

**PERANCANGAN ROBOT LENGAN PEMINDAH BENDA
BERDASARKAN UKURAN DENGAN PENGOLAHAN CITRA**

SKRIPSI

**Disusun Sebagai Salah Satu Persyaratan Memperoleh Gelar
Sarjana Teknik pada Program Studi S1 Teknik Elektro**



ROBY SYAHBANI

G1D016015

**PROGRAM STUDI TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS BENGKULU
2022**

LEMBAR PENGESAHAN SKRIPSI

PERANCANGAN ROBOT LENGAN PEMINDAH BENDA BERDASARKAN UKURAN DENGAN PENGOLAHAN CITRA

Oleh:

ROBY SYAHBANI

NPM GID016015

Skripsi ini telah diuji dan dipertahankan di hadapan Tim Pengaji Skripsi
pada tanggal **26 JANUARI 2022**

1. Junas Hadi, S.T., M.T.

NIP. 198306022014041002

Pembimbing

Utama

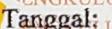


2. Hendy Santosa, S.T., M.T., Ph.D.

NIP. 198112102008121002

Pembimbing

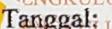
Pendamping



3. Indra Agustian, S.T., M.Eng.

NIP. 197707312002121005

Ketua Pengaji



4. Adhadi Kurniawan, S.T., M.Eng.

NIP. 198811272019031007

Anggota Pengaji

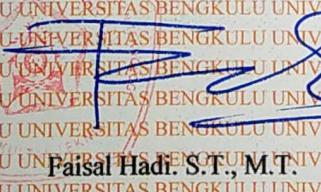


Mengesahkan,

Dekan Fakultas Teknik

Faisal Hadi, S.T., M.T.

NIP. 197707132002121005



PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan di bawah ini menyatakan dengan sesungguhnya bahwa skripsi dengan judul:

PERANCANGAN ROBOT LENGAN PEMINDAH BENDA BERDASARKAN UKURAN DENGAN PENGOLAHAN CITRA

sepenuhnya merupakan karya saya sendiri dan sepanjang pengetahuan saya bukan merupakan hasil duplikasidari skripsi dan/atau karya ilmiah lainnya yang pernah dipublikasikan dan/atau pernah dipergunakan untuk mendapatkan gelar kesarjanaan di Perguruan Tinggi atau instansi manapun, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan, kecuali bagian yang sumber informasinya dicantumkan sebagaimana mestinya.

Apabila ternyata di dalam skripsi ini terbukti terdapat unsur-unsur plagiasi, saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Bengkulu, 7 - februari - 2022

2022



Roby Syahbani

G1D016015

KATA PENGANTAR

Alhamdulillahirabbila'lamiiin,kata yang sudah sepatutnya penulis panjatkan kepada Allah SWT karena atas karunia, rahmat, serta hidayah-Nya yang begitu besar maka penulis dapat menyelesaikan skripsi yang berjudul “perancangan robot lengan pemindah benda berdasarkan ukuran dengan pengolahan citra”.

Skripsi ini dibuat untuk memenuhi syarat lulusnya mata kuliah skripsi (TEE-403) yang merupakan salah satu mata kuliah wajib dalam kurikulum Program Studi Teknik Elektro Universitas Bengkulu dan merupakan salah satu persyaratan untuk menyelesaikan pendidikan sarjana di Program Studi Teknik Elektro Universitas Bengkulu.

Dalam penyusunan skripsi ini, penulis menyadari bahwa masih banyak terdapat kekurangan dan ketidak sempurnaan, namun semua ini tidak menjadi suatu kendala bagi penulis untuk terus mencoba. Dengan terus mencoba maka penulis dapat mengetahui kekurangan dan kelebihan yang ada. Penulis sangat berterima kasih apabila ada saran ataupun kritik yang akan berguna bagi penulis untuk melakukan perbaikan-perbaikan demi kesempurnaan skripsi ini.

Penulis mengucapkan banyak terima kasih kepada pihak-pihak yang telah banyak membantu dan mengarahkan dalam penyusunan skripsi ini. Pada kesempatan ini penulis mengucapkan terima kasih kepada :

1. Ibu Dr. Retno Agustina Ekaputri, S.E., M.Sc. selaku Rektor Universitas Bengkulu.
2. Bapak Faisal Hadi, S.T., M.T. selaku Dekan Fakultas Teknik Universitas Bengkulu.
3. Ibu Ika Novia Anggraini, S.T., M.Eng. selaku Ketua Program Studi Teknik Elektro Universitas Bengkulu.
4. Bapak Junas Haidi, S.T., M.T. sebagai Dosen Pembimbing Utama yang senantiasa meluangkan waktu dalam bimbingan dan selalu memberikan masukan yang positif untuk kemajuan penelitian ini.
5. Bapak Hendy Santosa S.T., M.T., Ph.D sebagai Dosen Pembimbing Pendamping yang selalu memberikan saran-saran positif terhadap kemajuan penelitian ini.

6. Bapak Indra Agustian, S.T., M.Eng. dan bapak Adhadi Kurniawan, S.T., M.Eng. selaku dosen penguji yang telah memberikan masukan untuk lebih menyempurnakan penelitian yang telah dilakukan.
7. Bapak Irnanda Priyadi S.T., M.T. selaku dosen pembimbing akademik yang telah membantu dalam proses perkuliahan selama ini.
8. Teman-teman yang telah banyak membantu dalam mengerjakan penelitian.

ABSTRAK

Sensor kamera dapat ditambahkan dalam sebuah robot agar dapat bekerja secara otomatis, citra dari kamera harus diberikan pengolahan yang tepat agar diperoleh informasi yang dibutuhkan. Pendekripsi pinggir Canny digunakan untuk memperoleh properti dari kontur seperti titik tengah, sudut perputaran dan ukuran, pengolahan tambahan juga digunakan seperti “*Gaussian Blur*” dan “*HSV color space*” untuk menghilangkan bayangan dan *noise* pada citra. Penelitian menggunakan robot SCARA dengan tiga derajat kebebasan lalu diterapkan metode numerik dan analisis untuk menggerakkan robot, kedua metode akan dibandingkan dari sisi perilaku gerak, lama perhitungan serta keberhasilan perhitungan. Aplikasi pengolahan citra dibuat dengan bahasa pemrograman Python mampu mengukur ukuran, koordinat titik tengah serta sudut perputaran dari objek. Metode numerik untuk menggerakkan robot memiliki persentase keberhasilan sebesar 90 % serta *end of effector* robot bergerak mengikuti garis lurus dari titik awal hingga tujuan. Metode analisis menghasilkan persentase keberhasilan sebesar 100% dengan pola gerak yang acak tergantung dari kecepatan masing-masing *Joint* pada robot.

Kata Kunci : canny, numerik, analisis, *selective compliance assembly robot arm*.

DAFTAR ISI

Halaman Judul.....	i
Lembar Pengesahan Skripsi	iii
Pernyataan Keaslian Skripsi.....	v
Kata Pengantar	vii
Abstrak	ix
Daftar Isi.....	xi
Daftar Gambar.....	xiii
Daftar Tabel	xv
Bab I Pendahuluan	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Tujuan Penelitian	3
1.4 Batasan Masalah.....	3
Bab II Tinjauan Pustaka	5
2.1 Rangka Manipulator Robot Lengan.....	5
2.2 Kinematika Pada Manipulator Robot Lengan.....	7
1. Matriks perputaran (<i>Rotational Matrice</i>).	7
2. Vektor perpindahan (<i>Displacement Vector</i>).....	11
3. <i>Homogeneous transformation matrices</i>	15
4. kinematika maju (<i>foward kinematics</i>).	16
5. kinematika mundur (<i>inverse kinematics</i>).....	19
6. <i>Closed-form inverse kinematics</i>	20
7. <i>Newton Raphson root finding inverse kinematic</i>	22
2.3 Pengolahan Citra	22
1. Citra keabu-abuan.....	23
2. <i>Thresholding</i>	24
3. Konvolusi citra digital	25
4. Deteksi pinggir.	27
2.4 Perangkat Keras dan Lunak yang Digunakan Pada Penelitian.	29
5. Motor <i>stepper</i>	29
6. Mikrokontroler Arduino Mega 2560.....	31
7. Catu daya (<i>Power Supply</i>).	32
8. <i>Stepper driver A4988</i>	33
9. Bahasa pemrograman python.	34
10. OpenCV.....	35
Bab III Metode Penelitian	37

3.1	Metode Penelitian	37
3.2	Waktu dan Tempat.....	37
3.3	Alat dan Bahan	37
3.4	Diagram Blok Sistem	38
3.5	Perancangan Sistem	38
3.5.1	Perancangan perangkat keras.....	38
3.5.2	Perancangan perangkat lunak	51
3.6	Metode Pengujian	57
3.6.1	Pengujian pengolahan citra.....	57
3.6.2	Pengujian robot lengan	58
3.6.3	Pengujian keseluruhan sistem.....	59
	Bab IV Hasil Dan Pembahasan	61
4.1	Hasil perancangan Sistem.....	61
4.2	Hasil pengujian sistem.....	62
4.2.1	Pengujian performa aplikasi dalam mengolah citra masukan.	62
4.2.2	Pengujian performa gerak robot dan kinematika mundur.	69
4.2.3	Pengujian memindahkan benda berdasarkan ukuran	77
	Bab V Kesimpulan Dan Saran.....	83
5.1	Kesimpulan	83
5.2	Saran	83
	Daftar Pustaka	85

DAFTAR GAMBAR

Gambar 2.1 Ilustrasi Manipulator Robot Lengan dengan 2 <i>Degree Of Freedom</i> (www.researchgate.net).....	5
Gambar 2.2 Visualisasi <i>Frame</i> Pada Manipulator Robot Lengan (mrebelo.com)...	6
Gambar 2.3 Manipulator 2 DOF Jenis <i>Revolute</i>	8
Gambar 2.4 Manipulator dengan <i>Joint</i> 1 Berbeda Orientasi.....	9
Gambar 2.5 Manipulator Dengan 4 DOF.....	11
Gambar 2.6 Manipulator Dengan Variasi Panjang <i>Link</i>	12
Gambar 2.7 Manipulator Dilihat Dari Sisi Depan.	13
Gambar 2.8 Manipulator Ditinjau Dari Sisi Atas.....	14
Gambar 2.9 Manipulator Tampak Atas Saat Base Diberi Perputaran Sebesar θ ...	14
Gambar 2.10 Komposisi <i>Homogeneous Transformation Matrice</i>	16
Gambar 2.11 Manipulator Dengan Dua DOF Tipe Geser dan Putar.	17
Gambar 2.12 Manipulator tipe RR.....	20
Gambar 2.13 Manipulator Tipe RR Dengan Ilustrasi Segitiga Pendekatan.....	21
Gambar 2.14 Matriks Citra RGB (www.researchgate.net).	23
Gambar 2.15 Matriks Citra Keabu-abuan (csenotes.github.io).....	24
Gambar 2.16 Proses <i>Thresholding</i> Suatu Citra (sites.wustl.edu).	25
Gambar 2.17 Proses Konvolusi Suatu Citra dengan Sebuah Kernel[20]....	26
Gambar 2.18 Citra Hasil Deteksi Pinggir Sobel.	28
Gambar 2.19 Citra Hasil Deteksi Pinggir Canny	29
Gambar 2.20 Motor <i>Stepper</i> Nema 17 (www.google.com).	30
Gambar 2.21 Pemberian Sinyal Pada Motor <i>Stepper</i> (www.google.com).	30
Gambar 2.22 Arduino Mega 2560 (www.google.com).	31
Gambar 2.23 (a) Modul A4988 (electronicwing.com), (b) Pin Keluaran Modul A4988 (Pinterest.com).	33
Gambar 2.24 Logo Bahasa Pemrograman Python.	35
Gambar 3.1 Diagram Blok.....	38
Gambar 3.2 Rancangan Robot Lengan	39
Gambar 3.3 a. Bagian <i>Slider</i> , b. <i>Base Frame</i>	40
Gambar 3.4 Bagian Joint Pertama.....	41
Gambar 3.5 Bagian Lengan Pertama	42
Gambar 3.6 Bagian Lengan Kedua dan <i>End Of Effector</i>	43
Gambar 3.7 Manipulator Lengan Robot	44
Gambar 3.8 Peletakan Kamera dan Robot Lengan.	50
Gambar 3.9 Perbedaan Jarak dan Orientasi Koordinat nol pada Kamera dan Robot.	50
Gambar 3.10 Tampilan Program Arduino	51
Gambar 3.11 Tampilan GUI <i>Client App</i>	52
Gambar 3.12 Tampilan GUI <i>Clien App , Manual Control Menu</i>	52
Gambar 3.13 Diagram Alir Pengolahan Citra.....	54
Gambar 3.14 Diagram Alir Kinematika Mundur.....	55
Gambar 3.15 Diagram Alir Keseluruhan.	56
Gambar 3.16 Rangka Kerja Sistem Keseluruhan.....	57

Gambar 4.1 Hasil Pembuatan Robot Lengan	61
Gambar 4.2 Area Kerja Robot dan Peletakan Kamera.....	62
Gambar 4.3 Masukan Pengujian Canny 1	63
Gambar 4.4 Keluaran Pengujian Canny 1	63
Gambar 4.5 Masukan Pengujian Canny 2	63
Gambar 4.6 Keluaran Pengujian Canny 2	63
Gambar 4.7 Masukan Pengujian Canny 3	63
Gambar 4.8 Keluaran Pengujian Canny 3	63
Gambar 4.9 Masukan Pengujian 1.....	65
Gambar 4.10 Keluaran Dengan Semua Kontur Pengujian 1	65
Gambar 4.11 Keluaran dengan Kontur Hasil Seleksi Pengujian 1	65
Gambar 4.12 Masukan Pengujian 2.....	65
Gambar 4.13 Keluaran dengan Semua Kontur Pengujian 2.....	65
Gambar 4.14 Keluaran dengan Kontur Hasil Seleksi Pengujian 2	65
Gambar 4.15 Luas Permukaan Objek Yang Terukur Oleh Pengolahan Citra.....	66
Gambar 4.16 a. Pengukuran pada Sumbu y, b. Pengukuran pada Sumbu x, c. Hasil Pengukuran dari Pengolahan Citra	67
Gambar 4.17 Tampilan Menu Aplikasi untuk Memasukkan Nilai <i>Joint</i> Secara Manual.....	69
Gambar 4.18 a. Pengukuran <i>Joint</i> 1, b. Pengukuran <i>Joint</i> 2, c. Pengukuran <i>Joint</i> 3, d. Pengukuran <i>Joint</i> 4	70
Gambar 4.19 Jalur Gerak <i>End of Effector</i> Robot Lengan.	72
Gambar 4.20 Pengujian pemindahan benda (Tampilan Aplikasi).....	80
Gambar 4.21 Langkah robot memindahkan benda.....	80

DAFTAR TABEL

Tabel 2.1 Penjelasan Elemen Matriks Orientasi	9
Tabel 2.2 Spesifikasi Arduino Mega 2560.....	31
Tabel 2.3 Nama dan fungsi dari pin modul A4988.....	34
Tabel 3.1 Kebutuhan Daya Komponen.....	49
Tabel 4.1 Hasil Pengujian Deteksi Pinggir Objek dengan Canny Menggunakan Python.	63
Tabel 4.2 Hasil Pengujian Pencarian dan Seleksi Kontur.....	64
Tabel 4.3 Hasil Pengujian Ukuran Luas Permukaan Objek.....	66
Tabel 4.4 Hasil Pengujian Koordinat Objek	67
Tabel 4.5 Hasil pengujian gerak masing masing <i>joint</i>	70
Tabel 4.6 Data Hasil Pengujian Gerak Robot	73
Tabel 4.7 Hasil pengujian kinematik mundur metode analisis.	74
Tabel 4.8 Hasil Pengukuran Waktu Perhitungan Kinematik Mundur.	76
Tabel 4.9 Konfigurasi Masing-Masing Motor Pada <i>Joint</i>	77
Tabel 4.10 Hasil Pengukuran Waktu Perhitungan Kinematik Mundur dengan Aksi <i>Joint</i>	77
Tabel 4.11 Spesifikasi objek penelitian.	78
Tabel 4.12 Hasil percobaan memindahkan benda.....	80

BAB I

PENDAHULUAN

1.1 Latar Belakang

Robot merupakan suatu sistem terintegrasi yang membuat suatu sistem mampu mengerjakan suatu tindakan berdasarkan masukannya. Industri dengan sistem robot menjadi bagian penting dalam proses produksinya[1]. Robot dalam industri dapat didefinisikan sebagai kendali komputer dengan manipulator yang dirancang untuk meniru gerakan manusia, seperti gerakan untuk melaksanakan sejumlah tugas industri yang berbeda tanpa campur tangan manusia[2][3].

Selama beberapa dekade terakhir, penelitian tentang robotika telah memberikan dampak yang cukup besar pada bidang industri. Secara singkat prestasi didapatkan dari penelitian robotika dalam aplikasi dunia industri memberikan dampak baik dalam menciptakan hasil pekerjaan yang jauh lebih bersih, sisi keamanan yang memiliki risiko bahaya yang kecil dan kesulitan dalam proses pekerjaan[1][4].

Salah satu jenis robot yang cukup sering ditemui pada industri adalah robot lengan (manipulator).SCARA (*Selective Compliance Assemby Robot Arm*) merupakan robot yang menirukan fungsi dan kinerja dari lengan manusia dengan tujuan khusus untuk menyusun atau memindahkan benda. Robot dapat dilengkapi dengan sensor tertentu agar dapat bekerja sesuai dengan tujuan dirancangnya robot tersebut. Kamera dapat disematkan pada robot SCARA agar sistem dapat memindahkan barang sesuai kebutuhan[5][6].

Kinematika mundur merupakan masalah utama dalam mengendalikan robot manipulator. Perhitungan kinematika mundur secara langsung dibutuhkan dalam algoritma pengendalian robot, perhitungan kinematika mundur yang rumit terkadang membutuhkan kemampuan komputasi yang mumpuni serta akan membutuhkan waktu pemrosesan yang lama apabila ingin dihitung secara langsung. Terdapat dua metode utama dalam menyelesaikan perhitungan kinematika mundur, yaitu perhitungan dengan analisa dan numerik. Pada metode analisa perhitungan dilakukan dengan mengamati bentuk geometri pada robot manipulator sedangkan pada metode numerik perhitungan dilakukan dengan

melakukan iterasi. Perhitungan kinematika mundur yang baik hendaknya tidak memerlukan biaya komputasi yang tinggi serta perhitungan dapat diselesaikan dengan cepat[7].

Robot manipulator hendaknya dapat bekerja secara otomatis apabila masukan telah tersedia, pada robot pemindah barang sebuah sistem dapat ditambahkan ke robot agar robot mampu memindahkan barang saat tersedia. Selain memindahkan, robot juga juga diharapkan mampu memindahkan barang sesuai ukuran yang diinginkan. *Image subtraction* adalah metode yang memisahkan antara objek dengan latarnya, metode tersebut dapat digunakan pada proses pemindaian dan pemindahan barang menggunakan robot lengan dengan kamera sebagai sensornya. *Image subtraction* dapat diterapkan untuk mendeteksi suatu objek dan lengan robot dapat memindahkan barang tersebut. Metode *image subtraction* digunakan pada penelitian untuk mendeteksi suatu objek. hasil pendektsian akan diberi tanda dan titik tengah sehingga didapatkan koordinatnya. Koordinat tersebut dapat digunakan untuk menggerakan robot lengan agar menuju objek yang telah terdeteksi[8].

Pendeteksian pinggir canny merupakan algoritma untuk mendeteksi pinggir suatu objek di dalam suatu citra digital, algoritma ini dapat digunakan dalam proses *image subtraction*. Dalam suatu citra terkadang terdapat beberapa hal yang dapat membuat suatu objek susah untuk di ekstrak dari citra tersebut. Salah satu gangguan dalam subtraksi citra adalah bayangan serta tidak kontrasnya perbedaan antara suatu objek dengan *background* citra. Pengolahan citra *Gaussian blur* dan penggunaan ruang warna HSV (*Hue Saturation Value*) dapat diterapkan pada proses *image subtraction* sehingga diharapkan dapat menghasilkan koordinat tengah dan ukuran objek yang lebih akurat. Dari beberapa permasalahan diatas maka akan dilakukan penelitian dengan judul **“Perancangan Robot Lengan Pemindah Benda Berdasarkan Ukuran”**

1.2 Rumusan Masalah

1. bagaimana bentuk rancangan purwarupa robot lengan “SCARA” dengan tiga DOF yang mampu memindahkan benda.
2. Metode apa yang paling efisien antara solusi analitis dan numerik dalam mencari solusi kinematika mundur pada robot SCARA dengan tiga DOF.
3. apakah metode *Canny edge detection*, *Gaussian blur* dan penggunaan ruang warna HSV dapat digunakan untuk mendeteksi, mengukur ukuran serta mencari koordinat tengah dari sebuah objek di citra dengan menggunakan bahasa pemrograman Python.

1.3 Tujuan Penelitian

1. Membuat sebuah purwarupa robot lengan SCARA (*Selective Compliant Assembly Robot Arm*) dengan derajat tiga DOF yang mampu memindahkan benda.
2. Membandingkan performa metode *closed-form* dan *numerical-iteration* dengan menguji prilaku gerak robot, lama waktu dan hasil perhitungan kinematika mundurnya.
3. Menganalisis performa aplikasi *image subtraction* yang diterapkan dengan bahasa pemrograman Python dalam mendeteksi keberadaan, mengukur ukuran serta menentukan koordinat tengah sebuah benda.

1.4 Batasan Masalah

1. Benda yang dipindahkan robot lengan terbuat dari kayu dengan berat serta ukuran tertentu, benda hanya diberikan satu warna saja pada semua sisinya, serta benda hanya berbentuk kubus dan balok.
2. Area kerja robot terbatas, serta area pengambilan citra dengan kamera juga terbatas.
3. Tidak di bahas kecepatan gerak robot lengan, pengaruh berat benda terhadap kebutuhan daya listrik, Tidak dibahas efisiensi listrik dari sistem.

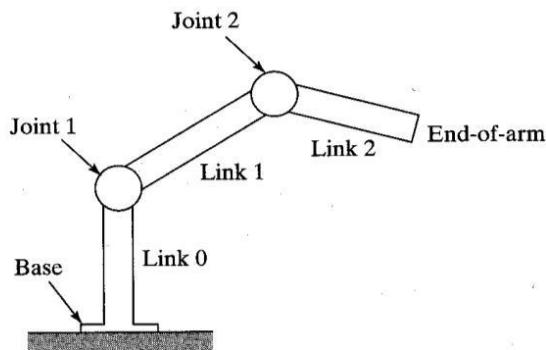
BAB II

TINJAUAN PUSTAKA

Pada bab 2 ini berisikan teori mengenai penelitian, mulai dari teori kinematika pada manipulator robot lengan dan contoh penerapannya serta informasi dari beberapa perangkat keras dan lunak yang digunakan.

2.1 Rangka Manipulator Robot Lengan.

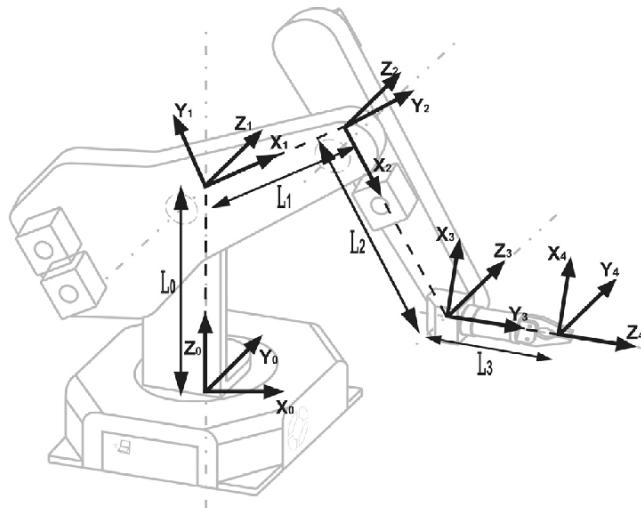
Robot lengan memiliki banyak variasi tergantung dari penggunaannya, perbedaan variasi ini menyebabkan konstruksi dari robot lengan juga berbeda-beda. Robot lengan biasanya terdiri dari bagian kaku (*rigid body*) atau bernama *link* dan sambungan antara bagian kaku itu sendiri yang bernama *joint*. Ilustrasi robot lengan dapat dilihat pada Gambar 2.1



Gambar 2.1 Ilustrasi Manipulator Robot Lengan dengan 2 *Degree Of Freedom*
(www.researchgate.net).

Pada Gambar 2.1 terdapat ilustrasi atau diagram dari manipulator robot lengan dengan 2 *degree of freedom*. Manipulator merupakan bagian yang mengendalikan pergerakan dari robot lengan itu sendiri, pada Gambar 2.1 manipulatornya terdiri dari *joint 1*, *joint 2*, *link 0*, *link 1*, dan *link 2*. *Degree of freedom* (DOF) merupakan derajat atau jumlah kebebasan gerak dari robot lengan, *degree of freedom* biasanya dihitung dari jumlah *joint* tetapi cara seperti tidak selalu benar dikarenakan ada beberapa jenis *joint* yang mampu memiliki lebih dari satu *degree of freedom*. Pada bagian kedua ujung robot lengan terdapat *base* yang merupakan dudukan dari robot lengan dan pada ujung lainnya terdapat *end of arm* atau biasa disebut *end of effector* yang merupakan bagian pada robot untuk diletakan suatu perangkat agar robot lengan dapat berinteraksi dengan objek lain.

Pada *base*, *end of effector* dan masing-masing *joint* pada robot lengan terdapat *frame*. *Frame* merupakan visualisasi bantuan untuk mempermudah menentukan gerak dari robot lengan, visualisasi *frame* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Visualisasi *Frame* Pada Manipulator Robot Lengan
(mrebelo.com).

Pada Gambar 2.2 dapat dilihat pada masing-masing *joint*, *base* dan *end of effector* terdapat visualisasi *frame* dimana *frame* ini berbentuk seperti koordinat 3 dimensi pada ruang *Euclidean*. Pada *base* terdapat satu *frame* yang pada Gambar 2.2 memiliki urutan *frame* ke-nol dengan komposisi sumbu (x_0, y_0, z_0). *Frame* pada *base* ini biasanya disebut sebagai *inertial frame*, selanjutnya pada masing-masing *joint* dan *end of effector* juga terdapat *frame*.

Penggunaan robot lengan tujuan nya adalah agar bagian pada *end of effector* berada pada posisi atau koordinat yang sesuai dengan objek yang akan berinteraksi dengan lengan robot, objek yang akan berinteraksi dengan robot lengan memiliki koordinatnya tersendiri, untuk objek koordinatnya titik pusat disamakan dengan koordinatnya koordinat *inertial frame*, dikarenakan *inertial frame* tidak akan mengalami perubahan apa pun saat salah satu *joint* pada robot lengan mengalami pergerakan. Bagian yang akan berinteraksi dengan objek adalah bagian *end of effector*, seperti yang telah diketahui *end of effector* memiliki *frame* yang berbeda dengan *base*, oleh karena itu perlu di cari relasi antara *frame* pada *end of effector*, *joint*, dan *base* agar *end of effector* dapat berada pada koordinat yang sama dengan objek saat akan berinteraksi. Kajian yang mempelajari relasi antara *frame* pada

joint, *end of effector*, dan *base* saat robot lengan mengalami pergerakan adalah kinematika robot lengan[9].

2.2 Kinematika Pada Manipulator Robot Lengan.

kinematika pada manipulator robot lengan adalah kajian yang mengamati hubungan antara *joint* dan *frame* saat mereka mengalami pergerakan dan melihat hasil pergerakan setelah semua *joint* dan *link* selesai bergerak. Ada dua macam kinematika pada manipulator robot lengan yaitu kinematika maju (*forward kinematics*) dan kinematika mundur (*inverse kinematics*). Kinematika maju akan menentukan letak atau koordinat *end of effector* saat salah satu atau semua *joint* pada manipulator robot lengan diberi pergerakan, sebaliknya kinematika mundur adalah proses menentukan pergerakan apa yang harus dilakukan masing-masing *joint* pada manipulator robot lengan agar *end of effector* berada pada koordinat yang diinginkan. Pergerakan dari *joint* pada manipulator dapat ditulis dalam bentuk matematika seperti berikut.

