BAB III
DRAWING ROBOT

2.1. Tujuan

1. Praktikum mampu mengidentifikasi komponen utama yang digunakan dalam *Drawing Robot* dan memahami fungsinya.
2. Praktikan mampu menguasai prinsip kerja serta kontrol pergerakan robot menggunakan motor DC dan servo.
3. Praktikan menganalisis dan mengontrol pergerakan *Drawing Robot* untuk menghasilkan pola gambar yang diinginkan.
4. Praktikan dapat mengembangkan pemahaman tentang cara mengintegrasikan Arduino Nano 33 IoT dengan motor dan sensor.
5. raktikan mampu mengaplikasikan konsep kontrol sistem tertutup (*closed-loop control*) dalam pergerakan robot.

2.2. Alat dan Bahan

2.2.1. PC/Laptop

Laptop adalah komputer pribadi yang dapat dipindahkan dan dibawa dengan mudah sehingga dapat digunakan di banyak tempat. Mayoritas laptop mempunyai fitur yang sama dengan komputer, seperti mampu menjalankan perangkat lunak dan mengelola berkas. Namun, laptop cenderung lebih mahal daripada komputer desktop.

Karena laptop dirancang terutama untuk aspek *port* abilitas, maka terdapat beberapa perbedaan penting antara laptop dan komputer desktop. Sebuah laptop dirancang mempunyai semua fitur komputer yang terpasang built-in, seperti monitor, *keyboard*, *touchpad* (menggantikan *mouse*), dan *speaker*. Dengan demikian sebuah laptop standar dapat berfungsi dengan normal tanpa tambahan perangkat lainnya. Sebuah laptop juga lebih cepat untuk dipakai, dan jumlah kabel yang terhubung ke laptop lebih sedikit.

Selain itu, Anda dapat menghubungkan perangkat lainnya ke laptop seperti *mouse*, monitor yang lebih besar, dan peripheral lainnya. Setelah dihubungkan dengan peripheral lainnya, maka laptop dapat berfungsi sebagai komputer *desktop* dengan satu perbedaan utama di antara keduanya yaitu laptop dapat Anda bawa kemanapun setelah peripheral tersebut dicopot dari laptop.

(Sumber: *https://edu.gcfglobal.org/en/tr_id-computer-basics/laptop/1/*)

2.2.2. Arduino Nano 33 IoT

Arduino Nano 33 IoT merupakan *development board* dari Arduino yang didesain untuk aplikasi *Internet of Things* (IoT) dengan mengusung mikrokontroler Arm® Cortex®-M0 32-bit SAMD21, yang dikenal memiliki konsumsi daya rendah sehingga cocok untuk perangkat hemat energi. *Board* ini dilengkapi modul WiFi dan Bluetooth NINA-W10 dari u-blox yang beroperasi pada frekuensi 2.4GHz, memungkinkan konektivitas nirkabel stabil untuk berbagai kebutuhan IoT. Selain itu, Arduino Nano 33 IoT juga dibekali sensor IMU (*Inertial Measurement Unit*) 6 sumbu yang mendukung pengukuran gerak, getaran, dan orientasi, menjadikannya ideal untuk aplikasi seperti sistem deteksi gempa, navigasi *drone*, atau pemantauan

posisi. *Board* ini juga kompatibel dengan berbagai platform IoT populer seperti Arduino IoT Cloud, Blynk, AWS IoT Core, Azure, Firebase, dan lainnya, sehingga memudahkan pengembangan prototipe maupun proyek skala besar. Dengan kombinasi fitur konektivitas, sensor canggih, dan efisiensi daya, Arduino Nano 33 IoT menjadi pilihan serbaguna untuk inovasi di bidang IoT.

(Sumber: *https://digiwarestore.com/id/internet-of-things-iot/arduino-nano-33-iot-442421.html*)

2.2.3. DC Motor

DC Motor merupakan perangkat yang dapat mengubah energi listrik menjadi energi kinetik atau gerakan. DC motor memiliki dua terminal atau kabel, yaitu *power* dan *ground*. Motor ini memerlukan tegangan DC agar dapat bergerak. Biasanya digunakan pada perangkat elektronik yang menggunakan sumber listrik DC seperti kipas pendingin komputer dan mobil *remote control*. DC motor menghasilkan sejumlah putaran per menit atau biasa dikenal dengan istilah RPM (*revolutions per minute*) dan dapat berputar searah jarum jam maupun berlawanan arah jarum jam, tergantung polaritas listrik yang diberikan.

