# BAB III DRAWING ROBOT

## 2.1. Tujuan

1. Praktikan dapat memahami prinsip kerja *Drawing Robot,* termasuk komponen utama seperti motor DC, motor servo, dan sistem kendali berbasis Arduino Nano 33 IoT.
2. Praktikan memahami bagaimana perhitungan koordinat dan trigonometri digunakan untuk menentukan pergerakan robot serta bagaimana sistem kontrol bekerja dalam mengatur posisi spidol.
3. Praktikan dapat memahami cara mengontrol motor DC dan motor servo menggunakan sinyal PWM serta bagaimana *encoder rotary* digunakan untuk membaca pergerakan motor.
4. Praktikan dapat memahami dan mengimplementasikan kode program di MATLAB untuk menghubungkan, mengendalikan, serta menganalisis pergerakan *Drawing* Robot berdasarkan *input* yang diberikan.
5. Praktikan dapat mengevaluasi hasil perhitungan dan pergerakan robot, serta memahami bagaimana faktor seperti tegangan, kecepatan motor, dan panjang lengan robot mempengaruhi akurasi dan efisiensi gerakan robot.

## 2.2. Alat dan Bahan

### 2.2.1. PC/Laptop

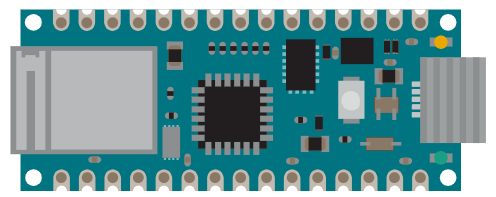


Gambar 3. Laptop

Laptop, atau komputer jinjing, adalah komputer pribadi berukuran kecil dan ringan sehingga mudah dibawa ke mana saja. Laptop memiliki fungsi yang sama dengan komputer desktop, tetapi komponennya didesain lebih kecil, ringan, hemat daya, dan tidak mudah panas. Umumnya, laptop dilengkapi dengan layar kristal cair berukuran 10 hingga 17 inci serta papan tik dan papan sentuh sebagai pengganti tetikus, yang juga dapat digantikan dengan perangkat tambahan melalui USB atau PS/2. Berbeda dengan desktop, laptop dirancang dengan komponen khusus yang lebih efisien dan hemat energi untuk mendukung sifatnya yang portabel.

*(Sumber: https://id.wikipedia.org/wiki/Laptop)*

### 2.2.2. Arduino Nano 33 IoT



Gambar 3. 2 Arduino Nano 33 IoT

Arduino Nano 33 IoT adalah papan terkecil dari Arduino yang dirancang untuk memudahkan pengembangan aplikasi *Internet of Things* (IoT). Dengan *form factor* yang sama seperti Arduino Nano, papan ini cocok untuk membangun jaringan sensor yang terhubung ke *router* atau perangkat Bluetooth *Low Energy* yang dapat mengirim data ke ponsel. Ditenagai oleh prosesor Arm Cortex-M0 32-bit SAMD21, Nano 33 IoT juga dilengkapi modul Wi-Fi u-blox NINA-W102 dan chip kripto ECC608A untuk keamanan data. Kompatibel dengan *platform Arduino Cloud*, papan ini memungkinkan pengembangan proyek IoT dalam hitungan menit. Selain itu, Nano 33 IoT memiliki fitur Bluetooth untuk mengendalikan perangkat periferal serta IMU (LSM6DS3) yang menggabungkan akselerometer dan giroskop, sehingga dapat digunakan untuk proyek pelacakan gerakan.

*(Sumber:* [*https://docs.arduino.cc/hardware/nano-33-iot/)*](https://docs.arduino.cc/hardware/nano-33-iot/))

### 2.2.3. DC Motor



Gambar 3. DC Motor

Motor arus searah (DC) adalah motor listrik yang mengubah energi listrik arus searah menjadi energi mekanis dalam bentuk putaran. Prinsip kerjanya didasarkan pada interaksi antara fluks magnetik dari kumparan medan dan kumparan jangkar. Motor ini banyak digunakan dalam aplikasi yang memerlukan variasi kecepatan dan beban, seperti transportasi rel (motor traksi) untuk lokomotif serta dalam sistem kelistrikan mobil dan peralatan elektronik. Kecepatan putaran rotor motor DC dapat dikendalikan menggunakan metode konvensional atau rangkaian elektronik, terutama dalam industri yang memerlukan variasi kecepatan operasional.

