BAB III
METODOLOGI PENELITIAN

Penelitian ini bertujuan untuk merancang sistem pergerakan *quadruped robot* dengan menggunakan *geometric approach inverse kinematics* dan *trot gait pattern* sesuai dengan rumusan masalah. Untuk merealisasikan tujuan penelitian, maka dibutuhkan suatu metode penelitian yang detail. Metode penelitian menggambarkan rancangan penelitian yang meliputi prosedur atau langkah yang harus ditempuh, sumber data, prosedur pengambilan data, serta prosedur analisis data. Metode penelitian akan sangat membantu peneliti untuk mengendalikan kegiatan atau tahapan penelitian.

Metode penelitian merupakan pendekatan ilmiah untuk mendapatkan informasi dan manfaat tertentu. Untuk menyelesaikan rumusan masalah dan merealisasikan tujuan penelitian maka diperlukan langkah-langkah untuk menyelesaikan masalah tersebut. Adapun langkah-langkah yang perlu dilakukan untuk merealisasikan sistem yang dirancang pada penelitian ini adalah sebagai berikut:

1. Penentuan spesifikasi desain
2. Perancangan diagram blok sistem
3. Pembuatan perangkat keras (*hardware*)
4. Perancangan algoritma dan perangkat lunak(*software*)
5. Perancangan Sistem Pergerakan *Quadruped Robot*
6. Pengujian sistem
7. Pengambilan kesimpulan dan saran

Metode penelitian tersebut dijelaskan sebagai berikut:

3.1 Penentuan Spesifikasi Desain

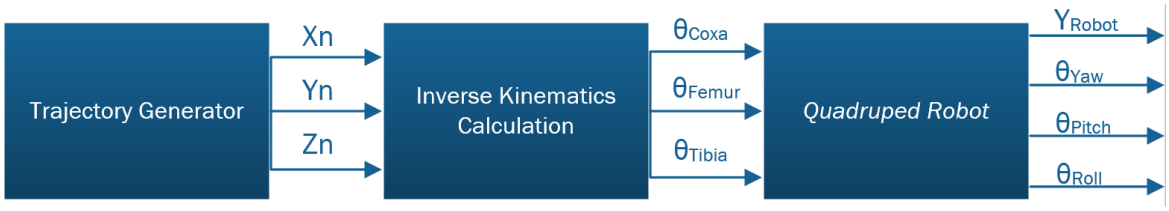
Spesifikasi desain secara global ditetapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya. Spesifikasi sistem pergerakan *quadruped robot* yang direncanakan adalah sebagai berikut:

1) Kesalahan rata-rata dari pergerakan maju *quadruped robot* kurang dari 5%.
2) Kesalahan rata-rata orientasi *yaw* dari *quadruped robot* ketika bergerak maju sejauh 20 cm kurang dari 10^{o}.

3) Deviasi *body* robot menurut orientasi *pitch* dan *roll* ketika berjalan tidak lebih dari 10º ketika berjalan pada bidang datar.

3.2 Perancangan Diagram Blok Sistem

Berdasarkan spesifikasi desain dan spesifikasi alat yang telah ditentukan sebelumnya, maka dapat dibuat diagram blok sistem yang dapat menjabarkan sistem secara garis besar. Diagram blok sistem dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Blok Sistem

Keterangan dari diagram blok sistem pada Gambar 3.1 adalah sebagai berikut:

1) Sistem pergerakan merupakan *open-loop control system*.
2) *Trajectory Generator* berfungsi untuk membangkitkan kumpulan koordinat *Cartesian* yang harus dilewati oleh end effector untuk berpindah dari posisi awal ke posisi tujuan
3) *Trajectory Generator* membangkitkan sinyal yang berbentuk *Half-Wave Rectified Sine*.
4) Kumpulan koordinat Cartesian yang dibangkitkan *Trajectory Generator* menjadi *input dari* kalkulasi *inverse kinematics*.
5) *Output* dari kalkulasi *inverse kinematics* adalah sudut yang harus dibentuk oleh sendi Coxa, Femur, dan Tibia agar *end-effector* dapat mencapai koordinat yang dituju.
6) Perubahan posisi *end effector* pada setiap kaki mempengaruhi perpindahan (Y_{Robot}), sudut orientasi *yaw* (θ_{Yaw}), sudut orientasi *pitch* (θ_{Pitch}), dan sudut orientasi *roll* (θ_{Roll}) dari *quadruped robot.*
7) Penelitian ini berfokus pada Perancangan algoritma *geometric approach inverse kinematics* dan *trajectory generator* pada sistem pergerakan *quadruped robot*. Dinamika robot tidak dibahas dalam penelitian ini, dinamika adalah hubungan antara gaya yang diberlakukan kepada robot dan akselerasi yang dihasilkan.

3.3 Karakterisasi Subsistem

Karakterisasi subsistem dilakukan untuk mempermudah analisis sistem pergerakan *quadruper robot*. Subsistem harus diuji untuk memastikan bahwa setiap subsistem berjalan

sebagai mestinya dan layak digunakan dalam sistem. Karakterisasi subsistem dibagi menjadi beberapa bagian, yaitu:

3.3.1 Karakterisasi Dynamixel AX-12A *Smart Servo*

a. Tujuan

Karakterisasi Dynamixel AX-12A Smart Servo bertujuan untuk mengetahui kelinieran kinerja Dynamixel AX-12A yang diamati dari sudut yang dibentuk oleh servo ketika diberikan suatu instruksi instruksi yang dikirim oleh mikrokontroler STM32F407VGT6.

b. Peralatan yang digunakan

1) LiPo (Lithium Polymer) Battery 3S 11.1 V 2200 mAh.

2) Dynamixel AX-12A Smart Servo

3) Quadruped Robot Electronics Board

4) USB-A Male to USB Mini-B Male cable

5) Laptop

6) Kabel penghubung

7) Penggaris busur derajat

c. Langkah Pengujian

1) Hubungkan *output* tegangan *LiPo (Lithium Polymer) Battery* dengan *Quadruped Robot Electronics Board* dan *Dynamixel AX-12A.*

2) Atur instruksi yang dikirimkan dari *STM32F407VGT6* ke *Dynamixel AX-12A* dari 0º hingga 300º.

3) Gunakan Penggaris bujur derajat untuk mengukur sudut aktual yang dibentuk oleh *Dynamixel AX-12A.*

4) Amati dan catat hasil pengukuran sudut servo di setiap kenaikan 10º.

Dynamixel AX-12A menerima instruksi sudut dalam bentuk bilangan hexadecimal, yang memiliki resolusi 10 bit. Sudut 0º hingga 300º direpresentasikan dengan nilai 0 (0x000) hingga 1023 (0x3FF). Maka dari itu diperlukan konversi sudut ke *hexadecimal* pada *STM32F407VGT6*, dengan memperhitungkan jangkauan sudut dan jangkauan nilai instruksi *Dynamixel AX-12A* maka didapatkan persamaan 3.1.

$$GOAL\_VALUE = \frac{GOAL\_ANGLE}{MAX\_ANGLE} \times MAX\_VALUE \quad \text{..... (3.1)}$$

Dengan diketahui bahwa,

- MAX_VALUE : 1023
- MAX_ANGLE : 300
- GOAL_ANGLE : Sudut yang diinginkan
- GOAL_VALUE : Hasil konversi untuk sudut yang diinginkan

Data pengujian *Dynamixel AX-12A Smart Servo* ditunjukkan dalam Tabel 3.1.

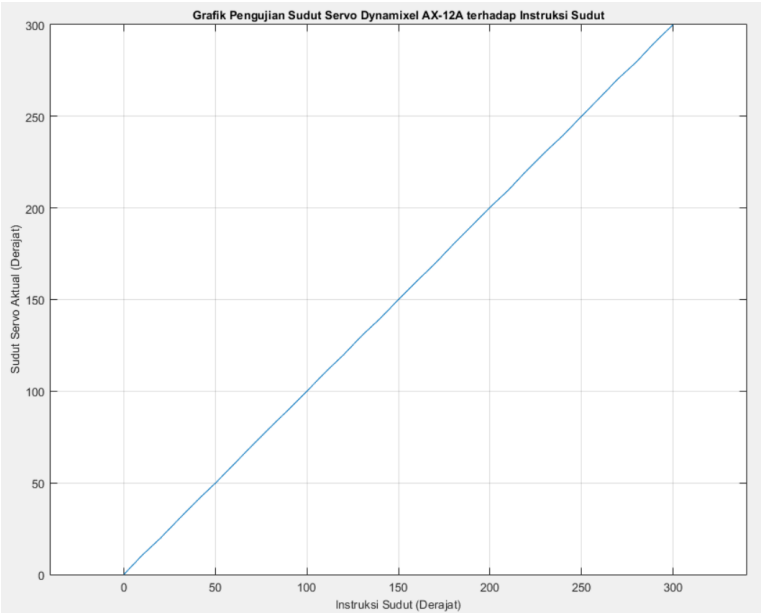
Tabel 3.1

Data Pengujian Dynamixel AX-12A *Smart Servo*

No	Instruksi Sudut (Derajat)	Hasil Konversi	Pengujian (Derajat)			Rata-Rata (Derajat)	Error (Derajat)
			1	2	3		
1	0	0	0	0	0	0	0
2	10	34	10	10.5	11	10.5	-0.5
3	20	68	20	19.5	20	19.833	0.167
4	30	102	30	30.5	30	30.167	-0.167
5	40	136	40	40	41	40.333	-0.333
6	50	170	50	50.5	49	49.833	0.167
7	60	204	60	60	60	60	0
8	70	238	71	70	70	70.333	-0.333
9	80	272	80	80	81	80.333	-0.333
10	90	306	90	90	90	90	0
11	100	341	100.5	99.5	100	100	0
12	110	375	111	110	110	110.333	-0.333
13	120	409	120	119.5	120	119.833	0.167
14	130	443	130	131	130	130.333	-0.333
15	140	477	139	140	140	139.667	0.333
16	150	511	150	150	150	150	0
17	160	545	160	161	159	160	0
18	170	579	170	169	170	169.667	0.333
19	180	613	180.5	180	180	180.167	-0.167
20	190	647	189.5	191	190	190.167	-0.167
21	200	682	200	201	200	200.333	-0.333
22	210	716	210	210	209	209.667	0.333

23	220	750	220	221	220	220.333	-0.333
24	230	784	231	230	230	230.333	-0.333
25	240	818	240	239	240	239.667	0.333
26	250	852	250	251	249	250	0
27	260	886	260.5	260	260	260.167	-0.167
28	270	920	271	270	270.5	270.5	-0.5
29	280	954	279	280	280	279.667	0.333
30	290	988	291	290	290	290.333	-0.333
31	300	1023	300	300	300	300	0

Data pengujian Dynamixel AX-12A Smart Servo direpresentasikan melalui grafik pada Gambar 3.2.



Gambar 3.2 Grafik Pengujian Sudut Servo Dynamixel AX-12A

3.3.2 Karakterisasi CMPS11 Magnetic Compass

Karakterisasi CMPS11 *Magnetic Compass* bertujuan untuk mengamati kelinieran hasil pembacaan sensor CMPS11 ketika orientasi robot diubah dari *pitch*, *roll*, dan *Yaw*. Pengujian juga dilakukan untuk mengamati apakah CMPS11 dapat digunakan sebagai umpan balik yang layak berdasarkan simpangan antara hasil pembacaan sensor dan orientasi aktual robot.

Pada pengujian CMPS11 *Magnetic Compass*, terdapat serangkaian prosedur yang dapat dibagi menjadi empat bagian menurut tujuannya, yaitu:

1. Kalibrasi CMPS11 *Magnetic Compass*
    a. Tujuan

Kalibrasi bertujuan untuk memperbaharui data referensi keadaan *level* pada setiap *axis* serta mengukur *sensor noise* di sekitar sensor agar ketidaksempurnaan teknis tersebut diperhitungkan ketika melakukan kalkulasi untuk mendapatkan nilai *pitch, roll,* dan *yaw.*

b. Peralatan yang Digunakan
    1) *LiPo (Lithium Polymer) Battery 3S 11.1 V 2200 mAh.*
    2) *Quadruped Robot Electronics Board*
    3) *CMPS11 Magnetic Compass*
    4) *USB-A Male to USB Mini-B Male cable*
    5) Laptop
    6) Kabel penghubung

c. Langkah Kalibrasi

Prosedur kalibrasi CMPS11 *Magnetic Compass* adalah sebagai berikut:

1) Siapkan program yang berfungsi untuk mengirimkan 3 *byte* data dengan nilai 0xF0, 0xF5, 0xF6 secara berurutan kepada *command register* dari CMPS11 *Magnetic Compass*. Data tersebut harus dikirimkan dalam 3 *frame* data I2C yang terpisah dan terdapat interval minimum 20ms setiap pengiriman data. Program ini befungsi untuk masuk ke dalam *calibration mode* dari CMPS11 *Magnetic Compass.* Kemudian dilanjutkan dengan instruksi untuk mengirimkan nilai 0xF8 ke *command register CMPS11 Magnetic Compass* ketika *user button* ditekan. Instruksi ini berfungsi untuk keluar dari *calibration mode*.
2) *Upload* program tersebut ke *STM32F407VGT6.*
3) Hubungkan baterai ke rangkaian elektronik robot.
4) Nyalakan robot dengan menekan *power switch.*
5) Masuk ke *calibration mode.*
6) Putar CMPS11 ke segala arah secara perlahan hingga LED pada CMPS11 tidak menyala lagi. Hal ini bertujuan untuk melakukan kalibrasi dari *magnetometer.*
7) Untuk melakukan kalibrasi *accelerometer* dan *gyro sensor*, CMPS11 harus diletakkan secara horizontal, terbalik, dan vertical pada keempat sisinya. CMPS11 harus diletakkan secara stabil di setiap titik yang diperlukan selama

200 ms. LED pada CMPS11 akan berkedip sebagai penanda data pada titik tersebut telah didapat.

