IMPLEMENTASI KENDALI FUZZY-PID SEBAGAI METODE *WALL FOLLOWING* PADA ROBOT *HEXAPOD* KONTES ROBOT PEMADAM API INDONESIA (KRPAI)

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK KONTROL

Ditujukan untuk memenuhi persyaratan

memperoleh gelar Sarjana Teknik

MEILAN SARBAINI SIREGAR

NIM. 145060301111077

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2018

LEMBAR PENGESAHAN

IMPLEMENTASI KENDALI FUZZY-PID SEBAGAI METODE *WALL FOLLOWING* PADA ROBOT *HEXAPOD* KONTES ROBOT PEMADAM API INDONESIA (KRPAI)

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK KONTROL

Ditujukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

MEILAN SARBAINI SIREGAR
NIM.145060301111077

Skripsi ini telah direvisi dan disetujui oleh dosen pembimbing
pada tanggal 24 Juli 2018

Mengetahui,
Ketua Jurusan Teknik Elektro

Ir. Hadi Suyono, S.T., M.T., Ph.D., IPM
NIP. 19730520 200801 1 013

Dosen Pembimbing

M. Aziz Muslim, S.T., M.T., Ph.D.
NIP. 19741203 200012 1 001

JUDUL SKRIPSI:

IMPELEMENTASI KENDALI FUZZY-PID SEBAGAI METODE *WALL FOLLOWING* PADA ROBOT *HEXAPOD* KONTES ROBOT PEMADAM API INDONESIA (KRPAI)

Nama Mahasiswa : MEILAN SARBAINI SIREGAR

NIM : 145060301111077

Program Studi : TEKNIK ELEKTRO

Konsentrasi : TEKNIK KONTROL

Komisi Pembimbing :

Ketua : M. Aziz Muslim ST., MT., Ph.D. .....

Tim Dosen Penguji :

Dosen Penguji 1 : Dr.Ir. Erni Yudaningtyas. MT .....

Dosen Penguji 2 : Ir. Purwanto. M.T .....

Dosen Penguji 3 : Dr. Ir. Bambang Siswojo. MT .....

Tanggal Ujian : 23 Juli 2018

SK Penguji : 1505 tahun 2018

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya dan berdasarkan hasil penelusuran berbagai karya ilmiah, gagasan dan masalah ilmiah yang diteliti dan diulas di dalam Naskah Skripsi ini adalah asli dari pemikiran saya. Tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu Perguruan Tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah Skripsi ini dapat dibuktikan terdapat unsur-unsur jiplakan, saya bersedia Skripsi dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, pasal 25 ayat 2 dan pasal 70).

Malang, 24 Juli 2018

Mahasiswa,

MEILAN SARBAINI SIREGAR

NIM. 145060301111077

Teriring Ucapan Terima Kasih kepada:

Ayahanda dan Ibunda tercinta

RINGKASAN

Meilan Sarbaini Siregar, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Juli 2018, *Implementasi Kendali Fuzzy-PID sebagai Metode Wall Following pada Robot Hexapod KRPAI*, Dosen Pembimbing: M. Aziz Muslim.

Kontes Robot Cerdas Pemadam Api Indonesia (KRPAI) divisi berkaki adalah salah satu kategori perlombaan robot yang sering diselenggarakan oleh Direktorat Jendral Pendidikan Tinggi (Dirjen Dikti) setiap tahun. Pada kompetisi ini, Robot memiliki misi untuk mampu memadamkan titik api berupa lilin, setiap robot peserta diwajibkan mampu memiliki kemampuan bernavigasi dan adaptasi terhadap lingkungan pada arena pertandingan yang berbentuk labirin yang dibatasi oleh dinding dinding yang menggambarkan kondisi rumah. Untuk tujuan tersebut robot diharap mampu untuk melakukan telusur dinding (*wall following)* dengan cara membaca jarak robot terhadap dinding dengan menggunakan sensor ultrasonik Parallax PING, Dimana untuk menjaga kondisi tersebut dibutuhkan satu kontrol yang mampu mengkombinasikan gerak robot terhadap jarak ke dinding. Kontroler yang umum digunakan yaitu kontrolir *Proportional Integral* dan *Derivatif (PID)* dan pada umumnya menggunakan metode *handtuning (trial dan eror)* dalam mendapatkan parameter PID namun dirasakan belum baik dalam proses tuning nilai parameter PID karena membutuhkan waktu yang lama dalam proses tuningnya.

Metode *wall following* dengan metode fuzzy PID merupakan satu cara untuk mendapatkan parameter kontrol PID dengan mempertimbangakn kondisi robot saat berada di arena pengujian. Dengan besar jarak robot yang terbaca adalah sebesar 16 cm dari dinding untuk posisi robot saat berada ditengah lajur kemudian didefinisikan sebagai nilai *setpoint,* dan didefinisikan juga nilai *setpoint atas* sebesar 17 cm dan *setpoint bawah* sebesar 16 cm, sebagai toleransi kondisi robot saat ditengah lajur, kemudian nilai tersebut akan menentukan eror yang akan ditentukan oleh *fuzzy logic algorithm* dengan nilai diantara 0 dan 1, yang akan menentukan parameter Kp, Ki, dan Kd, untuk kondisi dekat, sedang dan jauh dari dinding.

Hasil pengujian didapatkan bahwa robot mampu melakukan *wall following* dengan berbagai lintasan yaitu lintas lurus, berbelok kekiri dan kekanan, hal ini dikarenakan kemampuan robot untuk melakukan navigasi keseluruhan merupakan kombinasi dari gerak robot saat melakukan telusur untuk bentuk lintasan tersebut, kemudian dengan menetapkan bahwa waktu untuk mencapai nilai *steady* adalah sebesar 0.2 s, didapatkan hasil error rata-rata aksi kontrol yaitu sebesar 2.997 % saat telusur pada lintasan lurus, kemudian eror sebesar 3,472% untuk telusur pada lintasan lurus dengan gangguan, dan untuk telusur dinding kanan dengan lintasan berbelok kekanan didapat eror sebesar 3.409% dan berbelok kekiri sebesar 3.282%, Dimana nilai aksi kontrol ini masih dianggap layak dikarenakan error masih lebih kecil dari 5%.

Kata kunci*:* KRPAI divisi berkaki, Sensor Ultrasonik, fuzzy-PID, *wall following*, parameter PID

SUMMARY

Meilan Sarbaini Siregar, *Department of Electrical Engineering, Faculty of Engineering Brawijaya University July 2018, Implementation Fuzzy-PID control as a method of Following Wall Hexapod robots KRPAI, Academic Supervisor*: Academic supervisor M. Aziz Muslim.

Intelligent fire fighter Robot contest Indonesia (KRPAI) Legged Division-is one of the robot race categories are often organized by the Directorate General of higher education (Directorate General of higher education) each year. At this competition, the Robot has a mission to quench the fire point in the form of candles, each robot the participants are required to have the capability of being able to navigate and adaptation to the environment in an arena match that mazes are delimited by Wall wall that depicts the condition of the home. For the purpose of the robot should be able to do a search of the wall (wall following) and how to read the distance the robot against the wall by using Ultrasonic sensors Parallax PING, where the conditions required to maintain a control that being able to combine the motion of robots against the distance to the wall. Commonly used controllers namely controller Proportional Integral Derivative (PID) and and generally use the method handtuning (trial and error) in getting the parameters of the PID has not been felt yet either in the process of tuning PID parameter values because takes a long time in the process of tuningnya.

A method of fuzzy method wall following PID is one way to get the PID control parameters with he conditions the robots in the arena for testing. With the great distance of robots that read is of 16 cm from the wall for the position of the robot while the middle lanes is then defined as the value of the setpoint, setpoint value is also defined over by 17 cm and lower setpoint of 16 cm, as tolerance is a condition when the robot in the middle lanes, then that value will specify the error will be determined by fuzzy logic algorithm with a value between 0 and 1, which will determine the parameters, Ki, Kp and Kd, for the condition of the near, medium and far from the wall .

The test results obtained that the robot is able to do a wall following with various path i.e. the cross-straight, turn kekiri and to, this is due to the ability of robots to perform navigation overall is a combination of motion of the robot when do a search for the shape of the path, then by specifying that the time to reach steady value is 0.2 s, obtained as a result of the average error of the control action i.e. of 2,997% search on the path is straight, then error amounted to 3.472% to search on a trajectory straight with the disorder, and the search for the right wall with the path turns to the obtained error of 3,409% and turn kekiri of 3,282%, where the value of the control action is still considered worthy due to the error is still smaller than 5%.

Key words: KRPAI Legged Division, Ultrasonic Sensor, fuzzy-PID, wall following, PID parameters,

KATA PENGANTAR

Alhamdulillah, puji dan syukur penulis panjatkan kepada Allah SWT, karena atas segala petunjuk dan nikmat-Nya lah skripsi ini dapat diselesaikan. Skripsi berjudul "*Implementasi kendali fuzzy-PID sebagai metode wall following pada Robot Hexapod KRPAI*" ini disusun untuk memenuhi persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

Penulis menyadari bahwa dalam penyusunan skripsi ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, dengan ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada:

- Ayah Syamsuddin Siregar, Ibu Timeal Harahap selaku orang tua penulis atas segala inspirasi, nasehat, kasih sayang, perhatian dan kesabarannya didalam membesarkan dan mendidik penulis, serta telah banyak mendoakan kelancaran penulis hingga terselesaikannya skripsi ini,.
- Abang Syati Manaharawan Siregar, Raja Surya Sarbaini Siregar, dan Adik Putri Juni Aldina Sari Siregar Atas doa dan dukungan yang diberikan kepada penulis
- Yang Terhormat Bapak Hadi Suyono, ST., MT., Ph.D., IPM selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya.
- Yang Terhormat Ibu Ir. Nurussa'adah, MT., selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya.
- Yang Terhormat Bapak M.Aziz Muslim, S.T., M.T., Ph.D selaku Dosen Pembimbing atas segala bimbingan, pengarahan, saran, dan kritik yang telah diberikan selama proses pengerjaan skripsi.
- Seluruh dosen pengajar Jurusan Teknik Elektro Universitas Brawijaya,
- Seluruh staff recording Jurusan Teknik Elektro Universitas Brawijaya
- Keluarga Besar "Tim Robot 2014" atau asisten Laboratorium Mekatronika angkatan 2014 atas segala pengalaman, kebersamaan dan bantuan selama 3 tahun ini menjadi anggota tim robotika dan asisten.
- Keluarga Besar Tim Robotika TEUB terkhusus sub divisi KRPAI atas segala pengalaman, kebersamaan dan bantuan selama ini.

- Keluarga Besar Mahasiswa Teknik Elektro Universitas Brawijaya,
- Saudara – sodari "DIODA" angkatan 2014 atas segala bantuan dan kebersamaan yang telah diberikan selama 4 tahun ini.
- Seluruh teman-teman serta semua pihak yang tidak mungkin bagi penulis untuk mencantumkan satu-persatu, terimakasih banyak atas bantuan dan dukungannya.

Pada akhirnya, penulis menyadari bahwa skripsi ini masih belum sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun. Penulis berharap semoga skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi serta bagi masyarakat.

Malang, Juli 2018

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

No. Judul Halaman

DAFTAR GAMBAR

No. Judul Halaman

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan Ilmu pengetahuan menuntut kita untuk mampu melakukan berbagai macam inovasi dalam bidang teknologi, salah satu bentuk dari inovasi yang telah banyak digunakan di masa kini yaitu dalam bidang Robotika, dimana kita ketahui sendiri perkembangan Robotika di Indonesia sendiri sudah pesat. Di bidang industri misalnya, sudah banyak digunakan robot-robot sebagai pengganti tugas manusia yang dapat mempermudah pekerjaan pada hal hal yang cukup sulit untuk mampu dikerjakan oleh manusia dan dapat membahayakan keselamatan manusia. Di bidang yang lain yaitu pendidikan, teknologi robotika juga sudah mulai diperkenalkan sejak tingkat Sekolah Dasar.

Direktorat Jendral Riset Teknologi dan Pendidikan Tinggi (DIRJEN RISTEKDIKTI) mewadahi hal tersebut dengan mengadakan berbagai kompetisi dengan tujuan untuk mengembangkan jiwa inovasi mahasiswa. Salah satunya yaitu Kontes Robot Cerdas Pemadam Api Berkaki Indonesia

Kontes Robot Cerdas Pemadam Api Indonesia divisi berkaki merupakan kontes robot dengan misi untuk memadamkan api pada salah satu ruang yang di batasi oleh dinding sebagai pembeda antara ruang yang tersusun dalam sebuah model rumah, dimana robot diharuskan untuk mampu beradaptasi agar dapat menyusuri arena serta beradaptasi terhadaap lingkungannya. Untuk tujuan tersebut maka sebuah *mobile robot* harus dilengkapi dengan sensor yang dapat mengambil data dan kemudian diolah oleh mikrokontroler. Salah satu sensor yang banyak digunakan adalah sensor ultrasonik untuk mendeteksi jarak.

Permasalahan utama robot KRPAI divisi berkaki adalah merancang dan menentukan pergerakan *mobile robot* agar robot dapat bergerak menyusuri arena dan menyelesaikan tugasnya. Oleh karena itu, dibutuhkan suatu sistem kontrol untuk menunjang cara tersebut yang dapat mengatasi kelemahan-kelemahan dalam pergerakan.

Didalam skripsi ini akan dirancang suatu Sistem kendali untuk robot berjenis hexapod (berkaki 6) dengan menggunakan metode Fuzzy-PID. Sistem tersebut dirancang untuk

mendapatkan respon yang cepat dan handal untuk menentukan parameter kendali PID dalam melakukan navigasi sehingga *mobile robot* otomatis diharapkan dapat dengan cepat menyelesaikan tugasnya.

Implementasi dari kendali fuzzy-PID ini sendiri berfungsi dalam menentukan parameter kontrol gain Kp, Ki, dan Kd, yang akan digunakan dengan menyeleksi jarak atau posisi robot ke dinding berdasarkan nilai sensor jarak yang terbaca yang kemudian di ubah ke nilai lingusitik menggunakan logika fuzzy, yang akan menetukan perubahan perilaku yang akan diberikan pada kontroler PID, yang mana nilai dari perubahan ini akan digunakan sebagai signal yang akan dibaca untuk merubah posisi sudut dari pergerakan robot.

Metode ini dipilih dikarenakan kendala yang sering dihadapi peniliti atau mahasiswa dalam merancang navigasi dan pengendalian pergerakan robot, dimana sebelumnya proses kalibrasi nilai Penguatan Proposional – Integrative-Derivative (PID) sering dengan melakukan tuning manual yaitu dengan metode uji coba pada nilai PID tertentu dengan menaikan nilai secara linear sampai di dapat nilai yang diinginkan, karena diubah secara manual maka untuk mendapatkan nilai yang diinginkan akan membutuhkan waktu yang lama, dan juga nilai yang didapat tersebut tidak mempertimbangkan pembedaan perlakukan untuk robot dengan jarak yang dekat, sedang, maupun jauh dari dinding, karena nilai yang dianggap sama untuk semua kondisi.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan dapat disusun rumusan masalah sebagai berikut :

1. Bagaimana mengimplementasikan sistem navigasi wall following dengan metode fuzzy-PID pada robot berkaki 6 atau *Hexapod Robot*?
2. Bagaimana membuat aturan aturan fuzzyfikasi dan defuzzyfikasi yang sesuai untuk kalibrasi nilai PID?
3. Bagaimana merancang dan membuat algoritma pergerakan dan navigasi pada robot berkaki 6?

1.3 Batasan Masalah

Dengan mengacu pada permasalahan yang telah dirumuskan, batasan masalah untuk skripsi ini adalah sebagai berikut :

1. *Mobile robot* yang dibuat memiliki 6 kaki dengan 3 derajat kebebasan (Degree of Freedom (DOF)) pada setiap kakinya.
2. Motor servo yang digunakan adalah tipe dynamixel AX-18A.
3. Model lintasan yang digunakan berupa dinding yang membentuk simpangan.
4. Sistem bersifat *close loop*, hanya bernavigasi dan memantau posisi robot
5. Sistem aturan logika fuzzy diperuntukan untuk menetukan nilai Kp, Ki dan Kd. Untuk proses selanjutnya diambil dari perhitungan PID untuk pergerakan.
6. Parameter masukan yang digunakan berasal dari sensor jarak ultrasonik PING)).
7. Penelitian hanya difokuskan pada penerapan algoritma fuzzy-PID dan implementasi pada program, dan tidak melakukan pembahasan pada perhitungan invers kinematic setiap kaki.

.

1.4 Tujuan

Tujuan dari penelitian ini adalah merancang dan membuat suatu algoritma tuning nilai parameter kontrolir PID yaitu Kp, Ki, dan Kd yang akan digunakan untuk perhitungan PID yang digunakan sebagai kendali robot berkaki enam. Dan agar dalam perkembangannya memudahkan dalam proses tuning nilai gain kontroler.

1.5 Manfaat

Adapun manfaat dari hasil skripsi ini adalah sebagai berikut :

1. Manfaat utama dari skripsi ini yaitu membantu sistem navigasi robot saat melakukan telusur dinding dengan baik dan handal pada sistem robot *hexapod*,
2. Hasil skripsi ini dapat dikembangkan kembali untuk aplikasi pada algoritma pada deteksi kondisi api pada KRPAI.

1.6 Sistematika Penulisan

Sistematika penulisan dalam skripsi ini sebagai berikut :

BAB I PENDAHULUAN

Memuat latar belakang, rumusan masalah, batasan masalah, tujuan, dan sistematika pembahasan.

BAB II TINJAUAN PUSTAKA

Membahas teori – teori yang mendukung dalam peencanaan dan pembuatan alat.

BAB III METODOLOGI PENELITIAN

Berisi tentang metode metode yang dipakai dalam melakukan perancangan, pengujian, dan analisis data.

BAB IV PENGUJIAN DAN ANALISIS

Memuat Aspek pengujian meliputi penjelasan tentang cara pengujian dan hasil penguijian, aspek analisis meliputi penilaian atau komentar terhadap hasil-hasil pengujian. Pengujian dan analisis ini terhadap alat yang telah direalisasikan berdasarkan masing masing blok dan sistem secara keseluruhan.

BAB V KESIMPULAN DAN SARAN

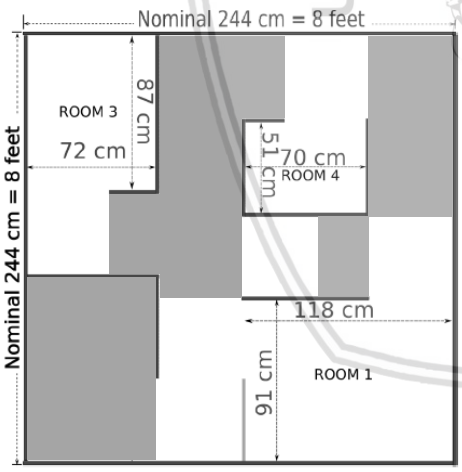
Memuat intisari hasil pengujian dan menjawab rumusan masalah serta memberikan rekomendasi untuk perbaikan kualitas penelitian di masa yang akan datang.

BAB II
TINJAUAN PUSTAKA

2.1 Kontes Robot Sepak Bola Indonesia

Kontes Robot Pemadam Api Indonesia (KRPAI) merupakan pertandingan robot tingkat nasional yang diadakan oleh Kementrian Riset Teknologi dan Pendidikan Tinggi (RISTEKDIKTI). Didalam pertandingan ini robot pada pertandingan ini merupakan robot berkaki. Untuk divisi ini sendiri memiliki tema *Kontes Robot Cerdas Pemadam Api Indonesia Berkaki*, yaitu robot bergerak dari posisi *home* menelusuri arena untuk memadamkan api kemudian kembali lagi ke posisi *home*.

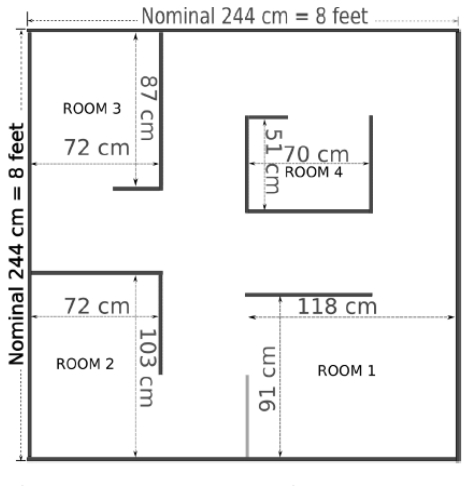
Robot yang digunakan pada Kontes Robot Cerdas Pemadam Api Indonesia memiliki panjang maksimum 31 cm, lebar maksimum 31 cm dan tinggi maksimum 27 cm. Pada Divisi Berkaki, arena lomba memiliki konfigurasi acak baik ruangan, posisi *home*, lokasi lilin, lokasi *furniture*. arena lomba pada KRPAI divisi beraki ditunjukkan dalam Gambar 2.1



Gambar 2. 1 Hasil Pengacakan Konfigurasi Arena Lomba KRPAI

Sumber : (RISTEKDIKTI, 2017)

Lapangan arena mensimulasikan interior sebuah rumah dengan 4 ruangan. Lapangan terbuat dari papan multipleks dengan ketebalan 1,8 s.d. 2 cm dan berukuran 244 cm x 244 cm x 30 cm. Lapangan terdiri dari 4 ruangan dengan posisi tetap namun dua diantaranya (ruang 1 dan 4) memiliki pintu yang dapat digeser posisinya. Ukuran tiap ruangan berbeda-beda. Ukuran arena lomba pada divisi berkaki ditunjukkan dalam Gambar 2.2.

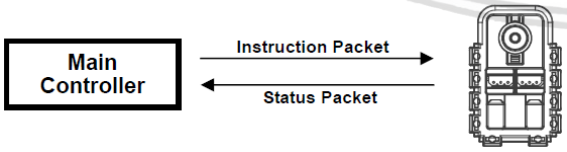


Gambar 2. 2 Dimensi arena lomba KRPAI divisi berkaki

Sumber: (RISTEKDIKTI, 2017).

2.2 *Servo Serial Dynamixel*

Dynamixel adalah aktuator robot cerdas modular yang menggabungkan *gear reducer*, motor DC presisi dan sirkuit kontrol, semua dalam satu paket. Meskipun ukurannya ringkas, *servo serial dynamixel* dapat menghasilkan torsi tinggi dan dibuat dari bahan berkualitas tinggi untuk memberikan kekuatan yang diperlukan dan ketahanan untuk menahan kekuatan eksternal yang besar. *servo serial dynamixel* ini juga memiliki kemampuan untuk mendeteksi dan bertindak pada kondisi internal seperti perubahan suhu atau tegangan suplai. Aktuator *Dynamixel* memiliki banyak keunggulan dibandingkan produk sejenis. Posisi dan kecepatan dapat dikendalikan secara serial dengan resolusi 1024 *step,* adapun gambaran umum sistem pengontrolan dynamixel ditunjukkan dalam Gambar 2.3 (Robotis, 2006).



Gambar 2. 3 Sistem pengontrolan servo serial dynamixel

Sumber : Robotis (2006,p.9).

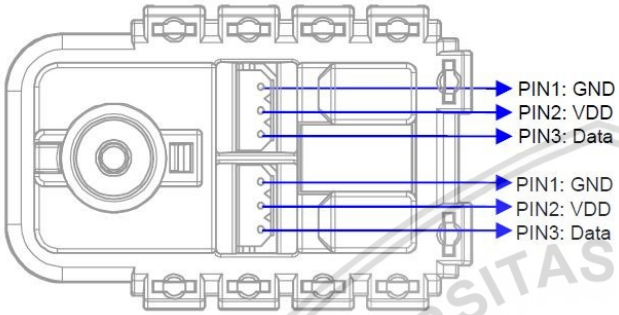
Kontroler utama berkomunikasi dengan unit *Dynamixel* dengan mengirim dan menerima paket data. Ada dua jenis paket; yang "*Instruction Packet*" (dikirim dari kontroler utama ke aktuator servo) dan "*Status Packet*" (dikirim dari Aktuator untuk mikrokontroler utama).

Struktur paket instruksi pada pengiriman data ke aktuator servo serial dapat dilihat pada Gambar 2.4, *pinout* dari *servo serial dynamixel* ditunjukkan dalam Gambar 2.5.

OXFF	0XFF	ID	LENGTH	ERROR	PARAMETER1	PARAMETER2	...	PARAMETER N	CHECK SUM

Gambar 2. 4. Paket instruksi pengiriman data pada *servo serial dynamixel*

Sumber : Robotis (2006,p.10)



Gambar 2. 5 Pinout servo serial dynamixel

Sumber : Robotis (2006,p.6)

Kedua paket data pertama yaitu 0xFF menandakan awal dari paket yang akan datang. ID adalah nomor identitas yang unik dari aktuator yang berjumlah 254 ID dan memiliki *range* 0x00 - 0xFD. LENGTH merupakan panjang paket data dimana memiliki nilai jumlah parameter (N) + 2. INSTRUCTION merupakan instruksi yang harus dilakukan oleh aktuator. PARAMETER adalah informasi taambahan yang diberikan pada instruksi tertentu. CHECKSUM merupakan metode komputasi untuk pengecekan data. Persamaan untuk mencari nilai CHECKSUM dapat dilihat pada persamaan (2-1). Jika nilai yang terhitung lebih besar dari 255 maka bit terendah dianggap sebagai nilai CHECKSUM. Simbol "~" menandakan logika not.

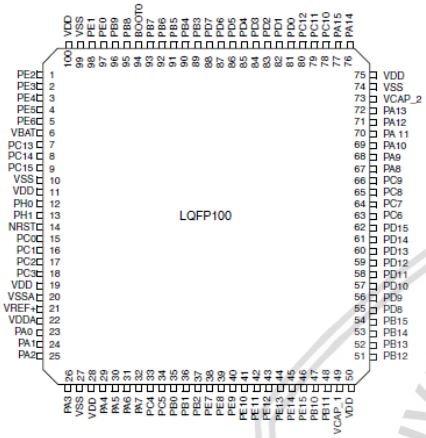
$$\boldsymbol{CHECKSUM} = (\boldsymbol{ID} + \boldsymbol{LENGTH} + \boldsymbol{INSTRUCTION} + \boldsymbol{PARAMETER\ 1} + \boldsymbol{PARAMETER\ N}) \quad (2.1)$$

2.3 Mikrokontroler STM32F4

Mikrokontroller merupakan sebuah permrosess data yang di dalamnya sudah terdapat mikroprosessor, memori dan perangkat-perangkat yang lain yang menjadi satu kesatuan dalam satu chip.

Mikrokontroller STM32F4 adalah mikrokontroler CMOS 32-bit dengan arsitektur RISC buatan STMicroelectronics keluarga ARM Cortex-M4 dengan *clock* sampai dengan 168 MHz, mampu mengeksekusi perintah sampai dengan 210 MIPS (*Million Instruction per Second*).

Mikrokontroller ini memiliki 1 MByte Flash PEROM (Flash Programmable and Eraseble Read Only Memory), 192 Kbyte SRAM, 100 pin, 5 buah port I/O yang mana setiap pin dalam masing masing port dapat diprogram tersendiri, memiliki dua belas buah *timer/counter* 16 bit dan dua buah *timer/counter* 32 bit, memiliki antarmuka kamera secara paralel 8-14 bit dengan kecepatan sampai 54 MByte/s, memiliki 24 *channel* ADC (*Analog Digital Converter*). Konfigurasi pin STM32F4 ditunjukan dalam gambar 2.6.



Gambar 2. 6 Konfigurasi pin STM32F4

Sumber: (STMicroelectronics, 2017)

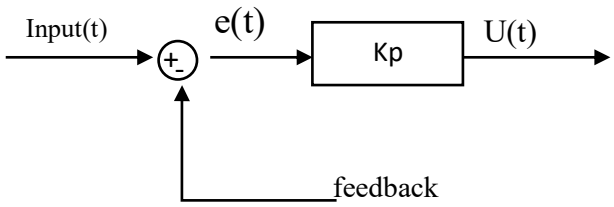
2.4 Kontroler

Kontroler dalam sebuah sistem kontrol mempunyai kontribusi yang besar terhadapa perilaku sistem. Hal itu disebabkan oleh tidak dapat dirubahnya komponen penyusun sistem tersebut. Artinya, karakteristik *plant* harus diterima sebagaimana adanya, perubahan perilaku sistem hanya dapat dilakukan melalui penambahan kontroler.

Salah satu fungsi kontroler adalah mengurangi sinyal *error*, sinyal *error* adalah perbedaan nilai *setpoint* dengan niali *output plant*. Adalah nilai aktual yang terukur pada *output plant.* Semakin kecil nilai sinyal error maka kinerja sistem kontrol dinilai semakin baik. Prinsip kerja kontroler adalah membandingkan nilai output dengan nilai *setpoint,* menentukan nilai kesalahan dan akhirnya menghasilkan sinyal kontrol untuk meminimalkan kesalahan (Ogata, 1997).

2.4.1 Kontroler Proporsional

Kontroler proporsional memiliki *output* yang besarnya sebanding dengan besarnya sinyal error. *Output* kontroler merupakan perkalian antara penguatan proporsional dengan sinyal error. diagram blok kontroler proporsional ditunjukan oleh gambar 2.7.



Gambar 2. 7 Diagram Blok Kontroler Proporsional

Dimana :

Kp = adalah gain proporsional

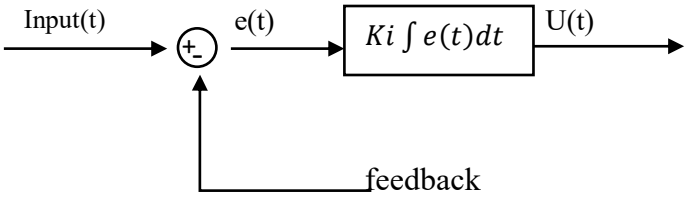
e(t) = sinyal error

u(t) = ouput keluaran

Penambahan Kp akaan mempercepat kecepatan respon *transient* dan mengurangi *error steady state*.

2.4.2 Kontroler Integral

Kontroler Integral memiliki karakteristik seperti sebuah operasi integral, output kontroler dipengaruhi oleh perubahan yang sebanding dengan perubahan nilai sinyal error. Output kontroler merupakan penjumlahan terus menerus dari perubahan sinyal *error*. diagram blok kontroler integral ditunjukan oleh gambar 2.8.



Gambar 2. 8 Diagram Blok Kontroler Intergal

Dimana :

Ki = adalah gain integral

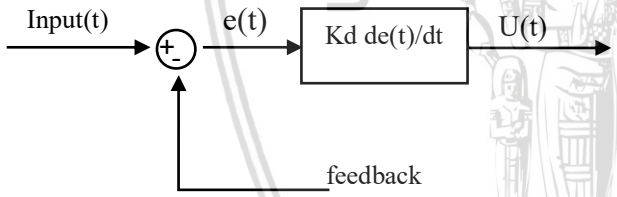
E(t) = sinyal error

U(t) = output kontroler

Aksi kontrol *integral* digunakan untuk menghilangkan sinyal error dalam *steady state.* Namun pemilihan Ki yang tidak tepat dapat menyebabkan respon transien yang tinggi, sehingga dapat menyebabkan ketidakstabilan sistem. Pemilihan Ki yang sangat tinggi justru dapat menyebabkan *output* berosilasi karena menambah *orde system*.

2.4.3 Kontroler Difrensial

Kontroler *difrensial* memiliki sifat seperti suatu operasi turunan. Perubahan yang mendadak pada masukan kontroler mengakibatkan perubahan yang sangat besasr dan cepat, kontroler ini tidak akan menghasilkan *output* saat sinyal *error* konstan sehingga tidak akan mempengaruhi keadaan mantap. diagram blok kontroler *difrensial* ditunjukan oleh gambar 2.9.



Gambar 2. 9 Diagram Blok Kontroler Difrensial

Dimana :

Kd = adalah gain *derivative*

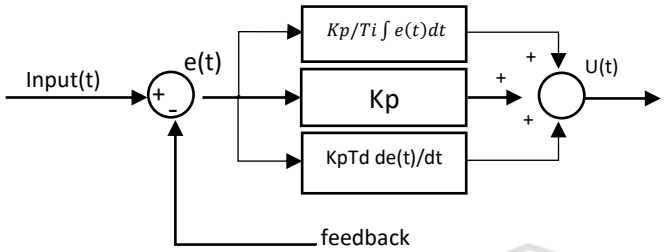
E(t) = sinyal *error*

U(t) = *output* kontroler

Kontroler ini digunakan untuk memperbaiki atau mempercepat respon transient. Kontrol diferensial hanya berubah saat ada perubahan *error* sehingga saat *error* statis kontrol ini tidak bereaksi, hal ini pula yang menyebabkan kontroler diferensial tidak dapat dipakai sendiri.

2.4.4 Kontroler PID

Kontroler PID merupakan gabungan aksi kontrol proporsional, integral dan difrensial yang terlihat dalam gambar 2.10 mempunyai keunggulan dapat saling menutupi kekurangan dan kelebihan dari masing-masing kontroler.