DC motor memberikan kecepatan rotasi sekitar 3000 - 8000 RPM dengan tegangan operasional dari 1.5 - 24V. Apabila tegangan yang diberikan lebih rendah dari tegangan operasional, maka akan memperlambat rotasi motor. Jika tegangan lebih tinggi dari tegangan operasional, rotasi motor akan menjadi lebih cepat. Namun, jika tegangan yang diberikan ke motor turun menjadi <50% tegangan operasional, motor tidak akan berputar atau terhenti. Sebaliknya, jika tegangan melebihi >30% tegangan operasional, motor akan sangat panas dan berpotensi rusak.

Saat DC motor berputar tanpa beban, hanya sedikit arus listrik yang digunakan. Namun, jika diberikan beban, jumlah arus yang digunakan akan meningkat hingga ratusan atau ribuan persen, tergantung jenis beban. Oleh karena itu, produsen DC motor biasanya mencantumkan *stall current*, yaitu arus maksimum yang terukur saat poros motor berhenti karena beban berlebihan.

(Sumber: *https://www.arduinoindonesia.id/2022/10/pengertian-dan-prinsip-kerja-motor-dc.html*)

2.2.4. Servo Motor

Servo motor atau motor servo adalah perangkat elektromekanis yang dirancang menggunakan sistem kontrol jenis loop tertutup (servo) sebagai penggerak dalam sebuah rangkaian yang menghasilkan torsi dan kecepatan yang berdasarkan arus listrik dan tegangan yang ada. Sederhananya motor servo ini perangkat listrik mandiri yang dapat mendorong, memutar objek dengan presisi tinggi. Jika ingin memutar suatu objek pada beberapa sudut atau jarak tertentu, maka bisa menggunakan motor servo. Servo motor merupakan perangkat yang terdiri atas komponen motor dan juga sensor. Gabungan kedua perangkat ini memungkinkan pengguna untuk mengatur beberapa komponen proses industri seperti mengatur kecepatan laju mesin, kemiringan, sudut, dan lain sebagainya. Terlebih semua ini dapat dilakukan secara presisi sehingga pengguna tidak perlu khawatir akan adanya kesalahan atau error.

(Sumber: https://misel.co.id/mengenal-lebih-lanjut-apa-itu-servo-motor/)

2.2.5. PWM (Pulse Width Modulation) Micro*controller*s

Pulse Width Modulation (PWM) secara umum merupakan sebuah cara untuk memanipulasi lebar sinyal pulsa dari gelombang elektronik dalam suatu perioda, untuk mendapatkan nilai tegangan rata-rata yang berbeda.

Pada dasarnya PWM biasanya digunakan untuk telekomunikasi (modulasi data), penguat (*amplifier*), pengatur daya dan juga sebagai regulator tegangan. Pada penelitian ini PWM digunakan untuk mengendalikan kecepatan motor. PWM merupakan sinyal analog yang memiliki *amplitude* dan frekuensi dasar tetap, yang mengalami perubahan hanya pada lebar pulsa dan memiliki *duty cycle* bervariasi antara 0% sampai 100% sesuai dengan kecepatan yang diinginkan, semakin besar persentasi maka semakin cepat perputaran motor tersebut. PWM digunakan untuk menghasilkan tegangan keluaran yang bervariasi, mulai dari tegangan 0 volt sampai dengan tegangan maksimal, sifat kenaikan tegangan adalah linier, menaikkan dan menurunkan lebar pulsa, dapat digunakan untuk mengatur aliran arus yang akan mengalir pada motor.

Pulse witdh modulation (PWM) merupakan salah satu teknik yang digunakan untuk mengendalikan kekuatan (*power*) biasanya mengatur berapa besar

tegangan yang akan digunakan dengan mengirim isyarat atau pulsa dalam bentuk sinyal. PWM pada penelitian ini akan digunakan untuk mengendalikan *duty cycle* pada sinyal yang akan digunakan untuk menggerakan motor sehingga kecepatan motor dapat dikendalikan.

(Sumber: *https://www.radius.co.id/apa-sih-yang-dimaksud-pulse-width-modulation-atau-pwm-pada-sistem-kontrol-dan-instrumentasi-industri/*)

2.2.6. Battery & Holder

Baterai dan tempat baterai (*battery* & *holder*) adalah komponen penting dalam sistem penyediaan daya. Baterai adalah sumber energi listrik yang dapat terdiri dari berbagai jenis, seperti baterai *lithium-ion*, *nickel-metal hydride*, atau *lead-acid*. Sementara itu, tempat baterai, yang juga dikenal sebagai kotak baterai atau paket baterai, adalah perangkat yang digunakan untuk menghubungkan satu atau lebih baterai menjadi satu kesatuan agar dapat menyediakan daya listrik.

