BAB III

PERANCANGAN DAN IMPLEMENTASI SISTEM

3.1 Tinjauan Desain Sistem

3.1.1 Desain Sistem Keseluruhan

- Pemilihan Komponen Robot Hectarus

Penggunaan komponen-komponen robot Hectarus dapat dilihat pada Tabel 3.1 berikut.

Tabel 3.1 Penggunaan Komponen Robot Hectarus

Komponen	Tipe	Jumlah	Spesifikasi
Mikroprosesor	Raspberry Pi 4B	1	5 Volt/3A
Sensor Jarak/Ultrasonik	HC-SR04	7	5 Volt/15mA
Sensor IMU	BMX160	1	3,3 – 5 Volt/1,6mA
Servo Driver	PCA9685	2	5 Volt/25mA
Servo	TD8120MG	14	4.8V/2.1A (Stall)
Servo	SONIC SN1030	4	4.8V/2.1A (Stall)
Konvertor	DC-DC Buck XL4015	1	5A/160Watt
Konvertor	DC-DC Buck	2	20A/300Watt
Baterai	Lithium Polymer	1	2S 45C 5200mAh

Robot Hectarus akan menggunakan mikroprosesor Raspberry Pi 4B. Raspberry Pi 4B digunakan untuk memproses algoritma-algoritma yang digunakan pada saat pengoperasian robot Hectarus. Selain itu untuk penelitian selanjutnya akan diimplementasikan LiDAR dan/atau *camera vision* yang tentu akan memerlukan daya kerja/proses komputasi yang lebih tinggi yang tidak dapat dilakukan oleh mikrokontroler seperti Arduino.

Penggunaan sensor jarak terdapat 2 pilihan yaitu sensor inframerah seperti *Time of Flight* (ToF) VL53L0X dan ultrasonik HC-SR04. Robot Hectarus menggunakan sensor ultrasonik HC-SR04 karena memiliki nilai minimum pendeteksi lebih rendah dari pada sensor ToF yaitu

sebesar 2 cm dan sensor ToF 3 cm. Selain itu, sensor HC-SR04 menggunakan protokol komunikasi biasa dengan menggunakan pin mikroprosesor sehingga potensi untuk terjadinya kehilangan komunikasi dapat lebih diminimalisasi. Berbeda dengan sensor ToF yang menggunakan I2C lebih rentan untuk mengalami koneksi yang terputus.

Sensor IMU yang digunakan adalah tipe BMX160 yang memiliki 9 *axis* (3 *axis magnetometer*, 3 *axis accelerometer* dan 3 *axis gyroscope*). Penggunaan sensor dengan tipe BMX160 bertujuan untuk mendapatkan nilai output yang lebih akurat karena memiliki 9 *axis* dan juga konsumsi daya-nya yang rendah.

Penggunaan Servo TD8120MG dan SONIC SN1030 dipilih berdasarkan hasil perhitungan torsi servo yang diperlukan. Penggunaan 2 tipe servo yang berbeda dengan spesifikasi torsi yang berbeda berguna untuk memberikan kemampuan torsi yang lebih tinggi pada bagian kaki robot Hectarus tertentu. Penjelasan lebih lanjut dapat dilihat pada bagian perhitungan servo.

Terdapat 2 jenis *DC-DC Buck Converter* yang digunakan yaitu 5A/160W (XL4015) dan 20A/300 Watt. Penggunaan 2 jenis konvertor digunakan berdasarkan spesifikasinya masing-masing. XL4015 digunakan untuk menurunkan tegangan dari baterai menjadi 5 Volt yang akan digunakan untuk menyuplai mikroprosesor atau Raspberry Pi 4B, 7 sensor ultrasonik dan modul servo *driver* PCA9685. Konvertor 20A/300 Watt digunakan untuk langsung menyuplai 18 servo yang digunakan pada robot Hectarus. Konvertor 20A/300 Watt yang digunakan berjumlah 2 untuk membagi beban arus pada *driver*-nya untuk memenuhi spesifikasi dari masing-masing modulnya. Masing-masing modul *driver* akan mengontrol 9 servo robot Hectarus. Dalam kasus terburuknya, servo akan menarik arus sebesar 2,1 Ampere dengan jumlah 9 servo yaitu 18,9 Ampere (94,5 Watt) dan nilai tersebut masih memenuhi spesifikasi konvertor yang digunakan. Jika semua jumlah servo hanya menggunakan 1 modul *driver* akan terjadinya malafungsi dan suhu yang tinggi sehingga dapat menyebabkan kerusakan komponen.

Spesifikasi Baterai Lithium Polymer ditentukan berdasarkan nilai kapasitas baterai dan nilai konstanta *C-rate* yang terdapat pada baterai tersebut. Parameter *C-rate* menunjukkan kemampuan baterai untuk melakukan *charge/discharge* [56]. Dengan kata lain, parameter *C-rate* menunjukkan batas output arus yang dapat diberikan/disuplai oleh baterai berdasarkan nilai *C-rate* dan besar kapasitas baterainya. Sebagai contoh, baterai 1000mAh dengan 2C menunjukkan bahwa baterai tersebut mampu menyuplai sebesar 2000 mAmpere dengan lama waktu 30 menit (perlu dilakukan pengisian ulang). Dari contoh tersebut, maka semakin besar nilai *C-rate* pada baterai maka semakin besar kemampuan baterai untuk dapat menyuplai robot.

Secara total, jumlah arus atau daya yang dibutuhkan oleh robot Hectarus dalam kondisi beroperasi normal adalah sekitar $\pm$ 7-10 Ampere/35-50 Watt dengan rating tegangan adalah 5 Volt. Karena robot Hectarus beroperasi pada rating tegangan 5 Volt maka pemilihan rating tegangan pada baterai adalah 7,4 Volt atau 2 *Cell*. Dengan rating tegangan 7,4 Volt, maka konsumsi arus pada baterai dengan rating daya 50 Watt adalah 6,7 Ampere. Penggunaan baterai pada robot diharapkan untuk setidaknya dapat menyelesaikan 1x putaran dari titik start robot hingga titik *finish*. Jika diasumsikan robot dapat menyelesaikan pergerakan dari titik start hingga *finish* adalah 15 menit, maka besar minimal kapasitas baterai robot adalah $6{,}7\ x\ 0{,}25h = 1675\text{mAh}$ dengan besar nilai *C-rate* minimum 4C.

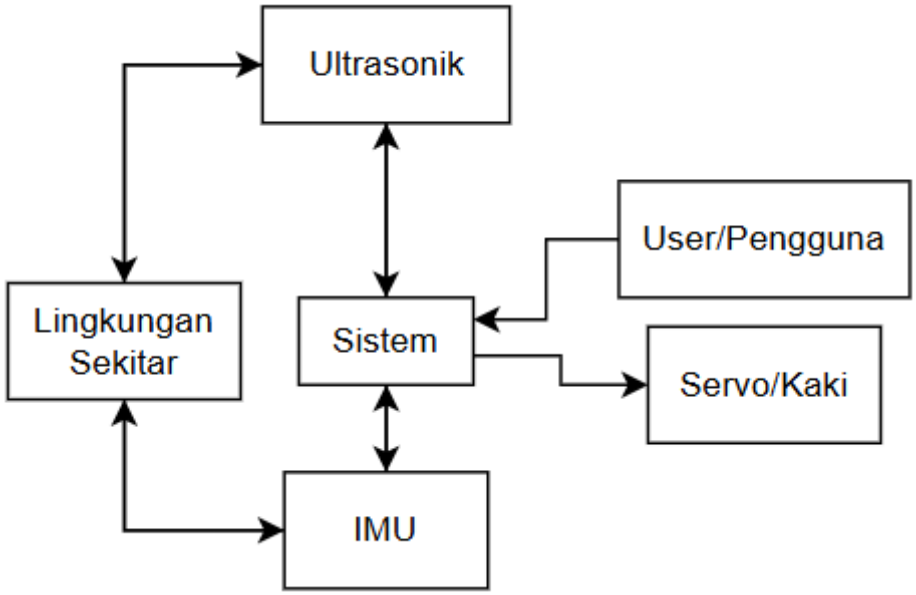
Untuk waktu pengoperasian yang lebih lama, maka ditetapkan besar kapasitas baterai robot adalah 5200mAh sehingga perkiraan lama waktu pengoperasi hingga baterai perlu dilakukan pengisian ulang adalah 46,6 menit. Dalam pengoperasian robot Hectarus, tidak menutup kemungkinan bahwa robot akan bekerja dalam kondisi *stall* sehingga akan mengonsumsi jumlah arus/daya yang lebih besar. Jika robot bekerja dalam kondisi terburuknya (semua servo dalam kondisi *stall*), maka nilai arus yang bekerja pada robot adalah $\pm$ 41 Ampere/205 Watt (5 Volt). Dengan rating daya tersebut, maka besar arus yang harus disuplai oleh

baterai adalah 27,7 Ampere. Dari nilai Ampere tersebut, maka nilai minimal *C-rate* pada baterai menjadi $\frac{27,7}{5,2} \approx 5,33 \approx 6C$.

Dalam implementasinya nilai *C-rate* yang digunakan pada robot Hectarus adalah 45-50 C. Nilai tersebut jauh dari hasil perhitungan ditujukan untuk memberikan nilai toleransi yang tinggi dan menghindari adanya kerusakan yang tidak diinginkan karena nilai *C-rate* yang terlalu dekat dengan nilai hasil perhitungan.

- Desain Kerangka/Fisik

Desain sistem keseluruhan sistem dapat dilihat pada gambar *Data Flow Diagram* berikut dengan penjalasannya.



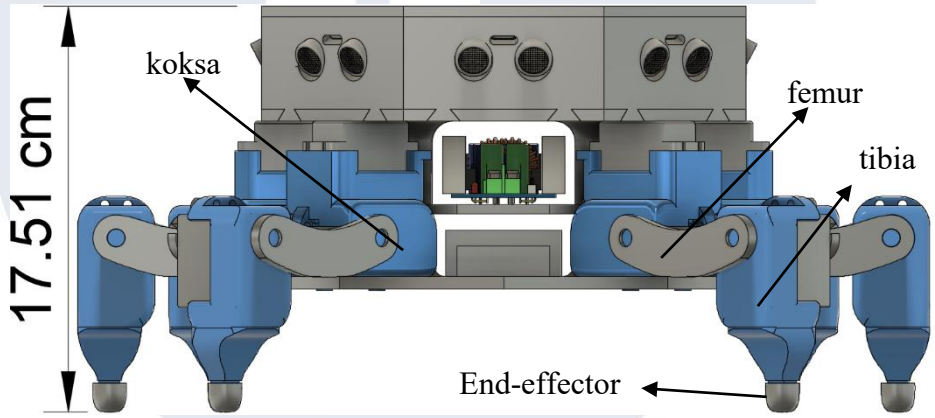
Gambar 3.1 *Data Flow Diagram* Desain Sistem Hectarus

Tabel 3.2 Penjelasan Input/*Output* Sistem

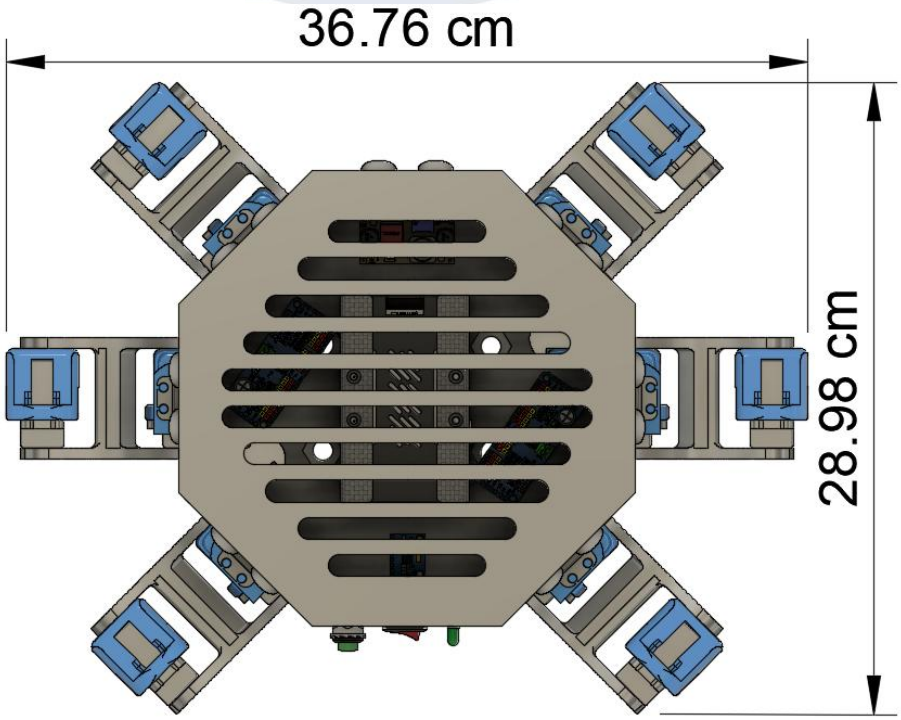
Parameter	Keterangan
Input	• Input tombol dari pengguna • Jarak halangan sekitar robot Hectarus terhadap Hectarus • Kondisi kemiringan dan hasil pergerakan robot Hectarus
Output	• Data jarak halangan robot Hectarus dengan sensor ultrasonik • Data kemiringan dan hasil pergerakan robot Hectarus dengan sensor IMU • Pergerakan robot Hectarus dengan pemicu start dari pengguna dan hasil pengolahan data dari sistem
Fungsi	• Inisialisasi awal pergerakan robot Hectarus • Mengetahui kondisi kemiringan dan jarak halangan robot Hectarus yang akan digunakan sebagai penentuan keputusan pergerakan Hectarus.

Sistem robot Hectarus akan terdiri dari komponen ultrasonik untuk mengetahui jarak antara robot dengan halangan disekitarnya, sensor IMU untuk mengetahui besar *error heading* dan kondisi kemiringan robot Hectarus, dan servo untuk melakukan pergerakan pada kaki. Input pertama dari robot Hectarus adalah input tombol start dari *user* yang digunakan untuk memulai pergerakan robot. Setelah tombol start ditekan, robot akan melakukan pengambilan data dari sensor ultrasonik untuk mengetahui jarak halangan disekitar untuk mengetahui hadapan robot yang seharusnya sehingga robot tidak tertabrak pada saat pergerakan. Sistem kemudian menghasilkan *output* proses pergerakan pada servo robot yang membuatnya bergerak. Pada saat bergerak, robot akan terus mengaktifkan dan mengambil data dari sensor IMU untuk mengetahui besar perubahan nilai *yaw* dan *pitch* pada robot. Pada saat pergerakan normal (lurus), robot diharapkan akan memiliki perubahan nilai *yaw* sebesar 0 sehingga pergerakannya akan lurus. Perubahan nilai pada *pitch* menandakan bahwa robot berada di permukaan yang menurun atau menanjak (tangga) yang memerlukan pergerakan, perubahan kaki atau *gait* khusus untuk melakukan pergerakan (jika dibutuhkan).

Seluruh desain fisik robot Hectarus akan dirancang dengan cetak 3D dengan bahan material PLA+ dengan *infill density* pada *base* (tutup, plat atas, plat bawah dan lainnya) adalah 30%, *link* koksa/paha dan tibia/betis adalah 50%, *link* femur/dengkul adalah 80%, *end-effector* adalah 80%. Nilai *infill density* pada femur yang lebih tinggi dikarenakan *link* tersebut yang memiliki beban paling besar untuk menopang keseluruhan tubuhnya ketika sedang beroperasi. Desain fisik sistem robot Hectarus dapat dilihat pada gambar berikut.

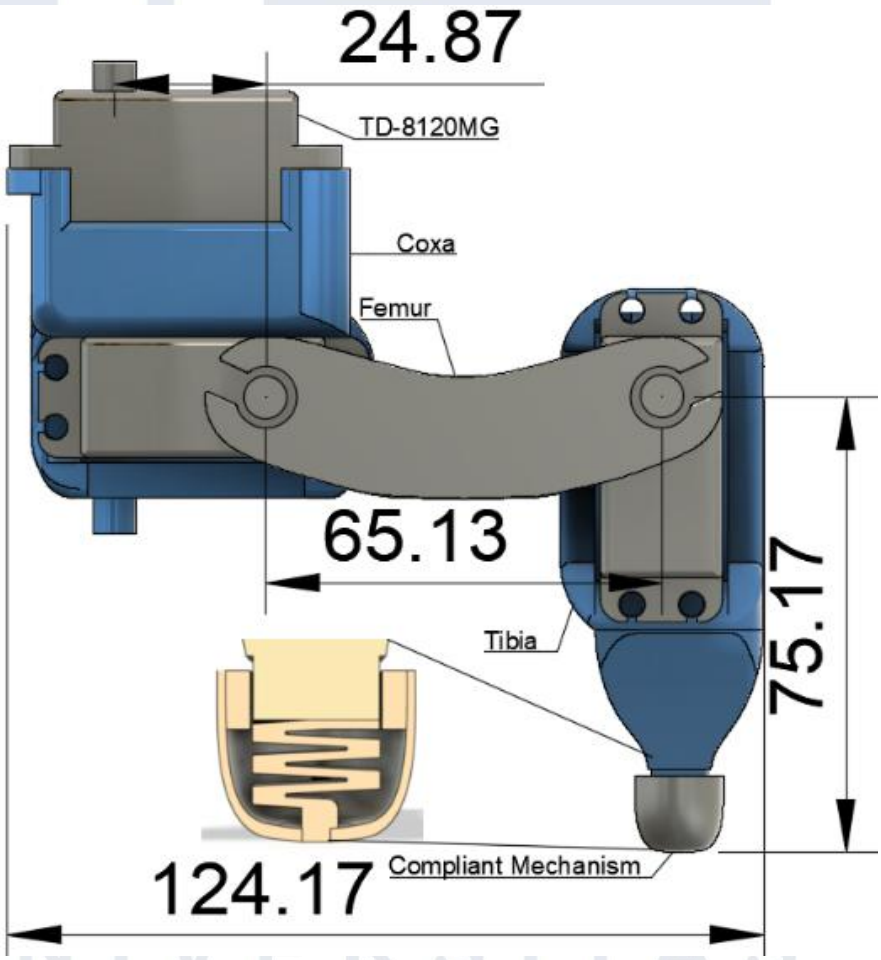


Gambar 3.2 Tampak Depan Robot Hectarus



Gambar 3.3 Tampak Atas Robot Hectarus

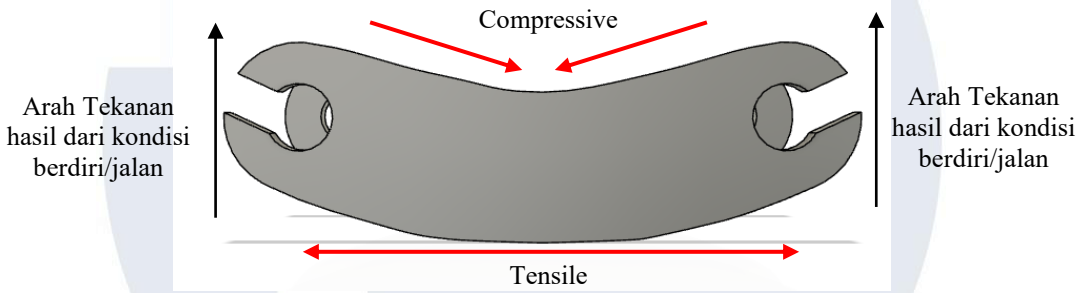
Robot Hectarus memiliki dimensi panjang x lebar x tinggi adalah 36,8 cm x 29 cm x 17,5 cm. Pada sistem pergerakan robot, terdapat 6 kaki yang terdiri dari koksa, femur, dan tibia. Pada *end-effector* di tibia dirancang sebuah *vibration isolator* berupa pegas yang dicetak 3D dengan material PETG. Pegas ini yang akan menjadi titik kontak kaki robot dengan permukaan yang akan dilewati robot, sehingga getaran dan guncangan akibat langkah kaki robot bisa diminimalisasi. Getaran yang diminimalisasi ini dapat meningkatkan akurasi pergerakan robot dan meminimalisasi kerusakan komponen [57].