Gambar 2. 10 Diagram Blok Kontroler Proporsional integral dan difrensial

Dimana :

Kp = adalah penguatan *proportional*

Ti = adalah waktu integral

Td = adalah waktu *derivative*

e(t) = sinyal *error*

u(t) = output kontroler

Ti merupakan waktu yang digunakan untuk mengatur aksi kontrol internal sedangkan Td adalah waktu internal dengan laju aksi kontroler proporsional. Sinyal keluaran PID didapat dari persamaan (2.2).

$$u(t) = Kp(e(t) + \frac{1}{Ti}\int_0^t e(t)dt + Td\,\frac{de(t)}{dt} \qquad (2.2)$$

2.5 Logika Fuzzy

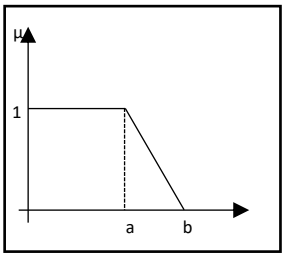
2.5.1 Logika Fuzzy

Suatu himpunan *fuzzy* A dalam semesta pembicaraan U dinyatakan dengan fungsi keanggotaan, uA yang harganya berada dalam interval [0,1] (Kuswadi,2000:27). Secara matematika hal ini dinyatakan pada persamaan (2.3).

$$\mu A : U \rightarrow [0,1] \qquad (2.3)$$

Adapun beberapa pendekatan dalam perancangan aturan fuzzy dirancang untuk kondisi yang terdiri dari reversegrade, triangle, grade

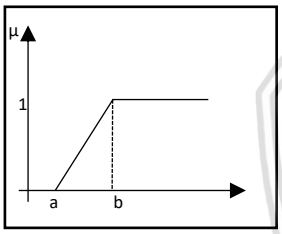
a) Fungsi keanggotaan *reverse grade* ditunjukan dalam gambar 2.11 dan persamaan matematis ditunjukan pada persamaan 2.4.



$$\mu[x] = \begin{cases} 0, x \geq b \\ \frac{x-a}{b-a}, a < x < b \\ 1, \ x \leq a \end{cases} \quad (2.4)$$

Gambar 2. 11 Grafik fungsi keanggotaan reverse grade

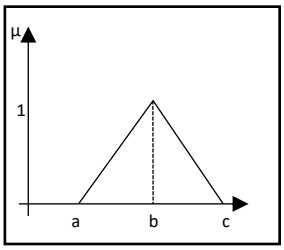
b) Fungsi keanggotaan *grade* ditunjukan dalam gambar 2.12 dan persamaan matematis ditunjukan pada persamaan 2.5.



$$\mu[x] = \begin{cases} 0, x \leq b \\ \frac{-x+a}{b-a}, a < x < b \\ 1, \ x \geq b \end{cases} \quad (2.5)$$

Gambar 2. 12 Grafik fungsi keanggotaan grade

c) Fungsi keanggotaan *triangle* ditunjukan pada gambar 2.13 dan persamaan matematis ditunjukan pada persamaan 2.6.



$$\mu[x] = \begin{cases} 0, x \geq c \ atau \ x \leq a \\ \frac{x-a}{b-a}, a < x < b \\ \frac{-x+c}{c-b}, b < x < c \\ 1, \ x = b \end{cases} \quad (2.6)$$

Gambar 2. 13 Grafik fungsi keanggotaan triangle

2.5.2 Fuzzyfikasi

Fuzzyfikasi merupakan proses dimana mengolah masukan berupa nilai crisp menjadi himpunan fuzzy, diperlukan metode proses ini dikarenakan apa yang kita yakini sebagai bentuk crisp dan deterministik sebenarnya tidak deterministrik sama sekali.

Sebagai contoh, jika ada proses atau pengukuran kebisingan, kita mungkin ingin untuk menjelaskan hal ini dengan menciptakan *fuzzy set* untuk takaran terukur dari pada dengan asumsi mereka akurat yang diukur.

Ada beberapa strategi yang dapat dilakukan dalam fuzzyfikasi diantaranya

1. Fuzzy Singletone
2. Fuzzy Number
3. Hybrid fuzzy/Bilangan Acak

Pedoman memilih fungsi keanggotaan untuk proses fuzzyfikasi dapat menggunakan cara sebagai berikut :

1. Himpunan fuzzy dengan distribusi simetris
2. Gunakan himpunan fuzzy dengan jumlah ganjil, berkaitan erat dengan jumlah kaidah (*rules*).
3. Mengatur himpunan *fuzzy* agar saling menumpuk.
4. Menggunakan fungsi keanggotaan bentuk segitiga, atau trapesium.

2.5.3 Rule Base Fuzzy

Fuzzy Rule adalah bagian yang menggambarkan dinamika suatu sistem terhadap masukan yang dikarakteristikan oleh sekumpulan variabel – variabel linguistik dan berbasis pengetahuan seorang operator ahli. Penyataan tersebut umumnya dinyatakan oleh suatu pernyataan bersyarat.

Dalam pengendali berbasis *fuzzy,* aturan pengendalian *fuzzy* berbentuk aturan “IF_THEN”. Untuk sebuah sistem *Multi Input Single Output (MISO)* basis aturan pengendalian *fuzzy* berbentuk seperti berikut ini,

Rule 1 IF X is A1 AND Y is B1 THEN Z is C1

Rule 2 IF X is A2 AND Y is B2 THEN Z is C2

....

Rule IF X is An AND Y is Bn THEN Z is Cn

Dengan X,Y,Z merupakan variabel linguistik, dimana X dan Y merupakan variabel masukan, dan Z merupakan variabel keluaran sistem. An, Bn, dan Cn merupakan nilai linguistik dari X,Y,Z (Lee, 1990).

2.5.4 Defuzzyfikasi

Defuzzifikasi adalah proses kebalikan fuzzyfikasi. Hal ini diperlukan untuk mengolah hasil berupa nilai himpunan fuzzy untuk diubah kembali kedalam bentuk *crisp.* Defuzzyfikasi adalah proses untuk mendapatkan nilai numerik dari data *fuzzy* yang dihasilkan dari proses inferensi. Proses defuzzifikasi dinyatakan pada persamaan 2.7.

$$yo = defuzzifier\ (y) \tag{2.7}$$

Dengan :

Y : aksi kontrol

Y0 : aksi kontrol crisp

Defuzzyfier : operator defuzzifikasi

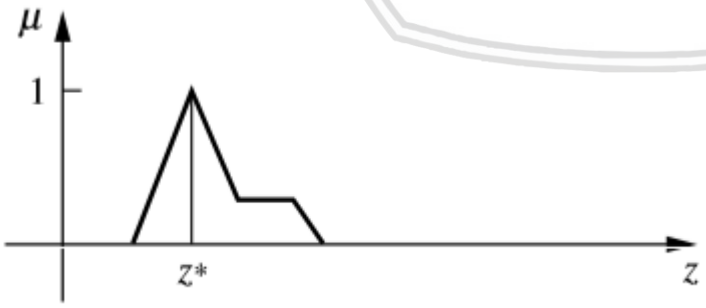
Ada beberapa cara yang bisa dilakukan dalam proses defuzzyfikasi antara lain

2.5.5 Prinsip Min-Max Membership

Max membership bisa dikatan sebagai metode ketinggian. Dalam metode ini terbatas pada keluaran yang memuncak. Max membership dapat dinyatakan dalam persamaan 2.8.

$$U_c{\sim}(z*) \geq \mu C \sim (z)\ for\ all\ z\ \in Z \tag{2.8}$$

Dengan z* adalah nilai defuzzyfikasi yang dapat dilihat pada gambar 2.14 dibawah ini.



Gambar 2. 14 Max Membership defuzzyfikasi

2.5.6 Metode Centroid

Metode ini juga disebut *centre of gravity.* Metode ini yang paling umum dan menarik untuk digunakan. (Sugeno,1985 : Lee.1990);

Diberikan ekspresi aljabar pada persamaan 2.9.

$$z = \frac{\int \mu c\,(z).\ z\,dz}{\int \mu c\,(z)\,dz} \tag{2.9}$$

Metode ini sangat sering digunakan karena cara pengunaanya sangat efisien. Metode ini digunakan untuk output membership yang simetris. Persamaan matematisnya ditunjukan pada persamaan 2.10.

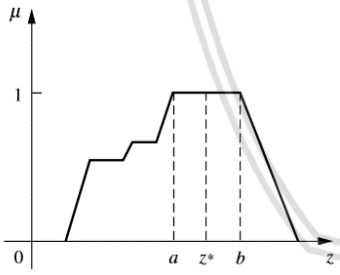
$$z = \frac{\sum \mu c\,(z).\ z}{\sum \mu c\,(z)} \tag{2.10}$$

2.5.7 Mean Max Membership

Dikenal juga sebagai middle of maxima, metode hampir serupa dengan Max Membership, kecuali dari metode Max membership bisa juga no-unik. Diberikan persamaan 2.11

$$z = \frac{a+b}{2} \tag{2.11}$$

Dimana nilai a dan b diberikan pada gambar 2.15.



Gambar 2. 15. Membership metode defuzzyfikasi

2.6 Sensor Ultrasonik PING)))

Sensor ultrasonik yang dipakai dalam skripsi ini adalah sensor PING))) produksi dari parallax yang berupa modul siap pakai lengkapdengan pengirim dan penerima. Sensor ultrasonik digunakan untuk mendeteksi jarak dengan frekuensi 40 kHz. Sinyal data sensor PING))) ini akan masuk ke kaki mikrokontroler. Bentuk sensor ultrasonik PING)) ditunjukan dalam gambar 2.16.

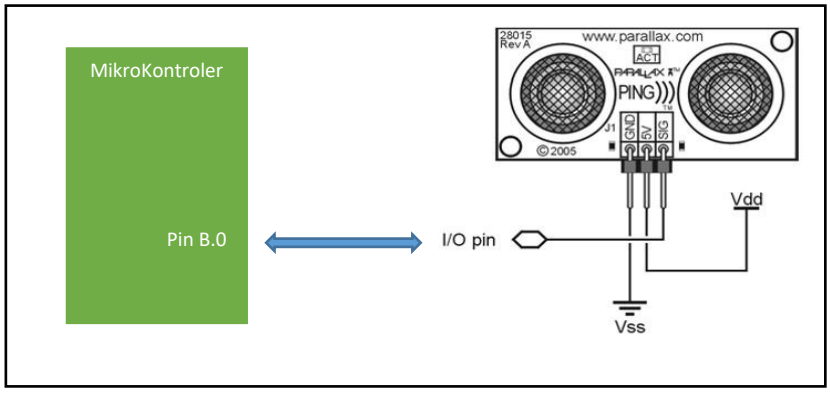
Gambar 2. 16 Sensor Ultrasonik PING)))
Sumber : (Parallax, 2017)

Modul PING))) mengukur jarak objek dengan cara memancarkan gelombang ultrasonik (40kHz) selama tBurst (200 us) kemudian menunggu pantulannya. Modul PING))) memancarkan gelombang ultrasonik sesuai dengan masukan kontrol dari pin SIG. Gelombang ultrasonik ini melalui udara dengan kecepatan kurang lebih 344 meter per detik, mengenai objek dan memantul kembali ke modul PING))). Modul PING))) akan mengeluarkan pulsa *high* pada pin SIG setelah memancarkan gelombang ultrasonik. Setelah pantulan gelombang terdeteksi, modul PING))) akan membuat pin SIG *low.* Lebar pulsa *high* (tIN) ini sesuai dengan lama waktu tempuh gelombang ultrasonik untuk 2 kali jarak objek, sehingga jarak objek yang terukur didefinisikan pada persamaan 2.12.

$$S = \frac{(t_{IN\,(s)}\, x(V_{(m/s)}\,)}{2} \qquad (2.12)$$

Dimana :
S : Jarak baca (m)
tIN : Waktu pembacaan (s)
V : Kecepatan Rambat udara (344 m/s)

Yang dapat terlihat pada gambar 2.15 sensor PING))) ini memiliki 3 buah kaki atau pin yakni pin VCC, pin GND, dan pin SIG. Pin yang digunakan untuk mengirim data adalah pin SIG, komunikasi mikrokontroler dengan PING ditunjukan dalam gambar 2.17.



Gambar 2. 17 Komunikasi mikrokontroler dengan PING)))

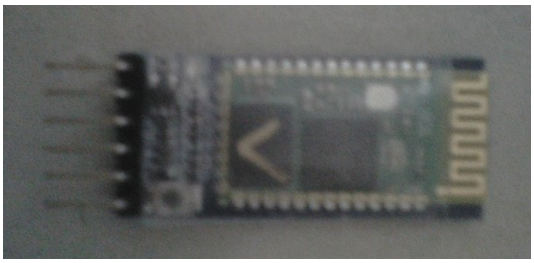
Sumber : (Parallax, 2017)

Sensor PING))) ini memiliki beberapa kelebihan diantaranya bahwa sensor ini memiliki tingkat akurat yang tinggi, sensor ini mampu mendeteksi benda walaupun benda tersebut, berukuran kecil.

2.7 Modul *Bluetooth* HC-05

Bluetooth Module HC-05 merupakan *module* komunikasi nirkabel pada frekuensi 2.4GHz dengan pilihan koneksi bisa sebagai *slave*, ataupun sebagai *master*. Dapat digunakan dengan mikrokontroler untuk membuat aplikasi *wireless* bentuk fisik modul *bluetooth* HC-05 ditunjukkan dalam Gambar 2.18. Adapun spesifikasinya adalah sebagai berikut:

- Frekuensi : 2.4GHz,
- Catu Daya : 5 V atau 3,3 V DC,
- Dimensi : 3.57cm x 1.52cm.



Gambar 2. 18 Bentuk fisik modul bluetooth HC-05

Sumber : Dokumentasi Penulis

BAB III
METODE PENELITIAN

Penyusunan skripsi ini didasarkan pada masalah yang bersifat aplikatif, yaitu perencanaan dan perealisasian alat agar dapat bekerja sesuai dengan yang direncanakan dengan mengacu pada rumusan masalah. Oleh karena itu dibutuhkan suatu metode penelitian agar perencanaan dan perealisasian alat dapat dilakukan. Langkah – langkah yang perlu dilakukan untuk merealisasikan alat yang dirancang adalah penentuan spesifikasi alat, perancangan dan perealisasian alat, dan pengujian alat.

3.1 Penentuan Spesifikasi Alat

Spesifikasi alat secara global ditetapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya. Spesifikasi alat yang direncanakan adalah sebagai berikut

2) Robot berbahan dasar alumunium dan mika *acrylic.*
3) Robot yang dibuat memiliki 6 kaki dengan 3 derajat kebebasan *(Degree of Freedom* (DOF)) pada setiap kakinya.
4) Dimensi maksimum robot adalah 31 cm x 31 cm x 27 cm sesuai Peraturan Kontes Robot Pemadam Api Indonesia 2017.
5) Mikrokontroler STM32F4 sebagai pengendali utama *mobile* robot.
6) Mikrokontroler ATMEL ATMEGA – 8A sebagai pengendali tingkat kedua.
7) Sensor yang digunakan adalah sensor Ultrasonik Parallax PING))) sebanyak 5 buah
8) Catu Daya yang digunakan adalah baterai Lithium Polymer Baterai 12 V DC 2200 mah.

3.2 Penentuan Spesifikasi Desain

Spesifikasi pengendalian ditetapkan terlebih dahulu sebagai acuan dalam perancangan sistem kendali robot, yaitu :

1. Set Point merupakan jarak robot ditengah lajur, nilai jarak merupakan nilai pembacaan sensor.
2. Waktu yang dibutuhkan untuk steady tidak lebih besar dari 200 ms.
3. Error (error rata rata) tidak lebih besar dari 5%

3.3 Perancangan Dan Perealisasian Alat

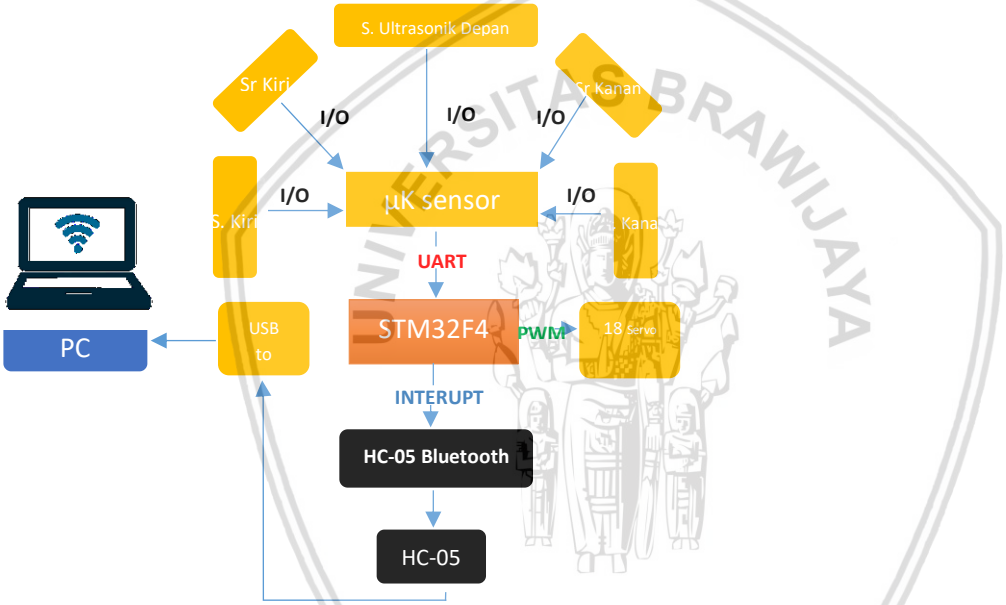
Perancangan dan pembuatan alat dalam skripsi ini dibagai menjadi dua bagian, yaitu pembuatan *hardware* dan *software*.

3.3.1 Perancangan dan Pembuatan Perangkat Keras (*Hardware*)

Secara garis besar perancangan dan pembuatan perangkat keras dapat dibagai menjadi beberapa bagian yaitu perancangan diagram blok sistem keseluruhan dan perancangan mekanika robot.

3.3.1.1 Perancangan Diagram Blok Sistem Keseluruhan

Diagram blok keseluruhan sistem yang dirancang ditunjukkan dalam Gambar 3.1.

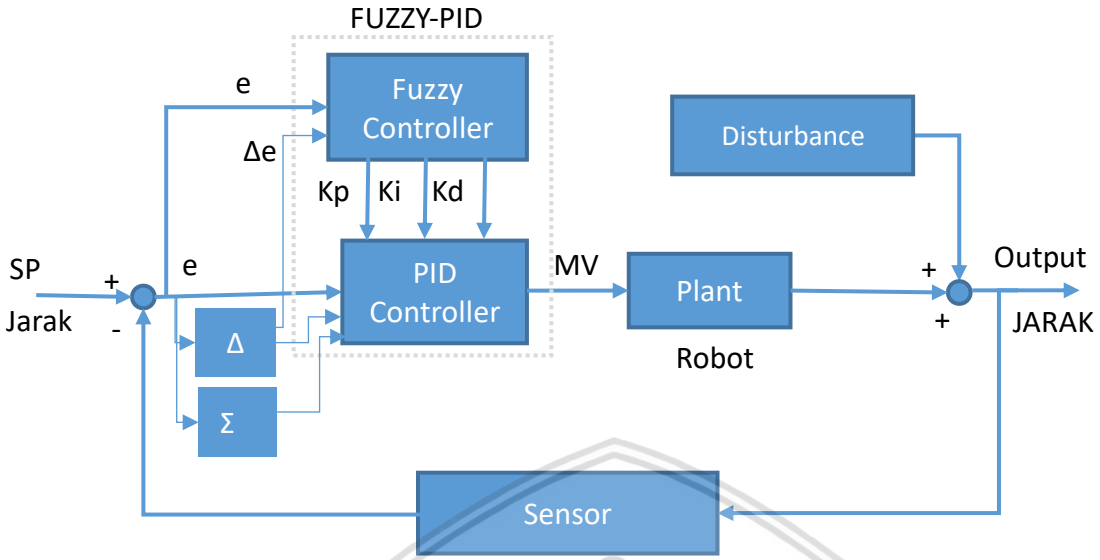


Gambar 3. 1 Diagram blok sistem keseluruhan

Fungsi masing-masing bagian dalam diagram blok ini adalah sebagai berikut :

1. STM32F4 digunakan sebagai mikrokontroler utama dari sistem untuk mengolah hasil pengolahan keseluruhan sistem
2. *ATMEGA8* sebagai mikrokontroler sensor jarak ultrasonik Parallax PING)), untuk mengolah hasil pembacaan jarak untuk dikirm ke mikrokontroler utama
3. USB to TTL berfungsi sebagai konverter tegangan,
4. 18 servo Dynamixel AX-18A sebagai aktuator penggerak sistem,
5. Komputer digunakan untuk pemantauan proses pembacaan yang dilakukan oleh robot,
6. Modul *bluetooth* HC-05 berfungsi untuk menerima data serial.

Diagram blok kendali sistem yang dirancang ditunjukkan dalam Gambar 3.2.



Gambar 3. 2 Diagram blok kendali sistem

Fungsi masing-masing bagian dalam diagram blok ini adalah sebagai berikut :

1. *Setpoint* berupa nilai jarak dari sensor
2. Masukan Kontroler Fuzzy berupa nilai Error dan derror, dan masukan PID berupa nilai error, derror, serror.
3. Keluaran Fuzzy berupa gain kontrol Kp, Ki, dan Kd, Keluaran PID adalah variabel pemanipulasi sudut.
4. Gangguan berupa perubahan jarak saat robot bergerak, dan pemberian objek pada dinding
5. Plant berupa robot atau kombinasi aktuator robot yaitu Motor Servo Dynamixel AX18A yang pada pergerakannya telah disusun oleh *gait* gerak yang didapat dari persamaan invers kinematik untuk robot berkaki enam.
6. Sensor berupa sensor jarak ultrasonik Parallax PING)).

3.3.1.2 Perancangan Mekanika Robot

Berdasarkan peraturan Kontes Robot Pemadam Api Indonesia (KRPAI) Divisi Berkaki tahun 2017, batasan dimensi robot baik saat posisi berhenti maupun saat posisi berjalan adalah,panjang = 31 cm, lebar = 31 cm, dan tinggi = 27 cm. Dengan adanya peraturan tersebut, maka desain robot berkaki enam ini harus dirancang agar tidak melebihi batas ukuran yang telah ditetapkan. Gambar desain mekanik robot berkaki enam ditunjukkan dalam Gambar 3.3.

Gambar 3. 3 Perspekstif robot secara keseluruhan

3.4 Perancangan Rangkaian Antarmuka Mikrokontroler Utama

Pada perancangan perangkat keras robot ini menggunakan modul mikrokontroller ARM sebagai pengolah utama untuk pemrosesan algoritma dan data sensor. Pada perancangan ini pin-pin yang digunakan adalah :

PIN A.2= Pin USART TX Dynamixel
PIN A.3= Pin USART RX Dynamixel
PIN C.1= Pin USART LOGIC Dynamixel
PIN B.0= Pin kaki Depan Kiri Coxa
PIN B.1= Pin kaki Depan Kiri femur
PIN B.2= Pin kaki Depan Kiri Tibia
PIN B.4= Pin kaki Tengah Kiri Coxa
PIN B.5= Pin kaki Tengah Kiri femur
PIN B.7= Pin kaki Tengah Kiri Tibia
PIN B.12= Pin kaki Belakang Kiri Coxa
PIN B.13= Pin kaki Belakang Kiri femur
PIN B.14= Pin kaki belakang Kiri Tibia
PIN D.0= Pin kaki Depan Kanan Coxa
PIN D.1= Pin kaki Depan Kanan femur
PIN D.3= Pin kaki Depan Kanan Tibia
PIN E.4= Pin kaki Tengah Kanan Coxa
PIN E.5= Pin kaki Tengah Kanan femur
PIN E.6= Pin kaki Tengah Kanan Tibia
PIN E.7= Pin kaki Belakang Kanan Coxa
PIN E.8= Pin kaki Belakang Kanan femur
PIN E.9= Pin kaki belakang Kanan Tibia

Pin ATMEGA 8

PIN B.1= Pin PING KANAN
PIN B.0= Pin PING Serong KANAN
PIN C.5= Pin PING Depan
PIN C.4= Pin PING Serong Kiri
PIN C.3= Pin PING Kiri

Mikrokontroler ARM merupakan kontroler utama yang nantinya akan memproses data dari ATMEGA8 sebagaoi slave mikrokontroler yang akan menerima data dari sensor. Mikrokontroler ARM juga berfungsi sebagai kontroler utama untuk mengatur pergerakan aktuator.

3.5 Perancangan Kontroler PID

Kontroler PID dibentuk dari gabungan kontroler *proporsional integral dan difrensial,* yang mana persamaan kontroler ini secara umum dapat ditunjukan pada persamaan 3.1

$$u(t) = Kp(e(t) + \frac{1}{Ti}\int_0^t e(t)dt + Td\ \frac{de(t)}{dt}\ ) \quad (3.1)$$

Dimana u(t) adalah signal kontrol dari PID, Kp adalah gain proporsional, Ti adalah waktu integral, dan Td waktu difrensial, dimana untuk e(t) didefinisikan sebagai selisih nilai *setpoint* dan pembacaan sensor jarak yang digunakan. Integral error (sigma error) didefinisikan sebagai jumlahan error sekarang dan sebelumnya sedang kan difrensial error (delta error) didefinisikan sebagai selisih error sekarang dan sebelumnya. Jika bentuk persamaan umum tersebut diubah dalam bentuk laplace akan terbentuk persamaan 3.2

$$u(s) = Kp(1 + \frac{1}{Tis} + Tds)E(s) \quad (3.2)$$

$Ki = Kp/Ti; Kd = Kp * Td;$ α = Ti/Td;

$Kp = Kp'(Kpmax - Kpmin) + Kpmin; Kd = Kd'(Kdmax - Kdmin) + Kdmin;$

Persamaan tersebut belum dapat dimasukan ke mikrokontroler maka dari itu persamaan kontinyu 3.2 diubah ke bentuk diskrit dengan Transformasi Z. Yang mana untuk transformasi ini dibutuhkan waktu sapmling (Ts) dengan *Bilinear Transform* nilai notasi s setara dengan persamaan 3.3

$$s = \frac{2}{Ts}(\frac{1-z^{-1}}{1+z^{-1}}) \quad (3.3)$$

Maka persamaan 3.2 setara dengan persamaan 3.4

$$u(z) = (Kp + \frac{KpTs(1+z^{-1})}{2Ti(1-z^{-1})} + KpTd\frac{2}{Ts}(\frac{1-z^{-1}}{1+z^{-1}})\ )E(z) \quad (3.4)$$

Berikutnya dibutuhkan modifikasi persamaan agar didapat bentuk yang lebih sederhana kedua ruas dikali dengan pembagi masing masing parameter. Seperti terlihat pada persamaan 3.4

$$u(z)(1-z^{-2}) = (Kp(1-z^{-2}) + \frac{KpTs(1+2z^{-1}+z^{-2})}{2Ti} + KpTd\frac{2}{Ts}(1-2z^{-1}+z^{-2})\ )E(z) \quad (3.5)$$

Kemudian disusun kembali untuk mendapatkan persamaan kontroler seperti terlihat pada persamaan 3.6.

$$u(z) = u(z)(z^{-2}) + \left(Kp(E(z) - E(z)(z^{-2})\right) + \frac{KpTs(E(z)+2E(z)(z^{-1})+E(z)(z^{-2}))}{2Ti} + \quad (3.6)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots KpTd\frac{2}{Ts}(E(z) - 2E(z)(z^{-1}) + E(z)(z^{-2})\ )$$

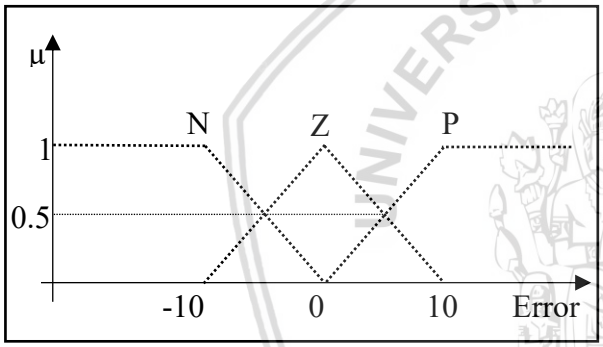
Kemudian persamaan tersebut diubah ke persamaan beda sehingga di dapat persamaan 3.7.

$$u(z) = u(k-2) + (Kp(e(k) - e(k-2)) + \frac{KpTs(E(k)+2E(k-1)+E(k-2))}{2Ti} + \quad (3.7)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots KpTd\frac{2}{Ts}(E(k) - 2E(k-1) + E(k-2))$$

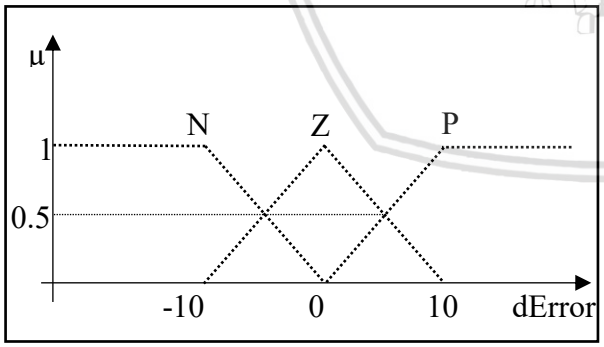
Dimana (k-n) adalah kondisi sebelumnya, yang kemudian bentuk ini dapat diterapkan pada pemerograman mikrokontroler.

3.6 Perancangan Aturan Fuzzy

Perancangan aturan fuzzyfikasi dengan metode fuzzy sugeno merupakan pendekatan untuk membangkitkan sinyal manipulasi yang akan digunakan untuk mendapatkan parameter kontrol, dalam perancangan algoritma fuzzy dirancang dengan masukan berupa *error* dan turunan *error,* dimana error pada perancangan ini merupakan selisih antara nilai *setpoint* dan pembacaan sensor jarak, error adalah parameter untuk menentukan posisi robot terhadap dinding, kemudian turunan error digunakan untuk mengetahui seberapa besar perubahan yang terjadi antara error sekarang dan error sebelumnya, yang mana untuk menentukan nilai parameter Kp, Ki, dan Kd, digunakan pendekatan Mean-Max atau nilai tertinggi antara keduanya, sedangkan defuzzyfikasi menggunakan *weighted average.* Dimana untuk fungsi keanggotaan dari error ditunjukan pada gambar 3.4 dan turunan error ditunjukan pada gambar 3.5.

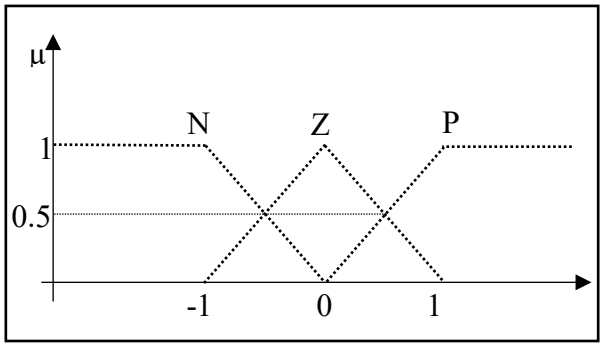


Gambar 3. 4 fungsi keanggotaan Error

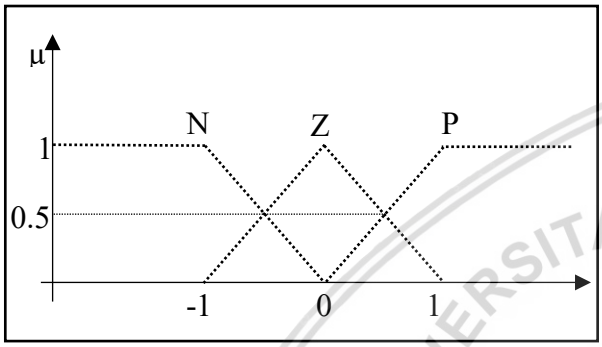


Gambar 3. 5 fungsi keanggotaan dError

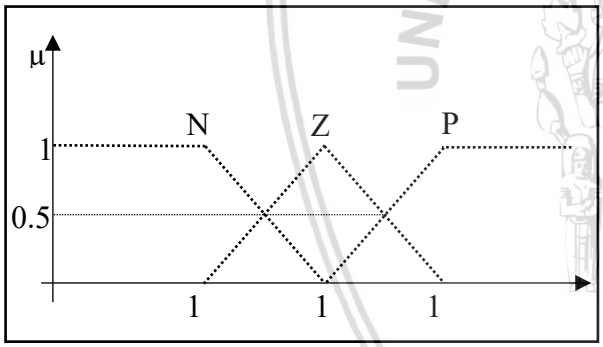
Untuk fungsi keanggotaan dari parameter Kp, Ki dan Kd, dapat diperoleh dengan persamaan PID, dimana pada penentuannya dibutuhkan nilai Ti dan Td, dan Kp, yang fungsi keanggotaannya ditunjukan oleh gambar 3.6 untuk fungsi keanggotaan Kp, fungsi keanggotaan Ti ditunjukan oleh gambar 3.7 dan fungsi keanggotaan Td ditunjukan pada gambar 3.8.



Gambar 3. 6 fungsi keanggotaan Kp



Gambar 3. 7 fungsi keanggotaan Kd



Gambar 3. 8 fungsi keanggotaan α

Setelah diketahui fungsi keanggotaan fuzzy dari parameter kontroler PID, agar didapatkan keluaran dari nilai error dan turunan error, dibutuhkan hubungan yang ditentukan dalam fuzzy rule.

Hubungan error dan turunan error dijabarkan sebagai berikut :

- Saat e relatif besar memperbesar nilai Kp dan memperkecil nilai Kd, dimana Ki juga relatif kecil
- Saat e berada disekitar nilai set (nilai batas) maka parameter Kp diperkecil dan Ki maupun Kd tetap.
- Saat e nilainya sangan kecil maka Kp dan Ki, dinaikan. Saat turunan error juga kecil maka Kd diperbesar.