8) Tekan *user button* untuk mengirimkan instruksi keluar dari *calibration mode.*

2. Pengujian Pembacaan Sudut Pitch CMPS11 *Magnetic Compass*

a. Tujuan

Pengujian pembacaan sudut pitch bertujuan untuk menguji apakah CMPS11 yang telah dikalibrasi telah layak untuk dipakai dalam pengujian sistem.

Peralatan yang Digunakan

1) LiPo (Lithium Polymer) Battery 3S 11.1 V 2200 mAh.

2) Quadruped Robot Electronics Board

3) *CMPS11 Magnetic Compass*

4) *USB-A Male to USB Mini-B Male cable*

5) Laptop

6) Kabel penghubung

7) Penggaris busur derajat

b. Langkah Pengujian

Prosedur pengujian pembacaan sudut *pitch* CMPS11 *Magnetic Compass* adalah sebagai berikut:

1) Siapkan program yang berfungsi untuk membaca nilai pitch dari CMPS11.

2) *Upload* program tersebut ke *STM32F407VGT6.*

3) Hubungkan baterai ke rangkaian elektronik robot.

4) Nyalakan robot dengan menekan *power switch.*

5) Ukur nilai pembacaan pitch pada setiap sudut menurut *axis x* dengan memberikan sudut roll 0º (CMPS11 dalam keadaan level menurut *axis y*). Percobaan dimulai dari keadaan level hingga -90º dan kemudian dari keadaan level hingga +90º. Pembacaan dilakukan setiap interval 10º.

6) Amati, catat, dan analisis nilai pembacaan sudut *pitch.*

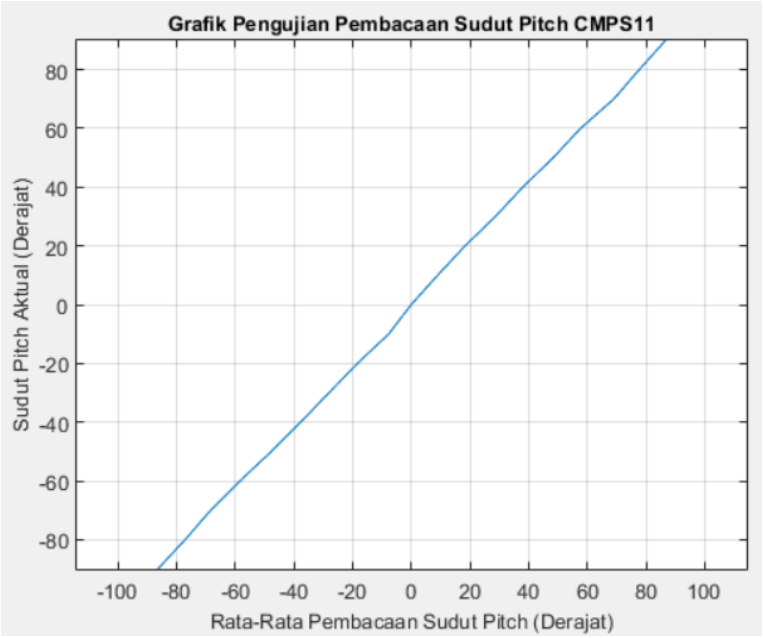
Data pengujian pembacaan sudut pitch CMPS11 ditunjukkan dalam Tabel 3.2.

Tabel 3.2

Data Pengujian Pembacaan Sudut Pitch CMPS11

No	Sudut Aktual (Derajat)	Pembacaan Sudut (Derajat)			Rata-Rata Pembacaan Sudut (Derajat)	Rata-Rata Error (Derajat)
		1	2	3		
1	-90	-88	-86	-85	-86.333	-3.667
2	-80	-77	-78	-76	-77	-3
3	-70	-69	-67	-69	-68.333	-1.667
4	-60	-59	-59	-57	-58.333	-1.667
5	-50	-48	-46	-49	-47.667	-2.333
6	-40	-39	-37	-37	-37.667	-2.333
7	-30	-28	-27	-29	-28	-2
8	-20	-18	-19	-18	-18.333	-1.667
9	-10	-9	-7	-7	-7.667	-2.333
10	0	0	0	0	0	0
11	10	9	8	10	9	1
12	20	18	18	19	18.333	1.667
13	30	29	27	30	28.667	1.333
14	40	38	39	37	38	2
15	50	49	47	49	48.333	1.667
16	60	57	58	58	57.667	2.333
17	70	68	69	70	69	1
18	80	79	77	77	77.667	2.333
19	90	88	85	87	86.667	3.333

Data pengujian pembacaan sudut pitch CMPS11 direpresentasikan melalui grafik pada Gambar 3.3.



Gambar 3.3 Grafik Pengujian Pembacaan Sudut *Pitch* CMPS11

3. Pengujian Pembacaan Sudut Roll CMPS11 *Magnetic Compass*

a. Tujuan

Pengujian pembacaan sudut roll bertujuan untuk menguji apakah CMPS11 yang telah dikalibrasi telah layak untuk dipakai dalam pengujian sistem.

b. Peralatan yang Digunakan

1) LiPo (Lithium Polymer) Battery 3S 11.1 V 2200 mAh.

2) Quadruped Robot Electronics Board

3) *CMPS11 Magnetic Compass*

4) *USB-A Male to USB Mini-B Male cable*

5) Laptop

6) Kabel penghubung

7) Penggaris busur derajat

c. Langkah Pengujian

1) Siapkan program yang berfungsi untuk membaca nilai pitch dari CMPS11.

2) *Upload* program tersebut ke *STM32F407VGT6.*

3) Hubungkan baterai ke rangkaian elektronik robot.

4) Nyalakan robot dengan menekan *power switch.*

5) Ukur nilai pembacaan roll pada setiap sudut menurut *axis y* dengan memberikan sudut pitch 0º (CMPS11 dalam keadaan level menurut *axis x*).

Percobaan dimulai dari keadaan level hingga -90º dan kemudian dari keadaan level hingga +90º. Pembacaan dilakukan setiap interval 10º.

6) Amati, catat, dan analisis nilai pembacaan sudut *roll.*

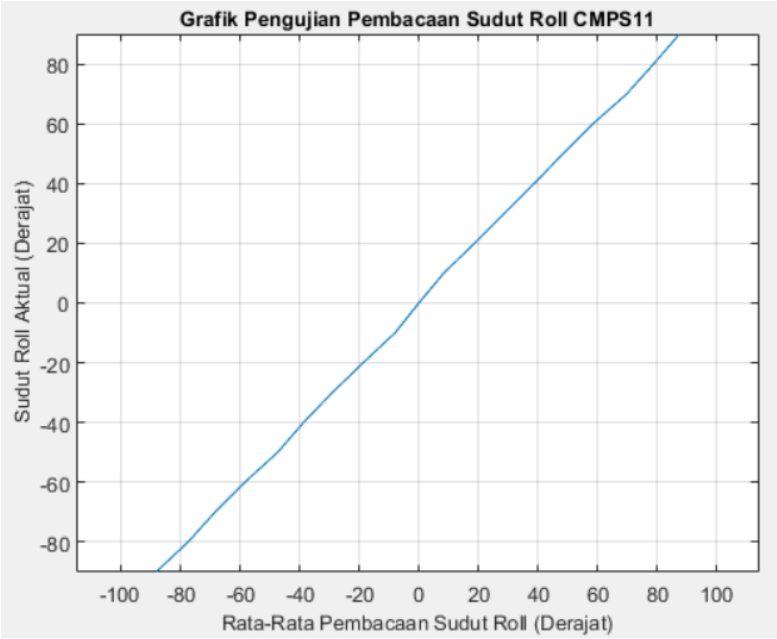
Data pengujian pembacaan sudut roll CMPS11 ditunjukkan dalam Tabel 3.3.

Tabel 3.3

Data Pengujian Pembacaan Sudut Roll CMPS11

No	Sudut Aktual (Derajat)	Pembacaan Sudut (Derajat)			Rata-Rata Pembacaan Sudut (Derajat)	Rata-Rata Error (Derajat)
		1	2	3		
1	-90	-89	-88	-87	-88	-2
2	-80	-78	-77	-77	-77.333	-2.667
3	-70	-68	-69	-68	-68.333	-1.667
4	-60	-57	-59	-59	-58.333	-1.667
5	-50	-47	-48	-47	-47.333	-2.667
6	-40	-38	-39	-39	-38.667	-1.333
7	-30	-29	-28	-30	-29	-1
8	-20	-19	-18	-19	-18.667	-1.333
9	-10	-8	-9	-7	-8	-2
10	0	0	0	0	0	0
11	10	7	9	9	8.333	1.667
12	20	19	18	19	18.667	1.333
13	30	29	29	28	28.667	1.333
14	40	39	38	39	38.667	1.333
15	50	47	49	49	48.333	1.667
16	60	58	57	60	58.333	1.667
17	70	71	68	70	69.667	0.333
18	80	78	79	79	78.667	1.333
19	90	87	88	87	87.333	2.667

Data pengujian pembacaan sudut roll CMPS11 direpresentasikan melalui grafik pada Gambar 3.4.



Gambar 3.4 Grafik Pengujian Pembacaan Sudut Roll CMPS11

4. Pengujian Pembacaan Sudut *Yaw* CMPS11 *Magnetic Compass*

a. Tujuan

Pengujian pembacaan sudut yaw bertujuan untuk menguji apakah CMPS11 yang telah dikalibrasi telah layak untuk dipakai dalam pengujian sistem.

b. Peralatan yang Digunakan

1) LiPo (Lithium Polymer) Battery 3S 11.1 V 2200 mAh.

2) Quadruped Robot Electronics Board

3) *CMPS11 Magnetic Compass*

4) *USB-A Male to USB Mini-B Male cable*

5) Laptop

6) Kabel penghubung

7) Penggaris busur derajat

c. Langkah Pengujian

1) Siapkan program yang berfungsi untuk membaca nilai yaw dari CMPS11

2) *Upload* program tersebut ke *STM32F407VGT6*

3) Hubungkan baterai ke rangkaian elektronik robot

4) Nyalakan robot dengan menekan *power switch*

5) Ukur nilai pembacaan yaw pada setiap sudut menurut *axis z* dengan memberikan sudut pitch 0º (CMPS11 dalam keadaan level menurut *axis x*)

serta sudut roll 0º (CMPS11 dalam keadaan level menurut *axis y*) Percobaan dimulai dari sudut 0º hingga 350º dengan interval pembacaan 10º

6) Amati, catat, dan analisis nilai pembacaan sudut *yaw*

Data pengujian pembacaan sudut yaw CMPS11 ditunjukkan dalam Tabel 3.4.