Gambar 3.4 Tampak Samping Kaki Robot Hectarus

Pada Gambar 3.4, femur pada robot Hectarus dibuat sedikit melengkung ke atas. Hal ini bertujuan untuk menanggulangi kekurangan ketahanan tarikan (*tensile strength*) pada material PLA+. Material PLA+ memiliki kekuatan tarik dan dorong (*tensile-compressive strength*) berkisar dari 39.9 MPa – 52.5MPa untuk kekuatan tarik dan 48.2 MPa –

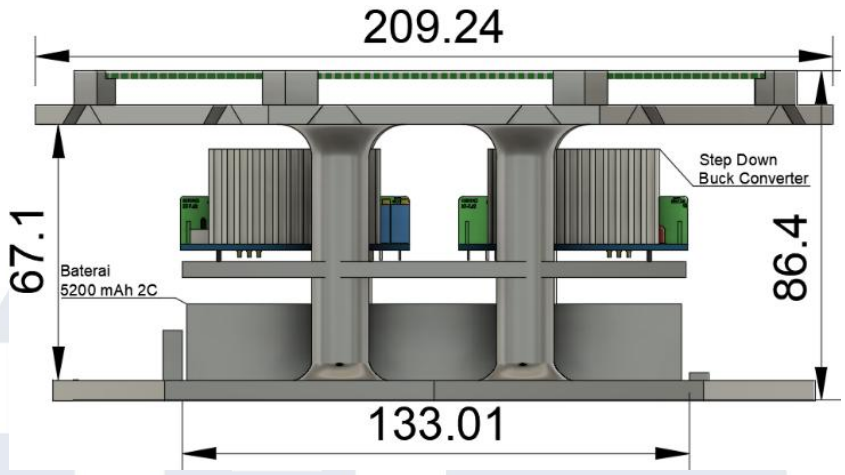
62 MPa untuk kekuatan dorongnya [58]. Kekuatan tarik dan dorong material PLA bervariasi bergantung pada persentase ketebalan material yang digunakan dalam pencetakan (*infill density*), kecepatan percetakan (*print speed*), ketebalan layer (*layer height*), dan warna material yang digunakan [58], [59], [60].



Gambar 3.5 Ilustrasi Arah Tekanan pada Femur

Pada kondisi berjalan dan berdiri, femur akan menerima tarikan dan dorongan seperti yang diilustrasikan pada Gambar 3.5. Gaya tarikan pada femur terjadi pada bagian bawah dan gaya dorongan akan terjadi pada bagian atas. Karena material PLA memiliki kekuatan tarikan lebih rendah daripada kekuatan dorongan, maka sesuai dengan rumus gaya dan tekanan, dengan asumsi tekanan yang sama, area bagian bawah femur harus lebih besar daripada area bagian atas femur, sehingga bentuk femur pada robot Hectarus dibuat melengkung kebawah untuk area bagian bawah lebih besar.

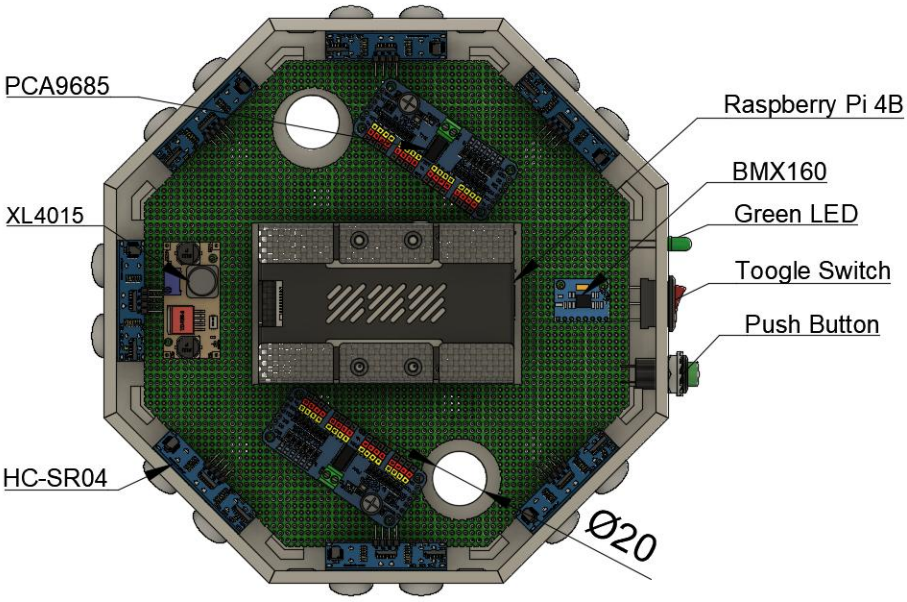
Pada Gambar 3.4 peletakan servo untuk sendi koksa dan femur dibuat bertumpuk. Hal ini bertujuan untuk mengurangi panjang kaki keseluruhan tanpa mengorbankan fleksibilitas pergerakan kaki robot. Dapat dilihat panjang dari koksa adalah 2,49 cm, panjang femur 6,5 cm, dan panjang tibia 7,5 cm sehingga panjang total kaki adalah 12,4 cm. Panjang kaki pada robot menjadi parameter yang penting karena semakin panjang kaki robot, selain dapat membuat dimensi robot semakin besar, gaya momen yang dirasakan servo pada kaki juga akan semakin besar.



Gambar 3.6 *Data Flow Diagram* Subsistem Pemeriksaan Halangan

Pada badan robot Hectarus bagian tengah diisi 3 buah komponen, yaitu 1 buah baterai 2 *Cell* kapasitas 5200 mAh dan 2 buah *DC-DC Step Down Converter* 20A. Baterai diletakkan di tengah badan robot ditujukan untuk baterai dapat diakses dengan mudah tanpa membuka bagian lain badan robot. Disisi lain, baterai merupakan salah satu komponen elektronik paling berat, maka baterai akan diletakkan serendah mungkin untuk membuat *center of gravity* dari robot bisa lebih rendah sehingga robot bisa menjadi lebih stabil. Selain itu *Step Down Converter* diletakkan berdekatan dengan baterai untuk memudahkan proses *wiring* pada robot sehingga tidak dibutuhkan kabel *power* yang tidak terlalu panjang. Tinggi dari pilar yang memisahkan pelat badan atas dan pelat badan bawah disesuaikan dengan tinggi dari koksa. Hal ini dilakukan agar badan atas dan badan bawah dapat menjadi tempat kaki robot.

Gambar 3.7 Bagian Badan Atas Robot Hectarus



Gambar 3.8 Tampak Atas Isi Badan Atas Robot Hectarus

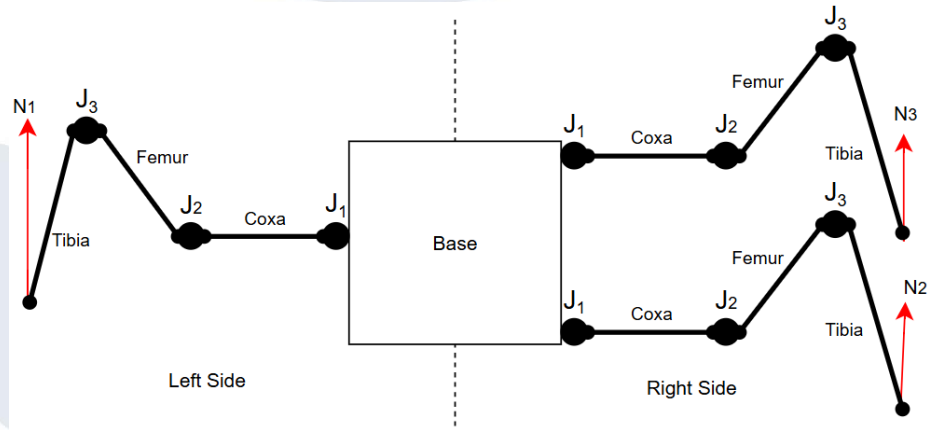
Bagian atas robot Hectarus merupakan tempat komponen-komponen integrasi robot seperti Raspberry Pi 4B, PCA9685 dan sensor-sensor seperti BMX160, dan Ultrasonik HC-SR04 serta komponen tambahan seperti *Step-down* 5V XL4015 untuk suplai listrik Raspberry Pi, LED untuk mengetahui *state* dari robot, *toggle switch* untuk saklar hidup-mati robot, dan *push button* untuk menjalankan robot.

Pada badan atas juga terdapat 2 lubang dengan diameter 2 cm sebagai akses kabel dari badan atas menuju badan bawah. Lubang ini akan menjadi akses kabel servo maupun kabel untuk kebutuhan daya dari robot. Tutup dari badan atas dibuat dengan ventilasi yang memanjang mengikuti bentuk dari badan atas. Ventilasi dibuat agar udara dalam badan atas robot dapat bergerak keluar dan masuk sehingga suhu komponen tetap terjaga.

Terdapat 7 buah sensor ultrasonik HC-SR04 yang dipasang pada masing-masing sisi badan atas robot (kecuali bagian belakang). Peletakan sensor ultrasonik HC-SR04 dimiringkan yang bertujuan untuk mengantisipasi kesalahan pendeteksi halangan yang rendah karena peletakan sensor ultrasoniknya yang terlalu tinggi.

- Perhitungan Torsi Servo

Perhitungan torsi dilakukan untuk mengetahui jenis servo yang akan digunakan berdasarkan nilai torsinya. Secara garis besar, robot Hectarus dapat diilustrasikan pada Gambar 3.9.



Gambar 3.9 Diagram Kinematik Hectarus dengan Tripod *Gait*

Perhitungan Torsi dimulai dari mencari nilai gaya normal pada ujung kaki kiri pada saat posisi berdiri dengan 3 kaki berada di atas (saat akan berjalan dengan Tripod *gait*). Robot diasumsikan memiliki bentuk yang simetris sehingga nilai gaya normal pada $N_2 = N_3$ dan nilai N_1 akan lebih besar.

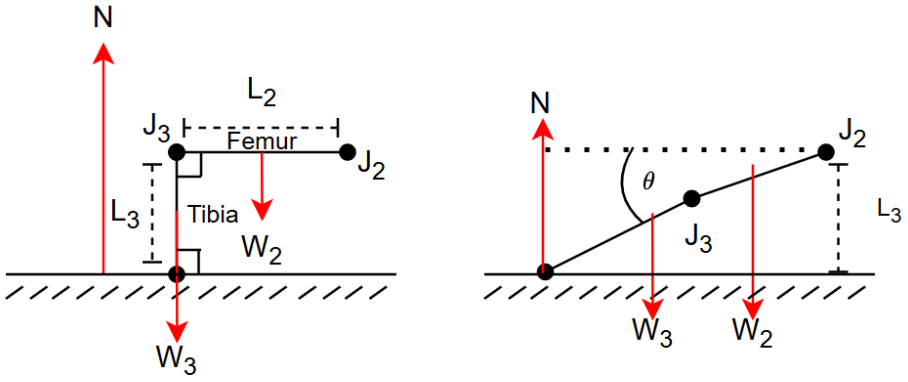
$$N_1 + 2N_2 = W_b + 6 * (W_1 + W_2 + W_3) \quad (1)$$

Setelah dilakukan pencetakan, dilakukan penimbangan dan pengukuran. Parameter berat dan panjang pada masing-masing *link* robot Hectarus adalah sebagai berikut.

Tabel 3.3 Parameter Berat dan Panjang *Link* Hectarus

Link	Panjang (meter)	Massa (kg)	Gaya Berat (W)
Base	0,156	2,873	28,1554
Koksa	0,02644	0,172	1,6856
Femur	0,06436	0,025	0,245
Tibia	0,08122	0,102	0,9996

Perhitungan torsi minimum yang dibutuhkan oleh masing-masing *joint* pada robot Hectarus dilakukan dengan kondisi terburuknya. Dalam implementasi, robot Hectarus akan berdiri dengan kondisi *link* tibia akan tegak lurus terhadap bidang lantainya yang diilustrasikan pada Gambar 3.10 (a). Hal tersebut bertujuan untuk mendapatkan ketinggian robot Hectarus yang optimal sehingga sensor ultrasonik pada robot tidak mendeteksi kakinya sendiri. Dengan kondisi tersebut, maka kondisi robot yang membuat servo membutuhkan torsi terbesarnya adalah ketika kaki robot ingin mencapai jarak jauhnya dengan ketinggian robot yang sama. Ilustrasi kondisi robot tersebut dapat dilihat pada Gambar 3.10 (b).



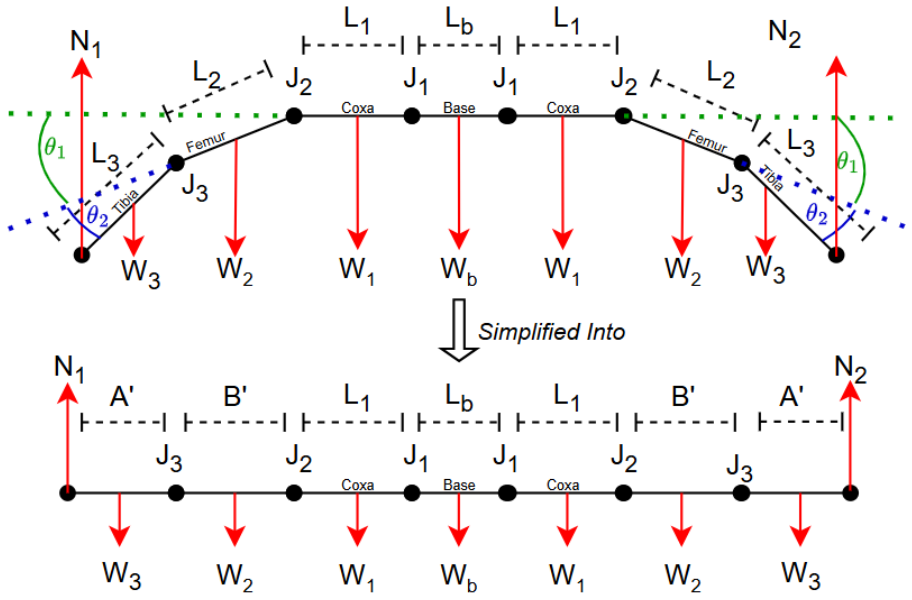
Gambar 3.10 (a) Ilustrasi Kondisi Berdiri (b) Ilustrasi Kondisi Kaki Hectarus Mencapai Jarak Terjauh dengan Ketinggian Sama

Berdasarkan ilustrasi pada Gambar 3.10 (b), maka nilai θ adalah sebagai berikut.

$$\theta = \arcsin(\frac{L_3}{L_2 + L_3})$$

$$\boldsymbol{\theta \approx 33,91^o}$$

Dengan besar nilai derajat θ tersebut, maka ilutrasi perhitungan untuk gaya normal maksimum yang bekerja robot Hectarus dapat dilihat pada Gambar 3.11 dengan nilai $\theta_1 = \theta$ dan $\theta_2 = 0$.



Gambar 3.11 Ilustrasi Perhitungan Gaya Normal Maksimum Kaki Hectarus

$$\sum \tau_{LeftFoot} = -W_3 * \left(\frac{A'}{2}\right) - W_2 * \left(\frac{B'}{2} + A'\right) - W_1 * \left(\frac{L_1}{2} + B' + A'\right) - W_b * \left(\frac{L_b}{2} + L_1 + B' + A'\right) - 2W_1 * \left(L_b + \frac{3L_1}{2} + B' + A'\right) - 2W_2 * \left(L_b + 2L_1 + \frac{3B'}{2} + A'\right) - 2W_3 * \left(L_b + 2L_1 + 2B' + \frac{3A'}{2}\right) + 2N_2 * (L_b + 2L_1 + 2B' + 2A')$$

$$A' = L_3 * \cos(\theta_1 + \theta_2)$$

$$B' = L_2 * \cos(\theta_1)$$

$$0 = -0{,}0337 - 0{,}0231 - 0{,}2259 - 6{,}3423 - 1{,}0669 - 0{,}1746 - 0{,}8333 + N_2 * 0{,}901$$

$$\boldsymbol{N_2 = 9{,}6553\ N}$$

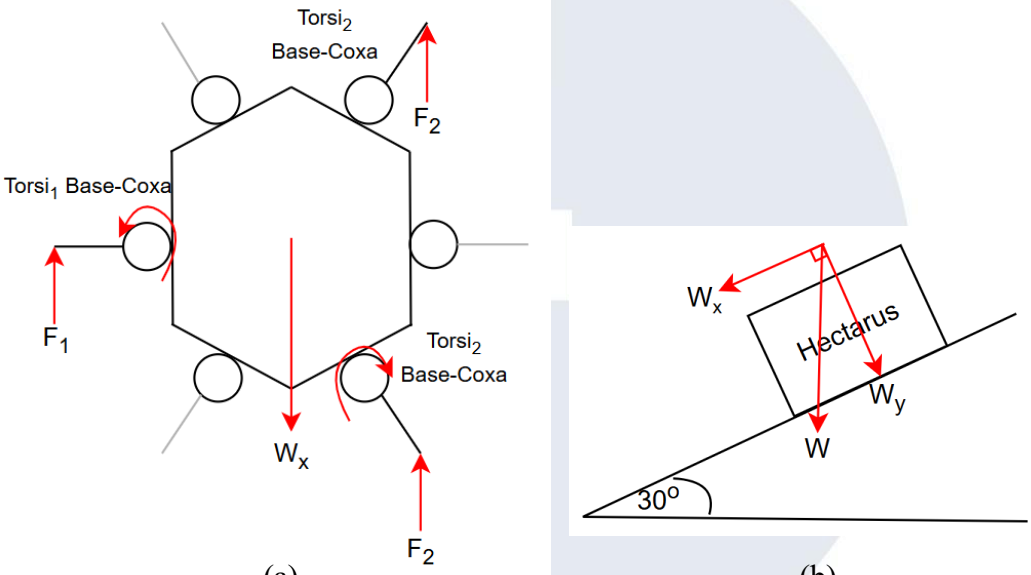
Dengan menggunakan persamaan (1), nilai N_1 adalah $\boldsymbol{26{,}426}\ N$. Nilai gaya normal tersebut digunakan untuk mendapatkan nilai torsi J_2 dan J_3.