Yang mana aturan *fuzzy rule* tersebut tergambar pada tabel 3.1 menunjukan *fuzzy rule Kp,* tabel 3.2 menunjukan *fuzzy rule Kd,* tabel 3.3 menunjukan *fuzzy rule α,*

Tabel 3. 1 fuzzy Rule Kp

Error \ dError	N	Z	P
N	N	N	Z
Z	N	Z	P
P	Z	P	P

Tabel 3. 2 fuzzy Rule Kd

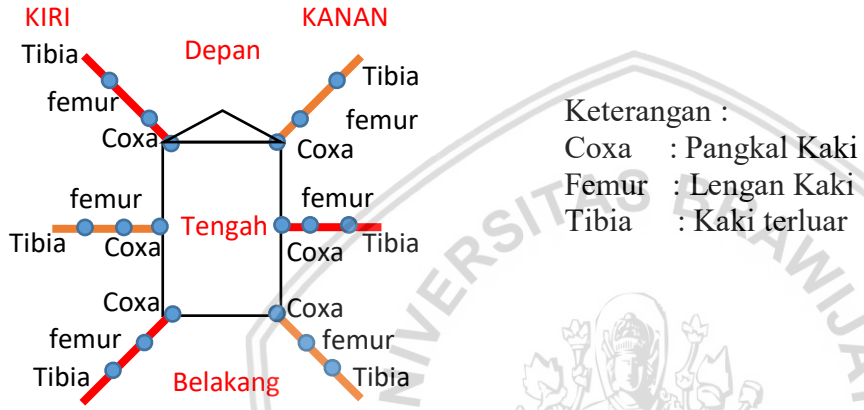
Error \ dError	N	Z	P
N	N	N	Z
Z	N	Z	P
P	Z	P	P

Tabel 3. 3 fuzzy Rule α

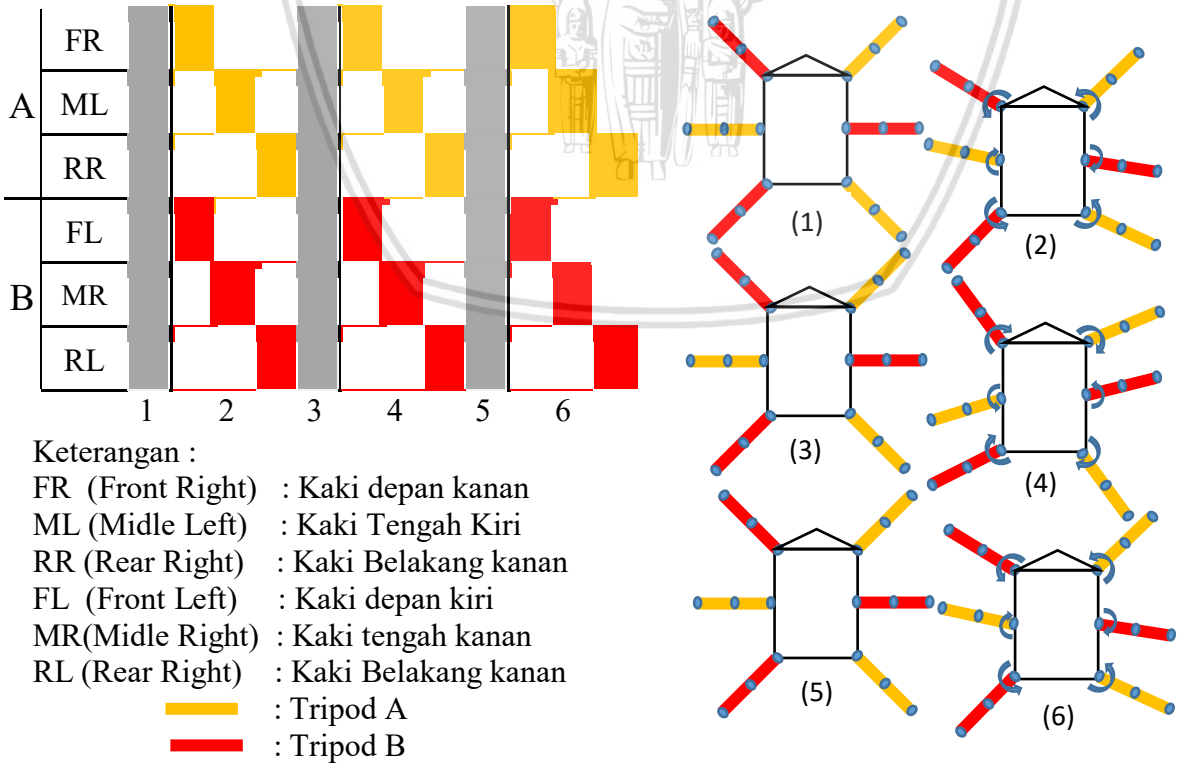
Error \ dError	N	Z	P
N	N	N	Z
Z	N	Z	P
P	Z	P	P

3.7 Perancangan Pola Langkah

Pola langkah merupakan model gait gerak robot untuk pergerakan maju, dan belok kiri, maupun kanan didapat dari nilai PID, yang kemudian di konversi sebagai nilai sinyal, untuk memanimpulasi pergerakan sudut kaki. Yang mana untuk robot berkaki enam ini di gunakan pola langkah dengan pola tripod yang dibagi menjadi tripod A dan B. untuk konfigurasi kaki ditunjukan pada gambar 3.9 dan untuk digram pewaktuan dan pola langkah ditunjukan dalam gambar 3.10.



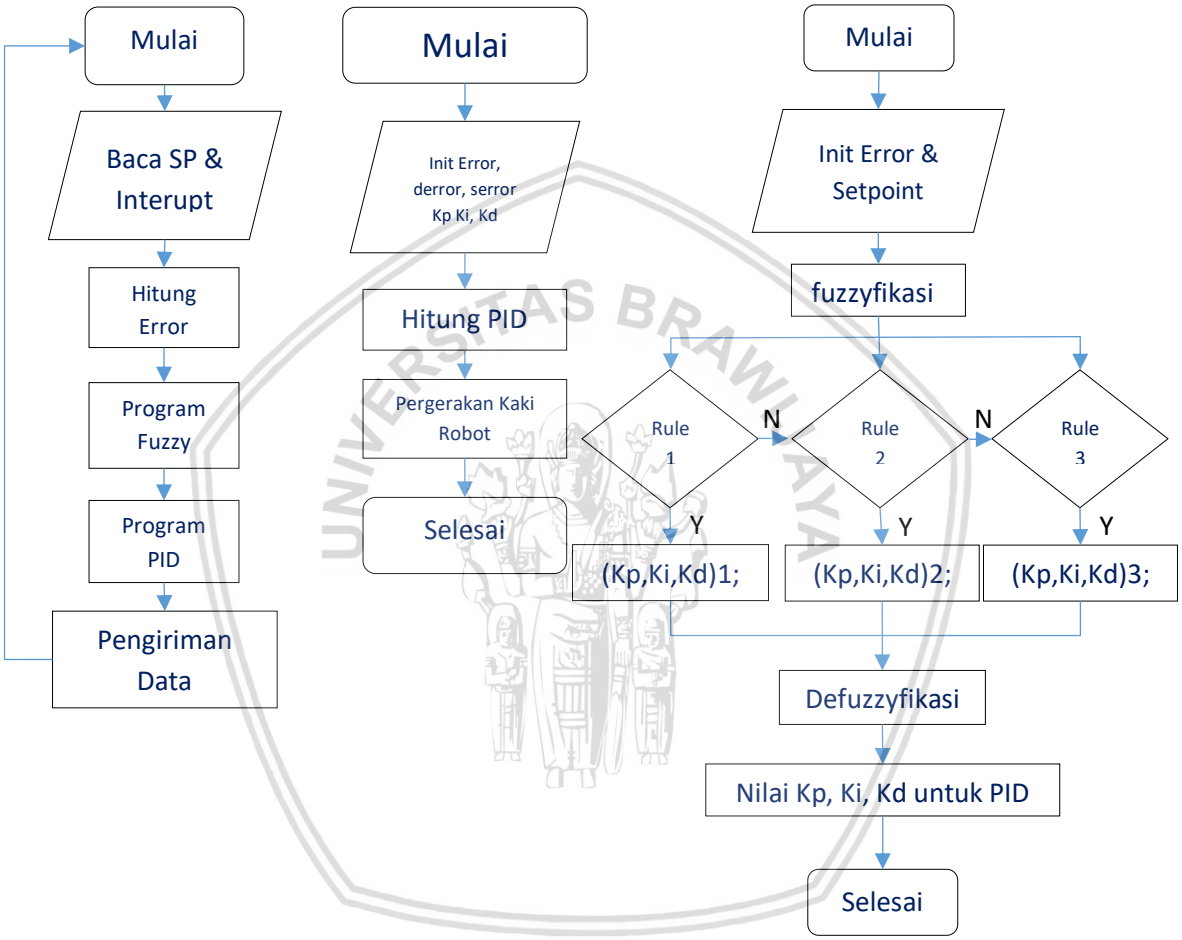
Gambar 3. 9 Konfigurasi susunan Kaki



Gambar 3. 10 Diagram pewaktuan dan ilustrasi pola langkah

3.8 Perancangan Algoritma

Perancangan Algortima adalah perancangan untuk algoritma untuk mendapatkan nilai *tuning* PID dengan seleksi sensor jarak dari nilai pembacaan, dimana secara garis besar algoritma untuk penelitian ini yaitu dibagi menjadi perancangan algoritma fuzzyfikasi, PID dan algoritma gabungan yang ditunjukan pada Gambar 3.11.



Gambar 3. 11 Digram alir program

Pembacaan *setpoint* dan *feedback* kecepatan akan digunakan sebagai masukan untuk program fuzzy dan PID. Untuk program Fuzzy dan PID dimasukan pada timer interupt 0.5 ms dengan prioritas fuzzy sebagai prioritas awal. Hasil program fuzzy akan digunakan untuk mendapatkan parameter kontrol PID yaitu kp, Ki, dan Kd. Nilai keluaran PID akan digunakan sebagai *manipulated variabel* iuntuk memanipulasi sudtu pada kaki kaki robot.

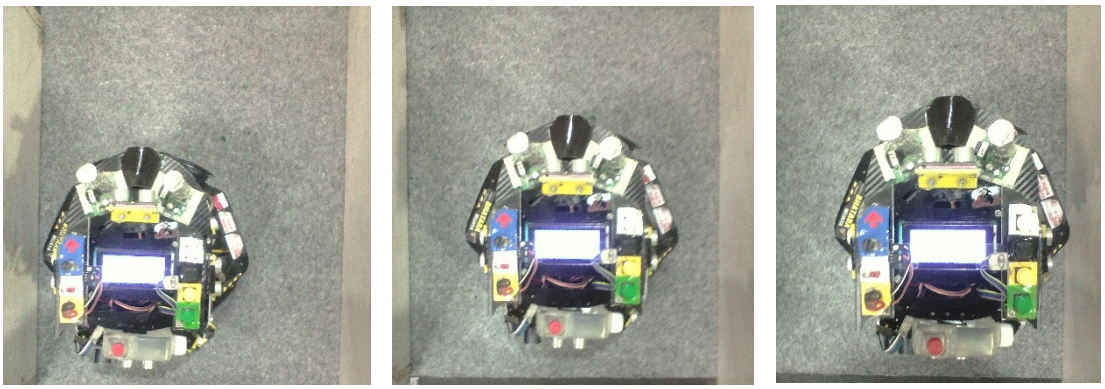
BAB IV
HASIL DAN PEMBAHASAN

Setelah melakukan perancangan langkah selanjutnya adalah pembahasan hasil pengujian yang bertujuan untuk menganalisis alat yang telah dirancang dan di implementasikan telah bekerja sesuai dengan perancangan yang diharapkan. Adapun pengujian yang perlu dilakukan sebagai berikut:

1. Pengujian Pembacaan Sensor Jarak
2. Pengujian *Servo Serial Dynamixel*
    a) Pengujian Sudut *Servo Serial Dynamixel*
3. Pengujian *Fuzzy Logic Algorithm*
    a) Pengujian mendapatkan nilai Kp, Ki, Kd dan aksi kontrol
4. Pengujian Keseluruhan sistem
    a) Pengujian *wall following* pada lintasan lurus
    b) Pengujian *wall following* pada lintasan lurus dengan gangguan
    c) Pengujian *wall following* pada lintasan berbelok ke kanan
    d) Pengujian *wall following* pada lintasan berbelok ke kiri

4.1 Pengujian Pembacaan Sensor Jarak

Pada Pengujian sensor jarak adalah pengujian yang ditujukan untuk mengetahui besarnya nilai pembacaan sensor jarak saat robot berada dilintasan, Pengujian dilakukan dengan menempatkan robot pada tengah lintasan, lalu jarak dekat dengan dinding sisi kanan robot, kemudian dekat dengan sisi kiri robot, begitu juga untuk jarak yang dianggap jauh dari dinding untuk sisi kiri dan kanan. Persamaan 2.8 digunakan untuk menghitung atau mengkonversi sinyal sensor agar didapat jarak sensor. Kemudian posisi robot saat diarena pengujian ditunjukan pada gambar 4.1 dan tabel 4.1 menunjukan besarnya pembacaan jarak pada berbagai kondisi robot dari dinding.



(a) (b) (c)

Gambar 4. 1 Pengujian sensor jarak robot pada lintasan untuk kondisi robot (a) dekat, (b) sedang dan (c) jauh dari dinding.

Tabel 4 1 Hasil Pembacaan Sensor Jarak

No	SENSOR	Set	SPL	SPU	N	e	Z	e	P	e
1	Sensor Jarak Depan	16	15	17	6	-10	15	-1	26	10
2	Sensor Jarak Kanan	16	15	17	6	-10	16	0	24	8
3	Sensor Jarak Kiri	16	15	17	6	-10	16	0	24	8
4	Sensor Jarak Kanan Depan	16	15	17	6	-10	17	1	26	10
5	Sensor Jarak Kiri Depan	16	15	17	6	-10	17	1	26	10
Jarak Rata Rata		16	15	17	6	-10	16	0	26	10
Jarak terdekat		16	15	17	6	-10	15	-1	24	8
Jarak terjauh		16	15	17	6	-10	17	1	26	10

Keterangan Tabel dan Gambar 11, 12 :

N (NEGATIF) : Dekat (jarak minimal sensor)

Z (ZERO) : Sedang (jarak robot di tengah)

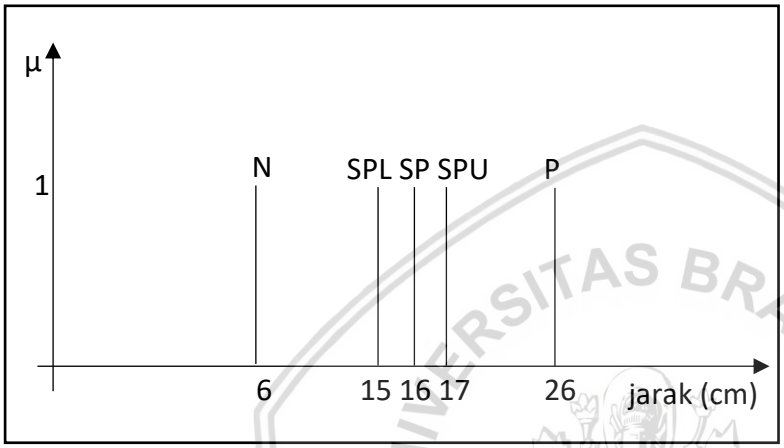
P (POSTIF) : Jauh (jarak terjauh robot)

SP : Set Point

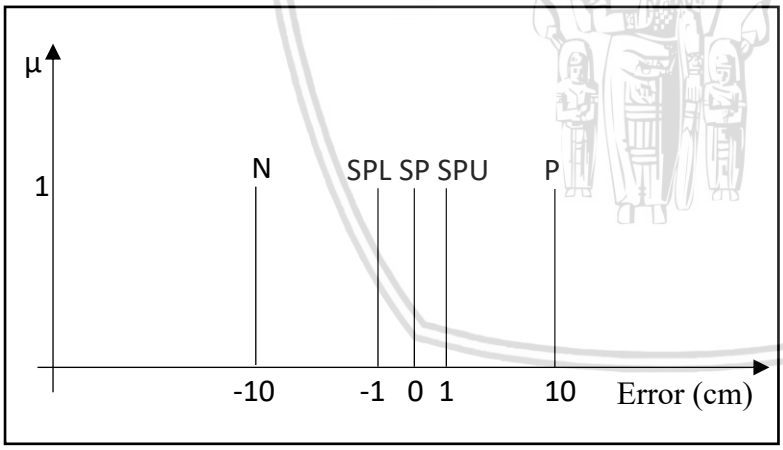
SPU : SetPoint Atas

SPL : SetPoint Bawah

Dari tabel 4.1 dapat diketahui nilai pengukuran jarak robot terhadap dinding ketika berada ditengah lajur adalah bernilai 16 cm yang pada peneltian ini diberi nilai toleransi untuk nilai tengah yang didefinisikan sebagai batas setpoint atas dan setpoint bawah, hal ini dikarenkan untuk nilai kondisi robot berada ditengah lintasan adalah disekitar 15, 16, dan 17 cm, jarak terdekat adalah 6 cm dan untuk posisi terjauh adalah 26 cm, kemudian fungsi keanggotaan nilai jarak di gambarkan dengan aturan fuzzy singletone ditunjukan oleh gambar 4.2 dan fungsi keanggotaan error ditunjukan pada gambar 4.3.



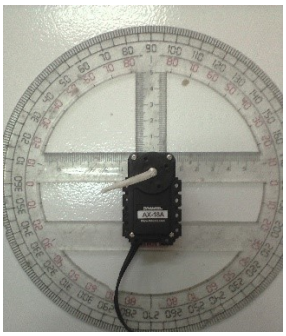
Gambar 4. 2 Fungsi keanggotaan jarak pada pembacaan sensor



Gambar 4. 3 Fungsi keanggotaan Error jarak pada pembacaan sensor

4.2 Pengujian *Servo Serial Dynamixel*

Pada pengujian sudut dari *servo serial dynamixel* servo dipasangi dengan penunjuk sudut busur 360° seperti yang ditunjukkan dalam gambar 4.4.



Gambar 4. 4 Servo serial dynamixel yang telah dipasangi busur 360°

Besar sudut hasil pengujian kemudian dibandingkan dengan besar sudut teori berdasarkan nilai data serial seperti pada Tabel 4.2.

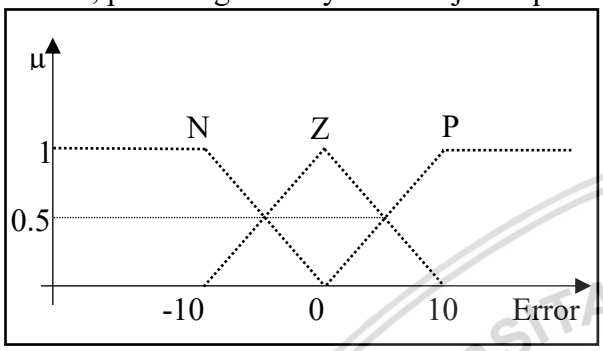
Tabel 4 2 Hasil Pengujian Sudut *Servo Serial Dynamixel*

Nilai Serial	Sudut Teori (°)	Pengujian 1	Pengujian 2	Pengujian 3	Rata- Rata Sudut Praktik (°)	Persen *error* (%)
0	0	0	0	0	0	0
150	43,98826979	44	44	44	44	0,0266595
300	87,97653959	89	90	89	89,33333333	1,5187989
450	131,9648094	133	133	133	133,3333333	1,0263929
600	175,9530792	177	176	176	176,3333333	0,2156450
750	219,941349	221	221	222	221,3333333	0,6289085
900	263,9296188	263	263	263	263	0,3534672
1000	293,255132	293	292	293	292,6666667	0,2010701

Pengujian untuk mengetahui data servo dilakukan sebanyak 3 kali dan kemudian pada data pengujian yang diambil adalah nilai rata-rata. Dari hasil pengujian pada Tabel 4.2 dapat dilihat bahwa nilai sudut perubahan servo berbanding lurus dengan sudut teori dan antara sudut teori dengan sudut sebenarnya memiliki *error* < 1,6%. Persen *error* terbesar perubahan sudut servo dari sudut teori ialah sebesar 1,52%, sedangkan yang terkecil adalah 0% pada sudut 0°. Nilai error ini masih dapat dikatakna baik karenan besar sudut yang dihasilkan memiliki error yang kecil dan tidak begitu berpengaruh pada sistem.

4.3 Pengujian *Fuzzy Logic Algorithm*

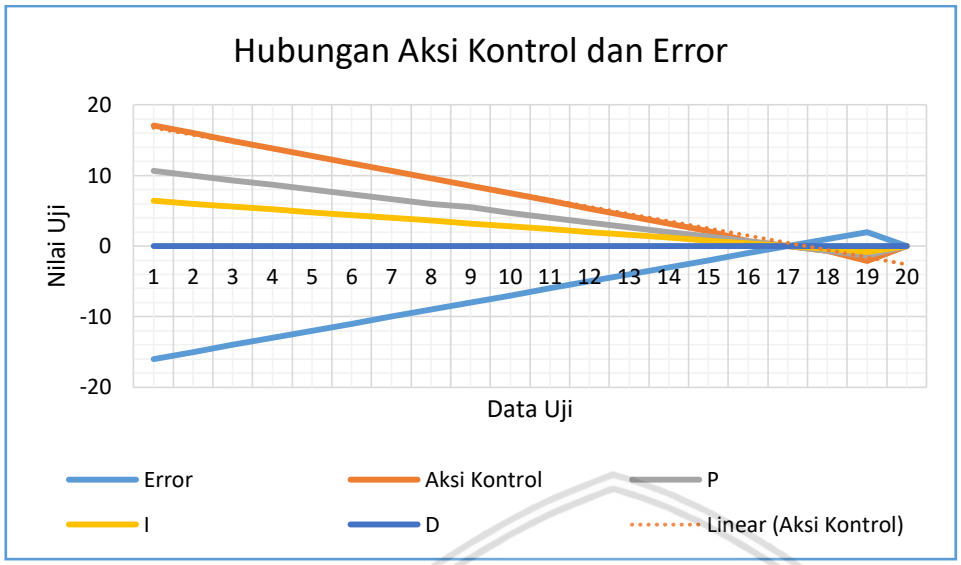
Pengujian dilakukan dengan memasukan nilai uji pada perhitungan fuzzy dengan mengacu pada fungsi keanggotaan nilai jarak yang ditunjukan pada gambar 4.5. Pengujian ini bertujuan untuk mengetahui hasil dari proses metode *fuzzy infrence* dalam mengolah nilai yang diterima yaitu nilai error dari jarak sehingga didapat kisaran nilai Kp, Ki dan Kd yang dihasilkan dari metode, perhitungan fuzzy ini ditunjukan pada tabel 4.3 dan gambar 4.6.



Gambar 4. 5 Fungsi keanggotaan nilai uji Error

Tabel 4 3 Pembacaan Nilai Kp, Ki, dan Kd

Uji	Data	Error	Kp	Ki	Kd	P	I	D	Aksi kontrol
1	0	-16	10.67	1	0.44	10.67	6.40	0.005	17.07
2	1	-15	10	1	0.44	10	6	0.005	16
3	2	-14	9.33	1	0.44	9.33	5.60	0.005	14.93
4	3	-13	8.67	1	0.44	8.67	5.20	0.005	13.87
5	4	-12	8	1	0.44	8	4.80	0.005	12.80
6	5	-11	7.33	1	0.44	7.33	4.40	0.005	11.73
7	6	-10	6.67	1	0.44	6.67	4	0.005	10.67
8	7	-9	5.4	1	0.36	5.4	3.60	0.005	9
9	8	-8	4.27	1	0.28	4.27	3.2	0.005	7.47
10	9	-7	3.27	1	0.22	3.27	2.8	0.005	6.07
11	10	-6	2.4	1	0.16	2.4	2.4	0.005	4.8
12	11	-5	1.67	1	0.11	1.67	2	0.005	3.67
13	12	-4	1.07	1	0.07	1.07	1.6	0.005	2.67
14	13	-3	0.6	1	0.04	0.6	1.2	0.005	1.8
15	14	-2	0.27	1	0.02	0.27	0.8	0.005	1.07
16	15	-1	0.07	1	0	0.07	0.4	0.005	0.47
17	16	0	0	0	0	0	0	0	0
18	17	1	0.07	1	0	-0.07	-0.4	-0.005	-0.47
19	18	2	0.13	1	0.02	-0.13	-0.8	-0.005	-1.07
20	16	0	0	0	0	0	0	0	0



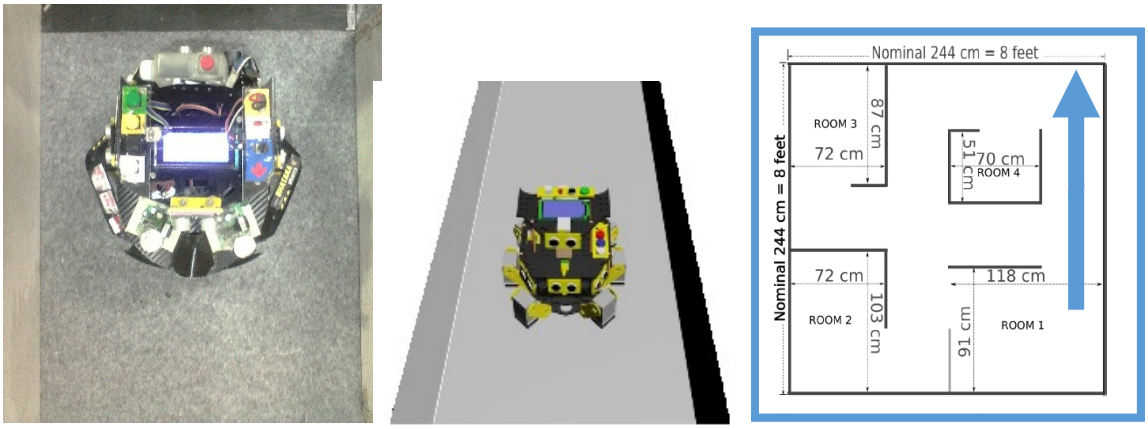
Gambar 4. 6 Grafik Hubungan Error dan Aksi Kontrol pengujjian metode fuzzy dalam menghitung parameter PID

Dari Tabel 4.3 dan juga gambar 4.6 dapat terlihat bahwa nilai *tuning* Kp, Ki, dan Kd, dengan menggunakan metode fuzzy dapat digunakan hal ini bisa disimpulkan jika melihat bahwa hubungan antara Error dan perubahan nilai aksi kontrol adalah sebanding.

4.4 Pengujian Keseluruhan Sistem

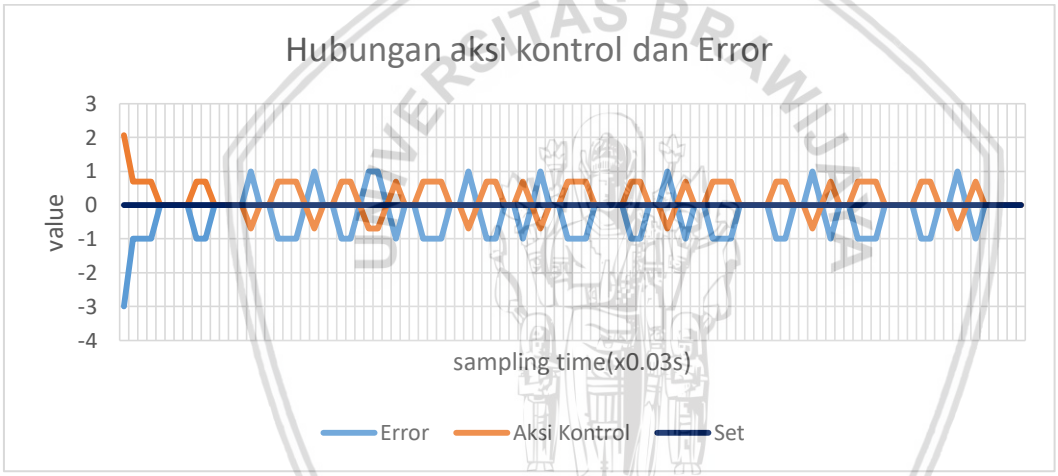
4.4.1 Pengujian *wall following* pada lintasan lurus maju

Pengujian Gerak Maju pada robot ini dilakukan pada jalur lurus, dengan dinding di sisi kiri dan kanannya, pengujian dilakukan dimana robot bergerak maju dengan dinding tanpa halangan. Nilai pembacaan ditampilkan pada tabel 4.1 pada lampiran dan untuk grafik data ditunjukan pada gambar 4.8 waktu tempuh dari robot untuk menyelesaikan lintasan adalah selama 3.02 s yang diukur menggunakan alat ukur *stopwatch*. Perspektif robot ketika berada dilintasa uji ditunjukan pada 4.7.



(a) (b) (c)

Gambar 4. 7 (a) foto robot (b) perspektif robot (c) perspektif lintasan robot di arena pengujian



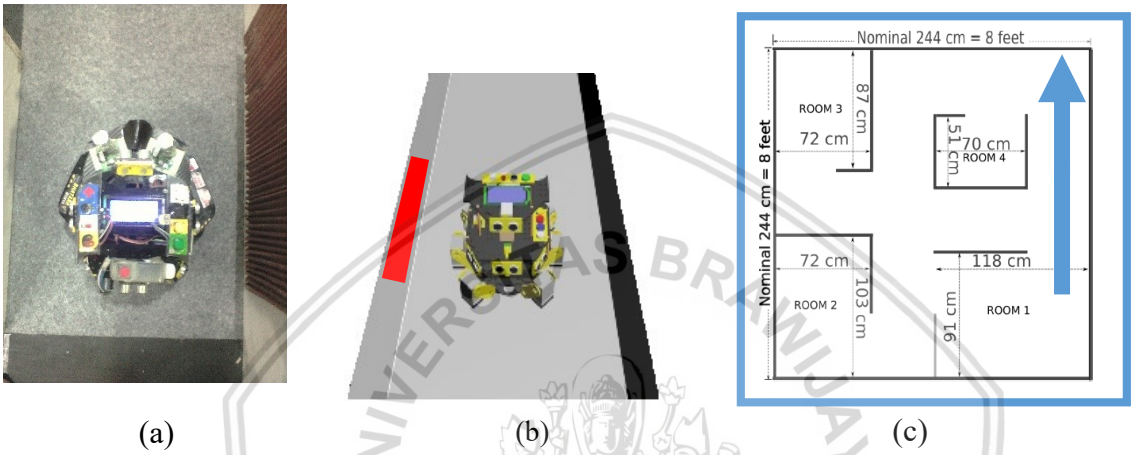
Gambar 4. 8 Grafik Hubungan Error dan Aksi Kontrol pada telusur saat lintasan lurus

Pada pengujian pada lintasan yang berbentuk lurus dapat terlihat robot mampu memberikan aksi kontrol sehingga robot berada di sekitar nilai set, dimana terlihat ketika respon nilai error mendekati nilai set, maka aksi kontrol diturunkan, lalu ketika terjadi fluktuasi dimana nilai berkurang dari nilai set maka aksi kontrol juga mengalami fluktuasi yang sama,dengan persen error 2.997%.

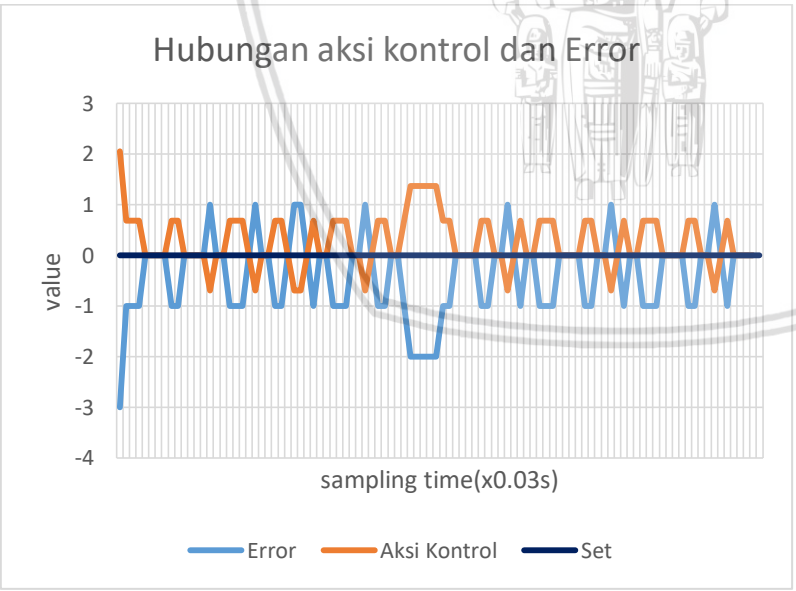
Pada pengujian berikutnya adalah pengujian dengan memberikan gangguan yang terlihat pada gambar 4.9 dan tabel 4.2 pada lampiran, di dinding lintasan ditujukan agar dinding lintasan tidak selalu rata. Dan pada pengujian ini dengan melakukan telusur kanan.

4.4.2 Pengujian *wall following* maju pada lintasan lurus dengan gangguan

Pengujian Gerak pada robot ini dilakukan pada jalur simpangan, dengan dinding disisi kiri dan kanannya, pengujian ditujukan untuk mengetahui apakah robot dengan perubahan nilai gain PID mampu atau tidak melakukan gerakan penyesuaian pada telusur dinding kanan. Tabel Pembacaan parameter PID untuk Telusur lurus pada lintasan lurus ditunjukan pada tabel 4.2 pada lampiran. Untuk perspektif robot ketika diberi gangguan ditunjukan pada gambar 4.9.



