command.valueX = NAN;
  new_command.valueY = NAN;
  new_command.valueZ = NAN;
  new_command.valueF = 0;
  new_command.valueE = NAN;
  new_command.valueS = 0;
  message = "";
  isRelativeCoord = false;
}

bool Command::handleGcode() {
  if (Serial.available()) {
    char c = Serial.read();
    if (c == '\n') {
       return false;
    }
    if (c == '\r') {
       bool b = processMessage(message);
       message = "";
       return b;
    } else {
       message += c;
    }
  }
  return false;
}

bool Command::processMessage(String msg){

  new_command.valueX = NAN;
  new_command.valueY = NAN;
  new_command.valueZ = NAN;
  new_command.valueE = NAN;
```

```
  new_command.valueF = 0;
  new_command.valueS = 0;
  msg.toUpperCase();
  msg.replace(" ", "");
  int active_index = 0;
  new_command.id = msg[active_index];
  if((new_command.id != 'G') && (new_command.id != 'M')){
    printErr();
    return false;
  }

  active_index++;
  int temp_index = active_index;
  while (temp_index<msg.length() && !isAlpha(msg[temp_index])){
    temp_index++;
  }
  new_command.num = msg.substring(active_index,
temp_index).toInt();
  active_index = temp_index;
  temp_index++;
  while (temp_index<msg.length()){
    while (!isAlpha(msg[temp_index]) || msg[temp_index]=='.'){
      temp_index++;
      if (temp_index == msg.length()){
        break;
      }
    }
    value_segment(msg.substring(active_index, temp_index));
    active_index = temp_index;
    temp_index++;
  }
  return true;
}

void Command::value_segment(String msg_segment){
  float msg_value = msg_segment.substring(1).toFloat();
  switch (msg_segment[0]){
    case 'X': new_command.valueX = msg_value; break;
    case 'Y': new_command.valueY = msg_value; break;
    case 'Z': new_command.valueZ = msg_value; break;
    case 'E': new_command.valueE = msg_value; break;
    case 'F': new_command.valueF = msg_value; break;
    case 'S': new_command.valueS = msg_value; break;
  }
}

Cmd Command::getCmd() const {
  return new_command;
}

void Command::cmdGetPosition(Point pos, Point pos_offset, float
highRad, float lowRad, float rotRad){
  if(isRelativeCoord) {
    Logger::logINFO("RELATIVE MODE");
  } else {
    Logger::logINFO("ABSOLUTE MODE");
```

```
  }
  Logger::logINFO("CURRENT POSITION: [X:"+String(pos.xmm -
pos_offset.xmm)+" Y:"+String(pos.ymm - pos_offset.ymm)+"
Z:"+String(pos.zmm - pos_offset.zmm)+" E:"+String(pos.emm -
pos_offset.emm)+"]");
  //Logger::logINFO("RADIANS: [HIGH:"+String(highRad)+"
LOW:"+String(lowRad)+" ROT:"+String(rotRad));
}

void Command::cmdToRelative(){
  isRelativeCoord = true;
  Logger::logINFO("RELATIVE MODE ON");
}

void Command::cmdToAbsolute(){
  isRelativeCoord = false;
  Logger::logINFO("ABSOLUTE MODE ON");
}

void cmdMove(Cmd(&cmd), Point pos, Point pos_offset, bool
isRelativeCoord){

  if(isRelativeCoord == true){
    cmd.valueX = isnan(cmd.valueX) ? pos.xmm : cmd.valueX +
pos.xmm;
    cmd.valueY = isnan(cmd.valueY) ? pos.ymm : cmd.valueY +
pos.ymm;
    cmd.valueZ = isnan(cmd.valueZ) ? pos.zmm : cmd.valueZ +
pos.zmm;
    cmd.valueE = isnan(cmd.valueE) ? pos.emm : cmd.valueE +
pos.emm;
  } else {
    cmd.valueX = isnan(cmd.valueX) ? pos.xmm : cmd.valueX +
pos_offset.xmm;
    cmd.valueY = isnan(cmd.valueY) ? pos.ymm : cmd.valueY +
pos_offset.ymm;
    cmd.valueZ = isnan(cmd.valueZ) ? pos.zmm : cmd.valueZ +
pos_offset.zmm;
    cmd.valueE = isnan(cmd.valueE) ? pos.emm : cmd.valueE +
pos_offset.emm;
  }
}

void cmdDwell(Cmd(&cmd)){
  delay(int(cmd.valueS * 1000));
}

void printErr() {
  Logger::logERROR("COMMAND NOT RECOGNIZED");
}
```

command.h

```
#ifndef COMMAND_H_
#define COMMAND_H_

#include <Arduino.h>
```

```
#include "interpolation.h"

struct Cmd {
  char id;
  int num;
  float valueX;
  float valueY;
  float valueZ;
  float valueF;
  float valueE;
  float valueS;
};

class Command {
  public:
    Command();
    bool handleGcode();
    bool processMessage(String msg);
    void value_segment(String msg_segment);
    Cmd getCmd() const;
    void cmdGetPosition(Point pos, Point pos_offset, float
highRad, float lowRad, float rotRad);
    void cmdToRelative();
    void cmdToAbsolute();
    bool isRelativeCoord;
    Cmd new_command;

  private:
    String message;
};

void cmdMove(Cmd(&cmd), Point pos, Point pos_offset, bool
isRelativeCoord);
void cmdDwell(Cmd(&cmd));
void printErr();

#endif
```

config.h

```
#ifndef CONFIG_H_
#define CONFIG_H_

//SERIAL SETTINGS
#define BAUD 115200

//CHOICE OF MCU BOARDS TO DRIVE ROBOT. BY DEFAULT: 0 (MEGA2560)
#define BOARD_CHOICE MEGA2560
     //CHOICES:
     //  MEGA2560
     //  UNO
     //  WEMOSD1R32

     //..DO NOT CHANGE BELOW BOARD VALUES..//
     #define MEGA2560    0 //ARDUINO MEGA2560 & RAMPS 1.4
     #define UNO         1 //ARDUINO UNO & CNC SHIELD
     #define WEMOSD1R32  2 //ESP32 - WEMOS D1 R32 & CNC SHIELD
```