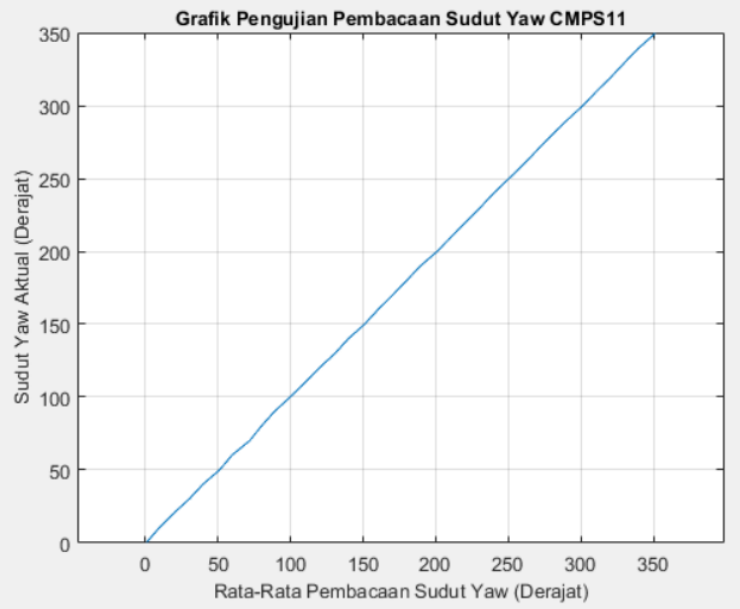
Tabel 3.4

Data Pengujian Pembacaan Sudut Yaw CMPS11

No	Sudut Aktual (Derajat)	Pembacaan Sudut (Derajat)			Rata-Rata Pembacaan Sudut (Derajat)	Rata-Rata Error (Derajat)
		1	2	3		
1	0	0	2.1	0.4	0.833	-0.833
2	10	8.8	9.5	10.1	9.467	0.533
3	20	19.2	18.7	20.4	19.433	0.567
4	30	31.5	30.1	29.5	30.367	-0.367
5	40	40.7	39.5	38.9	39.7	0.3
6	50	52.3	50.5	51.1	51.3	-1.3
7	60	59.1	60.4	59.6	59.7	0.3
8	70	72.4	71.5	71.9	71.93	-1.933
9	80	79.4	79.4	81.1	79.967	0.033
10	90	88.5	89.4	88.9	88.933	1.067
11	100	99.2	100.6	99.6	99.8	0.2
12	110	109.6	110.1	110.3	110	0
13	120	118.9	119.3	121.4	119.867	0.133
14	130	131.7	131.5	128.9	130.7	-0.7
15	140	139.1	139.8	140.5	139.8	0.2
16	150	151.6	150.6	150.8	151	-1
17	160	160.4	159.3	160.9	160.2	-0.2
18	170	169.3	171.2	170.4	170.3	-0.3
19	180	179.3	180.4	180.2	179.967	0.033
20	190	188.6	190.4	189.4	189.467	0.533
21	200	202.3	200.5	200.1	200.967	-0.967
22	210	211.2	210.6	210.3	210.7	-0.7
23	220	219.4	221.2	220.9	220.5	-0.5
24	230	231.4	229.6	230.4	230.467	-0.467

25	240	239.5	240.4	239.8	239.9	0.1
26	250	250.6	249.7	250.6	250.3	-0.3
27	260	260	261.3	260.5	260.6	-0.6
28	270	269.6	270.7	270.4	270.233	-0.233
29	280	279.1	280.1	280.9	280.033	-0.033
30	290	288.9	291.4	290.3	290.2	-0.2
31	300	301.2	301.5	300.7	301.133	-1.133
32	310	310.8	311.4	310.2	310.8	-0.8
33	320	320.5	321.6	320.6	320.9	-0.9
34	330	330.8	329.5	330.5	330.267	-0.267
35	340	340.2	340	340.1	340.1	-0.1
36	350	352.6	350.4	350.8	351.267	-1.267

Data pengujian pembacaan sudut yaw CMPS11 direpresentasikan melalui grafik pada Gambar 3.5.



Gambar 3.5 Grafik Pengujian Pembacaan Sudut Yaw CMPS11

3.3.3 Karakterisasi HC-SR04 Ultrasonic Rangefinder

a. Tujuan

Karakterisasi HC-SR04 bertujuan untuk mengetahui kelinieran dalam pembacaan jarak serta memastikan bahwa HC-SR04 ultrasonic rangefinder layak digunakan dalam pengujian keseluruhan sistem.

b. Peralatan yang Digunakan

1) LiPo (Lithium Polymer) Battery 3S 11.1 V 2200 mAh.

2) Quadruped Robot Electronics Board

3) USB-A Male to USB Mini-B Male cable

4) Kabel penghubung

5) HC-SR04

6) Objek penghalang

7) Penggaris

c. Langkah Pengujian

1) Upload program mikrokontroller yang berisi rutin untuk membaca jarak objek di depan modul HC-SR04

2) Hubungkan *output* tegangan *LiPo (Lithium Polymer) Battery* dengan *Quadruped Robot Electronics Board*

3) Nyalakan Robot dan jalankan instruksi pembacaan sensor HC-SR04

4) Atur jarak objek penghalang relatif terhadap sensor HC-SR04. Pastikan objek tegak lurus terhadap HC-SR04.

5) Catat hasil pembacaan sensor HC-SR04.

6) Lakukan percobaan serupa pada jarak 5 hingga 50 cm dengan interval 5 cm antar pengujian.

7) Amati, dokumentasikan, serta analisis hasil pengujian HC-SR04

Hasil pengujian sensor HC-SR04 *ultrasonic rangefinder* dijabarkan pada tabel 3.5

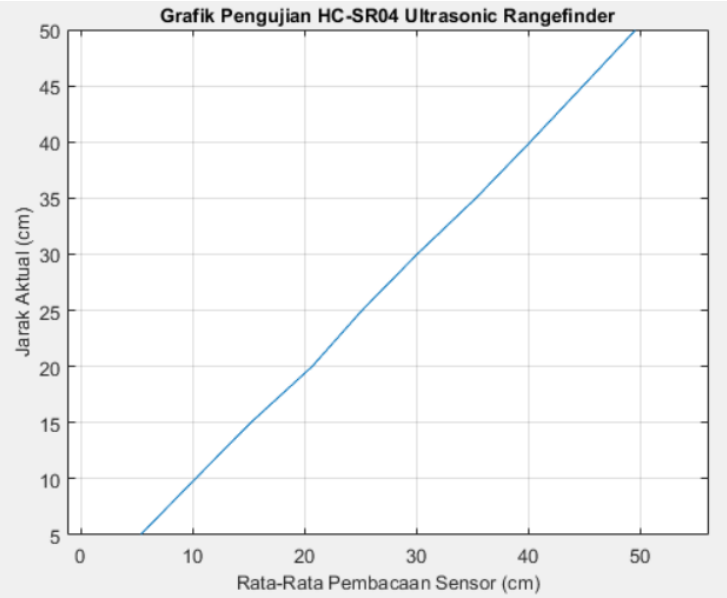
Tabel 3.5

Data Pengujian Pembacaan Jarak HC-SR04 *Ultrasonic Rangefinder*

No	Jarak Aktual (cm)	Hasil Pembacaan Sensor (cm)			Rata-Rata Pembacaan Sensor (cm)	Rata-Rata Error (cm)
		1	2	3		
1	5	5.48	5.41	5.25	5.38	-0.38
2	10	10.05	10.3	10.46	10.27	-0.27
3	15	15.2	15.18	15.28	15.22	-0.22
4	20	20.67	20.61	20.77	20.683	-0.683
5	25	25.17	25.27	24.96	25.133	-0.133
6	30	30.15	30.25	29.74	30.047	-0.047

7	35	35.15	35.42	35.3	35.29	-0.29
8	40	39.74	40.15	40.5	40.13	-0.13
9	45	44.17	45.44	44.94	44.85	0.15
10	50	49.41	49.69	49.53	49.543	0.457

Data pengujian pembacaan jarak pada HC-SR04 *ultrasonic rangefinder* direpresentasikan melalui grafik pada Gambar 3.6.



Gambar 3.6 Grafik Pengujian HC-SR04 *Ultrasonic Rangefinder*

3.3.4 Karakterisasi Transmisi Data

a. Tujuan

Karakterisasi transmisi data bertujuan untuk memastikan bahwa tidak ada error akibat transmisi data. Pengujian dilakukan dengan mengatur perangkat UART dengan *baudrate* 9600 bps dan melakukan pengiriman data *hexadecimal* bernilai 0xAA. Dikarenakan pengujian menggunakan *logic analyzer* yang menampilkan data berupa sinyal kotak sebagai representasi bilangan biner, maka untuk mengetahui apakah data yang dikirimkan sudah benar, data harus dikonversi menjadi bilangan biner terlebih dahulu. Dalam kasus ini, bilangan *hexadecimal* 0xAA dikonversi menjadi bilangan biner bernilai 0b10101010. Data diawali dengan *start bit* yang selalu berlogika *low* kemudian diikuti bit data ke 0 hingga ke 7 dan diakhiri dengan *stop bit* yang berlogika *high.*

b. Peralatan yang digunakan

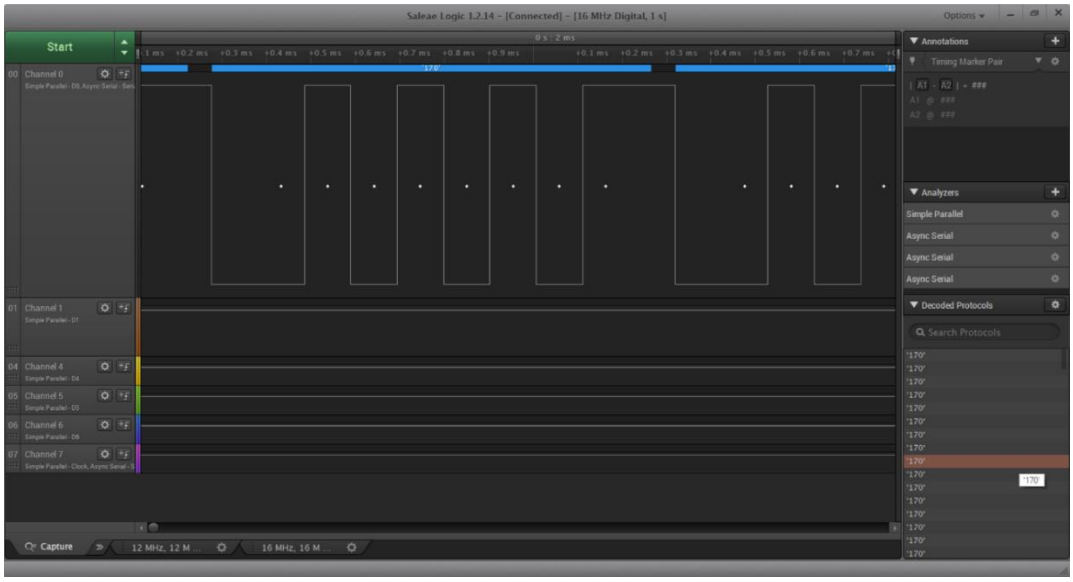
1) LiPo (Lithium Polymer) Battery 3S 11.1 V 2200 mAh.

2) *Quadruped Robot Electronics Board*
3) *USB-A Male to USB Mini-B Male cable*
4) Kabel penghubung
5) *Saleae USB Logic Analyzer*

c. Langkah Pengujian

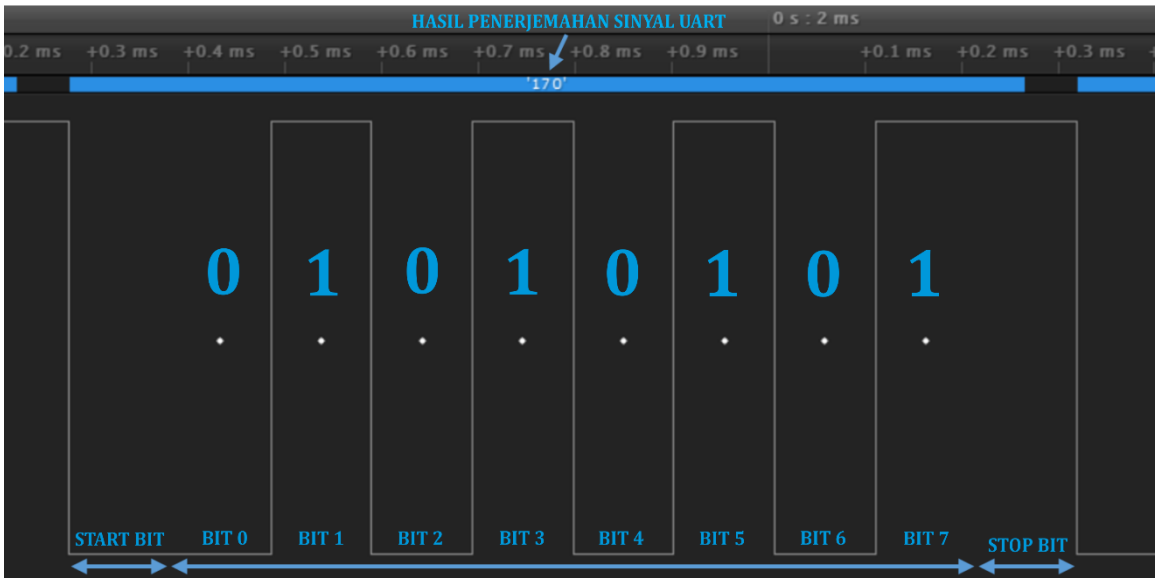
1) Upload program mikrokontroller yang berisi rutin untuk mengirimkan nilai 0xAA melalui protocol komunikasi UART dengan *baudrate* 9600 bps.
2) Hubungkan *output* tegangan *LiPo (Lithium Polymer) Battery* dengan *Quadruped Robot Electronics Board*
3) Hubungkan pin TX UART 1 (Pin B6) dari STM32F407VGT6 ke input signal *oscilloscope* serta GND dari STM32F407VGT6 ke GND *Saleae USB Logic Analyzer*.
4) Nyalakan Robot dan jalankan instruksi pengiriman nilai 0xAA
5) Amati, dokumentasikan, serta analisis sinyal yang dibaca oleh *Saleae USB Logic Analyzer*

Hasil pengujian dan tampilan software *Saleae USB Logic Analyzer* ditunjukkan pada Gambar 3.7



Gambar 3.7 Hasil Pengujian Sinyal UART

Pada gambar 3.8 dapat disimpulkan bahwa sebuah *frame* data diawali dengan *start bit* kemudian diikuti data 8 bit yang diawali LSB hingga MSB, serta diakhiri dengan *stop bit*. Data yang terbaca adalah 170, yang merupakan konversi *decimal* dari nilai *hexadecimal* 0xAA yang dikirimkan mikrokontroler. Berdasarkan pengujian dapat ditarik kesimpulan bahwa konfigurasi UART pada mikrokontroler telah sesuai dan siap dipakai dalam sistem.