1. Matriks perputaran (*Rotational Matrice*).

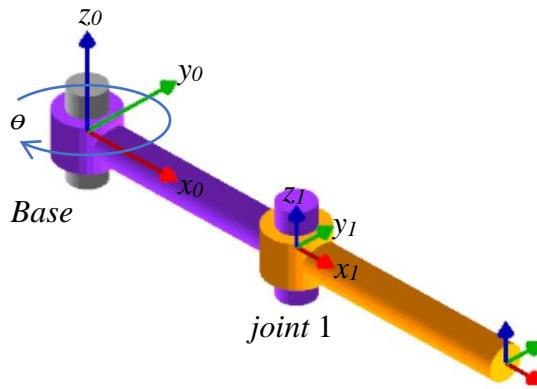
Matriks perputaran merupakan matriks 3x3 yang berisikan informasi perputaran dan orientasi dari *frame* suatu *joint* relatif terhadap *frame* dari *joint* atau *base* lainnya. Untuk membuat sebuah matriks perputaran maka dibutuhkan setidaknya dua *frame*. Matriks perputaran sendiri dibentuk dari hasil kali titik antara matriks orientasi dan matriks putarnya. matriks putar hanya terdiri dari tiga jenis matriks saja untuk semua jenis *joint* pada robot lengan, matriks putar dapat dilihat pada persamaan 2.1 – 2.3.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \quad 2.1$$

$$R_y = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad 2.2$$

$$R_z = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad 2.3$$

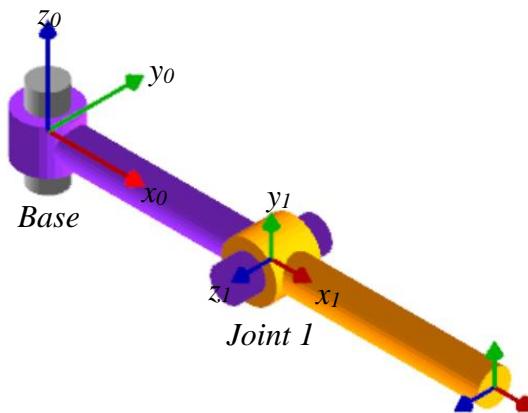
Untuk menjelaskan persamaan matriks 2.1 – 2.3 dapat dilihat Gambar 2.3



Gambar 2.3 Manipulator 2 DOF Jenis *Revolute*.

Pada Gambar 2.3 apabila ingin dibuat matriks putarnya maka harus dianggap *frame 0* pada *base* dan *frame 1* pada *joint 1* berada pada pusat yang sama, dikarenakan pada matriks perputaran hanya melihat nilai perputaran dan orientasi. Pada Gambar 2.3 dikarenakan pada *frame 0* sumbu z_0 merupakan pusat rotasinya maka matriks putar *joint 1* relatif terhadap *base* adalah persamaan 2.3, persamaan 2.1 digunakan saat sumbu x menjadi pusat rotasi dan persamaan 2.2 digunakan saat sumbu y menjadi pusat rotasi. Untuk *joint* atau *base* yang jenisnya selain sambungan putar (*revolute joint*) seperti sambungan geser (*prism joint*) maka matriks putarnya merupakan sebuah matriks identitas.

Selain matriks putar, dibutuhkan juga matriks orientasi untuk menghasilkan matriks perputaran, matriks orientasi dibutuhkan untuk menyatakan keadaan awal orientasi suatu *frame* relatif terhadap *frame* lain. Pada Gambar 2.3 *frame 1* memiliki orientasi yang sama dengan *frame 0*, dikatakan sama dikarenakan saat sudut θ belum mengalami perputaran, sumbu x pada *frame 1* memiliki arah yang sama dengan sumbu x pada *frame 0*, begitu juga dengan sumbu y dan z , untuk kasus seperti ini matriks orientasinya merupakan matriks identitas 3×3 . Untuk kasus lain dapat dilihat pada Gambar 2.4.



Gambar 2.4 Manipulator dengan *Joint 1* Berbeda Orientasi.

Pada Gambar 2.4 matriks orientasinya berbeda dengan Gambar 2.3, pada Gambar 2.4 saat belum terjadi putaran apa pun pada θ hanya sumbu x pada *frame* 1 yang searah dengan sumbu x pada *frame* 0, sedangkan sumbu y dan z berbeda arah, untuk dapat membuat matriks orientasi pada Gambar 2.4 dapat dilihat pada penyusunan matriks berikut.

$$R_o = \begin{bmatrix} o_{11} & o_{12} & o_{13} \\ o_{21} & o_{22} & o_{23} \\ o_{31} & o_{32} & o_{33} \end{bmatrix} \quad 2.4$$

Pada persamaan 2.4 elemen matriks o_{11} merupakan relasi orientasi sumbu x pada *frame* 1 terhadap sumbu x pada *frame* 0, dikarenakan kedua sumbu berada pada arah atau orientasi yang sama maka nilai $o_{11} = 1$, untuk o_{12} merupakan relasi orientasi sumbu y pada *frame* 1 terhadap sumbu x pada *frame* 0, dikarenakan kedua sumbu ini saling tidak searah maka nilai untuk elemen $o_{12} = 0$, untuk elemen lainnya dapat dilihat pada Tabel 2.1.

Tabel 2.1 Penjelasan Elemen Matriks Orientasi

Elemen	Penjelasan
o_{11}	Proyeksi sumbu x <i>frame</i> 1 terhadap sumbu x <i>frame</i> 0
o_{12}	Proyeksi sumbu y <i>frame</i> 1 terhadap sumbu x <i>frame</i> 0
o_{13}	Proyeksi sumbu z <i>frame</i> 1 terhadap sumbu x <i>frame</i> 0
o_{21}	Proyeksi sumbu x <i>frame</i> 1 terhadap sumbu y <i>frame</i> 0
o_{22}	Proyeksi sumbu y <i>frame</i> 1 terhadap sumbu y <i>frame</i> 0
o_{23}	Proyeksi sumbu z <i>frame</i> 1 terhadap sumbu y <i>frame</i> 0

Elemen	Penjelasan
o_{31}	Proyeksi sumbu x <i>frame 1</i> terhadap sumbu z <i>frame 0</i>
o_{32}	Proyeksi sumbu y <i>frame 1</i> terhadap sumbu z <i>frame 0</i>
o_{33}	Proyeksi sumbu z <i>frame 1</i> terhadap sumbu z <i>frame 0</i>

Sehingga matriks orientasi *frame 1* relatif terhadap *frame 0* untuk gambar 2.4 dapat dilihat pada persamaan 2.5

$$R_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \quad 2.5$$

Matriks perputaran dapat diperoleh dari hasil kali titik antara matriks putar dan matriks orientasi, matriks perputaran antara *frame 1* relatif terhadap *frame 0* dapat dihitung dengan persamaan 2.6.

$$R_n^m = R_{p_n}^m \circ R_{o_n}^m \quad 2.6$$

Dimana : R_n^m adalah matriks perputaran *frame n* relatif terhadap *frame m*

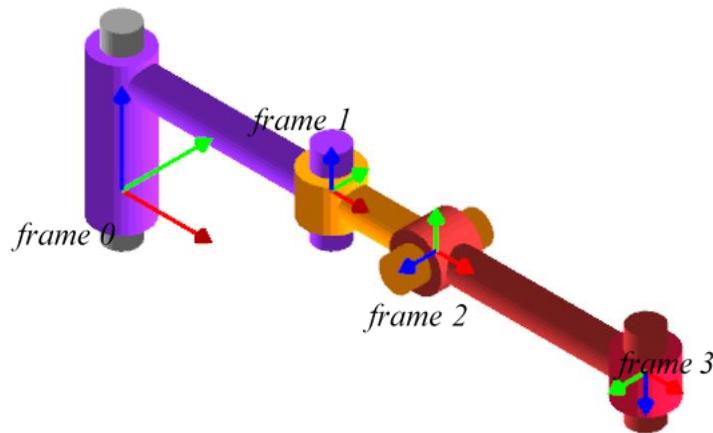
$R_{p_n}^m$ matriks putar *frame n* relatif terhadap *frame m*

$R_{o_n}^m$ matriks orientasi *frame n* relatif terhadap *frame m*

Sehingga matriks perputaran *frame 1* relatif terhadap *frame 0* pada Gambar 2.4 dapat dihitung seperti pada persamaan 2.7

$$R_1^0 = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \circ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ \sin \theta & 0 & \cos \theta \\ 0 & -1 & 0 \end{bmatrix} \quad 2.7$$

Pada persamaan 2.7 hanya mempresentasikan relasi antara gerak perputaran antara *frame 1* dan *frame 0* pada Gambar 2.4, sedangkan manipulator pada Gambar 2.4 memiliki tiga *frame* mulai dari *base* hingga *end of effector*-nya. Semua relasi antar *frame* pada manipulator harus ditentukan agar *end of effector* dapat berada pada koordinat yang sama dengan objek yang akan berinteraksi dengan robot lengan, maka matriks perputarannya harus matriks yang menyatakan hubungan *frame* pada *end of effector* relatif terhadap *frame* pada *base*[10]



Gambar 2.5 Manipulator Dengan 4 DOF.

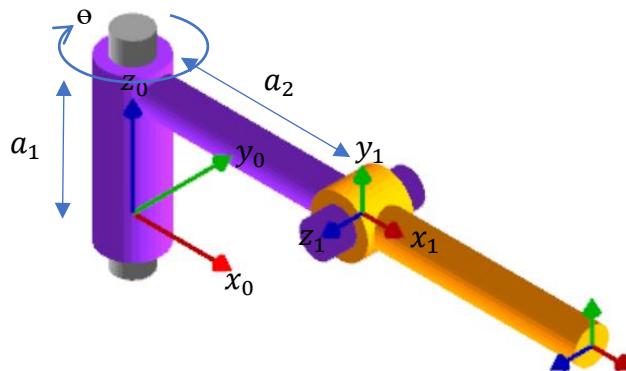
Pada Gambar 2.5 terdapat 4 *frame* sehingga matriks perputaran totalnya dihitung dengan persamaan 2.8

$$R_3^0 = R_1^0 R_2^1 R_3^2 \quad 2.8$$

Matriks R_3^0 akan menyatakan kedudukan perputaran *frame* 3 relatif terhadap *frame* 0

2. Vektor perpindahan (*Displacement Vector*).

Perputaran bukanlah satu-satunya pergerakan yang dapat terjadi pada sebuah *joint*. sebuah *joint* juga dapat bergerak dengan bergeser atau berpindah tempat terutama pada *joint* yang berjenis geser (*prism joint*). Sama seperti matriks perputaran, vektor perpindahan merupakan vektor tiga dimensi yang menyatakan relasi gerak pada suatu *frame* relatif terhadap *frame* lainnya, perbedaannya vektor perpindahan menyatakan gerak pergeseran atau perpindahan tempat sedangkan matriks perputaran menyatakan gerak perputaran.



Gambar 2.6 Manipulator Dengan Variasi Panjang *Link*.

Format vektor perpindahannya dapat dilihat pada persamaan 2.9

$$d_n^m = \begin{bmatrix} d_{11} \\ d_{12} \\ d_{13} \end{bmatrix} \quad 2.9$$

d_n^m : vektor perpindahan *frame n* relatif terhadap *frame m*

d_{11} : perpindahan pusat *frame 1* relatif terhadap sumbu x pada *frame 0*

d_{12} : perpindahan pusat *frame 1* relatif terhadap sumbu y pada *frame 0*

d_{13} : perpindahan pusat *frame 1* relatif terhadap sumbu z pada *frame 0*

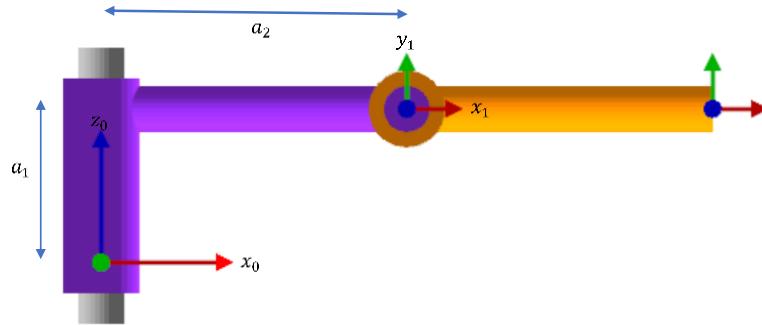
Pada matriks perputaran seperti pada gambar 2.3 untuk membuat matriks perputaran *frame 1* relatif terhadap *frame 0* maka pusat pada *frame 1* dianggap saling bertindihan lalu dicari perbedaan orientasi dan perputaran antara *frame 1* dan *frame 0*, pada vektor perpindahan ini kedua *frame* tidak dianggap bertindihan karena pada vektor perpindahan berisikan informasi perbedaan koordinat antara *frame 1* dengan *frame* yang lain. Pada Gambar 2.6 dapat dibuat vektor perpindahan antara *frame 1* relatif terhadap *frame 0* seperti pada persamaan 2.10.

$$d_1^0 = \begin{bmatrix} a_2 \cos \theta \\ a_2 \sin \theta \\ a_1 \end{bmatrix} \quad 2.10$$

Pada persamaan 2.7 elemen vektor d_{11} dan d_{12} tergantung dari nilai θ yaitu perputaran yang terjadi pada *base*, untuk memahami lebih dalam mengapa pada vektor perpindahan dipengaruhi oleh sudut θ amati Gambar 2.7. Gambar 2.7 merupakan manipulator yang sama seperti pada Gambar 2.6 akan tetapi dilihat dari

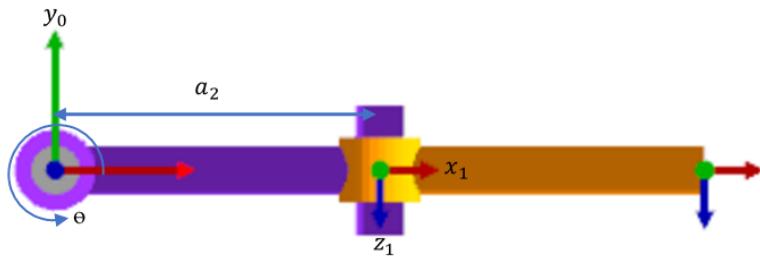
sisi depan saja. Pada Gambar 2.7 *frame* 1 mengalami perpindahan sebesar a_1 pada sumbu z_0 dan a_2 pada sumbu x_0 , sedangkan pada sumbu y_0 pusat *frame* 1 tidak mengalami pergeseran sehingga apabila dibuat vektor perpindahannya dapat dilihat pada persamaan 2.8.

$$d_1^0 = \begin{bmatrix} a_2 \\ 0 \\ a_1 \end{bmatrix} \quad 2.11$$



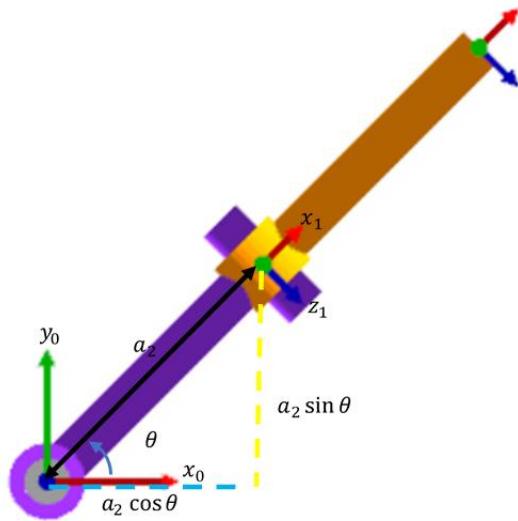
Gambar 2.7 Manipulator Dilihat Dari Sisi Depan.

Vektor pada persamaan 2.8 tidak selalu dapat digunakan dikarenakan *base* pada manipulator pada Gambar 2.6 merupakan *base* yang dapat berputar, sehingga saat diberi putaran, pusat *frame* 1 tidak lagi hanya berpindah relatif dari sumbu x_0 dan z_0 saja tetapi juga berpindah relatif terhadap sumbu y_0 . Amati Gambar 2.8, dimana gambar ini merupakan manipulator pada Gambar 2.6 tetapi dilihat dari atas. Pada Gambar 2.8 saat *base* tidak berputar atau nilai $\theta = 0$, perpindahan pusat *frame* 1 relatif terhadap *frame* 0 sama seperti pada vektor persamaan 2.8, saat *base* mengalami perputaran seperti pada Gambar 2.9, pusat *frame* 1 mengalami perpindahan di semua sumbu pada *frame* 0.



Gambar 2.8 Manipulator Ditinjau Dari Sisi Atas

Perpindahan *frame* 1 terhadap sumbu x_0 setelah *base* berputar, tidak lagi sama dengan a_2 . Pada Gambar 2.9 besar perpindahan *frame* 1 terhadap sumbu x_0 ditunjukkan oleh garis putus-putus bewarna biru, a_2 tidak lagi menjadi jarak perpindahan *frame* 1 terhadap sumbu x_0 dikarenakan saat *base* diberi putaran sebesar θ , a_2 tidak lagi sejajar dengan sumbu x_0 .



Gambar 2.9 Manipulator Tampak Atas Saat Base Diberi Perputaran Sebesar θ

Garis berwarna kuning pada Gambar 2.9 menyatakan besar perpindahan *frame* 1 terhadap sumbu y_0 . Besar perpindahan *frame* 1 terhadap sumbu x_0 dan y_0 dapat dihitung dengan persamaan trigonometri dimana a_2 menjadi panjang sisi miringnya. dengan analogi ini vektor perpindahan pada persamaan 2.10 akan selalu

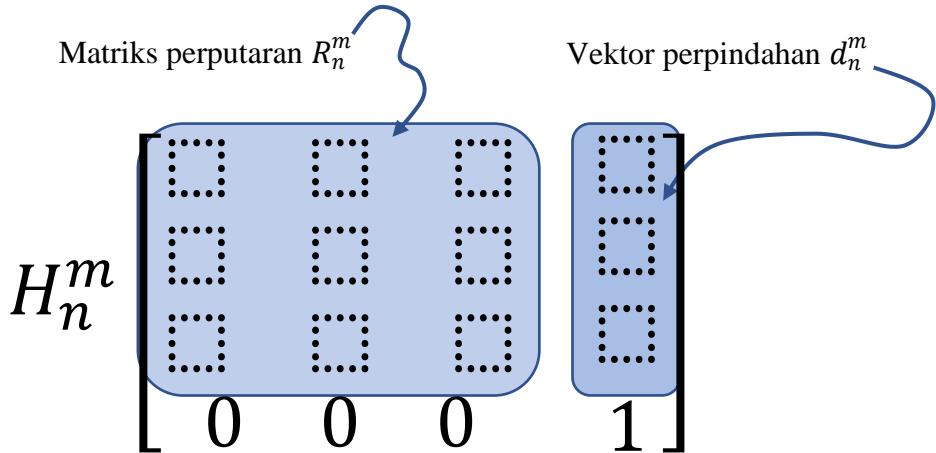
benar saat nilai θ pada manipulator di Gambar 2.6 berubah-ubah. Untuk perpindahan *frame* 1 terhadap sumbu z_0 tidak tergantung dari θ dikarenakan sumbu z_0 menjadi pusat perputaran dari *base*, sehingga apabila nilai θ berubah-ubah, nilai perpindahan *frame* 1 terhadap sumbu z_0 tidak akan ikut berubah.

Sama seperti matriks perputaran, Vektor perpindahan semua *frame* pada sebuah manipulator harus dihitung keterkaitannya. Pada matriks perputaran untuk mencari hubungan *frame* mulai dari *frame* pada *base* hingga *frame* pada *end of effector*, dihitung keterkaitan *frame* secara estafet mulai dari *base* hingga *end of effector* lalu masing-masing matriksnya dapat langsung dikalikan sehingga menghasilkan matriks perputaran *end of effector* relatif terhadap *base*. Untuk vektor perpindahan teknik yang sama tidak berlaku, untuk menghitung vektor perpindahan *end of effector* relatif terhadap *base* tidak dapat dilakukan dengan mengalikan masing-masing vektor perpindahan antar *frame*, vektor perpindahan *end of effector* relatif terhadap *base* dapat ditentukan dengan melibatkan matriks perputaran[10][11].

3. *Homogeneous transformation matrices.*

Homogeneous transformation matrices (matriks transformasi homogen /HTM) merupakan matriks yang menyatakan relasi gerak perputaran dan perpindahan suatu *frame* relatif terhadap *frame* lainnya. Dapat dikatakan *homogeneous transformation matrix* merupakan gabungan dari matriks perputaran dan vektor perpindahan. Pada sub bab sebelumnya dikatakan bahwa vektor perpindahan tidak dapat dikaitkan langsung dengan vektor perpindahan lainnya, tetapi dengan memasukkan vektor perpindahan ke dalam matriks transformasi homogen, maka dapat dihitung keterkaitan antar vektor perpindahan dikarenakan matriks transformasi homogen dapat dikalikan (dihubungkan) langsung dengan matriks transformasi homogen lainnya.

Matriks transformasi homogen dapat menyatakan kedudukan *end of effector* secara tepat relatif terhadap *base* saat terjadi perputaran dan pergeseran di salah satu atau semua *joint* dan *base* pada manipulator robot lengan. Komposisi matriks transformasi homogen dapat diamati pada Gambar 2.10.



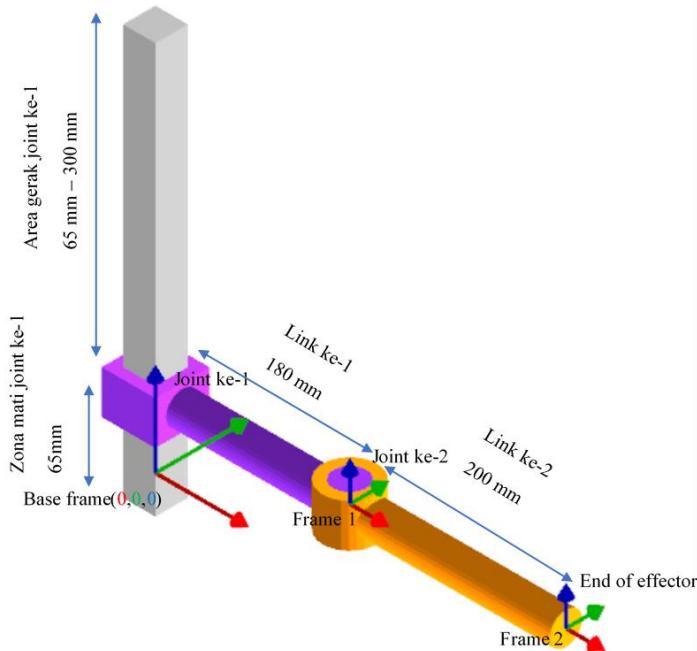
Gambar 2.10 Komposisi *Homogeneous Transformation Matrice*

Untuk membuat sebuah matriks transformasi homogen maka perlu dihitung dulu matriks perputaran dan vektor perpindahannya. Apabila persamaan 2.7 dan persamaan 2.10 dibuatkan sebuah HTM dapat diamati pada persamaan 2.12. Elemen h_{14} pada persamaan 2.12 menyatakan koordinat *end of effector* terhadap *base* pada sumbu x, elemen h_{24} terhadap sumbu y dan elemen h_{34} terhadap sumbu z

$$H_1^0 = \begin{bmatrix} \cos \theta & 0 & -\sin \theta & a_2 \cos \theta \\ \sin \theta & 0 & \cos \theta & a_2 \sin \theta \\ 0 & -1 & 0 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad 2.12$$

4. kinematika maju (*foward kinematics*).

kinematika maju merupakan algoritma untuk menentukan koordinat x, y, dan z *end of effector* suatu manipulator robot lengan relatif terhadap *base*-Nya. dengan memasukkan masing-masing nilai variabel *joint* pada sebuah manipulator ke matriks transformasi homogennya akan diperoleh koordinat x, y, dan z *end of effector*-Nya dengan melihat elemen h_{14} , h_{24} , dan h_{34} . Berikut contoh penerapan algoritma kinematika maju pada sebuah manipulator seperti pada Gambar 2.11.



Gambar 2.11 Manipulator Dengan Dua DOF Tipe Geser dan Putar.

Gambar 2.11 menunjukkan sebuah manipulator robot lengan dengan dua derajat kebebasan, untuk menerapkan kinematika maju pada manipulator ini terlebih dahulu harus dibuat matriks transformasi homogennya, untuk membuat HTM-nya pertama harus dibuat matriks perputaran dan vektor perpindahannya. Berikut matriks perputaran dan vektor perpindahannya.

$$R_1^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad 2.13$$

Persamaan 2.13 merupakan matriks perputaran *frame 1* relatif terhadap *frame 0* atau *frame base*, matriks pada persamaan 2.13 diperoleh dengan mengalikan matriks putar dan matriks orientasi *frame 1* relatif terhadap *frame 0*. Diperoleh matriks identitas karena *joint ke-1* merupakan *joint* tipe geser dan orientasi antara *frame 1* dan *0* sudah sama. Setelah diperoleh matriks perputaran R_1^0 selanjutnya dihitung matriks perputaran R_2^1 . *Joint ke-2* merupakan *joint* tipe putar yang berpusat pada sumbu z serta orientasi antara *frame 2* dan *1* telah sama sehingga diperoleh matriks perputarannya seperti pada persamaan 2.14.

$$R_2^1 = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad 2.14$$

Setelah diperoleh matriks perputarannya dihitung juga vektor perpindahannya, vektor perpindahan *frame* 1 relatif terhadap *frame* 0 dapat dilihat pada persamaan 2.15. Sedangkan vektor perpindahan *frame* 2 relatif terhadap *frame* 1 dapat dilihat pada persamaan 2.16.

$$D_1^0 = \begin{bmatrix} 180 \\ 0 \\ 65 + Z \end{bmatrix} \quad 2.15$$

$$D_2^1 = \begin{bmatrix} 200 \cos \theta \\ 200 \sin \theta \\ 0 \end{bmatrix} \quad 2.16$$

Variabel *z* pada persamaan 2.15 merupakan nilai variabel seberapa banyak *joint* ke-1 bergeser dan variabel θ pada persamaan 2.16 merupakan nilai seberapa banyak *joint* ke-2 berputar. Setelah diperoleh matriks perputaran dan vektor perpindahannya dapat langsung dibuat matriks transformasi homogennya. HTM *frame* 1 relatif terhadap *frame* 0 serta HTM *frame* 2 relatif terhadap *frame* 1 dapat dilihat pada persamaan 2.17 dan 2.18.

$$H_1^0 = \begin{bmatrix} 1 & 0 & 0 & 180 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 65 + Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad 2.17$$

$$H_2^1 = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 200 \cos \theta \\ \sin \theta & \cos \theta & 0 & 200 \sin \theta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad 2.18$$

Untuk menerapkan kinematika maju diperlukan HTM antara *end of effector* dan *base* sehingga matriks pada persamaan 2.17 dan 2.18 dapat dikalikan agar diperoleh HTM *frame* 2 relatif terhadap *frame* 0 seperti pada persamaan 2.19.

$$H_2^0 = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & (200 \cos(\theta)) + 180 \\ \sin \theta & \cos \theta & 0 & 200 \sin \theta \\ 0 & 0 & 1 & 65 + z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad 2.19$$

Pada penjelasan sebelumnya telah dijelaskan elemen h_{14}, h_{24} , dan h_{34} pada persamaan 2.19 berturut turut adalah koordinat *end of effector* pada sumbu x, y, dan z relatif terhadap *base*. Pada kinematika maju tujuannya adalah mencari nilai koordinat x, y, dan z ini saat variabel *z* dan θ ditentukan nilainya. Saat nilai *z* = 120 mm dan θ = 90° maka nilai koordinat x,y,dan z *end of effector* dapat dihitung seperti berikut

$$x = 200 * \cos(90^\circ) + 180 = 180 \text{ mm}$$

$$y = 200 * \sin 90^\circ = 200 \text{ mm}$$

$$z = 65 + 120 = 185 \text{ mm}$$

Setelah dimasukkan nilai variabel $z = 120$ dan $\theta = 90^\circ$ maka koordinat *end of effector* relatif terhadap *base* adalah (180, 200, 185). Dengan kinematika maju dapat ditentukan atau diprediksi dimana koordinat *end of effector* saat salah satu atau seluruh *joint* pada manipulator diubah nilainya[11].