$$\sum \tau_{J_2} = \tau_{servo} - N_1 * (B' + A') + \left(W_2 * \frac{B'}{2}\right) + \left\{W_3 * \left(B' + \frac{A'}{2}\right)\right\}$$

$$\sum \boldsymbol{\tau_{J_2\ left\ side}} = \boldsymbol{\tau_{servo} - 3{,}099\ Nm}$$

$$\sum \tau_{J_3} = \tau_{servo} - N_1 * A' + \left(W_3 * \frac{A'}{2}\right)$$

$$\boldsymbol{\sum \tau_{J_3\ left\ side} = \tau_{servo} - 1,748\ Nm}$$



(a) (b)

Gambar 3.12 (a) Ilustrasi Perhitungan *Joint Base*-Koksa Hectarus (b) Ilustrasi Hectarus di atas Tangga

Perhitungan torsi *joint Base*-Koksa (J_1) dihitung pada saat robot berada dalam bidang miring atau dalam lokasi pengujian tangga. Berdasarkan ilustrasi pada Gambar 3.12, untuk dapat bergerak maju atau menaiki tangga, robot Hectarus harus memiliki gaya dorong yang melawan arah "tarikan" dari gaya beratnya sendiri dan gaya yang dibutuhkan untuk menggerakkan atau mengangkat tubuhnya.

$$F_{total} = W_x + (m_{hectarus}\ x\ a_{hectarus})$$

$$W_x = massa_{total\ robot}\ x\ g\ x \sin 30^o = 4{,}667 * 9{,}8 * 0{,}5$$

$$\boldsymbol{W_x = 22,8683\ N}$$

Sehingga total gaya yang dibutuhkan oleh kaki robot Hectarus untuk memberikan gaya dorong adalah

$$F_{total} = 22{,}8683 + (4{,}667 * a)$$

Persamaan F_{total} diatas menandakan bahwa dalam pergerakan robot bergantung pada percepatan servo pada servo itu sendiri. Nilai F_{total} kemudian dibagi untuk per masing-masing kaki robot yang digunakan untuk melangkah. Dengan begitu, untuk mendapatkan pergerakan maju

yang lurus, percepatan yang dihasilkan dari kaki robot Hectarus harus memiliki nilai yang sama pada sisi kaki kiri dan kanan robot. Dengan pernyataan tersebut, maka nilai $a_1 = 2a_2$ atau dengan massa robot yang sama, $F_1 = 2F_2$.

$$\boldsymbol{F_1 + 2F_2 = 22,8683 + (4,667 * a)}$$

$$\boldsymbol{F_1 = 11,43415 + 2,3335a}$$

$$\boldsymbol{F_2 = 5,712 + 1,1668a}$$

Dengan nilai F_1 dan F_2 tersebut, didapatkan nilai torsi minimum pada J_1 yang dibutuhkan oleh robot Hectarus untuk menaiki tangga adalah sebagai berikut.

$$\tau_{J1} = \tau_{servo} - (F_1 * panjang\ kaki)$$

$$panjang\ kaki = L_2 \sin\theta_1 + L_3 \sin(\theta_1 + \theta_2) + L_1$$

$$panjang\ kaki = 0{,}1473$$

$$\boldsymbol{\tau_{J1} = \tau_{servo} - 1,684 - 0,344a\ Nm}$$

Dari persamaan tersebut didapatkan bahwa, untuk membuat robot dapat bergerak secara konstan (a = 0), nilai torsi minimum yang dibutuhkan oleh servo J_1 adalah 1,684 Nm.

Berdasarkan perhitungan diatas, nilai torsi maksimum yang dibutuhkan oleh robot Hectarus pada masing-masing *joint* adalah 1,684 Nm untuk J_1, 3,099Nm untuk J_2, dan 1,748 Nm untuk J_3. Perlu diketahui bahwa semua perhitungan tersebut dilakukan pada pola Tripod *gait* yang membuat sisi kaki robot yang hanya menggunakan 1 kaki untuk mengangkat badannya sendiri. Nilai torsi servo yang dibutuhkan pada *joint* kaki servo di sisi lainnya (yang menggunakan 2 kaki untuk menopang badan) akan lebih kecil karena distribusi beratnya pada 2 kaki tersebut. Dengan pernyataan tersebut, maka digunakan 2 tipe servo dalam pengembangan robot Hectarus yaitu servo TD8120MG yang memiliki nilai torsi 2,01036 Nm pada 5 Volt dan SN1030 yang memiliki nilai torsi 2,89296 Nm pada 5 Volt. Dalam implementasi robot Hectarus, servo yang akan digunakan adalah Servo TD8120MG akan digunakan pada semua *joint* kaki Hectarus kecuali pada *joint base*-koksa (J_1) dan

joint koksa-femur (J_2) kaki bagian tengah Hectarus yang menggunakan servo SN1030.

Dapat diperhatikan bahwa torsi servo yang dibutuhkan oleh robot pada *joint* J_2 adalah 3,099 Nm namun dalam implementasinya torsi servo yang dimiliki adalah 2,89296 Nm. Nilai tersebut tidak mencukupi kebutuhan servo tersebut. Nilai 3,099 Nm tersebut merupakan kondisi robot ketika kaki-kakinya pada robot Hectarus melebar dan mencapai jarak terjauhnya. Dalam implementasinya, robot Hectarus berdiri dan melangkah dengan kondisi *link* tibia tegak lurus terhadap permukaan lainnya sehingga nilai $\theta_1 = 0^o$ dan nilai $\theta_2 = 90^o$ pada penurunan persamaan diatas. Dengan nilai tersebut, maka nilai torsi yang dibutuhkan pada *joint* J_2 adalah 1,615 Nm. Dengan nilai tersebut, dengan Servo SN1030 cocok dan sudah mencukupi torsi servo yang dibutuhkan.

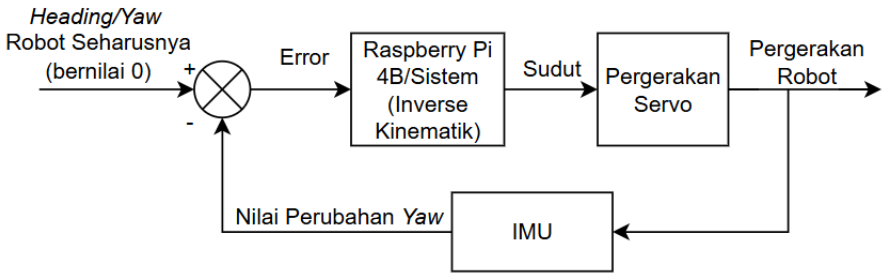
Berdasarkan penurunan persamaan diatas, dapat juga diketahui batas penambahan beban pada robot Hectarus. Jika batas penambahan bebannya hanya dihitung ketika robot berdiri (nilai $\theta_1 = 0^o$, nilai $\theta_2 = 90^o$ dan robot tidak berada dalam kondisi berjalan khususnya dengan Tripod *gait*), maka batas penambahan bebannya mencapai 8 kg sebelum servo pada *joint* J_2 robot khususnya pada bagian kaki depan dan belakang (Servo TD8120MG) mengalami kerusakan. Jika batas penambahan bebannya hingga mencakup dalam pergerakan robot khususnya dengan Tripod *gait*, maka batas penambahan bebannya adalah 4 kg sebelum servo yang digunakan pada J_2 kaki tengah (SN1030) mengalami kerusakan. Perlu diketahui bahwa semua perhitungan tersebut dilakukan dengan asumsi bahwa robot berada dalam kondisi idealnya dengan *Center of Mass* robot berada tepat ditengah robot dan kekuatan ketahanan *link* dari robot tidak diperhitungkan sehingga jika dilakukan pengujian ketahanan robot, terdapat kemungkinan bahwa sebelum mencapai 8 kg penambahan beban akan terjadi deformasi permanen pada *link* kaki robot atau kerusakan pada gir *servo arm* servo yang membuat servo tidak dapat menggerakkan *link* kakinya.

3.1.2 Desain Subsistem

Dari tinjauan desain sistem secara umum, terdapat 3 subsistem yang diimplementasikan pada robot Hectarus yaitu subsistem Pergerakan Lurus, subsistem Pemeriksaan Halangan dan subsistem Pergerakan Otomatis Robot.

– Subsistem Pergerakan Lurus

Subsistem Pergerakan Lurus pada robot Hectarus berfungsi untuk membuat robot dapat berjalan lurus tanpa adanya *error heading* yang membuatnya berjalan miring/tidak linear. Subsistem ini menggunakan sensor IMU yang akan digunakan sebagai *feedback* dalam sistem kendali yang akan digunakan.



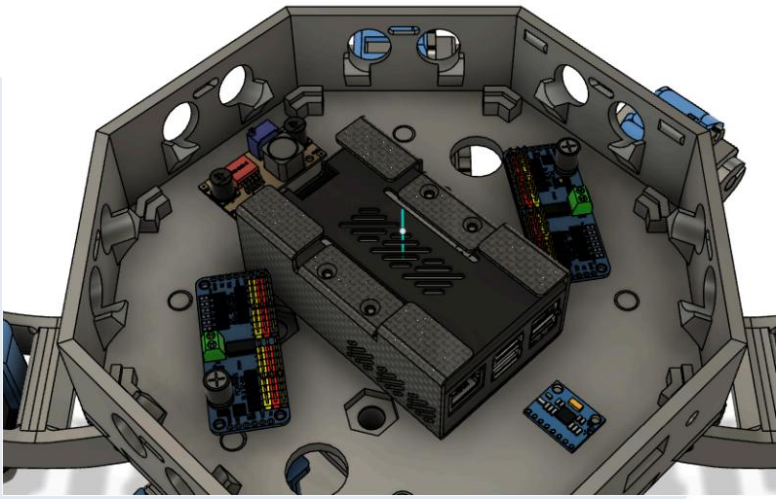
Gambar 3.13 Diagram Blok Sistem Kendali Pergerakan Lurus

Tabel 3.4 Penjelasan Input/*Output* Sistem Kendali Pergerakan Lurus

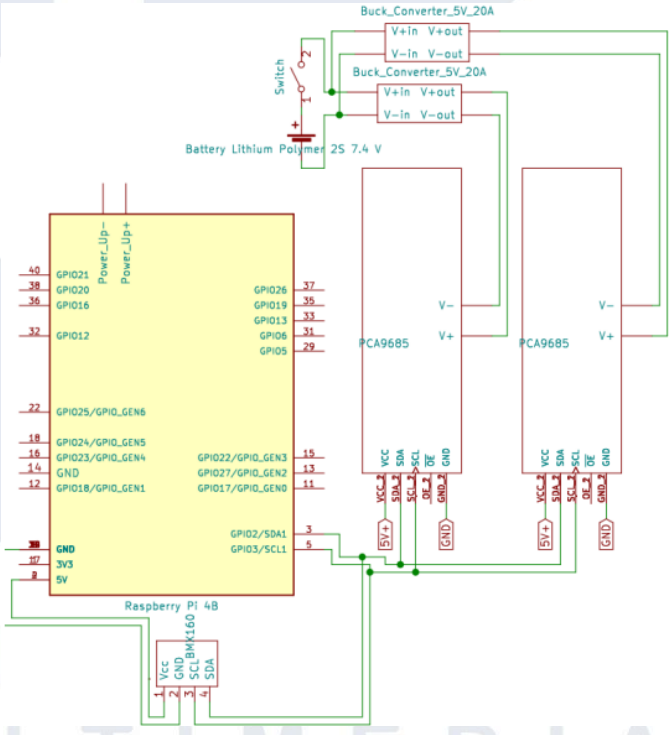
Parameter	Keterangan
Input	• Besar perubahan nilai *yaw* dari sensor IMU • Nilai *yaw* yang diharapkan atau yang seharusnya
Output	• Pergerakan servo yang memicu pergerakan robot Hectarus • Kondisi *heading/yaw* pada robot Hectarus
Fungsi	• Kompensasi *error* perubahan *heading/yaw* pada robot Hectarus dengan sensor IMU

Ketika robot melakukan suatu pergerakan maju, sensor IMU akan terus aktif dan menghitung perubahan nilai *yaw* pada robot. Nilai perubahan *yaw* ini kemudian dijadikan sebagai *error* input yang akan dikompensasi dengan perhitungan *inverse kinematic* robot. Perhitungan *inverse kinematic* ini kemudian menghasilkan output yang membuat robot melangkah sesuai dengan perubahan nilai *yaw* yang bertambah pada pergerakan sebelumnya.

Wiring subsistem Pergerakan Lurus dapat dilihat pada Gambar 3.15. Sensor IMU akan langsung dihubungkan dengan pin mikrokontroler Raspberry Pi 4B yang akan diletakkan pada bagian atas robot Hectarus pada Gambar 3.14.



Gambar 3.14 Peletakan Komponen atau *Hardware* Subsistem Pergerakan Lurus



Gambar 3.15 *Wiring Diagram* Subsistem Pergerakan Lurus

– Subsistem Pemeriksaan Halangan

Subsistem Pemeriksaan Halangan pada robot Hectarus digunakan untuk mengetahui jarak antara robot dengan halangan seperti tembok

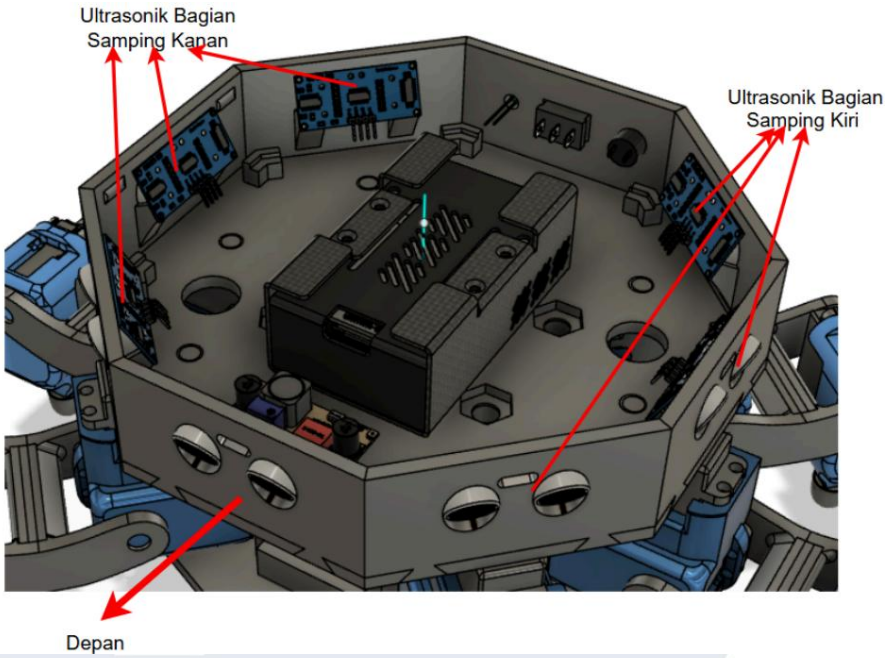
di sekitarnya. Data jarak tersebut kemudian akan digunakan sebagai input sistem robot untuk melakukan kalkulasi titik tengah area untuk melakukan pergerakan maju dan juga untuk mengetahui arah perbelokan yang harus dilakukan oleh robot ketika terdapat halangan tepat di sekeliling robot yang mengharuskan untuk menghadap kiri/kanan. Subsistem ini menggunakan 7 sensor ultrasonik HC-SR04. Penggunaan sensor ultrasonik berjumlah 7 berguna untuk mendapatkan informasi halangan yang lebih akurat sehingga robot Hectarus dapat mengetahui posisi halangan dan posisi robot yang seharusnya untuk melakukan perputaran arah.

Gambar 3.16 Diagram Blok Subsistem Pemeriksaan Halangan

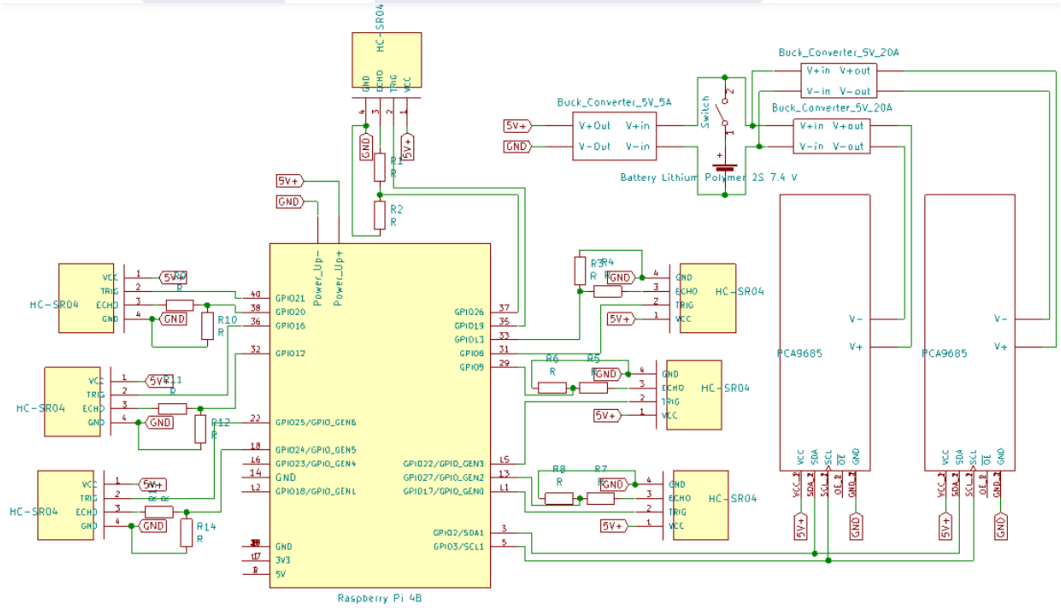
Tabel 3.5 Penjelasan Input/*Output* Subsistem Pemeriksaan Halangan

Parameter	Keterangan
Input	• Kondisi halangan disekitar robot Hectarus yang berubah karena pergerakan robot • Besar jarak halangan di sekitar robot Hectarus
Output	• Penentuan keputusan pergerakan robot Hectarus yang membuat servo dan robot bergerak
Fungsi	• Penentuan keputusan secara otomatis oleh robot Hectarus yang membuatnya dapat menghindari halangan-halangan di sekitarnya

Subsistem Pemeriksaan Halangan robot Hectarus akan terpicu setiap robot akan ingin melakukan pergerakan. Robot akan memeriksa halangan disekitarnya terlebih dahulu untuk melakukan penentuan keputusan. *Wiring* subsistem Pemeriksaan Halangan dapat dilihat pada Gambar 3.18. Tujuh sensor ultrasonik HC-SR04 akan disuplai langsung oleh baterai yang sudah diturunkan tegangannya. Hal ini bertujuan untuk menghindari adanya limit arus oleh ultrasonik HC-SR04 jika disuplai langsung oleh Raspberry Pi 4B.