Tempat baterai biasanya terdiri dari beberapa bagian utama: paket baterai, sirkuit kontrol, sirkuit perlindungan, dan konektor. Paket baterai menyusun baterai-baterai secara teratur untuk menghasilkan tegangan dan kapasitas yang sesuai. Sirkuit kontrol dan perlindungan berfungsi untuk memantau kondisi kerja baterai serta melindungi baterai dari masalah seperti *overcharge* (pengisian berlebih) atau *over-discharge* (pengosongan berlebih). Konektor, di sisi lain, digunakan untuk menghubungkan tempat baterai dengan perangkat lain.

Tempat baterai dirancang dengan komponen seperti kawat pegas melingkar dan tab datar untuk memastikan kontak yang baik dengan sel baterai, sehingga tercipta sambungan listrik yang aman. Koneksi eksternal pada tempat baterai dapat dibuat melalui berbagai cara, seperti pin, lug solder, kaki permukaan (*surface mount feet*), atau kabel penghubung.

Pada baterai kering (*dry cells*), kontak listrik terjadi langsung dengan terminal baterai. Sedangkan pada baterai basah (*wet cells*), kabel biasanya digunakan untuk menghubungkan terminal baterai dengan sistem lainnya. Dengan demikian, baterai dan tempat baterai bekerja sama untuk menyediakan daya yang stabil dan aman bagi berbagai perangkat.

(Sumber: *https://www.bituoelec.com/new/the-complete-guide-to-battery-holders/*)

2.2.7. *Drawing Robot*

Robot gambar (*drawing robot*) adalah sebuah sistem otomatis yang dirancang untuk meniru pola atau gambar berdasarkan *input* visual atau data digital. Dalam proyek ini, robot gambar dibangun menggunakan komponen dari Arduino Engineering Kit Rev 2, yang mencakup papan mikrokontroler Arduino Nano 33 IoT, motor DC dengan *encoder*, motor servo mikro, dan modul Motor Carrier. Prinsip dasar dari robot ini adalah menggabungkan teknologi pengolahan citra (*image processing*) dengan kontrol gerakan presisi untuk mereproduksi gambar pada permukaan seperti *whiteboard*.

Papan Arduino Nano 33 IoT bertindak sebagai otak dari robot, yang mengontrol semua operasi, termasuk navigasi dan pengangkatan pena. Pengolahan citra dilakukan menggunakan MATLAB, yang mendukung integrasi dengan perangkat keras Arduino melalui MATLAB Sup*port* Package for Arduino Hardware. Gambar yang diambil oleh webcam diproses untuk mengekstraksi jejak garis (*line traces*), yang kemudian diubah menjadi serangkaian perintah motor. *Encoder* pada motor DC memungkinkan robot untuk bergerak secara akurat ke koordinat tertentu, sementara motor servo digunakan untuk mengangkat atau menurunkan pena berwarna sesuai dengan kebutuhan gambar.

Sistem dua motor DC memberikan mobilitas robot dalam dua arah (sumbu X dan Y) pada whiteboard. *Encoder* pada motor ini memastikan bahwa setiap pergerakan robot dapat diukur dan dikontrol dengan presisi tinggi, sehingga gambar yang dihasilkan sesuai dengan *input* aslinya. Selain itu, penggunaan dua pena berbeda warna menambah fleksibilitas robot dalam mereproduksi gambar dengan detail lebih kompleks. Motor servo yang terhubung ke mekanisme pena memungkinkan pergantian antara pena yang aktif dan non-aktif, serta pengangkatan pena saat robot perlu berpindah posisi tanpa menggambar.

Secara keseluruhan, robot gambar ini merupakan aplikasi nyata dari konsep-konsep robotika, pengolahan citra, dan kontrol otomatis. Proyek ini tidak hanya mendemonstrasikan integrasi antara perangkat lunak dan perangkat keras tetapi juga menunjukkan bagaimana teknologi modern dapat digunakan untuk