Gambar 3.8 Analisis Sinyal UART

3.3.5 Karakterisasi *Interrupt Timer-Based Counter*

a. Tujuan

Ketika melakukan pengujian yang berhubungan dengan waktu eksekusi suatu *routine*, dibutuhkan ketelitian yang sangat tinggi, dikarenakan dibutuhkan kemampuan pembacaan hingga orde *millisecond* bahkan *microsecond*. Pengujian waktu eksekusi secara manual memberikan risiko yang besar terhadap terjadinya kesalahan paralaks. Maka dari itu, pada peneiltian ini pengujian waktu eksekusi dilakukan secara otomatis dengan menggunakan fitur *interrupt timer* pada mikrokontroler STM32F407VGT6. Fitur ini memungkinkan mikrokontroler untuk menjalankan suatu routine pencacah (counter) dengan frekuensi yang dapat diatur serta tidak mengganggu performa robot.

Untuk menghasilkan pengukuran waktu eksekusi yang presisi, maka pada penelitian ini dirancanglah *counter* dengan ketelitian 1 ms. *Counter* dibuat dengan mengatur konfigurasi *interrupt timer* dengan mengacu pada persamaan 3.2.

$$Interrupt\ Frequency = \frac{1}{Timer\ Period} \times \frac{Timer\ Clock}{Timer\ Prescaler+1} \quad \text{..... (3.2)}$$

Dengan diketahui bahwa,

1) Interrupt Frequency = 1000 Hz
2) Timer Clock = 84000000 Hz

Interrupt frequency merupakan nilai frekuensi dari interrupt yang diinginkan, karena diinginkan counter dengan periode 1 ms, maka dengan rumus f= 1/T maka didapatkan nilai f = 1000 Hz. Timer clock merupakan frekuensi timer dari mikrokontroler. Frekuensi timer 7

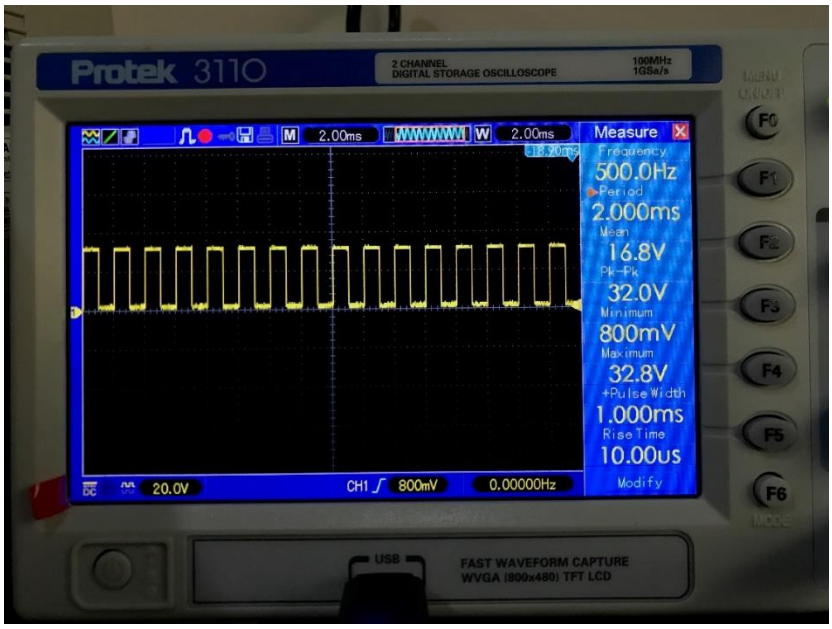
dari mikrokontroler STM32F407VGT6 adalah 84 MHz. Maka, berdasarkan rumus diatas, didapatkanlah nilai sebagai berikut:

1) Timer Prescaler = 3359
2) Timer Period = 25

Nilai Timer Prescaler dan Timer Period akan digunakan dalam konfigurasi *interrupt timer*. Pengujian dilakukan dengan memanfaatkan *interrupt routine* agar melakukan *toggle* pada pin GPIO C11 dari STM32F407VGT6. Kondisi pin GPIO C11 diamati dengan menggunakan *oscilloscope Protek 3110.*

b. Peralatan yang digunakan
    1) *LiPo (Lithium Polymer) Battery 3S 11.1 V 2200 mAh.*
    2) *Quadruped Robot Electronics Board*
    3) *USB-A Male to USB Mini-B Male cable*
    4) Kabel penghubung
    5) *Oscilloscope Protek 3110*
c. Langkah Pengujian
    1) Upload program mikrokontroller yang berisi konfigurasi dan aktivasi *interrupt timer*.
    2) Hubungkan *output* tegangan *LiPo (Lithium Polymer) Battery* dengan *Quadruped Robot Electronics Board*
    3) Hubungkan pin GPIO C11 kepada pin SIGNAL dari *oscilloscope* hubungkan pula GND dari STM32F407VGT6 kepada GND dari *oscilloscope*
    4) Nyalakan Robot dan jalankan instruksi
    5) Amati, dokumentasikan, serta analisis sinyal yang dibaca oleh *oscilloscope*

Hasil pengujian *interrupt timer based counter* ditunjukkan pada gambar 3.9. Diketahui bahwa logika dari Pin C11 mengalami perubahan setiap 1.000 ms, maka dari itu konfigurasi counter telah sesuai dengan rancangan awal dan layak dipakai dalam pengujian.



Gambar 3.9 Pengujian *Interrupt Timer Based Counter*

3.4 Pembuatan Perangkat Keras (*Hardware)*

Untuk membuktikan algoritma sistem pergerakan yang merupakan fokus pada penelitian ini, harus dibuat *quadruped robot platform* dengan ketentuan sebagai berikut:

1) Dimensi maksimal robot ketika bergerak dalam keadaan apapun tidak boleh melebihi 31 x 31 x 27 cm sesuai peraturan KRPAI 2017.
2) Rangka dari *quadruped robot* berbahan dasar plastik dan mika acrylic.
3) Mikrokontroler yang digunakan adalah STM32F407VGT6 pabrikan STMicroelectronics.
4) Pemrograman mikrokontroler STM32F407VGT6 dilakukan menggunakan CooCox IDE dan ditulis dalam bahasa C.
5) Aktuator yang digunakan adalah 12 unit Dynamixel AX-12A Smart Servo pabrikan Robotis.
6) Sensor yang digunakan sebagai umpan balik perpindahan robot adalah HC-SR04 *Ultrasonic Rangefinder*.
7) Sensor yang digunakan sebagai umpan balik *attitude* robot adalah CMPS11 *Magnetic Compass* produksi Devantech.
8) Catu daya elektrik yang digunakan adalah baterai *Lithium Polymer* 3S 11,1V 30C.
9) Komunikasi robot ke Dynamixel AX-12A menggunakan protocol komunikasi UART dengan baudrate 1 Mbps.

10) Komunikasi robot ke PC menggunakan *USB to TTL Converter* dan HC-05 *via Bluetooth* dengan *baudrate* 9600 bps.

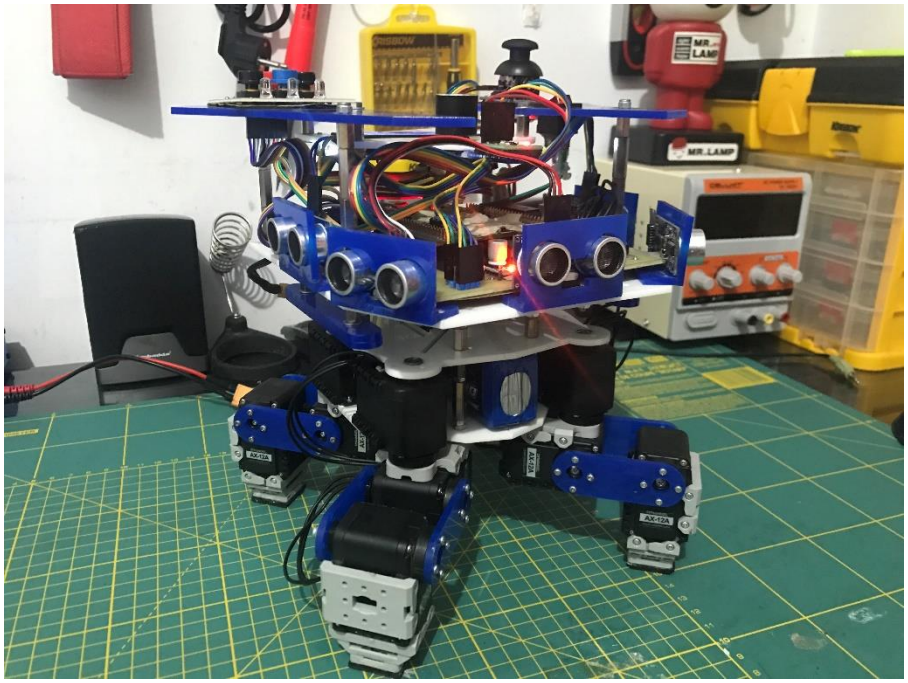
Pembuatan Perangkat Keras (*Hardware)* dibagi menjadi dua bagian, yaitu:

3.4.1 Pembuatan Mekanika *Quadruped Robot*

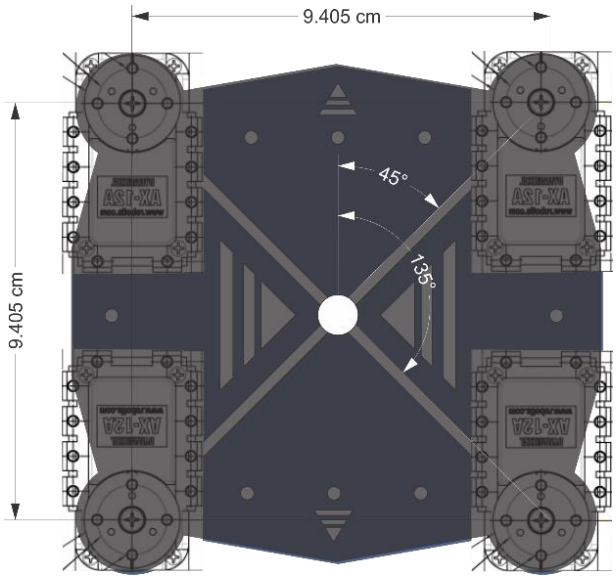
Sistem mekanika yang baik, merupakan fondasi yang mendukung pergerakan robot menjadi lebih baik pula. Oleh sebab itu perancangan mekanika robot harus proporsional dengan berat, panjang, lebar, serta tinggi dari robot. Agar beban robot tidak terlalu berat, maka rangka robot dibuat dari plastik dan mika acrylic. Berdasarkan Peraturan Kontes Robot Pemadam Api Indonesia (KRPAI) tahun 2017, batasan dimensi robot adalah sebagai berikut:

1) panjang maksimum = 31 cm.
2) lebar maksimum = 31 cm.
3) tinggi maksimum = 27 cm.

Dengan adanya peraturan tersebut, maka dimensi *quadruped robot* harus dirancang agar tidak melebihi batas aturan yang telah ditetapkan. Mekanika *quadruped robot* ditunjukkan pada Gambar 3.10.