```
     //PLEASE SEE & ADJUST ESP32 PARAMETERS IN [config_esp32.h]

//ROBOT ARM LENGTH
#define SHANK_LENGTH 140.0
#define LOW_SHANK_LENGTH 140.0
#define HIGH_SHANK_LENGTH 140.0

#define END_EFFECTOR_OFFSET 76.9// LENGTH FROM UPPER SHANK BEARING
TO MIDPOINT OF END EFFECTOR IN MM

//INITIAL INTERPOLATION SETTINGS
//  INITIAL_XYZ FORMS VERTICAL LOWER ARM & HORIZONTAL UPPER ARM IN
90 DEGREES
#define INITIAL_X 0.0 // CARTESIAN COORDINATE X
#define INITIAL_Y (HIGH_SHANK_LENGTH+END_EFFECTOR_OFFSET) //
CARTESIAN COORDINATE Y
#define INITIAL_Z 138.0 //LOW_SHANK_LENGTH // CARTESIAN COORDINATE
Z

#define INITIAL_E0 0.0 // RAIL STEPPER ENDSTOP POSITION

//  CALIBRATE HOME STEPS TO REACH DESIRED INITIAL_XYZ POSITIONS
#define X_HOME_STEPS 0 //1200 //765 //860 // STEPS FROM X_ENDSTOP
TO INITIAL_XYZ FOR UPPER ARM (Z)
#define Y_HOME_STEPS 0 //870 //1940 // STEPS FROM Y_ENDSTOP TO
INITIAL_XYZ FOR LOWER ARM (Y)
#define Z_HOME_STEPS 0 //3400 // STEPS FROM Z_ENDSTOP TO
INITIAL_XYZ FOR ROTATION CENTER (X)
#define E0_HOME_STEPS 0 //200 // STEPS FROM E0_ENDSTOP TO
INITIAL_E0

//HOMING SETTINGS:
#define HOME_X_STEPPER true // "true" IF ENDSTOP IS INSTALLED
#define HOME_Y_STEPPER true // "true" IF ENDSTOP IS INSTALLED
#define HOME_Z_STEPPER true // "true" IF ENDSTOP IS INSTALLED
#define HOME_E0_STEPPER true // "true" IF ENDSTOP IS INSTALLED
#define HOME_ON_BOOT false // "true" IF HOMING REQUIRED AFTER
POWER ON
#define HOME_DWELL 1200 // INCREASE TO SLOW DOWN HOMING SPEED

//STEPPER SETTINGS:
#define MICROSTEPS 16 // MICROSTEPPING CONFIGURATION ON RAMPS1.4
#define STEPS_PER_REV 200 // NEMA17 STEPS PER REVOLUTION
#define INVERSE_X_STEPPER false // CHANGE IF STEPPER MOVES OTHER
WAY
#define INVERSE_Y_STEPPER true // CHANGE IF STEPPER MOVES OTHER
WAY
#define INVERSE_Z_STEPPER false // CHANGE IF STEPPER MOVES OTHER
WAY
#define INVERSE_E0_STEPPER false // CHANGE IF STEPPER MOVES OTHER
WAY

//RAIL SETTINGS: (microsteps 8)
#define RAIL false // Set nilai ke "true" jika menggunakan rel,
set ke "false" jika robot tidak pakai rel.
#define STEPS_PER_MM_RAIL 80.0 // STEPS PER MM FOR RAIL MOTOR
```

```
        //FORMULA: STEPS_PER_REV * MICROSTEPS / MOTOR_GEAR_TEETH /
2
#define RAIL_LENGTH 355.0 // MAX LENGTH OF RAIL IN MM

//ENDSTOP SETTINGS:
#define X_MIN_INPUT 0 // OUTPUT VALUE WHEN SWITCH ACTIVATED - NO:
0, NC: 1
#define Y_MIN_INPUT 0 // OUTPUT VALUE WHEN SWITCH ACTIVATED - NO:
0, NC: 1
#define Z_MIN_INPUT 0 // OUTPUT VALUE WHEN SWITCH ACTIVATED - NO:
0, NC: 1
#define E0_MIN_INPUT 0 // OUTPUT VALUE WHEN SWITCH ACTIVATED - NO:
0, NC: 1

//GEAR RATIO SETTINGS
#define MOTOR_GEAR_TEETH 20.0 // 20.0 FOR 20SFFACTORY BELT
VERSION   9.0 FOR FTOBLER GEAR VERSION
#define MAIN_GEAR_TEETH 90.0 // 90.0 FOR 20SFFACTORY BELT
VERSION   32.0 FOR FTOBLER GEAR VERSION

//EQUIPMENT SETTINGS
#define LASER false // 12V LASER CONNECTED TO LASER_PIN
#define PUMP false // 12V AIR PUMP CONNECTED TO PUMP_PIN
#define FAN_DELAY 1000 // FAN ON IN SECONDS

//GRIPPER SETTINGS
#define GRIPPER 1 //GRIPPER MOTOR IN USE
     // 0: 28BYJ-48 MICRO STEPPER MOTOR (WORKS ON BOARD_CHOICE
MEGA2560 ONLY)
     // 1: 9G SERVO OR MG996 SERVO EQUIVALENT

     //..DO NOT CHANGE BELOW MOTOR DEFINE VALUES..//
     #define BYJ 0
     #define SERVO 1

//28BYJ GRIPPER SETTINGS
#define BYJ_GRIP_STEPS 1200 //FTOBLER: 1200
//SERVO GRIPPER SETTINGS
#define SERVO_GRIP_DEGREE 70.0 // Nilai Maksimal servo terbuka
#define SERVO_UNGRIP_DEGREE 3.0 //Nilai Maksimal servo mencapit