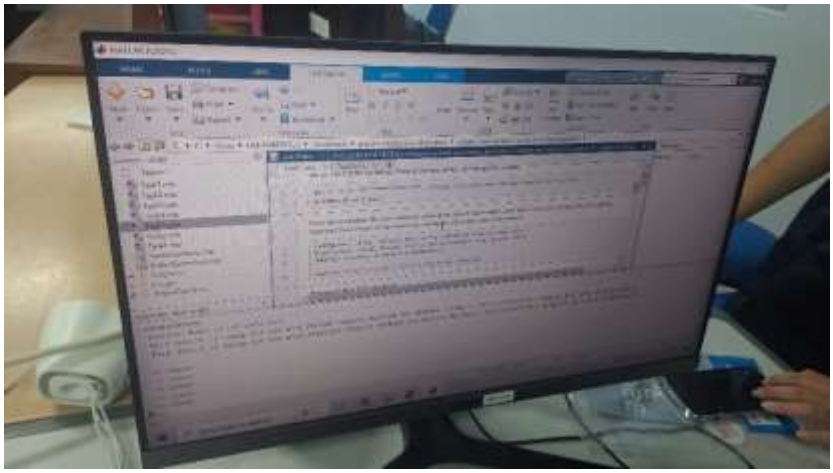
mengotomatiskan tugas-tugas artistik. Keberhasilan proyek ini bergantung pada ketepatan algoritma pengolahan citra, responsifnya sistem kontrol motor, dan desain mekanis robot itu sendiri. Dengan kombinasi teknologi ini, *Drawing Robot* mampu menghasilkan reproduksi gambar yang cukup akurat dan estetis pada permukaan *whiteboard*.

(Sumber: https://www.mathworks.com/help/matlab/sup*port* pkg/aek-drawingrobot-example.html)

2.3. Langkah Kerja

2.3.1. Percobaan 1

1. Membuka percobaan1.mlx



Gambar 2. 1 Percobaan1.mlx

2. Menjalankan *live script* untuk menyambungkan *board* Arduino dengan matlab dan motorcarrier ke Arduino

```
a = arduino;
carrier = motorCarrier(a);
s = servo(carrier,3);
```

3. Ubah variabel pos untuk mengubah posisi spidol di motor servo

```
pos = 0.38;
writePosition(s,pos)
```

4. Buat variabel mL dan mR untuk inisialisasi dc motor kiri dan kanan. Lalu eL dan eR sebagai inisialisasi motor carrier 1 dan 2

```
mL = dcmotor(carrier,'M2');
mR = dcmotor(carrier,'M1');
eL = rotary Encoder(carrier,2);
eR = rotary Encoder(carrier,1);
```

5. *Reset* eL dan eR menjadi 0, agar perhitungan *count* sesuai pergerakan motor

```
reset Count(eL)
reset Count(eR)
```

6. Atur Vset sebagai tegangan yang diinginkan dan Vmax sebagai tegangan maksimum, lalu atur *Speed* pada mL dan mR.

```
Vmax = 12;
Vset = 3;
mL.Speed = Vset/Vmax; mR.Speed = Vset/Vmax;
```

7. Nyalakan atau mulai variabel mL dan mR untuk menaikkan atau menurunkan robot, lalu pause() untuk mengatur lamanya robot berjalan (detik).

```
start(mL) start(mR)
pause(2) % Wait 3 seconds stop(mL)
stop(mR)
```

8. Membaca *Encoder* eL dan eR yang sudah dijalankan sesuai pergerakan motor.

```
count1 = readCount(eL)
count2 = readCount(eR)
```

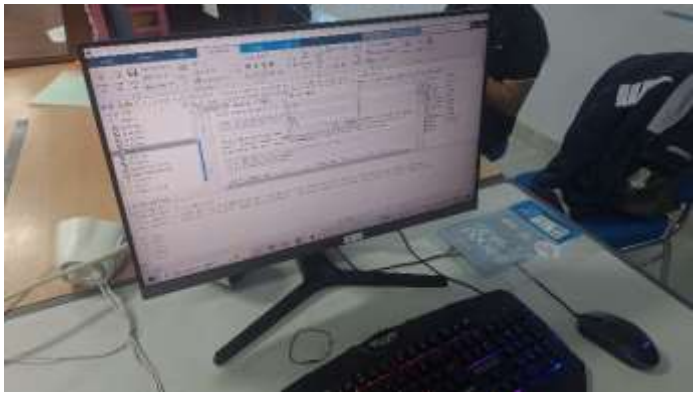
2.3.2. Percobaan 2

1. Masuklah ke bagian Task2.Mlx pada MATLAB. lalu ukur jarak base, L1, dan L2 dari *Drawing Robot* yang ada.

Gambar 2. 2 ukur jarak L1

Gambar 2. 3 ukur jarak L2

2. Ubah variabel base dengan ukuran yang diukur tadi, lalu jalankan program sampai muncul jendela untuk memasukkan nilai L1 dan L2



Gambar 2. 4 Masukan nilai L1 dan L2

3. Jalankan terus program sampai keluar nilai x dan y dari drawing robot
4. Setelah itu sambungkan Arduino dari *Drawing Robot* dengan menjalankan program selanjutnya
5. Jalankan program selanjutnya, jendela baru akan muncul.Cobalah untuk mengubah posisi robot dari semula dengan menjalankan *Left* Motor dan *Right* Motor.