Gambar 4. 9 (a) foto robot (b) perspektif robot diberi gangguan (c) perspektif lintasan robot di arena pengujian



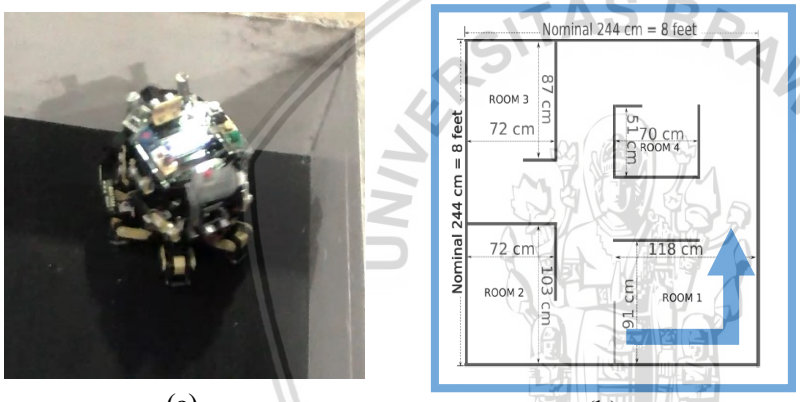
Gambar 4. 10 Grafik Hubungan Error dan Aksi Kontrol pada telusur saat lintasan lurus dengan gangguan

Pada pengujian pada lintasan yang berbentuk lurus dengan gangguan seperti terlihat pada gambar 4.10 dapat terlihat robot mampu memberikan aksi kontrol sehingga robot berada di sekitar nilai set, dengan persentase error sebesar 3.472% dimana terlihat ketika respon nilai error

mendekati nilai set, maka aksi kontrol diturunkan, lalu ketika terjadi fluktuasi dimana nilai berkurang dari nilai set maka aksi kontrol juga mengalami fluktuasi yang sama, dan ketika terjadi gangguan maka aksi kontrol yang dilakukan lebih besar

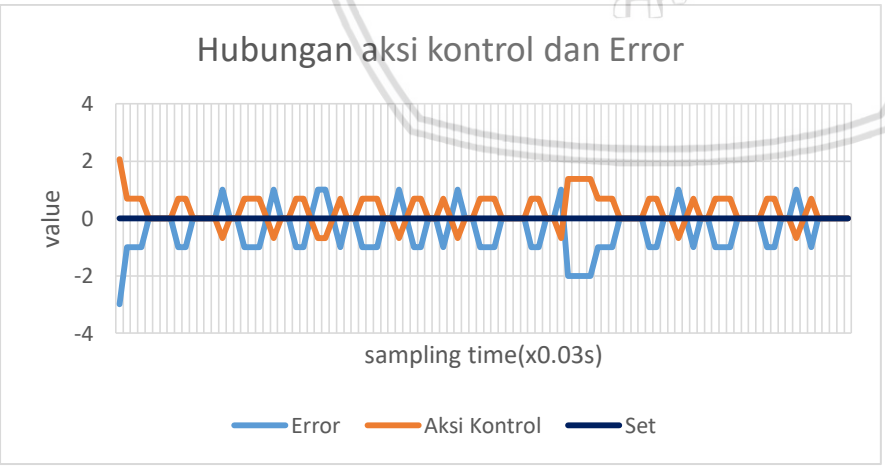
4.4.3 Pengujian *wall following* kanan

Pengujian Gerak pada robot ini dilakukan pada jalur simpangan, dengan dinding disisi kiri dan kanannya, pengujian ditujukan untuk mengetahui apakah robot dengan perubahan nilai gain PID mampu atau tidak melakukan gerakan penyesuaian pada telusur dinding kanan. Tabel Pembacaan parameter PID untuk Telusur Kanan pada lintasan berbelok Kanan ditampilkan pada tabel 4.3 pada lampiran dan grafik pada gambar 4.12. Perspektif robot saat berada dilintasan ditunjukan pada gambar 4.11.



(a) (b)

Gambar 4. 11. (a) foto robot (b) perspektif lintasan robot di arena pengujian telusur kanan

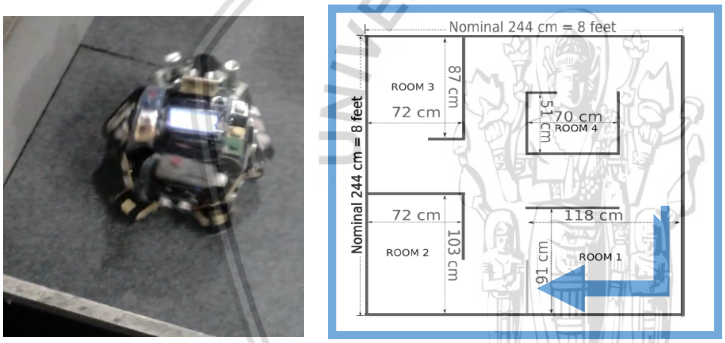


Gambar 4. 12 Grafik hubungan aksi kontrol dan Error pada telusur kanan

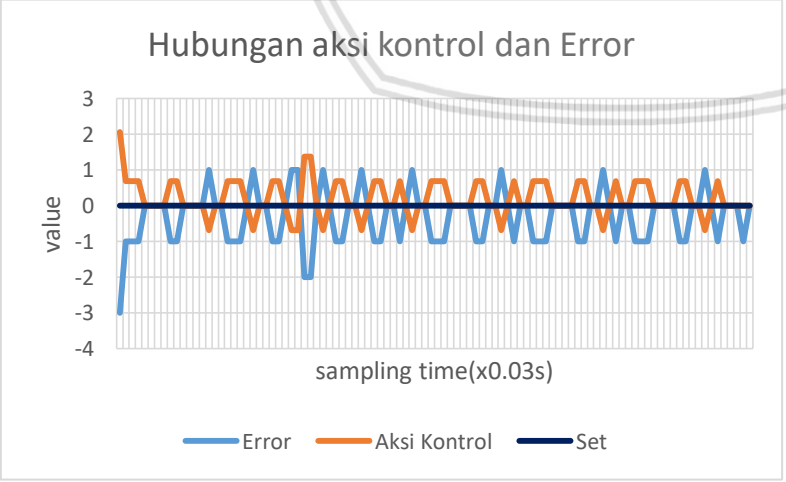
Pada pengujian pada lintasan yang berbentuk siku dapat terlihat grafik pada gambar 4.12 robot mampu memberikan aksi kontrol sehingga robot berada di sekitar nilai set, dengan persentase error sebesar 3.409% dimana terlihat ketika respon nilai error mendekati nilai set, maka aksi kontrol diturunkan, lalu ketika terjadi fluktuasi dimana nilai berkurang dari nilai set maka aksi kontrol juga mengalami fluktuasi yang sama

4.4.4 Pengujian *wall following* kiri

Pengujian Gerak pada robot ini dilakukan pada jalur simpangan, dengan dinding disisi kiri dan kanannya, pengujian ditujukan untuk mengetahui apakah robot dengan perubahan nilai gain PID mampu atau tidak melakukan gerakan penyesuaian pada telusur dinding kanan. Tabel Pembacaan parameter PID untuk Telusur Kiri pada lintasan berbelok ditampilkan pada tabel 4.4 pada lampiran dan pada gambar 4.14. untuk perspektif robot saat berada dilintasan ditunjukan pada gambar 4.13.



Gambar 4. 13 (a) foto robot (b) perspektif robot (c) perspektif lintasan robot di arena pengujian



Gambar 4. 14 Grafik hubungan aksi kontrol dan Error pada telusur kiri

Pada pengujian pada lintasan yang berbentuk siku terlihat pada grafik gambar 4.14 robot mampu memberikan aksi kontrol sehingga robot berada di sekitar nilai set, dengan persentase error sebesar 3.282% dimana terlihat ketika respon nilai error mendekati nilai set, maka aksi kontrol diturunkan, lalu ketika terjadi fluktuasi dimana nilai berkurang dari nilai set maka aksi kontrol juga mengalami fluktuasi yang sama.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil analisis dan pengujian setiap bagian dan keseluruhan sistem yang telah dilakukan, maka dapat dilambil kesimpulan sebagai berikut:

1. Penerapan logika fuzzy ditujukan untuk mendapatkan parameter kontrol PID, sesuai dengan perancangan, serta sistem elektronik, mekanik maupun pola langkah robot *hexapod* agar dapat diimplementasikan pada robot *hexapod.* Dimana saat pembacaan sensor jarak semakin mendekati nilai set (kondisi jarak ideal terhadap dinding) maka aksi kontrol akan berkurang dan error semakin kecil dimana terlihat bahwa pada telusur dinding kanan pada lintasan lurus didapat persentasi error rata rata yaitu 2.997%, dan ketika diberi gangguan persentase error adalah 3.472%, untuk telusur dinding kanan dengan lintasan berbentuk siku didapat persentase error sebesar 3.409% dan juga untuk telusur kiri pada lintasan berbentuk siku persentase error 3.409%. Nilai persentase error ini masih dikategorikan layak karena masih <5%.
2. Pembacaan sensor jarak ultrasonik PING))) dapat digunakan sebagai nilai batas yang menetukan aturan logika fuzzy, Dimana nilai kondisi yang diharapkan yaitu robot berada dijarak 16 cm dari dinding sebagai kondisi ideal robot ditengah lajur lintasan. Kemudian nilai permbacaan sensor secara aktual akan digolongkan sebagai kondisi dekat, sedang dan jauh oleh defuzzyfikasi (Weighted Average), dengan mencari nilai rata rata dari keluaran seleksi kondisi *fuzzyfier.* Kemudian nilai tersebut dijadikan parameter masukan nilai *gain* kontroler PID. Yang kemudian dijadikan sinyal manipulasi gerak untuk memanipulasi sudut Coxa dan tibia pada gait gerak robot.
3. Algoritma pergerakan dan pola langkah dari robot berkaki 6 didapatkan dengan melalui perancangan urutan kerja kaki robot berdasarkan aturan tripod (pola gerak 3 kaki) yang dibedakan menjadi tripod A dan tripod B, yang mana robot gait gerak robot akan mendefinisikan pemanggilan aktuator untuk tiap kaki, yang pada pemanggilan tiap kakinya akan mengakses 3 alamat servo (*Coxa, femur, tibia*). pada alamat tersebut akan diberikan nilai besarnya sudut gerak tiap kaki dan besar perubahannya juga ditentukan oleh sinyal manipulasi PID.

5.2 Saran

Untuk pengembangan penelitian ini, ada beberapa saran yang dapat dilakukan antara lain:

1. Untuk Peneliti agar memperhatikan dengan baik perancangan mekanik robot, terutama pada peletakan sensor, serta kualitas sensor yang digunakan karena sangat berpengaruh pada aksi kontrol.
2. Untuk Peneliti agar memperhatikan jumlah penatapan kondisi yang ingin ditetapkan sebagai aturan dari Logika Fuzzy Kontroler .

DAFTAR PUSTAKA

Febrian, W. (2012). *Penggunaan Sistem Tiga Sendi Pada Robot KRCI Divisi Berkaki untuk mengatasi onjek uneven Floor*. Malang: Universitas Brawijaya.

Kuswadi, S. (2007). *Kendali Cerdas : Teori dan Aplikasi Praktisnya* . Yogyakarta: ANDI.

Lee, C. (1990). *Fuzzy logic in control systems: fuzzy logic controller, Parts I and II. IEEE Trans Syst Man Cybern.*

Ogata, K. (1997). *Modern Control Engineering Fifth Edition.* New Jersey: Prentince Hall.

Parallax. (2017). *PING))) Ultrasonic Distance Sensor*. California, USA: Parallax.

Pitowarno, E. (2006). *Robotika : Desain, Kontrol dan Kecerdasan Buatan*. Yogyakarta: Penerbit ANDI.

Putra, D. A. (2015). *Penerapan Kontroler Self Tuning Parameter PI dengan metode Logika Fuzzy pada Mobile Robot*. Malang: Universitas Brawijaya.

RISTEKDIKTI. (2017). *Panduan Kontes Robot Cerdas Indonesia 2017*. Jakarta: RISTEKDIKTI.

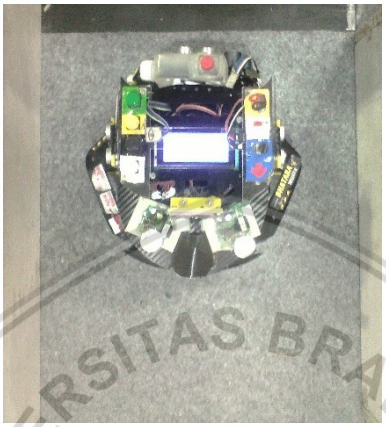
STMicroelectronics. (2017, Oktober 3). *STM32F4DISCOVERY, STM32F4 high performance discovery board*. Diambil kembali dari my.st.com: my.st.com

Sugeno, M., & Takagi, T. (1985*). Fuzzy identification of systems and its applications to modeling and control*. IEEE Trans. Systems Man Cybern, 116–132.

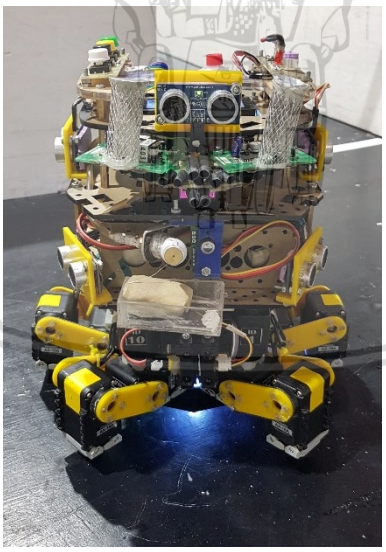
Tara, R. Y. (2010). *Desain dan Implementasi Logika Fuzzy sebagai Sistem Navigasi Wall Following pada Mobile Robot KRCI*. Malang: Universitas Brawijaya.

LAMPIRAN 1

DOKUMENTASI ALAT



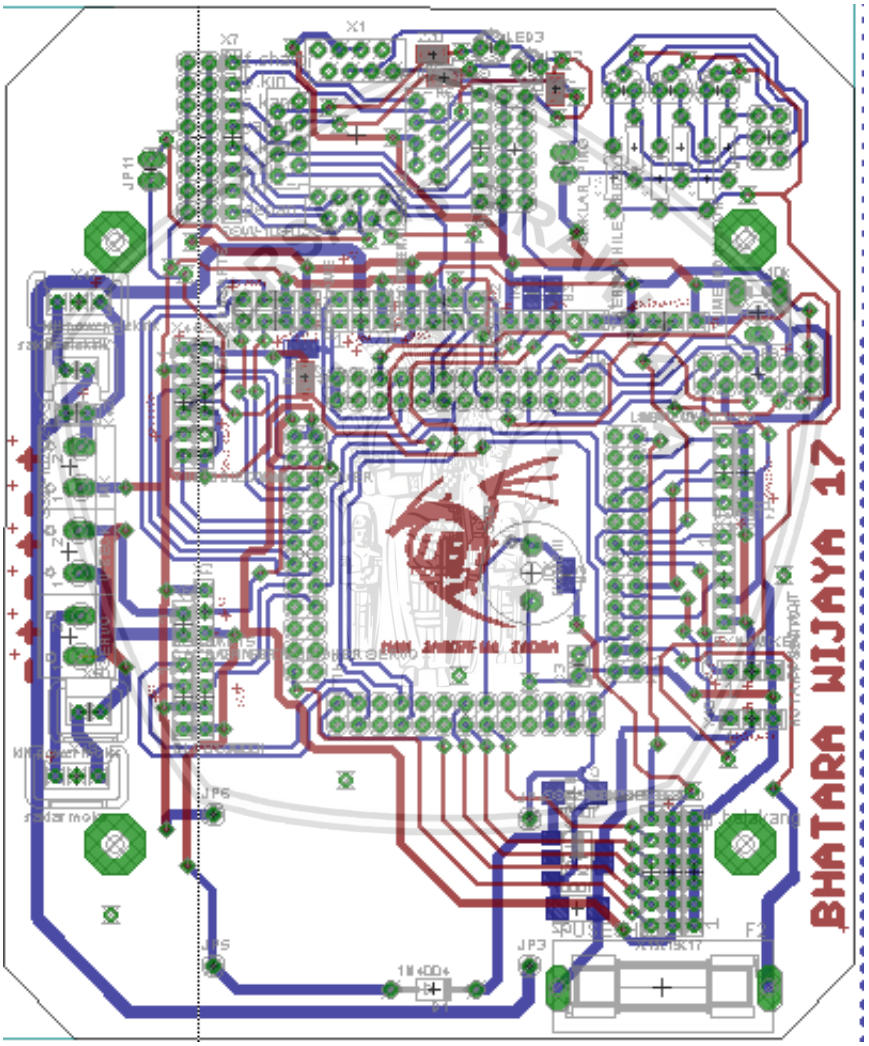
Gambar 1.1 Robot di lintasan



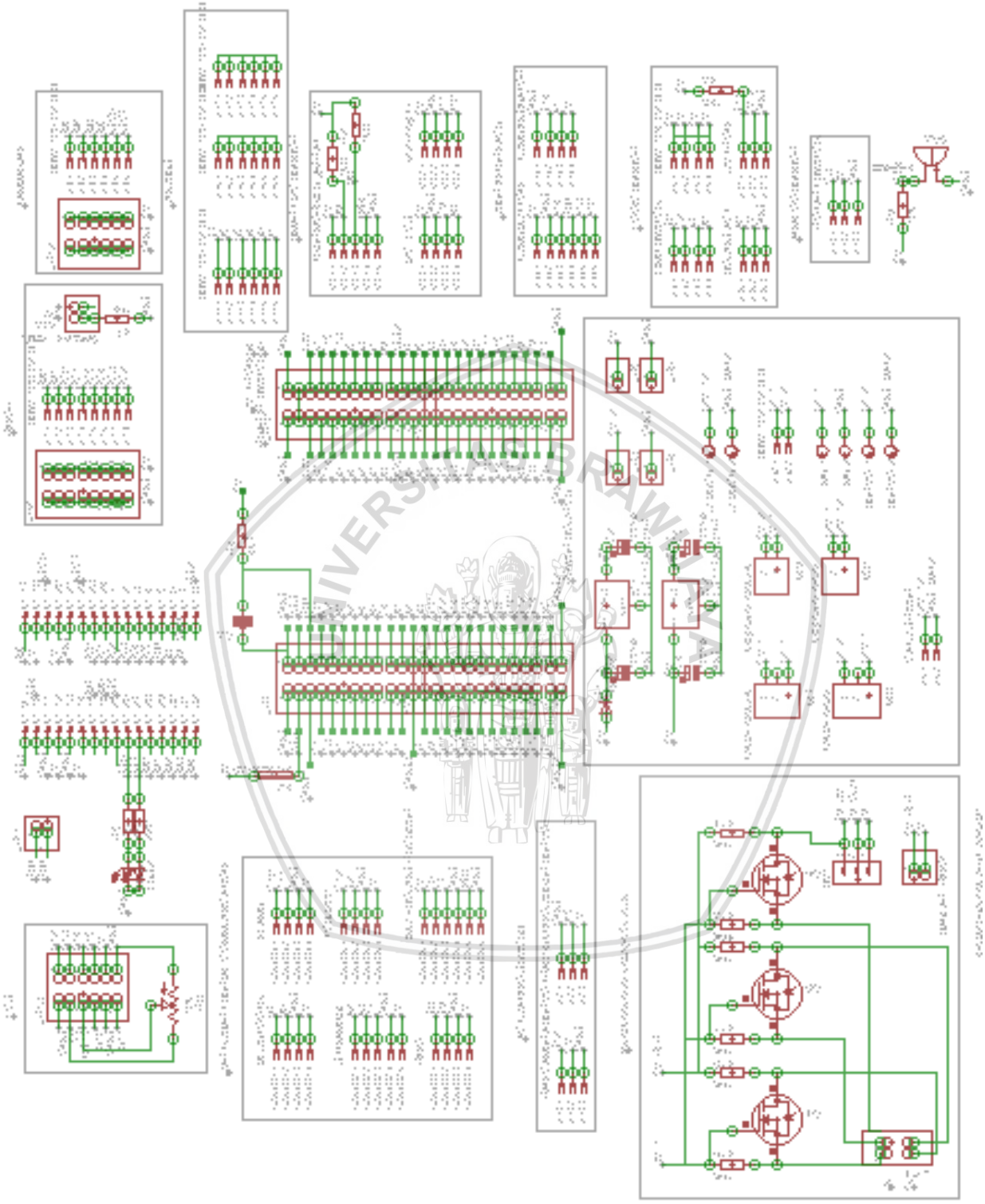
Gambar 1.2 Robot tampak depan

LAMPIRAN 2

Layout Board



Gambar 2.1 lay out Board Utama



Gambar 2.2 desain skematik board utama

LAMPIRAN 3

LISTING PROGRAM

Listing Program Mikrokontroler Utama

Library main.h

```
/*PROJECT BW Ver.1.MAIN LIBRARY*/
#ifndef MAIN_H
#define MAIN_H

/* BW STANDARD LIBRARY*/
#include "stm32f4xx.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_adc.h"
#include "stm32f4xx_tim.h"
#include "stm32f4xx_usart.h"
#include "stm32f4xx_i2c.h"
#include "misc.h"
#include "math.h"
#include "stdio.h"
#include "stdlib.h"
//#include "stdint.h"
#include "string.h"
#include "stdarg.h"
#include "stdbool.h"
//-----------------KRPAI Library--------------//
//#include "delay.h"
//#include "lcd.h"

//ESSENTIAL GLOBALLY
#include <bw_macro_lib.h>
#include <bw_global_var.h>
#include <bw_systick_delay.h>

//=====NEW LIBRARY (PHOENIX CODE)=== //
#include "BW_Hex_Cfg.h"
#include "BW_Phoenix.h"
#include "BW_Phoenix_Code.h"
#include "BW_Phoenix_Driver_AX12.h"

//USER INTERFACES
#include <bw_user_interface.h>
#include <bw_rotary_switch.h>
#include <bw_lcd.h>

//COLOUR AND LINE SENSOR
//#include <bw_tracer_decoder.h>

//BW SOUND ACTIVATION
#include "bw_sound_activation.h"

//ULTRASONIC/SONAR
#include <bw_infrared_proximity.h>

//CONTACT BUMPER
#include "bw_bumper.h"

//FLAME SENSOR AND THERMOPHILE
ARRAY
#include <bw_UV-TRON.h>
#include <bw_tpa_servo.h>

//BRUSHLESS DC FAN
#include <bw_bldc_fan.h>

//Hydro Pump Extinguisher
#include "bw_extinguisher.h"

#include "bw_ping_variables.h"
#include "bw_pid_var.h"
//OTHER
//#include <bw_easy_init.h>

//BW PING))) RECEIVER
#include "bw_ping_receiver.h"

//BW HEXAPOD COMMAND
#include "bw_hexapod_cmd.h"

//BW USART COMMUNICATION
#include "bw_usart_comm.h"

//BW MPU6050 IMU
//#include "bw_mpu6050.h"
//#include "bw_gy_85.h"
#include "bw_cmps11.h"
//PID CONTROLLER
#include <bw_pid_controller.h>

//BW UTILITIES
```

```
#include "bw_utilities.h"

//BW DYNAMIXEL XL-320
#include "bw_dynamixel_xl_320.h"

//BW ARTICULATED MANIPULATOR
//#include "bw_articulated_manipulator.h"

//BW SHARP INFRARED SENSOR
#include "bw_sharp_ir.h"

//BW I2C LCD
//#include "bw_i2c_lcd.h";

#include "bw2017_algorithm.h"

//BW ADDITIONAL FLAME SENSOR SIDE
#include "bw_flame_sensor.h"

//BW DYNAMIXEL AX-12A LIBRARY
#include "bw_dynamixel_ax_12a.h"

//BW DYNAMIXEL HEXAPOD LOCOMOTION

#include "bw_dynamixel_hexapod_cmd.h"

//BW MPU6050 RECEIVER
#include "bw_mpu6050_receiver.h"

//BW TCS3200
#include "bw_tcs3200_receiver.h"

//Global Typedef
GPIO_InitTypeDef  GPIO_InitStructure,
GPIO_I2C,GPIO_InitStructure;
I2C_InitTypeDef  I2C_InitStructure;
TIM_TimeBaseInitTypeDef
TIM_TimeBaseStructure;
TIM_OCInitTypeDef          TIM_OCInitStructure;

#define BW_MODE_DEFAULT 0
#define BW_MODE_1 1
#define BW_MODE_2 2
#define BW_MODE_3 3
#define BW_MODE_4 4
#define BW_MODE_5 5
#define BW_MODE_6 6

extern unsigned int BW_LOCATION[];
extern unsigned int LOCATION_CORRIDOR;
extern unsigned int LOCATION_ROOM;
#define PRESENT_LOC 0
#define LAST_LOC 1

extern unsigned int INITIAL_ZONE;
extern unsigned int ZONE;
extern unsigned int ZONE_BLACK;
extern unsigned int ZONE_RED;
extern unsigned int ZONE_BLUE;
extern unsigned int TRACER;
extern unsigned int FIRE_PUTOFF;

extern unsigned int COLOUR_STAT;

extern unsigned int ZONE;
extern unsigned int ZONE_RED;
extern unsigned int ZONE_BLUE;

/*BW FOLLOW CARPET VARIABLES*/

//extern unsigned int NORMAL;
//extern unsigned int INVERSE;

/*BW CAT COLLISION SHIFT FOLLOW*/

extern unsigned int CAT_SHIFT_FOLLOW;
extern unsigned int SHIFT_FOLLOW;
extern unsigned int NO_SHIFT;

extern uint16_t led_off_counter;
extern uint16_t led_snd_off_counter;

/*BW CROSS_ZONE VARIABLES*/

extern unsigned int
BW_CROSSING_AUTHORITY;
extern unsigned int CROSS_DISABLED;
extern unsigned int CROSS_ENABLED;
extern unsigned int BW_CROSSING_STATUS;
extern unsigned int NO_CROSSING;
extern unsigned int CROSSING_SUCCESS;
extern unsigned int CROSSING_FAILED;

/* BW SOUND SCAN*/

#define SOUND_SCAN
GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_0)

//CARPET CODE
extern unsigned int COLOUR_NONE;
extern unsigned int COLOUR_BLACK;
extern unsigned int COLOUR_GRAY;
extern unsigned int COLOUR_WHITE;
extern unsigned int COLOUR_RED;
extern unsigned int COLOUR_BLUE;


//FIRE FLAG
extern unsigned int FIRE;
```

```
//MANDATORY COMMAND HYBRID IMU
PROCESSOR
extern const int IMU_CMD_SEND_RAW;
//DATA
```

Library main.c

```
/****************************************
Project   : BW* Main Controller
University: University of Brawijaya
Department: Electrical Engineering Department,
Engineering Faculty
Division  : Legged Fire-Fighting Division (KRPAI)
Board Type: STM32F4 Discovery Board
Chip  Type: STM32F407VG
****************************************/

#include "main.h"
int toggle=1;
void BW_Initialization(void);
int main(void)
{
int BUTTON_ACT;
int SOUND_ACT;
BW_Initialization();
delay_ms(500);
Dynamx_Mov_Static(HEXSPD_ULTRASLOW,
IKCALC_DISABLE);
while( (BW_BUTTON_UNCLICKED) &&
(SOUND_INACTIVE) )
{
if(BW_BUTTON_CLICKED){BUTTON_ACT=1;
break;}
if(SOUND_ACTIVATED){SOUND_ACT=1;break
;}
RotSwitch_Sampling();
Display_MODE();
if(BW_BUTTON_CLICKED){BUTTON_ACT=1;
break;}
if(SOUND_ACTIVATED){SOUND_ACT=1;break
;}
}

if(BW_BUTTON_CLICKED){BUTTON_ACT=1;
}
if(SOUND_ACTIVATED){SOUND_ACT=1;}


if(BUTTON_ACT==1)
{
BW_Buzz(1);
Sendto_PC(USART1,"BW BUTTON
ACTIVATED! \r");
lcd_display_clear();
lcd_gotoxy(5,0);sprintf(lcd,"PROJECT
BW");lcd_putstr(lcd);
lcd_gotoxy(4,1);sprintf(lcd,"BRAWIJAYA");lcd_p
utstr(lcd);
lcd_gotoxy(0,2);sprintf(lcd,"FIREFIGHTER
ROBO");lcd_putstr(lcd);
lcd_gotoxy(0,3);sprintf(lcd,"BUTTON
ACTIVATED");lcd_putstr(lcd);
delay_ms(100);}
else if(SOUND_ACT==1)
{
BW_Buzz(2);
Sendto_PC(USART1,"SOUND ACTIVATED!\r");
lcd_display_clear();
lcd_gotoxy(5,0);sprintf(lcd,"PROJECT BW");
lcd_putstr(lcd);
lcd_gotoxy(4,1);sprintf(lcd,"BRAWIJAYA");
lcd_putstr(lcd);
lcd_gotoxy(0,2);sprintf(lcd,"FIREFIGHTER
ROBO");
lcd_putstr(lcd);
lcd_gotoxy(0,3);sprintf(lcd,"SOUND ACTIVE");
lcd_putstr(lcd);
delay_ms(100);
SND_ACT_LED_ON; //turn on sound led indicator
}

switch(rot_switch_mode)
{
case BW_MODE_DEFAULT:{}break;

case BW_MODE_1:
{BW_Initial_Setup(rot_switch_mode);
while(1)
{
H_2017_Algorithm();
while(1)
{
BW_Buzz(1);
Dynamx_Mov_Static(HEXSPD_SLOW,
IKCALC_DISABLE);
}
}
}break;
case BW_MODE_2:
{
BW_Initial_Setup(rot_switch_mode);
lcd_display_clear();
```

```
extern unsigned int BW_SOUND_ACTIVATED;
//SOUND
extern const int SOUNDCHECK;
#endif
```

```
lcd_gotoxy(1,0);sprintf(lcd,"UB");lcd_putstr(lcd);
lcd_gotoxy(2,2);sprintf(lcd,"FOLLOW");
lcd_putstr(lcd);
CAT_FLAG_B=CAT_NOT_DETECTED;
BW_LOCATION[0]=LOCATION_CORRIDOR;
R4ATO3_SP_ROUTE=R4ATO3_SP_ACTIVE;
BW_PID_Init();

BW_LOCATION[0]=LOCATION_ROOM;
BW_PID_Init();
while(1)
{
BW_FollowTracer_Right();
Get_Data_PID ();
}}break;
case BW_MODE_3:
{
BW_Initial_Setup(rot_switch_mode);
lcd_display_clear();
lcd_gotoxy(1,0);sprintf(lcd,"UB");lcd_putstr(lcd);
lcd_gotoxy(4,2);sprintf(lcd,"test drive");
lcd_putstr(lcd);
CAT_FLAG_B=CAT_NOT_DETECTED;
BW_LOCATION[0]=LOCATION_CORRIDOR;
R4ATO3_SP_ROUTE=R4ATO3_SP_ACTIVE;
BW_PID_Init();

while(1)
{
Dynamx_MovFwd4cm(4,HEXSPD_ULTRAFAST,
IKCALC_DISABLE);
}}break;

case BW_MODE_4:
{
BW_Initial_Setup(rot_switch_mode);
lcd_display_clear();



BW_PID_Init();
while(1)
{
Hybrid_TCS3200Tracer_MainMenu();
}}break;

case BW_MODE_5:
{
BW_Initial_Setup(rot_switch_mode);
lcd_display_clear();
Dynamx_Rot_Right(HEXSPD_FAST,HEXSTEP_
VERYCLOSE,IKCALC_DISABLE);
while(1)
{
Sensor_Menu();
}}break;

case BW_MODE_6:
{
BW_Initial_Setup(rot_switch_mode);
lcd_display_clear();
BW_PID_Flame_Init();
FIRESCAN_DIRECTION=SCAN_RIGHT;
while(1)
{
Dynamx_MovFwd4cm_slow(4,HEXSPD_ULTRA
FAST,IKCALC_DISABLE);
BW_FlameFollowDemo_Dynamixel();
}}break;
}
while(1)
return 0;
}

void BW_Initialization(void)
{
//CLOCK CONFIG
SystemInit();
//SYSTICK DELAY INIT
SysTick_Init();
//BW EXTINGUISHER INIT
BW_Extinguisher_Init();
//BW HEXAPOD SERVO INIT
//BW_Servo_Initialization();

//LCD INIT
delay_ms(50);
BW_LCD_Init();
lcd_cursor_off_blink_off();
lcd_display_clear();
lcd_display_clear();
lcd_gotoxy(2,0);sprintf(lcd,"UB");lcd_putstr(lcd);
lcd_gotoxy(5,1);sprintf(lcd,"SYSTEM");
lcd_putstr(lcd);
lcd_gotoxy(1,2);sprintf(lcd,"INISIAL");
lcd_putstr(lcd);
lcd_gotoxy(1,3);sprintf(lcd,"UB2017");
lcd_putstr(lcd);
delay_ms(500);

//INERTIAL MEASUREMENT UNIT (IMU)
SENSOR
BW_CMPS11_Init();

//BW USART COMMUNICATION PROTOCOLS
//USART1_Init(9600);
//USART2_Init(9600);
USART3_Init(9600);
//UART4_Init(115200);
UART4_Init(9600);
UART5_Init(9600);
USART6_Init(9600);
```

```
Dynamixel_USART2_Init(1000000);
//   Dynamixel_USART6_Init(1000000);

//BW USER INTERFACE
Button_Init();
Buzzer_Init();
FIRE_LED_Init();
BW_LED_Interrupt_Init();
RotSwitch_Init();
RotSwitch_Sampling();

SND_ACT_LED_Init();

//INFRARED PROXIMITY SENSOR
IR_Proximity_Init();

//BW PID CONTROLLER INITIALIZATION
BW_PID_Init();
PID_Calculate_Rule_Interrupt_Init();

//Cat_Avoider_Interrupt_Init();
//BW CONTACT BUMPER
Bumper_Init();
//BW SENSOR STAT INTERRUPT
FlameSensor_Init();
//BW FLAME TRACKING PID
FlameSense_PID_Init();
```