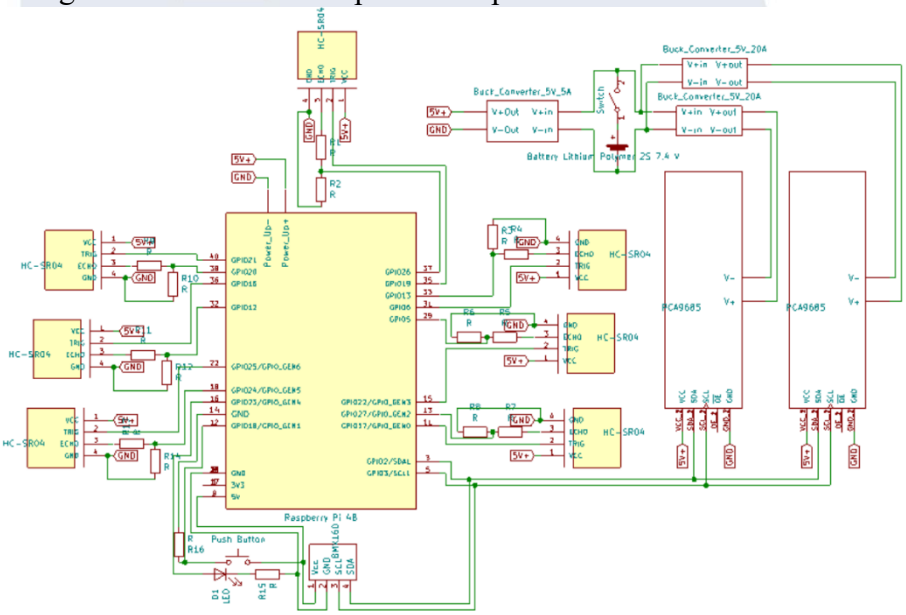
Gambar 3.17 Peletakan Komponen/*Hardware* Subsistem Pemeriksaan Halangan



Gambar 3.18 *Wiring Diagram* Subsistem Pemeriksaan Halangan

– Subsistem Pergerakan Otomatis

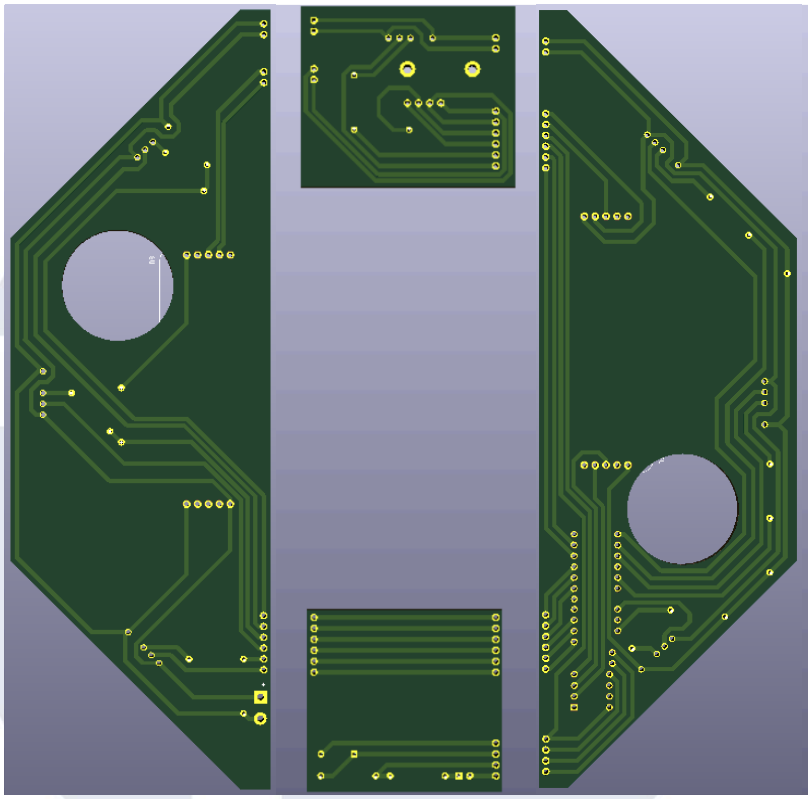
Subsistem Pergerakan Otomatis pada dasarnya merupakan gabungan dari kedua subsistem sebelumnya. Robot Hectarus melakukan pergerakan setelah sistem selesai mengolah data dan menentukan keputusan melangkah yang harus dilakukan dengan sensor/subsistem Pergerakan Lurus dan Pemeriksaan Halangan. Blok diagram dari subsistem dapat dilihat pada Gambar 2.3.



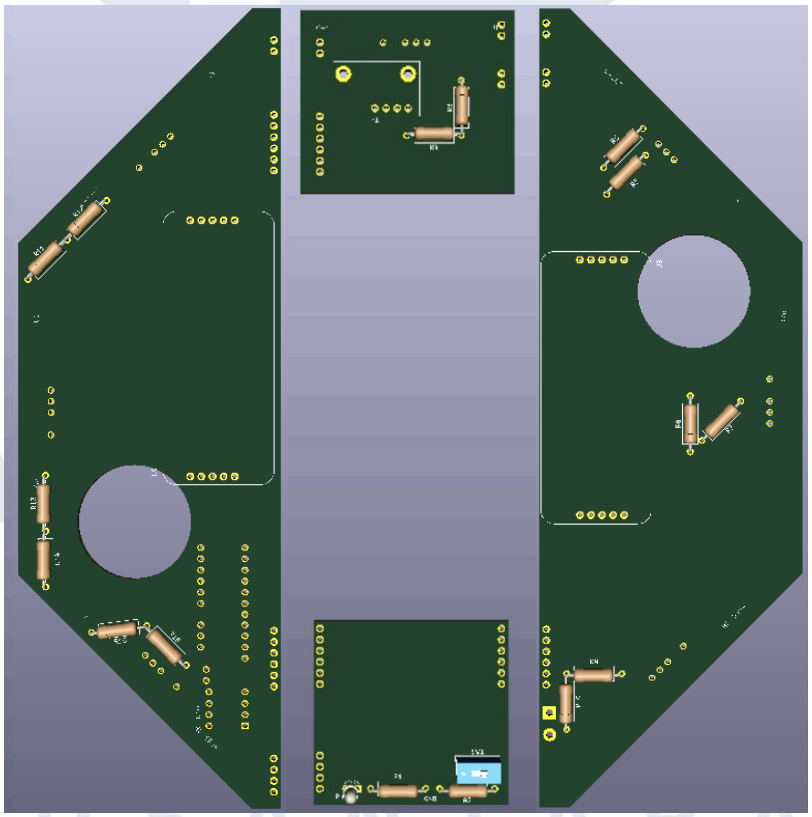
Gambar 3.19 *Wiring Diagram* Robot Hectarus

3.1.3 Diagram Sistem

Baterai Lithium Polymer 7,4 Volt akan menyuplai 3 *step down Buck converter* pada robot Hectarus. Terdapat 2 jenis *Buck converter* yang digunakan yaitu 2 modul dengan spesifikasi 20A dan 1 modul dengan spesifikasi 5A. *Step down Buck converter* 20A akan digunakan untuk menyuplai servo *driver* PCA9685 untuk sumber daya servo dengan masing-masing servo *driver* terhubung dengan 3 kaki robot atau 9 servo. *Step down Buck Converter* 5A akan digunakan untuk menyuplai Raspberry Pi 4B, 7 sensor ultrasonik HC-SR04 dan PCA9685 (untuk mengaktifkan modulnya). Sensor IMU BMX160, LED dan *push button* akan bersumber daya langsung dari pin 5V Raspberry Pi 4B.



Gambar 3.20 Tampak Bawah Desain PCB Implementasi *Wiring Diagram* Robot Hectarus



Gambar 3.21 Tampak Atas Desain PCB Implementasi *Wiring Diagram* Robot Hectarus

PCB dipecah menjadi 4 bagian bertujuan untuk memperkecil biaya percetakan yang akan menggunakan mesin CNC. Sambungan antar bagian PCB akan menggunakan pin *U-header* yang akan disolder pada pin yang sudah dibuat.

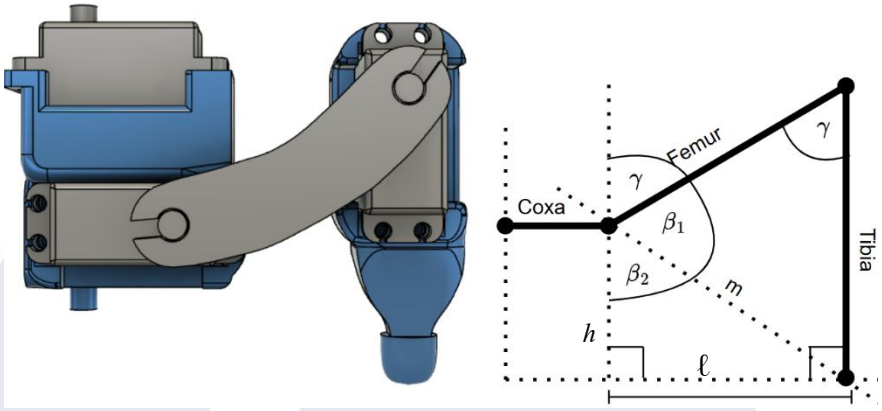
3.2 Implementasi Sistem

Implementasi sistem dimulai dari subsistem pengecekan kemiringan dan pergerakan lurus dengan sensor IMU karena melibatkan *Inverse Kinematic* (IK) yang akan digunakan pada algoritma Tripod *gait*, *Wave gait* dan *Tetrapod gait*. Algoritma *gait* ini dibagi menjadi 2 yaitu algoritma jalan dan menaiki tangga. Implementasi selanjutnya yaitu implementasi pengecekan halangan dengan ultrasonik, dan implementasi pergerakan otomatis.

3.2.1 Hasil Implementasi

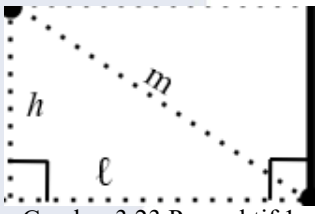
- Pengecekan Kemiringan dan Pergerakan Lurus

  Subsistem pengecekan kemiringan dan pergerakan lurus dimulai dari mencari persamaan IK yang akan digunakan pada masing-masing kaki robot Hectarus. Perhitungan persamaan IK dimulai dari kondisi Hectarus berdiri untuk mendapatkan hubungan besar lebar kaki robot dengan ketinggian robot. Hasil hubungan tersebut kemudian digunakan untuk menemukan besaran derajat masing-masing *joint*/servo Hectarus untuk menggerakkan kaki. Terdapat 3 persamaan yang dihasilkan untuk 6 kaki robot Hectarus yaitu kaki bagian depan, tengah dan belakang. Sudut/derajat *joint* yang menghubungkan antara *base* dan koksa disimbolkan dengan α, sudut/derajat *joint* yang menghubungkan antara koksa dan femur disimbolkan dengan β, sudut/derajat *joint* yang menghubungkan antara femur dan tibia disimbolkan dengan γ.



Gambar 3.22 Diagram Kinematik Hectarus dengan Bantuan Proyeksi Garis

Untuk mendapatkan hubungan antara nilai ℓ dengan nilai h, diperlukan persamaan ℓ yang akan dieliminasi sehingga menjadi sebuah persamaan h. Dengan rumus Phytagoras, nilai ℓ dengan perspektif pada Gambar 3.23.



Gambar 3.23 Perspektif 1 Segitiga Siku-Siku

$$\ell = \sqrt{m^2 - h^2} \tag{2}$$

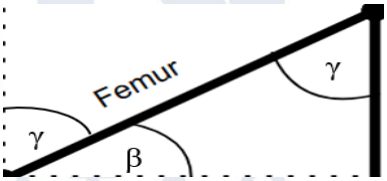
Berdasrakan Gambar 3.22, variabel m didapatkan dengan rumus cosinus.

$$m = \sqrt{femur^2 + tibia^2 - 2 * femur * tibia * \cos\gamma}$$

Sehingga persamaan (2) menjadi

$$\ell = \sqrt{femur^2 + tibia^2 - 2 * femur * tibia * \cos\gamma - h^2} \tag{3}$$

Persamaan ℓ juga dapat dihitung dengan rumus sinus dengan perspektif pada Gambar 3.24.



Gambar 3.24 Perspektif 2 Segitiga

$$\boldsymbol{\ell = \sin\gamma * femur} \tag{4}$$

Dengan mengeliminasi variabel ℓ pada persamaan (3) dan (4), didapatkan persamaan h adalah sebagai berikut

$$\sin\gamma * femur = \sqrt{femur^2 + tibia^2 - 2 * femur * tibia * \cos\gamma - h^2}$$

$$\sin^2\gamma * femur^2 = femur^2 + tibia^2 - 2 * femur * tibia * \cos\gamma - h^2$$

$$h = \sqrt{femur^2 + tibia^2 - 2 * femur * tibia * \cos\gamma - \sin^2\gamma * femur^2} \quad (5)$$

Untuk mendapatkan nilai variabel ℓ, nilai derajat γ perlu diketahui sehingga, rumus (5) diubah menjadi fungsi γ = f(h).

$$h = \sqrt{femur^2 + tibia^2 - 2 * femur * tibia * \cos\gamma - \sin^2\gamma * femur^2}$$

$$h^2 = femur^2 + tibia^2 - 2 * femur * tibia * \cos\gamma - \sin^2\gamma * femur^2$$

$$h^2 - femur^2 - tibia^2 + \sin^2\gamma * femur^2 = -2 * femur * tibia * \cos\gamma$$

$$h^2 - femur^2 - tibia^2 + (1 - \cos^2\gamma) * femur^2 = -2 * femur * tibia * \cos\gamma$$

$$h^2 - femur^2 - tibia^2 + femur^2 - \cos^2\gamma\, femur^2 = -2 * femur * tibia * \cos\gamma$$

$$h^2 - tibia^2 = -2 * femur * tibia * \cos\gamma + \cos^2\gamma\, femur^2$$

Asumsikan $\cos\gamma = X$

$$h^2 - tibia^2 = -2 * femur * tibia * X + X^2 femur^2$$

$$X^2 femur^2 - 2 * femur * tibia * X + (h^2 - tibia^2) = 0$$

Dengan rumus persamaan kuadrat nilai X adalah sebagai berikut

$$X = \frac{femur * tibia \pm \sqrt{femur^2 * tibia^2 + femur^2(h^2 - tibia^2)}}{femur^2}$$

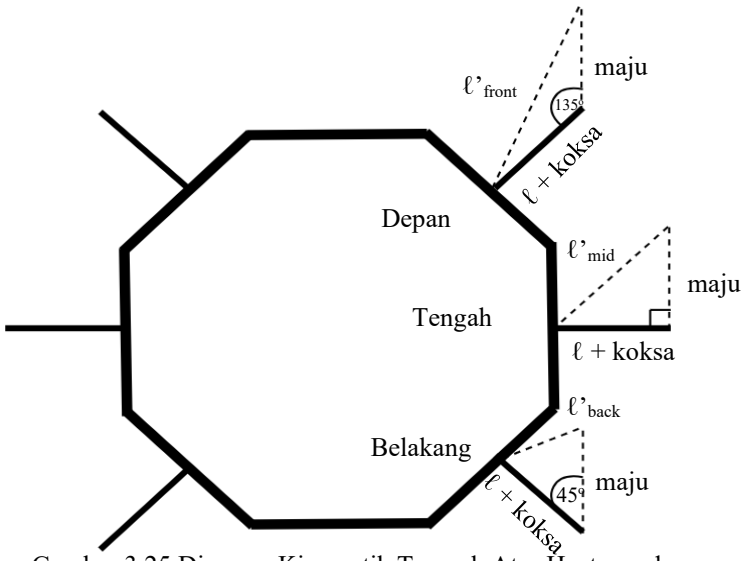
$$\cos\gamma = \frac{femur * tibia \pm \sqrt{femur^2 * tibia^2 + femur^2(h^2 - tibia^2)}}{femur^2}$$

$$\boldsymbol{\gamma = \arccos\left(\frac{femur * tibia \pm \sqrt{femur^2 * tibia^2 + femur^2(h^2 - tibia^2)}}{femur^2}\right)} \quad (6)$$

Dengan perhitungan IK di atas, didapatkan hubungan antara lebar kaki robot Hectarus dengan ketinggian robot sebelum robot melangkahkan kakinya. Hubungan tersebut akan digunakan untuk menghitung besar sudut/derajat α, β dan γ ketika robot dalam kondisi berjalan.

Pada saat implementasi, kaki *link* tibia robot Hectarus akan tegak lurus terhadap bidang permukaannya sehingga nilai $h = tibia$. Hal tersebut bertujuan untuk menghindari ultrasonik pada robot Hectarus mendeteksi kakinya sendiri.

Setelah nilai variabel ℓ didapatkan dengan persamaan (4) dan (6), dilanjutkan dengan menghitung IK robot untuk pergerakan maju dan koreksi *heading* untuk ketiga kaki (depan, tengah, belakang).



Gambar 3.25 Diagram Kinematik Tampak Atas Hectarus dengan Bantuan Garis Proyeksi

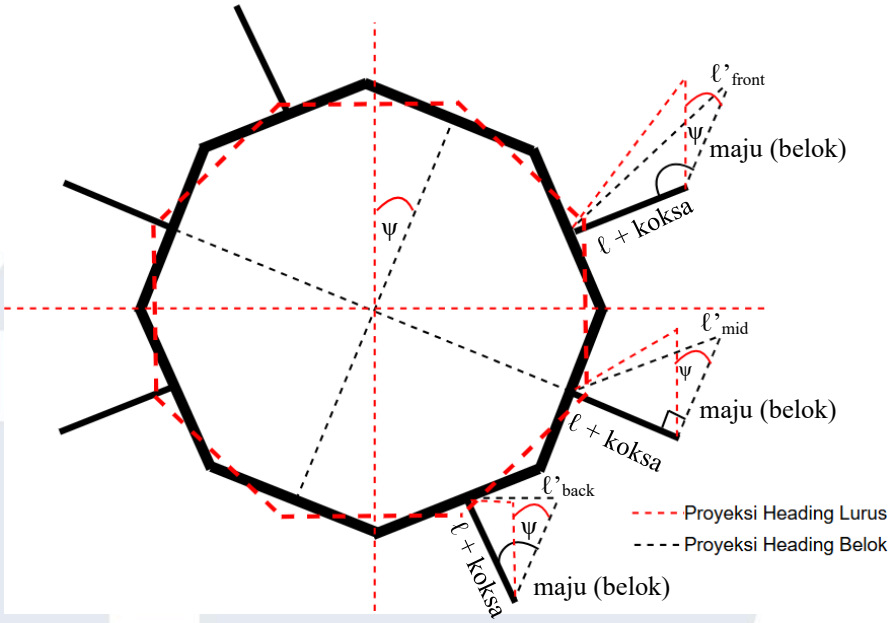
Perbedaan ketiga persamaan IK adalah pada parameter derajat kakinya dan nilai ℓ' untuk jarak maju yang sama. Dengan menggunakan rumus segitiga cosinus, perhitungan IK-nya untuk *joint* base-koksa (α) adalah sebagai berikut.

$$\ell'_{front} = \sqrt{maju^2 + (\ell + coxa)^2 - 2 * maju * (\ell + coxa) * \cos 135^o}$$

$$\ell'_{mid} = \sqrt{maju^2 + (\ell + coxa)^2 - 2 * maju * (\ell + coxa) * \cos 90^o}$$

$$\ell'_{back} = \sqrt{maju^2 + (\ell + coxa)^2 - 2 * maju * (\ell + coxa) * \cos 45^o}$$

Nilai ℓ' dapat berubah bergantung pada kondisi *heading* robot Hectarus untuk subsistem pergerakan lurus. Untuk ilustrasinya dapat dilihat pada Gambar 3.26.