//COMMAND QUEUE SETTINGS
#define QUEUE_SIZE 15

//PRINT REPLY SETTING
#define PRINT_REPLY true // "true" TO PRINT MSG AFTER ONE COMMAND
IS PROCESSED
#define PRINT_REPLY_MSG "ok" // MSG SENT FOR USER'S POST
PROCESSING WITH OTHER SOFTWARE

//DEFAULT SPEED PROFILE SETTING
#define SPEED_PROFILE 2 // OPTIONS BELOW
//0: FLAT SPEED CURVE (CONSTANT SPEED PER MOVEMENT, SUITABLE FOR
REALTIME CONTROL SOFTWARE)
//1: ARCTAN APPROX (SLIGHT BELL CURVE ACCELERATION & DECELERATION)
```

```
//2: COSIN APPROX (TOTAL BELL CURVE ACCEL FROM 0 & DECEL TO 0,
SUITABLE FOR PRESET COMMAND MOVEMENTS)

//LOG SETTINGS
#define LOG_LEVEL 2
//0: ERROR
//1: INFO
//2: DEBUG

//MOVE LIMIT PARAMETERS
#define Z_MIN -140.0 //MINIMUM Z HEIGHT OF TOOLHEAD TOUCHING
GROUND
#define Z_MAX (LOW_SHANK_LENGTH+70.0) //SHANK_LENGTH ADDING
ARBITUARY NUMBER FOR Z_MAX
#define SHANKS_MIN_ANGLE_COS 0.791436948
#define SHANKS_MAX_ANGLE_COS -0.774944489
#define R_MIN (sqrt((sq(LOW_SHANK_LENGTH) + sq(HIGH_SHANK_LENGTH))
- (2*LOW_SHANK_LENGTH*HIGH_SHANK_LENGTH*SHANKS_MIN_ANGLE_COS) ))
#define R_MAX (sqrt((sq(LOW_SHANK_LENGTH) + sq(HIGH_SHANK_LENGTH))
- (2*LOW_SHANK_LENGTH*HIGH_SHANK_LENGTH*SHANKS_MAX_ANGLE_COS) ))

#endif
```

config_esp32.h

```
//ESP32 SETTINGS (FOR ESP32 USERS ONLY, OTHERWISE LEAVE UNCHANGED)
  //-------
  //TO USE WEMOSD1R32 OPTION, BOARD ESP32 V1.0.4 NEEDS TO BE
INSTALLED IN ARDUINO IDE
  //TUTORIAL: www.randomnerdtutorials.com/installing-the-esp32-
board-in-arduino-ide-windows-instructions/
  //-------
  //LIBARIES OF ESP32 TO BE ADDED
  //  1) ESP32Servo <
www.arduino.cc/reference/en/libraries/esp32servo/ >
  //  2) PS4Controller < https://github.com/aed3/PS4-esp32 >
  //  3) ESP32Wiimote https://github.com/bigw00d/Arduino-
ESP32Wiimote
  //-------
  //ESP32 SUPPORTS PS4 CONTROLLER: USE 'SIXAXISPAIRTOOL' TO FIND
MAC ADDRESS < https://sixaxispairtool.software.informer.com/ >

#define ESP32_JOYSTICK NONE
    //CHOICES:
    //  NONE              < NOT USING JOYSTICK >
    //  DUALSHOCK4        < PLAYSTATION 4 CONTROLLER >
    //  WIIMOTE           < WII REMOTE >

    //..DO NOT CHANGE BELOW VALUES..//
    #define NONE            0
    #define DUALSHOCK4      1
    #define WIIMOTE         2

//JOYSTICK CONTROL SPEED MULTIPLIER - REDUCE TO ADJUST
SENSITIVITY/SPEED
#define JOYSTICK_SPEED_MULTIPLIER 1.0
```

```
//PS4 CONTROLLER TARGET MAC ADDRESS
#define PS4_MAC "20:ff:fa:cc:00:ff"
```

controller_ps4.cpp

```
#include <Arduino.h>

#include "config_esp32.h"
#include "controller_ps4.h"
#include "logger.h"

#if BOARD_CHOICE == WEMOSD1R32
#include <PS4Controller.h>
Controller_PS4::Controller_PS4(char* aMacAddress){
  macAddress = aMacAddress;
}

void Controller_PS4::setup(){
  PS4.begin(macAddress);
  while (!PS4.isConnected()){
    if (PS4.isConnected()){
      Logger::logINFO("PS4 CONTROLLER CONNECTED");
      break;
    }
    Logger::logINFO("CONNECTING TO PS4 CONTROLLER");
    delay(1000);
  }
}

bool Controller_PS4::checkConnection(){
  return PS4.isConnected();
}

void Controller_PS4::update(){
  buttons[PS4_RIGHT] = PS4.Right();
  buttons[PS4_DOWN] = PS4.Down();
  buttons[PS4_UP] = PS4.Up();
  buttons[PS4_LEFT] = PS4.Left();
  buttons[PS4_SQUARE] = PS4.Square();
  buttons[PS4_CROSS] = PS4.Cross();
  buttons[PS4_CIRCLE] = PS4.Circle();
  buttons[PS4_TRIANGLE] = PS4.Triangle();
  buttons[PS4_L1] = PS4.L1();
  buttons[PS4_R1] = PS4.R1();
  buttons[PS4_SHARE] = PS4.Share();
  buttons[PS4_OPTIONS] = PS4.Options();
  buttons[PS4_L3] = PS4.L3();
  buttons[PS4_R3] = PS4.R3();
  buttons[PS4_PSBUTTON] = PS4.PSButton();
  buttons[PS4_TOUCHPAD] = PS4.Touchpad();
  buttons[PS4_LSTICKX] = PS4.LStickX();
  buttons[PS4_LSTICKY] = PS4.LStickY();
  buttons[PS4_RSTICKX] = PS4.RStickX();
  buttons[PS4_RSTICKY] = PS4.RStickY();
  buttons[PS4_L2VALUE] = PS4.L2Value();
  buttons[PS4_R2VALUE] = PS4.R2Value();
```

```
}
#endif
```

controller_ps4.h

```
#ifndef CONTROLLER_PS4_H_
#define CONTROLLER_PS4_H_

#include "config.h"