Gambar 3.10 YUME *Quadruped Robot*



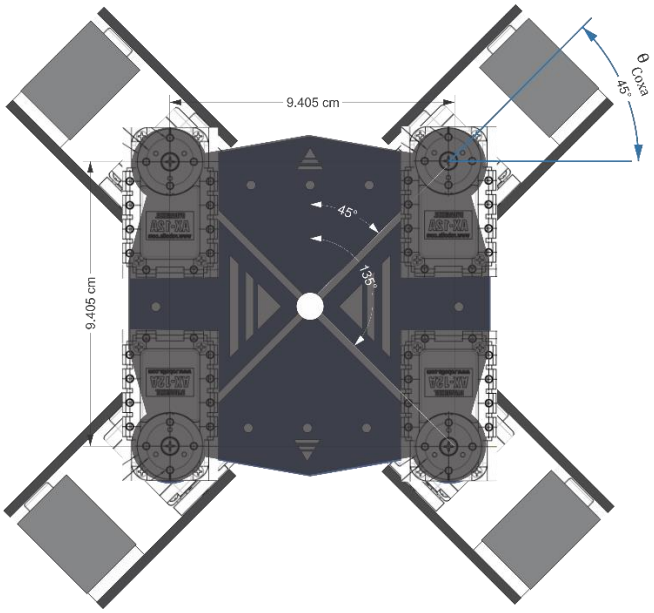
Gambar 3.11 Konstruksi Base Frame Quadruped Robot

Gambar 3.11 menunjukkan konstruksi base frame dan peletakan servo *coxa* dari *quadruped robot*. Titik pusat robot berada di tengah-tengah badan robot yang direpresentasikan dengan lingkaran berwarna putih. Besar sudut yang dibentuk oleh kaki depan terhadap titik pusat robot adalah sebesar 45º sedangkan sudut yang dibentuk oleh kaki belakang terhadap titik pusat robot adalah sebesar 135º. Jarak antar servo apabila diukur secara horizontal dan vertikal adalah 9.405 cm, sedangkan jarak antar servo apabila diukur secara diagonal adalah 13 cm.

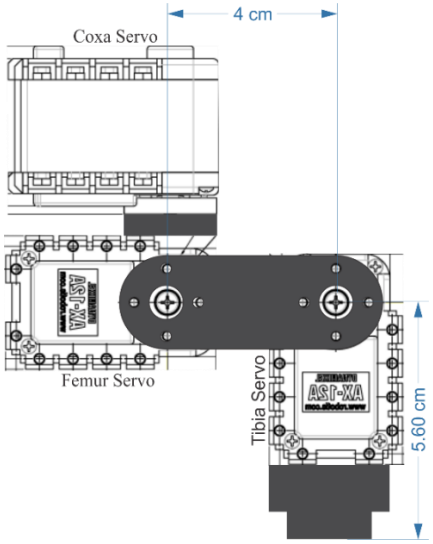
Jarak antar servo diatur sedemikian rupa agar didapatkan konstruksi robot yang seringkas mungkin, namun tetap mampu memenuhi kebutuhan sebagai berikut:

1) Masih memiliki ruang yang cukup untuk menyimpan *lipo battery* 3S 2200 mAh berukuran 3 cm x 10 cm.
2) Dapat dipasang minimal enam *spacer* penyangga rangka *robot* agar konstruksi robot lebih kokoh.
3) Jarak antar servo *coxa* tidak menghalangi pergerakan robot ketika berjalan

Konstruksi kaki robot apabila dilihat dari atas ditunjukkan pada Gambar 3.12. Sumbu X menjadi acuan titik nol dari sudut θ_{Coxa}. Apabila kaki mengayun ke depan θ_{Coxa} bernilai positif, sebaliknya θ_{Coxa} akan bernilai negatif apabila kaki mengayun ke belakang. Aturan tersebut diberlakukan untuk semua kaki.



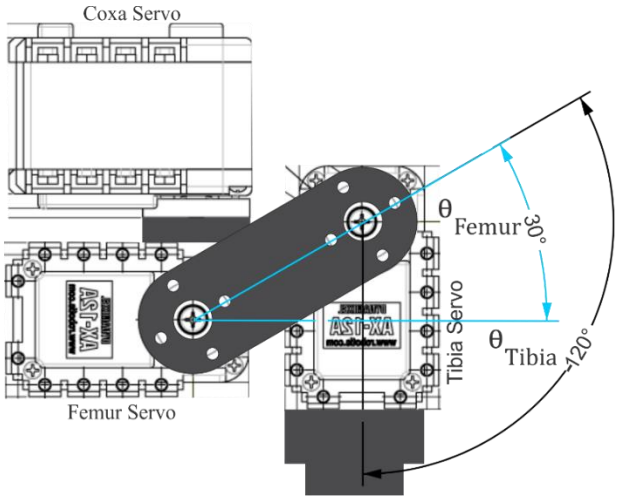
Gambar 3.12 Tampak Atas *Konstruksi Sistem Pergerakan Quadruped Robot*



Gambar 3.13 Tampak Samping Konstruksi Kaki *Quadruped Robot*

Gambar 3.13 menunjukkan konstruksi kaki *quadruped robot* apabila dilihat dari samping. Poros servo coxa dan servo femur sejajar dalam satu *axis Z,* maka dari itu nilai *coxa_offset* adalah 0 cm. Jarak antara poros servo femur dan poros servo tibia adalah 4 cm, yang akan kita sebut dengan *length_femur*. Sedangkan Jarak antara poros servo Tibia dan *end effector* adalah 5.60 cm, yang akan kita sebut dengan *length_tibia. Length_femur dan length_tibia* merupakan variable konstan yang akan dipakai pada kalkulasi *inverse kinematics*.

1) *coxa_offset* = 0 cm
2) *length_femur* = 4 cm
3) *length_tibia* = 5.60 cm

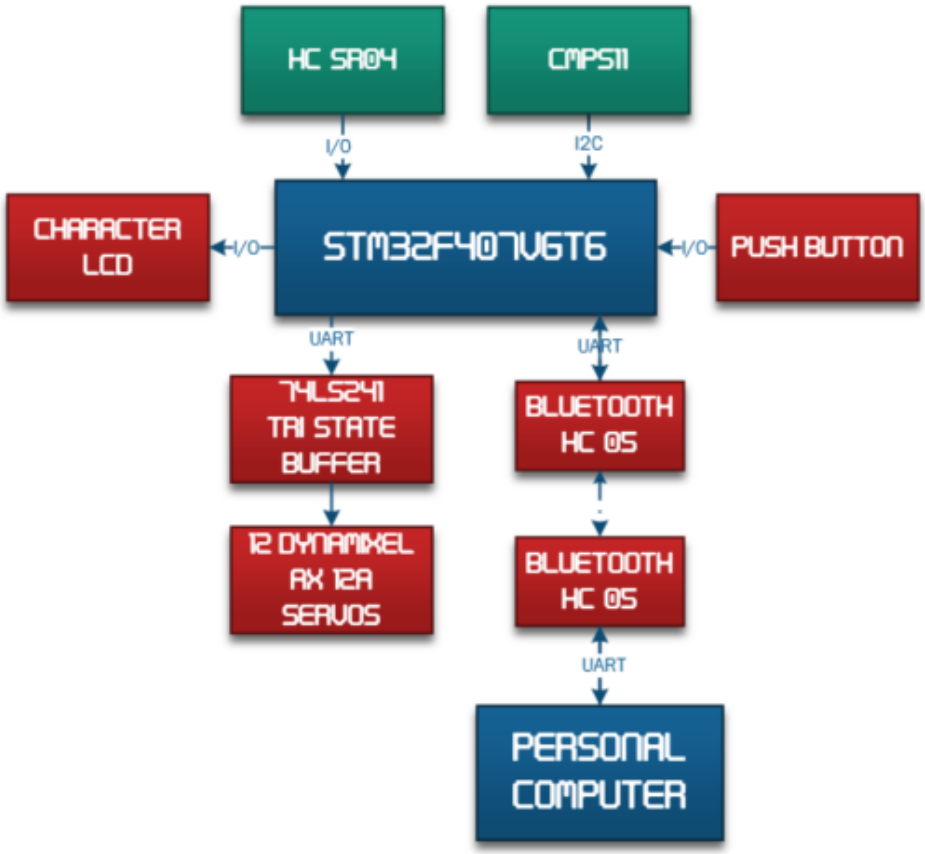


Gambar 3.14 Acuan Sudut *Femur* dan *Tibia Quadruped Robot*

Gambar 3.14 menunjukkan acuan sudut femur dan tibia *quadruped robot.* Acuan 0º dari sudut femur merupakan *axis X*. θ_{Femur} akan bernilai positif apabila femur bergerak ke atas dan akan bernilai negatif apabila bergerak ke bawah titik acuan. Acuan 0º dari sudut tibia merupakan sudut femur, yang menyebabkan nilai sudut acuan dari tibia harus selalu diperbaharui setiap kali dilakukan kalkulasi *inverse kinematics*. Θ_{Tibia} akan bernilai positif apabila tibia berada diatas femur dan bernilai negatif apabila tibia berada di bawah femur.

3.4.2 Pembuatan Elektronik *Quadruped Robot*

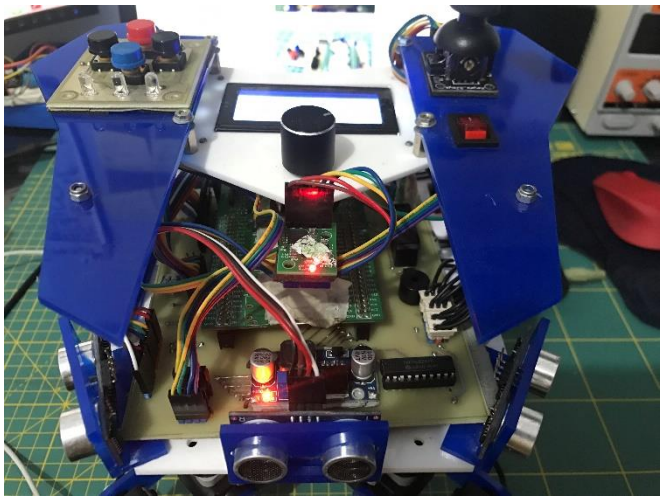
Gambar 3.15 Menampilkan diagram sistem elektronik *quadruped robot.* Dapat dilihat pada diagram blok sistem bahwa sebelum dihubungkan dengan Dynamixel AX-12A, pin UART dari STM32F407VGT6 harus dihubungkan ke Tri-state buffer 74LS241 untuk mengubah UART dari mikrokontroler menjadi *half-duplex* yang dibutuhkan oleh Dynamixel AX-12A. *Feedback* robot juga didapatkan dari CMPS11 untuk membaca orientasi *pitch, roll,* dan *yaw* robot. Robot dapat dikendalikan dengan menggunakan push button maupun instruksi serial *via* Bluetooth dari PC. Hasil pembacaan sensor kemudian ditampilkan melalui *LCD Character* dan dikirimkan ke PC *via* Bluetooth.



Gambar 3.15 Diagram Sistem Elektronik *Quadruped Robot*

Gambar 3.16 dan 3.17 menampilkan sistem elektronik *quadruped robot* setelah disolder dan dirangkai.

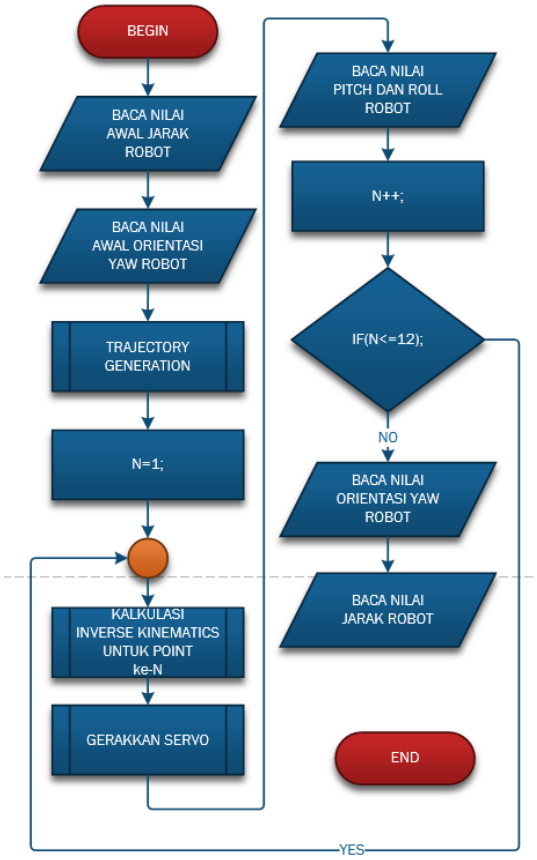
Gambar 3.16 Quadruped Robot Electronics Board



Gambar 3.17 Sistem Elektronik Quadruped Robot Setelah Dirangkai

3.5 Perancangan Algoritma dan Perangkat Lunak (*Software*)

Perancangan perangkat lunak (*software)* dibutuhkan agar mikrokontroler dapat mengendalikan sistem sesuai dengan yang diharapkan. Desain algoritma yang diterapkan ke dalam mikrokontroler STM32F407VGT6 ditunjukkan pada Gambar 3.18.