Gambar 3.26 Diagram Kinematik Robot Hectarus dalam Keadaan Tidak Lurus

Ketika robot sedang berjalan, robot tidak akan selalu berada dalam kondisi lurus atau dengan kata lain, *heading* robot Hectarus tidak selalu dalam keadaan menghadap ke depan. Pergerakan robot Hectarus memungkinkan *heading* robot untuk berbelok ke kiri atau kekanan baik karena kecepatan servo yang tidak seimbang atau *end-effector* kaki yang tidak tepat menapak pada permukaan tanah. Hal tersebut dikompensasi dengan nilai *error* yang didapatkan dari sensor IMU dalam bentuk nilai *yaw* (ψ). Berdasarkan rumus ℓ' sebelumnya, nilai derajatnya akan berkurang sebesar nilai derajat *yaw* (ψ) pada sensor IMU. Nilai ±*yaw* pada persamaan bergantung pada sisi kaki yang digunakan.

$$\ell'_{front} = \sqrt{maju^2 + (\ell + coxa)^2 - 2 * maju * (\ell + coxa) * \cos(135^o \pm yaw)}$$

$$\ell'_{mid} = \sqrt{maju^2 + (\ell + coxa)^2 - 2 * maju * (\ell + coxa) * \cos(90^o \pm yaw)}$$

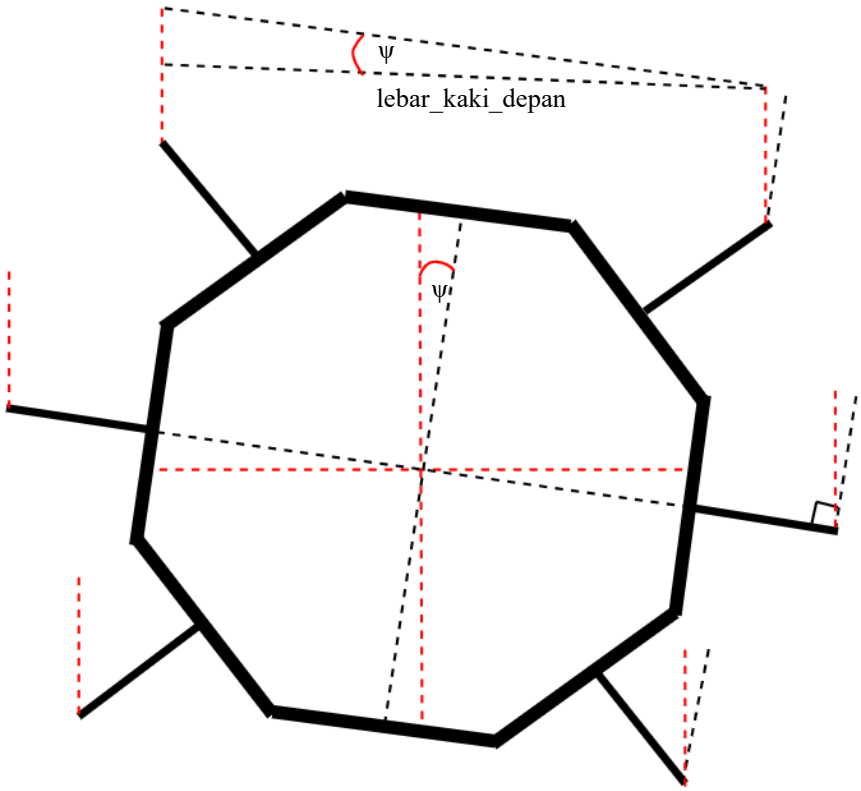
$$\ell'_{back} = \sqrt{maju^2 + (\ell + coxa)^2 - 2 * maju * (\ell + coxa) * \cos(45^o \pm yaw)}$$

$$\alpha_{front} = \cos^{-1}\left(\frac{(\ell + coxa)^2 + \ell'^{\,2}_{front} - maju^2}{2 * (\ell + coxa) * \ell'_{front}}\right)$$

$$\alpha_{mid} = \cos^{-1}\left(\frac{(\ell + coxa)^2 + \ell'^{\,2}_{mid} - maju^2}{2 * (\ell + coxa) * \ell'_{mid}}\right)$$

$$\alpha_{back} = cos^{-1}\left(\frac{(\ell + coxa)^2 + \ell'^{\,2}_{back} - maju^2}{2 * (\ell + coxa) * \ell'_{back}}\right)$$

Kendali/kontrol untuk subsistem pergerakan lurus tidak cukup dengan rumus di atas. Pengambilan jarak langkah/maju yang dilakukan oleh robot Hectarus akan dibuat berubah-ubah bergantung pada arah *heading error* robot Hectarus. Jika *error heading* robot Hectarus adalah ke kanan (searah jarum jam, nilai ψ = negatif), maka jarak pengambilan kaki bagian kiri akan lebih sedikit dari pada jarak pengambilan kaki bagian kanan untuk membuat robot Hectarus kembali menghadap ke kiri. Jarak pengurangan pengambilan langkah kaki ini dihitung dengan rumus $\tan\psi * lebar\_kedua\_kaki$. Variabel lebar_kedua_kaki merupakan variabel lebar kedua kaki pada bagian depan/belakang dan tengah. Untuk ilustrasi lebih lanjut dapat dilihat pada Gambar 3.27.



Gambar 3.27 Diagram Kinematik Robot Hectarus Pengurangan Jarak Pengambilan Langkah Kaki

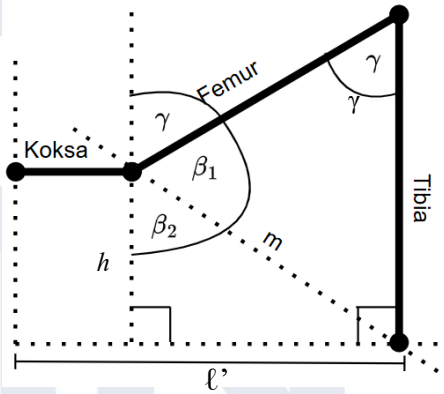
Pengurangan jarak pengambilan langkah kaki bagian depan atau belakang dengan tengah akan memiliki nilai yang berbeda karena dengan nilai *yaw* yang sama, lebar kedua kaki bagian tengah dan depan/belakang berbeda. Setelah dilakukan pengukuran, lebar kaki

bagian tengah adalah 33,5 cm dan lebar kaki bagian depan dan belakang adalah 24,5 cm.

Nilai *yaw* yang digunakan untuk pengurangan pengambilan langkah akan dibatasi hingga sampai membuat kaki robot tidak bergerak maju. Hal ini bertujuan untuk memudahkan logika IK pada robot Hectarus karena dengan nilai parameter maju yang negatif membuat nilai derajatnya menjadi ambigu. Jarak pengambilan langkah maju robot Hectarus dibuat 4 cm. Hal ini dibuat berdasarkan hasil observasi dengan nilai jarak maju lebih dari 4 cm, akan membuat jarak pengambilan kaki bagian tengah robot Hectarus akan menabrak kaki bagian depan.

Berdasarkan hal tersebut, maka nilai *yaw* akan dibatasi dengan nilai sebesar $\arctan\left(\frac{4}{33,5}\right) \approx \pm 6,8^{o}$. Nilai positif menandakan bahwa robot Hectarus miring ke arah kiri dan nilai negatif menandakan bahwa robot Hectarus miring ke arah kanan.

Untuk persamaan *joint* koksa-femur ($\beta_1 + \beta_2$) dan *joint* femur-tibia dihitung dengan perspektif segitiga pada Gambar 3.28 dengan nilai ℓ' bergantung pada jenis kaki yang digunakan. Dengan menggunakan rumus segitiga cosinus. Perhitungan IK-nya adalah sebagai berikut.



Gambar 3.28 Perspektif Segitiga untuk Perhitungan β

$$m = \sqrt{h^2 + (\ell' - \text{koksa})^2}$$

$$\boldsymbol{\beta_1 = \cos^{-1}\left(\frac{femur^2 + m^2 - tibia^2}{2 * femur * m}\right)}$$

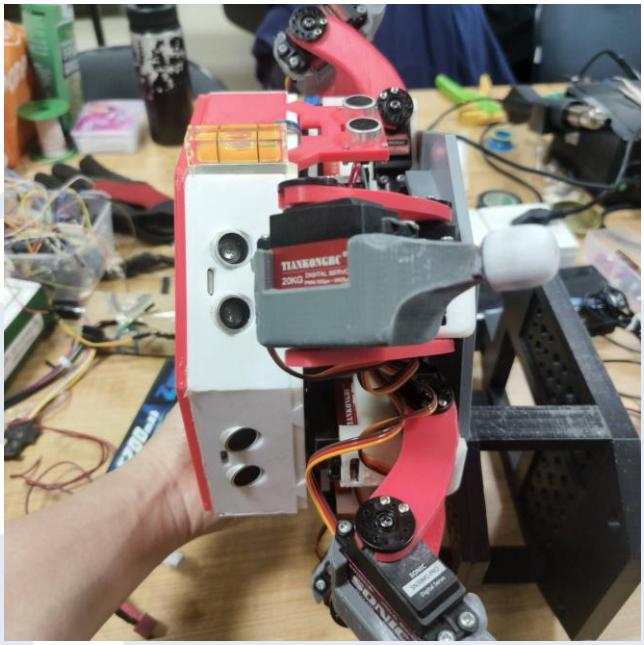
$$\boldsymbol{\beta_2 = \cos^{-1}\left(\frac{m^2 + h^2 - (\ell' - koksa)^2}{2 * m * h}\right)}$$

$$\boldsymbol{\gamma = cos^{-1}\left(\frac{femur^2 + tibia^2 - m^2}{2 * femur * tibia}\right)}$$

Setelah didapatkan persamaan untuk IK dan implementasinya dengan algoritma Pergerakan Lurus, dilakukan pengujian untuk melihat fungsionalitas sensor IMU BMX160. Dalam pengujian fungsionalitas ini, sensor IMU akan menggunakan metode *Complementary Filter*, salah satu metode yang paling sederhana dan cepat untuk menggabungkan 2 sensor (*sensor fusion*) yaitu sensor *accelerometer* dan *gyroscope*.

Complementary Filter merupakan jenis filter yang sederhana dan tidak mengonsumsi waktu komputasi yang tinggi, yang menggabungkan *high-pass* filter dan *low-pass* filter [61]. Dalam integrasi-nya, data sensor *gyroscope* dilakukan *high-pass* filter dan data sensor *accelerometer* akan dilakukan *low-pass* filter. Hal tersebut karena *gyroscope* lebih akurat dalam jangka waktu pendek. Data *accelerometer* lebih akurat dalam jangka waktu panjang karena bergantung pada gravitasi bumi. Oleh karena itu, dalam implementasinya data sensor *gyroscope* akan dikalikan dengan nilai konstanta yang lebih besar (nilai yang digunakan adalah 0,98) dari pada data sensor *accelerometer* (0,02).

Setelah dilakukan implementasi, dilakukan pengujian untuk melihat keakuratan dan besar nilai *drift* yang terjadi yang diukur selama 2 menit. Dokumentasi dan hasil data pengujian tersebut dapat dilihat pada Gambar 3.29, Gambar 3.30, Tabel 3.6 dan Tabel 3.7.



Gambar 3.29 Cuplikan Pengujian Akurasi Sensor IMU

```
Yaw: -0.7516988383111517
Roll: -0.451282389818366
Pitch: 0.1620031731466736

Yaw: -0.75246724149735 47
Roll: -0.45264026958639136
Pitch: 0.1664538419636229

Yaw: -0.7532874315166875
Roll: -0.4528237425276726
Pitch: 0.167480416754124 84

Yaw: -0.7532136566789953
Roll: -0.4488962466588504
Pitch: 0.1624362973297362

Yaw: -0.7540581886498511
Roll: -0.44723220444981 54
Pitch: 0.16131108629888138
```

Gambar 3.30 Dokumentasi Hasil Pengujian Akurasi Sensor IMU

Tabel 3.6 Data Hasil Akurasi Sensor IMU terhadap Perubahan Sudut dengan *Complementary Filter*

Nilai Sebenarnya	**Nilai Pengukuran (º)**			**Akurasi (%)**		
	Yaw	***Pitch***	***Roll***	***Yaw***	***Pitch***	***Roll***
90º	89,11	88,8	89,3	99,0	98,7	99,2
90º	89,7	89	91,2	99,7	98,9	98,7
90º	89,4	88,5	91,3	99,3	98,3	98,6
90º	89,85	88,9	88,3	99,8	98,8	98,1
90º	89,75	88,5	89,3	99,7	98,3	99,2
90º	89,4	89	89	99,3	98,9	98,9
90º	89,5	89,2	89,6	99,4	99,1	99,6
90º	89,7	89,3	89,4	99,7	99,2	99,3
90º	89,6	89,8	89,7	99,6	99,8	99,7
90º	89,8	88,3	89	99,8	98,1	98,9
Rata - rata				**99,5**	**98,8**	**99,0**

Tabel 3.7 Data Hasil Akurasi Sensor IMU terhadap Nilai *Drift* selama 2 Menit dengan *Complementary Filter*

Nilai Sebenarnya	**Nilai Pengukuran (º)**			***Error Drift* (%)**		
	Yaw	***Pitch***	***Roll***	***Yaw***	***Pitch***	***Roll***
0º	-0,75	0,16	-0,45	75,40	16,10	44,70
0º	0,45	-1,49	-0,29	44,80	149,00	28,70
0º	0,68	0,81	1,32	67,50	80,70	131,90
0º	0,89	0,94	0,72	88,50	94,10	72,10
0º	-0,17	2,55	-0,55	17,30	255,20	54,60
0º	-0,73	1,40	-0,57	73,40	140,30	56,80
0º	0,27	-0,74	-0,87	27,10	74,00	86,70
0º	0,36	0,76	1,45	35,90	75,80	144,50
0º	-0,03	0,88	-0,22	2,50	88,10	21,50
0º	0,66	1,44	1,01	66,10	143,50	100,60
Rata - rata				**49,85**	**111,68**	**74,21**

Dari data hasil pengujian pada tabel 3.6 dan tabel 3.7, meskipun nilai derajat output yang dihasilkan oleh sensor cukup akurat (Tabel 3.6), terdapat *drift* yang dihasilkan pada sensor IMU tersebut

khususnya pada parameter *pitch* hingga nilai rata – rata *error* mencapai 111,68% (Tabel 3.7). *Drift* pada sensor IMU akan menjadi hal yang cukup krusial dalam pengoperasian robot Hectarus karena tentu akan mempengaruhi pergerakan dan proses pencapaian titik *finish* robot. Adanya *drift* tersebut dapat disebabkan karena data sensor *gyroscope* yang dilakukan integrasi dan diakumulasikan seiring waktu. Selain itu dapat disebabkan karena nilai konstanta filter yang perlu dilakukan *tuning* lebih lanjut yang tentu memerlukan waktu. Untuk menangani hal tersebut, digunakan metode *sensor fusion* lainnya yaitu *Madgwick Filter*.

Madgwick Filter merupakan jenis atau metode filter yang menggunakan representasi *quaternion* dan memanfaatkan algoritma *gradient-descent* [62]. Representasi *quaternion* tersebut memiliki keunggulan untuk mencegah adanya kesalahan perhitungan derajat karena adanya 2 axis mencegah adanya kesalahan perhitungan derajat karena adanya 2 sumbu yang saling bertumpu. *Madgwick Filter* juga memiliki keunggulan lebih dibandingkan dengan jenis filter lainnya seperti Kalman Filter yang lebih umum digunakan karena proses komputasinya yang lebih ringan dan dapat bekerja dengan frekuensi *sampling* rendah [62].

Madgwick Filter mendukung untuk penggabungan 2 sensor seperti *gyroscope* dan *accelerometer* saja atau penggabungan 3 sensor seperti *gyroscope, accelerometer,* dan *magnetometer*. Tujuan utama penggunaan *Madgwick Filter* adalah kemampuannya untuk dapat menggabungkan 3 sensor (*gyroscope*, *magnetometer*, dan *accelerometer*). Namun dalam implementasinya, robot Hectarus hanya menggunakan data sensor *gyroscope* dan *accelerometer* untuk proses implementasi yang sederhana tanpa adanya kalibrasi *soft-iron* dan *hard-iron* pada *magnetometer*. Hasil implementasi *Madgwick Filter* pada robot Hectarus dapat dilihat pada tabel 3.8 dan tabel 3.9.

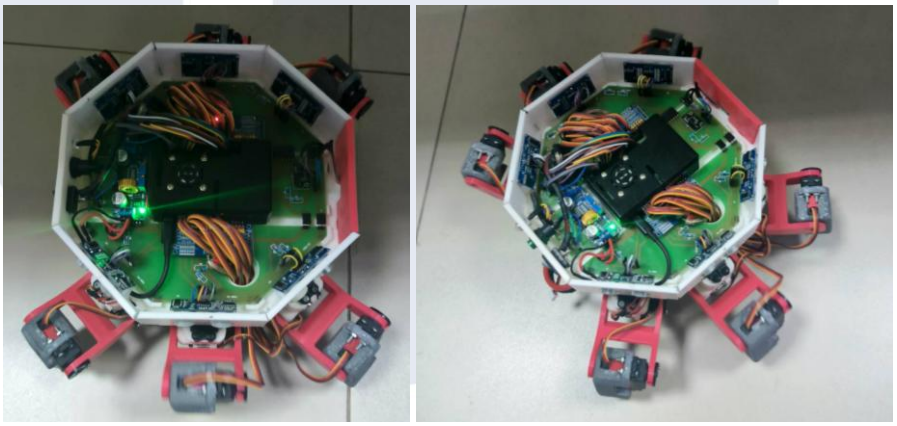
Tabel 3.8 Data Hasil Akurasi Sensor IMU terhadap Perubahan Sudut dengan *Madgwick Filter*

Nilai Sebenarnya	Nilai Pengukuran (º)			Akurasi (%)		
	Yaw	*Pitch*	*Roll*	*Yaw*	*Pitch*	*Roll*
90º	89,12	86,3	86,5	99,0	95,9	96,1
90º	89,16	93,7	86,2	99,1	95,9	95,8
90º	89,51	87,2	85,9	99,5	96,9	95,4
90º	89,29	87,3	86,7	99,2	97,0	96,3
90º	89,36	86,9	86,7	99,3	96,6	96,3
90º	89,61	94,8	84,9	99,6	94,7	94,3
90º	89,12	86,5	86,9	99,0	96,1	96,6
90º	89,57	92,9	86,5	99,5	96,8	96,1
90º	89,76	86,6	87,2	99,7	96,2	96,9
90º	89,95	91,2	86,4	99,9	98,7	96,0
Rata - rata				**99,4**	**96,5**	**96,0**

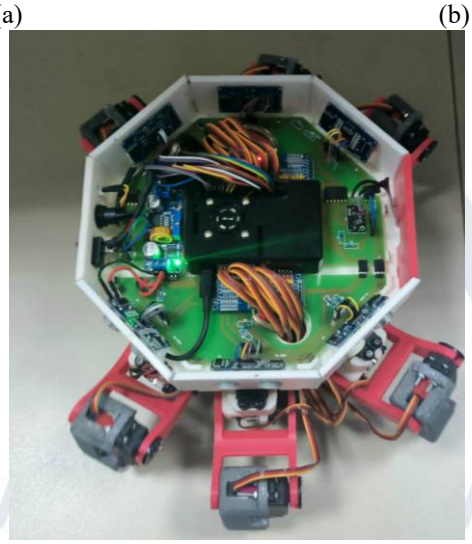
Tabel 3.9 Data Hasil Akurasi Sensor IMU terhadap Nilai *Drift* selama 2 Menit dengan *Madgwick Filter*

Nilai Sebenarnya	Nilai Pengukuran (º)			*Error Drift* (%)		
	Yaw	*Pitch*	*Roll*	*Yaw*	*Pitch*	*Roll*
0º	0,19	0,02	0,01	19,3	0,7	1,8
0º	0,29	0,05	0,05	29,2	5,4	5,2
0º	0,56	-0,09	0,04	56,1	4,2	9,2
0º	0,34	-0,11	-0,06	34	6,2	11,4
0º	-0,07	0,02	0,01	7,1	0,9	1,64
0º	0,11	-0,07	-0,02	11,4	1,9	7,43
0º	0,22	-0,08	0,02	22	1,6	7,6
0º	0,26	-0,05	-0,05	25,5	5,2	5
0º	-0,14	-0,11	-0,05	13,5	4,5	11,1
0º	0,40	-0,07	0,03	39,8	3,1	6,7
Rata-rata				**25,79**	**3,37**	**6,707**

Berdasarkan tabel 3.8 dan 3.9, data sensor IMU yang dihasilkan memiliki nilai yang cukup akurat namun dengan nilai yang lebih rendah dari *Complementary Filter* dengan nilai akurasi terendahnya adalah 96%. Namun dengan nilai akurasi yang lebih rendah, *drift* yang dimiliki dengan *Madgwick Filter* dapat diminimalisasi secara signifikan dengan nilai persentase *error* menurun dari 49,85% untuk *yaw*, 111,68% untuk *pitch*, 74,21% untuk *roll*, menjadi 25,79% untuk *yaw*, 3,37% untuk *pitch* dan 6,71% untuk *roll*. Dengan peningkatan yang signifikan tersebut, maka digunakan algoritma *Madgwick Filter* untuk subsistem Pergerakan Lurus.