*(Sumber:* [*https://id.wikipedia.org/wiki/Motor\_arus\_searah*](https://id.wikipedia.org/wiki/Motor_arus_searah)))

[*https://assunmotor.com/blog/brushed-dc-motor/)*](https://assunmotor.com/blog/brushed-dc-motor/))

### 2.2.4. Servo Motor

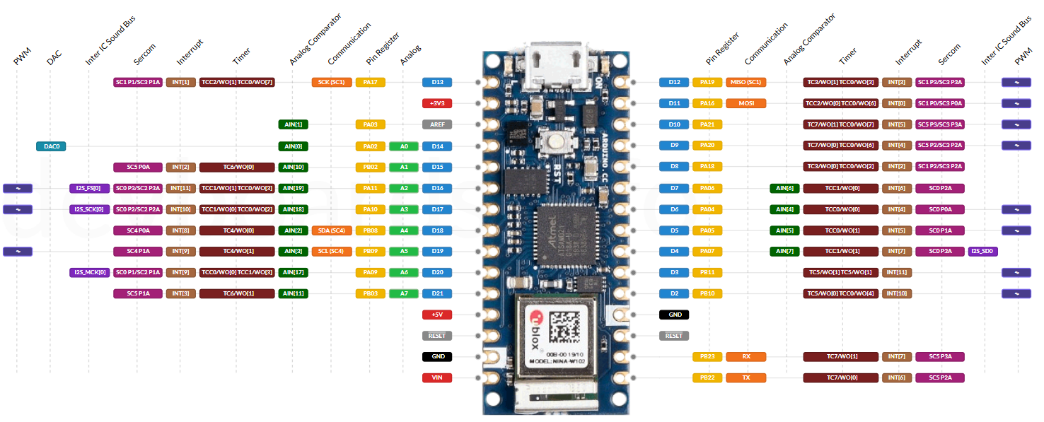


Gambar 3. Servo Motor

Motor servo adalah mekanisme yang menggunakan sistem kontrol umpan balik (*closed-loop control*) untuk mengendalikan pergerakan motor, di mana berbagai jenis motor, seperti motor DC atau *brushless*, dapat digunakan. Sistem ini bekerja dengan mendeteksi posisi poros menggunakan sensor, seperti potensiometer pada servo sederhana atau *rotary encoder* pada versi lebih canggih, lalu mengirimkan umpan balik ke kontrol internal untuk menyesuaikan gerakan hingga mencapai posisi yang diinginkan. Motor servo umumnya dikendalikan menggunakan algoritma PID (*Proportional Integral Derivative*) dan memiliki tiga kabel utama: merah untuk daya (5V–9V), hitam/cokelat untuk *ground*, serta oranye/kuning untuk sinyal data. Komponen utama dalam motor servo meliputi motor DC, rangkaian pengendali, dan sensor posisi, yang biasanya dihubungkan dengan *gearbox* untuk meningkatkan torsi motor.

*(Sumber:* [*https://www.mahirelektro.com/2021/01/pengertian-dan-cara-kontrol-motor-servo-arduino.html#google\_vignette)*](https://www.mahirelektro.com/2021/01/pengertian-dan-cara-kontrol-motor-servo-arduino.html#google_vignette))

### 2.2.5. PWM (Pulse Width Modulation) Microcontrollers



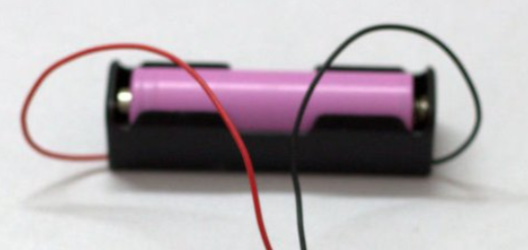
Gambar 3. Pin Arduino Nano 33 IoT

Salah satu metode utama dalam mengendalikan motor adalah *Pulse Width Modulation* (PWM), yang berfungsi untuk memanipulasi lebar pulsa sinyal guna mengatur tegangan dan arus pada motor. PWM digunakan dalam berbagai aplikasi, termasuk telekomunikasi, penguat daya, dan pengatur tegangan, serta efektif dalam mengendalikan kecepatan motor dengan mengatur *duty cycle* antara 0% hingga 100%. Dengan sistem ini, tegangan keluaran dapat bervariasi secara linier dari 0 volt hingga tegangan maksimum, memungkinkan kontrol kecepatan motor yang lebih presisi.