Library global_variabel.h

```
#ifndef BW_GLOBAL_VAR_H
#define BW_GLOBAL_VAR_H
#include "main.h"
#endif
```

Library global_variabel.c

```
#include "bw_global_var.h"

unsigned int BW_CROSS_MODE = 0;

//BW START MODE
unsigned int START_MODE=0;
enum BW_START_MODE
{NON_ARBITRARY_START = 1,
ARBITRARY_START = 2};

//ROOM AND CORRIDOR VARIABLES

unsigned int BW_LOCATION[2]={1,0};
unsigned int LOCATION_CORRIDOR=0;
unsigned int LOCATION_ROOM=1;
unsigned int TANDA_FOLLOW[3]={0,0,0};
```

```
//BW TPA81 INIT
BW_TPA81_I2C_Init();
//BW SOUND ACTIVATION INIT
BW_Sound_Activation_Init();
//BW TPA SERVO INIT
PanServo_Init();

//BW DYNAMIXEL INTERRUPT INIT
//   Dynamixel_Drive_Interrupt_Init();
BW_Buzz_New(2);
lcd_display_clear();
lcd_gotoxy(2,0);sprintf(lcd,"PROJECT BW");
lcd_putstr(lcd);
lcd_gotoxy(5,1);sprintf(lcd,"SYSTEM");
lcd_putstr(lcd);
lcd_gotoxy(1,2);sprintf(lcd,"INITIALIZATION");
lcd_putstr(lcd);
lcd_gotoxy(3,3);sprintf(lcd,"COMPLETED");
lcd_putstr(lcd);
delay_ms(50);
Sendto_PC(USART1,"PROJECT BW 2017 \r");
Sendto_PC(USART1,"TEUB\r");
Sendto_PC(USART1,"SYSTEM INITIAL");
Sendto_PC(USART1,".");
Sendto_PC(USART1,".");
Sendto_PC(USART1,". \r");
Sendto_PC(USART1,"INITIALCOMPLETED \r");
}
```

```
unsigned int FOLLOW_KIRI=1;
unsigned int FOLLOW_KANAN=2;
unsigned int ADCResult;
int DATA;
int sudut_servo=0;
//USART COMM VARIABLES
char received_string[MAX_STRLEN],
buf[16],data1_lcd[16],
data2_lcd[16],tampil_adc[16];
unsigned int Ping[9],USART_Count=0;
unsigned int USARTFlame_Counter=0;
unsigned int USARTFlameDigi_Counter=0;
unsigned int USART_Bluetooth_Count=0;
unsigned int COMMAND_FLAG_A = 0x3C;
unsigned int COMMAND_FLAG_B = 0x3E;
unsigned int COMMAND_COMPLETED = 0x7C;
unsigned int COMMAND_NOT_RECEIVED =
0x3F;
char command_code[5];

/*DYNAMIXEL HEXAPOD MOVING
VARIABLES
 */
```

```
//NEW VAR
unsigned int GERAK=0;
 //COXA,FEMUR,TIBIA
int SUDUT_STATIC[18]=
{  90 ,  18 , 105 ,  //FR
   90 ,  18 , 105 ,  //ML
   90 ,  20 , 105 ,  //RR
     90 ,  18 , 105 ,          //FL
     90 ,  18 , 105 ,          //MR
     90 ,  20 , 105 };         //RL

 //COXA,FEMUR,TIBIA
int SUDUT_STATIC_BWD[18]=
{  90 ,  15 , 105 ,  //FR
   90 ,  15 , 105 ,  //ML
   90 ,  15 , 105 ,  //RR

  90 ,  15 , 105 ,   //FL
  90 ,  15 , 105 ,   //MR
  90 ,  15 , 105 };  //RL

int SUDUT_ANGKAT=15;

// MOV FWD ZCOXA = 30 and Z TIBIA = 30
int IKFWD_COXA_R[3]={11,31,32};
int IKFWD_TIBIA_R[3]={18,18,48};
int IKFWD_COXA_L[3]={11,31,32};
int IKFWD_TIBIA_L[3]={18,18,48};
int IKFWD_COXA_R1[3]={8,22,17};
int IKFWD_TIBIA_R1[3]={15,15,26};
int IKFWD_COXA_L1[3]={8,22,17};
int IKFWD_TIBIA_L1[3]={15,15,26};
int IKFWD_COXA_R2[3]={5,11,7};
int IKFWD_TIBIA_R2[3]={9,1,11};
int IKFWD_COXA_L2[3]={5,11,7};
int IKFWD_TIBIA_L2[3]={9,1,11};
int IKFWD_COXA_R3[3]={3,6,3};
int IKFWD_TIBIA_R3[3]={5,0,5};
int IKFWD_COXA_L3[3]={3,6,3};
int IKFWD_TIBIA_L3[3]={5,0,5};
int IKROT_COXA[2]={0,0};
int IKROT_COXA1[2]={0,0};
int IKROT_COXA2[2]={0,0};

int IKCRVR_COXA_R2[3]={3,6,3};
int IKCRVR_TIBIA_R2[3]={5,0,5};
int IKCRVR_COXA_L2[3]={8,22,17};
int IKCRVR_TIBIA_L2[3]={15,5,26};

int IKRPID_COXA_R2[3]={5,16,14};
int IKRPID_TIBIA_R2[3]={10,5,21};

int IKCRVR_COXA_R[3]={3,6,3};
int IKCRVR_TIBIA_R[3]={5,0,5};
int IKCRVR_COXA_L[3]={12,35,42};
int IKCRVR_TIBIA_L[3]={17,7,36};

int IKRPID_COXA_R[3]={9,29,39};
int IKRPID_TIBIA_R[3]={12,7,31};

int IKSLR_COXA[3]={16,0,23};
int IKSLR_TIBIA[4]={7,18,30,17};

int IKSLR_COXA1[3]={13,0,17};
int IKSLR_TIBIA1[4]={6,14,21,12};

int IKSLR_COXA2[3]={9,0,11};
int IKSLR_TIBIA2[4]={5,10,13,7};

int IKCRVL_COXA_R[3]={12,35,42};
int IKCRVL_TIBIA_R[3]={17,7,36};
int IKCRVL_COXA_L[3]={3,6,3};
int IKCRVL_TIBIA_L[3]={5,0,5};

int IKLPID_COXA_L[3]={9,29,39};
int IKLPID_TIBIA_L[3]={12,7,31};

int IKCRVL_COXA_R2[3]={8,22,17};
int IKCRVL_TIBIA_R2[3]={15,5,26};
int IKCRVL_COXA_L2[3]={3,6,3};
int IKCRVL_TIBIA_L2[3]={5,0,5};

int IKLPID_COXA_L2[3]={5,16,14};
int IKLPID_TIBIA_L2[3]={10,5,21};

int IKSLL_COXA[3]={16,0,23};
int IKSLL_TIBIA[4]={7,18,30,17};

int IKSLL_COXA1[3]={13,0,17};
int IKSLL_TIBIA1[4]={6,14,21,12};

int IKSLL_COXA2[3]={9,0,11};
int IKSLL_TIBIA2[4]={5,10,13,7};

int IKBWD_COXA_R[3]={7,17,12};
int IKBWD_TIBIA_R[3]={12,3,18};
int IKBWD_COXA_L[3]={7,17,12};
int IKBWD_TIBIA_L[3]={12,3,18};

int IKCRVRR_COXA_L[3]={3,6,3};
int IKCRVRR_TIBIA_L[3]={5,0,5};
int IKCRVRR_COXA_R[3]={7,17,12};
int IKCRVRR_TIBIA_R[3]={12,3,18};

int IKPIDR_COXA_L[3]={4,11,9};
int IKPIDR_TIBIA_L[3]={7,3,13};

int IKCRVLR_COXA_L[3]={7,17,12};
int IKCRVLR_TIBIA_L[3]={12,3,18};
int IKCRVLR_COXA_R[3]={3,6,3};
int IKCRVLR_TIBIA_R[3]={5,0,5};

int IKPIDR_COXA_R[3]={4,11,9};
```

```
int IKPIDR_TIBIA_R[3]={7,3,13};
uint32_t Count;
uint32_t nCount;

int sudut_target[18],
    sudut_awal[18],
    x=0,
    z_A[18],
    pembagi,
    y_A[18],
    sudut_tahap1[18],
    sudut_tahap2[18];

int leg;
int ww[6];
float beta[6];
float leg_next[6];
float A_body;      // translasi pusat badan
float A_leg[6];    // translasi setiap pangkal kaki
float P_body;      // jarak titik pusat robot
int epsilon_body; // besar sudut vektor
float phi[6];
float lambpsi[6];  //lambda / psi temporal
float P_leg[6];    // jarak titik pusat kaki ke titik
imajiner putar
float epsilon[6];  // epsilon tiap N

unsigned int sudut[18]=
{
4915,4915,4915,4915,4915,4915,
4915,4915,4915,4915,4915,4915,
4915,4915,4915,4915,4915,4915

//3615 == 1 ms   4915 == 1,5 ms
//6550 == 2 ms
};

unsigned char
y,a,b,c,d,v,index_bantu,index_servo3,index_servo2;
uint32_t PrescalerValue;

const float A_const[18] =
{
26.68,26.68,26.68,26.68,26.68,26.68,
26.68,26.68,26.68,26.68,26.68,26.68,
26.68,26.68,26.68,26.68,26.68,26.68
};

const int B_const[18] =
{
2681,2681,2681,2681,2681,2681,
2681,2681,2681,2681,2681,2681,
2681,2681,2681,2681,2681,2681
};

 float correct[18]      =
{
   24,100,13,      //FRONT LEFT LEG
   0,98,6,         //REAR LEFT LEG
   35,53,14,       //FRONT RIGHT LEG
   30,115,13,      //MIDDLE RIGHT LEG
   38,94,6,        //REAR RIGHT LEG
};

const float leg_1[6] =
{7.27,5.06,7.27,7.27,5.06,7.27};              //L1
leg_norm[6]={8,8,8,8,8,8};                    //L2
const float alpha[6]={30,90,150,30,90,150}; //alpha

float c1,c2;
float Lsem,c3,c4;
int i;
float temp_var,temp_var2;

/*CALCULATING DISTANCE*/
double PI =
3.14159265358979323846264338327 95;
double GRAVITY = 9806.65;

float dummy_correct[18];
unsigned int TEST = 0x5A;

unsigned int RETURN_COUNTER=0;

/*FOLLOW CARPET FLAG*/

unsigned int FOLLOW_CARPET_FLAG=0;
unsigned int FOLLOW_CARPET_ENABLED=1;
unsigned int FOLLOW_CARPET_DISABLED=0;
unsigned int
FOLLOW_CARPET_ESCAPE_TRAP=2;

int Arm_Angle[3]={0,0,0};
int Arm_Correction[3]= {0,0,0};

uint32_t ARM_CCR1_Val, ARM_CCR2_Val,
ARM_CCR3_Val;

/*I2C MULTIBYTE READ */

uint8_t   I2C1_DATA[10];
uint8_t   I2C2_DATA[10];
uint8_t    I2C3_DATA[10];

/*FOLLOW TRACER FLAG VARIABLES*/

unsigned int FOLLOWTRACER_FLAG=0;
unsigned int FOLLOWTRACER_ENABLE=1;
unsigned int FOLLOWTRACER_DISABLE=0;
```

```
/*EXIT FLAG*/

unsigned int EXIT_FLAG;
unsigned int EXIT_ACTIVE        =1;
unsigned int EXIT_INACTIVE      =0;

unsigned int FOLLOW_FLAG;
unsigned int FLAME                      =5;
unsigned int KANAN_BELAKANG             =4;
unsigned int KIRI_BELAKANG              =3;
unsigned int KANAN                      =2;
unsigned int KIRI                       =1;
unsigned int KOSONG                     =0;
unsigned int FOLLOW_BREAK;
unsigned int FOLLOW_CALC;

/*DEFLECT WALL OVERFLOW COUNTER*/
int RIGHT_OVERFLOW_COUNTER      =0;
int LEFT_OVERFLOW_COUNTER       =0;
```

Library ping_receiver.h

```
#ifndef BW_PING_RECEIVER_H
#define BW_PING_RECEIVER_H
#include "main.h"

//USART COMM MACRO
#define ON_Ping 0x01
#define OFF_Ping 0x02
#define TO 0x04
#define USART_ACTIVE_MODE 1
#define MAX_STRLEN 10000

//USART COMM VARIABLES
extern char received_string[MAX_STRLEN];
extern char buf[16];
extern char data1_lcd[16];
extern char data2_lcd[16];
extern char tampil_adc[16];
extern unsigned int Ping[9];
extern unsigned int USART_Count;

extern unsigned int Ping[9],USART_Count;
extern unsigned int COMMAND_FLAG_A;
extern unsigned int COMMAND_FLAG_B;
extern unsigned int COMMAND_COMPLETED;
extern unsigned int
COMMAND_NOT_RECEIVED;

#define PING_LEFT 0
#define PING_RIGHT 1
#define PING_FRONT 2
//#define PING_REAR 3
```

```
/*BW TURNAROUND NAVIGATION*/

unsigned int TURN_LEFT  = 1;
unsigned int TURN_RIGHT = 2;

int DynamxCorrection[18]=
{

0,0,0, //Front Left Leg
0,0,0, //Middle Left Leg
0,0,0, //Rear Left Leg
0,0,0, //Front Right Leg
0,0,0, //Middle Right Leg
0,0,0, //Rear Right Leg
};

float Dynamx_LinkFemur          = 4;
float Dynamx_LinkTibia          = 8;
```

```
#define PING_ASKEW_LEFT 3
#define PING_ASKEW_RIGHT 4
#define PING_REAR 8
#define PING_REAR_LEFT 6
//#define PING_REAR_LEFT 5
#define PING_REAR_RIGHT 7

//SHARP IR RANGEFINDER

extern unsigned int SHARP[];
#define SHARP_FRONT_R 0
#define SHARP_FRONT_L 1

//USART COMM FUNCTION PROTOTYPE

void USART3_Init(uint32_t baudrate);

void usart_puts(char *data);

/*USART PING*/

void Display_Ping_Status (void);
void Send_Ping_Status (void);
void Display_Sharp_Status (void);
void USART3_Init(uint32_t baudrate); //TO
RANGEFINDER uCU;

#endif
```

Library ping_receiver.c

```
#include "bw_ping_receiver.h"

void USART3_Init(uint32_t baudrate) //TO
RANGEFINDER uCU
{
RCC_APB1PeriphClockCmd(RCC_APB1Periph_
USART3, ENABLE);
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_
GPIOD, ENABLE);
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8 |
GPIO_Pin_9;
GPIO_InitStructure.GPIO_Mode=GPIO_Mode_AF
    ;
GPIO_InitStructure.GPIO_Speed =
GPIO_Speed_50MHz;

GPIO_InitStructure.GPIO_OType =
GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd =
GPIO_PuPd_UP;
GPIO_Init(GPIOD, &GPIO_InitStructure);
GPIO_PinAFConfig(GPIOD, GPIO_PinSource8,
GPIO_AF_USART3); //
GPIO_PinAFConfig(GPIOD, GPIO_PinSource9,
GPIO_AF_USART3);

USART_InitStructure.USART_BaudRate =
baudrate;
USART_InitStructure.USART_WordLength =
USART_WordLength_8b
USART_InitStructure.USART_StopBits =
USART_StopBits_1;
USART_InitStructure.USART_Parity =
USART_Parity_No;
USART_InitStructure.USART_HardwareFlowCont
rol = USART_HardwareFlowControl_None;
USART_InitStructure.USART_Mode =
USART_Mode_Tx | USART_Mode_Rx;
USART_Init(USART3, &USART_InitStructure);

USART_ITConfig(USART3, USART_IT_RXNE,
ENABLE);
NVIC_InitStructure.NVIC_IRQChannel =
USART3_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemption
Priority = 0
NVIC_InitStructure.NVIC_IRQChannelSubPriority
= 1;
NVIC_InitStructure.NVIC_IRQChannelCmd =
ENABLE;
NVIC_Init(&NVIC_InitStructure);

USART_Cmd(USART3, ENABLE);
}

void USART3_PutChar(char c)
{
uint8_t ch;
ch = c;
USART_SendData(USART3, (uint8_t) ch);
while (USART_GetFlagStatus(USART3,
USART_FLAG_TC) == RESET){}
}

void usart_puts(char *data)
{
if(USART_ACTIVE_MODE&&1==1)
 {
   int i=0;int n = strlen(data);
   for(i=0;i<n;i++)
   {USART3_PutChar(data[i]);}
 }
}

void USART3_IRQHandler(void)
{
if( USART_GetITStatus(USART3,
USART_IT_RXNE) )
{
// the character from the USART3 data register is saved in t
char t = USART3->DR;
char static last_data;
switch(USART_Count)
 {
case  1 : Ping[2]=USART3->DR;
USART_Count+=1; break;
case  2 : Ping[1]=USART3->DR;
USART_Count+=1; break;
case  3 : Ping[0]=USART3->DR;
USART_Count+=1; break;
case  4 : SHARP[0]=USART3->DR;
USART_Count+=1; break;
case  5 : Ping[3]=USART3->DR;
USART_Count+=1; break;
case  6 : Ping[4]=USART3->DR;
USART_Count+=1; break;
case  7 : Ping[8]=USART3->DR;
USART_Count+=1; break;
case  8 : SHARP[1]=USART3->DR;
USART_Count+=1; break;
case  9 : Ping[6]=USART3->DR;
USART_Count+=1; break;
case  10: Ping[7]=USART3->DR;
USART_Count=811; break;
}
if(t==COMMAND_FLAG_B &&
last_data==COMMAND_FLAG_A)
USART_Count=1;
last_data=t;
}
}
```

```
void Display_Ping_Status (void)
{lcd_display_clear();
lcd_gotoxy(0,0);sprintf(lcd,"RF Monitor ");
lcd_putstr(lcd);
lcd_gotoxy(0,1);sprintf(lcd,"AL%d",
Ping[PING_ASKEW_LEFT]);lcd_putstr(lcd);
lcd_gotoxy(6,1);sprintf(lcd,"F%d",
Ping[PING_FRONT]);lcd_putstr(lcd);
lcd_gotoxy(11,1);sprintf(lcd,"AR%d",
Ping[PING_ASKEW_RIGHT]);lcd_putstr(lcd);
lcd_gotoxy(2,2);sprintf(lcd,"L%d",
Ping[PING_LEFT]);lcd_putstr(lcd);
lcd_gotoxy(9,2);sprintf(lcd,"R%d",
Ping[PING_RIGHT]);lcd_putstr(lcd);
lcd_gotoxy(0,3);sprintf(lcd,"RL%d",
Ping[PING_REAR_LEFT]);lcd_putstr(lcd);
lcd_gotoxy(11,3);sprintf(lcd,"RR%d",
Ping[PING_REAR_RIGHT]);lcd_putstr(lcd);
lcd_gotoxy(6,3);sprintf(lcd,"B%d",
Ping[PING_REAR]);lcd_putstr(lcd);
delay_ms(50);
}
```

```
void Send_Ping_Status (void)
{
Sendto_PC(USART1,"BW Ping Status \r");
Sendto_PC(USART1," Ping Front: %d\\r",
Ping[PING_FRONT]);

Sendto_PC(USART1,"Ping Askew Left: %d Ping
Askew Right: %d\r",
Ping[PING_ASKEW_LEFT],
Ping[PING_ASKEW_RIGHT]);

Sendto_PC(USART1,"Ping Left: %d Ping Right:
%d \r",
Ping[PING_LEFT],
Ping[PING_RIGHT]);\r",
Ping[PING_REAR]);
delay_ms(60);
}
```

Library pid_controller.h

```
#ifndef BW_PID_CONTROLLER_H
#define BW_PID_CONTROLLER_H
#include "main.h"

GPIO_InitTypeDef            GPIO_InitStructure;
TIM_OCInitTypeDef        TIM_OCInitStructure;
TIM_TimeBaseInitTypeDef
TIM_TimeBaseStructure;
NVIC_InitTypeDef         NVIC_InitStructure;

#define RIGHT 0
#define LEFT  1

#define NB  0
#define ZB  1
#define PB  2

float centre;
float umin;

/*PID CONTROLLER GLOBAL VARIABLES*/

struct PID_t
{
    float P[3];
    float I[3];
    float D[3];
    float Kp;
    float Ki;
    float Kd;

    float KpF;
    float KiF;
    float KdF;

    float Ts;

    float error[3];
    float Error[3];
    float derror[3];
    float serror[3];

    float error_fuzzy[3];
    float derror_fuzzy[3];
    float serror_fuzzy[3];

    float pid_value[3];
    float set_point;
    float set_point_upper;
    float set_point_lower;
}PID_F_R,PID_F_L;

float hasil;
extern unsigned int Ping[9];
extern unsigned int usart_count;
extern unsigned int FIRE;
extern unsigned int CAT_DETECTOR;
extern unsigned int CAT_ACTIVATED;
```

```
extern unsigned int CAT_DIACTIVATED;
void BW_PID_Interrupt_Init(void);
void BW_PID_Init(void);
void PID_Calc_RightRule(void);
void PID_Calc_LeftRule(void);
void PID_Calculate_Rule_Interrupt_Init(void);
float Error[3];
float dError[3];
float Num, Denum;
float MIN_Value(float data1, float data2);
float MAX_Value(float data1, float data2);
float triangle(float value,float x0, float x1, float x2);
float grade(float value, float x0, float x1);
float reverse_grade(float value,float x0, float x1);
float rule_base_Kp();
float rule_base_Ki();
float rule_base_Kd();
float fuzzy_Kp();
float fuzzy_Ki();
float fuzzy_Kd();
#endif
```

Library pid_controller.c

```
#include "bw_pid_controller.h"

int count=1;

void BW_PID_Init(void)
{
CAT_DETECTOR=CAT_DIACTIVATED;
//BW PID WALL FOLLOWING RIGHT RULE

PID_F_R.P[0]=0;
PID_F_R.P[1]=0;
PID_F_R.P[2]=0;
PID_F_R.I[0]=0;
PID_F_R.I[1]=0;
PID_F_R.I[2]=0;
PID_F_R.D[0]=0;
PID_F_R.D[1]=0;
PID_F_R.D[2]=0;
PID_F_R.Kp=fuzzy_Kp();
PID_F_R.Ki= PID_F_R.Kp /fuzzy_Ki();
PID_F_R.Kd= PID_F_R.Kp *fuzzy_Kd();
PID_F_R.Ts=0.2;
PID_F_R.set_point_upper=17;
PID_F_R.set_point_lower=16;
PID_F_R.set_point=16;
PID_F_R.error[0]=0;
PID_F_R.error[1]=0;
PID_F_R.error[2]=0;
PID_F_R.pid_value[0]=0;
PID_F_R.pid_value[1]=0;
PID_F_R.pid_value[2]=0;

PID_F_L.P[1]=0;
PID_F_L.P[2]=0;
PID_F_L.I[0]=0;
PID_F_L.I[1]=0;
PID_F_L.I[2]=0;
PID_F_L.D[0]=0;
PID_F_L.D[1]=0;
PID_F_L.D[2]=0;
PID_F_L.Kp=fuzzy_Kp();
PID_F_L.Ki= PID_F_L.Kp /fuzzy_Ki();
PID_F_L.Kd= PID_F_L.Kp *fuzzy_Kd();
PID_F_L.Ts=0.2;
PID_F_L.error[0]=0;
PID_F_L.error[1]=0;
PID_F_L.error[2]=0;
PID_F_L.pid_value[0]=0;
PID_F_L.pid_value[1]=0;
PID_F_L.pid_value[2]=0;
PID_F_L.set_point=16;
PID_F_L.set_point_upper=17;
PID_F_L.set_point_lower=16;

void PID_Calc_RightRule(void)
{
PID_F_R.pid_value[2]=PID_F_R.pid_value[1];
PID_F_R.pid_value[1]=PID_F_R.pid_value[0];
PID_F_R.error[2]=PID_F_R.error[1];
PID_F_R.error[1]=PID_F_R.error[0];
if (Ping[PING_ASKEW_RIGHT]==
PID_F_R.set_point) {PID_F_R.error[0]=0;}

else if (Ping[PING_ASKEW_RIGHT] >
PID_F_R.set_point)
{
PID_F_R.error[0]=
Ping[PING_ASKEW_RIGHT]-PID_F_R.set_point;
}
else if (Ping[PING_ASKEW_RIGHT] <
PID_F_R.set_point)
{
PID_F_R.error[0]=
PID_F_R.set_point-Ping[PING_ASKEW_RIGHT];
}

//Proportional Controller
PID_F_R.P[2]= PID_F_R.error[0];
PID_F_R.P[1]= PID_F_R.Kp;
PID_F_R.P[0]= PID_F_R.P[1]*PID_F_R.P[2];

//Integral Controller
PID_F_R.I[2]= PID_F_R.Ki*PID_F_R.Ts/2;
PID_F_R.I[1]=
PID_F_R.error[0]+(2*PID_F_R.error[1])+
PID_F_R.error[2];
PID_F_R.I[0]= PID_F_R.I[2]*PID_F_R.I[1];
```

```
//Derivative Controller
PID_F_R.D[2]= 2*PID_F_R.Kd/PID_F_R.Ts;
PID_F_R.D[1]= PID_F_R.error[0]-
(2*PID_F_R.error[1])+PID_F_R.error[2];
PID_F_R.D[0]= PID_F_R.D[2]*PID_F_R.D[1];

//PID
PID_F_R.pid_value[0]= PID_F_R.pid_value[2] +
PID_F_R.P[0] + PID_F_R.I[0] + PID_F_R.D[0];
}

void PID_Calc_LeftRule(void)
{
PID_F_L.pid_value[2]=PID_F_L.pid_value[1];
PID_F_L.pid_value[1]=PID_F_L.pid_value[0];

PID_F_L.error[2]=PID_F_L.error[1];
PID_F_L.error[1]=PID_F_L.error[0];

if
(Ping[PING_ASKEW_LEFT]==PID_F_L.set_point
) {PID_F_L.error[0]=0;}

else if (Ping[PING_ASKEW_LEFT] >
PID_F_L.set_point)
{
PID_F_L.error[0] = Ping[PING_ASKEW_LEFT] -
PID_F_L.set_point;
}
else if (Ping[PING_ASKEW_LEFT] <
PID_F_L.set_point)
{
PID_F_L.error[0] = PID_F_L.set_point -
Ping[PING_ASKEW_LEFT];
}
//Proportional Controller
PID_F_L.P[2]= PID_F_L.error[0];
PID_F_L.P[1]= PID_F_L.Kp;
PID_F_L.P[0]= PID_F_L.P[1]*PID_F_L.P[2];

//Integral Controller
PID_F_L.I[2]= PID_F_L.Ki*PID_F_L.Ts/2;
PID_F_L.I[1]=
PID_F_L.error[0]+(2*PID_F_L.error[1])+
PID_F_L.error[2];
PID_F_L.I[0]= PID_F_L.I[2]*PID_F_L.I[1];

//Derivative Controller
PID_F_L.D[2]= 2*PID_F_L.Kd/PID_F_L.Ts;
PID_F_L.D[1]= PID_F_L.error[0]-
(2*PID_F_L.error[1])+PID_F_L.error[2];
PID_F_L.D[0]= PID_F_L.D[2]*PID_F_L.D[1];

PID_F_L.pid_value[0]= PID_F_L.pid_value[2] +
PID_F_L.P[0] + PID_F_L.I[0] + PID_F_L.D[0];
}

void PID_Calculate_Rule_Interrupt_Init(void)
{
RCC_APB1PeriphClockCmd(RCC_APB1Periph_T
IM3, ENABLE);
TIM_TimeBaseInitTypeDef
TIM_TimeBaseStructure;
TIM_TimeBaseStructure.TIM_Prescaler = 839;
TIM_TimeBaseStructure.TIM_CounterMode =
TIM_CounterMode_Up;
TIM_TimeBaseStructure.TIM_Period = 50;
TIM_TimeBaseStructure.TIM_ClockDivision =
TIM_CKD_DIV1;
TIM_TimeBaseStructure.TIM_RepetitionCounter=
0;
TIM_TimeBaseInit(TIM3,
&TIM_TimeBaseStructure);
TIM_Cmd(TIM3, ENABLE);
TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);
NVIC_InitTypeDef NVIC_InitStructure;
NVIC_InitStructure.NVIC_IRQChannel =
TIM3_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemption
Priority = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority
= 0;
NVIC_InitStructure.NVIC_IRQChannelCmd =
ENABLE;
NVIC_Init(&NVIC_InitStructure);
}

void TIM3_IRQHandler(void)
{
float MIN_Value(float data1, float data2);
float MAX_Value(float data1, float data2);

if (TIM_GetITStatus(TIM3, TIM_IT_Update) !=
RESET)
{
if(FOLLOW_CALC==KANAN)
{
PID_Calc_RightRule();
fuzzy_Kp1();fuzzy_Ki1();fuzzy_Kd1();
}
 else if(FOLLOW_CALC==KIRI)
 {
 PID_Calc_LeftRule();
 fuzzy_Kp2();fuzzy_Ki2();fuzzy_Kd2();}
}

float MIN_Value(float data1, float data2)
{
float output = 0 ;
if (data1 < data2) output=data1 ;
else output=data2 ;
return output;
}
```

```
float MAX_Value(float data1, float data2)
{
 float output = 0 ;
 if (data1 < data2) output=data2 ;
 else output=data1 ;
 return output;
}

// ----FUZZY- MAMDANI RULE---------//
float triangle(float value,float x0, float x1, float x2)
{
float result = 0;
float x;
x = value;  // x = alamat dari PING yang diakses
if ((x <=x0)||(x>=x2)) {result=0;}
else if(x==x1) {result=1;}
else if((x>=x0)&&(x<x1))
{result=((x-x0)/(x1-x0));}
else{result = (((-x)+x2)/(x2-x1));}
return result;
}

float grade(float value, float x0, float x1)
{
float result = 0;
float x;
x = value;
if (x<=x0){result=0;}
else if(x>=x1){result=1;}
else{result = (x-x0)/(x1-x0);}
return result;
}

float reverse_grade(float value,float x0, float x1)
{
float result = 0;
float x;
x = value;
if (x<=x0) {result=1; }
else if(x>=x1){result=0;}
else{result =(-x+x1)/(x1-x0);}
return result;
}

void fuzzyfikasi1()
{
 Error[NB] = reverse_grade
 (PID_F_R.error[0],-10,0);
 Error[ZB] = triangle
 (PID_F_R.error[0],-10,0,10);
 Error[PB] = grade
 (PID_F_R.error[0],0,10);

dError[NB] = reverse_grade
(PID_F_R.derror[0],-10,0);
dError[ZB] = triangle
(PID_F_R.derror[0],-10,0,10);
dError[PB] = grade
(PID_F_R.derror[0],0,10);
}

void fuzzyfikasi2()
{
Error[NB] = reverse_grade
(PID_F_R.error[0],-10,0);
Error[ZB] = triangle(PID_F_R.error[0],-10,0,10);
Error[PB] = grade(PID_F_R.error[0],0,10);

dError[NB] = reverse_grade
(PID_F_L.derror[0],-10,0);
dError[ZB] = triangle
(PID_F_L.derror[0],-10,0,10);
dError[PB] = grade
(PID_F_L.derror[0],0,10);
}

float rule_base_Kp()
{

char x,y;
float Gain[3];
float Gain_out = 0;

for (x=NB;x<=PB;x++)
{

for (y=NB;y<=PB;y++)
{
umax = Max_Value(Error[x],dError[y]);

if ((x==NB)&&(y==NB)) Gain[0]       = -1*umax;
else if ((x==NB)&&(y==ZB)) Gain[0]=0.5*umax;
else if ((x==NB)&&(y==PB)) Gain[0]= -0.5*umax;
else if ((x==ZB)&&(y==NB)) Gain[0]=-0.5*umax;
else if ((x==ZB)&&(y==ZB)) Gain[0] = 0*umax;
else if ((x==ZB)&&(y==PB)) Gain[0] = 0.5*umax;
else if ((x==PB)&&(y==NB)) Gain[0] = 0*umax;
else if ((x==PB)&&(y==ZB)) Gain[0] = 0.5*umax;
else if ((x==PB)&&(y==PB)) Gain[0]   = 1*umax;
}
}

Gain[2]=Gain[1];
Gain[1]=Gain[0];
Gain_out = (Gain[0]+Gain[1]+Gain[2])/3*umax;

return Gain_out;

}
```

```
float rule_base_Ki()
{

char x,y;
float Gain[3];
float Gain_out = 0;

for (x=NB;x<=PB;x++)
{

for (y=NB;y<=PB;y++)
{

umax = Max_Value(Error[x],dError[y]);

if     ((x==NB)&&(y==NB)) Gain[0]    = umax;
else if ((x==NB)&&(y==ZB)) Gain[0]   = umax;
else if ((x==NB)&&(y==PB)) Gain[0]   = umax;
else if ((x==ZB)&&(y==NB)) Gain[0]   = umax;
else if ((x==ZB)&&(y==ZB)) Gain[0]   = umax;
else if ((x==ZB)&&(y==PB)) Gain[0]   = umax;
else if ((x==PB)&&(y==NB)) Gain[0]   = umax;
else if ((x==PB)&&(y==ZB)) Gain[0]   = umax;
else if ((x==PB)&&(y==PB)) Gain[0]   = umax;

}

}

Gain[2]=Gain[1];
Gain[1]=Gain[0];
Gain_out = (Gain[0]+Gain[1]+Gain[2])/3*umax;

 return Gain_out;

}
```