(a) (b)



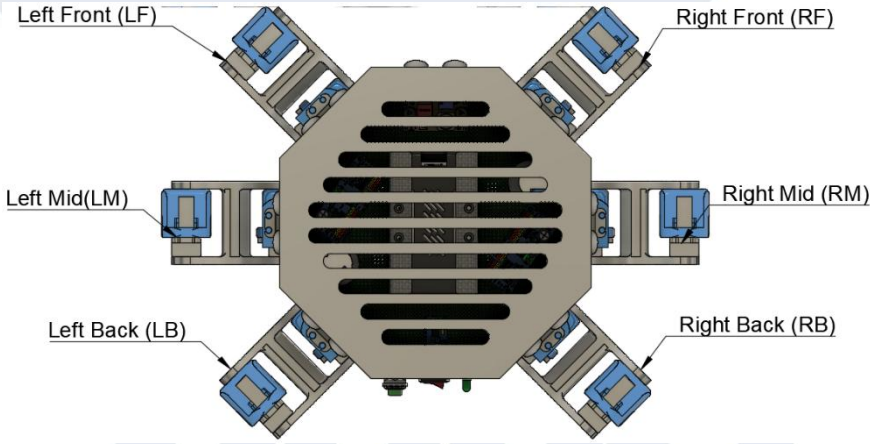
(c)

Gambar 3.31 (a) Kondisi Robot Hectarus sedang Beroperasi (b) Kondisi Robot Hectarus setelah *Heading* diubah (c) Kondisi Robot Hectarus setelah Beberapa Pergerakan

Dari hasil penurunan yang telah dilakukan, dapat dilihat dokumentasinya pada Gambar 3.31. Robot Hectarus pada awalnya berjalan lurus dan diubah *heading*-nya untuk melihat respon pergerakannya. Setelah beberapa pergerakan, robot dapat kembali ke posisi semula dengan *heading* yang sama sebelum *heading*-nya diubah (Gambar 3.31 a).

- *Gait*

Implementasi tiga *gait* pada robot Hectarus dibagi menjadi 2 tahap yaitu pada saat berjalan dan pada saat menaiki tangga. Hal ini terjadi karena algoritmanya yang berubah dengan mengubah kondisi ketinggian kaki pada masing-masing kaki. Implementasi *gait* akan menggunakan Gambar 3.32 untuk penamaan setiap kaki robot Hectarus.



Gambar 3.32 Penamaan Kaki Hectarus

– Kondisi Berjalan

a. Tripod

Implementasi Tripod *gait* dilakukan dengan menerapkan algoritma terlebih dahulu tanpa mengimplementasikan IK. Hal tersebut ditujukan untuk mempermudah integrasi sehingga meminimalisasi *error* yang terjadi. Pola pergerakan Tripod *gait* terdiri dari 2 tahap untuk menyelesaikan 1 siklus pergerakan dimulai dari kaki LF, LB, dan RM mengangkat dengan mengubah nilai derajat femur (β) dan melangkah, kemudian dilanjutkan dengan LM, RB dan RF mengangkat dengan

mengubah nilai derajat femur (β). Untuk *pseudocode* algoritma pertama dari Tripod *gait* dapat dilihat pada Gambar 3.33.

```
declare:
time_delay = 0.2

Algorithm:

LF, LB, RM mengangkat dengan mengubah derajat femur
delay(time_delay)

while True:
    LF, LB, RM melangkah dengan mengubah derajat coxa
    delay(time_delay)

    LF, LB, RM menurunkan kaki
    delay(time_delay)

    LF, LB, RM kembali ke posisi semula dengan coxa kondisi idle
    LM, RB, RG mengangkat kaki dengan mengubah derajat femur
    delay(time_delay)

    LM, RB, RF melangkah dengan mengubah derajat coxa
    delay(time_delay)

    LM, RB, RF menurunkan kaki
    delay(time_delay)

    LM, RB, RF kembali ke posisi semula dengan coxa kondisi idle

    If (stop):
        break
    Else:
        LF, LB, RM melangkah dengan mengubah derajat coxa
        delay(time_delay)
```

Gambar 3.33 *Pseudocode* Algoritma 1 Tripod *Gait*

Dari algoritma pertama tersebut, dilakukan penyesuaian lebih lanjut dengan menambahkan perubahan derajat pada *joint* femur-tibia (γ) karena pada saat implementasi, kaki Hectarus sering menabrak pembatas/tembok pada lokasi pengujian karena ketika mengangkat kaki, lebar robot Hectarus akan bertambah. Penyesuaian juga dilakukan pada saat Hectarus mengangkat kaki dengan mengubah nilai derajat femur.

Setelah penyesuaian tersebut, dilanjutkan dengan implementasi IK pada Tripod *gait*. Implementasi ini dimulai dengan melakukan pengukuran parameter koksa, femur, tibia, dan lebar kaki (ℓ) pada robot Hectarus. Dari hasil pengukuran tersebut, didapatkan nilai koksa = 2,644 cm, femur = 6,436 cm, tibia = 8,122.

Implementasi kemudian dilanjutkan dengan *zeroing* derajat yang dihasilkan dari hasil perhitungan IK dengan nilai derajat yang diterapkan pada servo. Setelah *zeroing* selesai dilakukan, diimplementasikan IK pada algoritma jalan Tripod *gait*.

Implementasi algoritma Tripod *gait* dengan IK dilakukan dengan menggunakan 3 persamaan IK yaitu “IK_tengah” untuk perhitungan kaki tengah, “IK_depan” untuk perhitungan kaki depan dan “IK_belakang” untuk perhitungan kaki belakang. Untuk *pseudocode*-nya dapat dilihat pada Gambar 3.34.

```
declare:
time_delay = 0.2
Function IK_tengah(jarak_maju)
Function IK_depan(jarak_maju)
Function IK_belakang(jarak_maju)
maju = 4
sudut_berdiri = []
sudut_maju = []

Algorithm:
sudut_berdiri = IK_tengah(0), IK_belakang(0), IK_depan(0)
sudut_maju = IK_tengah(maju), IK_belakang(maju), IK_depan(maju)

LF, LB, RM mengangkat dengan mengubah derajat femur dan tibia
delay(time_delay)

while True:
    coxa LF, LB, RM melangkah berdasarkan parameter coxa sudut_maju
    delay(time_delay)

    femur dan tibia LF, LB, RM menurunkan kaki berdasarkan parameter sudut_maju
    delay(time_delay)

    LF, LB, RM kembali ke posisi semula dengan parameter sudut_berdiri
    LM, RB, RG mengangkat kaki dengan mengubah derajat femur dan tibia
    delay(time_delay)

    coxa LM, RB, RF melangkah berdasarkan parameter sudut_maju
    delay(time_delay)

    femur dan tibia LM, RB, RF menurunkan kaki berdasarkan parameter sudut_maju
    delay(time_delay)

    LM, RB, RF kembali ke posisi semula dengan parameter sudut_berdiri

    If (stop):
        break
    Else:
        coxa LF, LB, RM melangkah berdasarkan parameter coxa sudut_maju
        delay(time_delay)
```

Gambar 3.34 *Pseudocode* Algoritma Tripod *Gait* dengan IK

Pada saat implementasi IK pada algoritma Tripod *gait*, didapatkan parameter variabel maju yaitu 4 cm yang diterapkan pada persamaan IK. Parameter tersebut didapatkan dengan menguji jarak pelangkahan kaki paling besar yang dapat dijangkau oleh Hectarus. Parameter yang lebih besar akan membuat kaki robot Hectarus mengubah derajat *joint base*-koksa lebih besar yang membuat kakinya saling bertumbukan satu sama lain baik dengan kaki lainnya, dan membuat komponen atau menjepit kabel servo kakinya. Parameter maju tersebut akan digunakan untuk kedua sisa *gait* lainnya untuk dilakukan pengujian dalam kondisi parameter yang sama.

b. *Wave*

Implementasi *Wave gait* lebih sederhana dan lebih cepat dibandingkan Tripod *gait* karena hanya memodifikasi dari algoritma/pola pergerakan Tripod *gait*. Implementasi langsung dilakukan bersamaan dengan implementasi IK. Pola pergerakan *Wave gait* akan terdiri dari 6 tahap untuk mencapai 1 siklus karena robot harus mengangkat dan melangkah ke-6 kakinya satu per satu. Urutan kaki yang digunakan untuk melakukan *Wave gait* dimulai dari LF, LM, LB, kemudian RB, RM, dan RF. Selama Hectarus sedang mengangkat dan melangkahkan 1 kaki-nya, sisa kakinya akan mengikuti melangkah dengan mengubah nilai derajat *joint base*-koksa (α). Karena kaki Hectarus akan selalu melangkah di setiap siklusnya, maka akan terdapat 6 titik yang perlu dicapai oleh setiap kaki pada 1 siklusnya.

Algoritma *Wave gait* dimulai dengan mendapatkan parameter sudut yang harus ditempuh oleh masing-masing kaki Hectarus dalam 6 tahap. Dengan parameter maju 4 cm, maka jarak maju yang harus ditempuh oleh kaki adalah dari 0,8 – 4 cm dengan kelipatan 0,8 cm. *Pseudocode* algoritma dari *Wave gait* dapat dilihat pada Gambar 3.35.

```
declare:
time_delay = 0.2
Function IK_tengah(jarak_maju)
Function IK_depan(jarak_maju)
Function IK_belakang(jarak_maju)
maju = 4
sudut_berdiri = []
sudut_maju = []
decrement = maju/5

Algorithm:
for i in range (6):
    sudut_maju = IK_tengah(maju), IK_belakang(maju), IK_depan(maju)
    maju = maju - decrement

LF mengangkat kaki dengan mengubah derajat femur dan tibia
delay(time_delay)

while True:
    LF melangkah kedepan, sisa kaki menarik kakinya
    delay(time_delay)

    LM mengangkat kaki dengan mengubah derajat femur dan tibia
    delay(time_delay)

    LM melangkah kedepan, sisa kaki menarik kakinya
    delay(time_delay)

    LB mengangkat kaki dengan mengubah derajat femur dan tibia
    delay(time_delay)

    LB melangkah kedepan, sisa kaki menarik kakinya
    delay(time_delay)

    RB mengangkat kaki dengan mengubah derajat femur dan tibia
    delay(time_delay)

    RB melangkah kedepan, sisa kaki menarik kakinya
    delay(time_delay)

    RM mengangkat kaki dengan mengubah derajat femur dan tibia
    delay(time_delay)

    RM melangkah kedepan, sisa kaki menarik kakinya
    delay(time_delay)

    RF mengangkat kaki dengan mengubah derajat femur dan tibia
    delay(time_delay)

    RF melangkah kedepan, sisa kaki menarik kakinya
    delay(time_delay)

    If (stop):
        break
    Else:
        LF mengangkat kaki dengan mengubah derajat femur dan tibia
        delay(time_delay)
```

Gambar 3.35 *Pseudocode Wave Gait* dengan IK

c. *Tetrapod*

Implementasi *Tetrapod* hampir sama dengan implementasi dengan *Wave gait* yang yang hanya mengubah urutan pergerakan kaki dari algoritma *Wave gait*. Pola *Tetrapod* akan terdiri dari 3 kakinya dimulai dari LF/RB, LB/RM, dan LM/RF. Selama kedua kaki sedang melangkah kedepan, sisa kaki

lainnya merubah derajat *base*-koksa (α) untuk membuat badan Hectarus maju. *Pseudocode* dari algoritma *Tetrapod gait* dapat dilihat pada Gambar 3.36.

```
declare:
time_delay = 0.2
Function IK_tengah(jarak_maju)
Function IK_depan(jarak_maju)
Function IK_belakang(jarak_maju)
maju = 4
sudut_berdiri = []
sudut_maju = []
decrement = maju/2

Algorithm:
for i in range (3):
    sudut_maju = IK_tengah(maju), IK_belakang(maju), IK_depan(maju)
    maju = maju - decrement

LF dan RB mengangkat kaki dengan mengubah derajat femur dan tibia
delay(time_delay)

while True:
    LF dan RB melangkah kedepan, sisa kaki menarik kakinya
    delay(time_delay)

    LB dan RM mengangkat kaki dengan mengubah derajat femur dan tibia
    delay(time_delay)

    LB dan RM melangkah kedepan, sisa kaki menarik kakinya
    delay(time_delay)

    LM dan RF mengangkat kaki dengan mengubah derajat femur dan tibia
    delay(time_delay)

    LM dan RF melangkah kedepan, sisa kaki menarik kakinya
    delay(time_delay)

    If (stop):
        break
    Else:
        LF dan RB mengangkat kaki dengan mengubah derajat femur dan tibia
        delay(time_delay)
```

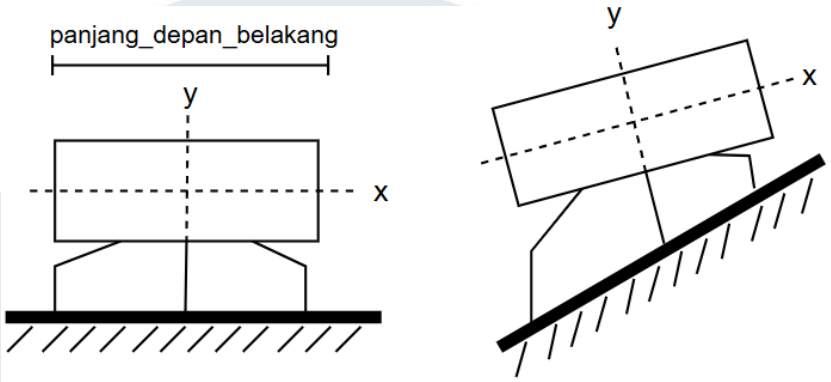
Gambar 3.36 *Pseudocode Tetrapod Gait* dengan IK

- Menaiki Tangga

Pada saat robot Hectarus berada di rintangan tangga, robot Hectarus akan memiliki algoritma *gait*-nya tersendiri. Algoritma tersebut berfungsi untuk mengubah ketinggian kaki belakang, tengah dan depan robot sehingga kondisi tubuh/*base* robot tidak berada dalam kondisi yang sama miringnya dengan lantai. Alasan penggunaan algoritma ini untuk menghindari robot terjungkir pada saat menaiki tangga karena titik tengah robotnya yang berubah karena kemiringan tangga.

Pertambahan dan pengurangan ketinggian kaki dihitung berdasarkan data *pitch* yang dihitung oleh sensor IMU. Data kemiringan tersebut kemudian akan digunakan dalam persamaan untuk menentukan penambahan ketinggian dan pengurangan kaki Hectarus. Karena berhubungan dengan ketinggian kaki, maka

parameter *h* pada IK akan mengalami penyesuaian. Ilustrasi kondisi robot di bidang miring dapat dilihat pada Gambar 3.37.



(a) (b)

Gambar 3.37 (a) Ilustrasi Robot Hectarus di Bidang Datar (b) Ilustrasi Robot Hectarus di Bidang Tangga

Di bidang miring, kondisi robot Hectarus akan mengalami kemiringan sebesar kemiringan tangga. Untuk mengimbangi kemiringan tersebut, maka ketinggian kaki robot belakang harus ditambah dan ketinggian kaki depan robot harus dikurangi bersamaan dengan penambahan ketinggian kaki belakang robot. Dari pernyataan tersebut, dengan menggunakan rumus sinus, pertambahan dan pengurangan ketinggian kaki depan dan belakang robot adalah sebagai berikut.

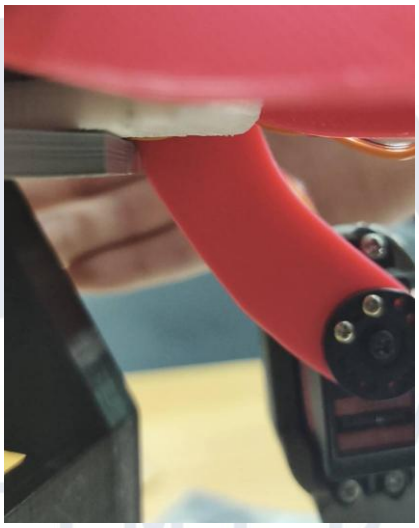
$$\mathrm{h}_{\mathrm{tambah\_kurang}} = \tan(\pm pitch^{o}) * \frac{panjang\_depan\_belakang}{2}$$

Untuk pertambahan ketinggian kaki belakang, maka nilai derajat *yaw*-nya bernilai positif, untuk kaki depan bernilai negatif. Dengan adanya penambahan logika tersebut, maka rumus persamaan *h* pada persamaan (5) menjadi $\boldsymbol{h = \sqrt{femur^2 + tibia^2 - 2 * femur * tibia * \cos\gamma - \sin^2\gamma * femur^2} + h_{tambah\_kurang}}$.