*(Sumber:* [*https://devboards.info/boards/arduino-nano-33-iot*](https://devboards.info/boards/arduino-nano-33-iot)

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

### Battery & Holder



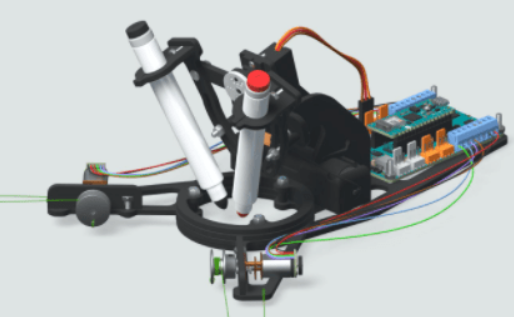
Gambar 3. *Battery* dan *Holder*

Baterai adalah komponen yang menyediakan arus listrik dalam suatu rangkaian. Pada praktikum ini, digunakan baterai Lithium-ion (Li-ion), yang populer dalam perangkat seperti tablet dan ponsel karena ringan dan fleksibel dalam bentuk. Berbeda dari baterai lithium lainnya, Li-ion menggunakan elektrolit polimer semipadat, sehingga lebih ringan tetapi juga lebih rentan terhadap kerusakan dan risiko kebakaran jika bocor. *Battery holder* berfungsi sebagai tempat untuk menampung baterai dalam rangkaian.

*(Sumber:* [*https://duino4projects.com/diy-arduino-battery-capacity-tester/*](https://duino4projects.com/diy-arduino-battery-capacity-tester/)

[*https://id.wikipedia.org/wiki/Baterai\_ion\_litium)*](https://id.wikipedia.org/wiki/Baterai_ion_litium))

### Drawing Robot



Gambar 3. *Drawing* Robot

*Drawing Robot* adalah robot yang dikendalikan oleh Arduino Nano 33 IoT dan terhubung dengan Arduino Nano *Motor Carrier*, dua motor DC *encoder*, serta satu motor mikro-servo. Robot ini dilengkapi dua spidol dengan warna berbeda yang dapat dinaikkan dan diturunkan menggunakan motor servo. Sementara itu, dua motor DC berperan dalam menggerakkan robot agar dapat menggambar dan berpindah di atas papan tulis.

*(Sumber:* [*https://www.mathworks.com/help/simulink/supportpkg/arduino\_ug/aek-drawingrobot-example.html)*](https://www.mathworks.com/help/simulink/supportpkg/arduino_ug/aek-drawingrobot-example.html))

## 2.3. Langkah Kerja

### 2.3.1. Percobaan 1

1. Buka percobaan1.mlx
2. Kemudian jalankan *live script* untuk menyambungkan papan Arduino dengan Matlab dan *motor carrier* ke Arduino.

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

1. Untuk mengubah posisi spidol di motor servo, variabel pos diubah menjadi seperti ini:

|  |
| --- |
| pos = 0.38;  writePosition(s,pos) |

1. Buat variabel mL dan mR untuk inisialisasi dc motor kiri dan kanan. Dan juga buat eL dan eR untuk inisialisasi *motor carier* 1 dan 2.

|  |
| --- |
| mL = dcmotor(carrier,'M2');  mR = dcmotor(carrier,'M1');  eL = rotaryEncoder(carrier,2);  eR = rotaryEncoder(carrier,1); |

1. Supaya perhitungan *count* sesuai pergerakan motor, *reset* eL dan eR menjadi 0 seperti ini:

|  |
| --- |
| resetCount(eL)  resetCount(eR) |

1. Vset diatur sebagai tegangan yang diinginkan dan Vmax diatur sebagai tegangan maksimum, kemudian atur kecepatan pada mL dan mR.

|  |
| --- |
| Vmax = 12;  Vset = 3;  mL.Speed = Vset/Vmax;  mR.Speed = Vset/Vmax; |

1. Setelah itu, mulai variabel mL dan mR untuk menaikkan atau menurunkan robot, lalu pause() untuk mengatur lama robot berjalan (detik).

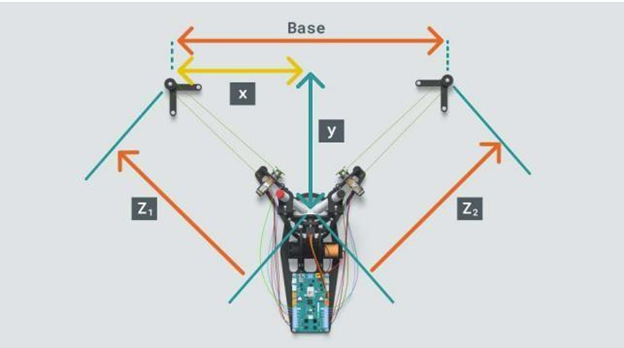
|  |
| --- |
| start(mL)  start(mR)  pause(2) % Wait 3 seconds  stop(mL)  stop(mR) |