5. kinematika mundur (*inverse kinematics*).

kinematika mundur merupakan kebalikan dari kinematika maju, pada kinematika mundur memiliki tujuan menentukan nilai variabel *joint* dengan mengetahui koordinat *end of effector* nya. Pada kinematika maju, hanya memiliki satu solusi yaitu koordinat *end of effector* saat nilai variabel *joint* dimasukkan ke dalam matriks transformasi homogennya. Pada kinematika mundur biasanya dapat diperoleh lebih dari satu solusi, dikarenakan beberapa kombinasi nilai *joint* dapat menghasilkan nilai koordinat yang sama pada *end of effector*-nya. pada kinematika mundur dilakukan iterasi dengan menguji semua nilai variabel *joint* sehingga diperoleh nilai koordinat *end of effector* yang diinginkan, berikut contoh penggunaan kinematika mundur pada suatu manipulator.

Manipulator yang digunakan dapat dilihat pada Gambar 2.11 dan matriks transformasi homogennya dapat dilihat pada persamaan 2.19. *End of effector* harus berada pada koordinat (280, 173, 100) mm, dengan menggunakan HTM pada persamaan 2.19 dan melakukan iterasi pada elemen h_{14}, h_{24} , dan h_{34} diperoleh nilai variabel *joint* yang memenuhi seperti berikut.

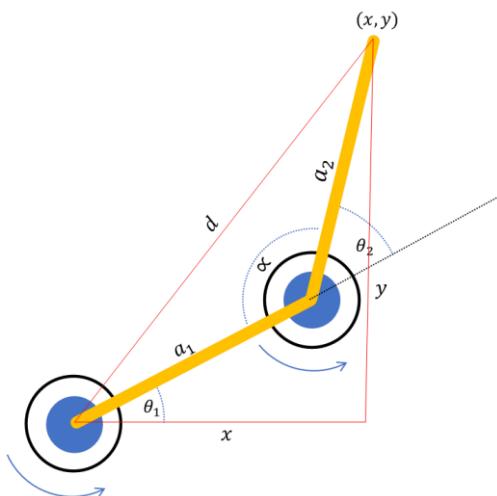
$$\theta = 60$$

$$Z = 35 \text{ mm}$$

Setelah dilakukan iterasi mulai dari sudut 0° hingga 360° pada sudut θ dan iterasi mulai dari nilai 0 hingga 235 pada Z diperoleh nilai $\theta = 60^\circ$ dan $Z = 35 \text{ mm}$ untuk memenuhi koordinat *end of effector* (280, 173, 100) mm. Pada iterasi hanya diperoleh satu solusi saja, hal ini dikarenakan manipulator yang memiliki derajat kebebasan yang kecil (2 DOF), untuk derajat kebebasan yang lebih tinggi akan diperoleh lebih dari satu solusi untuk kinematika mundurnya[12], [13]

6. Closed-form inverse kinematics

Pada sub-bab 5, kinematika mundur dilakukan dengan melakukan iterasi pada persamaan yang diperoleh dari elemen pada matriks transformasi homogen. Dalam proses ini akan memakan waktu untuk proses iterasinya serta dapat menghasilkan lebih dari satu solusi untuk satu permasalahan (redundancy). Untuk jenis manipulator yang memiliki jumlah DOF yang kecil, metode iterasi tidak cocok digunakan karena robot jenis ini jarang mengalami *redundancy*, sehingga perhitungan dengan metode iterasi hanya akan menambah waktu perhitungan. Metode lain yang dapat digunakan yaitu metode analitis (formula tertutup), metode ini menyediakan satu atau dua solusi untuk satu permasalahan kinematika mundur, dengan ini pemilihan solusi akan lebih mudah dilakukan serta perhitungan tidak memakan banyak waktu. Metode analisis dilakukan dengan mempelajari bentuk geometri dari manipulator lalu menggunakan fungsi-fungsi trigometri untuk menentukan nilai sudut suatu joint. Metode analisis cocok digunakan untuk manipulator dengan jumlah DOF yang kecil[14], [15], berikut contoh analisis manipulator untuk menentukan persamaan solusi kinematika mundur pada manipulator dengan 2 DOF, manipulator dapat diamati pada Gambar 2.12.



Gambar 2.12 Manipulator tipe RR

Langkah pertama untuk menganalisis yaitu mengamati segitiga merah pada Gambar 2.12, dimana pada segitiga tersebut untuk menentukan panjang dari sisi “ d ” dapat menggunakan persamaan pythagoras sehingga :

$$d^2 = x^2 + y^2 \quad 2.20$$

Dengan menggunakan hukum cosinus panjang sisi “ d ” juga dapat dihitung dengan

$$d^2 = a_1^2 + a_2^2 - 2a_1a_2 \cos \alpha$$

$$\cos \alpha = \frac{a_1^2 + a_2^2 - d^2}{2a_1a_2} \quad 2.21$$

Dari persamaan 2.21 dapat ditentukan besar sudut θ_2 dimana :

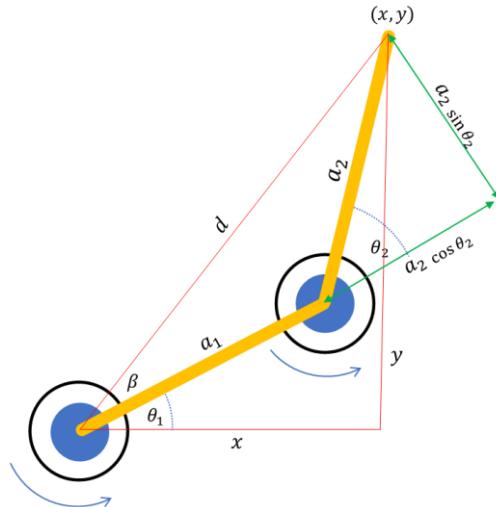
$$\theta_2 = 180^\circ - \alpha$$

$$\theta_2 = 180^\circ - \cos^{-1} \left(\frac{a_1^2 + a_2^2 - d^2}{2a_1a_2} \right) \quad 2.22$$

Persamaan solusi untuk θ_2 telah diperoleh pada persamaan 2.22, persamaan ini dapat digunakan untuk menentukan persamaan solusi untuk θ_1 .

Sama seperti untuk mencari persamaan solusi θ_2 , segitiga merah pada Gambar 2.13 dengan menggunakan hukum tangen maka diperoleh :

$$\tan^{-1} \left(\frac{y}{x} \right) = \theta_1 + \beta \quad 2.23$$



Gambar 2.13 Manipulator Tipe RR Dengan Ilustrasi Segitiga Pendekatan.

Selain pada segitiga merah, pendekatan segitiga lain dapat digunakan untuk menentukan nilai β , sebuah segitiga siku-siku dengan sisi miring “ d ”, sisi alas “ $a_1 + a_2 \cos \theta_2$ ”, dan sisi tinggi “ $a_2 \sin \theta_2$ ”. Dengan menggunakan hukum tangen diperoleh persamaan :

$$\tan^{-1} \frac{a_2 \sin \theta_2}{a_1 + a_2 \cos \theta_2} = \beta \quad 2.24$$

Dengan mensubtitusikan persamaan 2.24 ke persamaan 2.23 diperoleh persamaan 2.25 berikut :

$$\theta_1 = \tan^{-1} \left(\frac{y}{x} \right) - \tan^{-1} \left(\frac{a_2 \sin \theta_2}{a_1 + a_2 \cos \theta_2} \right) \quad 2.25$$

Persamaan 2.25 dan 2.22 merupakan solusi kinematik mundur untuk manipulator pada Gambar 2.12. nilai θ_1 dan θ_2 dapat diperoleh dengan memasukkan nilai a_1, a_2, x dan y yang telah diketahui pada persamaan 2.25 dan 2.22.

7. Newton Raphson root finding inverse kinematic.

Selain iterasi dengan menguji semua kombinasi sudut, iterasi *newton raphson* juga dapat digunakan untuk mencari solusi dari *inverse kinematic*. Persamaan iterasi *newton raphson* dapat diamati pada persamaan 2.26 [16].

$$x_{n+1} = x_n - \frac{f(x)}{f'(x)} \quad 2.26$$

Dalam kasus untuk menentukan kinematika mundur, maka persamaan 2.26 berubah menjadi persamaan 2.27

$$\theta_{n+1} = \theta_n - \frac{f(\theta_n)}{J(\theta_n)}$$

$$\theta_{n+1} = \theta_n - f(\theta_n)J^{-1}(\theta_n) \quad 2.27$$

2.3 Pengolahan Citra

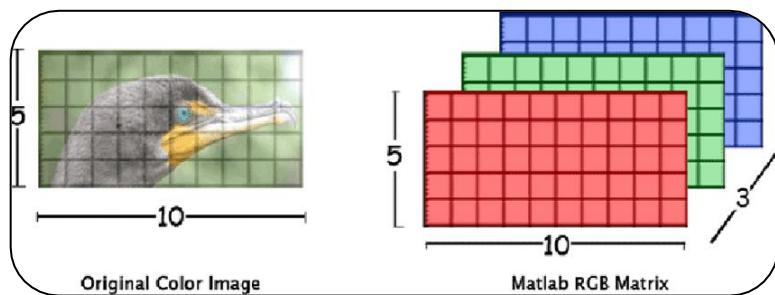
Citra atau *image* adalah angka, dari segi estetika, citra atau gambar adalah kumpulan warna yang bisa terlihat, memiliki pola, berbentuk abstrak dan lain sebagainya. Citra dapat berupa foto, penampang lintang (*cross section*) dari suatu benda, gambar wajah, hasil tomografi otak dan lain sebagainya. Citra didefinisikan sebagai suatu fungsi dua dimensi $f(x,y)$, dimana nilai atau amplitudo dari f pada koordinat spasial (x,y) menyatakan intensitas (kecerahan) citra pada titik tersebut. Jika x, y , dan nilai intensitas f terbatas sifatnya, citra tersebut dapat disebut sebagai citra digital[19] [20].

Citra digital merupakan citra $f(x,y)$ yang telah dilakukan digitalisasi baik koordinat area maupun brightness level. Nilai f di koordinat (x,y) menunjukkan

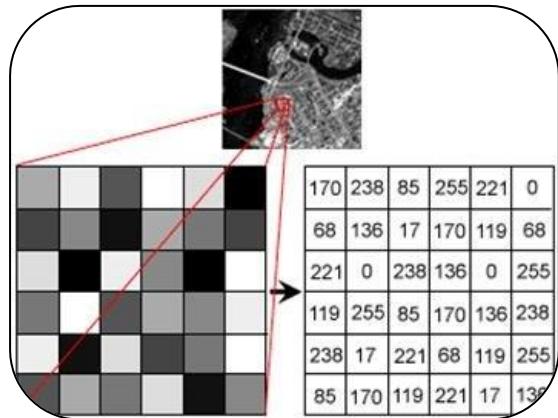
brightness atau *grayness* level dari citra pada titik tersebut. Dengan kata lain sederhananya bahwa citra digital adalah citra yang telah disimpan atau dikonversi ke dalam format digital. Pada penelitian ini citra digital dari objek didapatkan dari sensor kamera berupa webcam. Untuk memperoleh informasi dari sebuah citra, citra tersebut harus diolah terlebih dahulu berikut beberapa jenis pengolahan pada sebuah citra.

1. Citra keabu-abuan

Seperti yang telah dijelaskan pada sub-bab 2.4, bahwa citra digital merupakan sebuah matriks yang masing-masing elemennya mengandung nilai level kecerahan. Sebagai contoh untuk citra bertipe RGB (*Red, Green, Blue*) maka nilai dari elemen citra ini merupakan kombinasi dari level kecerahan dari ketiga warna tersebut, biasanya 0-255, bentuk matriks citra RGB dapat diamati pada Gambar 2.14. Gambar 2.14 menunjukkan bahwa matriks citra RGB mempunyai 3 dimensi. Jenis citra lain yaitu citra keabu-abuan (*Grayscale*), matriks citra jenis ini hanya memiliki 2 dimensi saja[19], bentuk matriks citra keabu-abuan dapat diamati pada Gambar 2.15. Dalam pengolahan citra sering kali hanya dibutuhkan beberapa properti saja dari citra agar dapat diambil informasi dari citra tersebut, sehingga citra RGB diubah terlebih dahulu ke citra keabu-abuan sebelum diolah.



Gambar 2.14 Matriks Citra RGB (www.researchgate.net).



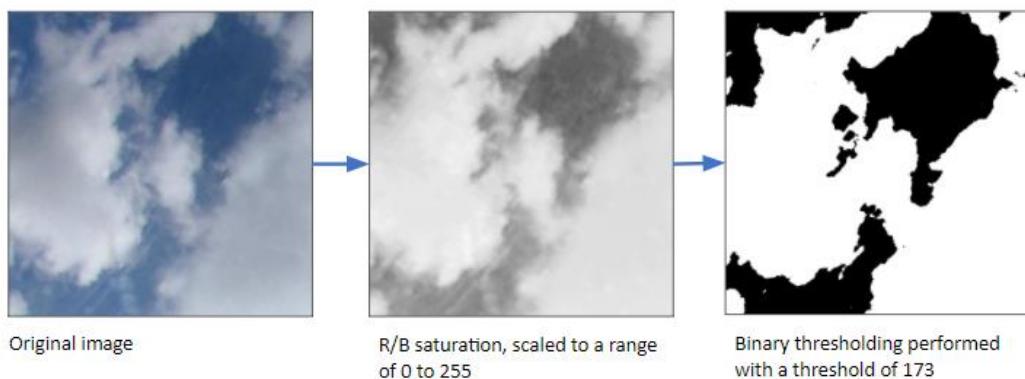
Gambar 2.15 Matriks Citra Keabu-abuan (csenotes.github.io).

Dalam mengubah citra RGB ke *grayscale* sebuah persamaan dapat digunakan untuk mengubah nilai elemen matriks RGB menjadi *grayscale*. Persamaan yang umum digunakan untuk mengubah citra RGB menjadi *grayscale* sebagai berikut

$$\text{Grayscale value} = (0.30 \times \text{Red Value}) + (0.59 \times \text{Green Value}) + (0.11 \times \text{Blue Value}) \quad 2.26$$

2. Thresholding.

Thresholding pada pengolahan citra merupakan proses seleksi pada nilai elemen matriks suatu citra yang biasanya bertujuan untuk segmentasi pada suatu citra, pada proses *thresholding* ditentukan terlebih dahulu nilai *threshold* nya yang akan dijadikan sebagai nilai batas seleksinya. Apabila nilai salah satu elemen pada matriks citra lebih besar dari nilai *threshold* maka nilai tersebut akan diubah menjadi nilai dengan intensitas tertinggi (1 untuk citra biner), sebaliknya apabila nilai elemen matriks lebih kecil dari nilai *threshold* maka nilai akan diubah menjadi nilai dengan intensitas terendah (0 untuk citra biner). Keluaran dari proses *thresholding* merupakan matriks citra yang hanya berisikan nilai 0 dan 1 (citra biner) dan masukannya berupa citra *grayscale*. Gambar 2.16 menunjukkan proses *thresholding* suatu citra.



Gambar 2.16 Proses *Thresholding* Suatu Citra (sites.wustl.edu).

3. Konvolusi citra digital

Konvolusi adalah operator matematika yang penting untuk banyak operator dalam image processing. Konvolusi menyediakan cara untuk menggabungkan dua array, biasanya untuk ukuran array yang berbeda, tetapi untuk dimensi array yang sama, menghasilkan array ketiga yang mempunyai dimensi yang sama. Konvolusi dapat digunakan dalam image processing untuk menerapkan operator yang mempunyai nilai output dari piksel yang berasal dari kombinasi linear nilai input piksel tertentu. Konvolusi memiliki dua buah fungsi $f(x)$ dan $g(x)$ yang didefinisikan sebagai berikut :

$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(a) \cdot g(x - a) da \quad 2.27$$

yang dalam hal ini, tanda (*) menyatakan operator konvolusi dan peubah (variable) “ a ” adalah peubah bantu. Untuk pengolahan citra, operasi yang dilakukan adalah diskrit karena nilai koordinat piksel merupakan nilai yang diskrit. Selanjutnya filter atau *mask* yang digunakan pada pengolahan citra biasanya berukuran terbatas, dalam artian bobot atau pengaruh dari titik-titik yang cukup jauh sudah tidak signifikan, sehingga dapat diabaikan (dianggap nol). Bentuk diskrit dari operasi konvolusi satu dimensi pada pengolahan citra adalah:

$$h(x) = f(x) * g(x) = \sum_{a=-\infty}^{\infty} f(a) \cdot g(x - a) \quad 2.28$$

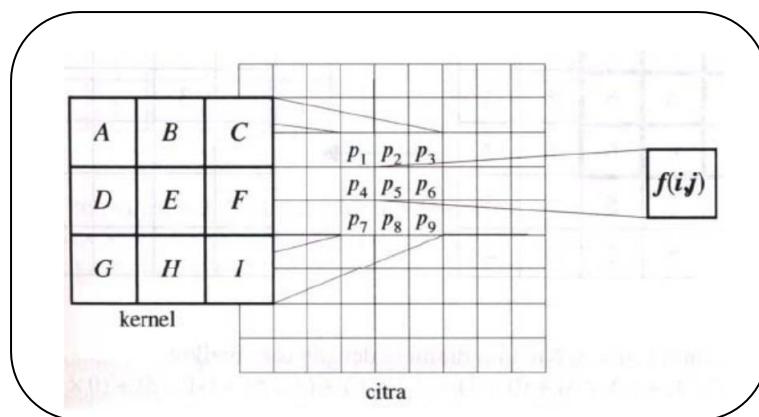
Untuk fungsi dengan dua dimensi, operasi konvolusi didefinisikan sebagai berikut:
Untuk fungsi integral:

$$h(x, y) = f(x, y) * g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(a, b) \cdot g(x - a, y - b) da db \quad 2.29$$

Untuk fungsi diskrit:

$$h(x, y) = f(x, y) * g(x, y) = \sum_{a=-\infty}^{\infty} \sum_{b=-\infty}^{\infty} f(a, b) \cdot g(x - a, y - b) \quad 2.30$$

Fungsi penapis $g(x,y)$ disebut juga konvolusi filter, konvolusi *mask*, konvolusi *kernel*, atau *template*. Dalam bentuk diskrit konvolusi *kernel* dinyatakan dalam bentuk matriks (umumnya matriks 3x3). Ukuran matriks ini biasanya lebih kecil dari ukuran citra. Setiap elemen matriks disebut koefisien konvolusi. Ilustrasi konvolusi ditunjukkan pada Gambar 2.17.



Gambar 2.17 Proses Konvolusi Suatu Citra dengan Sebuah Kernel[20].

Dengan menggunakan persamaan 2.30 maka masing-masing elemen pada matriks citra hasil konvolusi dihitung seperti berikut:

$$f(i,j) = Ap_1 + Bp_2 + Cp_3 + Dp_4 + Ep_5 + Fp_6 + Gp_7 + Hp_8 + Ip_9$$

Jika hasil konvolusi menghasilkan nilai piksel negatif, nilai tersebut dijadikan 0. Sebaliknya jika hasil konvolusi menghasilkan nilai piksel lebih besar dari nilai keabuan maksimum (255), nilai tersebut dijadikan ke nilai keabuan maksimum. Masalah timbul bila piksel yang dikonvolusi adalah piksel pinggir, karena beberapa koefisien konvolusi tidak dapat diposisikan pada piksel-piksel citra. Solusi untuk masalah ini adalah:

- a) piksel-piksel pinggir diabaikan, tidak dikonvolusi, jadi nilai piksel pinggir sama dengan nilai pada citra semula

- b) duplikasi elemen citra, misalnya elemen kolom pertama disalin ke kolom M+1 dst.

Solusi dengan pendekatan di atas mengasumsikan bagian pinggir citra lebarnya sangat kecil (hanya satu piksel) relatif dibandingkan dengan ukuran citra, sehingga piksel-piksel pinggir tidak memperlihatkan efek yang kasat mata.

4. Deteksi pinggir.

Deteksi pinggir merupakan suatu proses citra untuk menemukan pinggiran sebuah objek pada suatu citra. Proses ini dilakukan dengan mencari bagian elemen pada matriks citra yang mengalami perubahan nilai kecerahan secara mendadak. Untuk lebih lanjut terdapat dua teknik deteksi pinggir yang cukup terkenal yaitu deteksi pinggir sobel dan canny.

- a. Sobel *Edge Detection*.

Teknik deteksi pinggir sobel merupakan proses konvolusi dua buah kernel sobel pada sebuah citra, kernel sobel dapat diamati pada persamaan 2.31 dan 2.32.

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad 2.31$$

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad 2.32$$

Persamaan 2.31 merupakan kernel sobel untuk arah vertikal dan persamaan 2.32 untuk arah horizontal, dalam proses deteksi pinggir sobel kedua kernel ini dikonvolusikan dengan citra yang ingin di proses, hasil konvolusi citra dengan kernel vertikal dan hasil konvolusi citra dengan kernel horizontal dihitung magnitudo serta sudutnya lalu mengasilkan citra yang sudah terdeteksi pinggiran objeknya. Citra yang akan diproses dengan deteksi pinggir sobel harus citra bertipe keabuan, berikut persamaan dalam menghitung deteksi pinggir sobel :

$$S_x = G_x * P \quad 2.33$$

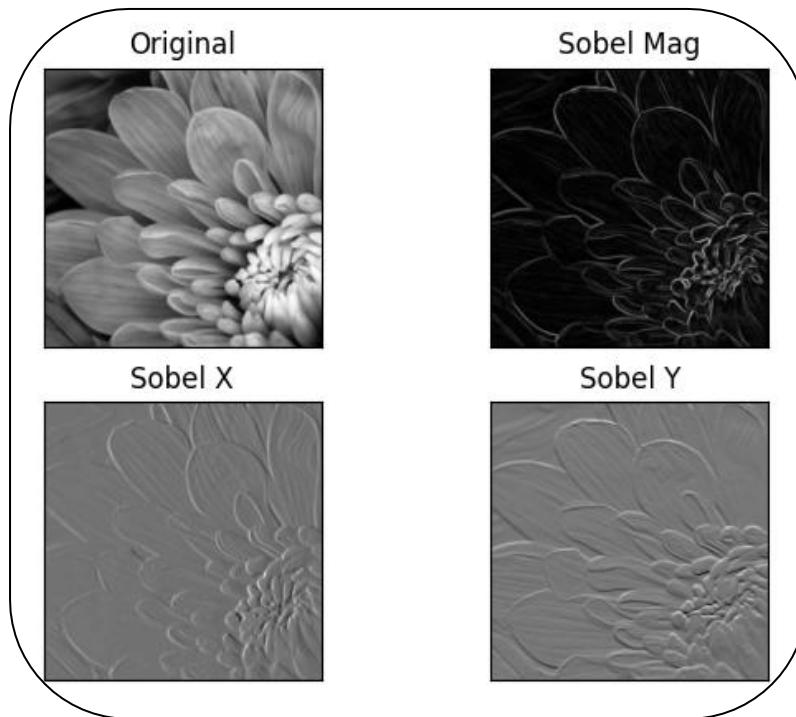
$$S_y = G_y * P \quad 2.34$$

$$\text{Mag} = \sqrt{G_x^2 + G_y^2} \quad 2.35$$

$$\theta = \text{atan2}(G_x, G_y) \quad 2.36$$

Dimana : P = citra yang diproses
 S_x = citra gradien X
 S_y = citra gradien Y
 Mag = nilai magnitude citra S_x & S_y
 θ = nilai sudut citra S_x & S_y

Ilustrasi pengolahan citra sobel dapat diamati pada Gambar 2.18



Gambar 2.18 Citra Hasil Deteksi Pinggir Sobel.

b. *Canny Edge Detection.*

Deteksi pinggir Canny merupakan deteksi pinggir lanjutan yang menyempurnakan deteksi pinggir Sobel. Pada Gambar 2.18, Citra Magnitud, bagian pinggir dari objek masih belum terlalu jelas atau belum membentuk sebuah garis pinggir. Langkah dalam pengolahan deteksi pinggir Canny terdapat pada list berikut:

- a) Citra yang akan diolah diterapkan dengan Gaussian filter untuk menghilangkan *noise*.
- b) Citra hasil Gaussian filter lalu ditentukan nilai gradien dan arahnya.

- c) Citra dengan nilai gradien lalu diterapkan dua kali *thresholding* dengan nilai *thresholding* tinggi dan rendah.
- d) Penyambungan piksel, penyambungan dilakukan pada piksel yang saling berdekatan dan memiliki intensitas yang sama, penyambungan dimulai dari piksel citra dengan *thresholding* tinggi lalu dilanjutkan dengan piksel citra dengan *thresholding* rendah hingga menghasilkan sebuah garis

Dari keempat langkah dalam pengolahan deteksi pinggir Canny, dapat dilihat bahwa dua langkah pertama merupakan proses pada deteksi pinggir Sobel, sehingga masukan dari proses Canny merupakan keluaran dari proses Sobel. Berikut ilustrasi pengolahan deteksi pinggir Canny dapat diamati pada Gambar 2.19.



Gambar 2.19 Citra Hasil Deteksi Pinggir Canny

2.4 Perangkat Keras dan Lunak yang Digunakan Pada Penelitian.

Berikut informasi dan teori beberapa perangkat keras dan lunak yang digunakan dalam penelitian ini.

5. Motor *stepper*.

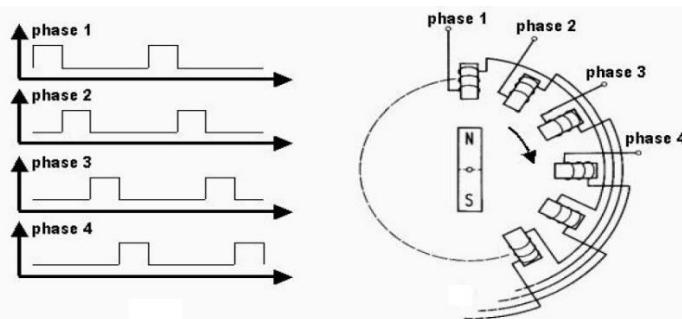
Motor *stepper* merupakan salah satu jenis dari motor dc, motor ini sering digunakan sebagai aktuator yang harus memiliki gerak yang presisi. Motor *stepper* yang digunakan pada penelitian ini merupakan motor *stepper* jenis magnet permanen, rotor motor *stepper* merupakan magnet permanen dan stator-Nya kumparan tembaga, dengan desain seperti ini motor tidak memerlukan sikat lagi. Untuk memutar motor *stepper* pada kumparan stator-Nya harus diberikan sinyal listrik berupa tegangan berbentuk pulsa. Pemberian pulsa harus sesuai dengan tipe

motornya, pada penelitian ini digunakan motor *stepper* Nema 17. Gambar motor *stepper* dapat dilihat pada Gambar 2.12.

Motor *stepper* Nema 17 memiliki 4 kutub, untuk dapat memutar motor *stepper* tegangan diberikan secara bergantian pada kutub ini, urutan pemberian tegangan pada ke empat kutub akan mempengaruhi arah putaran motor *stepper*. Pemberian sinyal pada motor *stepper* dapat diamati pada Gambar 2.20.



Gambar 2.20 Motor Stepper Nema 17 (www.google.com).



Gambar 2.21 Pemberian Sinyal Pada Motor Stepper (www.google.com).

Gambar 2.21 menunjukkan pemberian sinyal ke motor *stepper*, pada gambar untuk memutar motor searah jarum jam sinyal diberikan secara urut mulai dari fasa 1, sinyal pada fasa 1 lalu dimatikan dan diberikan sinyal ke fasa 2 dan seterusnya. Sinyal seperti ini dapat dibuat dengan bantuan IC seperti NE555, pada penelitian ini digunakan mikrokontroler Arduino untuk menghasilkan pulsa, sinyal Arduino juga diperkuat dengan IC LM781 agar mampu menghasilkan arus listrik yang cukup. Kecepatan putar motor *stepper* dapat diatur dengan mengatur lamanya jeda pemberian pulsa antar fasa[21].

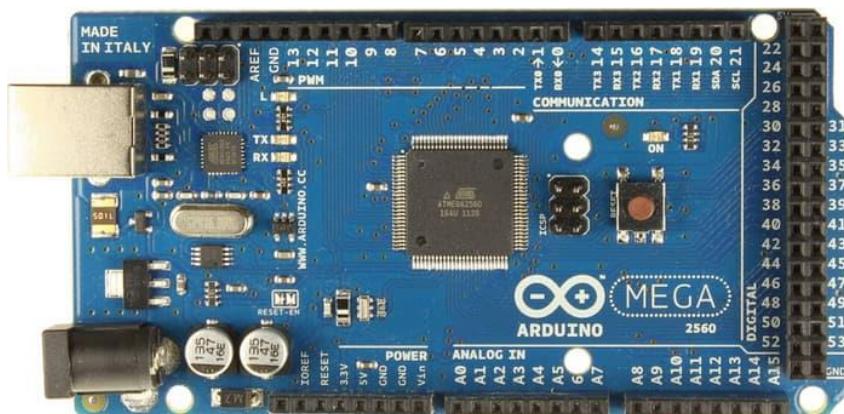
6. Mikrokontroler Arduino Mega 2560.