Gambar 2. 5 Jalankan program

6. Jalankan program selanjutnya untuk melakukan kalkulasi perpindahan dan memutus sambungan Arduino drawing robot.

2.4. Hasil Percobaan dan Analisis

2.4.1. Percobaan 1

1. Connect to the hardware

```
a = arduino;
carrier = motorCarrier(a);
```

Pada *section* ini, proses dimulai dengan menghubungkan MATLAB ke perangkat keras Arduino dan motor carrier. Baris pertama, `a = arduino;`, membuat objek Arduino yang memungkinkan komunikasi antara MATLAB dan papan Arduino yang telah disiapkan sebelumnya. Baris berikutnya, `carrier = motorCarrier(a);`, menginisialisasi objek motor carrier yang terkait dengan Arduino melalui variabel a. Objek ini digunakan untuk mengontrol periferal lain pada robot, seperti motor dan servo.

2. Control the servo

```
s = servo(carrier,3);

pos = 0.5; %Change this value and continue running this
section

writePosition(s,pos)
```

Section ini fokus pada pengendalian servo motor yang digunakan untuk mengangkat dan menurunkan spidol papan tulis, yang terhubung ke *port* SERVO3 pada carrier. Baris `s = servo(carrier,3);` membuat objek servo yang terkait dengan *port* ketiga pada motor carrier, memungkinkan MATLAB mengirim perintah ke servo tersebut. Baris berikutnya, `pos = 0.5;` menetapkan nilai awal posisi servo (antara 0 dan 1), dan `writePosition(s,pos)` mengirimkan perintah untuk menggerakkan servo ke posisi tersebut. Dengan mengubah nilai pos secara bertahap dan menjalankan kembali perintah ini, pengguna dapat menentukan nilai yang tepat untuk menurunkan spidol kiri, menurunkan spidol kanan, dan mengangkat keduanya, yang kemudian dapat disimpan ke dalam variabel dan file MAT untuk digunakan pada latihan berikutnya.

3.* Control *the DC motors

```
mL = dcmotor(carrier,'M2');
mR = dcmotor(carrier,'M1');
eL = rotary Encoder(carrier,2);
```

```
eR = rotaryEncoder(carrier,1);

reset Count(eL)
reset Count(eR)

Vmax = 11.1; %Battery voltage (Volts)
Vset = 3; %Target voltage (Volts)

mL.Speed = Vset/Vmax;
mR.Speed = Vset/Vmax;


start(mL)
start(mR)
pause(3) % Wait 3 seconds
stop(mL)
stop(mR)
```

Di *section* ini, DC motor dan *encoder* dihubungkan untuk menggerakkan robot. Baris `mL = dcmotor(carrier,'M2');` dan `mR = dcmotor(carrier,'M1');` membuat objek untuk motor DC kiri (terhubung ke *port* M2) dan kanan (terhubung ke *port* M1) pada carrier. Selanjutnya, `eL = rotaryEncoder(carrier,2);` dan `eR = rotaryEncoder(carrier,1);` menginisialisasi *encoder* untuk motor kiri dan kanan, yang digunakan untuk melacak posisi. Baris `reset Count(eL)` dan `reset Count(eR)` mengatur ulang hitungan *encoder* ke nol, sehingga posisi diukur relatif terhadap posisi awal saat ini. Kemudian, `Vmax = 11.1;` mendefinisikan tegangan maksimum baterai (11.1 V), dan `Vset = 3;` menetapkan tegangan target untuk motor. Baris `mL.Speed = Vset/Vmax;` dan `mR.Speed = Vset/Vmax;` menghitung dan menetapkan kecepatan motor (dalam rentang -1 hingga 1) berdasarkan rasio tegangan target terhadap maksimum. Terakhir, `start(mL)` dan `start(mR)` memulai motor, pause(3) menjeda eksekusi selama 3 detik, lalu `stop(mL)` dan `stop(mR)` menghentikan motor setelah waktu tersebut.

4. Read the encoders

```
count1 = readCount(eL)
count2 = readCount(eR)
```

Setelah motor bergerak, *section* ini membaca perpindahan yang dicatat oleh

encoder. Baris `count1 = readCount(eL)` dan `count2 = readCount(eR)` mengambil jumlah pulsa dari *encoder* kiri dan kanan sejak *reset* terakhir. Nilai ini menunjukkan seberapa jauh motor telah berputar, yang nantinya dapat dikonversi menjadi satuan jarak fisik pada pelajaran berikutnya. *Section* ini sederhana namun penting untuk memverifikasi bahwa *encoder* berfungsi dan motor telah bergerak sesuai perintah sebelumnya.