//PS4 BUTTON DECLARATIONS
#define PS4_RIGHT             0
#define PS4_DOWN              1
#define PS4_UP                2
#define PS4_LEFT              3
#define PS4_SQUARE            4
#define PS4_CROSS             5
#define PS4_CIRCLE            6
#define PS4_TRIANGLE          7
#define PS4_L1                8
#define PS4_R1                9
#define PS4_SHARE             10
#define PS4_OPTIONS           11
#define PS4_L3                12
#define PS4_R3                13
#define PS4_PSBUTTON          14
#define PS4_TOUCHPAD          15
#define PS4_LSTICKX           16
#define PS4_LSTICKY           17
#define PS4_RSTICKX           18
#define PS4_RSTICKY           19
#define PS4_L2VALUE           20
#define PS4_R2VALUE           21

#if BOARD_CHOICE == WEMOSD1R32

class Controller_PS4 {
  public:
    Controller_PS4(char* aMacAddress);
    int buttons[22];
    void setup();
    void update();
    bool checkConnection();

  private:
    char* macAddress;
};

#endif
#endif
```

controller_wiimote.cpp

```
#include <Arduino.h>

#include "config_esp32.h"
#include "controller_wiimote.h"
#include "logger.h"
```

```
#if BOARD_CHOICE == WEMOSD1R32
#include "ESP32Wiimote.h"

Controller_Wiimote::Controller_Wiimote(){
  ESP32Wiimote wiimote;
}

void Controller_Wiimote::setup(){
  Logger::logINFO("HOLD 1+2 BUTTON TO CONNECT WIIMOTE");
  Logger::logINFO("WHEN LED1 LIGHTS UP. PRESS ANY BUTTON TO
START");
  wiimote.init();
  wiimote.addFilter(ACTION_IGNORE, FILTER_NUNCHUK_ACCEL);
  while (wiimote.available() == 0){
    wiimote.task();
    if (wiimote.available() > 0){
      Logger::logINFO("WIIMOTE CONNECTED");
      break;
    }
  Logger::logINFO("CONNECTING TO WIIMOTE");
  delay(10);
  }
}

void Controller_Wiimote::update(){
  wiimote.task();
  if (wiimote.available() > 0){
    button = wiimote.getButtonState();
  }
  delayMicroseconds(10);
}

#endif
```

controller_wiimote.h

```
#ifndef CONTROLLER_WIIMOTE_H_
#define CONTROLLER_WIIMOTE_H_

#include "config.h"

#if BOARD_CHOICE == WEMOSD1R32

#include "ESP32Wiimote.h"

#define WII_LEFT 0x0800
#define WII_RIGHT 0x0400
#define WII_UP 0x0200
#define WII_DOWN 0x0100
#define WII_A 0x0008
#define WII_B 0x0004
#define WII_PLUS 0x1000
#define WII_HOME 0x0080
#define WII_MINUS 0x0010
#define WII_ONE 0x0002
#define WII_TWO 0x0001
```

```
class Controller_Wiimote {
  public:
    Controller_Wiimote();
    uint16_t button;
    void setup();
    void update();
  private:
    ESP32Wiimote wiimote;
};

#endif
#endif
```

endstop.cpp

```
#include "endstop.h"
#include <Arduino.h>

Endstop::Endstop(int a_min_pin, int a_dir_pin, int a_step_pin, int
a_en_pin, int a_switch_input, int a_step_offset, int a_home_dwell,
bool does_swap_pin){
  min_pin = a_min_pin;
  dir_pin = a_dir_pin;
  step_pin = a_step_pin;
  en_pin = a_en_pin;
  switch_input = a_switch_input;
  home_dwell = a_home_dwell;
  step_offset = a_step_offset;
  swap_pin = does_swap_pin;
  if (swap_pin == false){
    pinMode(min_pin, INPUT_PULLUP);
  }
}

void Endstop::home(bool dir) {
  if (swap_pin == true){
    pinMode(min_pin, INPUT_PULLUP);
    delayMicroseconds(5);
  }
  digitalWrite(en_pin, LOW);
  delayMicroseconds(5);
  if (dir==1){
    digitalWrite(dir_pin, HIGH);
  } else {
    digitalWrite(dir_pin, LOW);
  }
  delayMicroseconds(5);
  bState = !(digitalRead(min_pin) ^ switch_input);
  while (!bState) {
    digitalWrite(step_pin, HIGH);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(home_dwell);
    bState = !(digitalRead(min_pin) ^ switch_input);
  }
  homeOffset(dir);
  if (swap_pin == true){
    pinMode(min_pin, OUTPUT);
```

```
    delayMicroseconds(5);
  }
}

void Endstop::homeOffset(bool dir){
  if (dir==1){
    digitalWrite(dir_pin, LOW);
  }
  else{
    digitalWrite(dir_pin, HIGH);
  }
  delayMicroseconds(5);
  for (int i = 1; i <= step_offset; i++) {
    digitalWrite(step_pin, HIGH);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(home_dwell);
  }
}

void Endstop::oneStepToEndstop(bool dir){
  if (swap_pin == true){
    pinMode(min_pin, INPUT_PULLUP);
  }
  digitalWrite(en_pin, LOW);
  delayMicroseconds(5);
  if (dir==1){
    digitalWrite(dir_pin, HIGH);
  } else {
    digitalWrite(dir_pin, LOW);
  }
  delayMicroseconds(5);
  bState = !(digitalRead(min_pin) ^ switch_input);

  if (!bState) {
    digitalWrite(step_pin, HIGH);
    digitalWrite(step_pin, LOW);
    delayMicroseconds(home_dwell);
  }
  bState = !(digitalRead(min_pin) ^ switch_input);
}

bool Endstop::state(){
  if (swap_pin == true){
    pinMode(min_pin, INPUT_PULLUP);
    delayMicroseconds(5);
  }
  bState = !(digitalRead(min_pin) ^ switch_input);
  if (swap_pin == true){
    pinMode(min_pin, OUTPUT);
    delayMicroseconds(5);
  }
  return bState;
}
```

endstop.h