1. Membaca *encoder* eL dan eR yang sudah dijalankan sesuai pergerakan motor.

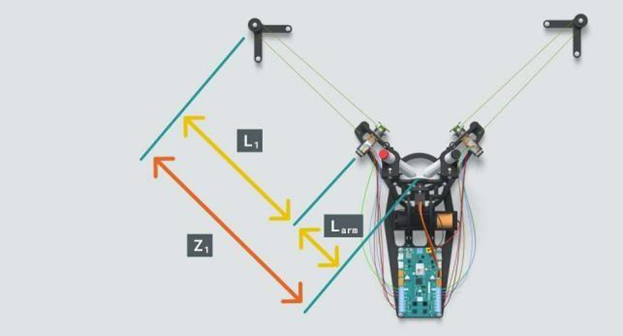
|  |
| --- |
| count1 = readCount(eL)  count2 = readCount(eR) |

### 2.3.2. Percobaan 2

1. Pergi ke bagian Task2.mlx pada Matlab, lalu ukur jarak *base*, L1, dan L2 dari *drawing robot* yang ada.

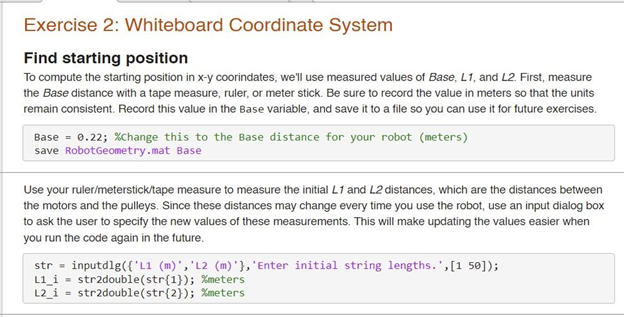


Gambar 3. Skema pengukuran



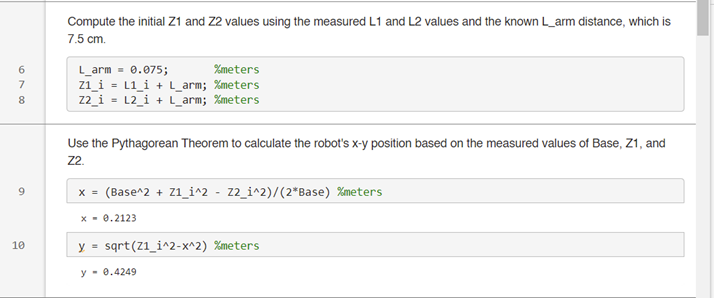
Gambar 3. Ukuran panjang lengan

1. Variabel *base* diubah dengan ukuran yang didapat dari hasil pengukuran sebelumnya, lalu jalankan sampai muncul jendela untuk memasukkan nilai L1 dan L2.