5. Closed-loop control

```
clear mL mR eL eR


pidML = pidMotor(carrier,2,closed-loop,3,[0.18 0.0 0.01]);
% Modify the PID gains [Kp Ki Kd] as per your requirements
pidMR = pidMotor(carrier,1,closed-loop,3,[0.18 0.0 0.01]);
% Modify the PID gains [Kp Ki Kd] as per your requirements

gearRatio = 100;
theta = 5*2*pi*gearRatio;   % 5 revolutions of the motor
in radians

writeAngularPosition(pidML,theta,'rel');
writeAngularPosition(pidMR,theta,'rel');
```

Section ini memperkenalkan kontrol posisi closed-loop menggunakan PID. Baris clear mL mR eL eR membersihkan variabel motor dan *encoder* sebelumnya untuk menghindari konflik. Kemudian, `pidML = pidMotor(carrier,2,closed-loop,3,[0.18 0.0 0.01]);` dan `pidMR = pidMotor(carrier,1,closed-loop,3,[0.18 0.0 0.01]);` mengkonfigurasi objek PID untuk motor kiri (*port* 2) dan kanan (*port* 1), dengan mode *closed-loop*, 3 pulsa per putaran (dari spesifikasi *encoder*), dan gain PID [Kp Ki Kd] = [0.18 0.0 0.01]. Baris gearRatio = 100; menetapkan rasio gigi motor, dan `theta = 5*2*pi*gearRatio;` menghitung target perpindahan sudut (5 putaran dalam radian). Baris `writeAngularPosition(pidML,theta,'rel');` dan `writeAngularPosition(pidMR,theta,'rel');` memerintahkan motor untuk mencapai perpindahan tersebut dalam mode relatif, secara otomatis memulai dan menghentikan motor saat target tercapai.

6. Read the controller variabel

```
readAngularPosition(pidML)
readAngularPosition(pidMR)

clear a carrier s pidML pidMR
```

Section ini memverifikasi akurasi kontrol *closed-loop*. Baris `readAngularPosition(pidML)` dan `readAngularPosition(pidMR)` membaca perpindahan sudut akhir dari motor kiri dan kanan setelah perintah PID dieksekusi. Nilai ini memungkinkan pengguna membandingkan hasil aktual dengan target (*theta*) untuk mengevaluasi performa *controller*. Terakhir, `clear a carrier s pidML pidMR` membersihkan semua variabel perangkat keras yang digunakan, memastikan tidak ada koneksi yang tertinggal setelah eksperimen selesai.

2.4.2. Percobaan 2

1. Find starting position

```
Base = 1.25; %Change this to the Base distance for your
robot (meters)
save RobotGeometry.mat Base

str = input dlg({'L1 (m)','L2 (m)'},'Enter initial string
lengths.',[1 50]);
L1_i = str2double(str{1}); %meters
L2_i = str2double(str{2}); %meters

L_arm = 0.075;       %meters
Z1_i = L1_i + L_arm; %meters
Z2_i = L2_i + L_arm; %meters

x = (Base^2 + Z1_i^2 - Z2_i^2)/(2*Base) %meters
y = sqrt(Z1_i^2-x^2) %meters
```

Section ini bertujuan untuk menghitung posisi awal robot dalam koordinat x-y berdasarkan pengukuran fisik. Baris `Base = 1.25;` menetapkan jarak antara dua titik referensi (misalnya *pulley*) dalam meter, yang harus diukur dengan alat seperti penggaris atau pita ukur, lalu disimpan ke file `RobotGeometry.mat` menggunakan save `RobotGeometry.mat` Base untuk penggunaan di masa depan. Baris `str = input dlg({'L1 (m)','L2 (m)'},'Enter initial string lengths.',[1 50]);` membuka dialog *input* untuk meminta pengguna memasukkan panjang awal tali L1 dan L2 (dalam meter), yang diukur dari motor

ke *pulley*. Hasil *input* dikonversi ke angka dengan `L1_i = str2double(str{1});` dan `L2_i = str2double(str{2});`. Selanjutnya, `L_arm = 0.075;` mendefinisikan panjang lengan tetap (7.5 cm) dalam meter, dan `Z1_i = L1_i + L_arm;` serta `Z2_i = L2_i + L_arm;` menghitung jarak total awal Z1 dan Z2 dari motor ke titik robot dengan menambahkan `L_arm`. Terakhir, posisi x-y dihitung menggunakan *Teorema Pythagoras*: `x = (Base^2 + Z1_i^2 - Z2_i^2)/(2*Base)` menentukan koordinat x, dan `y = sqrt(Z1_i^2-x^2)` menghitung koordinat y berdasarkan Z1.