Arduino Mega 2560 merupakan salah satu jenis mikrokontroler keluaran Arduino, pada penelitian ini mikrokontroler digunakan untuk memberikan sinyal pada motor *stepper*, Arduino mega memiliki spesifikasi seperti pada Tabel 2.2.

Tabel 2.2 Spesifikasi Arduino Mega 2560

Komponen	spesifikasi
Chip mikrokontroler	Atmega 2560
Tegangan operasi	5 V
Tegangan input (Rekomendasi)	7V – 12 V
Tegangan input (Limit)	6V – 20 V
Digital I/O pin	54 pin, 6 PWM output pin
Analog Input pin	16 pin
Arus DC per pin I/O	20 mA
Arus DC pin 3.3 V	50 mA
Memory Flash	256 KB, 8 KB untuk bootloader
SRAM	8 KB
EEPROM	4 KB
Clock speed	16 Mhz
Dimensi	101.5 mm x 53.4 mm
Berat	37 g

Gambar Arduino Mega 2560 dapat dilihat pada Gambar 2.22



Gambar 2.22 Arduino Mega 2560 (www.google.com).

Arduino adalah *board* berbasis mikrokontroler atau papan rangkaian elektronik *open source* yang di dalamnya terdapat komponen utama yaitu sebuah

chip mikrokontroler dengan jenis AVR dari perusahaan Atmel. Mikrokontroler itu sendiri adalah chip atau IC (*integrated circuit*) yang bisa diprogram menggunakan komputer. Tujuan menanamkan program pada mikrokontroler adalah agar rangkaian elektronik dapat membaca masukan, memproses masukan tersebut dan kemudian menghasilkan keluaran sesuai yang diinginkan. Jadi mikrokontroler bertugas sebagai otak yang mengendalikan proses masukan, dan keluaran sebuah rangkaian elektronik.

Pada Gambar 2.14 merupakan jenis Arduino Mega tipe 2560, Arduino Mega 2560 adalah papan pengembangan mikrokontroler yang berbasis Arduino dengan menggunakan chip ATmega2560. *Board* ini memiliki pin I/O yang cukup banyak, sejumlah 54 buah digital I/O pin (15 pin diantara-Nya adalah PWM), 16 pin analog *input*, 4 pin UART (*serial port hardware*). Arduino Mega 2560 dilengkapi dengan sebuah osilator 16 Mhz, sebuah port USB, power jack DC, ICSP header, dan tombol reset. *Board* ini sudah sangat lengkap, sudah memiliki segala sesuatu yang dibutuhkan untuk sebuah mikrokontroler[22].

7. Catu daya (*Power Supply*).

Power Supply atau dalam bahasa Indonesia disebut dengan catu daya adalah suatu alat listrik yang dapat menyediakan energi listrik untuk perangkat listrik ataupun elektronika lainnya. Pada dasarnya *power supply* atau catu daya ini memerlukan sumber energi listrik yang kemudian mengubahnya menjadi energi listrik yang dibutuhkan oleh perangkat elektronika lainnya[23]. Oleh karena itu, *power supply* kadang-kadang disebut juga dengan istilah *electric power converter*.

Pada umumnya *power supply* dapat diklasifikasikan menjadi 3 kelompok besar, yakni berdasarkan Fungsinya, berdasarkan bentuk mekanikkannya dan juga berdasarkan metode konversinya. Berikut ini merupakan penjelasan singkat mengenai ketiga kelompok tersebut.

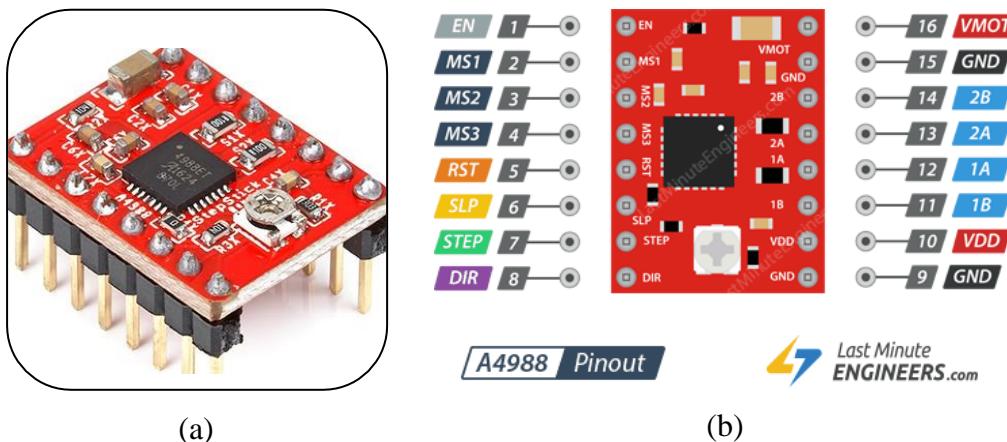
Berdasarkan fungsinya, *power supply* dapat dibedakan menjadi *regulated power supply*, *unregulated power supply* dan *adjustable power supply*.

- a) *Regulated power supply* adalah *power supply* yang dapat menjaga kestabilan tegangan dan arus listrik meskipun terdapat perubahan atau variasi pada beban atau sumber listrik (tegangan dan arus masukan).

- b) *Unregulated power supply* adalah *power supply* tegangan ataupun arus listriknya dapat berubah ketika beban berubah atau sumber listriknya mengalami perubahan.
- c) *Adjustable power supply* adalah *power supply* yang tegangan atau arusnya dapat diatur sesuai kebutuhan dengan menggunakan knob mekanik. Terdapat 2 jenis *adjustable power supply* yaitu *regulated adjustable power supply* dan *unregulated adjustable power supply*.

8. *Stepper driver A4988.*

Modul A4988 merupakan modul elektronik yang berfungsi untuk mengatur putaran dari motor *stepper*. Pada modul ini terdapat sirkuit terintegrasi bernama "A4988 DMOS Microstepping", modul A4988 diproduksi oleh perusahaan "Allegro MicroSystem", pada sirkuit terintegrasi A4988 terdapat rangkaian jembatan H serta rangkaian proteksi arus, rangkaian jembatan H inilah yang menerjemahkan perintah dari mikrokontroler agar motor *stepper* dapat berputar, adapun bentuk dan kegunaan pin pada modul a4988 dapat diamati pada Gambar 2.23



Gambar 2.23 (a) Modul A4988 (electronicwing.com), (b) Pin Keluaran Modul A4988 (Pinterest.com).

Gambar 2.23 (b) menunjukkan nama pin pada modul A4988, kegunaan pin pada modul A4988 dapat diamati pada Tabel 2.3

Tabel 2.3 Nama dan fungsi dari pin modul A4988

No	Nama Pin	Sinyal Pin	Keterangan
1	EN (Enable)	Digital (0V-5V)	Berfungsi untuk menghidup atau mematikan sirkuit pada modul
2	MS1 (MicroStepping 1)	Digital (0V-5V)	Mengatur jenis step pada motor
3	MS2 (MicroStepping 2)	Digital (0V-5V)	Mengatur jenis step pada motor
4	MS3 (MicroStepping 3)	Digital (0V-5V)	Mengatur jenis step pada motor
5	RST (Reset)	Digital (0V-5V)	Mereset Modul
6	SLP (Sleep)	Digital (0V-5V)	Berfungsi untuk menghidup atau mematikan modul dengan memutus aliran listrik dari supply ke motor
7	STEP	Digital (0V-5V)	Mengatur detakan pada rangkaian H
8	DIR (Direction)	Digital (0V-5V)	Mengatur sakelar yang bekerja pada rangkaian H
9	GND (Ground)	Digital (0V-5V)	Pin ground dari mikrokontoler
10	VDD (Digital Voltage)	5v	Pin digital 5V dari mikrokontroler
11	1B	0V – Vmot	Pin ke motor
12	1A	0V – Vmot	Pin ke motor
13	2B	0V – Vmot	Pin ke motor
14	2A	0V – Vmot	Pin ke motor
15	GND (Ground)	Ground Supply	Ground dari supply
16	VMot (Tegangan Motor)	0V – 32V	Tegangan dari supply

9. Bahasa pemrograman python.

Python adalah bahasa pemrograman tingkat tinggi yang banyak digunakan untuk pemrograman tujuan umum, dibuat oleh Guido van Rossum dan pertama kali dirilis pada tahun 1991. Python memiliki fitur sistem tipe dinamis dan manajemen memori otomatis dan mendukung beberapa paradigma pemrograman, termasuk pemrograman berorientasi objek, imperatif, fungsional, dan gaya prosedural. Ini memiliki perpustakaan standar yang besar dan komprehensif[24]. Logo bahasa pemrograman Python dapat dilihat pada Gambar 2.16.



Gambar 2.24 Logo Bahasa Pemrograman Python.

10. OpenCV.

Computer Vision adalah kemampuan mesin/komputer dalam melihat hingga mampu mengekstrak informasi dari sebuah gambar. Salah satu bidang yang berkaitan dengan *Computer Vision* adalah pengolahan citra atau biasa disebut *image processing*.

OpenCV adalah sebuah *library* (perpustakaan) yang digunakan untuk mengolah gambar dan video hingga kita mampu mengekstrak informasi didalamnya. OpenCV dapat berjalan di berbagai bahasa pemrograman, seperti C, C++, Java, Python, dan juga mendukung di berbagai platform seperti Windows, Linux, Mac OS, iOS dan Android[25]

BAB III

METODE PENELITIAN

3.1 Metode Penelitian

Penelitian ini bertujuan untuk merancang sebuah robot lengan pemindah barang menggunakan algoritma kinematika mundur dan membuat sebuah perangkat lunak pada PC sebagai klien agar robot mampu memindahkan benda berdasarkan ukuran. Penelitian ini dilakukan melalui tiga tahapan yaitu perancangan sistem, pengujian sistem dan evaluasi sistem.

3.2 Waktu dan Tempat

Penelitian ini dilaksanakan pada bulan September 2020 sampai dengan selesai di Laboratorium Teknik Elektro Universitas Bengkulu. Penelitian akan dimulai dari pengumpulan bahan-bahan berupa teori-teori dan metode yang mendukung penelitian, dilanjutkan dengan pengumpulan komponen-komponen berupa Arduino, sensor dan alat-alat pendukung perancangan alat, kemudian realisasi rancangan berupa pembuatan prototipe “ robot lengan ”, dan berbagai pengujian sistem, serta pembuatan laporan yang membahas berbagai analisa dari pengujian sistem.

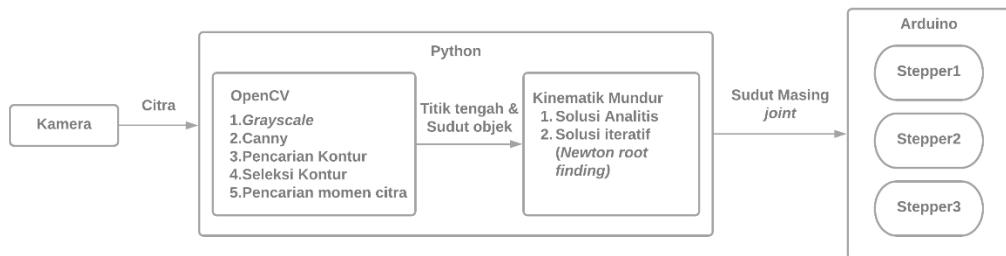
3.3 Alat dan Bahan

Adapun alat dan bahan yang digunakan dalam pembuatan robot lengan dan perancangan perangkat lunaknya sebagai berikut :

1. Komputer dilengkapi dengan
 - a. Bahasa pemrograman Python
 - b. Arduino IDE
2. Motor *stepper*
3. Motor servo
4. Akrilik
5. Catu daya
6. Kamera smartphone
7. Arduino Mega
8. Kabel USB
9. DLL

3.4 Diagram Blok Sistem

Pada diagram blok akan dijelaskan alur kinerja sistem, sistem terdiri dari kamera, komputer, mikrokontroler Arduino dan robot lengan, diagram blok dapat diamati pada Gambar 3.1.



Gambar 3.1 Diagram Blok.

Pada Gambar 3.1 masukan dari sistem merupakan citra dari kamera, kamera yang digunakan merupakan kamera pada telepon pintar. Kamera ini akan mengambil citra di daerah khusus (wilayah kerja robot lengan) yang telah diletakan objek yang akan berinteraksi dengan robot lengan. Citra yang diambil kamera telepon pintar secara langsung dikirim ke komputer. Pada komputer akan dibuat sebuah aplikasi khusus menggunakan bahasa pemrograman Python, aplikasi ini akan mengolah citra dari kamera dan menentukan koordinat dari objek yang akan dipindahkan oleh robot lengan. Setelah koordinat diperoleh maka pada aplikasi akan langsung dihitung kinematika mundurnya dan akan diperoleh besar sudut atau pergeseran masing-masing *joint* dan *base*. Data ini lalu dikirimkan ke Arduino melalui koneksi serial, lalu Arduino akan menerjemahkan data dan menggerakkan masing-masing motor pada robot lengan.

3.5 Perancangan Sistem

Perancangan sistem akan dibagi menjadi dua tahap, yaitu tahap pembuatan perangkat keras (*hardware*) dan tahap perancangan perangkat lunak (*software*).

3.5.1 Perancangan perangkat keras

Perancangan perangkat keras dibagi lagi menjadi beberapa bagian, mulai dari perancangan peletakan kamera, perancangan matriks transformasi homogen, serta perancangan robot lengan.

a. Perancangan robot lengan

Robot lengan pada penelitian ini dibuat dari akrilik dan *joint* serta *base*-nya digerakkan menggunakan motor *stepper* dan motor *servo*. Robot lengan yang dibuat memiliki konstruksi yang sama dengan robot SCARA (Selective Compliance Assembly Robot Arm). Ilustrasi rancangan robot lengan dapat dilihat pada Gambar 3.2

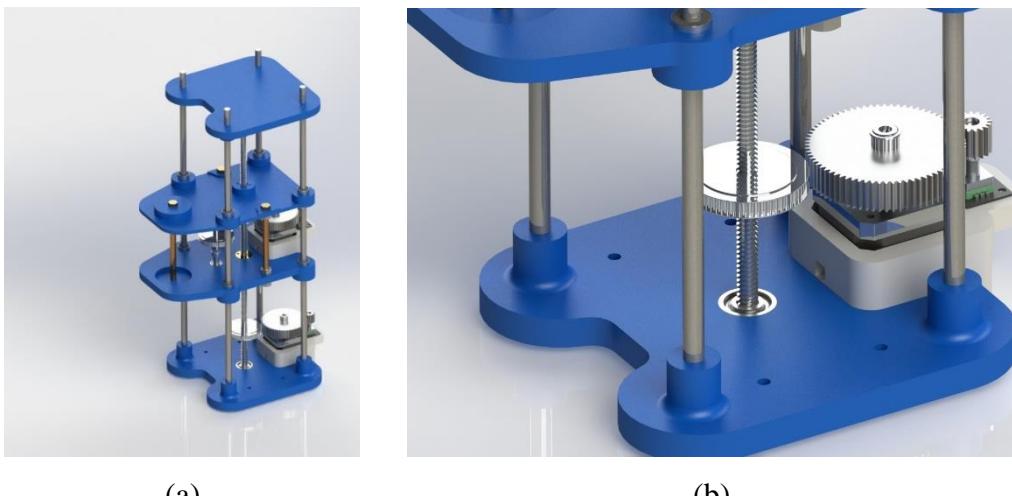


Gambar 3.2 Rancangan Robot Lengan

Pada Gambar 3.2 rancangan robot dibuat menggunakan perangkat lunak Solidworks, robot tersusun dari beberapa perangkat jadi dan kerangka buatan sendiri dari akrilik. Sebagian besar bagian robot dibuat dari akrilik seperti *base* robot, dudukan motor *stepper* dan lengan robot, bagian lain merupakan perangkat yang telah jadi seperti batang besi, *bearing*, sekrup dan mur, *gear*, *spacer*, dan *lead screw*. Robot dibagi menjadi tiga bagian utama dibagi berdasarkan *joint*-nya, yaitu bagian *slider*, lengan pertama dan lengan kedua.

1. Bagian *slider*

Pada bagian ini terdiri dari *main base* dan *joint* pertama dari robot lengan. gambar bagian *slider* dapat dilihat pada Gambar 3.3



Gambar 3.3 a. Bagian *Slider*, b. *Base Frame*

Pada Gambar 3.3a terdapat rancangan dari bagian *slider*, terdapat dua bagian utama yaitu bagian *base* seperti pada Gambar 3.3b dan bagian *joint* pertama yang menghubungkan *base* dengan *link* pertama seperti pada Gambar 3.4. Pada Gambar 3.3b bagian *base* terdiri dari sembilan komponen yaitu; *main base*, *ash pole (linear shaft)*, *bearing*, *spacer*, *stepper motor*, *stepper base*, *timing belt*, *timing gear pulley*, dan *lead screw*. Masing-masing komponen memiliki fungsi tersendiri.

Komponen pertama pada *base* sendiri merupakan *main base*, *main base* dibuat dari bahan kayu. *Main base* diberi lubang, lubang ini digunakan sebagai dudukan komponen lain dan untuk pemasangan sekrup, adapun komponen yang di dudukan pada *main base* adalah satu buah *bearing*, empat *linear shaft* dan empat *spacer*. *Bearing* yang digunakan memiliki ukuran diameter luar 16mm dan diameter dalam 8mm, *bearing* ini berfungsi sebagai dudukan dari *lead screw*, *lead screw* dipasangkan pada bagian dalam *bearing*, saat *lead screw* diputar oleh motor *stepper* perputarannya akan menjadi lebih halus akibat penggunaan *bearing*.

Komponen lain yang dipasang pada *main base* adalah *linear shaft*, *linear shaft* dipasang pada keempat sudut *main base*, fungsinya sebagai fondasi dari *base* dan sebagai *shaft* saat bagian *joint* pertama bergerak naik atau turun. Komponen lain yang dipasang adalah dudukan motor *stepper*, dudukan ini dibuat dari akrilik dan digabung dengan *spacer*. Motor *stepper* pada *base* tidak untuk menggerakkan *base* tetapi untuk memutar *lead screw*, pada *lead screw* dan *shaft* pada motor *stepper* dipasangkan *timing gear pulley* lalu dihubungkan dengan *belt*. Tujuan *lead*

screw diputar agar dapat menggerakkan *joint* pertama naik dan turun. Dari desain yang telah dibuat terdapat perhitungan, kecepatan motor *stepper* yang berputar dalam step/detik harus diubah ke satuan linier agar diketahui berapa banyak gerak linier yang ditempuh per satuan derajat motor berputar, beberapa yang mempengaruhi perhitungan ini adalah ; jenis motor, perbandingan *pulley*, jenis *MicroStepping* pada motor, dan panjang ulir dari *lead screw*. Berikut perhitungannya :

Jenis Motor : Nema 17 (200 step/putaran)

Perbandingan *pulley* : 1 : 3

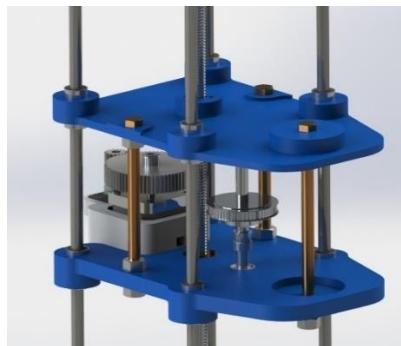
MicroStepping : *half-step*

Panjang Ulir : 2 mm / putaran

Besarnya konstanta geser (mm/step) :

$$= \frac{200 \text{ step/putaran} \times 3 \times 2}{2 \times 2 \text{ mm/putaran}} = \frac{300 \text{ step}}{\text{mm}} = 0.0033 \text{ mm/step}$$

Selain *base*, *joint* pertama juga diletakan pada bagian *slider*, *joint* pertama dapat diamati pada Gambar 3.4



Gambar 3.4 Bagian Joint Pertama

Bagian *joint* pertama terdiri dari motor *stepper*, *lead screw*, *lead screw head*, *linear bearing*, dan *linear shaft*. *Lead screw* dan *linear shaft* pada bagian ini, merupakan perangkat yang sama seperti pada bagian *base*. Kerangka dari *joint* pertama dibuat dari akrilik dapat dilihat pada Gambar 3.4 dengan nama *joint 1 base*, pada bagian tengah *base joint* dipasang *lead screw head*, sehingga saat *lead screw* berputar, *lead screw head* akan bergerak naik atau turun dan menyebabkan *base joint* juga bergerak naik atau turun. *Linear shaft* dipasang pada ujung-ujung *base joint* agar pergerakan naik atau turun *base joint* menjadi lebih halus dan

menahannya agar tidak berputar saat *lead screw* berputar. Motor *stepper* berfungsi untuk memutar bagian lengan pertama.

2. Bagian lengan pertama (*joint* ke-2)

Bagian lengan pertama terdiri dari joint ke dua dan *link* kedua, bagian lengan pertama dapat dilihat pada Gambar 3.5.



Gambar 3.5 Bagian Lengan Pertama

Bagian lengan pertama pada Gambar 3.5 dimulai dari *rotation shaft* hingga akhir dari lengan pada gambar, bagian sebelumnya merupakan *joint pertama*. Pada pangkal lengan pertama dipasang *rotation shaft* dan *gear pulley* lalu dihubungkan dengan *gear pulley* motor *stepper* dengan *belt*. Saat motor *stepper* berputar, lengan pertama juga akan berputar dengan *rotation shaft* sebagai pusat perputarannya. Pada lengan pertama terdapat satu motor *stepper*, motor ini berfungsi untuk menggerakan bagian selanjutnya yaitu lengan ke dua. Sama seperti bagian *joint* ke-1 terdapat perhitungan pada bagian ini, walaupun *joint* ke-2 bergerak dengan cara berputar sama seperti motor *stepper*, tetapi terdapat perbedaan satuan, dimana motor *stepper* bergerak dalam satuan step sedangkan *joint* ke-2 dalam satuan derajat, serta digunakan juga *gear pulley* yang akan mengubah jumlah putaran pada *joint* ke-2. Berikut perhitungannya:

Rasio Gear : $1:3 + 1:3 \rightarrow 1:9$

MicroStepping : $1/8$ step

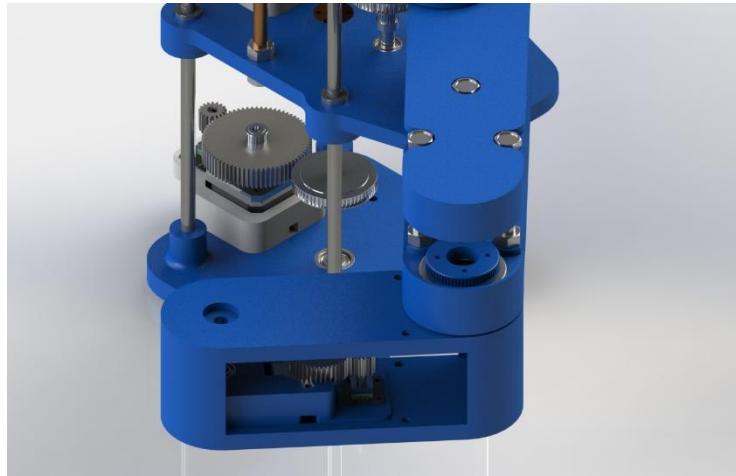
Jenis Motor : Nema 17 ($1.8^\circ/\text{step}$)

$$\text{Konstanta Putar } (\text{step}^\circ) = ((9 \times 8)/2)/1.8^\circ/\text{step} = 20 \text{ step}^\circ = 0.05^\circ/\text{step}$$

3. Bagian lengan kedua (*joint* ke-3 dan *end of effector*)

Bagian lengan kedua terdiri dari *joint* ke-3, *link* ke-3 dan *end of effector*.

Bagian lengan kedua dapat diamati pada Gambar 3.6

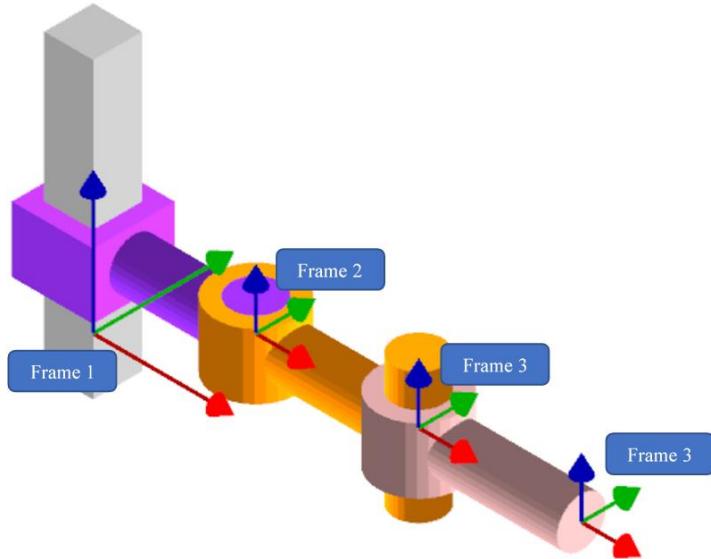


Gambar 3.6 Bagian Lengan Kedua dan *End Of Effector*.

Bagian lengan kedua terdiri dari *joint* ke-3, *joint* ini menghubungkan lengan kedua dan lengan pertama,pada lengan kedua juga terdapat *link* ke-3 menghubungkan *joint* ke-3 dan *end of effector*. Sama seperti rancangan sebelumnya, *joint* ke-3 pada lengan kedua merupakan *rotation shaft* yang dipasangkan dengan *gear pulley* dan diputar oleh motor *stepper* yang ada pada lengan pertama. Perhitungan yang berlaku untuk *joint* ke-3 ini sama seperti perhitungan pada *joint* ke-2 dikarenakan memiliki desain yang sama, baik jumlah rasio pada *gear pulley*, jenis *microstepping* dan jenis motor.

b. Perhitungan persamaan kinematik robot lengan.

Pada sub-bab ini akan dilakukan perhitungan persamaan kinematik maju dan mundur untuk manipulator yang telah dibuat, untuk kinematika maju dihitung dengan menggunakan matriks transformasi homogen, sedangkan kinematik mundurnya menggunakan teknik analisis (*closed form solution*). Manipulator robot lengan yang telah dirancang dapat diamati pada Gambar 3.7.



Gambar 3.7 Manipulator Lengan Robot

Dari Gambar manipulator 3.7 dapat dibuat matriks transformasi homogennya dengan peraturan matriks perputaran dan vektor perpindahan seperti berikut.

a. Matriks perputaran.

1. Matriks perputaran *frame 1* relatif terhadap *frame 0* (R_1^0).

Joint ke-1 merupakan *joint* bertipe geser (*prism*) sehingga matriks putarnya sebuah matriks identitas dan matriks orientasinya juga matriks identitas dikarenakan *frame 0* dan *frame 1* memiliki orientasi yang sama. Matriks perputaran *frame 1* relatif terhadap *frame 0* (R_1^0) dapat dilihat pada persamaan 3.1.

$$R_1^0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \circ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad 3.1$$

2. Matriks perputaran *frame 2* relatif terhadap *frame 1* (R_2^1).

Joint ke-2 merupakan *joint* bertipe putar, *joint* berputar dengan sumbu z sebagai porosnya sehingga matriks putarnya sama dengan persamaan 2.3. matriks orientasinya merupakan matriks identitas karena *frame 2* sudah memiliki orientasi yang sama dengan *frame 1*. Matriks perputaran *frame 2* relatif terhadap *frame 1* (R_2^1) dapat dilihat pada persamaan 3.2

$$R_2^1 = \begin{bmatrix} \cos \theta_1 & -\sin \theta_1 & 0 \\ \sin \theta_1 & \cos \theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \circ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \theta_1 & -\sin \theta_1 & 0 \\ \sin \theta_1 & \cos \theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad 3.2$$

θ_1 = besar sudut perputaran *joint* ke-2

3. Matriks perputaran *frame* 3 relatif terhadap *frame* 2 (R_3^2).
4. Pada matriks ini juga memiliki nilai yang sama dengan persamaan 3.2 diakrenakan *joint* bertipe putar dengan sumbu z sebagai porosnya, serta orientasi *frame* sudah sama. Matriks perputaran *frame* 3 relatif terhadap *frame* 2 (R_3^2) dapat dilihat pada persamaan 3.3

$$R_3^2 = \begin{bmatrix} \cos \theta_2 & -\sin \theta_2 & 0 \\ \sin \theta_2 & \cos \theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \circ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos \theta_2 & -\sin \theta_2 & 0 \\ \sin \theta_2 & \cos \theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad 3.3$$

θ_2 = besar sudut perputaran *joint* ke-3

b. Vektor perpindahan.