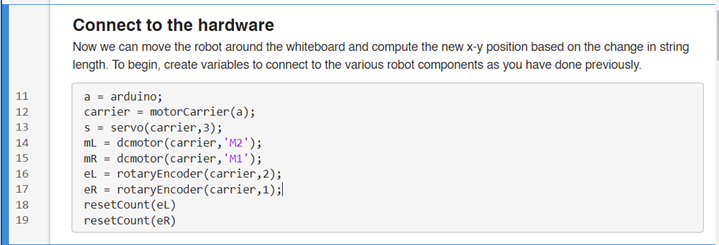
Gambar 3. Pemasukan hasil pengukuran

1. Program terus dijalankan sampai keluar nilai x dan y dari *drawing robot*.



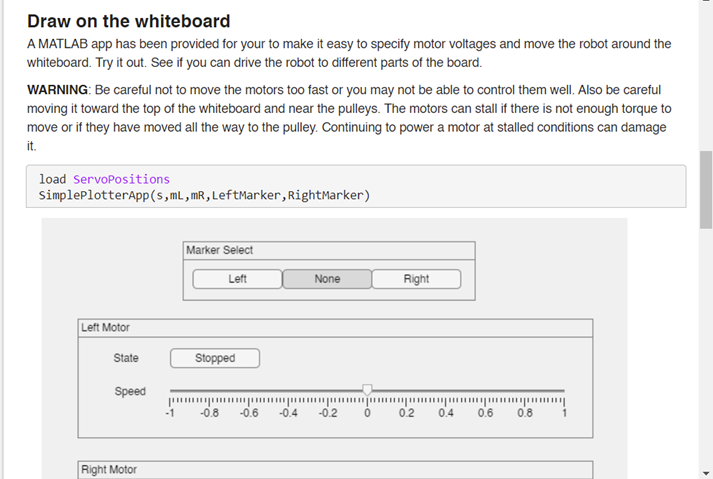
Gambar 3. Pemrosesan nilai pengukuran

1. Kemudian sambungkan Arduino dari *drawing robot* dengan menjalankan program selanjutnya.



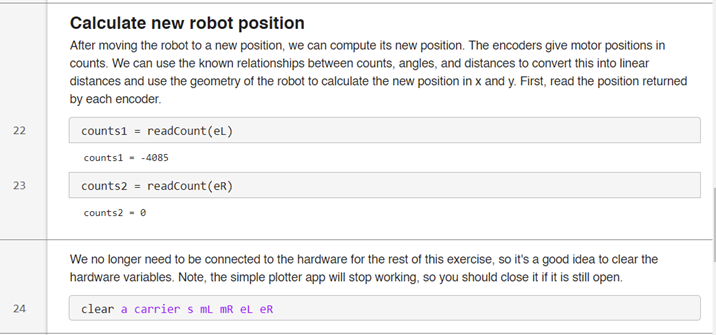
Gambar 3. Koneksi arduino

1. Jalankan program selanjutnya, dan jendela baru akan muncul. Lalu coba untuk mengubah posisi robot dari semula dengan menjalankan *Left M*otor dan *Right Motor*.

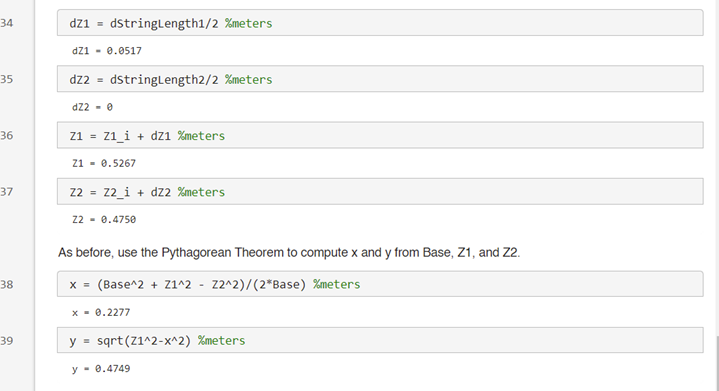


Gambar 3. Mulai mengontrol robot

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



Gambar 3. Perhitungan posisi baru



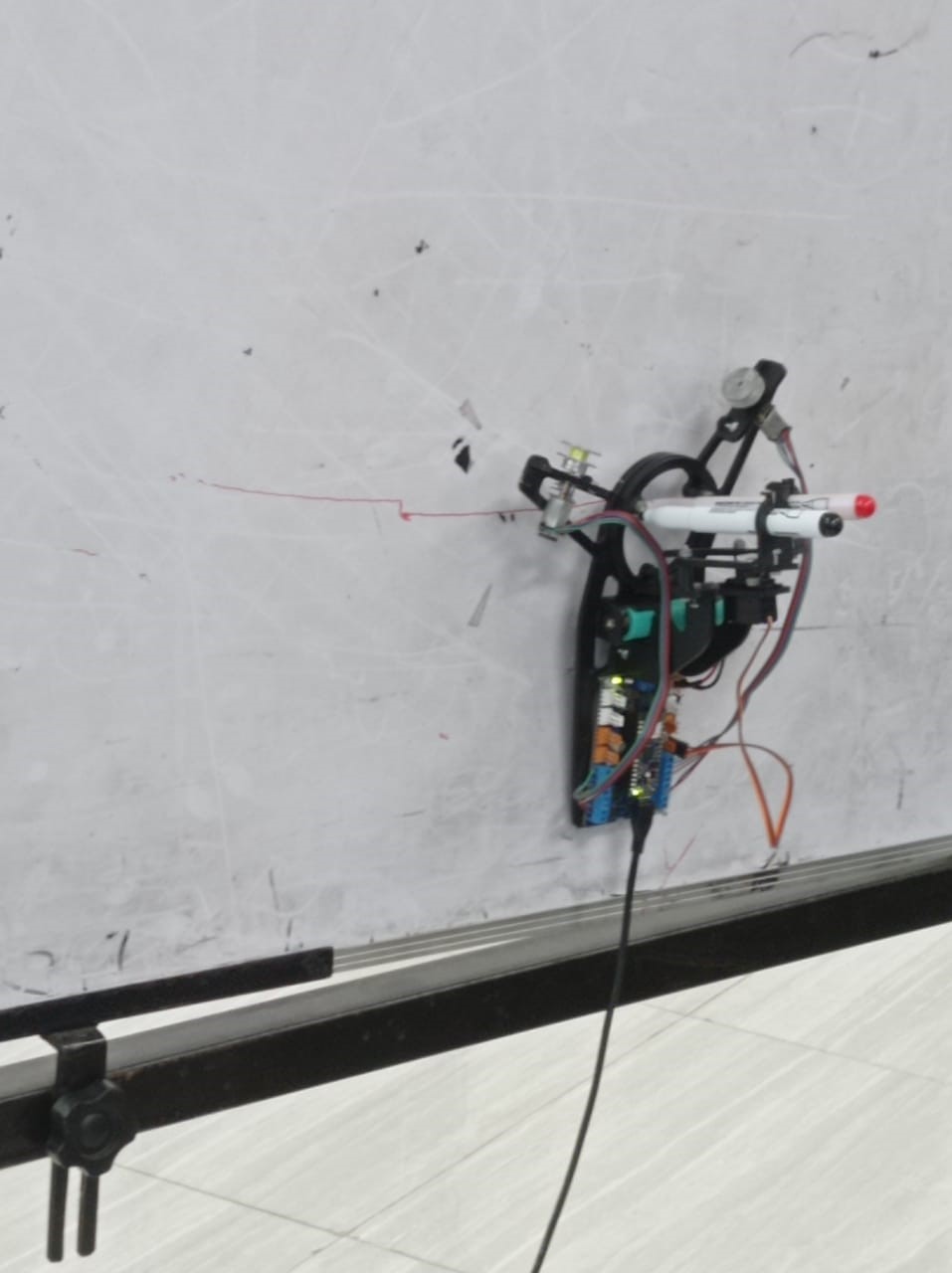
Gambar 3. Perhitungan posisi baru lanjutan