Library Algortihm .h

```
#ifndef BW2016_ALGORITHM_H
#define BW2016_ALGORITHM_H\
#include "main.h"

extern unsigned int Ping[9];
extern unsigned int usart_count;

extern unsigned int TANDA_FOLLOW[];
extern unsigned int FOLLOW_KIRI;
extern unsigned int FOLLOW_KANAN;
extern int INFRARED[];
extern unsigned int BUMPER[];
extern unsigned int CAT_FOUND[];

#define CAT_PREVIOUS 1
#define CAT_PRESENT  0
```

```
float rule_base_Kd()
{
char x,y;
float Gain[3];
float Gain_out = 0;

for (x=NB;x<=PB;x++)
{
for (y=NB;y<=PB;y++)
{
umax = Max_Value(Error[x],dError[y]);

if ((x==NB)&&(y==NB)) Gain[0]       = -1*umax;
else if ((x==NB)&&(y==ZB)) Gain[0]=0.5*umax;
else if ((x==NB)&&(y==PB)) Gain[0]= -0.5*umax;
else if ((x==ZB)&&(y==NB)) Gain[0]=-0.5*umax;
else if ((x==ZB)&&(y==ZB)) Gain[0] = 0*umax;
else if ((x==ZB)&&(y==PB)) Gain[0] = 0.5*umax;
else if ((x==PB)&&(y==NB)) Gain[0] = 0*umax;
else if ((x==PB)&&(y==ZB)) Gain[0] = 0.5*umax;
else if ((x==PB)&&(y==PB)) Gain[0]   = 1*umax;
}}
Gain[2]=Gain[1];Gain[1]=Gain[0];
Gain_out = (Gain[0]+Gain[1]+Gain[2])/3*umax;

return Gain_out;
}

//-----------------SAMPING DEPAN----------------//
float fuzzy_Kp()
{fuzzyfikasi();return rule_base_Kp();}
float fuzzy_Ki()
{fuzzyfikasi();return rule_base_Ki();}
float fuzzy_Kd()
{fuzzyfikasi();return rule_base_Kd();}
```

```
extern unsigned int CAT_SHIFT_FOLLOW;
extern unsigned int SHIFT_FOLLOW;
extern unsigned int NO_SHIFT;

extern unsigned int CAT_FLAG;
extern unsigned int CAT_FLAG_A;
extern unsigned int CAT_FLAG_B;
extern unsigned int CAT_FLAG_C;
extern unsigned int CAT_DETECTED;
extern unsigned int CAT_NOT_DETECTED;
extern unsigned int CAT_DETECTOR;
extern unsigned int CAT_ACTIVATED;
extern unsigned int CAT_DIACTIVATED;

extern unsigned int RETURN_COUNTER;

void BW_FollowCounter_Right(int limit);
void PID_BW_FollowCounter_Right(void);
```

```
void BW_FollowCounter_Left(int limit);
void PID_BW_FollowCounter_Left(void);

extern unsigned int EXIT_FLAG;
extern unsigned int EXIT_ACTIVE;
extern unsigned int EXIT_INACTIVE;
extern unsigned int FOLLOW_FLAG;
extern unsigned int FLAME;
extern unsigned int KANAN_BELAKANG;
extern unsigned int KIRI_BELAKANG;
extern unsigned int KANAN;
extern unsigned int KIRI;
extern unsigned int KOSONG;
extern unsigned int FOLLOW_BREAK;
extern unsigned int FOLLOW_CALC;

#endif
```

Library Algortihm .c

```
#include "bw2017_algorithm.h"

void BW_FollowCounter_Right(int limit)
{
int counter;
FOLLOW_CALC=KANAN;
CAT_FLAG = CAT_NOT_DETECTED;
for(counter=1;counter<=limit;counter++)
{
    Cat_Avoider();
    Furniture_Avoider();
if (Ping[PING_FRONT]<=15)
{
    Dynamx_Rot_Left(HEXSPD_FAST,
    HEXSTEP_FAR,IKCALC_DISABLE);
}
else{PID_BW_FollowCounter_Right();}
if(FIRE==FIRE_ON){Bumper_Follow();}
else{Bumper_Action_Right();}

}

FOLLOW_CALC=KOSONG;

}

void PID_BW_FollowCounter_Right(void)
{

//int limit= *COMMAND_LOOP;
int OVERRIDE=0;

static int WINDUP_RIGHT_COUNTER=0;
static int WINDUP_LEFT_COUNTER=0;

while(OVERRIDE==0)
{
if(OVERRIDE==0)
{
//KONDISI ROBOT SESUAI
if ( (Ping[PING_ASKEW_RIGHT]<=
PID_F_R.set_point_upper) &&
(Ping[PING_ASKEW_RIGHT]>=
PID_F_R.set_point_lower) )
{
if (Ping[PING_FRONT]<=15)
{
Dynamx_Rot_Left(HEXSPD_ULTRAFAST,
HEXSTEP_FAR,IKCALC_DISABLE);
Dynamx_Rot_Left(HEXSPD_FAST,
HEXSTEP_FAR,IKCALC_DISABLE);
}
if(OVERRIDE==1){break;}
if (OVERRIDE==0)
{
    RIGHT_OVERFLOW_COUNTER=0;
    LEFT_OVERFLOW_COUNTER=0;
Sendto_PC(USART1,"BW FORWARD \r");
Dynamx_MovFwd4cm(4, HEXSPD_MEDFAST,
IKCALC_DISABLE);
OVERRIDE=1;
}
}

//KONDISI ROBOT JAUH DARI DINDING
else if ( Ping[PING_ASKEW_RIGHT]>
PID_F_R.set_point )
{
   //RESET WINDUP LEFT COUNTER
   WINDUP_LEFT_COUNTER=0;
    if (Ping[PING_FRONT]<=15)
{
Dynamx_Rot_Left(HEXSPD_ULTRAFAST,
HEXSTEP_FAR,IKCALC_DISABLE);
Dynamx_Rot_Left(HEXSPD_FAST,
HEXSTEP_FAR,IKCALC_DISABLE);
}

if(OVERRIDE==1){break;}
if (OVERRIDE==0)
{
if(WINDUP_RIGHT_COUNTER>=25)
{
BW_Buzz(5);
lcd_display_clear();
lcd_gotoxy(0,0);sprintf(lcd,"FOLLOW
OVERFLOW R");lcd_putstr(lcd);
delay_ms(100);
```

```
while(Ping[PING_FRONT]>=15)
{
Dynamx_MovFwd4cm(4,HEXSPD_MEDFAST,
IKCALC_DISABLE);
}

while(Ping[PING_RIGHT]>=16)
{
Dynamx_Rot_Left(HEXSPD_ULTRAFAST,
HEXSTEP_FAR,IKCALC_DISABLE);
Dynamx_Rot_Left(HEXSPD_FAST,
HEXSTEP_FAR,IKCALC_DISABLE);
}
//RESET WINDUP RIGHT COUNTER
WINDUP_RIGHT_COUNTER=0;
//REINIT PID
BW_PID_Init();
}

else
{
Furniture_Avoider();
RIGHT_OVERFLOW_COUNTER=0;
LEFT_OVERFLOW_COUNTER=0;
Dynamx_CurveRight_PID(HEXSPD_MEDFAST,
HEXSTEP_FAR, &PID_F_R.pid_value[0],
IKCALC_DISABLE);
}
OVERRIDE=1;
}
}

//KONDISI ROBOT DEKAT DENGAN DINDING
else if (
Ping[PING_ASKEW_RIGHT]<PID_F_R.set_point
)
{
//RESET WINDUP LEFT COUNTER
WINDUP_RIGHT_COUNTER=0;

if (Ping[PING_FRONT]<=15)
{
Dynamx_Rot_Left(HEXSPD_ULTRAFAST,
HEXSTEP_FAR,IKCALC_DISABLE);
Dynamx_Rot_Left(HEXSPD_FAST,
HEXSTEP_FAR,IKCALC_DISABLE);
}
if(OVERRIDE==1){break;}
if (OVERRIDE==0)
{

if(WINDUP_LEFT_COUNTER>=25)
{
BW_Buzz(5);
lcd_display_clear();
lcd_gotoxy(0,0);
sprintf(lcd,"FOLLOW OVERFLOW L");
lcd_putstr(lcd);
delay_ms(100);

while(Ping[PING_FRONT]>=15)
{
Dynamx_MovFwd4cm(4, HEXSPD_MEDFAST,
IKCALC_DISABLE);
}

while(Ping[PING_RIGHT]>=16)
{
Dynamx_Rot_Left(HEXSPD_FAST,
HEXSTEP_FAR,IKCALC_DISABLE);
Dynamx_Rot_Left(HEXSPD_FAST,
HEXSTEP_FAR,IKCALC_DISABLE);
}

//RESET WINDUP RIGHT COUNTER
WINDUP_LEFT_COUNTER=0;
//REINIT PID
BW_PID_Init();
}

else
{
Furniture_Avoider();
RIGHT_OVERFLOW_COUNTER=0;
LEFT_OVERFLOW_COUNTER=0;
Dynamx_CurveLeft_PID(HEXSPD_MEDFAST,
HEXSTEP_FAR, &PID_F_R.pid_value[0],
IKCALC_DISABLE);
}

OVERRIDE=1;
}
}
}
}
}

void BW_FollowCounter_Left(int limit)
{
int counter;
FOLLOW_CALC=KIRI;
//int limit=15;
for(counter=1;counter<=limit;counter++)
{Cat_Avoider();Furniture_Avoider();

if (Ping[PING_FRONT]<=15)
{
Sendto_PC(USART1,"BW ROTATE RIGHT \r");
Dynamx_Rot_Right(HEXSPD_FAST,
HEXSTEP_FAR,IKCALC_DISABLE);
}
```

```
else{PID_BW_FollowCounter_Left();}
if(FIRE==FIRE_ON){Bumper_Follow();}
else{Bumper_Action_Left();}}
FOLLOW_CALC=KOSONG;
}

void PID_BW_FollowCounter_Left(void)
{
    //int limit= *COMMAND_LOOP;
    int OVERRIDE=0;
    static int WINDUP_RIGHT_COUNTER=0;
    static int WINDUP_LEFT_COUNTER=0;

while(OVERRIDE==0)
{
if(OVERRIDE==0)
{
    //KONDISI ROBOT SESUAI
if
((Ping[PING_ASKEW_LEFT]<=PID_F_L.set_poin
t_upper) &&
(Ping[PING_ASKEW_LEFT]>=PID_F_L.set_point
_lower) )
{
if (Ping[PING_FRONT]<=15)
{
Sendto_PC(USART1,"BW ROTATE RIGHT \r");
Dynamx_Rot_Right(HEXSPD_FAST,HEXSTEP_F
AR,IKCALC_DISABLE);
}
if(OVERRIDE==1){break;}
if (OVERRIDE==0)
{
RIGHT_OVERFLOW_COUNTER=0;
LEFT_OVERFLOW_COUNTER=0;

Sendto_PC(USART1,"BW FORWARD \r");
Dynamx_MovFwd4cm(4, HEXSPD_MEDFAST,
IKCALC_DISABLE);
OVERRIDE=1;
}
}
//KONDISI ROBOT DEKAT DARI DINDING
else if (
Ping[PING_ASKEW_LEFT]<PID_F_L.set_point )
{
//RESET WINDUP RIGHT COUNTER
WINDUP_LEFT_COUNTER=0;

if (Ping[PING_FRONT]<=15)
{
Sendto_PC(USART1,"BW ROTATE RIGHT \r");
Dynamx_Rot_Right(HEXSPD_FAST,
HEXSTEP_FAR,IKCALC_DISABLE);
}
if(OVERRIDE==1){break;}
if (OVERRIDE==0)
{
if(WINDUP_RIGHT_COUNTER>=25)
{
BW_Buzz(5);
lcd_display_clear();
lcd_gotoxy(0,0);sprintf(lcd,"FOLLOW
OVERFLOW R");lcd_putstr(lcd);
delay_ms(100);

while(Ping[PING_FRONT]>=15)
{
Dynamx_MovFwd4cm(4, HEXSPD_MEDFAST,
IKCALC_DISABLE);
}
while(Ping[PING_LEFT]>=15)
{
Dynamx_Rot_Right(HEXSPD_FAST,HEXSTEP_F
AR,IKCALC_DISABLE);
Dynamx_Rot_Right(HEXSPD_ULTRAFAST,HEX
STEP_FAR, IKCALC_DISABLE);
}
//RESET WINDUP RIGHT COUNTER
WINDUP_RIGHT_COUNTER=0;

//REINIT PID
BW_PID_Init_Room();
}
else
{
Furniture_Avoider();
RIGHT_OVERFLOW_COUNTER=0;
LEFT_OVERFLOW_COUNTER=0;
Dynamx_CurveRight_PID(HEXSPD_MEDFAST,
HEXSTEP_FAR, &PID_F_L.pid_value[0],
IKCALC_DISABLE);
}
OVERRIDE=1;
}
}
//KONDISI ROBOT JAUH DENGAN DINDING
else if (
Ping[PING_ASKEW_LEFT]>PID_F_L.set_point )
{
//RESET WINDUP LEFT COUNTER
WINDUP_RIGHT_COUNTER=0;
if (Ping[PING_FRONT]<=15)
{
Sendto_PC(USART1,"BW ROTATE RIGHT \r");
Dynamx_Rot_Right(HEXSPD_FAST,HEXSTEP_F
AR,IKCALC_DISABLE);
}
```

```
if(OVERRIDE==1){break;}
if (OVERRIDE==0)
{
if(WINDUP_LEFT_COUNTER>=25)
{
BW_Buzz(5);
lcd_display_clear();
lcd_gotoxy(0,0);sprintf(lcd,"FOLLOW
OVERFLOW L");lcd_putstr(lcd);
delay_ms(100);

while(Ping[PING_FRONT]>=15)
{
Dynamx_MovFwd4cm(4, HEXSPD_MEDFAST,
IKCALC_DISABLE);
}
while(Ping[PING_LEFT]>=15)
{
Dynamx_Rot_Right(HEXSPD_ULTRAFAST,
HEXSTEP_FAR, IKCALC_DISABLE);
Dynamx_Rot_Right(HEXSPD_FAST,
HEXSTEP_FAR,IKCALC_DISABLE);
}
```

```
//RESET WINDUP RIGHT COUNTER
WINDUP_LEFT_COUNTER=0;
//REINIT PID
BW_PID_Init_Room();
}
else
{
Furniture_Avoider();
RIGHT_OVERFLOW_COUNTER=0;
LEFT_OVERFLOW_COUNTER=0;

Dynamx_CurveLeft_PID(HEXSPD_MEDFAST,
HEXSTEP_FAR, &PID_F_L.pid_value[0],
IKCALC_DISABLE);
Dynamx_MovFwd1cm(1,HEXSPD_MEDFAST,IK
CALC_ENABLE);
}
OVERRIDE=1;
}
}
}
}}
```

Library dynamixel_hexapod_cmd.h

```
#include "main.h"

#ifndef BW_DYNAMIXEL_HEXAPOD_CMD_H
#defie BW_DYNAMIXEL_HEXAPOD_CMD_H

extern int DynamxCorrection[];

#define link_A                  5      // besar La
#define link_B                  8      // besar Lb
#define L_junction              3.5    // besar
L_junction
#define RAD2DEG                 57.29
#define ANGLE_CORRECTION        45     //koreksi
sudut 0 di teta3,

struct
{
      int ThetaCoxa;
      int ThetaFemur;
      int ThetaTibia;
}JointAngle;

extern float Dynamx_LinkFemur;
extern float Dynamx_LinkTibia;

/*SPEED TRIGONOMETRY*/
extern double PI;
//extern float PIby2;
```

```
#define MAX_UINT  65535
#define MIN_INT -32768
#define MAX_INT  32767

#define DEC1 10
#define DEC2 100
#define DEC3 1000
#define DEC4 10000
extern const float SIN_TABLE[];
extern const int ACOS_TABLE[];
/*
 * BW DYNAMIXEL HEXAPOD VARIABLES
 */
extern int SUDUT_STATIC[];
extern int SUDUT_STATIC_BWD[];
extern int SUDUT_ANGKAT;
// MOV FWD ZCOXA = 35 and Z TIBIA = 25
extern int IKFWD_COXA_R[];
extern int IKFWD_TIBIA_R[];
extern int IKFWD_COXA_L[];
extern int IKFWD_TIBIA_L[];

// MOV FWD ZCOXA = 15 and Z TIBIA = 15
extern int IKFWD_COXA_R1[];
extern int IKFWD_TIBIA_R1[];
extern int IKFWD_COXA_L1[];
extern int IKFWD_TIBIA_L1[];
```

```
// MOV FWD ZCOXA = 10 and Z TIBIA = 10
extern int IKFWD_COXA_R2[];
extern int IKFWD_TIBIA_R2[];
extern int IKFWD_COXA_L2[];
extern int IKFWD_TIBIA_L2[];

// MOV FWD ZCOXA = 5 and Z TIBIA = 5
extern int IKFWD_COXA_R3[];
extern int IKFWD_TIBIA_R3[];
extern int IKFWD_COXA_L3[];
extern int IKFWD_TIBIA_L3[];

//// MOV FWD ZCOXA = 15 and Z TIBIA = 15
//extern int IKFWD_COXA_R2[];
//extern int IKFWD_TIBIA_R2[];
//extern int IKFWD_COXA_L2[];
//extern int IKFWD_TIBIA_L2[];

// MOV ROT ZCOXA_L = 15
extern int IKROT_COXA[];

// MOV ROT ZCOXA_L = 10
extern int IKROT_COXA1[];

// MOV ROT ZCOXA_L = 5
extern int IKROT_COXA2[];

extern int IKCRVR_COXA_R2[];
extern int IKCRVR_TIBIA_R2[];
extern int IKCRVR_COXA_L2[];
extern int IKCRVR_TIBIA_L2[];

extern int IKRPID_COXA_R2[];
extern int IKRPID_TIBIA_R2[];

extern int IKCRVR_COXA_R[];
extern int IKCRVR_TIBIA_R[];
extern int IKCRVR_COXA_L[];
extern int IKCRVR_TIBIA_L[];

extern int IKRPID_COXA_R[];
extern int IKRPID_TIBIA_R[];

extern int IKSLR_COXA[];
extern int IKSLR_TIBIA[];

extern int IKSLR_COXA1[];
extern int IKSLR_TIBIA1[];

extern int IKSLR_COXA2[];
extern int IKSLR_TIBIA2[];

extern int IKCRVL_COXA_R[];
extern int IKCRVL_TIBIA_R[];
extern int IKCRVL_COXA_L[];
extern int IKCRVL_TIBIA_L[];

extern int IKLPID_COXA_L[];
extern int IKLPID_TIBIA_L[];

extern int IKCRVL_COXA_R2[];
extern int IKCRVL_TIBIA_R2[];
extern int IKCRVL_COXA_L2[];
extern int IKCRVL_TIBIA_L2[];

extern int IKLPID_COXA_L2[];
extern int IKLPID_TIBIA_L2[];

extern int IKSLL_COXA[];
extern int IKSLL_TIBIA[];
extern int IKSLL_COXA1[];
extern int IKSLL_TIBIA1[];
extern int IKSLL_COXA2[];
extern int IKSLL_TIBIA2[];
extern int IKBWD_COXA_R[];
extern int IKBWD_TIBIA_R[];
extern int IKBWD_COXA_L[];
extern int IKBWD_TIBIA_L[];
extern int IKCRVRR_COXA_L[];
extern int IKCRVRR_TIBIA_L[];
extern int IKCRVRR_COXA_R[];
extern int IKCRVRR_TIBIA_R[];
extern int IKPIDR_COXA_L[];
extern int IKPIDR_TIBIA_L[];
extern int IKCRVLR_COXA_L[];
extern int IKCRVLR_TIBIA_L[];
extern int IKCRVLR_COXA_R[];
extern int IKCRVLR_TIBIA_R[];
extern int IKPIDR_COXA_R[];
extern int IKPIDR_TIBIA_R[];

/*CONVERSION FROM OLD ALGORITHM*/

extern int sudut_target[];
extern int sudut_awal[];
extern int x;
extern int z_A[];
extern int pembagi;
extern int y_A[];
extern int sudut_tahap1[];
extern int sudut_tahap2[];

//float gamma[6];
extern int leg;
extern int ww[];
extern float beta[];
extern float leg_next[];
extern float A_body;        // translasi pusat badan
extern float A_leg[];       // translasi pangkal kaki
extern float P_body;        // jarak titik pusat robot
ke titik imajiner putar
extern int epsilon_body;    // besar sudut vektor
```

```
extern float phi[];
extern float lambpsi[];       //lambda / psi temporal
extern float P_leg[];         // jarak titik pusat kaki ke
titik imajiner putar
extern float epsilon[];       // epsilon tiap N

extern unsigned int sudut[];

extern unsigned char y;
extern unsigned char a;
extern unsigned char b;
extern unsigned char c;
extern unsigned char d;
extern unsigned char v;
extern unsigned char index_bantu;
extern unsigned char index_servo2;
extern unsigned char index_servo3;
extern uint32_t PrescalerValue;
extern const float A_const[];
extern const int B_const[];
extern float correct[]; //EDITED
extern const float leg_1[];  //L1
extern const float leg_norm[]; //L2
extern const float alpha[];  //alpha
extern float c1;
extern float c2;
extern float Lsem;
extern float c3;
extern float c4;
extern int i;
extern float temp_var;
extern float temp_var2;


//LEG POSITION MACRO
#define FRONT_LEFT_DX      1
#define FRONT_RIGHT_DX     2
#define MIDDLE_LEFT_DX     3
#define MIDDLE_RIGHT_DX    4
#define REAR_LEFT_DX       5
#define REAR_RIGHT_DX      6

#define IKCALC_ENABLE          1
#define IKCALC_DISABLE         0

extern char command_code[];

enum DYNAMX_HEX_SPEED
{HEXSPD_ULTRASLOW=80,
HEXSPD_SLOW=300, HEXSPD_MED=500,
HEXSPD_MEDFAST=650,
HEXSPD_FAST=800,
HEXSPD_ULTRAFAST=1000};
enum
DYNAMX_HEX_STEP{HEXSTEP_VERYCLOSE=1,
HEXSTEP_CLOSE= 2, HEXSTEP_MED=3,
HEXSTEP_FAR=4};

void Dynamx_CurveRight_PID
(int SPEED, int STEP, float *ACTUATE_SIGNAL,
int IKCALC_STAT);
void Dynamx_CurveLeft_PID
(int SPEED, int STEP, float *ACTUATE_SIGNAL,
int IKCALC_STAT);

#endif
```

