TUGAS AKHIR – TM 145502

APLIKASI SENSOR *INERTIA MEASUREMENT UNIT* (IMU) UNTUK MEMPERBAIKI GERAK BERJALAN LURUS PADA ROBOT QUADRUPED

BAGOES PRAWIRA N
NRP 10211500000057

Dosen Pembimbing
Hendro Nurhadi, Dipl. Ing., PhD
NIP. 19751120 200212 1 002

PROGRAM STUDI DIPLOMA III TEKNIK MESIN
DEPARTEMEN TEKNIK MESIN INDUSTRI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Halaman ini sengaja dikosongkan

FINAL PROJECT – TM 145502

INERTIA MEASUREMENT UNIT (IMU) SENSOR APPLICATION TO IMPROVE WALKING STRAIGHT MOTION ON QUADRUPED ROBOT

BAGOES PRAWIRA N
NRP 10211500000057

Advisor
Hendro Nurhadi, Dipl. Ing., PhD
NIP. 19751120 200212 1 002

STUDY PROGRAM DIPLOMA III MECHANICAL ENGINEERING
DEPARTEMENT OF MECHANICAL ENGINEERING INDUSTRY
Faculty of Vocation
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Halaman ini sengaja dikosongkan

APLIKASI SENSOR *INERTIA MEASUREMENT UNIT* (IMU) UNTUK MEMPERBAIKI GERAK BERJALAN LURUS PADA ROBOT QUADRUPED

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Ahli Madya
Pada Bidang Studi Manufaktur
Program Studi Diploma III
Departemen Teknik Mesin Industri
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya

Oleh :
BAGOES PRAWIRA NATAKUSUMA
NRP. 10211500000057

Menyetujui,
Dosen Pembimbing Tugas Akhir

Dosen Pembimbing

Hendro Nurhadi, Dipl. Ing., PhD
NIP. 19751120 200212 1 002

SURABAYA, JULI 2018

Halaman ini sengaja dikosongkan

APLIKASI SENSOR IMU PADA ROBOT QUADRUPED UNTUK MEMPERBAIKI GERAK BERJALAN LURUS

Nama : Bagoes Prawira Natakusuma
NRP : 10211500000057
Departemen : Teknik Mesin Industri FV-ITS
Dosen Pembimbing : Hendro Nurhadi, Dipl.-Ing, Ph.D

ABSTRAK

Sensor *unit pengukuran inersia* atau *inersia measurement unit* (IMU) merupakan sebuah tranducer atau unit sensor (gabungan dari beberapa sensor) yang memanfaatkan sistem pengukuran seperti akselerometer, giroskop, dan magnetometer yang digunakan untuk mengukur perubahan posisi objek pada titik acuan tertentu. Selanjutnya untuk mewujudkan sistem sensor tersebut dengan ukuran yang sangant kecil, maka digunakan teknologi MEMS yang merupakan sebuah miniatur sistem elektronik dan mekanik dengan ukuran mikro.

Pada tugas akhir ini akan dibahas mengenai perancangan sistem sensor IMU yang diterapkan pada robot berkaki empat (*quadruped*) mulai dari perancangan sistem mekanik, sistem elektronik, hingga perancangan sistem kontrol proporsional, integral, dan derivative (PID) pada robot dengan menggunakan sensor IMU sebagai umpan balik (*feedback*) dari pergerakan sistem tersebut. Tujuan pengaplikasian sensor IMU pada robot tersebut adalah untuk memperbaiki gerak berjalan lurus pada robot agar tetap lurus, karena kurang presisinya sistem mekanik pada robot sehingga menybabkan robot berbelok saat melakukan gerak berjalan lurus.