## 2.4. Hasil Percobaan dan Analisis

### 2.4.1. Percobaan 1



Gambar 3. Dokumentasi 1 percobaan 1



Gambar 3. Dokumentasi 2 percobaan 1

Penjelasan setiap baris kode pada *task* 1:

1. *Connect to the Hardware*

|  |
| --- |
| a = arduino;  carrier = motorCarrier(a); |

* a = arduino;

Menghubungkan MATLAB dengan perangkat keras Arduino. Ini membuat objek a yang merepresentasikan koneksi ke *board* Arduino.

* carrier = motorcarrier(a);

Menghubungkan MATLAB dengan *motor* *carrier* yang terpasang pada Arduino. Motor carrier ini berfungsi untuk mengontrol motor dan sensor yang digunakan pada robot.

1. *Control the Servo*

|  |
| --- |
| s = servo(carrier,1); |

* s = servo(carrier,1);  
  Membuat objek s untuk mengontrol servo motor yang terhubung ke *port* SERVO3 pada *motor* *carrier*. Servo ini digunakan untuk menaikkan dan menurunkan spidol pada robot.

|  |
| --- |
| pos = 0.5; %Change this value and continue running this section  writePosition(s,pos) |

* pos = 0.5;

Menentukan posisi awal servo motor dengan nilai 0.5 (posisi menengah). Nilai ini bisa diubah untuk menguji pergerakan servo.

* writePosition(s,pos);  
  Menggerakkan servo ke posisi yang ditentukan (pos).

|  |
| --- |
| LeftMarker = 0.05;  RightMarker = 0.44;  NoMarker = mean([LeftMarker RightMarker]); |

* LeftMarker = 0.05;  
  Menyimpan nilai posisi servo untuk menurunkan spidol kiri.
* RightMarker = 0.44;  
  Menyimpan nilai posisi servo untuk menurunkan spidol kanan.
* NoMarker = mean([LeftMarker RightMarker]);  
  Menghitung nilai rata-rata antara LeftMarker dan RightMarker, yang kemungkinan digunakan untuk menaikkan kedua spidol.

|  |
| --- |
| save ServoPositions.mat LeftMarker RightMarker NoMarker |

* save ServoPositions.mat LeftMarker RightMarker NoMarker;  
  Menyimpan nilai posisi servo dalam file MAT-file (ServoPositions.mat) agar dapat digunakan kembali pada latihan berikutnya.