Library dynamixel_hexapod_cmd.c

```
#include "bw_dynamixel_hexapod_cmd.h"
// DYNAMIXEL ID
const unsigned int FRONT_LEFT_DX_COXA =      1;
const unsigned int FRONT_LEFT_DX_FEMUR=      2;
const unsigned int FRONT_LEFT_DX_TIBIA=      3;

const unsigned int MIDDLE_LEFT_DX_COXA =     4;
const unsigned int MIDDLE_LEFT_DX_FEMUR=     5;
const unsigned int MIDDLE_LEFT_DX_TIBIA=     6;

const unsigned int REAR_LEFT_DX_COXA         =7;
const unsigned int REAR_LEFT_DX_FEMUR        =8;
const unsigned int REAR_LEFT_DX_TIBIA        =9;

const unsigned int FRONT_RIGHT_DX_COXA =     10;
const unsigned int FRONT_RIGHT_DX_FEMUR=     11;
const unsigned int FRONT_RIGHT_DX_TIBIA=     12;

const unsigned int MIDDLE_RIGHT_DX_COXA =    13;
const unsigned int MIDDLE_RIGHT_DX_FEMUR=    14;
const unsigned int MIDDLE_RIGHT_DX_TIBIA=    15;

const unsigned int REAR_RIGHT_DX_COXA        =16;
const unsigned int REAR_RIGHT_DX_FEMUR       =17;
const unsigned int REAR_RIGHT_DX_TIBIA       =18;

void Dynamx_CurveLeft_PID(int SPEED, int STEP, float *ACTUATE_SIGNAL,int IKCALC_STAT)
{
    float signal=*ACTUATE_SIGNAL/5;
    float error_L;//femur_L,tibia_L;
    float error_R;//femur_R,tibia_R;
    //Sendto_PC(USART1," PID: %d\\r",*ACTUATE_SIGNAL);
    //Sendto_PC(USART1," signal: %d\\r",*signal);
    if (signal>=IKFWD_TIBIA_L[2]){error_L=IKFWD_TIBIA_L[2];}
    else {error_R=signal;error_L=signal;}
    float DELTA_L=0 + (1 * error_L);
    int DELTA3[3];
    int DELTA2[3];
    int DELTA1[3];
    int DELTA[3];
    DELTA3[2] = (DELTA_L/IKFWD_TIBIA_L[2])*IKFWD_COXA_L3[2];
    DELTA3[1] = (DELTA_L/IKFWD_TIBIA_L[2])*IKFWD_COXA_L3[1];
    DELTA3[0] = (DELTA_L/IKFWD_TIBIA_L[2])*IKFWD_COXA_L3[0];

    DELTA2[2] = (DELTA_L/IKFWD_TIBIA_L[2])*IKFWD_COXA_L2[2];
    DELTA2[1] = (DELTA_L/IKFWD_TIBIA_L[2])*IKFWD_COXA_L2[1];
    DELTA2[0] = (DELTA_L/IKFWD_TIBIA_L[2])*IKFWD_COXA_L2[0];

    DELTA1[2] = (DELTA_L/IKFWD_TIBIA_L[2])*IKFWD_COXA_L1[2];
    DELTA1[1] = (DELTA_L/IKFWD_TIBIA_L[2])*IKFWD_COXA_L1[1];
    DELTA1[0] = (DELTA_L/IKFWD_TIBIA_L[2])*IKFWD_COXA_L1[0];

    DELTA[2] = (DELTA_L/IKFWD_TIBIA_L[2])*IKFWD_COXA_L[2];
    DELTA[1] = (DELTA_L/IKFWD_TIBIA_L[2])*IKFWD_COXA_L[1];
    DELTA[0] = (DELTA_L/IKFWD_TIBIA_L[2])*IKFWD_COXA_L[0];
```

```
switch(IKCALC_STAT)
{
case IKCALC_ENABLE:{}break;
case IKCALC_DISABLE:
{
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
//2
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX,&SPEED,SUDUT_STATIC[0],SUDUT_STATIC[1],SUDUT_STATIC[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX,&SPEED,SUDUT_STATIC[3],SUDUT_STATIC[4],SUDUT_STATIC[5]);
Dynamx_MoveLeg(REAR_RIGHT_DX,&SPEED,SUDUT_STATIC[6],SUDUT_STATIC[7],SUDUT_STATIC[8]);

//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,&SPEED,SUDUT_STATIC[9],SUDUT_STATIC[10]+SUDUT_ANGKAT,
SUDUT_STATIC[11]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12],SUDUT_STATIC[13]+SUDUT_ANGKAT,
SUDUT_STATIC[14]+SUDUT_ANGKAT);
Dynamx_MoveLeg(REAR_LEFT_DX,&SPEED,SUDUT_STATIC[15],SUDUT_STATIC[16]+SUDUT_ANGKAT,
SUDUT_STATIC[17]+SUDUT_ANGKAT);

if(CAT_DETECTOR==CAT_ACTIVATED){Cat_Avoider();if(CAT_FLAG ==CAT_DETECTED)break;}
if (Ping[PING_FRONT]<=15)break;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;

//3
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R3[2],SUDUT_STATIC[1],
SUDUT_STATIC[2]+IKFWD_TIBIA_R3[0]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L3[1]+DELTA3[1],
SUDUT_STATIC[4],SUDUT_STATIC[5]-IKFWD_TIBIA_L3[1]);//belakang menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,&SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R3[0],SUDUT_STATIC[7],
SUDUT_STATIC[8]-IKFWD_TIBIA_R3[2]);

//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L3[0]-DELTA3[0],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]-IKFWD_TIBIA_L3[2]);          //depan angkat
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R3[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]-IKFWD_TIBIA_R3[1]);
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L3[2]-DELTA3[2],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+IKFWD_TIBIA_L3[0]);
//    delay_ms(5000);

//3
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R2[2],SUDUT_STATIC[1],
SUDUT_STATIC[2]+IKFWD_TIBIA_R2[0]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L2[1]+DELTA2[1],
SUDUT_STATIC[4],SUDUT_STATIC[5]-IKFWD_TIBIA_L2[1]);//belakang menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R2[0],SUDUT_STATIC[7],
SUDUT_STATIC[8]-IKFWD_TIBIA_R2[2]);
```

```
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L2[0]-DELTA2[0],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]-IKFWD_TIBIA_L2[2]);//depan angkat
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R2[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]-IKFWD_TIBIA_R2[1]);
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L2[2]-DELTA2[2],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+IKFWD_TIBIA_L2[0]);
//                      delay_ms(5000);

//3
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R1[2],SUDUT_STATIC[1],
SUDUT_STATIC[2]+IKFWD_TIBIA_R1[0]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L1[1]+DELTA1[1],
SUDUT_STATIC[4]-3,SUDUT_STATIC[5]-IKFWD_TIBIA_L1[1]);  //belakang menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R1[0],SUDUT_STATIC[7]-6,
SUDUT_STATIC[8]-IKFWD_TIBIA_R1[2]);

//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L1[0]-DELTA1[0],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]-IKFWD_TIBIA_L1[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R1[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]-IKFWD_TIBIA_R1[1]);          //depan angkat
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L1[2]-DELTA1[2],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+IKFWD_TIBIA_L1[0]);
//                      delay_ms(5000);

//3
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R[2],
SUDUT_STATIC[1],SUDUT_STATIC[2]+IKFWD_TIBIA_R[0]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L[1]+DELTA[1],
SUDUT_STATIC[4]-5,SUDUT_STATIC[5]-IKFWD_TIBIA_L[1]); //belakang menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R[0],
SUDUT_STATIC[7]-10,SUDUT_STATIC[8]-IKFWD_TIBIA_R[2]);

//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L[0]-DELTA[0],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]-IKFWD_TIBIA_L[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]-IKFWD_TIBIA_R[1]);          //depan angkat
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L[2]-DELTA[2],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+IKFWD_TIBIA_L[0]);
//                      delay_ms(5000);

if(CAT_DETECTOR==CAT_ACTIVATED){Cat_Avoider();if(CAT_FLAG ==CAT_DETECTED)break;}
if (Ping[PING_FRONT]<=15)break;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;

//4
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L[0]-DELTA[0],
SUDUT_STATIC[10]-7,SUDUT_STATIC[11]-IKFWD_TIBIA_L[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R[1],
SUDUT_STATIC[13]-3,SUDUT_STATIC[14]-IKFWD_TIBIA_R[1]);                  //depan menapak
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L[2]-DELTA[2],
SUDUT_STATIC[16],SUDUT_STATIC[17]+IKFWD_TIBIA_L[0]);
```

```
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R[2],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L[1]+DELTA[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]+SUDUT_ANGKAT);   //belakang angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R[0],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+SUDUT_ANGKAT);
//                      delay_ms(5000);
//4
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L1[0]-DELTA1[0],
SUDUT_STATIC[10]-4,SUDUT_STATIC[11]-IKFWD_TIBIA_L1[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R1[1],
SUDUT_STATIC[13]-2,SUDUT_STATIC[14]-IKFWD_TIBIA_R1[1]);                  //depan menapak
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L1[2]-DELTA1[2],
SUDUT_STATIC[16],SUDUT_STATIC[17]+IKFWD_TIBIA_L1[0]);

//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R1[2],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L1[1]+DELTA1[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]+SUDUT_ANGKAT);   //belakang angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R1[0],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+SUDUT_ANGKAT);
//                      delay_ms(5000);
//4
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L2[0]-DELTA2[0],
SUDUT_STATIC[10],SUDUT_STATIC[11]-IKFWD_TIBIA_L2[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R2[1],
SUDUT_STATIC[13],SUDUT_STATIC[14]-IKFWD_TIBIA_R2[1]);                    //depan menapak
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L[2]-DELTA2[2],
SUDUT_STATIC[16],SUDUT_STATIC[17]+IKFWD_TIBIA_L2[0]);

//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R2[2],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L2[1]+DELTA2[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]+SUDUT_ANGKAT);   //belakang angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R2[0],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+SUDUT_ANGKAT);
//                      delay_ms(5000);

//4
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L3[0]-DELTA3[0],
SUDUT_STATIC[10],SUDUT_STATIC[11]-IKFWD_TIBIA_L3[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R3[1],
SUDUT_STATIC[13],SUDUT_STATIC[14]-IKFWD_TIBIA_R3[1]);                    //depan menapak
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L3[2]-DELTA3[2],
SUDUT_STATIC[16],SUDUT_STATIC[17]+IKFWD_TIBIA_L3[0]);
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R3[2],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L3[1]+DELTA3[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]+SUDUT_ANGKAT);   //belakang angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R3[0],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+SUDUT_ANGKAT);
//                      delay_ms(5000);
```

```
if(CAT_DETECTOR==CAT_ACTIVATED){Cat_Avoider();if(CAT_FLAG ==CAT_DETECTED)break;}
if (Ping[PING_FRONT]<=15)break;

//BW_Tracer_Check();
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;

//5
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0],SUDUT_STATIC[1]+
SUDUT_ANGKAT,SUDUT_STATIC[2]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_LEFT_DX,&SPEED,SUDUT_STATIC[3],SUDUT_STATIC[4]+
SUDUT_ANGKAT,SUDUT_STATIC[5]+SUDUT_ANGKAT);//tengah angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,&SPEED,SUDUT_STATIC[6],SUDUT_STATIC[7]+
SUDUT_ANGKAT,SUDUT_STATIC[8]+SUDUT_ANGKAT);

//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,&SPEED,SUDUT_STATIC[9],SUDUT_STATIC[10],
SUDUT_STATIC[11]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12],SUDUT_STATIC[13],
SUDUT_STATIC[14]);
Dynamx_MoveLeg(REAR_LEFT_DX,&SPEED,SUDUT_STATIC[15],SUDUT_STATIC[16],
SUDUT_STATIC[17]);
//                      delay_ms(5000);

if(CAT_DETECTOR==CAT_ACTIVATED){Cat_Avoider();if(CAT_FLAG ==CAT_DETECTED)break;}
if (Ping[PING_FRONT]<=15)break;

//BW_Tracer_Check();
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;

//6
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX,&SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R3[0],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]-IKFWD_TIBIA_R3[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L3[1]-DELTA3[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]-IKFWD_TIBIA_L3[1]);
//depan angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,&SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R3[2],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+IKFWD_TIBIA_R3[0]);



//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L3[2]+DELTA3[2],
SUDUT_STATIC[10],SUDUT_STATIC[11]+IKFWD_TIBIA_L3[0]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R3[1],
SUDUT_STATIC[13],SUDUT_STATIC[14]-IKFWD_TIBIA_R3[1]);          //belakang menapak
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L3[0]+DELTA3[0],
SUDUT_STATIC[16],SUDUT_STATIC[17]-IKFWD_TIBIA_L3[2]);
//                      delay_ms(5000);
```

```
//6
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX,&SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R2[0],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]-IKFWD_TIBIA_R2[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L2[1]-DELTA2[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]-IKFWD_TIBIA_L2[1]);
//depan angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,&SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R2[2],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+IKFWD_TIBIA_R2[0]);

//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L2[2]+DELTA2[2],
SUDUT_STATIC[10],SUDUT_STATIC[11]+IKFWD_TIBIA_L2[0]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R2[1],
SUDUT_STATIC[13],SUDUT_STATIC[14]-IKFWD_TIBIA_R2[1]);          //belakang menapak
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L2[0]+DELTA2[0],
SUDUT_STATIC[16],SUDUT_STATIC[17]-IKFWD_TIBIA_L2[2]);
//                    delay_ms(5000);

//6
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX,&SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R1[0],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]-IKFWD_TIBIA_R1[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L1[1]-DELTA1[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]-IKFWD_TIBIA_L1[1]);          //depan angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,&SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R1[2],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+IKFWD_TIBIA_R1[0]);
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L1[2]+DELTA1[2],
SUDUT_STATIC[10],SUDUT_STATIC[11]+IKFWD_TIBIA_L1[0]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R1[1],
SUDUT_STATIC[13]-3,SUDUT_STATIC[14]-IKFWD_TIBIA_R1[1]);          //belakang menapak
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L1[0]+DELTA1[0],
SUDUT_STATIC[16]-6,SUDUT_STATIC[17]-IKFWD_TIBIA_L1[2]);
//                    delay_ms(5000);
//6
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX,&SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R[0],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]-IKFWD_TIBIA_R[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L[1]-DELTA[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]-IKFWD_TIBIA_L[1]);  //depan angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R[2],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+IKFWD_TIBIA_R[0]);
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L[2]+DELTA[2],
SUDUT_STATIC[10],SUDUT_STATIC[11]+IKFWD_TIBIA_L[0]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R[1],
SUDUT_STATIC[13]-5,SUDUT_STATIC[14]-IKFWD_TIBIA_R[1]);          //belakang menapak
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L[0]+DELTA[0],
SUDUT_STATIC[16]-10,SUDUT_STATIC[17]-IKFWD_TIBIA_L[2]);
//                    delay_ms(5000);
if(CAT_DETECTOR==CAT_ACTIVATED){Cat_Avoider();if(CAT_FLAG ==CAT_DETECTED)break;}
if (Ping[PING_FRONT]<=15)break;

//BW_Tracer_Check();
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
```

```
//1
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX,&SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R[0],
SUDUT_STATIC[1]-7,SUDUT_STATIC[2]-IKFWD_TIBIA_R[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L[1]-DELTA[1],
SUDUT_STATIC[4]-3,SUDUT_STATIC[5]-IKFWD_TIBIA_L[1]); //depan menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,&SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R[2],
SUDUT_STATIC[7],SUDUT_STATIC[8]+IKFWD_TIBIA_R[0]);
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L[2]+DELTA[2],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]+SUDUT_ANGKAT);//belakang angkat
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L[0]+DELTA[0],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+SUDUT_ANGKAT);
//                      delay_ms(5000);

//1
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R1[0],
SUDUT_STATIC[1]-4,SUDUT_STATIC[2]-IKFWD_TIBIA_R1[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L1[1]-DELTA1[1],
SUDUT_STATIC[4]-2,SUDUT_STATIC[5]-IKFWD_TIBIA_L1[1]);          //depan menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,&SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R1[2],
SUDUT_STATIC[7],SUDUT_STATIC[8]+IKFWD_TIBIA_R1[0]);
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L1[2]+DELTA1[2],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R1[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]+SUDUT_ANGKAT);          //belakang angkat
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L1[0]+DELTA1[0],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+SUDUT_ANGKAT);
//                      delay_ms(5000);

//1
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX,&SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R2[0],
SUDUT_STATIC[1],SUDUT_STATIC[2]-IKFWD_TIBIA_R2[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L2[1]-DELTA2[1],
SUDUT_STATIC[4],SUDUT_STATIC[5]-IKFWD_TIBIA_L2[1]);  //depan menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,&SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R2[2],
SUDUT_STATIC[7],SUDUT_STATIC[8]+IKFWD_TIBIA_R2[0]);
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L2[2]+DELTA2[2],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R2[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]+SUDUT_ANGKAT); //belakang angkat
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L2[0]+DELTA2[0],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+SUDUT_ANGKAT);
//                      delay_ms(5000);

//1
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX,&SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R3[0],
SUDUT_STATIC[1],SUDUT_STATIC[2]-IKFWD_TIBIA_R3[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L3[1]-DELTA3[1],
SUDUT_STATIC[4],SUDUT_STATIC[5]-IKFWD_TIBIA_L3[1]);  //depan menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,&SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R3[2],
SUDUT_STATIC[7],SUDUT_STATIC[8]+IKFWD_TIBIA_R3[0]);
```

```
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L3[2]+DELTA3[2],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R3[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]+SUDUT_ANGKAT);//belakang angkat
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L3[0]+DELTA3[0],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+SUDUT_ANGKAT);
//                              delay_ms(5000);
if(CAT_DETECTOR==CAT_ACTIVATED){Cat_Avoider();if(CAT_FLAG ==CAT_DETECTED)break;}
if (Ping[PING_FRONT]<=15)break;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
}break;
}
Bumper_Action();
}

void Dynamx_CurveRight_PID(int SPEED, int STEP, float *ACTUATE_SIGNAL, int IKCALC_STAT)
{
    float signal=*ACTUATE_SIGNAL/5;
    float error_L;//femur_L,tibia_L;
    float error_R;//femur_R,tibia_R;
    if (signal>=IKFWD_TIBIA_R2[2]){error_R=IKFWD_TIBIA_R2[2];}
    else {error_R=signal;error_L=signal;}
    float DELTA_R=0 + (1 * error_R);
    //Sendto_PC(USART1," PID: %d\\r",*ACTUATE_SIGNAL);
    //Sendto_PC(USART1," signal: %d\\r",*signal);
    int DELTA3[3];
    int DELTA2[3];
    int DELTA1[3];
    int DELTA[3];

    DELTA3[2] = (DELTA_R/IKFWD_TIBIA_R2[2])*IKFWD_COXA_R3[2];
    DELTA3[1] = (DELTA_R/IKFWD_TIBIA_R2[2])*IKFWD_COXA_R3[1];
    DELTA3[0] = (DELTA_R/IKFWD_TIBIA_R2[2])*IKFWD_COXA_R3[0];

    DELTA2[2] = (DELTA_R/IKFWD_TIBIA_R2[2])*IKFWD_COXA_R2[2];
    DELTA2[1] = (DELTA_R/IKFWD_TIBIA_R2[2])*IKFWD_COXA_R2[1];
    DELTA2[0] = (DELTA_R/IKFWD_TIBIA_R2[2])*IKFWD_COXA_R2[0];

    DELTA1[2] = (DELTA_R/IKFWD_TIBIA_R2[2])*IKFWD_COXA_R1[2];
    DELTA1[1] = (DELTA_R/IKFWD_TIBIA_R2[2])*IKFWD_COXA_R1[1];
    DELTA1[0] = (DELTA_R/IKFWD_TIBIA_R2[2])*IKFWD_COXA_R1[0];

    DELTA[2] = (DELTA_R/IKFWD_TIBIA_R2[2])*IKFWD_COXA_R[2];
    DELTA[1] = (DELTA_R/IKFWD_TIBIA_R2[2])*IKFWD_COXA_R[1];
    DELTA[0] = (DELTA_R/IKFWD_TIBIA_R2[2])*IKFWD_COXA_R[0];

switch(IKCALC_STAT){case IKCALC_ENABLE:{}break;
case IKCALC_DISABLE:
{
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
```

```
//2
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX,&SPEED,SUDUT_STATIC[0],SUDUT_STATIC[1],
SUDUT_STATIC[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX,&SPEED,SUDUT_STATIC[3],SUDUT_STATIC[4],
SUDUT_STATIC[5]);
Dynamx_MoveLeg(REAR_RIGHT_DX,&SPEED,SUDUT_STATIC[6],SUDUT_STATIC[7],
SUDUT_STATIC[8]);
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,&SPEED,SUDUT_STATIC[9],SUDUT_STATIC[10]+SUDUT_ANGKAT,
SUDUT_STATIC[11]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12],SUDUT_STATIC[13]+SUDUT_ANGKAT,
SUDUT_STATIC[14]+SUDUT_ANGKAT);
Dynamx_MoveLeg(REAR_LEFT_DX,&SPEED,SUDUT_STATIC[15],SUDUT_STATIC[16]+SUDUT_ANGKAT,
SUDUT_STATIC[17]+SUDUT_ANGKAT);

if(CAT_DETECTOR==CAT_ACTIVATED) {Cat_Avoider();if(CAT_FLAG ==CAT_DETECTED)break;
if (Ping[PING_FRONT]<=15)break;

TCS3200_Transmit(UART5, CMD_GET_TRACER);

if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;

//3
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R3[2]+DELTA3[2],
SUDUT_STATIC[1],SUDUT_STATIC[2]+IKFWD_TIBIA_R3[0]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L3[1],
SUDUT_STATIC[4],SUDUT_STATIC[5]-IKFWD_TIBIA_L3[1]);  //belakang menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R3[0]+DELTA3[0],
SUDUT_STATIC[7],SUDUT_STATIC[8]-IKFWD_TIBIA_R3[2]);
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,&SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L3[0],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]-IKFWD_TIBIA_L3[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R3[1]-DELTA3[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]-IKFWD_TIBIA_R3[1]);          //depan angkat
Dynamx_MoveLeg(REAR_LEFT_DX,&SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L3[2],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+IKFWD_TIBIA_L3[0]);
//                      delay_ms(5000);

//3
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R2[2]+DELTA2[2],
SUDUT_STATIC[1],SUDUT_STATIC[2]+IKFWD_TIBIA_R2[0]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L2[1],
SUDUT_STATIC[4],SUDUT_STATIC[5]-IKFWD_TIBIA_L2[1]);  //belakang menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R2[0]+DELTA2[0],
SUDUT_STATIC[7],SUDUT_STATIC[8]-IKFWD_TIBIA_R2[2]);
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,&SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L2[0],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]-IKFWD_TIBIA_L2[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R2[1]-DELTA2[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]-IKFWD_TIBIA_R2[1]);          //depan angkat
Dynamx_MoveLeg(REAR_LEFT_DX,&SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L2[2],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+IKFWD_TIBIA_L2[0]);
//                      delay_ms(5000);
```

```
//3
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R1[2]+DELTA1[2],
SUDUT_STATIC[1],SUDUT_STATIC[2]+IKFWD_TIBIA_R1[0]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L1[1],
SUDUT_STATIC[4]-3,SUDUT_STATIC[5]-IKFWD_TIBIA_L1[1]);          //belakang menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R1[0]+DELTA1[0],
SUDUT_STATIC[7]-6,SUDUT_STATIC[8]-IKFWD_TIBIA_R1[2]);

//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L1[0],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]-IKFWD_TIBIA_L1[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R1[1]-DELTA1[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]-IKFWD_TIBIA_R1[1]);      //depan angkat
Dynamx_MoveLeg(REAR_LEFT_DX,&SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L1[2],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+IKFWD_TIBIA_L1[0]);
//                  delay_ms(5000);

//3
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R[2]+DELTA[2],
SUDUT_STATIC[1],SUDUT_STATIC[2]+IKFWD_TIBIA_R[0]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L[1],
SUDUT_STATIC[4]-5,SUDUT_STATIC[5]-IKFWD_TIBIA_L[1]); //belakang menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R[0]+DELTA[0],
SUDUT_STATIC[7]-10,SUDUT_STATIC[8]-IKFWD_TIBIA_R[2]);

//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,&SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L[0],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]-IKFWD_TIBIA_L[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R[1]-DELTA[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]-IKFWD_TIBIA_R[1]);      //depan angkat
Dynamx_MoveLeg(REAR_LEFT_DX,&SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L[2],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+IKFWD_TIBIA_L[0]);
//                  delay_ms(5000);

if(CAT_DETECTOR==CAT_ACTIVATED){Cat_Avoider();if(CAT_FLAG ==CAT_DETECTED)break;}
if (Ping[PING_FRONT]<=15)break;
//BW_Tracer_Check();
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;

//4
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,&SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L[0],
SUDUT_STATIC[10]-7,SUDUT_STATIC[11]-IKFWD_TIBIA_L[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R[1]-DELTA[1],
SUDUT_STATIC[13]-3,SUDUT_STATIC[14]-IKFWD_TIBIA_R[1]);          //depan menapak
Dynamx_MoveLeg(REAR_LEFT_DX,&SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L[2],
SUDUT_STATIC[16],SUDUT_STATIC[17]+IKFWD_TIBIA_L[0]);
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R[2]+DELTA[2],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]+SUDUT_ANGKAT);//belakang angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R[0]+DELTA[0],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+SUDUT_ANGKAT);
//                  delay_ms(5000);
```

```
//4
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,&SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L1[0],
SUDUT_STATIC[10]-4,SUDUT_STATIC[11]-IKFWD_TIBIA_L1[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R1[1]-DELTA1[1],
SUDUT_STATIC[13]-2,SUDUT_STATIC[14]-IKFWD_TIBIA_R1[1]);                    //depan menapak
Dynamx_MoveLeg(REAR_LEFT_DX,&SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L1[2],
SUDUT_STATIC[16],SUDUT_STATIC[17]+IKFWD_TIBIA_L1[0]);
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R1[2]+DELTA1[2],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L1[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]+SUDUT_ANGKAT);   //belakang angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R1[0]+DELTA1[0],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+SUDUT_ANGKAT);
//                    delay_ms(5000);

//4
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,&SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L2[0],
SUDUT_STATIC[10],SUDUT_STATIC[11]-IKFWD_TIBIA_L2[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R2[1]-DELTA2[1],
SUDUT_STATIC[13],SUDUT_STATIC[14]-IKFWD_TIBIA_R2[1]);                    //depan menapak
Dynamx_MoveLeg(REAR_LEFT_DX,&SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L[2],
SUDUT_STATIC[16],SUDUT_STATIC[17]+IKFWD_TIBIA_L2[0]);
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R2[2]+DELTA2[2],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L2[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]+SUDUT_ANGKAT);   //belakang angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R2[0]+DELTA2[0],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+SUDUT_ANGKAT);
//                    delay_ms(5000);

//4
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX, &SPEED,SUDUT_STATIC[9]+IKFWD_COXA_L3[0]-DELTA3[0],
SUDUT_STATIC[10],SUDUT_STATIC[11]-IKFWD_TIBIA_L3[2]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]+IKFWD_COXA_R3[1],
SUDUT_STATIC[13],SUDUT_STATIC[14]-IKFWD_TIBIA_R3[1]);                    //depan menapak
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]+IKFWD_COXA_L3[2]-
DELTA3[2],SUDUT_STATIC[16],SUDUT_STATIC[17]+IKFWD_TIBIA_L3[0]);
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]-IKFWD_COXA_R3[2],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_LEFT_DX, &SPEED,SUDUT_STATIC[3]-IKFWD_COXA_L3[1]+DELTA3[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]+SUDUT_ANGKAT);   //belakang angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]-IKFWD_COXA_R3[0],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+SUDUT_ANGKAT);
//                    delay_ms(5000);

if(CAT_DETECTOR==CAT_ACTIVATED){Cat_Avoider();if(CAT_FLAG ==CAT_DETECTED)break;}
if (Ping[PING_FRONT]<=15)break;
//BW_Tracer_Check();
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
```

```
//5
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX,&SPEED,SUDUT_STATIC[0],SUDUT_STATIC[1]+SUDUT_ANGKAT,
SUDUT_STATIC[2]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_LEFT_DX,&SPEED,SUDUT_STATIC[3],SUDUT_STATIC[4]+SUDUT_ANGKAT,
SUDUT_STATIC[5]+SUDUT_ANGKAT);//tengah angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,&SPEED,SUDUT_STATIC[6],SUDUT_STATIC[7]+SUDUT_ANGKAT,
SUDUT_STATIC[8]+SUDUT_ANGKAT);

//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,&SPEED,SUDUT_STATIC[9],SUDUT_STATIC[10], SUDUT_STATIC[11]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12],SUDUT_STATIC[13], SUDUT_STATIC[14]);
Dynamx_MoveLeg(REAR_LEFT_DX,&SPEED,SUDUT_STATIC[15],SUDUT_STATIC[16], SUDUT_STATIC[17]);
//                      delay_ms(5000);

if(CAT_DETECTOR==CAT_ACTIVATED){Cat_Avoider();if(CAT_FLAG ==CAT_DETECTED)break;}
if (Ping[PING_FRONT]<=15)break;

//BW_Tracer_Check();
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;

//6
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R3[0]-DELTA3[0],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]-IKFWD_TIBIA_R3[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX,&SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L3[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]-IKFWD_TIBIA_L3[1]); //depan angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R3[2]-DELTA3[2],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+IKFWD_TIBIA_R3[0]);

//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L3[2],SUDUT_STATIC[10],
SUDUT_STATIC[11]+IKFWD_TIBIA_L3[0]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R3[1]+DELTA3[1],
SUDUT_STATIC[13],SUDUT_STATIC[14]-IKFWD_TIBIA_R3[1] //belakang menapak
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L3[0],SUDUT_STATIC[16],
SUDUT_STATIC[17]-IKFWD_TIBIA_L3[2]);
//                      delay_ms(5000);
//6
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R2[0]-DELTA2[0],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]-IKFWD_TIBIA_R2[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX,&SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L2[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]-IKFWD_TIBIA_L2[1]);              //depan angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R2[2]-DELTA2[2],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+IKFWD_TIBIA_R2[0]);
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L2[2],SUDUT_STATIC[10],
SUDUT_STATIC[11]+IKFWD_TIBIA_L2[0]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R2[1]+DELTA2[1],
SUDUT_STATIC[13],SUDUT_STATIC[14]-IKFWD_TIBIA_R2[1]);//belakang menapak
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L2[0],SUDUT_STATIC[16],
SUDUT_STATIC[17]-IKFWD_TIBIA_L2[2]);
//                      delay_ms(5000);
```

```
//6
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R1[0]-DELTA1[0],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]-IKFWD_TIBIA_R1[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX,&SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L1[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]-IKFWD_TIBIA_L1[1]); //depan angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R1[2]-DELTA1[2],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+IKFWD_TIBIA_R1[0]);

//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L1[2],
SUDUT_STATIC[10],SUDUT_STATIC[11]+IKFWD_TIBIA_L1[0]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R1[1]+DELTA1[1],
SUDUT_STATIC[13]-3,SUDUT_STATIC[14]-IKFWD_TIBIA_R1[1]);//belakang menapak
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L1[0],
SUDUT_STATIC[16]-6,SUDUT_STATIC[17]-IKFWD_TIBIA_L1[2]);
//                      delay_ms(5000);

//6
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R[0]-DELTA[0],
SUDUT_STATIC[1]+SUDUT_ANGKAT,SUDUT_STATIC[2]-IKFWD_TIBIA_R[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX,&SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L[1],
SUDUT_STATIC[4]+SUDUT_ANGKAT,SUDUT_STATIC[5]-IKFWD_TIBIA_L[1]);  //depan angkat
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R[2]-DELTA[2],
SUDUT_STATIC[7]+SUDUT_ANGKAT,SUDUT_STATIC[8]+IKFWD_TIBIA_R[0]);
/TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L[2],
SUDUT_STATIC[10],SUDUT_STATIC[11]+IKFWD_TIBIA_L[0]);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R[1]+DELTA[1],
SUDUT_STATIC[13]-5,SUDUT_STATIC[14]-IKFWD_TIBIA_R[1]);//belakang menapak
Dynamx_MoveLeg(REAR_LEFT_DX,&SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L[0],
SUDUT_STATIC[16]-10,SUDUT_STATIC[17]-IKFWD_TIBIA_L[2]);
//                      delay_ms(5000);

if(CAT_DETECTOR==CAT_ACTIVATED){Cat_Avoider();if(CAT_FLAG ==CAT_DETECTED)break;}
if (Ping[PING_FRONT]<=15)break;
//BW_Tracer_Check();
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;

//1
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R[0]-DELTA[0],
SUDUT_STATIC[1]-7,SUDUT_STATIC[2]-IKFWD_TIBIA_R[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX,&SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L[1],
SUDUT_STATIC[4]-3,SUDUT_STATIC[5]-IKFWD_TIBIA_L[1]); //depan menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R[2]-DELTA[2],
SUDUT_STATIC[7],SUDUT_STATIC[8]+IKFWD_TIBIA_R[0]);

//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L[2],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R[1]+DELTA[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]+SUDUT_ANGKAT);//belakang angkat
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L[0],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+SUDUT_ANGKAT);
//                      delay_ms(5000);
```

```
//1
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R1[0]-DELTA1[0],
SUDUT_STATIC[1]-4,SUDUT_STATIC[2]-IKFWD_TIBIA_R1[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX,&SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L1[1],
SUDUT_STATIC[4]-2,SUDUT_STATIC[5]-IKFWD_TIBIA_L1[1]);          //depan menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R1[2]-DELTA1[2],
SUDUT_STATIC[7],SUDUT_STATIC[8]+IKFWD_TIBIA_R1[0]);
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L1[2],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R1[1]+DELTA1[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]+SUDUT_ANGKAT);          //belakang angkat
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L1[0],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+SUDUT_ANGKAT);
//                    delay_ms(5000);

//1
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R2[0]-DELTA2[0],
SUDUT_STATIC[1],SUDUT_STATIC[2]-IKFWD_TIBIA_R2[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX,&SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L2[1],
SUDUT_STATIC[4],SUDUT_STATIC[5]-IKFWD_TIBIA_L2[1]);  //depan menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R2[2]-DELTA2[2],
SUDUT_STATIC[7],SUDUT_STATIC[8]+IKFWD_TIBIA_R2[0]);
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L2[2],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R2[1]+DELTA2[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]+ SUDUT_ANGKAT);//belakang angkat
Dynamx_MoveLeg(REAR_LEFT_DX,   &SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L2[0],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+SUDUT_ANGKAT);
//                    delay_ms(5000);

//1
//TRIPOD A
Dynamx_MoveLeg(FRONT_RIGHT_DX, &SPEED,SUDUT_STATIC[0]+IKFWD_COXA_R3[0]-DELTA3[0],
SUDUT_STATIC[1],SUDUT_STATIC[2]-IKFWD_TIBIA_R3[2]);
Dynamx_MoveLeg(MIDDLE_LEFT_DX,&SPEED,SUDUT_STATIC[3]+IKFWD_COXA_L3[1],
SUDUT_STATIC[4],SUDUT_STATIC[5]-IKFWD_TIBIA_L3[1]);  //depan menapak
Dynamx_MoveLeg(REAR_RIGHT_DX,  &SPEED,SUDUT_STATIC[6]+IKFWD_COXA_R3[2]-DELTA3[2],
SUDUT_STATIC[7],SUDUT_STATIC[8]+IKFWD_TIBIA_R3[0]);
//TRIPOD B
Dynamx_MoveLeg(FRONT_LEFT_DX,  &SPEED,SUDUT_STATIC[9]-IKFWD_COXA_L3[2],
SUDUT_STATIC[10]+SUDUT_ANGKAT,SUDUT_STATIC[11]+SUDUT_ANGKAT);
Dynamx_MoveLeg(MIDDLE_RIGHT_DX,&SPEED,SUDUT_STATIC[12]-IKFWD_COXA_R3[1]+DELTA3[1],
SUDUT_STATIC[13]+SUDUT_ANGKAT,SUDUT_STATIC[14]+SUDUT_ANGKAT);          //belakang angkat
Dynamx_MoveLeg(REAR_LEFT_DX,&SPEED,SUDUT_STATIC[15]-IKFWD_COXA_L3[0],
SUDUT_STATIC[16]+SUDUT_ANGKAT,SUDUT_STATIC[17]+SUDUT_ANGKAT);
//                    delay_ms(5000);

if(CAT_DETECTOR==CAT_ACTIVATED){Cat_Avoider();if(CAT_FLAG ==CAT_DETECTED)break;}
//BW_Tracer_Check();
if (Ping[PING_FRONT]<=15)break;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
TCS3200_Transmit(UART5, CMD_GET_TRACER);
if(TCS3200_SLV_DATA[1]==TRACER_WHITE) TRACER_STAT= TRACER_WHITE;
```

LAMPIRAN 4

DATA PENGUJIAN

Tabel 4.1 Data Pengujian pada Lintasan lurus Setpoint 16 cm waktu tempuh 3.02 s

Setpoint	Jarak	E	Kp	Ki	Kd	P	I	D	Aksi Kontrol	Persentase Error Jarak
16	13	-3	0.6	1	0.04	0.6	1,2	0.005	1.8	18.75
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25

16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0

16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	17	1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	13	-3	0.6	1	0.04	0.6	1,2	0.005	1.8	0
					Rata Rata Error					2.997449

Tabel 4.2 Data Pengujian Lintasan lurus dengan gangguan Setpoint 16 cm waktu tempuh 3.52 s

Setpoint	Jarak (cm)	E	Kp	Ki	Kd	P	I	D	Aksi Kontrol	Persentase Error
16	13	-3	0.6	1	0.04	0.6	1,2	0.005	1.8	18.75
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25

16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	14	-2	0.27	1	0.02	0.27	0,8	0.005	1.07	12.5
16	14	-2	0.27	1	0.02	0.27	0,8	0.005	1.07	12.5
16	14	-2	0.27	1	0.02	0.27	0,8	0.005	1.07	12.5
16	14	-2	0.27	1	0.02	0.27	0,8	0.005	1.07	12.5
16	14	2	0.27	1	0.02	0.27	0,8	0.005	1.07	12.5
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0

	16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
	16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
	16	16	0	0	0	0	0	0	0	0	0
	16	16	0	0	0	0	0	0	0	0	0
	16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
	16	16	0	0	0	0	0	0	0	0	0
	16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
	16	16	0	0	0	0	0	0	0	0	0
	16	16	0	0	0	0	0	0	0	0	0
	16	16	0	0	0	0	0	0	0	0	0
	16	16	0	0	0	0	0	0	0	0	0
						Rata Rata Error					3.472222

Tabel 4.3 Data Pengujian telusur kanan pada Lintasan berbelok setpoint 16 waktu tempuh 5.04 s

Setpoint	Jarak (cm)	E	Kp	Ki	Kd	P	I	D	Aksi Kontrol	Persentase Error
16	13	-3	0.6	1	0.04	0.6	1,2	0.005	1.8	18.75
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0,67	1	0,44	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.67	1	0.44	0,67	0,4	0.005	0,71	6.25
16	15	-1	0.67	1	0.44	0,67	0,4	0.005	0,71	6.25
16	15	-1	0.67	1	0.44	0,67	0,4	0.005	0,71	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25

16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	14	-2	0.27	1	0.02	0.27	0,8	0.005	1.07	12.5
16	14	-2	0.27	1	0.02	0.27	0,8	0.005	1.07	12.5
16	14	-2	0.27	1	0.02	0.27	0,8	0.005	1.07	12.5
16	14	-2	0.27	1	0.02	0.27	0,8	0.005	1.07	12.5
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0

16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
Rata Rata Error										3.409091

Tabel 4.4 Data Pengujian telusur kanan pada Lintasan berbelok setpoint 16 waktu tempuh 5.04 s

Setpoint	Jarak (cm)	E	Kp	Ki	Kd	P	I	D	Aksi Kontrol	Persentase Error
16	13	-3	0.6	1	0.04	0.6	1,2	0.005	1.8	18.75
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	14	-2	0.27	1	0.02	0.27	0,8	0.005	1.07	12.5
16	14	-2	0.27	1	0.02	0.27	0,8	0.005	1.07	12.5
16	14	-2	0.27	1	0.02	0.27	0,8	0.005	1.07	12.5
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25

16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0
16	16	0	0	0	0	0	0	0	0	0

	16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
	16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
	16	16	0	0	0	0	0	0	0	0	0
	16	16	0	0	0	0	0	0	0	0	0
	16	17	1	-0.07	-1	0	-0.07	-0,4	-0.005	-0.47	6.25
	16	16	0	0	0	0	0	0	0	0	0
	16	15	-1	0.07	1	0	0,07	0,4	0.005	0,47	6.25
	16	16	0	0	0	0	0	0	0	0	0
	16	16	0	0	0	0	0	0	0	0	0
	16	16	0	0	0	0	0	0	0	0	0
	16	16	0	0	0	0	0	0	0	0	0
						Rata Rata Error					3.282828

LAMPIRAN 5

DATASHEET

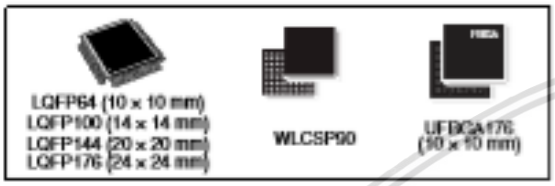
1. Datasheet Mikrokontroler Utama STM32F4VG

life.augmented

STM32F405xx
STM32F407xx

ARM Cortex-M4 32b MCU+FPU, 210DMIPS, up to 1MB Flash/192+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 15 comm. interfaces & camera

Datasheet - production data



Features

- Core: ARM 32-bit Cortex™-M4 CPU with FPU, Adaptive real-time accelerator (ART Accelerator™) allowing 0-wait state execution from Flash memory, frequency up to 168 MHz, memory protection unit, 210 DMIPS/ 1.25 DMIPS/MHz (Dhrystone 2.1), and DSP instructions
- Memories
  - Up to 1 Mbyte of Flash memory
  - Up to 192+4 Kbytes of SRAM including 64-Kbyte of CCM (core coupled memory) data RAM
  - Flexible static memory controller supporting Compact Flash, SRAM, PSRAM, NOR and NAND memories
- LCD parallel interface, 8080/6800 modes
- Clock, reset and supply management
  - 1.8 V to 3.6 V application supply and I/Os
  - POR, PDR, PVD and BOR
  - 4-to-26 MHz crystal oscillator
  - Internal 16 MHz factory-trimmed RC (1% accuracy)
  - 32 kHz oscillator for RTC with calibration
  - Internal 32 kHz RC with calibration
- Low power
  - Sleep, Stop and Standby modes
  - V_{BAT} supply for RTC, 20×32 bit backup registers + optional 4 KB backup SRAM
- 3×12-bit, 2.4 MSPS A/D converters: up to 24 channels and 7.2 MSPS in triple interleaved mode
- 2×12-bit D/A converters
- General-purpose DMA: 16-stream DMA controller with FIFOs and burst support
- Up to 17 timers: up to twelve 16-bit and two 32-bit timers up to 168 MHz, each with up to 4 IC/OC/PWM or pulse counter and quadrature (incremental) encoder input
- Debug mode
  - Serial wire debug (SWD) & JTAG interfaces
  - Cortex-M4 Embedded Trace Macrocell™
- Up to 140 I/O ports with interrupt capability
  - Up to 136 fast I/Os up to 84 MHz
  - Up to 138 5 V-tolerant I/Os
- Up to 15 communication interfaces
  - Up to 3 × I^2C interfaces (SMBus/PMBus)
  - Up to 4 USARTs/2 UARTs (10.5 Mbit/s, ISO 7816 interface, LIN, IrDA, modem control)
  - Up to 3 SPIs (42 Mbits/s), 2 with muxed full-duplex I^2S to achieve audio class accuracy via internal audio PLL or external clock
  - 2 x CAN interfaces (2.0B Active)
  - SDIO interface
- Advanced connectivity
  - USB 2.0 full-speed device/host/OTG controller with on-chip PHY
  - USB 2.0 high-speed/full-speed device/host/OTG controller with dedicated DMA, on-chip full-speed PHY and ULPI
  - 10/100 Ethernet MAC with dedicated DMA: supports IEEE 1588v2 hardware, MII/RMII
- 8- to 14-bit parallel camera interface up to 54 Mbytes/s
- True random number generator
- CRC calculation unit
- 96-bit unique ID
- RTC: subsecond accuracy, hardware calendar

Table 1. Device summary

Reference	Part number
STM32F405xx	STM32F405RG, STM32F405VG, STM32F405ZG, STM32F405OG, STM32F405OE
STM32F407xx	STM32F407VG, STM32F407IG, STM32F407ZG, STM32F407VE, STM32F407ZE, STM32F407IE

2. Datasheet sensor Ultrasonik Parrallax PING)))

PARALLAX

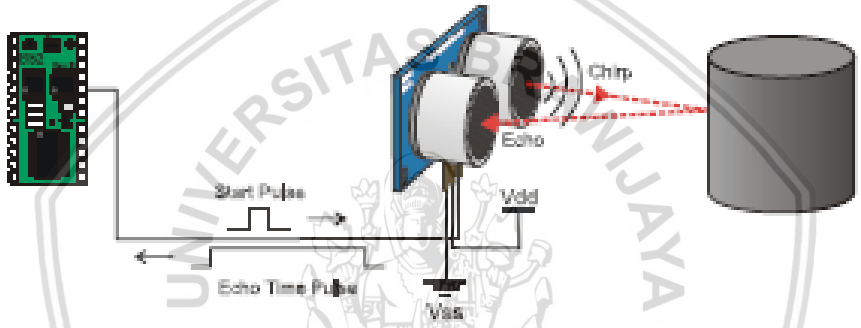
Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

PING))) Ultrasonic Distance Sensor (#28015)

The Parallax PING)))™ ultrasonic distance sensor provides precise, non-contact distance measurements from about 2 cm (0.8 inches) to 3 meters (3.3 yards). It is very easy to connect to microcontrollers such as the BASIC Stamp®, Propeller chip, or Arduino, requiring only one I/O pin.