```
#ifndef ENDSTOP_H_
#define ENDSTOP_H_
```

```
class Endstop {
  public:
    Endstop(int a_min_pin, int a_dir_pin, int a_step_pin, int
a_en_pin, int a_switch_input, int a_step_offset, int a_home_dwell,
bool does_swap_pin);
    void home(bool dir);
    void homeOffset(bool dir);
    void oneStepToEndstop(bool dir);
    bool state();
    bool bState;

  private:
    int min_pin;
    int dir_pin;
    int step_pin;
    int en_pin;
    int switch_input;
    int home_dwell;
    int step_offset;
    bool swap_pin;
};

#endif
```

equipment.cpp

```
#include "equipment.h"
#include <Arduino.h>

Equipment::Equipment(int equipment_pin){
  pin = equipment_pin;
  pinMode(pin, OUTPUT);
}

void Equipment::cmdOn(){
  digitalWrite(pin, HIGH);
}

void Equipment::cmdOff(){
  digitalWrite(pin, LOW);
}
```

equipment.h

```
#ifndef EQUIPMENT_H_
#define EQUIPMENT_H_

class Equipment {
public:
  Equipment(int equipment_pin);
  void cmdOn();
  void cmdOff();
private:
  int pin;
};

#endif
```

fanControl.cpp

```
#include "fanControl.h"
#include <Arduino.h>

FanControl::FanControl(int aPin, int aFanDelay) {
  fan_delay = aFanDelay * 1000;
  nextShutdown = 0;
  pin = aPin;
  pinMode(pin , OUTPUT);
  digitalWrite(pin , LOW);
  state = false;
}

void FanControl::enable(bool value) {
  if (value) {
    state = true;
    digitalWrite(pin, HIGH);
  } else {
    disable();
  }
}

void FanControl::disable() {
  state = false;
  nextShutdown = millis() + fan_delay;
  update();
}

void FanControl::update() {
  if (!state) {
     if (millis() >= nextShutdown) {
       digitalWrite(pin, LOW);
     }
  }
}
```

fanControl.h

```
#ifndef FANCONTROL_H_
#define FANCONTROL_H_

class FanControl {
public:
  FanControl(int aPin, int aFanDelay);
  void enable(bool value = true);
  void disable();
  void update();
private:
  bool state;
  int pin;
  long fan_delay;
  long nextShutdown;
};

#endif
```

interpolation.cpp

```
#include "interpolation.h"
```

```
#include "config.h"
#include "queue.h"
#include "logger.h"

Interpolation::Interpolation(){
  speed_profile = SPEED_PROFILE;
  pos_offset.xmm = 0.0;
  pos_offset.ymm = 0.0;
  pos_offset.zmm = 0.0;
  pos_offset.emm = 0.0;
}

void Interpolation::setSpeedProfile(int new_speed_profile){
  speed_profile = new_speed_profile;
}

//G92 POSITION OFFSET FUNCTIONS
void Interpolation::setPosOffset(float new_x, float new_y, float
new_z, float new_e) {
  pos_offset.xmm = xPosmm - new_x;
  pos_offset.ymm = yPosmm - new_y;
  pos_offset.zmm = zPosmm - new_z;
  pos_offset.emm = ePosmm - new_e;
  Logger::logINFO("POSITION OFFSET: [X" + String(pos_offset.xmm) +
" Y:" + String(pos_offset.ymm) + " Z:" + String(pos_offset.zmm) +
" E:" + String(pos_offset.emm) + "]");
  Logger::logINFO("CURRENT POSITION: [X:"+String(new_x)+"
Y:"+String(new_y)+" Z:"+String(new_z)+" E:"+String(new_e)+"]");
}

void Interpolation::resetPosOffset(){
  pos_offset.xmm = 0.0;
  pos_offset.ymm = 0.0;
  pos_offset.zmm = 0.0;
  pos_offset.emm = 0.0;
}

Point Interpolation::getPosOffset() const {
  return pos_offset;
}

void Interpolation::setCurrentPos(float px, float py, float pz,
float pe) {
  Point p;
  p.xmm = px;
  p.ymm = py;
  p.zmm = pz;
  p.emm = pe;
  setCurrentPos(p);
}

void Interpolation::setInterpolation(float px, float py, float pz,
float pe, float v) {
  Point p;
  p.xmm = px;
  p.ymm = py;
```

```
  p.zmm = pz;
  p.emm = pe;
  setInterpolation(p, v);
}

void Interpolation::setInterpolation(float p1x, float p1y, float
p1z, float p1e, float p2x, float p2y, float p2z, float p2e, float
v) {
  Point p1;
  Point p2;
  p1.xmm = p1x;
  p1.ymm = p1y;
  p1.zmm = p1z;
  p1.emm = p1e;
  p2.xmm = p2x;
  p2.ymm = p2y;
  p2.zmm = p2z;
  p2.emm = p2e;
  setInterpolation(p1, p2, v);
}

void Interpolation::setInterpolation(Point p1, float v) {
  Point p0;
  p0.xmm = xStartmm + xDelta;
  p0.ymm = yStartmm + yDelta;
  p0.zmm = zStartmm + zDelta;
  p0.emm = eStartmm + eDelta;
  setInterpolation(p0, p1, v);
}

void Interpolation::setInterpolation(Point p0, Point p1, float av)
{
  v = av; //mm/s

  float a = (p1.xmm - p0.xmm);
  float b = (p1.ymm - p0.ymm);
  float c = (p1.zmm - p0.zmm);
  float e = abs(p1.emm - p0.emm);
  float dist = sqrt(a*a + b*b + c*c);

  if (dist < e) {
    dist = e;
  }

  if (v < 5) { //includes 0 = default value
    v = sqrt(dist) * 10; //set a good value for v
  }
  if (v < 5) {
     v = 5;
  }

  tmul = v / dist;

  xStartmm = p0.xmm;
  yStartmm = p0.ymm;
  zStartmm = p0.zmm;
```

```
  eStartmm = p0.emm;