1. *Control the DC Motors*

|  |
| --- |
| mL = dcmotor(carrier,'M2');  mR = dcmotor(carrier,'M1');  eL = rotaryEncoder(carrier,2);  eR = rotaryEncoder(carrier,1); |

* mL = dcmotor(carrier,'M2');  
  Membuat objek mL untuk motor DC kiri yang terhubung ke *port* M2.
* mR = dcmotor(carrier,'M1');  
  Membuat objek mR untuk motor DC kanan yang terhubung ke *port* M1.
* eL = rotaryEncoder(carrier,2);  
  Membuat objek eL untuk enkoder *rotary* yang dipasang pada motor kiri di kanal 2.
* eR = rotaryEncoder(carrier,1);  
  Membuat objek eR untuk enkoder *rotary* yang dipasang pada motor kanan di kanal 1.

|  |
| --- |
| resetCount(eL)  resetCount(eR) |

* resetCount(eL);  
  Mengatur ulang (*reset*) nilai enkoder kiri ke nol agar perhitungan pergerakan motor dimulai dari titik awal.
* resetCount(eR);  
  Mengatur ulang nilai enkoder kanan ke nol.

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

* Vmax = 11.1;  
  Menentukan tegangan maksimum baterai (11.1V).
* Vset = 3;  
  Menentukan tegangan target yang akan diberikan ke motor (3V).

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

* mL.Speed = Vset/Vmax;  
  Mengatur kecepatan motor kiri berdasarkan tegangan target.
* mR.Speed = Vset/Vmax;  
  Mengatur kecepatan motor kanan berdasarkan tegangan target.

|  |
| --- |
| start(mL)  start(mR)  pause(3)  stop(mL)  stop(mR) |

* start(mL); start(mR);  
  Memulai motor kiri dan kanan.
* pause(3);  
  Menunggu selama 3 detik agar motor beroperasi.
* stop(mL); stop(mR);  
  Menghentikan motor kiri dan kanan setelah 3 detik.

1. *Read the Encoders*

|  |
| --- |
| count1 = readCount(eL)  count2 = readCount(eR) |

* count1 = readCount(eL);  
  Membaca jumlah pulsa dari enkoder kiri untuk mengetahui seberapa jauh motor telah bergerak.
* count2 = readCount(eR);  
  Membaca jumlah pulsa dari enkoder kanan.

1. *Closed*-*loop Control*

|  |
| --- |
| clear mL mR eL eR |

* clear mL mR eL eR;  
   Menghapus variabel mL, mR, eL, dan eR untuk memastikan tidak ada koneksi lama yang mengganggu.

|  |
| --- |
| pidML = pidMotor(carrier,2,'position',3,[0.18 0.0 0.01]);  pidMR = pidMotor(carrier,1,'position',3,[0.18 0.0 0.01]); |

* pidML = pidMotor(carrier,2,'position',3,[0.18 0.0 0.01]);  
  Menggunakan kontrol PID untuk mengendalikan motor kiri dengan *setpoint* posisi. Gain PID yang digunakan:
* Kp = 0.18 (proporsional)
* Ki = 0.0 (integral)
* Kd = 0.01 (derivatif)
* pidMR = pidMotor(carrier,1,'position',3,[0.18 0.0 0.01]);  
  Sama seperti motor kiri, tetapi untuk motor kanan.

|  |
| --- |
| gearRatio = 100;  theta = 52pi\*gearRatio; |

* gearRatio = 100;  
  Menentukan rasio roda gigi pada motor (100:1).
* theta = 5\*2\*pi\*gearRatio;  
  Menghitung target sudut rotasi motor dalam radian untuk 5 putaran penuh.

|  |
| --- |
| writeAngularPosition(pidML,theta,'rel');  writeAngularPosition(pidMR,theta,'rel'); |

* writeAngularPosition(pidML,theta,'rel');  
  Memerintahkan motor kiri untuk berputar sebanyak *theta* radian dalam mode relatif (*relative positioning*).
* writeAngularPosition(pidMR,theta,'rel');  
  Memerintahkan motor kanan untuk berputar sebanyak *theta* radian.

1. *Read the Controlled Variable*

|  |
| --- |
| readAngularPosition(pidML)  readAngularPosition(pidMR) |

* readAngularPosition(pidML);  
  Membaca posisi akhir motor kiri setelah pergerakan.
* readAngularPosition(pidMR);  
  Membaca posisi akhir motor kanan setelah pergerakan.

|  |
| --- |
| clear a carrier s pidML pidMR |

* clear a carrier s pidML pidMR;  
  Membersihkan semua variabel perangkat keras untuk menghindari konflik pada eksekusi berikutnya.

### 2.4.2. Percobaan 2



Gambar 3. Dokumentasi 1 percobaan 2



Gambar 3. Dokumentasi 2 percobaan 2

Penjelasan setiap baris kode pada *task* 2:

1. *Find Starting Position*

Bagian ini menghitung posisi awal robot dalam sistem koordinat x-y.