5. Vektor perpindahan *frame* 1 relatif terhadap *frame* 0 (D_1^0).

Elemen d_{11} pada vektor D_1^0 merupakan besar perpindahan sumbu x pada *frame* 1 terhadap pusat *frame* 0, dikarenakan *joint* pada *frame* 0 bertipe geser (*prism*) maka besar elemen d_{11} sama dengan panjang *link* 1 yaitu “89 mm”, dikarenakan *link* 1 yang searah dengan sumbu x.

Elemen d_{21} pada vektor D_1^0 merupakan besar perpindahan sumbu y pada *frame* 1 terhadap pusat *frame* 0, tidak terjadi perpindahan apa pun pada *frame* 1 di sumbu y sehingga nilai elemennya adalah nol

Elemen d_{31} pada vektor D_1^0 merupakan besar perpindahan sumbu z pada *frame* 1 terhadap pusat *frame* 0, pada rancangan robot elemen d_{31} tergantung dari pergeseran yang dilakukan oleh *joint* ke-1 sehingga nilainya “157 mm + J1”. 157 mm merupakan jarak antara *frame* 0 dan batas gerak dari *frame* 1 pada sumbu z, karena pada perancangan robot diletakan motor *stepper* di bawah *joint* ke-1 sehingga *frame* 1 pada *joint* tidak bisa saling bertindihan dengan *frame* 0 walaupun besar pergeseran (perpindahan) sama dengan nol, dan J1 merupakan variabel besarnya perpindahan yang dilakukan oleh *joint* ke-1. Vektor perpindahan *frame* 1 relatif terhadap *frame* 0 dapat dilihat pada persamaan 3.5

$$D_1^0 = \begin{bmatrix} 89 \\ 0 \\ 157 + J1 \end{bmatrix} \quad 3.4$$

6. Vektor perpindahan *frame* 2 relatif terhadap *frame* 1 (D_2^1).

Elemen d_{11} pada vektor D_2^1 bernilai “175 mm * cos θ_1 ”, 175 mm merupakan panjang *link* 2, elemen d_{21} bernilai “135mm * sin θ_1 ”, sedangkan elemen d_{31} bernilai -7 mm. Vektor perpindahan *frame* 2 relatif terhadap *frame* 1 (D_2^1) dapat dilihat pada persamaan 3.5.

$$D_2^1 = \begin{bmatrix} 175 \cos \theta_1 \\ 175 \sin \theta_1 \\ -70 \end{bmatrix} \quad 3.5$$

7. Vektor perpindahan *frame* 3 relatif terhadap *frame* 2 (D_3^2).

Elemen d_{11} pada vektor D_2^1 bernilai “135mm * cos θ_2 ”, 135 mm merupakan panjang *link* 3, elemen d_{21} bernilai “135mm * sin θ_2 ”, sedangkan elemen d_{31} bernilai -300. Vektor perpindahan *frame* 3 relatif terhadap *frame* 2 (D_3^2) dapat dilihat pada persamaan 3.6.

$$D_3^2 = \begin{bmatrix} 138 \cos \theta_2 \\ 138 \sin \theta_2 \\ -43 \end{bmatrix} \quad 3.6$$

c. Matriks transformasi homogen.

Setelah diperoleh matriks perputaran dan vektor perpindahan dapat dibuat matriks transformasi homogennya seperti berikut.

$$\begin{aligned} H_1^0 &= \begin{bmatrix} 1 & 0 & 0 & 50 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 157 + J1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ H_2^1 &= \begin{bmatrix} \cos \theta_1 & -\sin \theta_1 & 0 & 195 \cos \theta_1 \\ \sin \theta_1 & \cos \theta_1 & 0 & 195 \sin \theta_1 \\ 0 & 0 & 1 & -70 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ H_3^2 &= \begin{bmatrix} \cos \theta_2 & -\sin \theta_2 & 0 & 138 \cos \theta_2 \\ \sin \theta_2 & \cos \theta_2 & 0 & 138 \sin \theta_2 \\ 0 & 0 & 1 & -43 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ H_4^0 &= \begin{bmatrix} \cos(\theta_a) & -\sin(\theta_a) & 0 & \cos(\theta_a) + 130 \cos(\theta_b) + 195 \cos \theta_1 + 104 \\ \sin(\theta_a) & \cos(\theta_a) & 0 & \sin(\theta_a) + 130 \sin(\theta_b) + 195 \sin \theta_1 \\ 0 & 0 & 1 & -113 + 157 + J1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad 3.7 \end{aligned}$$

Dimana : $\theta_a = \theta_1 + \theta_2 + \theta_3$, $\theta_b = \theta_1 + \theta_2$

d. Persamaan kinematika mundur

Metode yang digunakan untuk kinematika mundur adalah metode analisis dan numerik, kedua metode memiliki persamaan yang berbeda dalam perhitungan untuk mencari solusi kinematiknya.

1. Persamaan kinematika mundur analisis

Robot lengan yang dibangun memiliki tiga derajat kebebasan dengan semua *joint* robot berjenis putar dengan sumbu z sebagai pusat rotasi sehingga persamaan kinematika mundur sama seperti persamaan 2.23

$$x' = x - \cos \theta_c \quad 3.8$$

$$y' = y - \sin \theta_c \quad 3.9$$

$$\theta_2 = 180^\circ - \cos^{-1} \left(\frac{a_1^2 + a_2^2 - d^2}{2a_1a_2} \right)$$

$$\theta_2 = 180^\circ - \cos^{-1} \left(\frac{38.025 + 16.900 - (x' - 104)^2 - (y')^2}{2 \times 195 \times 130} \right)$$

$$\theta_2 = 180^\circ - \cos^{-1} \left(\frac{38.025 + 16.900 - (x')^2 + 208x - 10.816 - (y')^2}{2 \times 195 \times 130} \right)$$

$$\theta_2 = 180^\circ - \cos^{-1} \left(\frac{-(x')^2 + 208x - (y')^2 + 44.109}{50.700} \right) \quad 3.10$$

Dan untuk nilai θ_1 sama dengan persamaan 2.25

$$\theta_1 = \tan^{-1} \left(\frac{y'}{x' - 104} \right) - \tan^{-1} \left(\frac{130 \sin \theta_2}{195 + 130 \cos \theta_2} \right) \quad 3.11$$

θ_c , x , dan y diperoleh dari properti kontur yang akan dituju oleh *end of effector* robot lengan. θ_c merupakan besar sudut yang dibentuk antara kontur dengan sumbu x robot, sedangkan x dan y merupakan titik tengah dari kontur. Ditentukan x' dan y' menggunakan persamaan 3.8 dan 3.9, x' dan y' lalu digunakan untuk menentukan θ_2 dan θ_1 , θ_3 dihitung dengan persamaan 3.12.

$$\theta_3 = \theta_c - \theta_1 - \theta_2 \quad 3.12$$

θ_1 , θ_2 dan θ_3 merupakan solusi dari kinematika mundur dari θ_c , x , dan y .

2. Persamaan kinematik numerik

Pencarian solusi kinematika mundur dengan metode numerik menggunakan persamaan *newton Raphson* seperti pada persamaan 2.27, untuk $f(x)$ merupakan

θ_c , x , dan y dikurang elemen (1,4), (2,4), dan (1,1) pada matrik H_4^0 di persamaan 3.7.

$$f(\theta_1, \theta_2, \theta_3) = \begin{bmatrix} x \\ y \\ \cos \theta_c \end{bmatrix} - \begin{bmatrix} H_{4(1,4)}^0 \\ H_{4(2,4)}^0 \\ H_{4(1,1)}^0 \end{bmatrix} = \begin{bmatrix} x \\ y \\ \cos \theta_c \end{bmatrix} - \begin{bmatrix} \cos(\theta_a) + 130 \cos(\theta_b) + 195 \cos \theta_1 + 104 \\ \sin(\theta_a) + 130 \sin(\theta_b) + 195 \sin \theta_1 \\ \cos(\theta_a) \end{bmatrix} \quad 3.13$$

Dimana : $\theta_a = \theta_1 + \theta_2 + \theta_3$, $\theta_b = \theta_1 + \theta_2$

$f'(x)$ pada persamaan 3.7 merupakan turunan dari $f(x)$ terhadap θ_1 , θ_2 dan θ_3 .

$$f'(\theta_1, \theta_2, \theta_3) = j(\theta_1, \theta_2, \theta_3) = \begin{bmatrix} \frac{\partial(H_{4(1,4)}^0)}{\partial \theta_1} & \frac{\partial(H_{4(1,4)}^0)}{\partial \theta_2} & \frac{\partial(H_{4(1,4)}^0)}{\partial \theta_3} \\ \frac{\partial(H_{4(2,4)}^0)}{\partial \theta_1} & \frac{\partial(H_{4(2,4)}^0)}{\partial \theta_2} & \frac{\partial(H_{4(2,4)}^0)}{\partial \theta_3} \\ \frac{\partial(H_{4(1,1)}^0)}{\partial \theta_1} & \frac{\partial(H_{4(1,1)}^0)}{\partial \theta_2} & \frac{\partial(H_{4(1,1)}^0)}{\partial \theta_3} \end{bmatrix} \quad 3.14$$

$$= \begin{bmatrix} -\sin(\theta_a) - 130 \sin(\theta_b) - 195 \sin \theta_1 & -\sin(\theta_a) - 130 \sin(\theta_b) & -\sin(\theta_a) \\ \cos(\theta_a) + 130 \cos(\theta_b) + 195 \cos \theta_1 & \cos(\theta_a) + 130 \cos(\theta_b) & \cos(\theta_a) \\ -\sin(\theta_a) & -\sin(\theta_a) & -\sin(\theta_a) \end{bmatrix}$$

Sehingga persamaan 2.27 menjadi persamaan 3.15

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}_{n+1} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}_n - \frac{\begin{bmatrix} x \\ y \\ \cos(\theta_c) \end{bmatrix} - \begin{bmatrix} \cos(\theta_a) + 130 \cos(\theta_b) + 195 \cos \theta_1 + 104 \\ \sin(\theta_a) + 130 \sin(\theta_b) + 195 \sin \theta_1 \\ \cos(\theta_a) \end{bmatrix}}{\begin{bmatrix} -\sin(\theta_a) - 130 \sin(\theta_b) - 195 \sin \theta_1 & -\sin(\theta_a) - 130 \sin(\theta_b) & -\sin(\theta_a) \\ \cos(\theta_a) + 130 \cos(\theta_b) + 195 \cos \theta_1 & \cos(\theta_a) + 130 \cos(\theta_b) & \cos(\theta_a) \\ -\sin(\theta_a) & -\sin(\theta_a) & -\sin(\theta_a) \end{bmatrix}}$$

Persamaan 3.15 dihitung secara berulang hingga solusi θ_1 , θ_2 dan θ_3 yang baru memiliki selisih dengan θ_1 , θ_2 dan θ_3 lama yang lebih kecil dari $\varepsilon = 10^{-3}$. Saat selisih dari θ_1 , θ_2 dan θ_3 sudah lebih kecil dari ε , maka θ_1 , θ_2 dan θ_3 yang baru merupakan solusi dari kinematika mundur.

c. Perancangan catu daya dan *driver* motor *stepper*

Dalam perancangan catu daya pertama harus diketahui terlebih dahulu kebutuhan listrik dari robot lengan. Pada robot lengan terdapat beberapa komponen yang memerlukan sinyal listrik seperti motor *stepper*, dan sakelar kontak. Selain pada robot lengan terdapat juga komponen lain yang memerlukan energi listrik seperti Arduino dan beberapa *driver* untuk motor *stepper*. Kebutuhan energi listrik beberapa komponen dapat dilihat pada tabel 3.1 berikut.

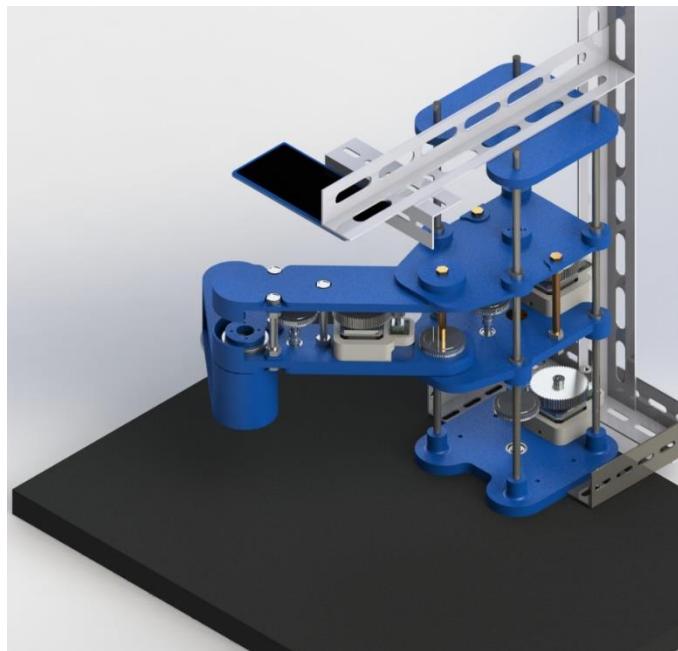
Tabel 3.1 Kebutuhan Daya Komponen

Komponen	Tegangan	Arus	Daya
Motor <i>Stepper</i>	4 V – 12 V	2 A (Max) @ 12	24W
Arduino	7 V	200mA (Max)	1.4 W
Lain ²	-	-	10 W
Total Daya			35.4 W

Dari Tabel 3.1 semua komponen pada penelitian membutuhkan daya maksimal sebesar 22.8 watt. Dengan ini digunakan catu daya dengan spesifikasi tegangan keluaran 12 V dan arus 3 A. Beberapa perangkat seperti motor *stepper* dan elektromagnet harus mendapat tegangan arus yang sesuai maka dari itu harus dibuat *driver*-nya. Motor *stepper* akan dapat berputar apabila kumparan medannya diberi sinyal berupa pulsa yang berurutan, sinyal ini dihasilkan dari mikrokontroler, mikrokontroler hanya mampu menghasilkan tegangan sebesar 5V dan arus sebesar 20mA, properti sinyal seperti ini tidak akan dapat memutar motor *stepper* sehingga dibutuhkan bantuan dari Modul A4988, Modul A4988 akan mengambil arus tambahan dari catu daya untuk memperkuat sinyal dari mikrokontroler sehingga motor dapat berputar.

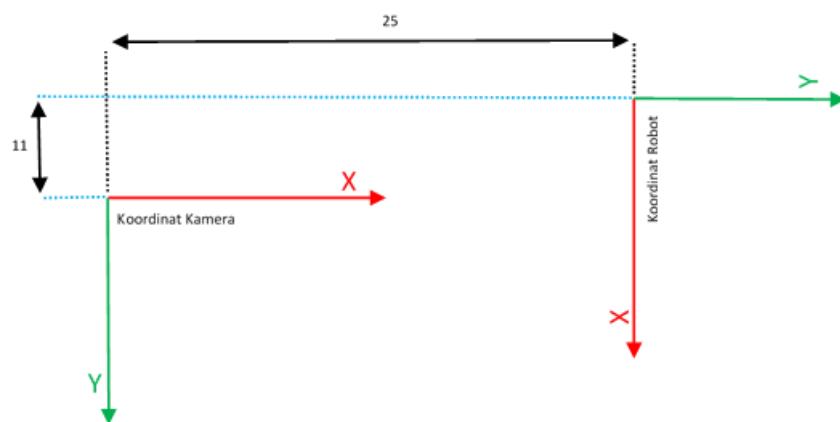
d. Perancangan Peletakan Kamera dan Robot Lengan

Pada penelitian ini kamera digunakan untuk mendeteksi lokasi dari benda yang akan berinteraksi dengan robot lengan. Gambar dan video dari kamera dikirimkan ke komputer untuk diolah citranya sehingga diperoleh koordinat dari benda, lalu dilakukan kinematika mundur. Peletakan kamera dan robot lengan dapat dilihat pada Gambar 3.8. Sebuah permasalahan muncul saat menggunakan kamera sebagai masukan dari robot yaitu acuan koordinat yang berbeda antara manipulator dengan citra tangkapan kamera. Hal ini dapat diatasi dengan mentransformasi koordinat kamera ke robot.



Gambar 3.8 Peletakan Kamera dan Robot Lengan.

Koordinat benda yang terbaca oleh kamera harus diubah ke koordinat *manipulator*. Ilustrasi transformasi koordinat kamera ke robot dapat diamati pada Gambar 3.9.



Gambar 3.9 Perbedaan Jarak dan Orientasi Koordinat nol pada Kamera dan Robot.

Pada Gambar 3.9 dapat dibuat matriks transformasi menggunakan teori *homogenous transformation matrices*. Matriks transformasi homogen koordinat kamera relatif terhadap koordinat robot dapat diamati pada persamaan 3.9.

$$HTM_{robot}^{kamera} = \begin{bmatrix} 0 & 1 & 0 & 11 \\ 1 & 0 & 0 & -25 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad 3.9$$

Dengan mengalikan koordinat benda yang diperoleh kamera terhadap matriks pada persamaan 3.9 maka akan diperoleh koordinat benda dengan perspektif dari robot.

3.5.2 Perancangan perangkat lunak

Perancangan perangkat lunak terdiri dari perancangan program Arduino, dan tampilan program Python.

A. Perancangan program arduino

Arduino digunakan sebagai penghubung robot lengan dengan komputer, Arduino akan menerima perintah dari komputer melalui koneksi serial berupa posisi masing-masing *joint* pada robot lengan lalu menggerakkan motor *stepper* pada robot lengan. Tampilan program Arduino dapat dilihat pada Gambar 3.10.

Bagian program pada Arduino terbagi menjadi dua bagian utama yaitu ”*void setup*” dan ”*void loop*”. *Void setup* berisikan inisiasi pertama yang dilakukan robot saat pertama kali digunakan, saat pertama kali dijalankan robot lengan akan menggerakkan lengannya ke lokasi awal. Hal ini dilakukan karena motor *stepper* tidak dapat mengirimkan umpan balik berupa posisi rotornya ke Arduino.