Gambar 3.18 Flowchart Program Utama Mikrokontroler *Quadruped Robot*

Proses diawali dengan pembacaan nilai awal jarak robot terhadap objek di depannya, kemudian dilakukan pula pembacaan orientasi robot untuk mengetahui nilai *pitch*, *roll*, dan

yaw robot ketika robot berdiri diam di bidang datar. Kemudian *Trajectory Generator* membangkitkan 12 koordinat *Cartesian* yang merupakan koordinat yang perlu dituju oleh setiap *end-effector* secara berurutan. Koordinat tersebut menjadi *input* dari kalkulasi *inverse kinematics* yang menghasilkan sudut tiap sendi servo agar *end-effector* mampu mencapai koordinat yang diinginkan, diselingi dengan pembacaan nilai orientasi robot untuk menguji deviasi *pitch* dan *roll* dari body robot ketika bergerak. Setelah semua servo selesai digerakkan, di akhir program dilakukan pembacaan nilai orientasi robot untuk mengetahui simpangan orientasi *yaw* dari robot serta jarak objek di depan robot untuk mengetahui sejauh mana robot telah bergerak relatif terhadap posisi awal.

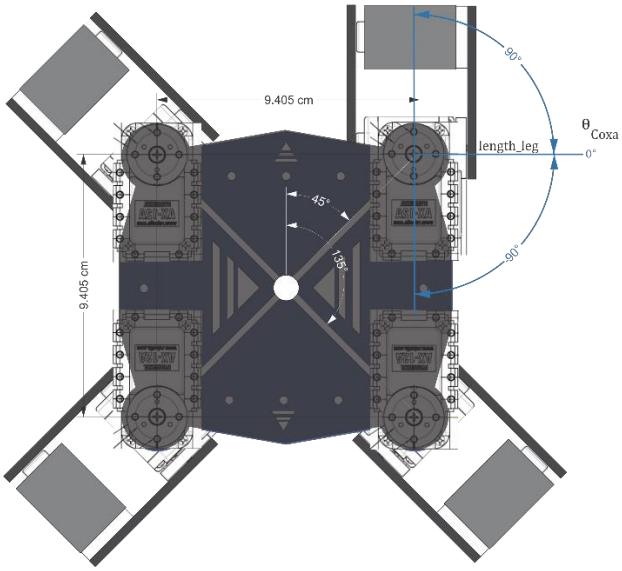
3.6 Perancangan Sistem Pergerakan *Quadruped Robot*

Dalam perancangan sistem pergerakan robot, proses dibagi menjadi empat proses, yaitu perancangan *workspace,* perancangan solusi *inverse kinematics,* perancangan *trot gait pattern,* dan perancangan *trajectory generator*. Masing-masing sub-proses perancangan kinematika tersebut dijabarkan sebagai berikut.

3.6.1 Perancangan Workspace Kaki *Quadruped Robot*

Untuk merancang workspace dari tiap kaki robot, maka problem kinematika tiga sendi dari *quadruped robot* dapat dipecah menjadi dua problem yang lebih sederhana, yaitu kinematika satu sendi untuk mencari workspace kaki robot terhadap sumbu X dan Y, serta kinematika dua sendi untuk mencari workspace kaki robot terhadap sumbu X dan Z. Workspace tiap kaki didapatkan dengan menggunakan *forward kinematics* dengan memperhitungkan semua sudut yang memungkinkan.

Seperti yang ditunjukkan pada Gambar 3.19, jangkauan sendi coxa *quadruped robot* adalah dari -90º hingga 90º. Dengan acuan titik 0º pada sumbu X. Dengan menggunakan *forward kinematics,* maka ada dua variabel yang mempengaruhi letak *end-effector* pada problem kinematika satu sendi diatas, yaitu *length_leg* dan θ_{Coxa}.



Gambar 3.19 Jangkauan Sudut dari Sendi Coxa *Quadruped Robot*

Jangkauan nilai length_leg adalah panjang kaki setelah sudut sendi femur dan sendi tibia *quadruped robot* diperhitungkan. Kita ketahui bahwa apabila nilai minimal dari length_leg didapatkan ketika sudut sendi tibia bernilai 0º dan bersamaan dengan sendi femur bernilai 90º atau -90º, yaitu 0 cm atau berada pada sumbu Z. Sedangkan nilai maksimal dari length_leg didapatkan ketika sudut sendi femur dan tibia bernilai 0º, yaitu 9,6 cm.

Dengan menggunakan persamaan trigonometri 3.3 dan 3.4, didapatkan koordinat *end_effector* terhadap sumbu X dan Y.

$X = leg\_length\ x \cos(\theta_{Coxa})$ ..... (3.3)

$Y = leg\_length\ x \sin(\theta_{Coxa})$ ..... (3.4)

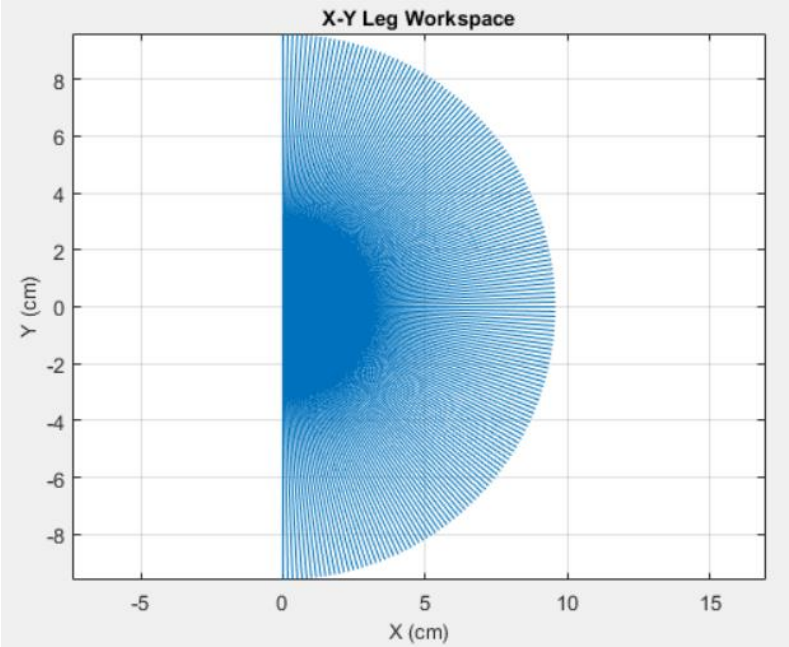
Dengan diketahui bahwa,

1) length_leg = 0 s/d 9.6 cm
2) θ_{Coxa} = -90º s/d 90º

Workspace X-Y didapatkan dengan menggunakan script MATLAB berikut:

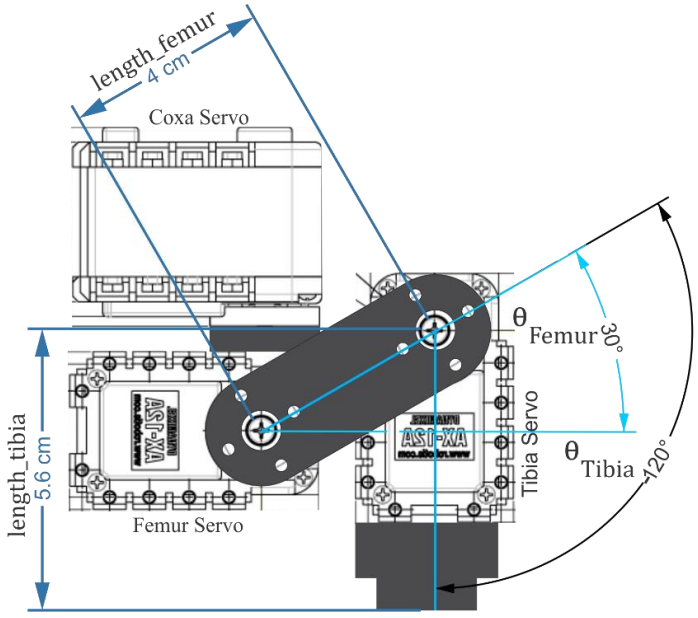
```
ThetaCoxa= [-90:1:90];
Leg_Length=[0:0.0532:9.6];
Leg_Length= Leg_Length.';

X= Leg_Length*cosd(ThetaCoxa);
Y= Leg_Length*sind(ThetaCoxa);

figure,
plot(X(:), Y(:));
axis equal;
xlabel('X (cm)','fontsize',10)
ylabel('Y (cm)','fontsize',10)
title('X-Y Leg Workspace','fontsize',10)
grid on;
```

Workspace kaki *quadruped robot* terhadap X dan Y ditunjukkan pada Gambar 3.20.



Gambar 3.20 X-Y Leg Workspace

Sesuai dengan Gambar 3.21, kita ketahui bahwa terdapat empat variabel yang mempengaruhi posisi *end-effector*, yaitu length_femur, length_tibia, θ_{Femur}, dan θ_{Tibia}. Kita asumsikan bahwa jangkauan dari θ_{Femur} dan θ_{Tibia} identik dengan θ_{Coxa}, yaitu -90º hingga 90º. Length_femur diperoleh dari jarak poros servo sendi *femur* hingga servo sendi *tibia*, yaitu 4 cm. Sedangkan length_tibia didapatkan dari jarak poros servo tibia hingga *end-effector*, yaitu 5,6 cm. Dengan menggunakan persamaan trigonometri 3.5 dan 3.6, didapatkan koordinat *end_effector* terhadap sumbu X dan Z.



Gambar 3.21 Problem Kinematika Dua Sendi *Quadruped Robot*

$$X = length\_femur\ x \cos(\theta_{Femur}) + length\_tibia\ x \cos(\theta_{Femur} + \theta_{Tibia}) \quad \ldots\ldots\ (3.5)$$

$$Z = length\_femur\ x \sin(\theta_{Femur}) + length\_tibia\ x\ sin(\theta_{Femur} + \theta_{Tibia}) \quad \ldots\ldots\ (3.6)$$

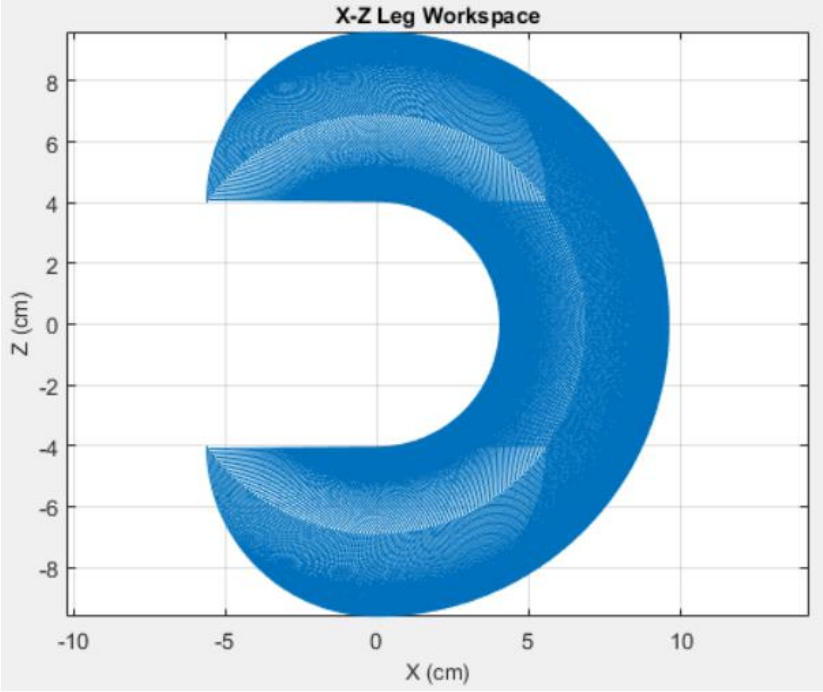
Dengan diketahui bahwa,

1) Length_femur = 4 cm
2) Length_tibia = 5,6 cm
3) θ_{Femur} = -90º s/d 90º
4) θ_{Tibia} = -90º s/d 90º

Workspace X-Z didapatkan dengan menggunakan script MATLAB berikut:

```
LengthFemur=4;
LengthTibia=5.6;

RAD2DEG= 57.296;
ThetaFemur=[-90:1:90];
ThetaTibia=[-90:1:90];

[ThetaFemur,ThetaTibia]= meshgrid(ThetaFemur, ThetaTibia);

X= LengthFemur*(cosd(ThetaFemur)) + LengthTibia*(cosd(ThetaFemur+ThetaTibia));
Z= LengthFemur*(sind(ThetaFemur)) + LengthTibia*(sind(ThetaFemur+ThetaTibia));

figure,
plot(X(:), Z(:));
axis equal;
xlabel('X','fontsize',10)
ylabel('Z','fontsize',10)
title('X Z Leg Workspace','fontsize',10)
grid on;
```

Workspace kaki *quadruped robot* terhadap X dan Z ditunjukkan pada Gambar 3.22.