2. Connect to the hardware

```
a = arduino;
carrier = motorCarrier(a);
s = servo(carrier,3);
mL = dcmotor(carrier,'M2');
mR = dcmotor(carrier,'M1');
eL = rotaryEncoder(carrier,2);
eR = rotaryEncoder(carrier,1);
reset Count(eL)
reset Count(eR)
```

Section ini menghubungkan MATLAB ke perangkat keras robot untuk menggerakkan dan melacak posisinya. Baris `a = arduino;` membuat objek Arduino untuk komunikasi dengan papan. Kemudian, `carrier = motorCarrier(a);` menginisialisasi motor carrier yang terhubung ke Arduino. Baris `s = servo(carrier,3);` membuat objek servo pada *port* SERVO3 untuk mengontrol spidol. Selanjutnya, `mL = dcmotor(carrier,'M2');` dan `mR = dcmotor(carrier,'M1');` menginisialisasi motor DC kiri (*port* M2) dan kanan (*port* M1), sedangkan `eL = rotaryEncoder(carrier,2);` dan `eR = rotaryEncoder(carrier,1);` mengatur *encoder* untuk melacak pergerakan motor kiri dan kanan. Baris `reset Count(eL)` dan `reset Count(eR)` mengatur ulang hitungan *encoder* ke nol, menetapkan posisi awal saat ini sebagai referensi.

3. Draw on the whiteboard

```
load ServoPositions
SimplePlotterApp(s,mL,mR,LeftMarker,RightMarker)
```

Section ini memungkinkan pengguna menggerakkan robot di papan tulis

menggunakan aplikasi MATLAB. Baris `load ServoPositions` memuat nilai posisi servo (misalnya `LeftMarker` dan `RightMarker`) yang telah ditentukan sebelumnya dari file MAT untuk mengontrol spidol. Baris `SimplePlotterApp(s,mL,mR,LeftMarker,RightMarker)` menjalankan aplikasi GUI bernama `SimplePlotterApp`, yang menerima objek `servo (s)`, `motor kiri (mL)`, motor `kanan (mR)`, serta posisi servo untuk spidol kiri dan kanan sebagai *input*. Aplikasi ini memungkinkan pengguna mengatur tegangan motor untuk menggerakkan robot, dengan peringatan untuk tidak menggerakkan motor terlalu cepat atau mendekati *pulley* agar tidak macet atau rusak.

4. Calculate new robot position

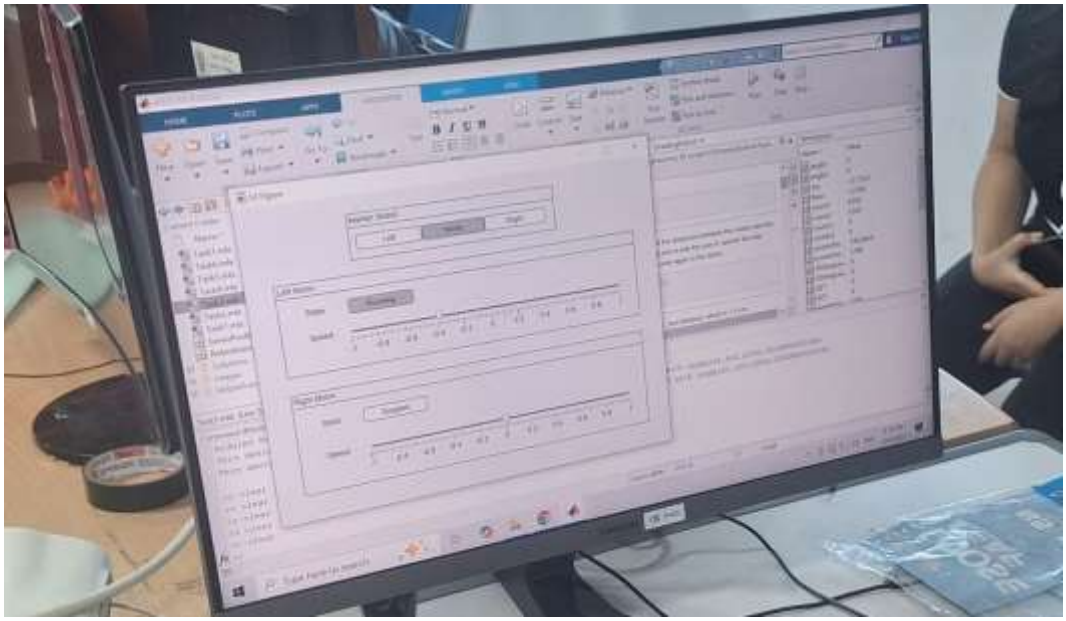
```
counts1 = readCount(eL)
counts2 = readCount(eR)

clear a carrier s mL mR eL eR

countsPerRevolution = 1200;
countsPerRadian = countsPerRevolution/(2*pi);
r_spool = 0.0045; %meters

% Convert counts1 to radians
angle1 = counts1/countsPerRadian %radians
% Convert connts2 to radians
angle2 = counts2/countsPerRadian %radians


dStringLength1 = r_spool*angle1 %meters
dStringLength2 = r_spool*angle2 %meters


dZ1 = dStringLength1/2 %meters
dZ2 = dStringLength2/2 %meters
Z1 = Z1_i + dZ1 %meters
Z2 = Z2_i + dZ2 %meters


x = (Base^2 + Z1^2 - Z2^2)/(2*Base) %meters
y = sqrt(Z1^2-x^2) %meters
```