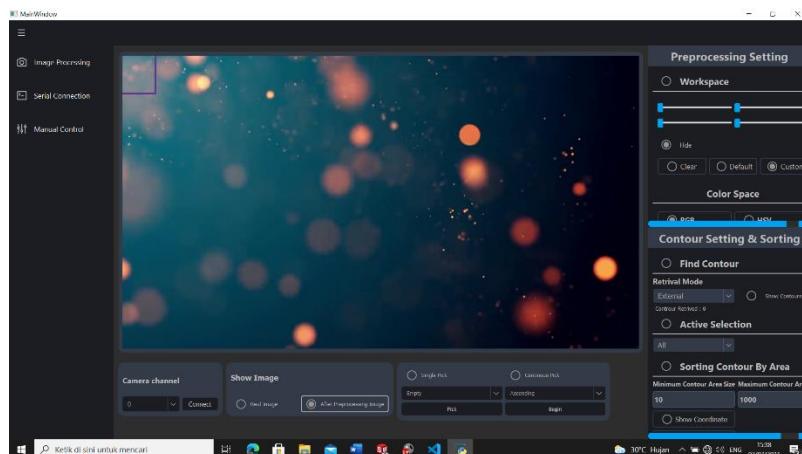
```
sketch_sep19a | Arduino 1.8.12
File Edit Sketch Tools Help
sketch_sep19a
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

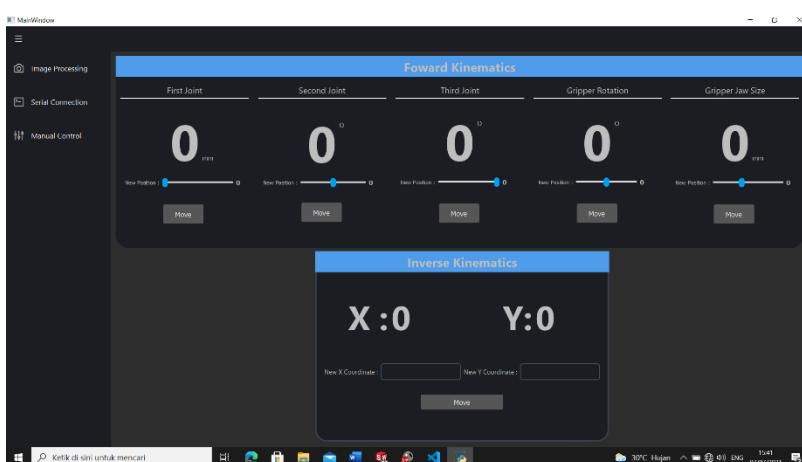
Gambar 3.10 Tampilan Program Arduino

B. Perancangan GUI dan program kinematika mundur dengan python

Bahasa pemrograman Python digunakan untuk menghitung kinematika mundur, mengolah citra dari kamera dan membuat GUI kendali manual untuk robot lengan. OpenCV digunakan sebagai *library* untuk mengolah gambar dari kamera, citra dari kamera akan diolah sehingga diperoleh koordinat dari benda selanjutnya koordinat dari benda digunakan untuk melakukan kinematika mundur. Selain untuk mengolah citra kamera dan menghitung kinematika mundur, program yang dibuat dengan Python juga dibuat sebuah GUI untuk melihat secara langsung penampakan kamera serta GUI untuk mengontrol pergerakan robot secara manual seperti saat hanya ingin menggerakkan salah satu *joint* saja pada lengan robot. Tampilan GUI program dapat diamati pada Gambar 3.12 dan 3.13.



Gambar 3.11 Tampilan GUI Client App.



Gambar 3.12 Tampilan GUI Client App , Manual Control Menu.

Gambar 3.12 menampilkan tampilan menu utama dari aplikasi klien pada komputer, pada tampilan tersebut terdapat beberapa tombol kontrol dan jendela, tombol kontrol yang terdapat pada tampilan berupa tombol untuk menghubungkan ke kamera, tombol pengatur area kerja kamera, serta tombol pengatur nilai *thresholding canny edge detection*. Pada tampilan juga terdapat jendela yang berisikan citra video yang ditangkap kamera, serta kontainer yang berisikan tombol untuk aksi kinematik pada robot lengan. Gambar 3.13 menunjukkan tampilan pada Tab kedua pada aplikasi *client*. Pada tampilan ini terdapat kontainer yang berisikan tombol untuk mengatur koneksi serial serta mengatur nilai masing-masing *joint* pada robot lengan secara manual.

Perhitungan kinematika mundur dan pengolahan citra dilakukan pada aplikasi *client* ini, pada program terdapat banyak sub-bagian, tetapi hanya akan dijelaskan proses kinematika serta pengolahan citra dengan Canny *edge detection*.
Diagram Alir Sistem

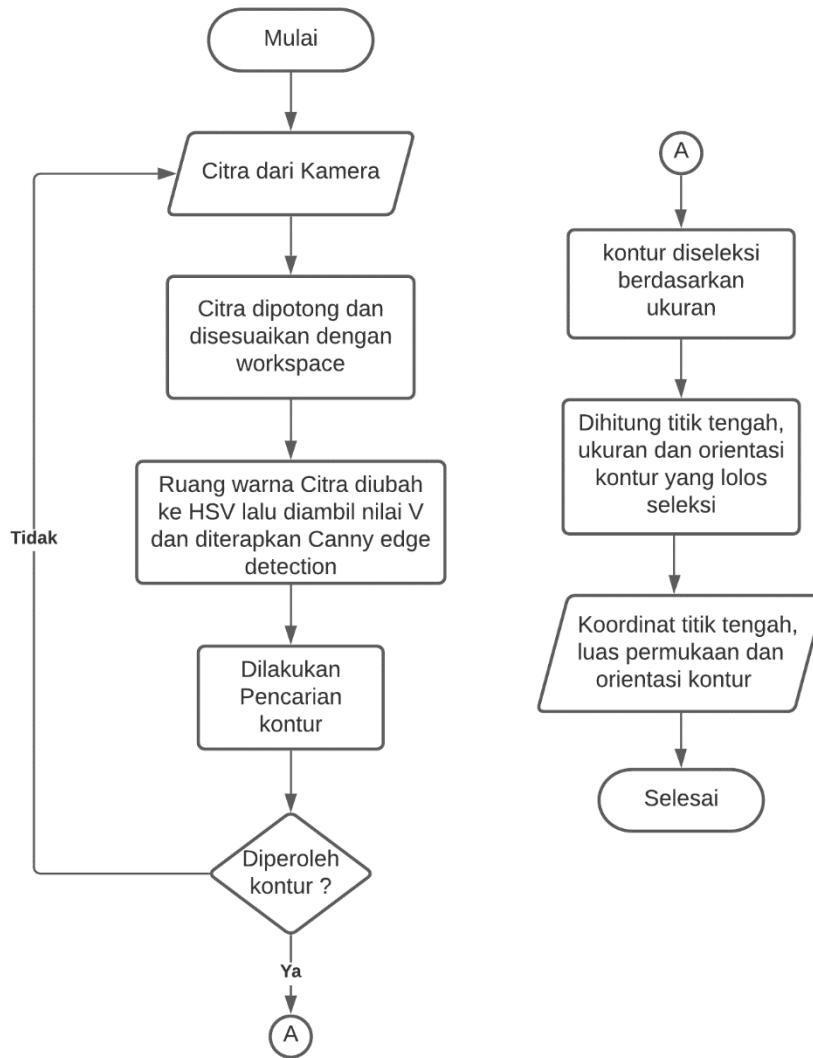
Diagram alir sistem robot lengan pemindah barang akan menjelaskan langkah-langkah kinerja secara lebih mendetail, diagram alir dibagi menjadi tiga yaitu diagram alir kinematika, diagram alir pengolahan citra dan diagram alir keseluruhan.

C. Diagram alir pengolahan citra

Diagram alir proses pengolahan citra hingga menghasilkan koordinat benda dapat diamati pada Gambar 3.14.

Diagram alir pada Gambar 3.14 dimulai dari memasukkan citra dari kamera, citra tersebut lalu dipotong berdasarkan ukuran area kerja, citra yang telah dipotong tersebut lalu dijadikan sebagai masukan dari pengolahan citra Canny, sebelum diolah citra terlebih dahulu diubah ke bentuk keabu-abuan. Citra keluaran dari pengolahan Canny berbentuk citra biner, citra biner ini kemudian dijadikan masukan untuk proses pencarian kontur. Kontur yang diperoleh dari citra biner jumlahnya tergantung dari teknik perolehan kontur dan citra biner hasil pengolahan, digunakan teknik pencarian kontur tertutup sehingga pinggiran objek yang dihitung sebagai kontur adalah pinggiran yang tertutup saja. Setelah kontur diperoleh, kontur-kontur tersebut diurutkan berdasarkan ukurannya. Dengan ini benda yang

terdeteksi dalam kamera akan diurutkan indeksnya berdasarkan ukuran, baik dari besar kecil atau kecil ke besar.



Gambar 3.13 Diagram Alir Pengolahan Citra

Kontur yang telah diurutkan lalu diseleksi kembali berdasarkan ukuran luas areanya, kontur tertutup dengan luas area yang terlalu kecil bisa saja bukan objek melainkan *noise* seperti bayangan atau refleksi cahaya yang dianggap sebagai pinggiran objek, sedangkan apabila area kontur terlalu besar bisa saja kontur tersebut merupakan pinggiran dari area kerja robot. Kontur yang lolos seleksi lalu dihitung koordinat titik tengahnya, koordinat ini akan digunakan sebagai keluaran dari pengolahan citra dan siap untuk diolah dengan program berikutnya.

Diagram alir kinematika

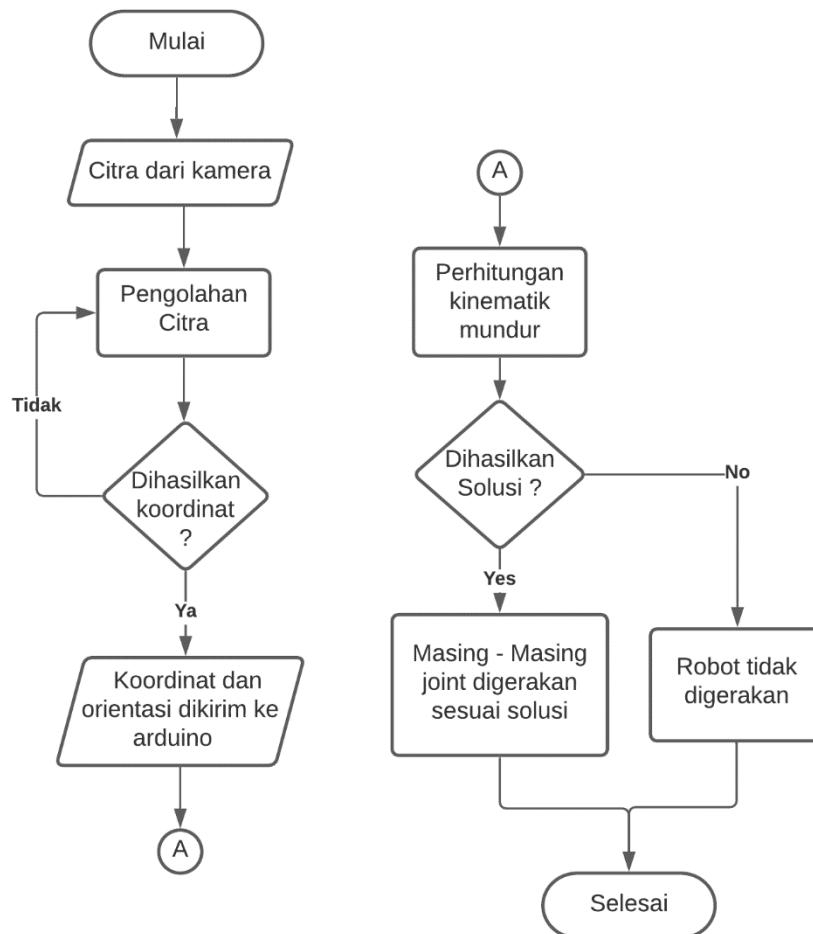
Diagram alir untuk kinematika mundur dapat diamati pada Gambar 3.15.



Gambar 3.14 Diagram Alir Kinematika Mundur.

Pada Gambar 3.14 menunjukkan aliran bagaimana koordinat benda diubah menjadi *joint space*. Koordinat benda hasil keluaran dari pengolahan citra dalam satuan panjang milimeter dijadikan masukan dari perhitungan dengan menggunakan persamaan 3.7 dan 3.8, apabila diperoleh solusi berupa sudut pada masing-masing *joint*, maka nilai sudut tersebut disimpan lalu perhitungan kinematika pun selesai. Apabila perhitungan tidak menghasilkan solusi berarti benda berada diluar area jangkauan robot, sehingga alirannya langsung menyelesaikan perhitungan tanpa menyimpan hasil.

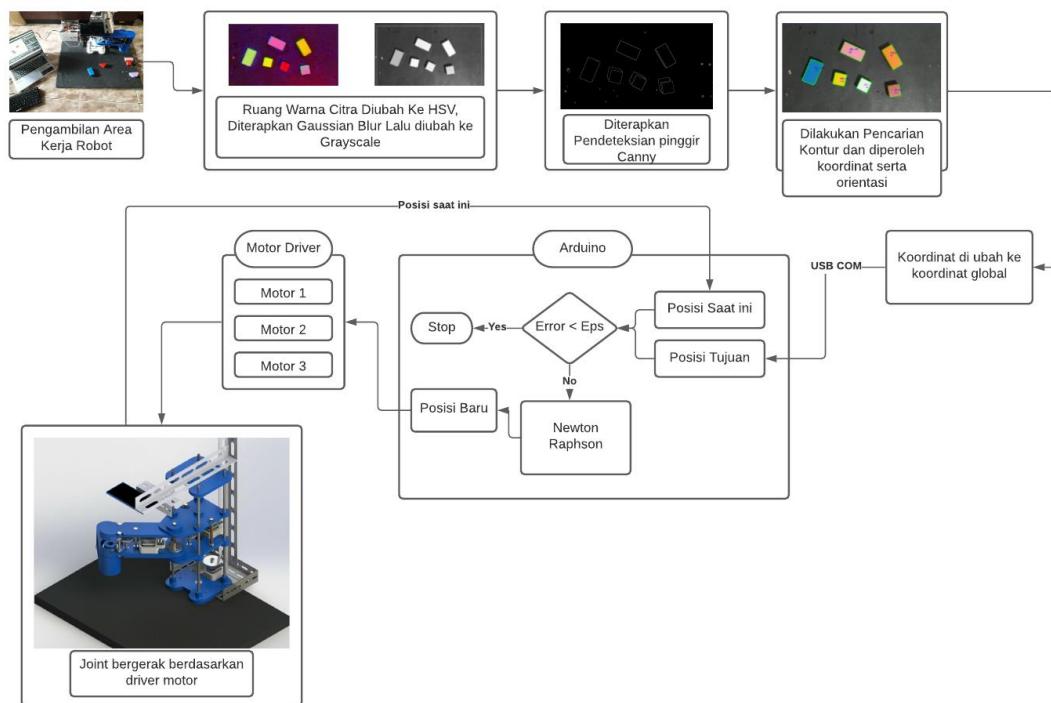
D. Diagram alir keseluruhan



Gambar 3.15 Diagram Alir Keseluruhan.

Diagram alir pada Gambar 3.16 menunjukkan aliran kinerja dari keseluruhan sistem, pertama pada sistem dimasukkan citra area kerja dari robot lengan. Kamera digunakan sebagai sensor untuk menangkap citra, citra lalu dikirim ke komputer untuk diolah. Pada komputer citra diolah, keluaran dari pengolahan citra ini menghasilkan koordinat titik tengah benda yang akan dipindahkan oleh robot lengan. Koordinat yang dihasilkan pengolahan masih memiliki satuan piksel serta tidak selaras dengan koordinat nol pada robot lengan, koordinat ini lalu diubah ke satuan milimeter dan diselaraskan dengan koordinat nol robot lengan. Setelah dikonversi koordinat ini menjadi masukan untuk perhitungan kinematika mundur, perhitungan kinematika mundur menghasilkan posisi sudut masing-masing *joint* pada manipulator robot lengan. Nilai sudut ini lalu dikonversi menjadi jenis data

yang siap dikirim melalui koneksi serial komputer ke Arduino, setelah dikonversi data langsung dikirim. Rangka kerja sistem keseluruhan dapat diamati pada Gambar 3.16.



Gambar 3.16 Rangka Kerja Sistem Keseluruhan

Pada sisi Arduino setelah menerima data dari komputer, data langsung diolah dengan dikonversi kembali nilai sudut *joint*. Nilai sudut ini lalu menjadi masukkan untuk fungsi penggerak motor *stepper* pada arduino. Robot lengan pun akan memindahkan robot lengan sesuai dengan informasi yang masuk dari komputer.

3.6 Metode Pengujian

Pada metode pengujian ini, akan dijelaskan langkah-langkah pengujian dengan tujuan untuk mengetahui keandalan dari sistem yang telah dirancang dalam memenuhi tujuan penelitian, adapun pengujian yang dilakukan sebagai berikut.

3.6.1 Pengujian pengolahan citra

Pengujian pengolahan citra dilakukan dengan beberapa poin pengujian, pengujian ini bertujuan untuk mengetahui apakah metode pengolahan citra yang

telah dirancang untuk mendeteksi keberadaan serta memperoleh koordinat benda dapat bekerja sebagai mana mestinya, pengujian yang dilakukan sebagai berikut :

1. Pengujian deteksi pinggir dengan metode Canny.

Pada pengujian ini akan dilihat apakah metode Canny yang diterapkan pada bahasa pemrograman Python mampu mendeteksi pinggiran dari objek yang diinginkan, pada pengujian ini kamera menangkap citra dengan area kerja robot yang dilapisi stiker putih, serta benda yang akan dipindahkan berbentuk persegi, persegi panjang dan lingkaran dengan benda hanya memiliki satu warna saja. Pengujian dilakukan dengan melihat citra hasil pengolahan, apakah pada citra tersebut terdapat garis putih yang berbentuk dan berlokasi hampir sama dengan objek yang dideteksinya.

2. Pengujian pencarian dan pengolahan kontur.

Pada pengujian ini untuk mengetahui apakah citra hasil pengolahan Canny dapat dilakukan pencarian konturnya, serta untuk mengetahui apakah kontur yang diperoleh sesuai dengan seleksi yang telah diterapkan, selain itu pengujian ini juga dilakukan untuk mengetahui apakah koordinat kontur yang diperoleh sama dengan koordinat benda yang sebenarnya pada area kerja robot. Untuk pengujian deteksi kontur dilakukan dengan menggambarkan titik pada piksel yang terdeteksi sebagai kontur. Pengujian seleksi kontur dilakukan dengan membandingkan bentuk dan ukuran kontur sesuai dengan syarat seleksinya. Pengujian koordinat dilakukan dengan membandingkan koordinat yang dihasilkan pengolahan dan koordinat benda asli yang diukur menggunakan penggaris.

3.6.2 Pengujian robot lengan

Pengujian robot lengan dilakukan dengan beberapa poin, pengujian ini bertujuan untuk mengetahui apakah robot lengan dapat bergerak sesuai dengan masukan. Pengujian yang dilakukan sebagai berikut :

1. Pengujian keakuratan gerak masing-masing *joint*

Pengujian ini dilakukan untuk mengetahui keakuratan gerak masing-masing *joint* pada manipulator robot lengan, untuk *joint* pertama dilakukan dengan membandingkan nilai masukkan dan nilai pengukuran jarak perpindahan *joint* pertama dengan penggaris.

Pengujian pada *joint* kedua dan ketiga dilakukan dengan membandingkan nilai masukan dan nilai pengukuran jarak perpindahan *joint* kedua dan ketiga dengan busur.

2. Pengujian keakuratan gerak *end of effector*.

Pengujian ini dilakukan untuk mengetahui apakah bagian *end of effector* dari robot lengan menuju lokasi yang sama dengan yang dimasukkan ke robot lengan. Pengujian dilakukan dengan membandingkan nilai masukan dan nilai pengukuran lokasi *end of effector* menggunakan penggaris.

3.6.3 Pengujian keseluruhan sistem

Pengujian ini dilakukan untuk mengetahui apakah sistem mampu bekerja sesuai keinginan serta tujuan penelitian. Adapun pengujian yang dilakukan sebagai berikut :

1. Pengujian sistem memindahkan benda.

Pengujian ini dilakukan untuk mengetahui apakah sistem mampu memindahkan sebuah benda pada area kerja, pengujian dilakukan dengan memberikan sebuah benda pada area kerja lalu memberikan perintah pada sistem untuk memindahkan benda tersebut ke titik tertentu.

2. Pengujian memindahkan benda berdasarkan ukuran

Pengujian ini dilakukan untuk mengetahui apakah sistem mampu memindahkan benda secara urut berdasarkan ukurannya. Pengujian dilakukan dengan memberikan perintah pada sistem untuk memindahkan beberapa benda pada area kerja robot, lalu benda yang dipindahkan oleh robot diukur dan diurutkan berdasarkan ukurannya. Urutan ukuran robot lalu dibandingkan dengan urutan robot memindahkan benda.

3. Pengujian pengelompokan benda.

Pengujian ini dilakukan untuk mengetahui apakah sistem mampu mengelompokkan beberapa benda pada area kerja robot, pengujian dilakukan dengan mengukur ukuran benda yang telah dipindahkan oleh robot lengan lalu dicocokkan dengan area penempatannya.

BAB IV

HASIL DAN PEMBAHASAN

Pada bab empat akan dipaparkan hasil serta pembahasan dari pengujian pada sistem yang telah dibuat. Sistem yang dibuat merupakan sebuah robot lengan manipulator dengan tiga derajat kebebasan, sebuah kamera dijadikan sebagai masukan untuk mendeteksi benda serta ukuran yang akan dipindahkan oleh robot lengan. Solusi analitis digunakan untuk menemukan solusi kinematika mundur dari manipulator robot lengan serta pengolahan citra deteksi pinggir Canny digunakan untuk mendeteksi dan mengukur ukuran objek, kedua metode ini diterapkan dengan menggunakan bahasa pemrograman Python pada komputer. Hasil pengolahan serta perhitungan dari aplikasi pada komputer lalu dikirim melalui koneksi serial ke Arduino untuk menggerakkan robot lengan. Berikut hasil dan pembahasan dari sistem yang telah dibuat.

4.1 Hasil perancangan Sistem.

1. Robot lengan.

Bentuk robot lengan yang telah di buat dapat diamati pada Gambar 4.1.



Gambar 4.1 Hasil Pembuatan Robot Lengan.

Robot lengan dibuat sesuai dengan perancangannya, digunakan motor *stepper* sebagai penggerak dari semua *joint* pada robot lengan, digunakan roda gigi untuk memperhalus gerakan *joint*. Fondasi dari robot dicetak dengan pencetak 3D. Robot lengan yang telah di buat memiliki nilai kebebasan gerak *joint* pertama yaitu 7 – 30 cm, *joint* kedua -90 derajat hingga 90 derajat, dan *joint* 3 -180 derajat hingga

180 derajat, pencapit juga memiliki kebebasan gerak putar -180 derajat hingga 180 derajat.

2. Area kerja robot

Bentuk area kerja robot serta peletakan kamera dapat diamati pada Gambar 4.2.



Gambar 4.2 Area Kerja Robot dan Peletakan Kamera

Area kerja robot dibuat dari bahan kayu, lalu ditutup oleh kertas karton berwarna hitam, hal ini bertujuan untuk mempermudah pemisahan objek dengan latar belakang pada citra, diberikan juga garis batas berwarna hitam pada area kerja robot, hal ini bertujuan untuk mempermudah pengaturan pemotongan citra sebelum diolah. Kamera menggunakan telepon pintar yang dihubungkan ke komputer, sedangkan fondasinya dibuat dari besi.

4.2 Hasil pengujian sistem

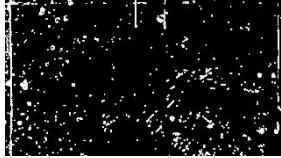
Pada bab tiga telah dijelaskan metode pengujian yang dilakukan pada sistem, pada sub bab ini akan di bahas hasil pengujian, variabel yang diuji serta pembahasan hasil pengujian. Adapun pengujian yang dilakukan sebagai berikut :

4.2.1 Pengujian performa aplikasi dalam mengolah citra masukan.

A. Pengujian aplikasi dalam mendeteksi pinggir objek.

Pada pengujian ini ada beberapa kondisi yang akan di uji, pertama yaitu saat tidak terdapat objek apa pun pada area kerja, saat diberikan sebuah objek pada area kerja, saat area kerja diberikan objek lain serta saat ada pantulan cahaya dan bayangan pada area kerja robot, pengujian dilakukan dengan nilai *Thresholding* rendah sebesar “31” dan nilai *Thresholding* tinggi sebesar “143” dalam pengaturan deteksi pinggir Canny . Berikut hasil pengujian dapat diamati pada Tabel 4.1.

Tabel 4.1 Hasil Pengujian Deteksi Pinggir Objek dengan Canny Menggunakan Python.

No	Variabel Pengujian	Gambar Masukan	Gambar Keluaran Pengolahan Canny
1	Pengujian dengan tanpa objek pada area kerja		 Gambar 4.4 Keluaran Pengujian Canny 1
2	Pengujian dengan ditambahkan objek pada area kerja		 Gambar 4.6 Keluaran Pengujian Canny 2
3	Pengujian dengan penambahan cahaya dan gaussian blur		 Gambar 4.8 Keluaran Pengujian Canny 3

Pada pengujian pertama, pada area kerja robot tidak diberikan objek apa pun lalu diambil citranya. Citra masukan pengujian pertama dapat diamati pada Gambar 4.3, sedangkan keluarannya yang telah diolah dengan metode Canny dapat diamati pada Gambar 4.4. Garis dan titik putih pada Gambar 4.4 merupakan pinggiran yang terdeteksi oleh pengolahan Canny, jika dibandingkan dengan gambar masukannya, garis dan titik putih yang dianggap pinggiran objek oleh pengolahan Canny merupakan bercak pada kertas karton dan pantulan cahaya dari lampu.

Pengujian kedua dilakukan dengan memberikan beberapa objek pada area kerja robot lalu diamati citra hasil pengolahannya. Objek yang diberikan berupa balok kayu yang hanya memiliki satu jenis warna saja pada seluruh permukaannya. Citra masukan pada pengujian ketiga ini dapat diamati pada Gambar 4.5. pada Gambar 4.5 muncul bayangan di area sekitar objek, sehingga hasil pendekripsi pinggir pun menganggap pinggiran bayangan tersebut sebagai pinggiran objek. Citra Hasil pengujian ketiga dapat diamati pada Gambar 4.6, pada Gambar tersebut

baik pinggiran objek atau pinggiran bayangan dari objek di anggap sebagai pinggiran objek.

Pengujian ketiga dilakukan dengan memberikan cahaya pada area kerja robot serta diterapkan pengolahan citra *gaussian blur* sebelum diterapkan pendekripsi pinggir *canny*. Citra masukan dapat diamati pada Gambar 4.7 dan citra keluaran pada Gambar 4.8. Pada citra keluaran dapat diamati bahwa pemberian cahaya dan penerapan *gaussian blur* mampu menseleksi *noise* dan bayangan.

Dari ketiga hasil pengujian dapat ditarik kesimpulan bahwa sistem pendekripsi pinggir Canny yang diterapkan dengan bahasa pemrograman Python mampu mendekripsi pinggiran sebuah objek pada area kerja robot, pemberian cahaya pada area kerja dan penerapan *gaussian blur* mampu menseleksi *noise* dan bayangan.

B. Pengujian aplikasi dalam mengolah citra keluaran deteksi pinggir Canny.

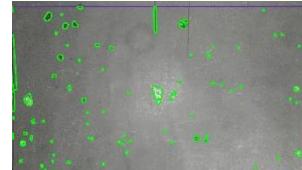
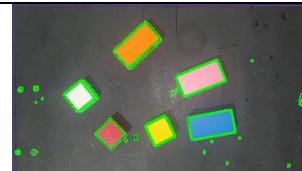
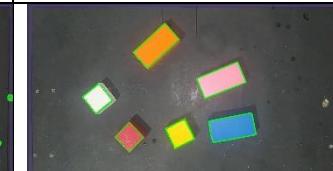
Pada pengujian ini variabel yang diuji yaitu saat pada area kerja robot tidak diberikan benda apa pun, serta saat diberikan objek untuk dipindahkan. Untuk pengujian pengolahan kontur, variabel yang diuji berupa pengukuran dengan ukuran objek yang berbeda beda. Pengujian dilakukan untuk mengetahui apakah pengolahan kontur yang telah dibuat dapat menghasilkan kontur dari objek, menghitung propertinya serta mengeliminasi kontur yang dianggap sebagai objek yang telah lolos pada deteksi pinggir Canny.

1. Pengujian pencarian dan seleksi kontur.

Hasil pengujian dapat diamati pada Tabel 4.2. pada pengujian yang pertama, untuk mengetahui apakah ada kontur yang lolos seleksi saat tidak ada objek pada area kerja. Pada Gambar 4.10 menunjukkan bahwa beberapa bercak terdeteksi sebagai kontur, tetapi kontur tersebut tidak lolos seleksi sehingga tidak akan masuk dalam perhitungan.

Tabel 4.2 Hasil Pengujian Pencarian dan Seleksi Kontur

No .	Gambar masukan	Gambar Dengan kontur yang terdeteksi	Gambar dengan kontur yang lolos seleksi

1			
2			

Pada pengujian yang kedua, untuk mengetahui apakah objek yang diletakan pada area kerja robot dapat dicari konturnya serta lolos seleksi. Gambar 4.13 menunjukkan bahwa objek berhasil dideteksi konturnya dan Gambar 4.14 menujukan kontur yang terseleksi hanya kontur bagian terluar dari objek saja.

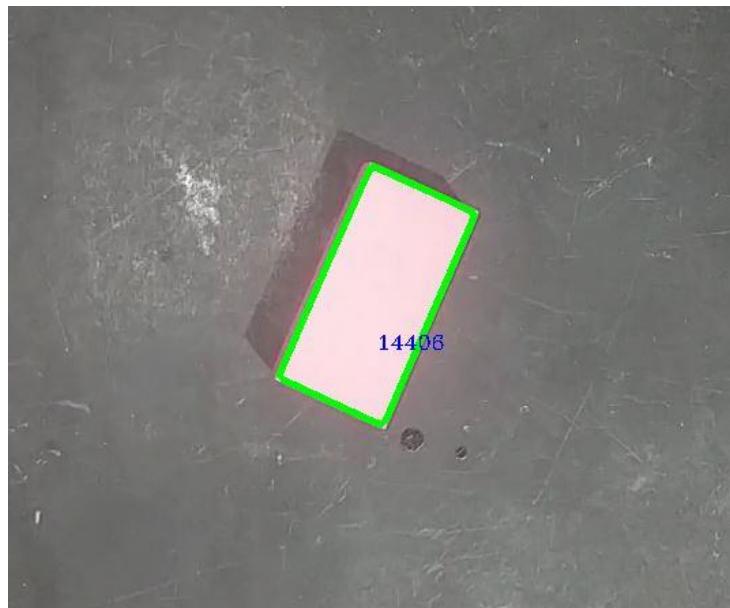
Dari dua pengujian ini dapat disimpulkan bahwa program yang telah dibuat mampu menemukan kontur dari citra hasil pengolahan Canny, serta mampu menyeleksi kontur yang bukan merupakan kontur dari objek.

2. Pengujian ukuran objek.

Pengujian ini untuk mengetahui seberapa akurat program yang telah dibuat dalam mengukur ukuran luas permukaan sebuah objek. Ukuran objek dalam satuan piksel diubah ke mm kuadrat lalu dibandingkan dengan ukuran objek yang sebenarnya, persamaan untuk mengonversi piksel ke mm terdapat pada persamaan 4.1. Ilustrasi pengukuran dapat diamati pada Gambar 4.15.

$$\text{Luas daerah objek (mm)} = K_l \times \text{Luas daerah objek (piksel)} \quad 4.1$$

Pada persamaan 4.1 K_l merupakan konstanta luas, pada pengujian ini digunakan konstanta luas sebesar “0.00153”.



Gambar 4.15 Luas Permukaan Objek Yang Terukur Oleh Pengolahan Citra.

Gambar 4.17 menunjukkan luas permukaan objek adalah sebesar 14406 piksel, menggunakan persamaan 4.1 maka dalam satuan milimeter diperoleh luas permukaan sebesar 2204 mm^2 . Luas permukaan benda sesungguhnya sebesar 2145 mm^2 , luas permukaan ini diukur dengan menggunakan penggaris. Nilai galat yang diperoleh pun sebesar 2.7%. Berikut data hasil pengujian dengan variasi ukuran.

Tabel 4.3 Hasil Pengujian Ukuran Luas Permukaan Objek

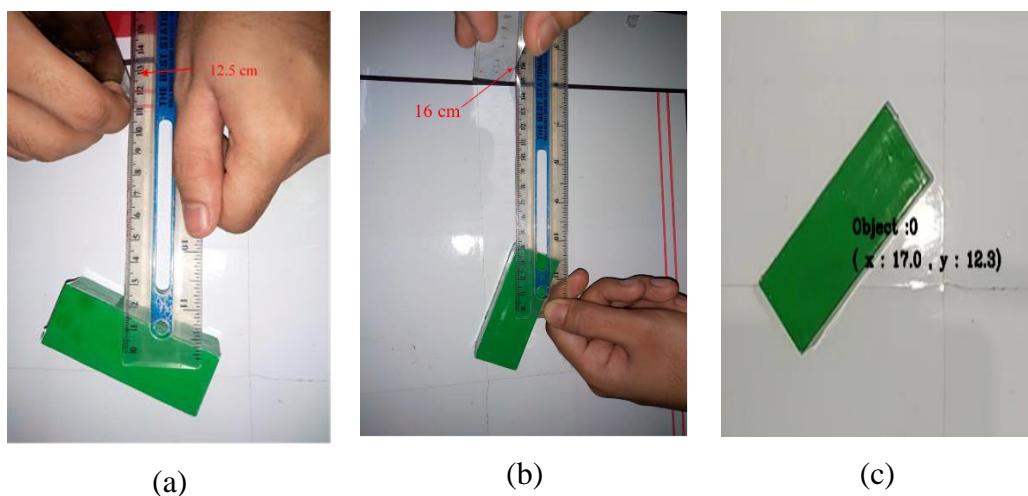
No	Luas permukaan objek (mm^2)	Luas permukaan objek hasil pengukuran		Galat (%)
		(Piksel)	(mm^2)	
1	1089	6925	1059	2.7
2	2145	14635	2204	2.7
3	1658	12495	1611	2.8
4	1012	6600	1009	0.29
5	450	2800	428	4.8
Galat rata-rata				2.6

Dari hasil pengujian diperoleh rata-rata galat sebesar 2.6%, dengan galat terbesar pada objek yang memiliki ukuran yang lebih kecil. Nilai galat ini cukup kecil,

Dari hasil pengujian, nilai rata rata galat sebesar 2.6%, hal ini menunjukkan bahwa program yang telah dibuat cukup baik dalam mengukur luas permukaan sebuah objek.

3. Pengujian Koordinat tengah objek.

Pengujian ini untuk mengetahui seberapa akurat program yang telah dibuat dalam mengukur koordinat tengah objek yang ada pada area kerja robot. Pengujian dilakukan dengan meletakan benda di sembarang tempat dalam area kerja robot lalu mengukur koordinatnya dengan penggaris, hasil pengukuran ini lalu dibandingkan dengan yang terukur oleh program. Pada pengujian ini menggunakan koordinat robot, pengukuran dilakukan dalam satuan sentimeter. Ilustrasi pengukuran dapat diamati pada Gambar 4.16.



Gambar 4.16 a. Pengukuran pada Sumbu y, b. Pengukuran pada Sumbu x, c. Hasil Pengukuran dari Pengolahan Citra

Gambar 4.16c menunjukkan hasil pengukuran dari pengolahan citra, terukur sumbu x sebesar 17 cm, pada Gambar 4.16b koordinat sumbu x yang terukur dengan penggaris sebesar 16 cm. Koordinat terukur sumbu y sebesar 12.3 cm dan pada Gambar 4.16a koordinat sumbu y yang terukur dengan penggaris sebesar 12.5 cm. Dari data ini, nilai galat pergeseran sebesar 3.36%. berikut beberapa variasi pengujian.

Tabel 4.4 Hasil Pengujian Koordinat Objek

No	Koordinat nyata		Koordinat Terukur		Galat	
	Sumbu x (mm)	Sumbu y (mm)	Sumbu x (mm)	Sumbu y (mm)	Sumbu x (%)	Sumbu y (%)
1	170	-210	165	-214	2,94	1,90

2	210	-210	206	-216	1,90	2,86
3	250	-210	244	-215	2,40	2,38
4	290	-210	284	-217	2,07	3,33
5	330	-210	322	-217	2,42	3,33
6	170	-130	168	-132	1,18	1,54
7	210	-130	205	-132	2,38	1,54
8	250	-130	245	-133	2,00	2,31
9	290	-130	284	-135	2,07	3,85
10	330	-130	324	-136	1,82	4,62
11	170	-50	169	-50	0,59	0,00
12	210	-50	205	-51	2,38	2,00
13	250	-50	243	-49	2,80	2,00
14	290	-50	283	-51	2,41	2,00
15	330	-50	322	-50	2,42	0,00
16	170	30	168	31	1,18	3,33
17	210	30	208	30	0,95	0,00
18	250	30	246	31	1,60	3,33
19	290	30	287	32	1,03	6,67
20	330	30	325	31	1,52	3,33
21	170	150	167	153	1,76	2,00
22	210	150	210	155	0,00	3,33
23	250	150	250	150	0,00	0,00
24	290	150	286	152	1,38	1,33
25	330	150	327	154	0,91	2,67
No	Koordinat nyata		Koordinat Terukur		Galat	
	Sumbu x (mm)	Sumbu y (mm)	Sumbu x (mm)	Sumbu y (mm)	Sumbu x (%)	Sumbu y (%)
26	170	-210	165	-136	1,82	4,62
27	210	-210	206	-50	0,59	0,00
Rata - rata					1,68	2,39

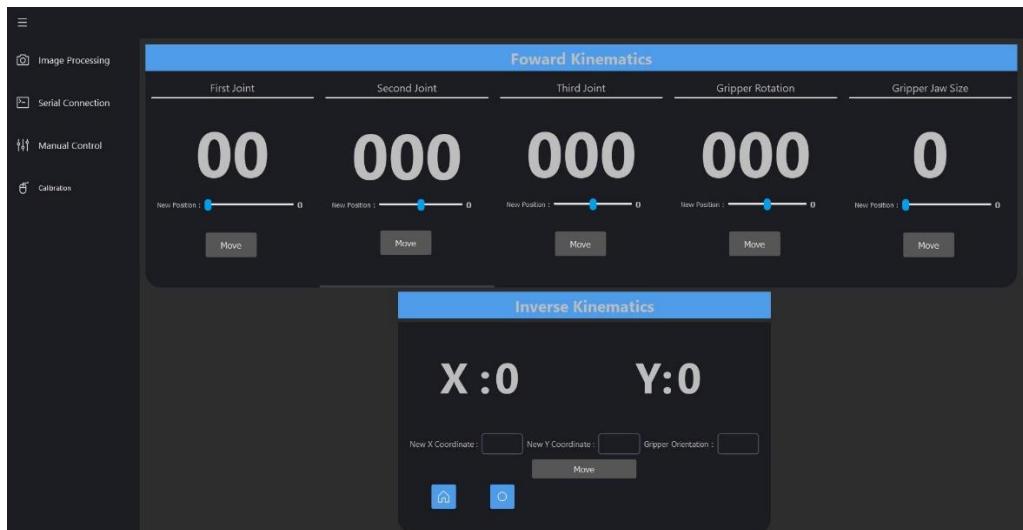