Dalam implementasinya, ketinggian kondisi awal kaki robot Hectarus akan setinggi panjang tibia-nya karena dalam kondisi berdiri, kondisi kaki tibia akan tegak lurus dengan bidang lantainya. Sehingga nilai rumus *h* dapat disederhanakan menjadi $\boldsymbol{h = tibia + h_{tambah\_kurang}}$.

Persamaan ketinggian *h* tersebut yang digunakan dalam implementasi pergerakan pada robot Hectarus. Dalam implementasinya, perubahan ketinggian kaki depan Hectarus tidak selalu sama dengan perubahan ketinggian kaki belakangnya. Hal ini dapat terjadi karena kaki bagian depan Hectarus tidak memiliki area pengurangan ketinggian kaki yang sama dengan penambahan kaki belakang robot. Sebagai ilustrasi, pada Gambar 3.37 (b), kaki bagian depan robot mengalami perubahan yang tidak sebanyak penambahan kaki robot belakang. Jika pengurangan kaki robot depan disamakan dengan penambahan kaki belakang robot, maka kaki depan robot akan bertabrakan dengan badan robotnya sendiri, sehingga perlu dilakukan batasan pada perubahan derajat kaki depan.

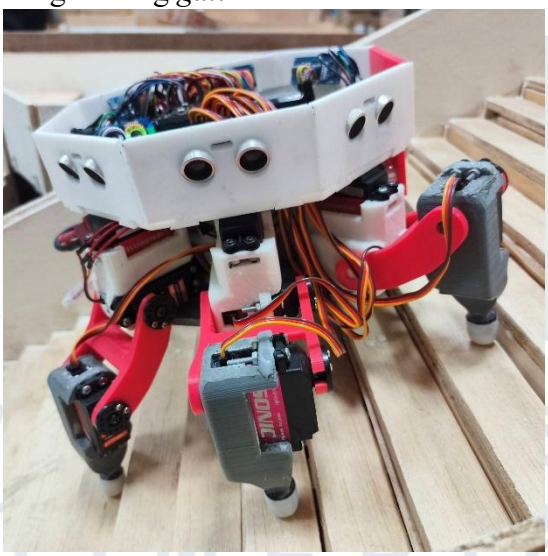
Dalam implementasinya nilai batasan pada kaki bagian depan adalah -15^{o} Nilai derajat kaki bagian depan yang lebih kecil akan membuat kaki depannya menjadi terlalu rendah. Kaki yang terlalu rendah tersebut akan membuat pergerakan kaki robotnya tidak dapat melangkah karena tidak dapat mengangkat kaki karena sudah mencapai batas.



Gambar 3.38 Kaki Belakang Hectarus Menabrak Badan

Dalam implementasinya, kaki belakang robot juga dilakukan batasan derajat sebesar 24º. Ketika nilai derajatnya melebihi nilai batasan tersebut, kaki robot Hectarus akan terlalu memanjang yang membuat *link* femur-nya menabrak badannya baik pada saat melakukan pergerakan melangkah atau ketika dalam kondisi berdiri. Hal tersebut dapat dilihat pada Gambar 3.38 di atas.

Hasil penurunan dan implementasi algoritma penambahan dan pengurangan ketinggian kaki berhasil diintegrasikan pada robot Hectarus dengan tampilan robotnya pada Gambar 3.39. Dalam pergerakan menaiki tangga, waktu perpindahan pergerakan kaki pada robot diperpanjang yang dari 0.2 detik menjadi 0.6 untuk membuat pergerakan robot Hectarus menjadi lebih stabil. Selain itu dalam implementasinya juga sudut *joint base*-koksa robot dilakukan penyesuaian dengan diubah sebesar $\pm$ 10º untuk membuat kakinya sedikit lebar kebelakang. Hal tersebut dilakukan bertujuan untuk membuat kaki belakang robot Hectarus dapat menapak lantai lebih baik karena *end-effector* robot berada di ujung anak tangga. Implementasi kemudian dilanjutkan dengan pergerakan masing-masing *gait*.



Gambar 3.39 Implementasi Penambahan dan Pengurangan Ketinggian Kaki Hectarus

Dengan menggunakan IK, pada ketiga *gait* yang telah diimplementasikan, baik Tripod, *Tetrapod*, dan *Wave gait* tidak

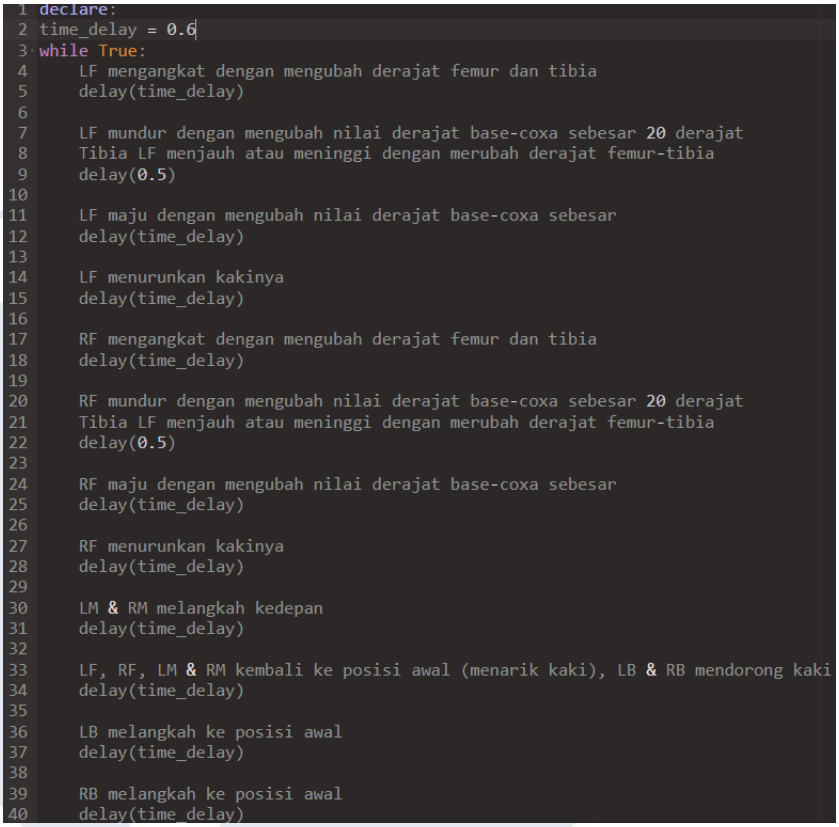
dapat menaiki tangga. Setelah dilakukan observasi, hal tersebut terjadi karena ujung kaki atau *end-effector* kaki robot Hectarus yang tidak cukup kuat untuk menarik badannya maju. Ketika kaki melangkah maju, *joint* femur-tibia akan berubah yang membuat *link* tibia tidak menjadi 90^{o} terhadap bidang lantainya. Hal tersebut yang memungkinkan robot tidak dapat menaiki tangga. Dengan adanya permasalahan tersebut, dilakukan penyesuaian pada robot Hectarus dengan mengubah derajat pergerakan robot yang dihasilkan dengan menggunakan IK menjadi *Forward Kinematic* (FK). Perubahan nilai derajat dengan IK menjadi FK dilakukan dengan hanya mengambil data *joint base*-koksa pada robot dan nilainya tersebut di-*scale* untuk pergerakan yang optimal. Hal tersebut bertujuan untuk tetap menerapkan subsistem pergerakan lurus. Nilai/konstanta skala derajat tersebut didapatkan dengan hasil pengujian nilai derajat yang paling optimal untuk pergerakan maju. Dengan menggunakan FK tersebut, *link* tibia akan tetap 90^{o} terhadap bidang lantainya. Hasil implementasi tersebut kemudian dilakukan pengujian pada ketiga *gait* sebelumnya.

Dengan menggunakan FK, dari ketiga *gait* yang digunakan, hanya Tripod *gait* yang berhasil dalam menaiki tangga. Robot berhasil menaiki tangga tanpa adanya indikasi bahwa robot akan kehilangan kestabilan (terjungkir balik). Robot dapat menaiki tangga dari awal hingga mencapai ujung tangga bersamaan dengan algoritma pergerakan lurus. Pada *Tetrapod* dan *Wave gait*, robot tidak dapat menaiki tangga karena sering terjadinya *slip*/tergelincir ketika akan menaiki tangga.

Hal tersebut dapat disebabkan karena perpindahan badan robot Hectarus yang lambat pada *Tetrapod* dan *Wave gait*. Ketika berada dalam bidang miring/tangga, titik berat robot Hectarus akan berpindah ke bagian belakang robot Hectarus. Pada saat robot Hectarus sedang melangkah, kaki pada robot akan berpindah sedikit demi sedikit pada *Tetrapod* dan *Wave gait*. Karena

pergerakan yang sedikit demi sedikit ini yang membuat robot menjadi sering *slip*/tergelincir. Berbeda halnya dengan Tripod *gait* ketika kaki robot melangkah, badan robot akan langsung bergeser.

Dengan adanya slip pada *gait Tetrapod* dan *Wave*, dibentuk *gait* gabungan antara keduanya (*Tetrapod* + *Wave*) untuk membuat robot Hectarus dapat menaiki tangga dengan menggunakan kedua *gait* tersebut. Pola gabungan ini menerapkan pergerakan kaki robot Hectarus secara satu-persatu untuk kaki dengan dan kaki belakang dan menggerakan kaki bagian tengah secara bersamaan. Pola pergerakan kaki *gait Tetrapod* + *Wave* dimulai dengan pergerakan kaki LF melangkah kedepan, kemudian RF kedepan. Pelangkahan kaki kedepan ini dilakukan dengan menarik kakinya terlebih dahulu dengan mengubah nilai derajat *joint base*-koksa dilanjutkan dengan mengangkat *link* tibia untuk menghindari kakinya tersangkut pada anak tangga. Pergerakan dilanjutkan dengan kaki LM dan RM melangkah maju secara bersamaan. Pergerakannya selanjutnya dilanjutkan dengan robot Hectarus menggerakan tubuhnya kedepan menaiki tangga dengan kaki LF, RF, LM dan RM kembali ke posisi awal (menarik kaki) dan kaki LB dan RB mendorong keseluruhan tubuhnya. Setelah menggerakan keseluruhan tubuhnya, kaki LB melangkah ke posisi semula dilanjutkan dengan kaki RB melangkah ke posisi semula. *Pseudocode* untuk algoritma dari *Tetrapod* + *Wave gait* dapat dilihat pada Gambar 3.40.



```
declare:
time_delay = 0.6
while True:
    LF mengangkat dengan mengubah derajat femur dan tibia
    delay(time_delay)

    LF mundur dengan mengubah nilai derajat base-coxa sebesar 20 derajat
    Tibia LF menjauh atau meninggi dengan merubah derajat femur-tibia
    delay(0.5)

    LF maju dengan mengubah nilai derajat base-coxa sebesar
    delay(time_delay)

    LF menurunkan kakinya
    delay(time_delay)

    RF mengangkat dengan mengubah derajat femur dan tibia
    delay(time_delay)

    RF mundur dengan mengubah nilai derajat base-coxa sebesar 20 derajat
    Tibia LF menjauh atau meninggi dengan merubah derajat femur-tibia
    delay(0.5)

    RF maju dengan mengubah nilai derajat base-coxa sebesar
    delay(time_delay)

    RF menurunkan kakinya
    delay(time_delay)

    LM & RM melangkah kedepan
    delay(time_delay)

    LF, RF, LM & RM kembali ke posisi awal (menarik kaki), LB & RB mendorong kaki
    delay(time_delay)

    LB melangkah ke posisi awal
    delay(time_delay)

    RB melangkah ke posisi awal
    delay(time_delay)
```

Gambar 3.40 *Pseudocode* Algoritma *Gait Tetrapod + Wave*

Pergerakan menaiki tangga dengan *gait Tetrapod + Wave* menghasilkan pergerakan yang lebih stabil secara observasi dengan kondisi robot berada dalam kondisi lurus/sejajar dengan tangga. Namun dalam pergerakan tersebut tentu akan terdapat slip yang kemudian membuat *heading* robot menjadi tidak lurus.

- Pergerakan Menyamping (*Strafe*)

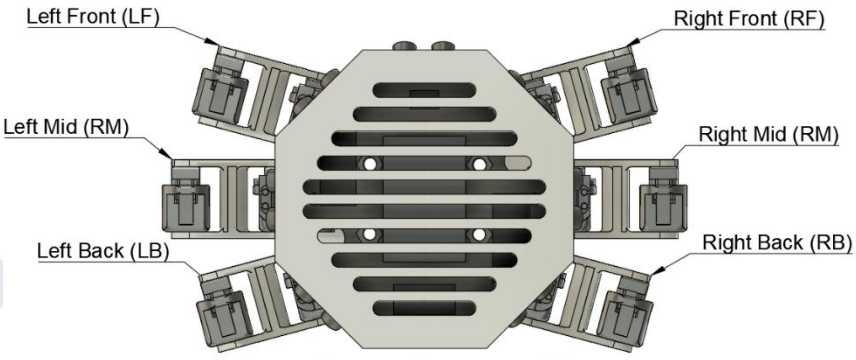
Dalam implementasi pergerakan atau *gait* robot Hectarus, terdapat *gait* tambahan untuk membantu proses pergerakan dari robot Hectarus pada saat pengoperasiannya. Pergerakan atau *gait* ini disebut sebagai pergerakan *strafe* atau pergerakan menyamping layaknya pergerakan kepiting.

Pola pergerakan *strafe* terdiri dari 2 tahap untuk menyelesaikan 1 siklusnya, sama seperti *Tripod gait*. Implementasi pergerakan *strafe* dilakukan langsung bersamaan dengan IK sehingga proses implementasinya dapat dilakukan dengan cepat karena hanya perlu mengubah 1 parameter IK yaitu parameter ℓ pada persamaan (4), pada saat robot Hectarus berada dalam kondisi

berdiri. Besar pengambilan jarak pergerakan menyamping pada robot Hectarus dilakukan sebesar 4 cm untuk mendapatkan pergerakan yang optimal dan cukup cepat.

Dalam pergerakan menyamping robot, terdapat algoritma tambahan untuk menanggulangi pergerakan menyamping yang membuat *heading* robot berubah. *Heading* robot yang dapat berubah ini dapat disebabkan karena permukaan yang tidak rata seperti adanya batu, jalan pecah atau kelereng yang membuat *heading*-nya berubah sehingga diperlukan adanya pengoresi *heading error* tersebut. Algoritma pengoreksi *heading* tersebut dilakukan dengan menggunakan *gait* untuk berbelok dan *gait Strafe*.

Pola pergerakan dari pergerakan *strafe* dimulai dari sudut α kaki LF, LB, RF dan RB berubah sebesar $\pm 30^{o}$ untuk membuat luas robot Hectarus menjadi lebih memanjang. Ilustrasi kondisi perubahan sudut α kaki LF, LB, RF dan RB robot Hectarus dapat dilihat pada Gambar 3.41. Perubahan sudut α tersebut ditujukan untuk membuat robot Hectarus dapat beroperasi pada kondisi jalan yang lebih sempit.



Gambar 3.41 Ilustrasi Robot Hectarus Setelah Perubahan Sudut α

Pada kondisi normalnya (tidak ada *heading error*), urutan pola pergerakan dari pergerakan *strafe* dimulai dari kaki LF, LB, dan RM mengangkat dengan mengubah nilai derajat femur (β), kemudian ketiga kaki tersebut kembali turun namun diikuti dengan perubahan nilai derajat femur-tibia (γ). Jika pergerakan *strafe*-nya menyamping ke arah kiri, maka kaki bagian kiri (LF dan LB) akan

melangkah ke samping kiri menjauhi badan/*base* robot, kaki bagian kanan (RM) akan mendekati badan/*base* robot, jika menyamping ke arah kanan, maka arah pergerakan tibia-nya dilakukan dengan arah yang sebaliknya. Setelah ketiga kaki menurunkan kakinya diikuti dengan perubahan nilai γ, ketiga kaki tersebut kembali ke posisi *idle* atau berdiri diikuti dengan ketiga kaki lainnya mengangkat dan melakukan proses pergerakan yang sama dengan ketiga kaki sebelumnya.

Jika dalam pergerakan menyampingnya terdapat *heading error*, maka akan terdapat pergerakan kaki yang berubah bergantung pada kondisi pergerakan menyampingnya (ke kiri atau ke kanan). Untuk mencapai titik *finish* dalam pengujian robot Hectarus, robot hanya akan menggunakan pergerakan menyamping ke arah kiri. Dalam pergerakan menyamping ke arah kiri, jika terdapat *error heading*, maka LF & LB akan digunakan sebagai pengoreksi *heading*. Algoritma pengoreksian *heading* dilakukan dengan LF, LB, dan RM mengangkat namun sudut *joint base*-koksa kaki LB dan LF juga akan langsung ikut mengoreksi berdasarkan nilai *error heading*. Jika robot memiliki *error heading* $< 0^o$ (*yaw* bernilai positif yang berarti berotasi berlawanan arah jarum jam), maka nilai sudut *joint base*-koksa LF akan berubah, begitu juga sebaliknya jika *error heading* $> 0^o$, sudut *joint base*-koksa LB yang akan berubah. Pergerakan kemudian dilanjutkan dengan ketiga kaki tersebut turun dan kembali ke posisi awal diikuti dengan ketiga kaki berikutnya mengangkat. Ketiga kaki tersebut tidak digunakan untuk pengoreksi *heading*. Dalam pengimplementasian algoritma tersebut, diterapkan *threshold* sebesar $\pm 3^o$ pada *error heading* untuk menghindari pergerakan robot yang berosilasi karena terus melakukan koreksi jika nilai *error heading*-nya $<> 0^o$.