  xDelta = (p1.xmm - p0.xmm);
  yDelta = (p1.ymm - p0.ymm);
  zDelta = (p1.zmm - p0.zmm);
  eDelta = (p1.emm - p0.emm);

  state = 0;

  startTime = micros();
}

void Interpolation::setCurrentPos(Point p) {
  xStartmm = p.xmm;
  yStartmm = p.ymm;
  zStartmm = p.zmm;
  eStartmm = p.emm;
  xDelta = 0;
  yDelta = 0;
  zDelta = 0;
  eDelta = 0;
}

void Interpolation::updateActualPosition() {
  if (state != 0) {
    return;
  }
  long microsek = micros();
  float t = (microsek - startTime) / 1000000.0;
    //float t = (microsek - startTime) / 900000.0;
  float progress;
  switch (speed_profile){
    // FLAT SPEED CURVE
    case 0:
      progress = t * tmul;
      if (progress >= 1.0){
        progress = 1.0;
        state = 1;
      }
      break;
    // ARCTAN APPROX
    case 1:
      progress = atan((PI * t * tmul) - (PI * 0.5)) * 0.5 + 0.5;
      if (progress >= 1.0) {
        progress = 1.0;
        state = 1;
      }
      break;
    // COSIN APPROX
    case 2:
      progress = -cos(t * tmul * PI) * 0.5 + 0.5;
      if ((t * tmul) >= 1.0) {
        progress = 1.0;
        state = 1;
      }
      break;
```

```
  }
  pos_tracker[X_AXIS] = xStartmm + progress * xDelta;
  pos_tracker[Y_AXIS] = yStartmm + progress * yDelta;
  pos_tracker[Z_AXIS] = zStartmm + progress * zDelta;
  pos_tracker[E_AXIS] = eStartmm + progress * eDelta;

  if(isAllowedPosition(pos_tracker)){
    xPosmm = pos_tracker[X_AXIS];
    yPosmm = pos_tracker[Y_AXIS];
    zPosmm = pos_tracker[Z_AXIS];
    ePosmm = pos_tracker[E_AXIS];
  } else {
    pos_tracker[X_AXIS] = xPosmm;
    pos_tracker[Y_AXIS] = yPosmm;
    pos_tracker[Z_AXIS] = zPosmm;
    pos_tracker[E_AXIS] = ePosmm;
    state = 1;
    progress = 1.0;
    xStartmm = xPosmm;
    yStartmm = yPosmm;
    zStartmm = zPosmm;
    eStartmm = ePosmm;
    xDelta = 0;
    yDelta = 0;
    zDelta = 0;
    eDelta = 0;
  }
  //FOR DECIPHERING SPEED CURVE
  //Serial.print("xPosmm:");
  //Serial.print(xPosmm);
  //Serial.print(" yPosmm:");
  //Serial.print(yPosmm);
  //Serial.print(" zPosmm:");
  //Serial.println(zPosmm);
}

bool Interpolation::isFinished() const {
  return state != 0;
}

float Interpolation::getXPosmm() const {
  return xPosmm;
}

float Interpolation::getYPosmm() const {
  return yPosmm;
}

float Interpolation::getZPosmm() const {
  return zPosmm;
}

float Interpolation::getEPosmm() const {
  return ePosmm;
}
```

```
Point Interpolation::getPosmm() const {
  Point p;
  p.xmm = xPosmm;
  p.ymm = yPosmm;
  p.zmm = zPosmm;
  p.emm = ePosmm;
  return p;
}

bool Interpolation::isAllowedPosition(float pos_tracker[4]) {
  float rrot_ee = hypot(pos_tracker[X_AXIS], pos_tracker[Y_AXIS]);
  float rrot = rrot_ee - END_EFFECTOR_OFFSET;
  float rrot_x = rrot * (pos_tracker[Y_AXIS] / rrot_ee);
  float rrot_y = rrot * (pos_tracker[X_AXIS] / rrot_ee);
  float squaredPositionModule = sq(rrot_x) + sq(rrot_y) +
sq(pos_tracker[Z_AXIS]);

      bool retVal = (
          squaredPositionModule <= sq(R_MAX)
          && squaredPositionModule >= sq(R_MIN)
          && pos_tracker[Z_AXIS] >= Z_MIN
          && pos_tracker[Z_AXIS] <= Z_MAX
          #if RAIL
          && pos_tracker[E_AXIS] <= RAIL_LENGTH
          && pos_tracker[E_AXIS] >= 0
          #endif
      );
  if(!retVal) {
    Logger::logERROR("LIMIT REACHED: [X:" +
String(pos_tracker[X_AXIS]) + " Y:" + String(pos_tracker[Y_AXIS])
+ " Z:" + String(pos_tracker[Z_AXIS]) + " E:" +
String(pos_tracker[E_AXIS]) + "]");
  }
  return retVal;
}
```

interpolation.h

```
#ifndef INTERPOLATION_H_
#define INTERPOLATION_H_
#include <Arduino.h>

#define X_AXIS 0
#define Y_AXIS 1
#define Z_AXIS 2
#define E_AXIS 3

struct Point {
  float xmm;
  float ymm;
  float zmm;
  float emm;
};
class Interpolation {
public:
  //void resetInterpolation(float px, float py, float pz);
```

```
  //void resetInterpolation(float p1x, float p1y, float p1z, float
p2x, float p2y, float p2z);
  //void resetInterpolation(Point p0, Point p1);
  Interpolation();
  void setCurrentPos(float px, float py, float pz, float pe);
  void setInterpolation(float px, float py, float pz, float pe,
float v = 0);
  void setInterpolation(float p1x, float p1y, float p1z, float
p1e, float p2x, float p2y, float p2z, float p2e, float av = 0);

  void setCurrentPos(Point p);
  void setInterpolation(Point p1, float v = 0);
  void setInterpolation(Point p0, Point p1, float v = 0);

  void updateActualPosition();
  bool isFinished() const;

  float getXPosmm() const;
  float getYPosmm() const;
  float getZPosmm() const;
  float getEPosmm() const;
  Point getPosmm() const;
  bool isAllowedPosition(float pos_tracker[4]);
  void setPosOffset(float new_x, float new_y, float new_z, float
new_e);
  void resetPosOffset();
  Point getPosOffset() const;
  int speed_profile;
  void setSpeedProfile(int new_speed_profile);

private:
  Point pos_offset;
  float pos_tracker[4];
  byte state;

  long startTime;

  float xStartmm;
  float yStartmm;
  float zStartmm;
  float eStartmm;
  float xDelta;
  float yDelta;
  float zDelta;
  float eDelta;
  float xPosmm;
  float yPosmm;
  float zPosmm;
  float ePosmm;
  float v;
  float tmul;
};