1. Menentukan jarak dasar (*Base*)

|  |
| --- |
| Base = 1.250; %Change this to the Base distance for your robot (meters)  save RobotGeometry.mat Base |

* Base adalah jarak antara dua motor yang mengontrol pergerakan robot.
* Nilai ini disimpan ke file ‘RobotGeometry.mat’ agar bisa digunakan nanti.

1. Memasukkan nilai awal panjang tali (L1 dan L2)

|  |
| --- |
| str = inputdlg({'L1 (m)','L2 (m)'},'Enter initial string lengths.',[1 50]);  L1\_i = str2double(str{1}); %meters  L2\_i = str2double(str{2}); %meters |

* Program meminta pengguna memasukkan panjang awal dari tali kiri (L1) dan kanan (L2).
* inputdlg menampilkan dialog *input* ke pengguna.
* str2double mengubah *input string* ke angka desimal.

1. Menghitung nilai awal Z1 dan Z2

|  |
| --- |
| L\_arm = 0.075; %meters  Z1\_i = L1\_i + L\_arm; %meters  Z2\_i = L2\_i + L\_arm; %meters |

* L\_arm adalah panjang lengan yang menambah panjang tali aktual.
* Z1 dan Z2 adalah panjang total dari masing-masing tali ditambah panjang lengan.

1. Menggunakan Teorema Pythagoras untuk menentukan posisi (x, y)

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

* Rumus ini didasarkan pada teorema *Pythagoras* untuk menemukan koordinat x dan y.

1. *Connect to the* *Hardware*

Bagian ini menghubungkan MATLAB dengan komponen robot.

Inisialisasi koneksi ke Arduino dan motor

|  |
| --- |
| a = arduino;  carrier = motorCarrier(a);  s = servo(carrier,1);  mL = dcmotor(carrier,'M2');  mR = dcmotor(carrier,'M1');  eL = rotaryEncoder(carrier,2);  eR = rotaryEncoder(carrier,1);  resetCount(eL)  resetCount(eR) |

* arduino menghubungkan MATLAB dengan *board* Arduino.
* motorCarrier(a) menghubungkan *shield motor*.
* servo, dcmotor, dan ‘rotaryEncoder’ masing-masing mengontrol servo, motor DC, dan *encoder*.
* resetCount mengatur ulang posisi *encoder* sebelum mulai digunakan.

1. *Draw on the* *Whiteboard*

Bagian ini mengaktifkan GUI MATLAB untuk mengontrol robot.

|  |
| --- |
| load ServoPositions |

* Memuat data posisi sebelumnya agar robot bisa melanjutkan dari posisi terakhir.

|  |
| --- |
| SimplePlotterApp(s,mL,mR,LeftMarker,RightMarker) |

* ‘SimplePlotterApp’ adalah aplikasi GUI yang memudahkan pengguna dalam mengontrol robot.
* GUI ini memungkinkan pengguna memilih motor yang akan dikontrol serta mengatur kecepatannya.

1. Penjelasan GUI

GUI (seperti pada gambar terlampir) terdiri dari beberapa bagian utama:

1. *Marker Select*

* Memilih *marker* (kiri, kanan, atau tidak ada) yang digunakan untuk menggambar.

1. *Left Motor* dan *Right Motor Control*

* *State*: Menampilkan status motor (Stopped atau berjalan).
* *Speed Slider*: Mengontrol kecepatan motor antara -1 hingga 1.

GUI ini mempermudah pengguna dalam menggerakkan robot di atas *whiteboard* dengan mengatur tegangan motor secara manual.

## 2.5. Kesimpulan

1. Arduino Nano 33 IoT, motor DC, motor servo, dan pengolahan sinyal dapat digunakan untuk menggerakkan robot dalam menggambar.
2. Motor DC pada robot dikendalikan menggunakan sinyal *Pulse Width Modulation* (PWM), yang memungkinkan kontrol kecepatan dan arah gerak robot secara presisi melalui pengaturan tegangan.
3. Percobaan menunjukkan bahwa *rotary encoder* pada motor DC berperan dalam menghitung jumlah putaran, yang digunakan untuk menentukan pergerakan robot di atas papan tulis.
4. Pada percobaan 2, pengukuran nilai *base*, L1, dan L2 digunakan untuk menghitung koordinat x dan y, yang mempengaruhi ketepatan robot dalam mereproduksi gambar di papan tulis.
5. Motor servo digunakan untuk menaikkan dan menurunkan spidol, memungkinkan robot menggambar dengan warna yang berbeda sesuai dengan perintah yang diberikan melalui program.