Dilakukan dua puluh tujuh pengujian dengan koordinat objek yang berbeda, galat terbesar terjadi pada pengujian ke-19 dengan nilai galat 6,67%, nilai rata-rata galat sebesar 1,68% untuk sumbu x dan 2,39 untuk sumbu y. Galat ini disebabkan oleh sudut pandang dari peletakan kamera serta kurang akuratnya perhitungan nilai konstanta peubah satuan piksel ke mili meter.

Dari ketiga pengujian pengolahan citra yang dilakukan dapat ditarik kesimpulan bahwa deteksi pinggir Canny yang diterapkan pada bahasa pemrograman Python mampu mendeteksi keberadaan objek, dengan mengolah citra hasil deteksi pinggir Canny dapat dicari kontur dari objek lalu diukur ukuran serta koordinat tengahnya, pengukuran ukuran luas objek memiliki galat rata-rata 41% yang artinya program yang dibuat belum terlalu mumpuni dalam mengukur luas ukuran objek, pengukuran titik tengah objek memiliki galat rata-rata sebesar 3.6% yang artinya program mampu mengukur koordinat tengah objek.

4.2.2 Pengujian performa gerak robot dan kinematika mundur.

A. Pengujian keakuratan gerak masing-masing *joint* pada robot lengan.

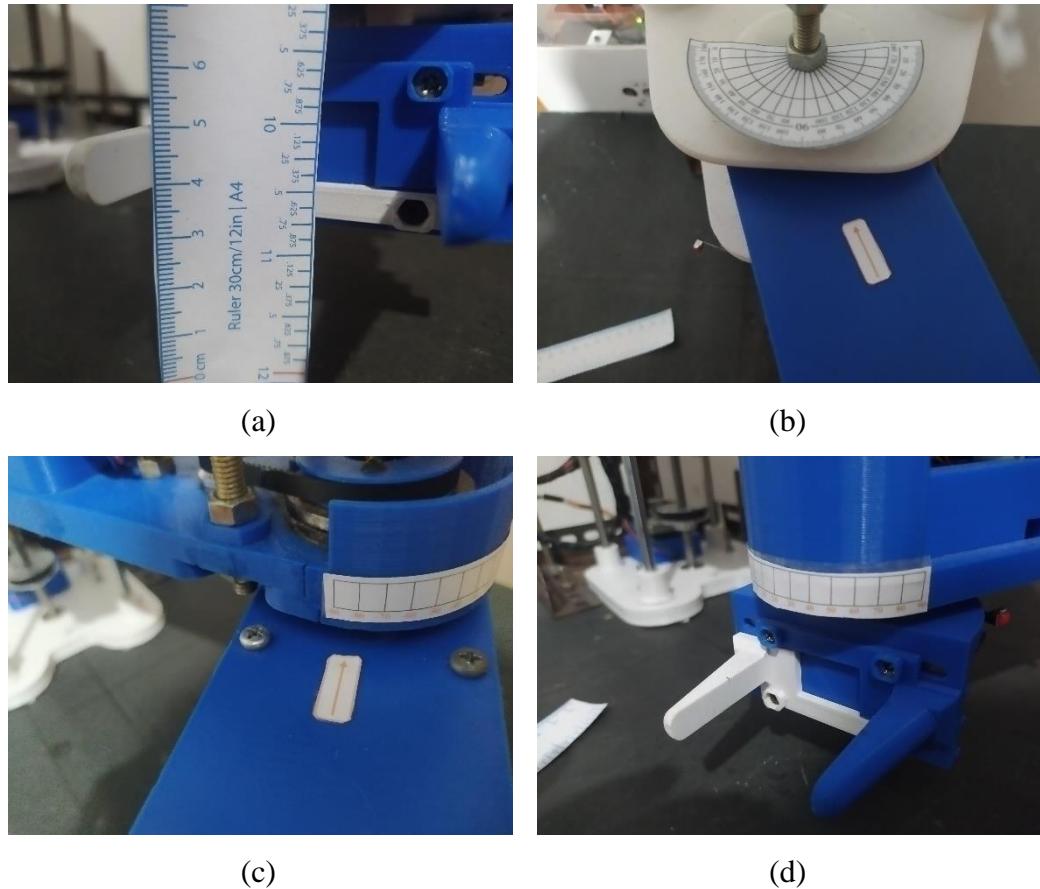
Pengujian ini dilakukan dengan memberikan masukan nilai *joint* yang diinginkan lalu membandingkannya dengan nilai *joint* yang diukur secara langsung. Ilustrasi pengujian dapat diamati pada Gambar 4.17 dan 4.18.



Gambar 4.17 Tampilan Menu Aplikasi untuk Memasukkan Nilai *Joint* Secara Manual.

Gambar 4.17 merupakan menu dari aplikasi yang digunakan untuk menggerakkan *joint* robot secara manual. pada setiap *joint* robot ditempel kertas

pengukur untuk membandingkan nilai inputan dari aplikasi terhadap pengukuran. Kertas pengukur dari setiap *joint* dapat diamati pada Gambar 4.18.



Gambar 4.18 a. Pengukuran *Joint* 1, b. Pengukuran *Joint* 2, c. Pengukuran *Joint* 3,
d. Pengukuran *Joint* 4

Hasil pengujian gerak masing masing *joint* dapat diamati pada Tabel 4.5.

Tabel 4.5 Hasil pengujian gerak masing masing *joint*

Joint 1 (Geser)			
No	Nilai Masukan (mm)	Nilai Terukur (mm)	Galat (%)
1	10	12	4
2	20	22	4
3	30	29	2
4	40	40	0
5	50	51	2
6	60	63	6
7	70	73	6
8	80	83	6

Joint 2 (Putar)			
No	Nilai Masukan (mm)	Nilai Terukur (mm)	Galat (%)
1	-80	-80	0
2	-50	-50	0
3	-10	-10	0
4	30	30	0
5	50	50	0
6	70	70	0
7	90	90	0
Joint 3 (Putar)			
No	Nilai Masukan (mm)	Nilai Terukur (mm)	Galat (%)
1	-80	-80	0
2	-50	-50	0
3	-10	-10	0
4	30	30	0
5	50	50	0
6	70	70	0
7	90	90	0
Joint 4 (Putar)			
No	Nilai Masukan (mm)	Nilai Terukur (mm)	Galat (%)
1	-90	-88	4
2	-45	-43	4
3	-30	-27	6
4	0	2	4
5	30	33	6
6	45	42	6
7	90	87	6

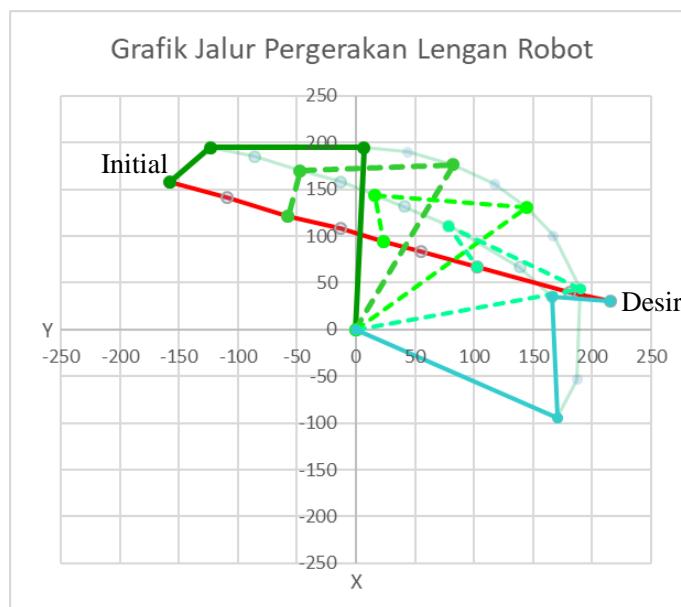
B. Pengujian inverse kinematic “Numerical iteration (Newton Root Finding).

Pengujian ini dilakukan dengan memberikan masukan berupa koordinat yang akan dituju oleh *end of effector* dari robot pada perangkat lunak yang telah dibuat, lalu dibandingkan antara koordinat yang di input terhadap koordinat yang dituju robot, hasil pengujian dapat diamati pada Tabel 4.5.

Koordinat yang di uji adalah bagian pinggir dan tengah dari daerah kerja. Dari hasil pengujian dapat dilihat bahwa semua koordinat yang dituju robot memiliki error, pada bagian pinggir daerah kerja, nilai error bertambah. Hal ini diakibatkan oleh keakuratan gerak robot, kondisi *joint* yang tidak terlalu kokoh

menyebabkan perbedaan panjang *link* pada robot pada setiap masing-masing variasi *joint*. Error pada bagian tengah daerah kerja sebesar 1 cm dan semakin ke daerah pinggir area kerja error bertambah hingga mencapai 2 cm error rata-rata yang diperoleh sebesar 5,61% dan error tertinggi 7,16%. Selain error pada konsidi aktual robot, beberapa perhitungan kinematika mundur tidak menghasilkan solusi dikarenakan iterasi yang tidak konvergen.

Dari setiap hasil iterasi dapat dibuat Grafik untuk melihat jalur pergerakan robot, berikut grafik jalur pergerakan robot saat bergerak dari posisi *joint 1, joint 2, joint 3* secara berurutan ($90^\circ, 90^\circ, 45^\circ$) dengan koordinat tujuan x,y dan sudut pencapit berurutan (215 mm, 30 mm , 0°). Grafik dapat diamati pada Gambar 4.18.



Gambar 4.19 Jalur Gerak *End of Effector* Robot Lengan.

Gambar 4.18 menunjukkan bagaimana prilaku gerak robot saat menggunakan iterasi *newton raphson*, jalur merah pada Gambar 4.18 menunjukkan jalur pergerakan dari *end of effector*, dapat dilihat bahwa jalur membentuk garis lurus mulai dari posisi awal (*initial*) hingga posisi akhir (*desired*). Pola pergerakan lengan robot juga dapat diamati pada Gambar 4.18, garis hijau hingga biru menunjukkan pola pergerakan lengan robot.

Tabel 4.6 Data Hasil Pengujian Gerak Robot

No	Desired (cm)		Initial (°)			Solution euler (cm)		Solution Joint (°)			iteration
	x	y	J1	J2	J3	x	y	J1	J2	J3	
1	38	-17	90	-90	-45	37.5	-17	-7	-38	-44	359
No	Desired (cm)		Initial (°)			Solution euler (cm)		Solution Joint (°)			iteration x
	x	y	J1	J2	J3	x	y	J1	J2	J3	
2	38	-13	90	-90	-45	37.5	-13	4	-51	-43	333
3	38	-9	90	-90	-45	37.5	-9	14	-57	-46	316
4	38	-5	90	-90	-45	37.5	-5	23	-60	-53	302
5	38	-1	90	-90	-45	37.5	-1	30	-57	-62	294
6	38	3	90	-90	-45	37.5	3	36	-51	-74	293
7	38	6	90	-90	-45	37.5	6	38	-38	-89	306
8	38	11	90	-90	-45	0	0	0	0	0	0
9	38	11	90	-90	45	37.5	11	33	-55	111	295
10	38	11	-90	90	45	37.5	11	-9	55	44	324
11	17	-17	90	-90	-45	17.5	-17	-19	-	65	303
12	21	-17	90	-90	-45	21.5	-17	-6	-	42	297
13	25	-17	90	-90	-45	25.5	-17	0	-	22	297
14	28	-17	90	-90	-45	28.5	-17	3	-	8	301
15	32	-17	90	-90	-45	32	-17	2	-83	-9	311
16	17	5	-90	90	45	17.5	5	-8	175	-77	387
17	21	5	-90	90	45	21.5	5	-39	149	-20	304
18	25	5	-90	90	45	25.5	5	-41	131	0	284
19	28	5	-90	90	45	28.5	5	-41	119	12	276
20	32	5	-90	90	45	32	5	-36	99	26	272

Table 4.5 menunjukkan data-data hasil pengujian kinematika mundur dengan metode numerik, pengujian dilakukan dengan memberikan masukan berupa koordinat yang harus di tuju *end of effector*. Dari Tabel 4.5 akan dilihat jumlah iterasi dari masing-masing percobaan dan kesuksesan perhitungan dalam mencari solusi kinematik. Variasi data dilakukan dengan merubah koordinat y dari -170 hingga 110 mm, lokasi awal (*initial*) berada pada koordinat (x = 26,9 y = 159,6) untuk teta1,teta2 dan teta3 sebesar (90°,-90° dan -45°). Pada percobaan pertama iterasi yang dilakukan berhasil dengan jumlah iterasi sebesar 359 iterasi, pada percobaan kedua koordinat tujuan digeser lebih dekat dari koordinat awal, perhitungan berhasil dengan jumlah iterasi turun menjadi 333 iterasi, koordinat tujuan terus digeser mendekati koordinat awal dan jumlah iterasi terus menurun hingga pada percobaan 8 perhitungan tidak menghasilkan solusi (tidak konvergen), posisi initial diubah pada percobaan 9 lalu perhitungan menghasilkan solusi.

C. Pengujian *inverse kinematic analitycal solution*.

Pengujian ini dilakukan dengan memberikan masukan berupa koordinat yang akan dituju oleh *end of effector* dari robot pada perangkat lunak yang telah dibuat, lalu dibandingkan antara koordinat yang di input terhadap koordinat yang dituju robot, hasil pengujian dapat diamati pada Tabel 4.6.

Tabel 4.7 Hasil pengujian kinematik mundur metode analisis.

No	Desired Coordinate			Joint Space			Actual Position	
	X (cm)	Y (cm)	Gripper (°)	Joint 1 (°)	Joint 2 (°)	Joint 3 (°)	X (cm)	Y (cm)
1	32	-27	0	-68	26	42	31,5	-29
2	32	-26	0	-72	37	34	31,5	-28
3	32	-25	0	-74	46	28	31,5	-27
4	32	-24	0	-76	53	22	32	-25
5	32	-23	0	-77	59	17	32	-24,5
6	32	-22	0	-78	65	12	31,5	-23
7	32	-21	0	-78	70	8	31,5	-22

No	Desired Coordinate			Joint Space			Actual Position	
	X (cm)	Y (cm)	Gripper (°)	Joint 1 (°)	Joint 2 (°)	Joint 3 (°)	X (cm)	Y (cm)
8	32	-20	0	-79	75	3	32	-22
9	32	0	0	-41	122	-81	32,5	-2
10	32	10	0	-8	110	-102	32	11
11	32	15	0	6	95	-102	32	15,5
12	32	20	0	21	75	-96	32,5	20,5
13	32	25	0	38	46	-84	32	25,5
14	32	1	0	-37	122	-84	32	-1
15	32	2	0	-34	122	-87	32	1
16	32	3	0	-30	121	-90	32,5	2
17	17	-15	0	-35	135	-99	17	-15,5
18	21	-15	0	-35	136	101	21,2	-15,5
19	25	-15	0	-34	138	-103	25	-15,5
20	29	-15	0	-34	140	-106	28,5	-15,5
21	33	-15	0	-33	142	-108	33	-15,5

Dari hasil pengujian pada Tabel 4.6 semua perhitungan kinematika mundur menghasilkan solusi, kondisi aktual dari robot terhadap kondisi tujuan memiliki sifat error yang hampir sama dengan perhitungan kinematik mundur pada metode iterasi, keakuratan posisi *end of effector* menurun pada daerah pinggir area kerja robot. Perilaku gerak robot dengan metode analisis sangat dipengaruhi dengan pengaturan kecepatan gerak masing-masing *joint* saat kalibrasi awal robot.

D. Pengujian lama proses waktu perhitungan kinematika mundur.

Proses perhitungan kinematika mundur baik metode analisa maupun numerik dilakukan pada mikrokontroller, dilakukan pengukuran lama proses pencarian solusi dengan melihat jeda waktu antara saat proses dimulai dan saat proses berakhir. Dilakukan dua jenis pengujian yaitu pengukuran waktu proses tanpa

memperhatikan lama waktu untuk mengeksekusi masing-masing *joint* serta pengukuran waktu proses dengan memperhatikan masing-masing *joint*.

1. Hasil pengukuran waktu proses tanpa memperhatikan waktu eksekusi *joint*.

Pengukuran dilakukan saat *end of effector* robot bergerak dari posisi rumah ke posisi tujuannya, hasil pengukuran dapat diamati pada Tabel 4.8

Tabel 4.8 Hasil Pengukuran Waktu Perhitungan Kinematik Mundur.

No	Posisi Awal ($X_4^0, Y_4^0, \theta_{x_4}^0$)	Posisi Tujuan ($X_4^0, Y_4^0, \theta_{x_4}^0$)	Numerik		Analisis Waktu (μs)
			Waktu (μs)	Iterasi	
1	269 mm, 159 mm, -45°	250 mm, 20 mm, 0°	2.172.308	728	1764
2	269 mm, 159 mm, -45°	270 mm, -40 mm, -30°	1.260.276	427	1860
3	269 mm, 159 mm, -45°	300 mm, -80 mm, -20°	1.230.448	419	1860
4	269 mm, 159 mm, -45°	300 mm, -150 mm, -20°	1.280.060	433	1860
5	269 mm, 159 mm, -45°	300 mm, -200 mm, -20°	1.338.784	455	1924
6	269 mm, 159 mm, -45°	320 mm, -200 mm, 10°	1.371.156	466	1912
Rata-rata waktu per iterasi metode numerik					2955 μs
Rata-rata waktu perhitungan analisis					1863 μs

Dari hasil pengukuran pada Tabel 4.8, metode numerik memiliki lama waktu perhitungan per iterasi sebesar 2955 μs dengan total waktu perhitungan untuk mencari solusi tergantung dari total iterasi yang dilakukan, dengan total iterasi sebanyak 728 akan memakan waktu sebesar 2,1 detik dan dengan total iterasi sebanyak 419 memakan waktu sebesar 1,2 detik. Pada metode analisis, rata-rata waktu perhitungan yang dibutuhkan untuk mencari solusi kinematik mundur sebesar 1863 μs . Dari hasil pengukuran ini didapatkan kesimpulan bahwa perhitungan analisis memiliki waktu perhitungan yang lebih cepat.

2. Hasil pengukuran waktu proses dengan memperhatikan masing-masing *joint*

Pada pengukuran ini, waktu aksi *joint* juga dihitung. Pengujian ini memiliki konfigurasi *joint* sebagai berikut:

Tabel 4.9 Konfigurasi Masing-Masing Motor Pada *Joint*.

No	<i>Joint Ke</i>	Percepatan (Step/detik)	Kecepatan maksimum (Step/detik)
1	1	1000	700
2	2	1000	200
3	3	1000	200
4	4	1000	200

Tabel 4.10 Hasil Pengukuran Waktu Perhitungan Kinematik Mundur dengan Aksi *Joint*.

No	Posisi Awal ($X_4^0, Y_4^0, \theta_{x_4}^0$)	Posisi Tujuan ($X_4^0, Y_4^0, \theta_{x_4}^0$)	Numerik		Analisis
			Waktu (μs)	Iterasi	Waktu (μs)
1	269 mm, 159 mm, -45°	250 mm, 20 mm, 0°	8.936.964	728	6.095.340
2	269 mm, 159 mm, -45°	270 mm, -40 mm, -30°	6.254.372	427	5.799.884
3	269 mm, 159 mm, -45°	300 mm, -80 mm, -20°	6.619.944	419	5.372.528
4	269 mm, 159 mm, -45°	300 mm, -150 mm, -20°	7.211.332	433	4.941.268
5	269 mm, 159 mm, -45°	300 mm, -200 mm, -20°	7.672.520	455	4.488.576
6	269 mm, 159 mm, -45°	320 mm, -200 mm, 10°	7.709.424	466	4.229.572
Rata-rata waktu metode numerik					7,4 s
Rata-rata waktu metode analisis					5,1 s

Pada Tabel 4.10 waktu rata-rata yang dibutukan *end of effector* untuk berpindah dari satu lokasi ke lokasi lain dengan metode numerik sebesar 7,4 detik sedangkan untuk metode analisis dibutuhkan waktu sebesar 5,1 detik.

4.2.3 Pengujian memindahkan benda berdasarkan ukuran

Pengujian ini dilakukan dengan menghubungkan aplikasi citra yang telah dibuat lalu menghubungkannya dengan robot lengan, sehingga robot lengan mampu mengetahui koordinat benda yang akan dipindahkan. Pengujian dilakukan dengan cara memberikan perintah kepada robot untuk memindahkan benda dari ukuran

kecil ke besar dan sebaliknya. Pada pengujian ini metode kinematik mundur yang digunakan adalah *newton raphson* dikarenakan agar robot dapat memindahkan benda mengikuti sebuah garis.

Objek yang akan dijadikan bahan pengujian berupa balok kayu dengan warna tertentu, pemberian warna bertujuan untuk mempermudah aplikasi memisahkan objek dengan latar area kerja robot. Spesifikasi dari balok kayu dapat diamati pada Tabel 4.7.

Tabel 4.11 Spesifikasi objek penelitian.

No	Nama	Gambar	Ukuran (cm ²) Luas	Panjang (cm)	Lebar (cm)	Tinggi (cm)	Diameter (cm)
1	A		9	3	3	1.6	-
2	B		10.12	4.5	4.5	1.6	-
3	C		17,01	6.3	2.7	1.6	-

4	D		20.79	6.3	3.3	1.6	-
5	E		17,09	-	-	1.6	6.6

1. Pengujian memindahkan benda.

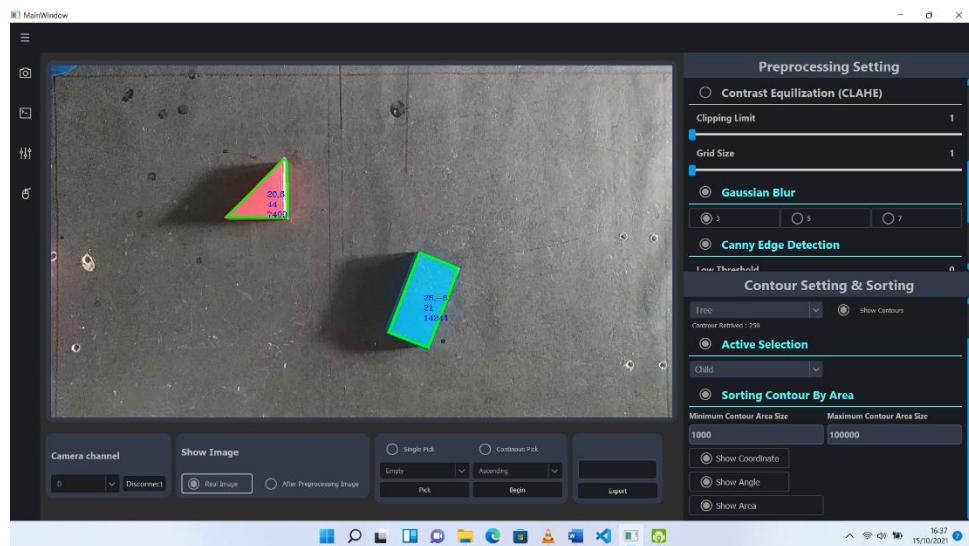
Pada pengujian memindahkan benda akan dilihat peforma robot dalam memindahkan benda, beberapa benda akan diletakan pada area kerja robot lalu akan dipilih benda mana yang akan dipindahkan oleh robot lengan, lalu diamati apakah robot mampu memindahkan benda, ilustrasi percobaan dapat diamati pada Gambar 4.19 dan hasil pengujian dapat diamati pada Tabel 4.8.



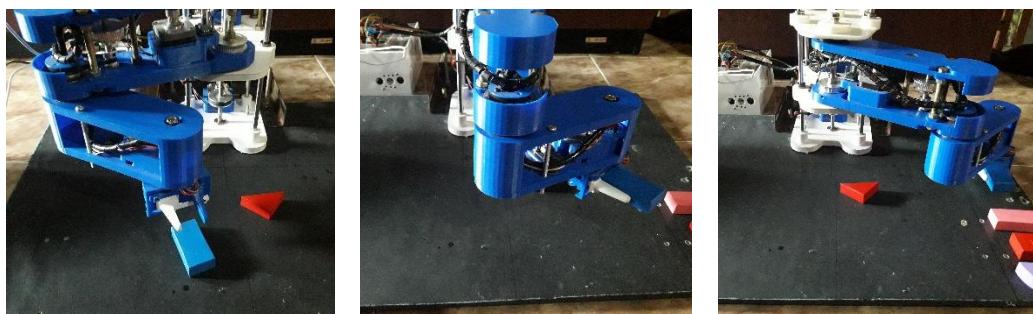
Gambar 4.19 Pengujian memindahkan benda (posisi awal robot).

Pada pengujian memindahkan benda, pengujian pertama dengan koordinat benda yang harus dipindahkan adalah ($x : 26$ cm, $y : -6$ cm, orientasi *gripper* : 21°) dan luas permukaan benda $19,22 \text{ cm}^2$ mendapatkan hasil robot lengan mampu meindahkan benda tersebut.

Tampilan aplikasi saat pemilihan benda yang akan dipindahkan dapat diamati pada Gambar 4.20.



Gambar 4.20 Pengujian pemindahan benda (Tampilan Aplikasi).



Gambar 4.21 Langkah robot memindahkan benda.

Tampilan robot saat memindahkan benda dapat diamati pada Gambar 4.21.

Tabel 4.12 Hasil percobaan memindahkan benda.

Percobaan	Koordinat Benda	Metode Analisis	Metode Newton
1	$x : 20$ cm, $y : -20$ cm	Berhasil	Berhasil

2	x : 20 cm, y : -4 cm	Berhasil	Berhasil
3	x : 20 cm, y : 15 cm	Berhasil	Berhasil

Tabel 4.8 menunjukkan semua pengujian yang dilakukan berhasil, kode program pada aplikasi harus diberikan kompensasi error sebesar 2 cm agar pencapit pada robot lengan tepat berada pada objek hal ini dikarenakan robot memiliki error antara koordinat yang input terhadan lokasi pencapit yang di tuju robot.

2. Pengujian memindahkan benda berdasarkan ukuran.

Pada pengujian memindahkan benda berdasarkan ukuran akan dilihat performa robot dapat memindahkan benda secara urut berdasarkan ukurannya.

Pada Pengujian pertama akan dipindahkan benda secara urut dari benda dengan ukuran terkecil hingga terbesar. Pada area kerja robot terdapat 5 benda dengan spesifikasi dapat diamati pada Tabel 4.18. Robot memindahkan benda secara berurutan A-B-C-E-D, berdasarkan hasil ini robot ternyata telah mampu memindahkan benda berdasarkan ukuran terkecil hingga terbesar pada area kerja robot. Selanjutnya dilakukan pengujian memindahkan benda dari ukuran terbesar hingga terkecil, robot memindahkan benda secara berurutan D-E-C-B-A, berdasarkan hasil ini robot juga telah mampu memindahkan benda berdasarkan ukuran dari terbesar hingga terkecil.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

1. Purwarupa robot scara dengan tiga derajat kebebasan yang telah dibuat mampu memindahkan benda berdasarkan ukuran dan lokasi yang spesifik, berdasarkan hasil percobaan kinematik mundur metode newton, error dari koordinat yang dituju robot sebesar 5,61 % dan untuk metode analisis sebesar 5%.
2. Pada metode newton raphson jalur pergerakan robot berupa garis lurus yang terbentuk dari koordinat awal hingga tujuan, sedangkan metode analisis memiliki jalur pergerakan yang acak, metode newton raphson memiliki iterasi tertinggi sebesar 728 iterasi serta dari 20 percobaan, 1 percobaan tidak menghasilkan solusi. Rata-rata waktu yang dibutuhkan untuk perhitungan sebesar 7,4 detik. semua percobaan pada metode analisis menghasilkan solusi kinematik mundur rata-rata waktu yang dibutuhkan untuk menyelesaikan perhitungan sebesar $1863 \mu\text{s}$ dengan total waktu sebesar 5,1 detik.
3. Metode subtraksi “*Canny Edge Detection*” yang diterapkan dengan bahasa pemograman python mampu mendeteksi dan menghasilkan pinggiran objek dengan akurat, pada pengujian ukuran benda error terbesar adalah 4,8 % dengan rata-rata error 2,6%, sedangkan pada pengujian koordinat benda error terbesar 11,98% dan error rata-rata sebesar 3,6%

5.2 Saran

Pada proses pendekripsi benda, sistem memiliki kendala dimana pada kondisi pencahayaan yang miring terhadap area kerja akan menghasilkan bayangan objek yang cukup pekat dan akan menyebabkan sistem untuk mendekripsi bayangan sebagai objek.

Dalam pembuatan robot dengan sambungan euler, harus diperhatikan massa robot dan kekuatan dari joint robot, untuk menghindari miringnya sambungan yang menyebabkan kurangnya keakuratan dari gerak robot.

DAFTAR PUSTAKA

- [1] E. Utomo, “Robot Application in Shipbuilding Industry,” *studi Literatur Dalam Industri Perkapalan*, pp. 31–38, 2015.
- [2] H. Kalsum, Toibah Umi, Dimas Aulia Trianggana, “Robot Pendeksi Api Menggunakan Bahasa Pemrograman Basic Stamp,” *Jurnal Media Infotama*, vol. Vol.9, no. 1, p. 21, 2013.
- [3] C. s. g. Lee and M. Ziegler, “Geometric Approach in Solving Inverse Kinematics of PUMA Robots,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-20, no. 6, pp. 695–706, Nov. 1984, doi: 10.1109/TAES.1984.310452.
- [4] H. A. F. Almurib, H. F. Al-Qirimli, and N. Kumar, “A review of application industrial robotic design,” in *2011 Ninth International Conference on ICT and Knowledge Engineering*, Jan. 2012, pp. 105–112. doi: 10.1109/ICTKE.2012.6152387.
- [5] G. E. Setyawan, “Implementasi Robot Lengan Pemindah Barang 3 DOF Menggunakan Metode Inverse Kinematics,” *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya*, vol. 2, no. April, pp. 2810–2816, 2018.
- [6] J. F. Reyes and L. E. Chiang, “Image-to-space path planning for a SCARA manipulator with single color camera,” *Robotica*, vol. 21, no. 3, pp. 245–254, Jun. 2003, doi: 10.1017/S0263574702004745.
- [7] S. Kucuk, “The Inverse Kinematics Solutions of Fundamental Robot Manipulators with Offset Wrist,” 2005.
- [8] N. F. Zulfardi, D. I. Saputra, and A. D. Ahkam, “Aplikasi Deteksi Benda Menggunakan Metode Image Subtraction Sebagai Masukan Koordinat Pada Robot Lengan 3 DOF,” no. September, pp. 30–37, 2019.
- [9] S. Shirafuji and J. Ota, “Kinematic Synthesis of a Serial Robotic Manipulator by Using Generalized Differential Inverse Kinematics,” *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 1047–1054, 2019, doi: 10.1109/TRO.2019.2907810.
- [10] M. M.Richard, Z. Li, and N. shankar Shastri, *A Mathematical Introduction to Robot Manipulator*. California: CRC Press, 1994.

- [11] R. P.Paul, *Robot Manipulator : Mathematics, Programing, and Control*, 6th ed. United States of America: The Massachusetts Institute of Technology, 1984.
- [12] IEEE Robotics and Automation Society and Institute of Electrical and Electronics Engineers, *2014 IEEE International Conference on Mechatronics and Automation : IEEE ICMA 2014 : August 3-6, 2014, Tianjin, China*.
- [13] Ben-Israel and Adi, “A Newton-Raphson method for the solution of systems of equations,” *Journal of Mathematical analysis and applications*, vol. 15, pp. 243–252, 1966.
- [14] S. Neppalli, M. A. Csencsits, B. A. Jones, and I. D. Walker, “Closed-Form Inverse Kinematics for Continuum Manipulators,” *Advanced Robotics*, vol. 23, no. 15, pp. 2077–2091, Jan. 2009, doi: 10.1163/016918609X12529299964101.
- [15] A. Hemami, “A more general closed-form solution to the inverse kinematics of mechanical arms,” *Advanced Robotics*, vol. 2, no. 4, pp. 315–325, Jan. 1987, doi: 10.1163/156855388X00010.
- [16] K. Tchoń, “Optimal extended Jacobian inverse kinematics algorithms for robotic manipulators,” *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1440–1445, 2008, doi: 10.1109/TRO.2008.2006240.
- [17] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, “Contour Detection and Hierarchical Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, May 2011, doi: 10.1109/TPAMI.2010.161.
- [18] Y. M. Mustafah, R. Noor, H. Hasbi, and A. W. Azma, “Stereo vision images processing for real-time object distance and size measurements,” *2012 International Conference on Computer and Communication Engineering, ICCCE 2012*, no. July, pp. 659–663, 2012, doi: 10.1109/ICCCE.2012.6271270.
- [19] A. D. Ker, “Steganalysis of LSB matching in grayscale images,” *IEEE Signal Processing Letters*, vol. 12, no. 6, pp. 441–444, Jun. 2005, doi: 10.1109/LSP.2005.847889.

- [20] Afif Aulia Rahman, Muhammad Rivai, and Tasripan, “Sistem Otomatisasi Pelacakan Objek Astronomi Menggunakan Teleskop Berdasarkan Stellarium,” *JURNAL TEKNIK ITS*, vol. 6, 2017.
- [21] C. Stephen J, *Electric Machinery Fundamental*. New York: McGraw-Hill, 2005.
- [22] Arduino.CC, “Getting Started with Arduino MEGA2560,” *Arduino*, 2020. <https://www.arduino.cc/en/Guide/ArduinoMega2560> (accessed Sep. 19, 2020).
- [23] et al. Vilariño, Fernando, “Universal Power Supply,” vol. 1, no. 12, 2013, [Online]. Available: <http://www.medscape.com/viewarticle/405488>
- [24] G. van Rossum and Python Team, “Python Tutorial,” 2018. bugs.python.org (accessed Sep. 02, 2020).
- [25] OpenCV, “opencv.org/about/,” 2020. <https://opencv.org/about/> (accessed Sep. 20, 2020).

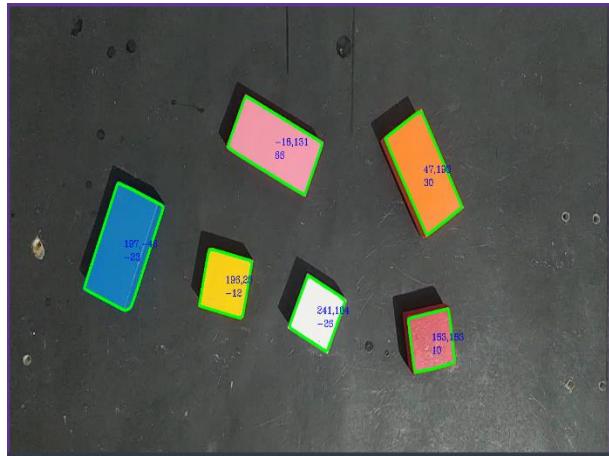
LAMPIRAN



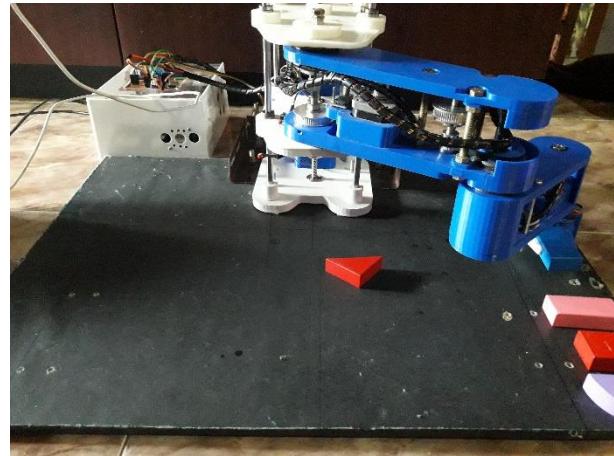
Gambar 1 Pengujian Robot



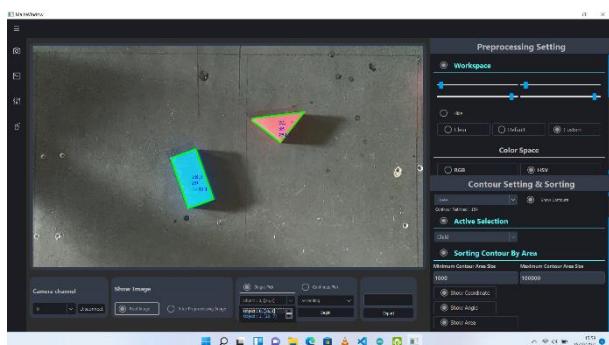
Gambar 2 Pengujian pengambilan benda



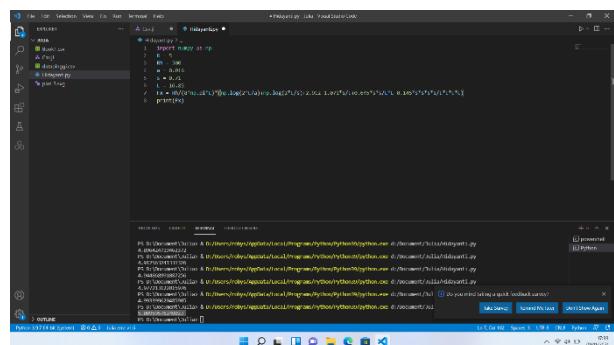
Gambar 3 Deteksi benda oleh komputer



Gambar 4 Area kerja robot



Gambar 5 Tampilan Aplikasi



Gambar 6 Tampilan Program

Kode Program Arduino