#endif
```

logger.cpp

```
#include "config.h"
#include "logger.h"

void Logger::log(String message, int level) {
  if(LOG_LEVEL >= level) {
    String logMsg;
    switch(level) {
      case LOG_ERROR:
        logMsg = "ERROR: ";
      break;
      case LOG_INFO:
        logMsg = "INFO: ";
      break;
      case LOG_DEBUG:
        logMsg = "DEBUG: ";
      break;
    }
    logMsg = logMsg + message;
    Serial.println(logMsg);
  }
}

void Logger::logERROR(String message) {
  log(message, LOG_ERROR);
}
void Logger::logINFO(String message) {
  log(message, LOG_INFO);
}
void Logger::logDEBUG(String message) {
  log(message, LOG_DEBUG);
}
```

logger.h

```
#ifndef LOGGER_H_
#define LOGGER_H_

#include <Arduino.h>

#define LOG_ERROR 0
#define LOG_INFO 1
#define LOG_DEBUG 2

class Logger {
  public:
    static void log(String message, int level);
    static void logINFO(String message);
    static void logERROR(String message);
    static void logDEBUG(String message);
};
#endif
```

queue.h

```
#ifndef QUEUE_H_
#define QUEUE_H_
```

```
template <typename Element> class Queue {
public:
  Queue(int alen);
  ~Queue();
  bool push(Element elem);
  Element pop();
  bool isFull() const;
  bool isEmpty() const;
  int getFreeSpace() const;
  int getMaxLength() const;
  inline int getUsedSpace() const;
private:
  Queue(Queue<Element>& q);  //copy const.
  Element* data;
  int len;
  int start;
  int count;
};

template <typename Element>
Queue<Element>::Queue(int alen) {
  data = new Element[alen];
  len = alen;
  start = 0;
  count = 0;
}

template <typename Element>
Queue<Element>::~Queue() {
  delete data;
}

template <typename Element>
Queue<Element>::Queue(Queue<Element>& q) {
  //nothing ever is allowed to do something here
}

template <typename Element>
bool Queue<Element>::push(Element elem) {
  data[(start + count++) % len] = elem;
}

template <typename Element>
Element Queue<Element>::pop() {
  count--;
  int s = start;
  start = (start + 1) % len;
  return data[(s) % len];
}

template <typename Element>
bool Queue<Element>::isFull() const {
  return count >= len;
}
```

```
template <typename Element>
bool Queue<Element>::isEmpty() const {
  return count <= 0;
}

template <typename Element>
int Queue<Element>::getFreeSpace() const {
  return len - count;
}

template <typename Element>
int Queue<Element>::getMaxLength() const {
  return len;
}

template <typename Element>
int Queue<Element>::getUsedSpace() const {
  return count;
}

#endif
```

robotGeometry.cpp

```
#include "robotGeometry.h"

#include <math.h>
#include <Arduino.h>

RobotGeometry::RobotGeometry(float a_ee_offset, float
a_low_shank_length, float a_high_shank_length) {
  ee_offset = a_ee_offset;
  low_shank_length = a_low_shank_length;
  high_shank_length = a_high_shank_length;
}

void RobotGeometry::set(float axmm, float aymm, float azmm) {
  xmm = axmm;
  ymm = aymm;
  zmm = azmm;
  calculateGrad();
}

float RobotGeometry::getXmm() const {
  return xmm;
}

float RobotGeometry::getYmm() const {
  return ymm;
}

float RobotGeometry::getZmm() const {
  return zmm;
}

float RobotGeometry::getRotRad() const {
  return rot;
```

```
}

float RobotGeometry::getLowRad() const {
  return low;
}

float RobotGeometry::getHighRad() const {
  return high;
}

float RobotGeometry::getHypot() const {
  return rrot_ee;
}

void RobotGeometry::calculateGrad() {
   rrot_ee =  hypot(xmm, ymm);
   float rrot = rrot_ee - ee_offset; //radius from Top View
   float rside = hypot(rrot, zmm);  //radius from Side View. Use
rrot instead of ymm..for everything
   float rside_2 = sq(rside);
   float low_2 = sq(low_shank_length);
   float high_2 = sq(high_shank_length);

   rot = asin(xmm / rrot_ee);
   high = PI - acos((low_2 + high_2 - rside_2) / (2 *
low_shank_length * high_shank_length));

   //Angle of Lower Stepper Motor  (asin()=Angle To Gripper)
   if (zmm > 0) {
     low =  acos(zmm / rside) - acos((low_2 - high_2 + rside_2) /
(2 * low_shank_length * rside));
   } else {
     low = PI - asin(rrot / rside) - acos((low_2 - high_2 +
rside_2) / (2 * low_shank_length * rside));
   }
   high = high + low;
}
```

robotGeometry.h

```
#ifndef ROBOTGEOMETRY_H_
#define ROBOTGEOMETRY_H_

class RobotGeometry {
public:
  RobotGeometry(float a_ee_offset, float a_low_shank_length, float
a_high_length);
  void set(float axmm, float aymm, float azmm);
  float getXmm() const;
  float getYmm() const;
  float getZmm() const;
  float getRotRad() const;
  float getLowRad() const;
  float getHighRad() const;
  float getHypot() const;
private:
  void calculateGrad();
```

```
  float ee_offset;
  float low_shank_length;
  float high_shank_length;
  float xmm;
  float ymm;
  float zmm;
  float rot;
  float low;
  float high;
  float rrot_ee;
};