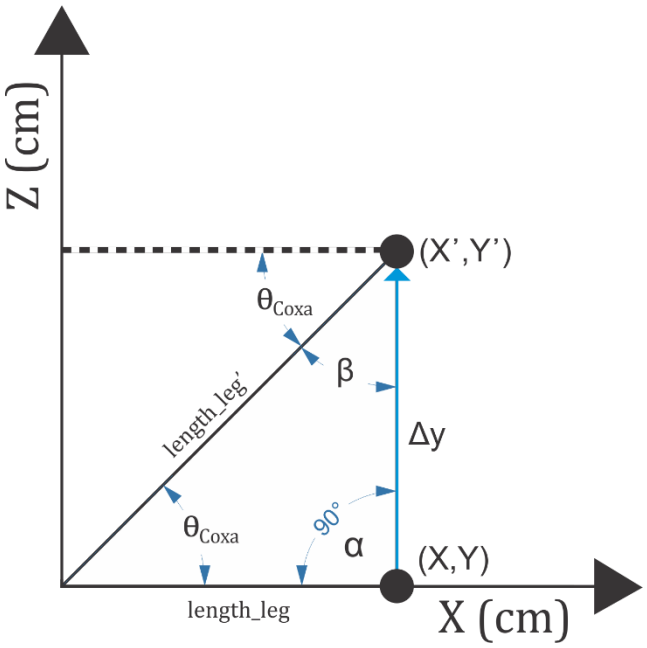


Gambar 3.22 X-Z Leg Workspace

3.6.2 Perancangan Solusi *Geometric Approach* Inverse Kinematics

Untuk menyelesaikan problem kinematika tiga sendi dari kaki *quadruped robot*, maka problem dipecah menjadi dua problem yang lebih sederhana, yaitu satu problem kinematika satu sendi untuk mencari nilai θ_{Coxa} dan satu problem kinematika dua sendi untuk mencari nilai θ_{Femur} dan θ_{Tibia}. Nilai θ_{Coxa} perlu kita cari dikarenakan berhubungan dengan jarak perpindahan *quadruped robot*, sedangakn nilai θ_{Femur} dan θ_{Tibia} berhubungan dengan seberapa tinggi *clearance* dari *quadruped robot* serta mengatur *swing phase* dan *support phase* dari *quadruped robot*.

Problem kinematika satu sendi yang bertujuan untuk mencari nilai sudut θ_{Coxa} ditunjukkan pada Gambar 3.23.



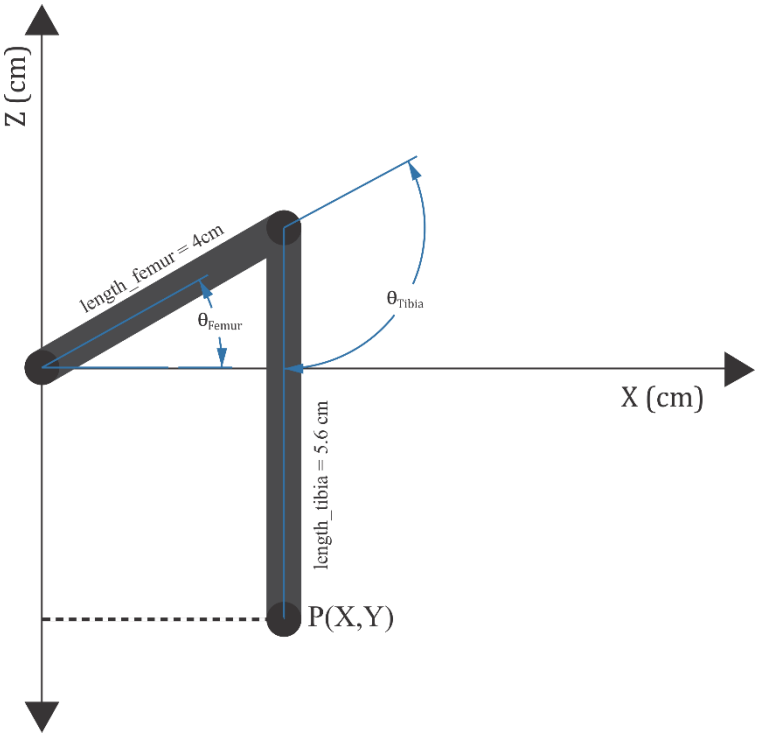
Gambar 3.23 Problem Kinematika Sendi Coxa

Pada problem mencari nilai θ_{Coxa} pada Gambar 3.23, diasumsikan masing-masing kaki sejajar dengan titik pusat robot, sehingga α bernilai 90°. Berdasarkan ilustrasi pada Gambar 3.23, didapatkan persamaan 3.7.

$$\theta_{Coxa} = tan^{-1}\left(\frac{Y}{X}\right) \quad \text{..... (3.7)}$$

Solusi pada persamaan 3.7 bertujuan untuk menyelesaikan problem kinematika pada kaki kanan *quadruped robo*t. Namun karena kaki kiri merupakan pencerminan dari sisi kanan, maka apabila pada kaki kanan servo sendi coxa harus bergerak sejauh 30° *counterclockwise*, maka pada kaki kiri servo sendi coxa harus bergerak sejauh 30° *clockwise.* Arah sudut putar servo diatur dengan menggunakan instruksi dari mikrokontroler *STM32F407VGT6* kepada *Dynamixel AX-12A Smart Servo*.

Problem kinematika dua sendi yang bertujuan untuk mencari nilai sudut θ_{Femur} dan θ_{Tibia} ditunjukkan pada Gambar 3.24. Length_femur bernilai 4 cm dan length_tibia bernilai 5,6 cm digunakan sebagai variabel konstan pada kalkulasi *inverse kinematics*.



Gambar 3.24 Problem Kinematika Sendi Femur dan Tibia

Berdasarkan problem kinematika pada Gambar 3.24, didapatkan persamaan 3.8 dan 3.9.

$$\theta_{Tibia} = cos^{-1}\left(\frac{|X|^2+Z^2-length\_femur^2-length\_tibia^2}{2\times length\_femur\times length\_tibia}\right) \quad \text{(3.8)}$$

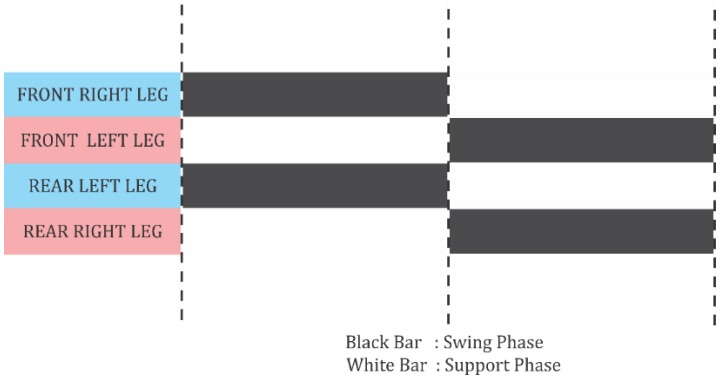
$$\theta_{Femur} = tan^{-1}\left(\frac{Z(length\_femur+length\_tibia\cos\theta_{Tibia})-(X\times length\_tibia\sin\theta_{Tibia})}{X\,(length\_femur+length\_tibia\cos\theta_{Tibia})-(Z\times length\_tibia\sin\theta_{Tibia})}\right). \quad \text{(3.9)}$$

Solusi pada persamaan 3.8 dan 3.9 bertujuan untuk menyelesaikan problem kinematika pada kaki kanan *quadruped robo*t. Namun karena kaki kiri merupakan pencerminan dari sisi kanan Arah sudut putar servo diatur dengan menggunakan instruksi dari mikrokontroler *STM32F407VGT6* kepada *Dynamixel AX-12A Smart Servo.*

3.6.3 Perancangan Trot Gait Pattern

Setelah mekanisme *quadruped robot* telah dibuat, diperlukan sebuah *gait pattern* agar *quadruped robot* mampu bergerak dengan baik. *Gait pattern* adalah suatu pola pergerakan masing-masing kaki yang dikoordinasikan dengan pergerakan tubuh robot baik itu translasi maupun rotasi yang dilakukan secara repetitif agar tubuh robot dapat bergerak dari satu tempat ke tempat lainnya (Pablo Gonzales et al, 2006).

Gait pattern yang diimplementasikan pada penelitian ini adalah *trot gait*. *Trot gait* adalah pola pergerakan dengan mengunakan dua kaki yang berada pada bidang diagonal satu sama lain untuk mengayunkan kaki (*swing phase)*, sedangkan dua kaki sisanya bertugas untuk menopang tubuh *quadruped robot (support phase)*. Ilustrasi *Trot Gait* ditunjukkan pada Gambar 3.25.



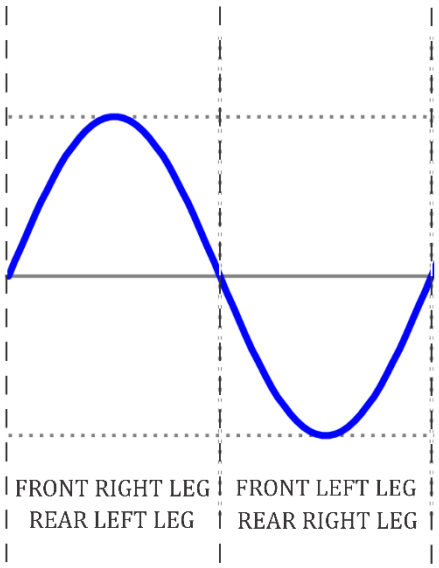
Gambar 3.25 Trot Gait Pattern

Pada Gambar 3.25, kotak berwarna hitam menggambarkan fase mengayunkan kaki ke depan (*swing phase)*, sedangkan kotak berwarna putih menggambarkan fase menopang tubuh untuk menjaga keseimbangan (*support phase)*. Pada *trot gait*, jarak yang ditempuh tiap kaki identik antar kaki satu dengan kaki lainnya. Pada satu siklus pergerakan, masing-masing-masing kaki akan mengalami satu kali *swing phase* dan satu kali *support phase* dengan timing sesuai yang dapat dilihat pada gambar 3.25.

Trot gait digunakan karena memiliki keunggulan dalam aspek kecepatan apabila dibandingkan dengan *creep gait*. Pada Kontes Robot Pemadam Api Indonesia (KRPAI) 2017, arena merupakan labirin dengan bidang datar dan tidak menggunakan aksesori lapangan *uneven floor*, sehingga *trot gait* merupakan *gait pattern* yang paling sesuai untuk digunakan dalam perlombaan.

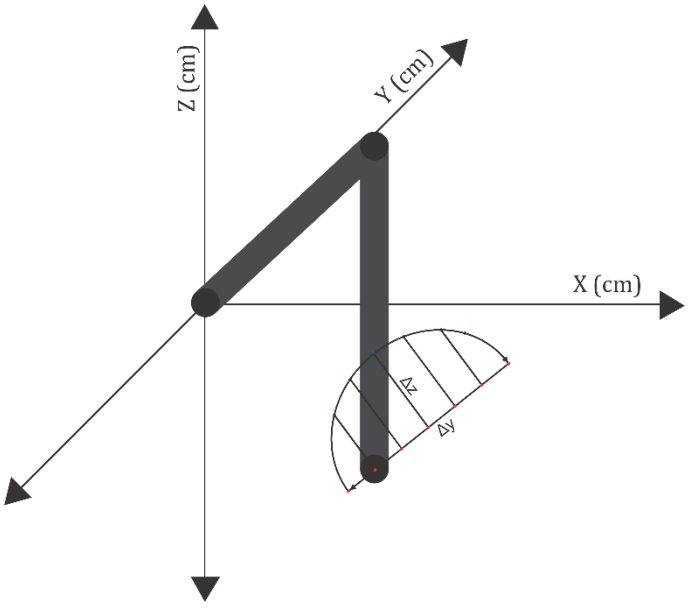
3.6.4 Perancangan *Trajectory Generator*

Trajectory Generator berfungsi untuk membangkitkan kumpulan koordinat tujuan *end-effector* sebagai lintasan gerak ketika dibutuhkan perpindahan *end-effector* dari titik awal ke titik tujuan. Hal itu bertujuan untuk memperhalus gerakan *quadruped robot* dan mengatur lintasan pergerakan kaki *quadruped robot* agar tidak melewati lintasan yang memungkinkan terjadinya tabrakan antar kaki robot maupun antara kaki dengan objek lainnya. Pola yang digunakan dalam perancangan *trajectory generator* dalam penelitian ini adalah *sinewave pattern* (gelombang sinus).



Gambar 3.26 Sinewave Pattern

Seperti yang ditunjukkan pada gambar 3.26, keempat kaki *quadruped robot* dibagi menjadi dua kategori menurut timing *swing phase* masing-masing. Kaki kanan depan dan kaki kiri belakang mengayun (*swing phase)* hanya ketika gelombang sinus bernilai positif, sedangkan kaki kiri depan dan kaki kanan belakang mengayun (*swing phase)* hanya ketika gelombang sinus bernilai negatif. Ilustrasi pergerakan kaki yang menggunakan *half-wave rectified sine trajectory pattern* bergerak sejauh Δy dan mengangkan setinggi Δz ditunjukkan pada gambar 3.27.