Pada pengujian karakteristik modul sensor GY-521 didapat kesalahan pembacaan posisi sudut yaw rata-rata oleh sensor adalah 4.05° dengan kesalahan tertinggi adalah 10°, dan kesalahan rata-rata pada CMPS11 adalah 4,55° dengan kesalahan tertinggi adalah 10°. Pada pengaplikasikan modul sensor IMU GY-521 dengan kontrol PID untuk meluruskan gerak berjalan lurus, didapat kecepatan trayektory terbaik untuk robot adalah 60 cm/s dengan presentase kesalahan 3% dan terburuk adalah 30 cm/s dengan presentase kesalahan 13%.

Kata Kunci : IMU, sudut yaw , robot quadruped, kontroller PID

Halaman ini sengaja dikosongkan

IMU SENSOR APPLICATION TO IMPROVE WALKING STRAIGHT MOTION ON QUADRUPED ROBOT

Name : Bagoes Prawira Natakusuma
NRP : 10211500000057
Department : Mechanical Engineering Industry FV-ITS
Advisor : Hendro Nurhadi, Dipl.-Ing, Ph.D

ABSTRACT

The inertial measurement unit sensor (IMU) is a tranducer or sensor unit (a combination of multiple sensors) utilizing a measurement system such as an accelerometer, gyroscope, and magnetometer used to measure the change of position of an object at a particular reference point. Furthermore, to realize the sensor system with a small size, then used MEMS technology which is a miniature electronic and mechanical systems with micro size.

In this final project will be discussed about designing IMU sensor system applied to quadruped robot from design of mechanical system, electronic system, to design of proportional, integral, and derivative (PID) control system on robot using IMU sensor as feed feedback from the movement of the system. The purpose of applying IMU sensors to the robot is to fix the motion of walking straight on the robot to keep it straight, due to the lack of precision of the mechanical system in the robot so as to make the robot turn when the motion goes straight.

In the module GY-521 sensor analysis, the average angle reading error reading by the sensor was 4.05° with the highest being 10°, and the average error in CMPS11 was 4.55° with the highest error being 10°. In the IMU GY-521 modulator pengapection sensor with PID control to straighten the straight motion, the best speed for the robot is 60 cm / second with 3% error percentage and worst is 30 cm / s with error percentage of 13%.

Key words : IMU, yaw angle, quadruped robot, PID Controller

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT yang telah melimpahkan berkah, rahmat, serta hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul: Aplikasi Sensor IMU Pada Robot Quadruped Untuk Memperbaiki Gerak Berjalan Lurus.

Tugas Akhir ini merupakan persyaratan dalam menyelesaikan pendidikan di Departemen Teknik Mesin Industri. Tugas Akhir ini dibuat berdasarkan teori-teori yang didapat selama mengikuti perkuliahan, berbagai literatur dan pengarahan dosen. Pada kesempatan ini, penulis ingin berterima kasih kepada pihak-pihak yang membantu pembuatan tugas akhir ini, khususnya kepada:

1. Keluarga penulis, khususnya bapak dan ibu yang senantiasa memberikan dukungan baik dukungan material maupun non material.
2. Bapak Dr. Ir. Heru Mirmanto, MT., Selaku Kepala Departement Teknik Mesin Industri Fakultas Vokasi.
3. Bapak Ir. Suhariyanto, MT. Selaku koordinator Tugas Akhir Program Studi D3 Derpatmenet Teknik Mesin Industri Fakultas Vokasi-ITS
4. Bapak Hendro Nurhadi, Dipl.-ing, Ph.D, selaku dosen pembimbing dalam Tugas Akhir ini yang selalu memberikan bimbimngan dan arahan.
5. Teman - teman Departemen Teknik Mesin Industri 2015 yang selalu memberikan motivasi dalam mengerjakan Tugas Akhir.
6. Teman - teman UKM robot ITS dan tim KRPAI berkaki ITS ABINARA-1 yang telah banyak membantu dalam proses pengerjaan Tugas Akhir ini.

Penulis sadar bahwa Tugas Akhir ini belum sempurna dan masih banyak hal yang dapat diperbaiki. Saran, kritik dan masukan dari semua pihak sangat membantu penulis untuk pengembangan lebih lanjut. Terakhir, penulis berharap Tugas Akhir ini dapat memberikan manfaat bagi banyak pihak, khususnya dalam pengembangan aplikasi IMU.