#endif
```

servo_gripper.cpp

```
#include "servo_gripper.h"
#include "config.h"

#include <Arduino.h>

#if BOARD_CHOICE == WEMOSD1R32
  #include <ESP32Servo.h>
#else
  #include <Servo.h>
#endif

Servo_Gripper::Servo_Gripper(int pin, float grip_degree, float
ungrip_degree){
  servo_pin = pin;
  servo_grip_deg = grip_degree;
  servo_ungrip_deg = ungrip_degree;
  Servo servo_motor;
}

void Servo_Gripper::cmdOn(){
  servo_motor.attach(servo_pin);
  delayMicroseconds(10);
  servo_motor.write(servo_grip_deg);
  delay(300);
  //servo_motor.detach();
}

void Servo_Gripper::cmdOff(){
  //servo_motor.attach(servo_pin);
  servo_motor.write(servo_ungrip_deg);
  delay(300);
  servo_motor.detach();
  delayMicroseconds(50);
}

float Servo_Gripper::readDegree(){
  float servo_current_degree = servo_motor.read();
  return servo_current_degree;
}

bool Servo_Gripper::isOn(){
```

```
  float servo_current_degree = servo_motor.read();
  if (servo_current_degree == servo_grip_deg){
    return true;
  } else {
    return false;
  }
}
```

servo_gripper.h

```
#ifndef SERVO_GRIPPER_H_
#define SERVO_GRIPPER_H_

#include "config.h"

#if BOARD_CHOICE == WEMOSD1R32
  #include <ESP32Servo.h>
#else
  #include <Servo.h>
#endif

class Servo_Gripper{
public:
  Servo_Gripper(int pin, float grip_degree, float ungrip_degree);
  void cmdOn();
  void cmdOff();
  float readDegree();
  bool isOn();
private:
  Servo servo_motor;
  int servo_pin;
  float servo_grip_deg;
  float servo_ungrip_deg;
};

#endif
```

pinout.h

```
#ifndef PINOUT_H_
#define PINOUT_H_

/*
 * pinout of RAMPS 1.4
 *
 * source: http://reprap.org/wiki/RAMPS_1.4
 */

//RAMPS 1.4 PINS
#define Z_STEP_PIN         54
#define Z_DIR_PIN          55
#define Z_ENABLE_PIN       38
#define Z_MIN_PIN           18
//#define X_MAX_PIN           2

#define Y_STEP_PIN         60
#define Y_DIR_PIN          61
#define Y_ENABLE_PIN       56
#define Y_MIN_PIN          14
```

```
//#define Y_MAX_PIN          15

#define X_STEP_PIN         46
#define X_DIR_PIN          48
#define X_ENABLE_PIN       62
#define X_MIN_PIN          3
//#define Z_MAX_PIN          19

#define E0_STEP_PIN        26
#define E0_DIR_PIN         28
#define E0_ENABLE_PIN      24
//#define E0_MIN_PIN         20

#define E1_STEP_PIN        36
#define E1_DIR_PIN         34
#define E1_ENABLE_PIN      30
#define E1_MIN_PIN         41
//#define E1_MAX_PIN         43

#define BYJ_PIN_0          40
#define BYJ_PIN_1          63
#define BYJ_PIN_2          59
#define BYJ_PIN_3          64

#define SERVO_PIN           4

#define LG1_PIN            8
#define LG2_PIN            10
#define LG3_PIN            9

#define LED_PIN            13

#define SDPOWER            -1
#define SDSS               53

#define FAN_PIN            22

#define PS_ON_PIN          12
#define KILL_PIN           -1

#define IO1_PIN            16
#define IO2_PIN            17
#define IO3_PIN            57
#define IO4_PIN            25
#define IO5_PIN            27
#define IO6_PIN            29
#define IO7_PIN            31
#define IO8_PIN            33
#define IO9_PIN            35
#define IO10_PIN           37

//#define HEATER_0_PIN       10
//#define HEATER_1_PIN        8
#define TEMP_0_PIN         13   // ANALOG NUMBERING
#define TEMP_1_PIN         14   // ANALOG NUMBERING
```

```
//RAMPS AUX-2

#endif
```

BIOGRAFI PENULIS 1

Nama Lengkap	:	Pilipus Kevin Putra Anggono
Tempat/Tanggal Lahir	:	Klaten/03 Mei 2001
Jenis Kelamin	:	Laki-laki
Alamat	:	Belangwetan Klaten Utara Klaten Rt 01 Rw 02
No Telepon./HP	:	08562713361
Email	:	pilipuskevin@gmail,com
Motivasi/Harapan	:	Jangan takut mencoba
Riwayat Pendidikan	:	

1. TK Indriyasana 1 - 2005 - 2007
2. SD Pangudi Luhur Sugiyopranoto - 2007 - 2013
3. SMP N 4 Klaten - 2013 - 2016
4. SMK Leonardo Klaten - 2016 - 2019
5. Universitas Sanata Dharma - 2021 - 2025

BIOGRAFI PENULIS 2

Nama Lengkap	:	Franchino Abimanyu Annesia
Tempat/Tanggal Lahir	:	Pematang Siantar, 17 November 2002
Jenis Kelamin	:	Laki-laki
Alamat	:	Pematang Siantar, Bopongan Tamanan bangun tapan
No Telepon./HP	:	082246490232
Email	:	franchinod305@gmail.com
Motivasi/Harapan	:	keinginan untuk membahagiakan orang tua, meningkatkan kualitas diri, mendapatkan pekerjaan yang lebih baik, atau bahkan membuka usaha sendiri di masa depan
Riwayat Pendidikan	:	

1. TK santa lusia pematang siantar - 2007 - 2008
2. SD Rk Cinta Rakyat 2 pematang siantar - 2008 - 2014
3. SMP Swasta Bintang Timur pematang siantar – 2014 - 2017
4. SMA Negeri 5 pematang siantar – 2017 - 2020
5. Universitas Sanata Dharma - 2021 - 2025