The PING))) sensor works by transmitting an ultrasonic (well above human hearing range) burst and providing an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width, the distance to target can easily be calculated.



Features

- Range: 2 cm to 3 m (0.8 in to 3.3 yd)
- Burst indicator LED shows sensor activity
- Bidirectional TTL pulse interface on a single I/O pin can communicate with 5 V TTL or 3.3 V CMOS microcontrollers
- Input trigger: positive TTL pulse, 2 µs min, 5 µs typ.
- Echo pulse: positive TTL pulse, 115 µs minimum to 18.5 ms maximum.
- RoHS Compliant

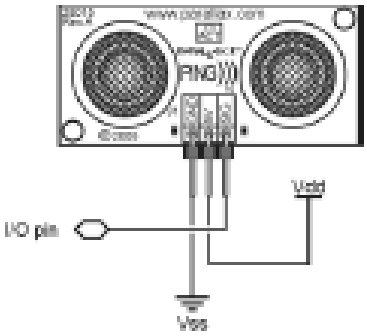
Key Specifications

- Supply voltage: +5 VDC
- Supply current: 30 mA typ; 35 mA max
- Communication: Positive TTL pulse
- Package: 3-pin SIP, 0.1" spacing (ground, power, signal)
- Operating temperature: 0 – 70° C.
- Size: 22 mm H x 46 mm W x 16 mm D (0.84 in x 1.8 in x 0.6 in)
- Weight: 9 g (0.32 oz)

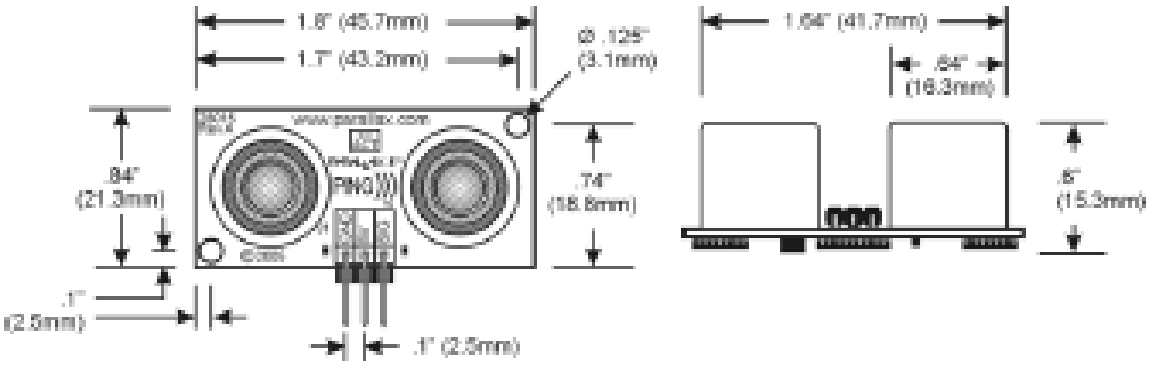
Pin Definitions

GND	Ground (Vss)
5 V	5 VDC (Vdd)
SIG	Signal (I/O pin)

The PING))) sensor has a male 3-pin header used to supply ground, power (+5 VDC) and signal. The header may be plugged into a directly into solderless breadboard, or into a standard 3-wire extension cable (Parallax part #800-00120).

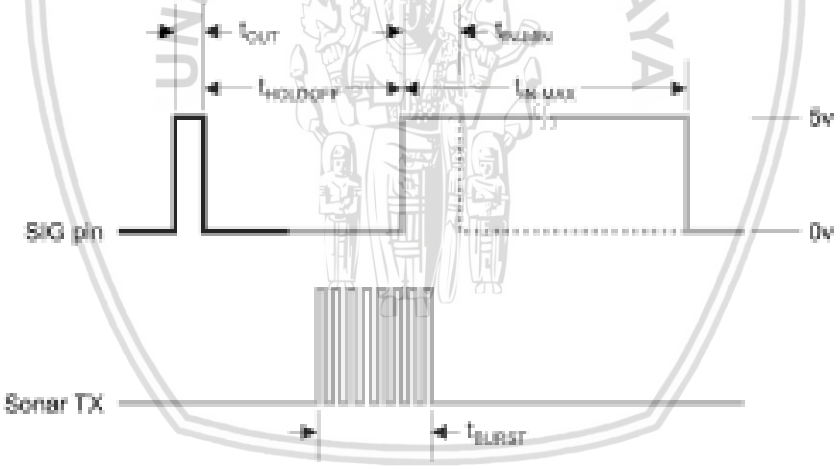


Dimensions



Communication Protocol

The PING))) sensor detects objects by emitting a short ultrasonic burst and then "listening" for the echo. Under control of a host microcontroller (trigger pulse), the sensor emits a short 40 kHz (ultrasonic) burst. This burst travels through the air, hits an object and then bounces back to the sensor. The PING))) sensor provides an output pulse to the host that will terminate when the echo is detected, hence the width of this pulse corresponds to the distance to the target.

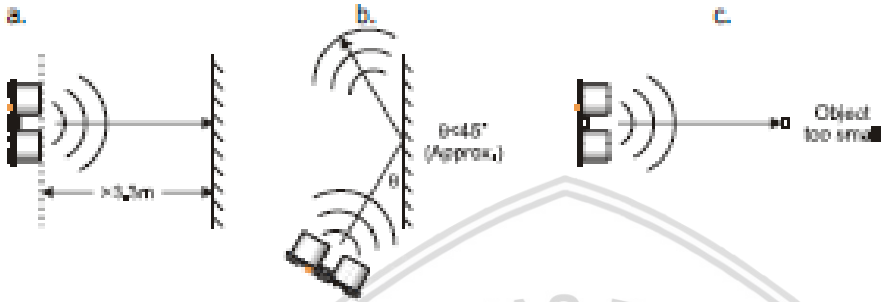


▬	Host Device	Input Trigger Pulse	t_{OUT}	2 µs (min), 5 µs typical
▬	PING))) Sensor	Echo Holdoff	$t_{HOLDOFF}$	750 µs
		Burst Frequency	t_{BURST}	200 µs @ 40 kHz
		Echo Return Pulse Minimum	$t_{IN\,MIN}$	115 µs
		Echo Return Pulse Maximum	$t_{IN\,MAX}$	18.5 ms
		Delay before next measurement		200 µs

Practical Considerations for Use

Object Positioning

The PING))) sensor cannot accurately measure the distance to an object that: a) is more than 3 meters away, b) that has its reflective surface at a shallow angle so that sound will not be reflected back towards the sensor, or c) is too small to reflect enough sound back to the sensor. In addition, if your PING))) sensor is mounted low on your device, you may detect sound reflecting off of the floor.



Target Object Material

In addition, objects that absorb sound or have a soft or irregular surface, such as a stuffed animal, may not reflect enough sound to be detected accurately. The PING))) sensor will detect the surface of water, however it is not rated for outdoor use or continual use in a wet environment. Condensation on its transducers may affect performance and lifespan of the device.

Air Temperature

Temperature has an effect on the speed of sound in air that is measurable by the PING))) sensor. If the temperature (°C) is known, the formula is:

$$C_{air} = 331.5 + (0.6 \times T_C)\ m/s$$

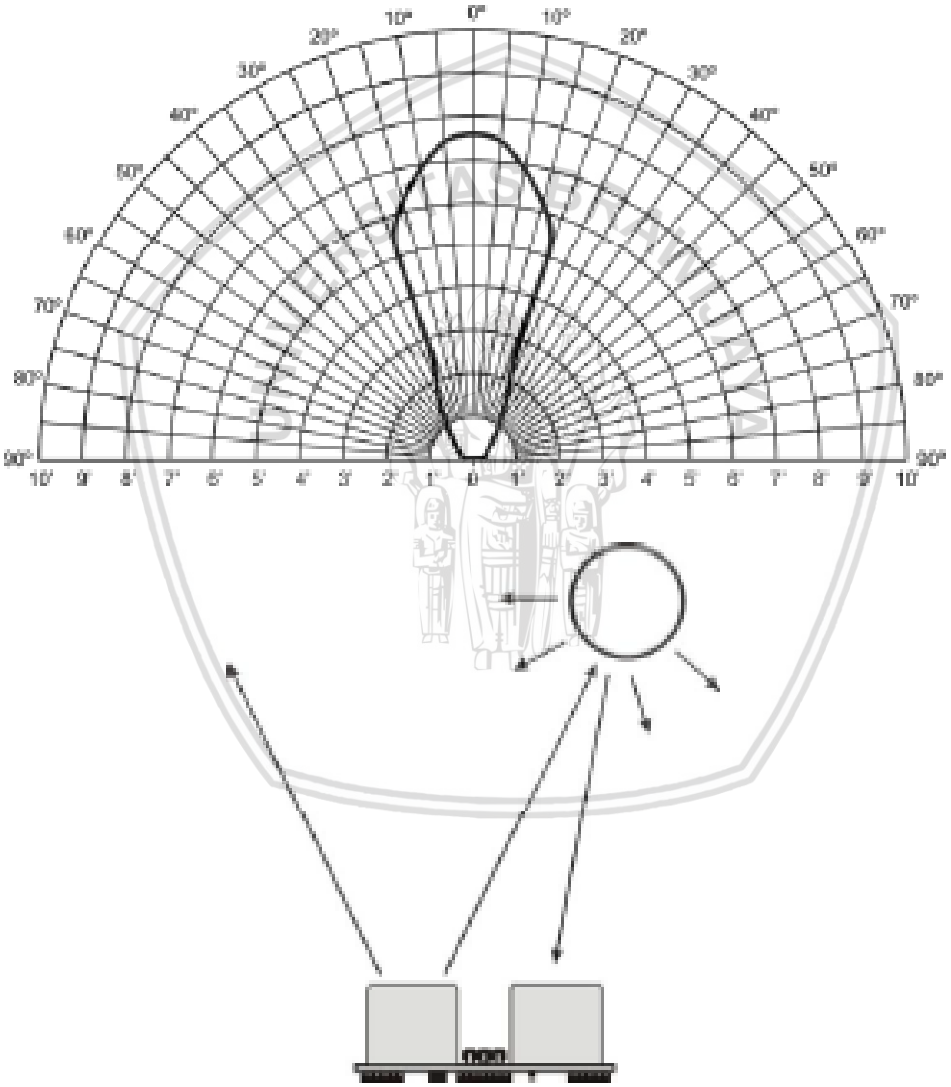
The percent error over the sensor's operating range of 0 to 70 ° C is significant, in the magnitude of 11 to 12 percent. The use of conversion constants to account for air temperature may be incorporated into your program (as is the case in the example BS2 program given in the Example Programs section below). Percent error and conversion constant calculations are introduced in Chapter 2 of *Smart Sensors and Applications*, a Stamps in Class text available for download from the 28029 product page at www.parallax.com.

Test Data

The test data on the following pages is based on the PING))) sensor, tested in the Parallax lab, while connected to a BASIC Stamp microcontroller module. The test surface was a linoleum floor, so the sensor was elevated to minimize floor reflections in the data. All tests were conducted at room temperature, indoors, in a protected environment. The target was always centered at the same elevation as the PING))) sensor.

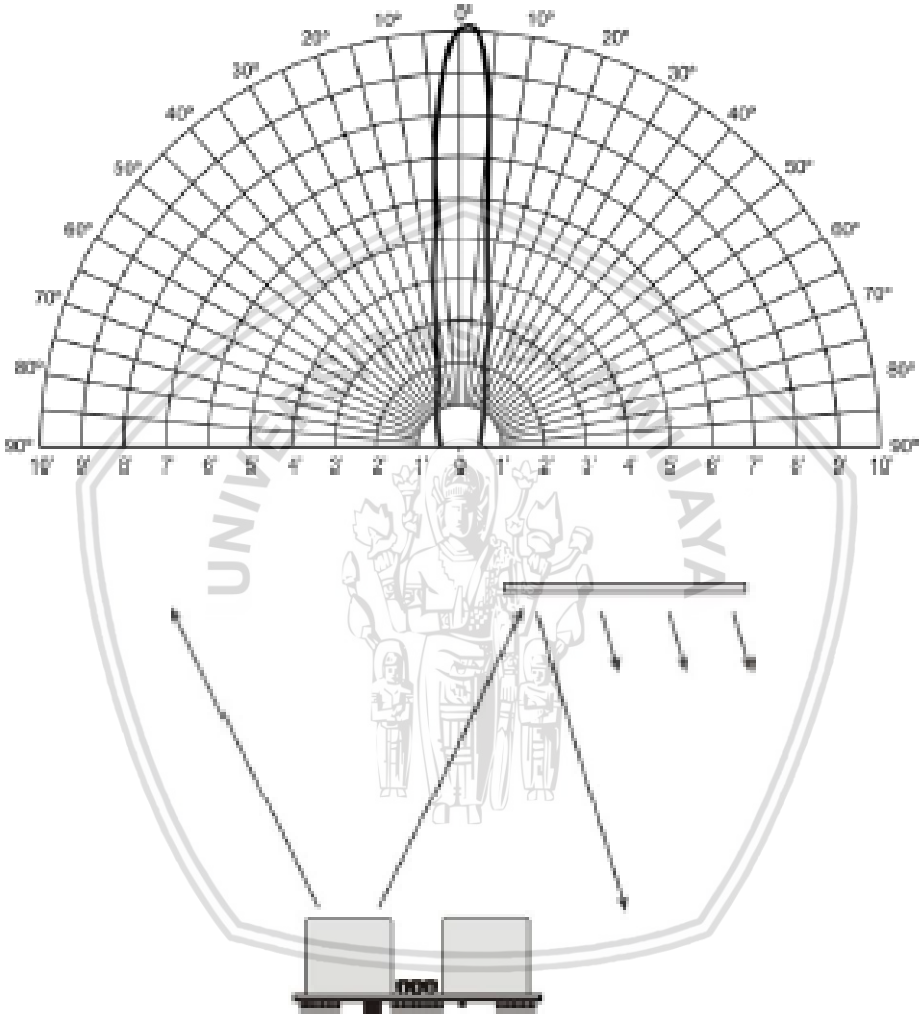
Test 1

Sensor Elevation: 40 in. (101.6 cm)
Target: 3.5 in. (8.9 cm) diameter cylinder, 4 ft. (121.9 cm) tall – vertical orientation



Test 2

Sensor Elevation:	40 in. (101.6 cm)
Target:	12 in. x 12 in. (30.5 cm x 30.5 cm) cardboard, mounted on 1 in. (2.5 cm) pole Target positioned parallel to backplane of sensor



3. Datasheet Servo Dynamixel AX18A Smart Servo

ROBOTIS e-Manual v1.29.00

AX-18F/ AX-18A

Part Photo

[AX-18F]

[AX-18A]

※ AX-18A is a new version of the AX-18F with the same performance but more advanced external design.

H/W Specification

- Weight : 54.5g (AX-18F), 55.9g(AX-18A)
- Dimension : 32mm * 50mm * 40mm
- Resolution : 0.29°
- Gear Reduction Ratio : 254 : 1
- Stall Torque : 1.8N.m (at 12.0V, 2.2A)
- No load speed : 97rpm (at 12V)
- Running Degree
  - 0° ~ 300°
  - Endless Turn
- Running Temperature : -5℃ ~ +75℃
- Voltage : 9 ~ 12V (Recommended Voltage 11.1V)
- Command Signal : Digital Packet
- Protocol Type : Half duplex Asynchronous Serial Communication (8bit,1stop,No Parity)
- Link (Physical) : TTL Level Multi Drop (daisy chain type Connector)
- ID : 254 ID (0~253)
- Communication Speed : 7343bps ~ 1 Mbps
- Feedback : Position, Temperature, Load, Input Voltage, etc.
- Material : Engineering Plastic

Stall torque is the maximum instantaneous and static torque
Stable motions are possible with robots designed for loads with 1/5 or less of the stall torque

Control Table

Control Table consists of data regarding the current status and operation, which exists inside of Dynamixel. The user can control Dynamixel by changing data of Control Table via Instruction Packet.

EEPROM and RAM

Data in RAM area is reset to the initial value whenever the power is turned on while data in EEPROM area is kept once the value is set even if the power is turned off.

Address

It represents the location of data. To read from or write data to Control Table, the user should assign the correct address in the Instruction Packet.

Access

Dynamixel has two kinds of data: Read-only data, which is mainly used for sensing, and Read-and-Write data, which is used for driving.

Initial Value

In case of data in the EEPROM Area, the initial values on the right side of the below Control Table are the factory default settings. In case of data in the RAM Area, the initial values on the right side of the above Control Tables are the ones when the power is turned on.

Highest/Lowest Byte

In the Control table, some data share the same name, but they are attached with (L) or (H) at the end of each name to distinguish the address. This data requires 16bit, but it is divided into 8bit each for the addresses (low) and (high). These two addresses should be written with one Instruction Packet at the same time.

Area	Address (Hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
EEPROM	0 (0X00)	Model Number(L)	Lowest byte of model number	R	18(0X12)
	1 (0X01)	Model Number(H)	Highest byte of model number	R	0 (0X00)
	2 (0X02)	Version of Firmware	Information on the version of firmware	R	-
	3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
	4 (0X04)	Baud Rate	Baud Rate of Dynamixel	RW	1 (0X01)
	5 (0X05)	Return Delay Time	Return Delay Time	RW	250 (0XFA)
	6 (0X06)	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit	RW	0 (0X00)
	7 (0X07)	CW Angle Limit(H)	Highest byte of clockwise Angle Limit	RW	0 (0X00)
	8 (0X08)	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit	RW	255 (0XFF)
	9 (0X09)	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit	RW	3 (0X03)
	11 (0X0B)	the Highest Limit Temperature	Internal Limit Temperature	RW	75 (0X46)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 (0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	140 (0X8C)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	215 (0XD7)
	15 (0X0F)	Max Torque(H)	Highest byte of Max. Torque	RW	3 (0X03)
	16 (0X10)	Status Return Level	Status Return Level	RW	2 (0X02)
	17 (0X11)	Alarm LED	LED for Alarm	RW	36(0x24)
RAM	24 (0X18)	Torque Enable	Torque On/Off	RW	0 (0X00)
	25 (0X19)	LED	LED On/Off	RW	0 (0X00)
	26 (0X1A)	CW Compliance Margin	CW Compliance margin	RW	1 (0X01)
	27 (0X1B)	CCW Compliance Margin	CCW Compliance margin	RW	1 (0X01)
	28 (0X1C)	CW Compliance Slope	CW Compliance slope	RW	32 (0X20)
	29 (0X1D)	CCW Compliance Slope	CCW Compliance slope	RW	32 (0X20)
	30 (0X1E)	Goal Position(L)	Lowest byte of Goal Position	RW	-
	31 (0X1F)	Goal Position(H)	Highest byte of Goal Position	RW	-
	32 (0X20)	Moving Speed(L)	Lowest byte of Moving Speed (Moving Velocity)	RW	-
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed (Moving Velocity)	RW	-
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit (Goal Torque)	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit (Goal Torque)	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position (Present Velocity)	R	-
	37 (0X25)	Present Position(H)	Highest byte of Current Position (Present Velocity)	R	-
	38 (0X26)	Present Speed(L)	Lowest byte of Current Speed	R	-
	39 (0X27)	Present Speed(H)	Highest byte of Current Speed	R	-
	40 (0X28)	Present Load(L)	Lowest byte of Current Load	R	-
	41 (0X29)	Present Load(H)	Highest byte of Current Load	R	-
	42 (0X2A)	Present Voltage	Current Voltage	R	-
	43 (0X2B)	Present Temperature	Current Temperature	R	-
	44 (0X2C)	Registered	Means if Instruction is registered	R	0 (0X00)
	46 (0X2E)	Moving	Means if there is any movement	R	0 (0X00)
	47 (0X2F)	Lock	Locking EEPROM	RW	0 (0X00)
	48 (0X30)	Punch(L)	Lowest byte of Punch	RW	32 (0X20)
	49 (0X31)	Punch(H)	Highest byte of Punch	RW	0 (0X00)

Address Function Help

EEPROM Area

Model Number

It represents the Model Number.

Firmware Version

It represents the firmware version.

ID

It is a unique number to identify Dynamixel.

The range from 0 to 252 (0xFC) can be used, and, especially, 254(0xFE) is used as the Broadcast ID.

If the Broadcast ID is used to transmit Instruction Packet, we can command to all Dynamixels.

> Please be cautious not to have the same IDs for the connected dynamixels. You may face communication issues or may not be able to search when IDs overlap.

Baud Rate

It represents the communication speed. 0 to 254 (0xFE) can be used for it.

This speed is calculated by using the below formula.

Speed(BPS) = 2000000/(Data+1)

Data	Set BPS	Target BPS	Tolerance
1	1000000.0	1000000.0	0.000 %
3	500000.0	500000.0	0.000 %
4	400000.0	400000.0	0.000 %
7	250000.0	250000.0	0.000 %
9	200000.0	200000.0	0.000 %
16	117647.1	115200.0	-2.124 %
34	57142.9	57600.0	0.794 %
103	19230.8	19200.0	-0.160 %
207	9615.4	9600.0	-0.160 %

Note : Maximum Baud Rate error of 3% is within the tolerance of UART communication.

Return Delay Time

It is the delay time per data value that takes from the transmission of Instruction Packet until the return of Status Packet.

0 to 254 (0xFE) can be used, and the delay time per data value is 2 usec.

That is to say, if the data value is 10, 20 usec is delayed. The initial value is 250 (0xFA) (i.e., 0.5 msec).

CW/CCW Angle Limit

The angle limit allows the motion to be restrained.

The range and the unit of the value is the same as Goal Position(Address 30, 31).

- CW Angle Limit: the minimum value of Goal Position(Address 30, 31)
- CCW Angle Limit: the maximum value of Goal Position(Address 30, 31)

The following two modes can be set pursuant to the value of CW and CCW.

Operation Type	CW / CCW
Wheel Mode	both are 0
Joint Mode	neither at 0

The wheel mode can be used to wheel-type operation robots since motors of the robots spin infinitely.

The joint mode can be used to multi-joints robot since the robots can be controlled with specific angles.

The Highest Limit Temperature

Caution : Do not set the temperature lower/higher than the default value.
When the temperature alarm shutdown occurs, wait 20 minutes to cool the temperature before re-use.
Using the product when the temperature is high may and can cause damage.

The Lowest (Highest) Limit Voltage

It is the operation range of voltage.

50 to 250 (0x32 ~ 0x96) can be used. The unit is 0.1V.

For example, if the value is 80, it is 8V.

If Present Voltage (Address42) is out of the range, Voltage Range Error Bit (Bit0) of Status Packet is returned as '1' and Alarm is triggered as set in the addresses 17 and 18.

Max Torque

It is the torque value of maximum output. 0 to 1023 (0x3FF) can be used, and the unit is about 0.1%.

For example, Data 1023 (0x3FF) means that Dynamixel will use 100% of the maximum torque it can produce while Data 512 (0x200) means that Dynamixel will use 50% of the maximum torque. When the power is turned on, Torque Limit (Addresses 34 and 35) uses the value as the initial value.

Status Return Level

It decides how to return Status Packet. There are three ways like the below table.

Value	Return of Status Packet
0	No return against all commands (Except PING Command)
1	Return only for the READ command
2	Return for all commands

When Instruction Packet is Broadcast ID, Status Packet is not returned regardless of Status Return Level.

Alarm LED

Alarm Shutdown

Dynamixel can protect itself by detecting errors occur during the operation.

The errors can be set are as the table below.

Bit	Name	Contents
Bit 7	0	-
Bit 6	Instruction Error	When undefined Instruction is transmitted or the Action command is delivered without the reg_write command
Bit 5	Overload Error	When the current load cannot be controlled with the set maximum torque
Bit 4	CheckSum Error	When the Checksum of the transmitted Instruction Packet is invalid
Bit 3	Range Error	When the command is given beyond the range of usage
Bit 2	OverHeating Error	When the internal temperature is out of the range of operating temperature set in the Control Table
Bit 1	Angle Limit Error	When Goal Position is written with the value that is not between CW Angle Limit and CCW Angle Limit
Bit 0	Input Voltage Error	When the applied voltage is out of the range of operating voltage set in the Control Table

It is possible to make duplicate set since the function of each bit is run by the logic of 'OR'. That is, if 0X05 (binary 00000101) is set, both Input Voltage Error and Overheating Error can be detected.

If errors occur, in case of Alarm LED, the LED blinks; in case of Alarm Shutdown, the motor output becomes 0 % by making the value of Torque Limit(Address 34, 35) as 0.

RAM Area

Torque Enable

Value	Meaning
0	Keeps Torque from generating by interrupting the power of motor.
1	Generates Torque by impressing the power to the motor.

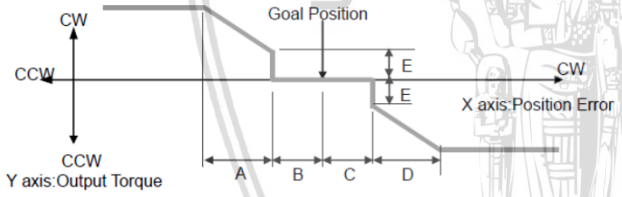
LED

Bit	Meaning
0	Turn OFF the LED
1	Turn ON the LED

Compliance

Compliance is to set the control flexibility of the motor.

The following diagram shows the relationship between output torque and position of the motor.



A : CCW Compliance Slope(Address0x1D)
B : CCW Compliance Margin(Address0x1B)
C : CW Compliance Margin(Address0x1A)
D : CW Compliance Slope (Address0x1C)
E : Punch(Address0x30,31)

Compliance Margin

It exists in each direction of CW/CCW and means the error between goal position and present position.

The range of the value is 0~255, and the unit is the same as Goal Position.(Address 30,31)

The greater the value, the more difference occurs.

Compliance Slope

It exists in each direction of CW/CCW and sets the level of Torque near the goal position.

Compliance Slope is set in 7 steps, the higher the value, the more flexibility is obtained.

Data representative value is actually used value. That is, even if the value is set to 25, 16 is used internally as the representative value.

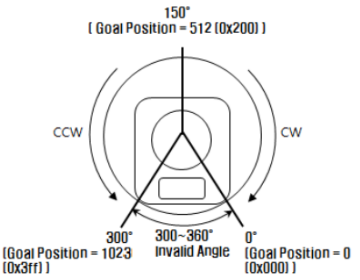
Step	Data Value	Data Representative Value
1	0 (0x00) ~ 3(0x03)	2 (0x02)
2	4(0x04) ~ 7(0x07)	4 (0x04)
3	8(0x08)~15(0x0F)	8 (0x08)
4	16(0x10)~31(0x1F)	16 (0x10)
5	32(0x20)~63(0x3F)	32 (0x20)
6	64(0x40)~127(0x7F)	64 (0x40)
7	128(0x80)~254(0xFE)	128 (0x80)

Goal Position

It is a position value of destination.

0 to 1023 (0x3FF) is available. The unit is 0.29 degree.

If Goal Position is out of the range, Angle Limit Error Bit (Bit1) of Status Packet is returned as '1' and Alarm is triggered as set in Alarm LED/Shutdown.



<The picture above is based on the front of relevant model>

If it is set to Wheel Mode, this value is not used.

Moving Speed

It is a moving speed to Goal Position.

The range and the unit of the value may vary depending on the operation mode.

- Join Mode

  0~1023 (0X3FF) can be used, and the unit is about 0.111rpm.

  If it is set to 0, it means the maximum rpm of the motor is used without controlling the speed.

  If it is 1023, it is about 114rpm.

  For example, if it is set to 300, it is about 33.3 rpm.

Notes: Please check the maximum rpm of relevant model in Joint Mode. Even if the motor is set to more than maximum rpm, it cannot generate the torque more than the maximum rpm.

- Wheel Mode

  0~2047(0X7FF) can be used, the unit is about 0.1%.

  If a value in the range of 0~1023 is used, it is stopped by setting to 0 while rotating to CCW direction.

  If a value in the range of 1024~2047 is used, it is stopped by setting to 1024 while rotating to CW direction.

  That is, the 10th bit becomes the direction bit to control the direction.

  In Wheel Mode, only the output control is possible, not speed.

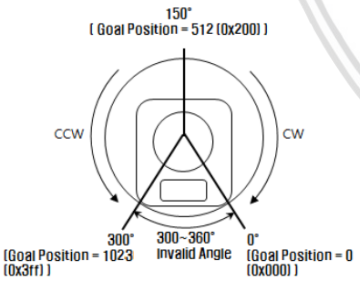
  For example, if it is set to 512, it means the output is controlled by 50% of the maximum output.

Torque Limit

Present Position

It is the current position value of Dynamixel.

The range of the value is 0~1023 (0x3FF), and the unit is 0.29 degree.



<The picture above is based on the front of relevant model>

Caution: If it is set to Wheel Mode, the value cannot be used to measure the moving distance and the rotation frequency.

Present Speed

It is the current moving speed.

0~2047 (0X7FF) can be used.

If a value is in the rage of 0~1023, it means that the motor rotates to the CCW direction.

If a value is in the rage of 1024~2047, it means that the motor rotates to the CW direction.

That is, the 10th bit becomes the direction bit to control the direction, and 0 and 1024 are equal.

The unit of this value varies depending on operation mode.

- Joint Mode

  The unit is about 0.111rpm.

  For example, if it is set to 300, it means that the motor is moving to the CCW direction at a rate of about 33.3rpm.
- Wheel Mode

  The unit is about 0.1%.

  For example, if it is set to 512, it means that the torque is controlled by 50% of the maximum torque to the CCW direction.

Present Load

It means currently applied load.

The range of the value is 0~2047, and the unit is about 0.1%.

If the value is 0~1023, it means the load works to the CCW direction.

If the value is 1024~2047, it means the load works to the CW direction.

That is, the 10th bit becomes the direction bit to control the direction, and 1024 is equal to 0.

For example, the value is 512, it means the load is detected in the direction of CCW about 50% of the maximum torque.

BIT	15~11	10	9	8	7	6	5	4	3	2	1	0
Value	0	Load Direction	Data (Load Ratio)									

Load Direction = 0 : CCW Load, Load Direction = 1: CW Load

Notes: Present load is an inferred value based on the internal output value; not a measured value using torque sensor, etc. Therefore, it may be inaccurate for measuring weight or torque. It is recommended to use it for predicting the direction and size of the force being applied to the joint.

Present Voltage

It is the size of the current voltage supplied.

This value is 10 times larger than the actual voltage. For example, when 10V is supplied, the data value is 100 (0x64)

Present Temperature

It is the internal temperature of Dynamixel in Celsius.

Data value is identical to the actual temperature in Celsius. For example, if the data value is 85 (0x55), the current internal temperature is 85°C.

Registered Instruction

Value	Meaning
0	There are no commands transmitted by REG_WRITE
1	There are commands transmitted by REG_WRITE.

Notes: If ACTION command is executed, the value is changed into 0.

Moving

Value	Meaning
0	Goal position command execution is completed.
1	Goal position command execution is in progress.

Lock

Value	Meaning
0	EEPROM area can be modified.
1	EEPROM area cannot be modified.

Caution: If Lock is set to 1, the power must be turned off and then turned on again to change into 0.

Punch

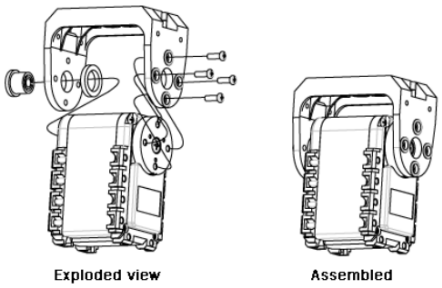
Current to drive motor is at minimum.

Can choose vales from 0x20 to 0x3FF.

Option Frame

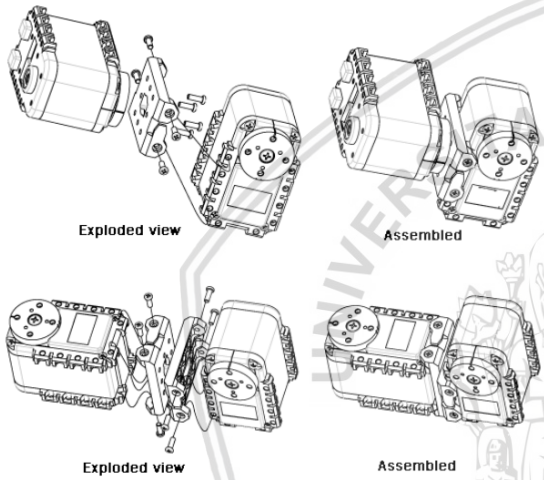
Applying F2

F2 is applied as below.



Applying F3

F3 is applied as below. F3 can be connected to 3 sides of AX-18A: Left, Right, and the bottom.



Dimension