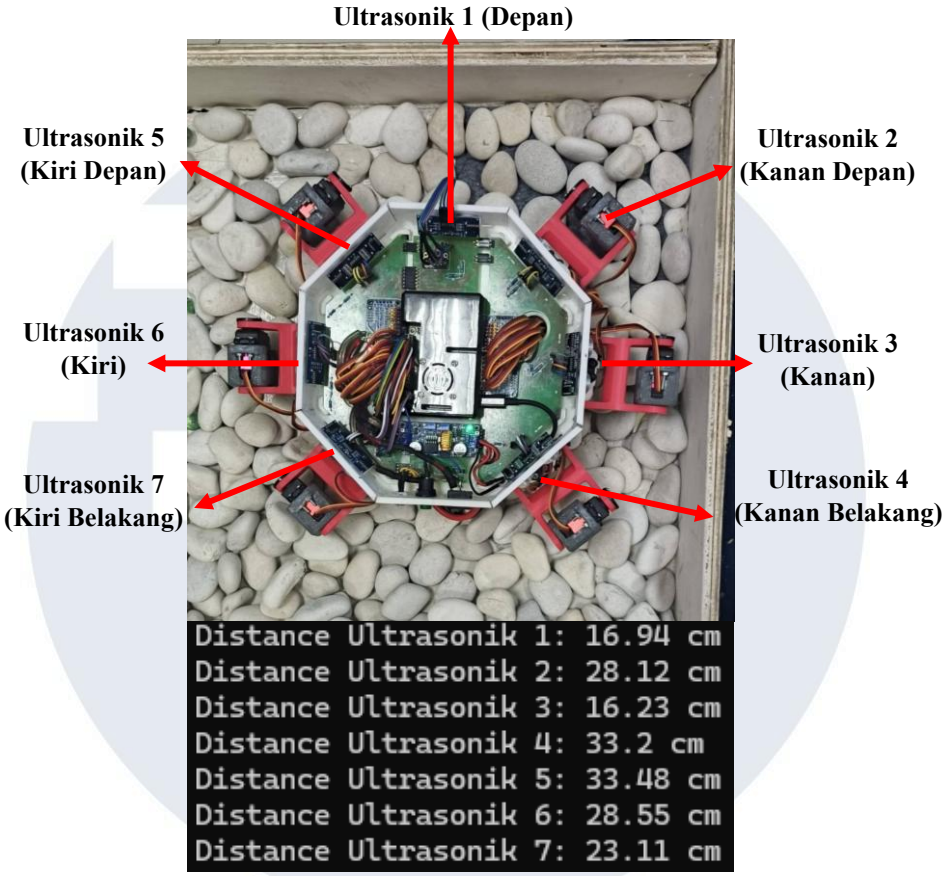
Jika dalam pergerakan menyampingnya ke arah kanan, maka kaki yang digunakan untuk pengoreksi *heading* adalah kaki RF dan RB dengan algoritma yang sama. Untuk *pseudocode* algoritma pergerakan *Strafe* ke arah kiri dengan pengoreksi *heading*-nya dapat dilihat pada Gambar 3.42.

```
declare:
time_delay = 0.2
Function IK_tengah(geser)
Function IK_belakang(geser)
Function IK_depan(geser)
geser_kiri = -4
geser_kanan = 4

BMX160 = BMX160

Algorithm:
sudut_berdiri = IK_tengah(0), IK_belakang(0), IK_depan(0)
sudut_geser = IK_tengah(geser_kiri), IK_belakang(geser_kiri), IK_depan(geser_kiri)

coxa LF, LB, RF, dan RB berubah sebesar +- 30 derajat

while True:
    yaw = BMX160.Sensor_Get_Yaw()

    if yaw <= -3 or yaw >= 3:
        femur dan tibia LF, LB, dan RM mengangkat kaki bersamaan diikuti dengan koreksi coxa LF dan LB
    else:
        femur dan tibia LF, LB, dan RM mengangkat kaki bersamaan
    delay(time_delay)

    femur dan tibia LF, LB, dan RM menurunkan kaki berdasarkan parameter sudut_geser
    delay(time_delay)

    LF, LB, RM kembali ke posisi semula dengan parameter sudut_berdiri
    femur dan tibia LM, RF, RB menangkat kaki dengan mengubah derajat femur dan tibia
    delay(time_delay)

    femur dan tibia LM, RF, RB menurunkan kaki berdasarkan parameter sudut_geser
    delay(time_delay)

    LM, RF, RB kembali ke posisi dengan parameter sudut_berdiri

    If (stop):
        break //berdasarkan data jarak sensor samping
```

Gambar 3.42 *Pseudocode Gait Strafe* ke arah kiri Hectarus

- Pengecekan Halangan

Subsistem pengecekan halangan robot Hectarus diimplementasikan dengan melihat fungsionalitas semua sensor ultrasonik yang digunakan. Sensor ultrasonik berhasil diimplementasikan dan dapat mendeteksi halangan tanpa mendeteksi kakinya sendiri. Hasil implementasi tersebut dapat dilihat pada Gambar 3.43 yang diuji langsung di atas arena.



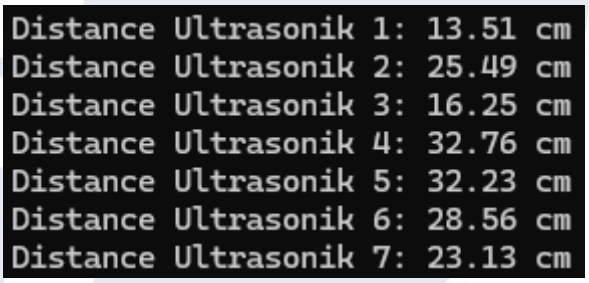
Gambar 3.43 Implementasi Fungsionalitas Ultrasonik di atas Arena

Dari hasil implementasi pada Gambar 3.43, terdapat sensor ultrasonik yang menunjukkan data yang kurang akurat. Hal ini dapat dilihat pada Tabel 3.10 yang berisi perbandingan pada masing-masing sensor antara pengukuran jarak manual dengan jangka sorong & mistar dan pengukuran jarak dengan sensor ultrasonik.

Tabel 3.10 Perbandingan Hasil Pengukuran Sensor dengan Pengukuran Manual

Ultrasonik ke-	**Pengukuran Sensor (cm)**	**Pengukuran Manual (cm)**	***Error*/ Galat (%)**
1 (Depan)	16,94	13,48	25,67
2 (Kanan Depan)	28,12	18,6	51,18
3 (Kanan)	16,23	11,92	36,16
4 (Kanan Belakang)	33,2	18,6	78,49
5 (Kiri Depan)	33,48	23,4	43,08
6 (Kiri)	28,55	41,4	31,04
7 (Kiri Belakang)	23,11	22,1	4,57

Dari hasil pengukuran pada Tabel 3.10, dapat diketahui bahwa hasil pengukuran dari sensor ultrasonik memiliki jarak yang tidak akurat hingga mencapai 78% untuk nilai *error*-nya. Hal ini dapat disebabkan karena sudut kemiringan sensor ultrasonik yang membuat sinyal gelombang yang di-*transmit* oleh sensor tidak terpantulkan kembali mengenai sisi *receiver* sensor ultrasonik. Selain itu, nilai *error* pengukuran jarak sensor yang besar juga dapat disebabkan karena ketinggian halangan/tembok yang kurang tinggi sehingga tidak dapat dideteksi oleh sensor dengan baik. Hal ini dibuktikan pada Gambar 3.44.



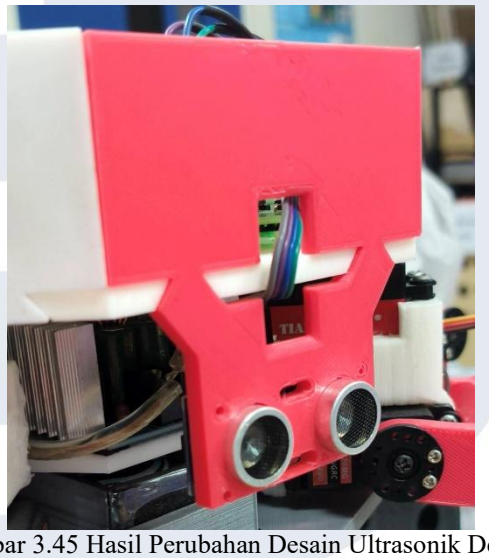
Gambar 3.44 Hasil Pengukuran oleh Sensor setelah Penghalang Bagian Depan ditingkatkan

Pada Gambar 3.44, didapatkan hasil pengukuran oleh sensor yang terbaru setelah ketinggian penghalang bagian depan robot Hectarus ditingkatkan. Hasil pengukuran menunjukkan bahwa nilai pengukuran oleh sensor tersebut menjadi jauh lebih akurat mendekat nilai sesungguhnya atau pengukuran dengan jangka sorong. Namun hasil pengukuran oleh sensor tersebut didapati nilainya yang tidak konsisten dan juga membuat hasil pengukurannya memiliki nilai yang sama meskipun diletakkan pada jarak dan posisi yang berbeda.

Dari hasil implementasi tersebut, dilakukan penyesuaian terdapat penggunaan dan implementasi sensor ultrasonik pada robot Hectarus. Pada Gambar 3.43 dan 3.44, ultrasonik bagian kiri (ultrasonik 6) dan bagian kanan (ultrasonik 3) masih dapat digunakan hanya untuk "*sensing*" atau mendeteksi ada atau tidaknya penghalang namun tidak dapat mengetahui jarak pastinya karena nilai *error* yang sudah dijelaskan sebelumnya. Sehingga dalam pengoperasiannya, hanya sensor ultrasonik depan, kiri dan kanan yang digunakan. Kegunaan

sensor ultrasonik yang hanya sebagai "*sensing*" membuat robot Hectarus tidak dapat mencari titik tengah dari arena.

Ultrasonik bagian depan robot Hectarus dilakukan penyesuaian dengan mengubah bentuk desain peletakan ultrasonik untuk mendapatkan hasil pengukuran yang akurat dan konsisten. Hasil perubahan desain ini dapat dilihat pada Gambar 3.45.



Gambar 3.45 Hasil Perubahan Desain Ultrasonik Depan

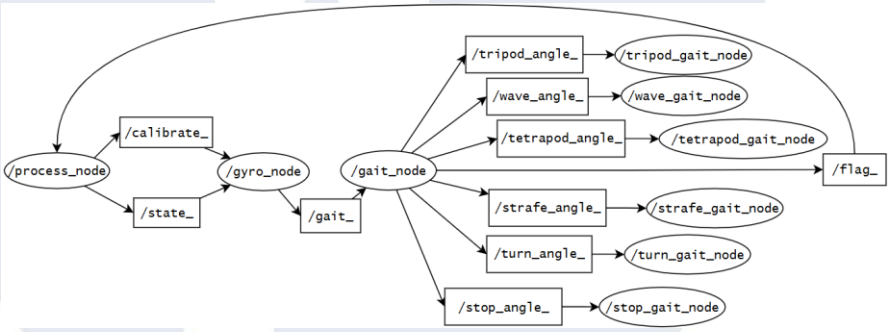
```
Distance Ultrasonik 1: 12.8 cm
Distance Ultrasonik 2: 28.1 cm
Distance Ultrasonik 3: 16.25 cm
Distance Ultrasonik 4: 40.73 cm
Distance Ultrasonik 5: 34.81 cm
Distance Ultrasonik 6: 31.7 cm
Distance Ultrasonik 7: 30.1 cm
```

Gambar 3.46 Hasil Pengukuran Sensor setelah Perubahan Desain

Perubahan desain pada ultrasonik depan dibuat dengan menurunkan posisi ultrasonik dan menghilangkan sudut kemiringan pada peletakan ultrasonik. Perubahan tersebut membuat hasil pendeteksian jarak sensor menjadi lebih konsisten dengan nilai *error*/galat-nya adalah sebesar 6,58%.

- Pergerakan Otomatis

Subsistem pergerakan otomatis pada dasarnya merupakan hasil dari penggabungan seluruh implementasi dan subsistem yang sudah dijelaskan sebelumnya. Implementasi subsistem Pergerakan Otomatis diintegrasikan dengan menggunakan ROS2 berbasis Python dengan jenis komunikasi *publisher-subscriber* untuk hubungan antar *node*-nya dalam pengoperasiannya. Hubungan antar *node* tersebut dapat dilihat pada *rqt_graph* pada Gambar 3.47.



Gambar 3.47 Hubungan antar *Node* pada Robot Hectarus

Terdapat total 9 *node* yang digunakan pada robot Hectarus yang dapat dibagi menjadi 2 bagian yang berurutan dan dijelaskan sebagai berikut.

1. Pengambilan dan Pengolahan Data

Sesuai namanya, bagian pengambilan dan pengolahan data merupakan *node-node* yang digunakan untuk mengambil data jarak dari sensor ultrasonik dan data kemiringan oleh BMX160. Bagian ini terdiri dari *node* "*/process_node*", "*/gyro_node*", dan "*/gait_node*"

– *process_node*

Node "*/process_node*" merupakan *node* yang berisikan pengambilan data jarak oleh ultrasonik sekaligus logika pemilihan jalur yang harus dilakukan oleh robot, baik maju, berbelok kiri, berbelok kanan atau berhenti. Proses pengambilan data dimulai dari pengambilan jarak bagian depan robot. *Threshold* yang ditetapkan untuk jarak bagian depan berbeda-beda bergantung pada kondisi atau lokasi robot pada saat pergerakan. Perbedaan jarak atau *threshold* bagian

depan tersebut didapatkan berdasarkan hasil pengujian/implementasi untuk pergerakan robot yang optimal seperti pergerakan menuju tangga, pergerakan saat di tangga dan pergerakan setelah menaiki tangga. Jika jarak bagian depan robot masih melebihi batas *threshold*-nya, robot akan terus melakukan pergerakan maju. Jika kurang dari batas *threshold*-nya, maka akan dilakukan pengambilan data jarak bagian sisi kiri dan kanan robot. Jika jarak sisi kanan lebih dekat dibandingkan sisi kiri, maka sisi kanan terdapat halangan sehingga robot harus berbelok kiri, begitu juga sebaliknya.

Penanda robot harus melakukan pergerakan maju, berbelok kiri, kanan atau berhenti akan dikirimkan ke *topic* yang dinamakan "/*state_*" Nilai tersebut kemudian akan diterima oleh *node* "/*gyro_node*". Selain *topic* "/*state_*", juga terdapat *topic "/calibrate_"*. *Topic* tersebut merupakan *topic* yang digunakan untuk mengkalibrasi perhitungan *yaw* pada *node* "/*gyro_node*" jika robot melakukan pergerakan belok kiri atau kanan.

Pada *node* "/*process_node*", terdapat parameter yang didapatkan *topic* "/*flag*". Parameter *topic* "/*flag*" tersebut digunakan sebagai penanda jika robot berada dalam kondisi miring atau tangga, jarak *threshold* jarak bagian depan robot diubah menjadi 3 cm untuk menghindari robot mendeteksi anak tangga sehingga membuatnya berbelok di tengah tangga.

- *gyro_node*

Node "/*gyro_node*" merupakan bagian pengambilan data *yaw* dan *pitch* oleh sensor *gyroscope*. Pada saat robot diaktifkan, "/*gyro_node*" akan langsung aktif dan terus menghitung besar perubahan nilai *yaw* dan *pitch*. Ketika *node* "/*gyro_node*" menerima data dari *topic* "*state_*" yang dikirim oleh *node* "/*process_node*", data perubahan nilai *yaw* dan *pitch* akan dikirimkan ke *node* "/*gait_node*" dengan *topic* "/*gait_*".

Pada "/*gyro_node*", akan ada parameter yang diterima yang dikirim oleh "/*process_node*" dengan *topic* "/*calibrate_*". *Topic* tersebut merupakan penanda bahwa nilai *yaw* pada *gyroscope* harus ditambah atau dikurangi 90^{o}. Jika robot berbelok kiri, maka nilainya dikurangi 90^{o} sehingga nilai *yaw*-nya menjadi 0^{o}, jika berbelok kekanan maka ditambah 90^{o}. Hal tersebut dibuat untuk membuat robot tidak kembali ke posisi semula karena nilai *yaw*-nya yang berubah menjadi $\pm$ 90^{o}.

- *gait_node*

  Node "/*gait_node*" berisikan implementasi IK yang telah dihitung sebelumnya. Pada *node* ini juga berisikan parameter yang digunakan untuk memilih jenis *gait* yang akan digunakan. Seluruh data berubah *state* yang dikirimkan oleh *node* "/*process_node*" dan nilai *yaw* pada *node* "/*gyro_node*" digunakan pada *node* ini untuk mendapatkan nilai derajat yang harus ditempuh oleh servo kaki dengan menggunakan IK. Hasil perhitungan dan nilai derajat kaki servo ditampung dalam sebuah array yang akan dikirimkan ke *node* selanjutnya yang merupakan bagian Pergerakan Robot.

  Node "/*gait_node*" juga mengirimkan sebuah data dengan *topic* "/*flag_*" ke *node* "/*process_node*" yang telah disebutkan sebelumnya. *Topic* tersebut merupakan *topic* yang berisikan penanda bahwa jarak *threshold* pada ultrasonik depan diperpendek menjadi 0 cm atau dengan kata lain, pendeteksian ultrasonik depan dimatikan untuk sementara.

2. Pergerakan Robot

   Bagian Pergerakan Robot pada dasarnya merupakan *node-node* yang berisikan logika pergerakan kaki robot. *Node* tersebut merupakan "*tripod_gait_node*", "*wave_gait_node*", "*tetrapod_gait_node*", "*strafe_gait_node*", "*turn_gait_node*", dan "*stop_gait_node*". *Node-node* tersebut digunakan sebagai

implementasi untuk menggerakan kaki servo berdasarkan pola pergerakan atau *gait*-nya. Data sudut pergerakan tersebut didapatkan setelah *node* "/*gait_node*" mengirimkan data ke *topic* yang berkaitan dengan jenis *gait*-nya.

3.2.2 Hambatan Implementasi

Selama proses implementasi robot Hectarus, mekanisme pegas pada robot Hectarus sering mengalami patah karena tidak kuat menahan tekanan dari samping. Kekuatan material PETG pada pegas juga diperkirakan mengalami penurunan setelah diberikan perekat untuk menghubungkan antara pegas dengan ujung kaki/*end-effector* Hectarus.

Daya baterai pada robot Hectarus juga tidak jarang harus dilakukan pengisian kembali sehingga diperlukan waktu untuk menunggu hingga daya baterai robot kembali terisi.

Pada awal implementasi robot Hectarus, protokol komunikasi I2C yang digunakan pada sensor IMU dan servo *driver* dapat bekerja dengan baik tanpa ada putus koneksi. Namun setelah beberapa pengujian dan percobaan algoritma dalam implementasinya, komunikasi pada I2C robot Hectarus menjadi sering putus koneksi.

3.2.3 Solusi yang Diterapkan

Solusi yang diterapkan untuk pegas pada robot Hectarus adalah dengan menggunakan pegas yang digunakan dengan karet paking. Meskipun nilai elastisitasnya berkurang, namun diharapkan dapat memberikan suspensi meskipun dengan nilai minimal untuk dapat mengurangi getaran pada robot.

Solusi yang diterapkan pada daya baterai robot yang harus dilakukan pengisian adalah dengan menambah jumlah baterai. Desain pada robot Hectarus dibuat sehingga proses penggantian baterai dapat dilakukan dengan mudah tanpa harus melepas semua kerangka badan robot. Robot

Hectarus dapat menggunakan baterai pengganti sambil menunggu proses pengisian daya baterai yang sudah habis kembali terisi.

Protokol komunikasi I2C pada servo *driver* dan sensor IMU yang sering mengalami putus koneksi setelah beberapa pengujian dan penerapan algoritma diperbaiki dengan mengubah tegangan input pada servo *driver* dan sensor IMU. Secara *wiring*, sensor IMU bekerja dengan tegangan 5 Volt yang disuplai oleh Raspberry Pi melalui pin 5 Volt dan servo *driver* bekerja dengan tegangan 5 Volt yang disuplai melalui tegangan output pada *DC-DC Buck Converter* XL4015. Sensor dan *driver* yang disuplai dengan tegangan 5 Volt membuat tegangan pada pin I2C (SDA dan SCL) modul tersebut memiliki nilai tegangan sebesar 5 Volt. Secara spesifikasi, pin pada Raspberry Pi 4B bekerja pada nilai tegangan sebesar 3,3 Volt, termasuk nilai tegangan pada pin I2C Raspberry Pi 4B. Perbedaan tegangan tersebut yang kemudian membuat protokol komunikasi I2C sering menjadi putus koneksi. Dengan adanya perbedaan tegangan tersebut, sensor IMU dan servo *driver* kemudian disuplai dengan tegangan 3,3 Volt dengan pin 3,3 Volt yang ada pada Raspberry Pi 4B.