```
#include <VarSpeedServo.h>
#include <AccelStepper.h>
#include <MultiStepper.h>
#include <BasicLinearAlgebra.h>
using namespace BLA;
#define limit1 4
#define limit2 5
#define limit3 6
#define limit4 7
AccelStepper stepper1(AccelStepper::HALF4WIRE,22,24,26,28);
AccelStepper stepper2(AccelStepper::HALF4WIRE,23,25,27,29);
AccelStepper stepper3(AccelStepper::HALF4WIRE,31,33,35,37);
AccelStepper stepper4(AccelStepper::FULL4WIRE,41,45,39,43);
VarSpeedServo Gripper;
BLA::Matrix<3, 1> Desired;
BLA::Matrix<3, 1> New_teta;
float error = 0.001;
String masuk,keluar;
int mode = 0;
int motor1[5];
int motor2[5];
int motor3[5];
int motor4[5];
int Prev_ite;
int Speed1 = 700;
int Speed2 = 200;
int Speed3 = 200;
int Speed4 = 200;
float pi = 3.14159265359;
```

```
unsigned long counter,previouscounter;
long positions[4];
MultiStepper steppers;
void setup() {
  Gripper.attach(9);
  Serial.begin(57600);
  pinMode(limit1, INPUT_PULLUP);
  pinMode(limit2, INPUT_PULLUP);
  pinMode(limit3, INPUT_PULLUP);
  pinMode(limit4, INPUT_PULLUP);
  stepper1.setMaxSpeed(700);
  stepper1.setAcceleration(1000);
  stepper2.setMaxSpeed(200);
  stepper2.setAcceleration(1000);
  stepper3.setMaxSpeed(200);
  stepper3.setAcceleration(1000);
  stepper4.setMaxSpeed(200);
  stepper4.setAcceleration(1000);
  steppers.addStepper(stepper1);
  steppers.addStepper(stepper2);
  steppers.addStepper(stepper3);
  steppers.addStepper(stepper4);
}

void loop() {
  if (Serial.available()>0){masuk = Serial.readString();}
  mode = (getValue(masuk,';',0)).toInt();
  if (mode==1){
    Calibration(masuk);
    masuk = "";
    Serial.read();
  }
}
```

```
}

else if (mode==2){

    Forward(masuk);

    masuk="";

    Serial.read();

}

else if(mode==3){

    single_pick(masuk);

    masuk="";

    Serial.read();

}

else if(mode==4){

    Newton(masuk);

    masuk="";

    Serial.read();

}

else if(mode==5){

    Single_Forward(masuk);

    masuk="";

    Serial.read();

}

else if(mode==6){

    float x = getValue(masuk,',',1).toFloat();

    float y = getValue(masuk,',',2).toFloat();

    float ee = getValue(masuk,',',3).toFloat();

    float Grip = getValue(masuk,',',4).toFloat();

    single_pick_newton(x,y,ee,Grip);

    masuk="";

    Serial.read();

}

counter = millis();
```

```
if ((counter-previouscounter)>=2000){  
    updateProgram();  
    previouscounter = millis();}  
}  
  
void Calibration(String masuk){  
    motor1[0] = (getValue(masuk,';',1)).toInt();  
    motor1[1] = (getValue(masuk,';',2)).toInt();  
    motor1[2] = stepper1.currentPosition();  
    motor1[3] = (getValue(masuk,';',3)).toInt();  
    motor1[4] = (getValue(masuk,';',4)).toInt();  
    motor2[0] = (getValue(masuk,';',5)).toInt();  
    motor2[1] = (getValue(masuk,';',6)).toInt();  
    motor2[2] = stepper2.currentPosition();  
    motor2[3] = (getValue(masuk,';',7)).toInt();  
    motor2[4] = (getValue(masuk,';',8)).toInt();  
    motor3[0] = (getValue(masuk,';',9)).toInt();  
    motor3[1] = (getValue(masuk,';',10)).toInt();  
    motor3[2] = stepper3.currentPosition();  
    motor3[3] = (getValue(masuk,';',11)).toInt();  
    motor3[4] = (getValue(masuk,';',12)).toInt();  
    motor4[0] = (getValue(masuk,';',13)).toInt();  
    motor4[1] = (getValue(masuk,';',14)).toInt();  
    motor4[2] = stepper4.currentPosition();  
    motor4[3] = (getValue(masuk,';',15)).toInt();  
    motor4[4] = (getValue(masuk,';',16)).toInt();  
    stepper1.setMaxSpeed(motor1[0]);  
    stepper1.setAcceleration(motor1[1]);  
    stepper2.setMaxSpeed(motor2[0]);  
    stepper2.setAcceleration(motor2[1]);  
    stepper3.setMaxSpeed(motor3[0]);  
    stepper3.setAcceleration(motor3[1]);
```

```
stepper4.setMaxSpeed(motor4[0]);
stepper4.setAcceleration(motor4[1]);

//=====
=====

if(motor1[3] == 1){
    while(!digitalRead(limit1)){
        stepper1.setSpeed(1000);stepper1.runSpeed();
    }
    stepper1.setCurrentPosition(0);
    stepper1.moveTo(-9500);
    stepper1.runToPosition();
    stepper1.setCurrentPosition(0);
    stepper1.moveTo(1000);
    stepper1.runToPosition();
}
else if (motor1[3] == 2){
    stepper1.moveTo(stepper1.currentPosition()+(motor1[4]/0.013333));
    stepper1.runToPosition();
    stepper1.setCurrentPosition(0);
}

//=====
=====

if(motor2[3] == 1){
    while(!digitalRead(limit2)){
        stepper2.setSpeed(-100);stepper2.runSpeed();
    }
    stepper2.setCurrentPosition(-410);
    stepper2.moveTo(0);
    stepper2.runToPosition();
```

```
}

else if (motor2[3] == 2){

    stepper2.moveTo(stepper3.currentPosition()+(motor2[4]/0.2));
    stepper2.runToPosition();
    stepper2.setCurrentPosition(0);
}

//=====================================================================
=====

if(motor3[3] == 1){

    while(!digitalRead(limit2)){

        stepper2.setSpeed(-100);stepper2.runSpeed();

    }

    stepper2.setCurrentPosition(-400);

    stepper2.moveTo(-405);

    stepper2.runToPosition();

    while(!digitalRead(limit3)){

        stepper3.setSpeed(100);stepper3.runSpeed();

    }

    stepper3.setCurrentPosition(850);

    stepper2.moveTo(0);

    stepper2.runToPosition();

    stepper3.moveTo(0);

    stepper3.runToPosition();

}

else if (motor3[3] == 2){

    stepper3.moveTo(stepper3.currentPosition()+(motor3[4]/0.2));
    stepper3.runToPosition();
    stepper3.setCurrentPosition(0);
}
```

```

//=====
=====

if(motor4[3] == 2){
    stepper4.moveTo(stepper4.currentPosition()+(motor4[4]/0.142));
    stepper4.runToPosition();
    stepper4.setCurrentPosition(0);
}

}

void updateProgram(){
    int S1s = stepper1.maxSpeed();
    int S1p = stepper1.currentPosition()*0.013333;
    int S2s = stepper2.maxSpeed();
    int S2p = stepper2.currentPosition()*0.2;
    int S3s = stepper3.maxSpeed();
    int S3p = stepper3.currentPosition()*0.2;
    int S4s = stepper4.maxSpeed();
    int S4p = stepper4.currentPosition()*0.142;
    int gr = map(Gripper.read(),180,70,20,50);
    String data =
String(mode)+"."+String(S1s)+";"+String(S1p)+";"+String(S2s)+";"+String(S2p)+";"+String(S3
s)+";"+String(S3p)+";"+String(S4s)+";"+String(S4p)+";"+String(gr);
    Serial.println(data);
}

void Forward(String masuk){
    positions[0] = (getValue(masuk,',',1)).toInt()/0.013333;
    positions[1] = (getValue(masuk,',',2)).toInt()/0.2;
    positions[2] = (getValue(masuk,',',3)).toInt()/0.2;
    positions[3] = (getValue(masuk,',',4)).toInt()/0.142;
    steppers.moveTo(positions);
}

```

```
int grip = (getValue(masuk,';',5)).toInt();
grip = map(grip,20,50,180,70);
while(stepper1.run() || stepper2.run() || stepper3.run() || stepper4.run()){
    stepper1.run();
    stepper2.run();
    stepper3.run();
    stepper4.run();
}
Gripper.write(grip,30,true);
}

void Single_Forward(String masuk){
    int Part = (getValue(masuk,';',1)).toInt();
    switch (Part){
        case 1:
            stepper1.moveTo((getValue(masuk,';',2)).toInt()/0.013333);
            stepper1.runToPosition();
            break;
        case 2:
            stepper2.moveTo((getValue(masuk,';',2)).toInt()/0.2);
            stepper2.runToPosition();
            break;
        case 3:
            stepper3.moveTo((getValue(masuk,';',2)).toInt()/0.2);
            stepper3.runToPosition();
            break;
        case 4:
            stepper4.moveTo((getValue(masuk,';',2)).toInt()/0.142);
            stepper4.runToPosition();
            break;
        case 5:
            int grip = (getValue(masuk,';',2)).toInt();
```

```

grip = map(grip,20,50,180,70);
Gripper.write(grip,30,true);
break;
}
}

void Newton(String Masuk){
float tt1 = stepper2.currentPosition()*0.2;
float tt2 = stepper3.currentPosition()*0.2;
float tt3 = stepper4.currentPosition()*0.142;
BLA::Matrix<3, 1> Teta = {tt1,tt2,tt3};
BLA::Matrix<3, 1> Desired;
BLA::Matrix<3, 1> New_teta;
Desired =
{getValue(Masuk,';',1).toFloat(),getValue(Masuk,';',2).toFloat(),getValue(Masuk,';',3).toFloat()};
int ite = 0;
Prev_ite = 0;
while(true){
ite++;
BLA::Matrix<4, 4> Forward = {cos((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi),-
1*sin((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi),0,50*cos((Teta(0, 0)+Teta(1, 0)+Teta(2,
0))/180.0*pi)+130*cos((Teta(0, 0)+Teta(1, 0))/180.0*pi)+195*cos(Teta(0,
0)/180.0*pi)+104,sin((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi),cos((Teta(0, 0)+Teta(1,
0)+Teta(2, 0))/180.0*pi),0,50*sin((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi)+130*sin((Teta(0,
0)+Teta(1, 0))/180.0*pi)+195*sin(Teta(0, 0)/180.0*pi),0,0,1,0,0,0,0,1};
BLA::Matrix<3, 1> FWRD = {Forward(0, 3),Forward(1, 3),Forward(0, 0)};
BLA::Matrix<3, 3> Jacobian = {-50*sin((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi)-
130*sin((Teta(0, 0)+Teta(1, 0))/180.0*pi)-195*sin(Teta(0, 0)/180.0*pi),-50*sin((Teta(0,
0)+Teta(1, 0)+Teta(2, 0))/180.0*pi)-130*sin((Teta(0, 0)+Teta(1, 0))/180.0*pi),-50*sin((Teta(0,
0)+Teta(1, 0)+Teta(2, 0))/180.0*pi)+130*cos((Teta(0, 0)+Teta(1, 0))/180.0*pi)+195*cos(Teta(0,
0)/180.0*pi),50*cos((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi)+130*cos((Teta(0, 0)+Teta(1,
0))/180.0*pi)+195*cos(Teta(0, 0)/180.0*pi),50*cos((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi)+130*cos((Teta(0, 0)+Teta(1, 0))/180.0*pi)+195*cos(Teta(0, 0)/180.0*pi)};

```

```

0))/180.0*pi),50*cos((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi),-1*sin((Teta(0, 0)+Teta(1,
0)+Teta(2, 0))/180.0*pi),-1*sin((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi),-1*sin((Teta(0,
0)+Teta(1, 0)+Teta(2, 0))/180.0*pi)}};

BLA::Matrix<3, 3> Inv_Jacobian = Jacobian;
bool is_nonsingular = Invert(Inv_Jacobian);
New_teta = Teta-(-Inv_Jacobian*(Desired-FWRD));
BLA::Matrix<3, 1> Error = {abs(New_teta(0,0)-Teta(0,0)),abs(New_teta(1,0)-
Teta(1,0)),abs(New_teta(2,0)-Teta(2,0))};
if(Error(0,0)<=error && Error(1,0)<=error && Error(2,0)<=error){
break;}
Serial<<"Current Teta : "<<New_teta<<"\n";
Move(New_teta(0,0),New_teta(1,0),New_teta(2,0));
Teta = New_teta;
}
}

void single_pick(String masuk){
positions[0] = stepper1.currentPosition();
positions[1] = (getValue(masuk,';',2)).toInt()/0.2;
positions[2] = (getValue(masuk,';',3)).toInt()/0.2;
positions[3] = (getValue(masuk,';',4)).toInt()/0.142;
steppers.moveTo(positions);
int grip = 30 ;
grip = map(grip,20,50,180,70);
steppers.runSpeedToPosition();
Gripper.write(70,50,true);
stepper1.moveTo(0);
stepper1.runToPosition();
Gripper.write(grip,50,true);
stepper1.moveTo(5/0.013333);
stepper1.runToPosition();
positions[0] = 10/0.013333;
}

```

```

positions[1] = 45/0.2;
positions[2] = 80/0.2;
positions[3] = -125/0.142;
steppers.moveTo(positions);
steppers.runSpeedToPosition();
stepper1.moveTo(0);
stepper1.runToPosition();
Gripper.write(70,50,true);
stepper1.moveTo(10/0.013333);
stepper1.runToPosition();
positions[0] = 30/0.013333;
positions[1] = 90/0.2;
positions[2] = -90/0.2;
positions[3] = -45/0.142;
steppers.moveTo(positions);
steppers.runSpeedToPosition();
}

void Move(float teta2,float teta3,float teta4){
    positions[0]=stepper1.currentPosition();
    positions[1]=teta2/0.2;
    positions[2]=teta3/0.2;
    positions[3]=teta4/0.142;
    steppers.moveTo(positions);
    steppers.runSpeedToPosition();
}

void Newton2(float x,float y,float Grip){
    float tt1 = stepper2.currentPosition()*0.2;
    float tt2 = stepper3.currentPosition()*0.2;
    float tt3 = stepper4.currentPosition()*0.142;
    BLA::Matrix<3, 1> Teta = {tt1,tt2,tt3};
    BLA::Matrix<3, 1> Desired;
}

```

```

BLA::Matrix<3, 1> New_teta;
Desired = {x,y,Grip};
int ite = 0;
Prev_ite = 0;
while(true){
    ite++;
    BLA::Matrix<4, 4> Forward = {cos((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi),-
1*sin((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi),0,50*cos((Teta(0, 0)+Teta(1, 0)+Teta(2,
0))/180.0*pi)+130*cos((Teta(0, 0)+Teta(1, 0))/180.0*pi)+195*cos(Teta(0,
0)/180.0*pi)+104,sin((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi),cos((Teta(0, 0)+Teta(1,
0)+Teta(2, 0))/180.0*pi),0,50*sin((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi)+130*sin((Teta(0,
0)+Teta(1, 0))/180.0*pi)+195*sin(Teta(0, 0)/180.0*pi),0,0,1,0,0,0,0,1};

    BLA::Matrix<3, 1> FWRD = {Forward(0, 3),Forward(1, 3),Forward(0, 0)};

    BLA::Matrix<3, 3> Jacobian = {-50*sin((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi)-
130*sin((Teta(0, 0)+Teta(1, 0))/180.0*pi)-195*sin(Teta(0, 0)/180.0*pi),-50*sin((Teta(0,
0)+Teta(1, 0)+Teta(2, 0))/180.0*pi)-130*sin((Teta(0, 0)+Teta(1, 0))/180.0*pi),-50*sin((Teta(0,
0)+Teta(1, 0)+Teta(2, 0))/180.0*pi),50*cos((Teta(0, 0)+Teta(1, 0)+Teta(2,
0))/180.0*pi)+130*cos((Teta(0, 0)+Teta(1, 0))/180.0*pi)+195*cos(Teta(0,
0)/180.0*pi),50*cos((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi)+130*cos((Teta(0, 0)+Teta(1,
0))/180.0*pi),50*cos((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi),-1*sin((Teta(0, 0)+Teta(1,
0)+Teta(2, 0))/180.0*pi),-1*sin((Teta(0, 0)+Teta(1, 0)+Teta(2, 0))/180.0*pi),-1*sin((Teta(0,
0)+Teta(1, 0)+Teta(2, 0))/180.0*pi)};

    BLA::Matrix<3, 3> Inv_Jacobian = Jacobian;
    bool is_nonsingular = Invert(Inv_Jacobian);
    New_teta = Teta-(-Inv_Jacobian*(Desired-FWRD));
    BLA::Matrix<3, 1> Error = {abs(New_teta(0,0)-Teta(0,0)),abs(New_teta(1,0)-
Teta(1,0)),abs(New_teta(2,0)-Teta(2,0))};

    if(Error(0,0)<=error && Error(1,0)<=error && Error(2,0)<=error){
        break;
    }
    Serial<<"Current Teta : "<<New_teta<<'\n';
    Move(New_teta(0,0),New_teta(1,0),New_teta(2,0));
}

```

```

Teta = New_teta;
}

}

void single_pick_newton(float x,float y,float ee,float grip){
    stepper1.moveTo(60/0.013333);
    stepper1.runToPosition();
    Newton2(x,y,ee);
    Gripper.write(70,50,true);
    stepper1.moveTo(0);
    stepper1.runToPosition();
    grip = map(grip,20,50,180,70);
    Gripper.write(grip,50,true);
    stepper1.moveTo(30/0.013333);
    stepper1.runToPosition();
    Newton2(330,250,1);
    stepper1.moveTo(0);
    stepper1.runToPosition();
    Gripper.write(70,50,true);
    stepper1.moveTo(30/0.013333);
    stepper1.runToPosition();
    Move(90,-90,-45);
}

String getValue(String data, char separator, int index)
{
    int found = 0;
    int strIndex[] = {0, -1};
    int maxIndex = data.length()-1;

    for(int i=0; i<=maxIndex && found<=index; i++){
        if(data.charAt(i)==separator || i==maxIndex){
            found++;
        }
    }
}

```

```
strIndex[0] = strIndex[1]+1;  
strIndex[1] = (i == maxIndex) ? i+1 : i;  
}  
}  
return found>index ? data.substring(strIndex[0], strIndex[1]) : "";  
}
```