Gambar 3.27 Half-Wave Rectified Sine Trajectory Pattern

Pada problem yang ditunjukkan pada gambar 3.27, diketahui bahwa nilai X selalu konstan, dengan Δy dan Δz sebagai *input* dari *Trajectory Generator*. Agar didapatkan pola

lintasan seperti pada gambar 3.27, maka digunakan empat persamaan yang berbeda menurut sudut dari gelombang sinus.

- θ = 0° - 179°

$$Y = \left(\frac{Actual\_Distance \times \theta}{360}\right) cm \quad \text{..... (3.10)}$$

$$Z = \Delta Z \sin\theta \text{ cm} \quad \text{..... (3.11)}$$

- θ = 180° - 359°

$$Y = Actual\_Distance - \left(\frac{Actual\_Distance \times \theta}{360}\right) cm \quad \text{..... (3.12)}$$

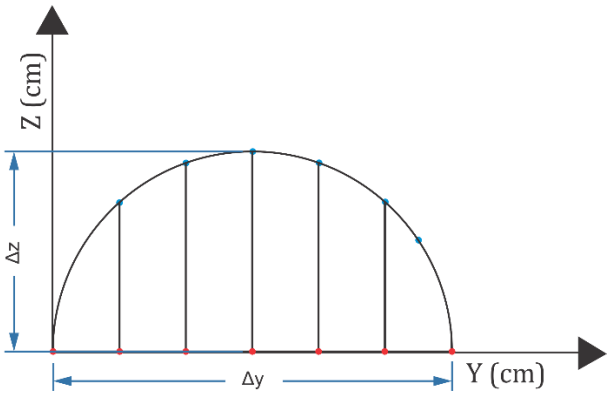
$$Z = 0\ cm \quad \text{..... (3.13)}$$

Dengan diketahui bahwa,

$$Actual\_Distance = 2 \times \Delta y\ cm$$

Actual_Distance merupakan jarak aktual dari perpindahan *quadruped robot*, dikarenakan Δy hanya mencakup satu *swing phase* dan *support phase*, sedangkan dalam satu iterasi dari *trot gait pattern*, terdapat dua kali *swing phase* dan *support phase*.

Setelah persamaan untuk mendapatkan nilai koordinat Y dan Z didapatkan, nilai interval antar θ_n harus ditentukan untuk mengatur resolusi pergerakan kaki. Dalam penelitian ini, digunakan interval 30° yang mencacah pola gelombang sinus menjadi 12 titik. Kumpulan koordinat Y dan Z yang dihasilkan oleh *trajectory generator* ditunjukkan pada Gambar 3.28. Titik berwarna biru menunjukkan koordinat tujuan ketika kaki mengayun (*swing phase)* sedangkan titik berwarna merah menunjukkan koordinat tujuan ketika kaki menopang (*support phase).*



Gambar 3.28 Output dari *Trajectory Generator*

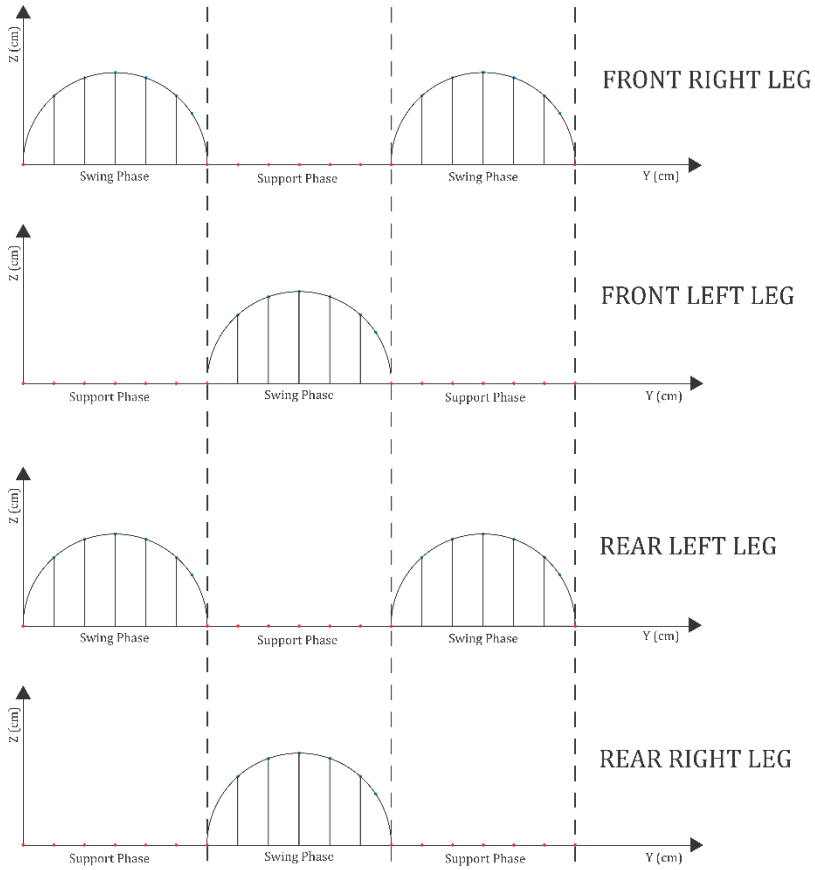
Tabel 3.6

Half-Wave Rectified Sine Pattern Trajectory Kaki *Quadruped Robot*

No	θ (Derajat)	X (cm)	Y (cm)	Z (cm)
1	0	4	-1.000	0.000
2	30	4	-0.667	0.500
3	60	4	-0.333	0.866
4	90	4	0.000	1.000
5	120	4	0.333	0.866
6	150	4	0.667	0.500
7	180	4	1.000	0.000
8	210	4	0.667	0.000
9	240	4	0.333	0.000
10	270	4	0.000	0.000
11	300	4	-0.333	0.000
12	330	4	-0.667	0.000

Tabel 3.6 menunjukkan 12 koordinat *end-effector* P(X, Y, Z) yang dihasilkan oleh *trajectory generator* dengan input Δy bernilai 2 cm, Δz bernilai 1 cm, serta X bernilai konstan 4 cm.

Output dari *trajectory generator* setelah dikombinasikan dengan algoritma *trot gait pattern* yang dibahas pada sub-bab 3.6.3 ditunjukkan pada Gambar 3.29.



Gambar 3.29 Kombinasi Trajectory Generator dan Trot Gait

Tabel 3.7 menunjukkan deskripsi pola pergerakan *quadruped robot* dengan menggunakan persamaan 3.10, 3.11, 3.12, dan 3.13. Pergerakan *quadruped robot* dibuat dengan menggunakan *trajectory generator* dan *trot gait pattern* yang telah dirancang pada sub-bab ini. *Quadruped robot* bergerak maju sejauh 2 cm setiap *swing phase* dan *support phase*, sehingga menghasilkan gerakan 4 cm dalam satu siklus pergerakan *trot gait pattern*. *Clearance* robot adalah 5,6 cm (Δz= -5,6 cm). x=4 dan berupa konstanta.

Tabel 3.7

Pola Pergerakan *Quadruped Robot*

No	Koordinat	Sudut
1	$\begin{bmatrix} (4, -1, -5{,}6) \\ (-4, 1, -5{,}6) \\ (4, 1, -5{,}6) \\ (-4, -1, -5{,}6) \end{bmatrix}$	$\begin{bmatrix} (59^o, 0^o, -89^o) \\ (121^o, 0^o, -89^o) \\ (31^o, 0^o, -89^o) \\ (149^o, 0^o, -89^o) \end{bmatrix}$
2	$\begin{bmatrix} (4, -0{,}667, -5{,}1) \\ (-4, 0{,}667, -5{,}6) \\ (4, 0{,}667, -5{,}6) \\ (-4, -0{,}667, -5{,}1) \end{bmatrix}$	$\begin{bmatrix} (54^o, 7^o, -96^o) \\ (126^o, 0^o, -89^o) \\ (36^o, 0^o, -89^o) \\ (144^o, 7^o, -96^o) \end{bmatrix}$

3	$\begin{bmatrix}(\ 4, -0{,}333, -4{,}734\ )\\(-4,\ 0{,}333,\ -5{,}6)\\(\ 4,\ 0{,}333,\ -5{,}6)\\(-4, -0{,}333, -4{,}734)\end{bmatrix}$	$\begin{bmatrix}(\ 50^o, 12^o, -101^o)\\(\ 130^o,\ 0^o,\ -90^o)\\(\ 40^o,\ 0^o,\ -90^o)\\(\ 140^o, 12^o, -101^o)\end{bmatrix}$
4	$\begin{bmatrix}(\ 4, 0, -4{,}6\ )\\(-4, 0, -5{,}6)\\(\ 4, 0, -5{,}6)\\(-4, 0, -4{,}6)\end{bmatrix}$	$\begin{bmatrix}(\ 45^o, 14^o, -103^o)\\(\ 135^o,\ 0^o,\ -90^o)\\(\ 45^o,\ 0^o,\ -90^o)\\(\ 135^o, 14^o, -103^o)\end{bmatrix}$
5	$\begin{bmatrix}(\ 4,\ 0{,}333,\ -4{,}734\ )\\(-4, -0{,}333,\ -5{,}6)\\(\ 4, -0{,}333,\ -5{,}6)\\(-4,\ 0{,}333,\ -4{,}734)\end{bmatrix}$	$\begin{bmatrix}(\ 40^o, 12, -101)\\(\ 140^o,\ 0,\ -90)\\(\ 50^o,\ 0,\ -90)\\(\ 130^o, 12, -101)\end{bmatrix}$
6	$\begin{bmatrix}(\ 4,\ 0{,}667,\ -5{,}1\ )\\(-4, -0{,}667, -5{,}6)\\(\ 4, -0{,}667, -5{,}6)\\(-4,\ 0{,}667,\ -5{,}1)\end{bmatrix}$	$\begin{bmatrix}(\ 36^o, 7, -96)\\(\ 144^o, 0, -89)\\(\ 54^o, 0, -89)\\(\ 126^o, 7, -96)\end{bmatrix}$
7	$\begin{bmatrix}(\ 4,\ 1,\ -5{,}6\ )\\(-4, -1, -5{,}6)\\(\ 4, -1, -5{,}6)\\(-4,\ 1,\ -5{,}6)\end{bmatrix}$	$\begin{bmatrix}(\ 31^o, 0, -89)\\(\ 149^o, 0, -89)\\(\ 59^o, 0, -89)\\(\ 121^o, 0, -89)\end{bmatrix}$
8	$\begin{bmatrix}(\ 4,\ 0{,}667,\ -5{,}6\ )\\(-4, -0{,}667, -5{,}1)\\(\ 4, -0{,}667, -5{,}1)\\(-4,\ 0{,}667,\ -5{,}6)\end{bmatrix}$	$\begin{bmatrix}(\ 36^o, 0, -89)\\(\ 144^o, 7, -96)\\(\ 54^o, 7, -96)\\(\ 126^o, 0, -89)\end{bmatrix}$
9	$\begin{bmatrix}(\ 4,\ 0{,}333,\ -5{,}6\ )\\(-4, -0{,}333, -4{,}734)\\(\ 4, -0{,}333, -4{,}734)\\(-4,\ 0{,}333,\ -5{,}6)\end{bmatrix}$	$\begin{bmatrix}(\ 40^o,\ 0,\ -90)\\(\ 140^o, 12, -101)\\(\ 50^o, 12, -101)\\(\ 130^o,\ 0,\ -90)\end{bmatrix}$
10	$\begin{bmatrix}(\ 4, 0, -5{,}6\ )\\(-4, 0, -4{,}6)\\(\ 4, 0, -4{,}6)\\(-4, 0, -5{,}6)\end{bmatrix}$	$\begin{bmatrix}(\ 45^o,\ 0^o,\ -90^o)\\(\ 135^o, 14^o, -103^o)\\(\ 45^o, 14^o, -103^o)\\(\ 135^o,\ 0^o,\ -90^o)\end{bmatrix}$
11	$\begin{bmatrix}(\ 4, -0{,}333,\ -5{,}6\ )\\(-4,\ 0{,}333,\ -4{,}734)\\(\ 4,\ 0{,}333,\ -4{,}734)\\(-4, -0{,}333,\ -5{,}6)\end{bmatrix}$	$\begin{bmatrix}(\ 50^o,\ 0^o,\ -90^o)\\(\ 130^o, 12^o, -101^o)\\(\ 40^o, 12^o,\ 101^o)\\(\ 140^o,\ 0^o,\ -90^o)\end{bmatrix}$
12	$\begin{bmatrix}(\ 4, -0{,}667, -5{,}6\ )\\(-4,\ 0{,}667,\ -5{,}1)\\(\ 4,\ 0{,}667,\ -5{,}1)\\(-4, -0{,}667, -5{,}6)\end{bmatrix}$	$\begin{bmatrix}(\ 54^o,\ 0^o, -89^o)\\(\ 126^o, 7^o, -96^o)\\(\ 36^o, 7^o, -96^o)\\(\ 144^o, 0^o, -89^o)\end{bmatrix}$