Setelah robot dipindahkan, *section* ini menghitung posisi baru berdasarkan perubahan panjang tali. Baris `counts1 = readCount(eL)` dan `counts2 = readCount(eR)` membaca jumlah pulsa dari *encoder* kiri dan kanan. Kemudian, `clear a carrier s mL mR eL eR` membersihkan variabel perangkat keras karena tidak lagi diperlukan untuk perhitungan berikutnya. Baris `countsPerRevolution`

`= 1200;` menghitung pulsa per putaran berdasarkan rasio gigi (100) dan spesifikasi *encoder* (12 pulsa per putaran), lalu `countsPerRadian = countsPerRevolution/(2*pi);` mengkonversi ke radian. Radius spool didefinisikan sebagai `r_spool = 0.0045;` (dalam meter). Pulsa *encoder* dikonversi ke sudut dengan `angle1 = counts1/countsPerRadian` dan `angle2 = counts2/countsPerRadian`, lalu ke panjang tali dengan `dStringLength1 = r_spool*angle1` dan `dStringLength2 = r_spool*angle2`. Karena tali berlipat ganda melalui *pulley*, perubahan jarak Z dihitung sebagai `dZ1 = dStringLength1/2` dan `dZ2 = dStringLength2/2`. Jarak baru `Z1` dan `Z2 adalah Z1 = Z1_i + dZ1` dan `Z2 = Z2_i + dZ2`. Terakhir, posisi x-y baru dihitung lagi dengan *Teorema Pythagoras*: `x = (Base^2 + Z1^2 - Z2^2)/(2*Base) dan y = sqrt(Z1^2-x^2).`

5. GUI

GUI `SimplePlotterApp` pada gambar di bawah berfungsi untuk mengontrol pergerakan robot gambar di papan tulis dengan antarmuka yang ramah pengguna. Aplikasi ini mengambil *input* berupa objek `servo (s)` untuk mengangkat/menurunkan spidol dan objek motor DC (`mL` dan `mR`) untuk menggerakkan robot, serta nilai posisi servo (`LeftMarker` dan `RightMarker`) untuk mengontrol spidol kiri dan kanan. Dalam GUI, pengguna dapat mengatur tegangan motor (melalui properti *Speed*) untuk menggerakkan robot ke berbagai arah di papan tulis. Aplikasi ini dirancang sederhana untuk memudahkan eksplorasi, namun pengguna harus berhati-hati agar tidak mengatur tegangan terlalu tinggi atau menggerakkan robot terlalu sampe keluar papan, karena dapat menyebabkan motor macet atau rusak.



Gambar 2. 6 GUI

2.5. Kesimpulan

1. MATLAB berhasil dikoneksikan ke Arduino dan motor carrier untuk mengendalikan berbagai komponen robot, seperti servo dan motor DC.
2. Servo digunakan untuk mengendalikan spidol papan tulis, sementara motor DC dikendalikan dengan pengaturan kecepatan berbasis tegangan untuk menggerakkan robot.
3. *Encoder* digunakan untuk melacak pergerakan motor, membaca jumlah putaran, dan memastikan robot bergerak sesuai perintah.
4. Sistem *closed-loop* menggunakan PID diterapkan untuk meningkatkan akurasi pergerakan motor, dengan parameter yang dapat disesuaikan.
5. Posisi awal robot dihitung menggunakan koordinat x-y berdasarkan panjang tali, dan posisinya terus dipantau selama praktikum untuk memastikan pergerakan sesuai dengan yang diharapkan.