Surabaya, 07 Juli 2017

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENNDAHULUAN

1.1. Latar Belakang

Sensor *unit pengukuran inersia* atau *inersia measurement unit* (IMU) merupakan sebuah unit sensor yang memanfaatkan sistem pengukuran seperti akselerometer, giroskop, dan magnetometer yang digunakan untuk mengukur perubahan posisi objek pada titik acuan tertentu. Pada tugas akhir ini, sensor IMU digunakan untuk mendeteksi offset kemiringan sumbu yaw pada pergerakan jalan lurus dari robot *quadrupet*. Karena dalam pembuatan sistem mekanik dari robot tersebut belum dapat sesempurna mungkin, sehingga hasilnya tidak sepresisi dari densain yang telah dibuat, maka akan mengakibatkan gerakan lurus pada robot *quadrupet* tersebut mengalami pembelokan pada sudut yaw. Sehingga permasalahan dalam robot tersebut adalah bagaimana cara membuat robot agar dapat tetap berjalan lurus walaupun sistem mekanik yang ada tidak begitu presisi dengan menerapkan sensor IMU pada robot sebagai *feedback* dari kesalahan yang terjadi dan memprosesnya dengan suatu sistem kontrol, yaitu sistem kontrol PID.

1.2. Perumusan Masalah

Permasalahan yang dibahas dalam tugas akhir ini adalah :

1. Bagaimana cara mengaplikasikan sensor IMU untuk memperbaiki gerak berjalan lurus pada robot quadruped yang digunakan dalam tugas akhir ini.
2. Bagaimana mendesain dan membangun sistem sensor IMU (mekanik dan elektronik) pada robot quadruped
3. Bagaimana mendesain sistem kontrol PID agar dapat mengaplikasikan sensor IMU pada robot quadruped ini.

1.3. Tujuan

Tujuan dari pelaksanaan tugas akhir ini adalah :

1. Didapatkannya sifat atau karakteristik sensor IMU dalam membaca nilai posisi sudut Yaw.
2. Didapatkannya desain sistem sensor IMU (mekanik dan elektronik) pada robot quadruped yang digunakan.

3. Robot dapat berjalan lurus dengan bantuan sensor IMU dan sistem kontrol PID.

1.4. Batasan Masalah

Batasan masalah dari tugas akhir ini adalah sebagai berikut :

1. Pengolahan data mentah (raw data) hasil sensor IMU agar menjadi sudut *yaw* masih menggunakan library pada Arduino IDE.
2. Hanya menggunakan orientasi sudut *yaw* dari sensor IMU untuk memperbaiki gerak berjalan lurus saja pada robot.
3. Tidak ada pengolahan data untuk memperbaiki kesalahan yang diakibatkan dari karakteristik sensor IMU.
4. Desain mekanik dan elektronik pada robot *quadruped* hanya dikhususkan untuk mengikuti Kontes Robot Pemadam Api (KRPAI) divisi berkaki.
5. Tidak terdapat perhitungan dinamika dari sistem mekanik robot tersebut.
6. Tidak terdapat perhitungan mekanika kekuatan material dari sistem mekanik robot tersebut.
7. Tidak terdapat analisa dan perhitungan pada sirkuit elektronik dari sistem elektronik robot tersebut.
8. Dalam sistem nilai Kp, Ki, dan Kd dari sistem kontrol PID menggunakan metode manual Tunning.
9. Lingkungan robot yang dipilih terbatas pada lingkungan yang datar

1.5. Metedologi

Langkah-langkah yang dikerjakan pada tugas akhir ini adalah sebagai berikut:

1. **Studi Literatur**
   Pada studi literatur dilakukan pengumpulan dasar teori yang menunjang dalam penulisan tugas akhir yang meliputi buku, jurnal, artikel, forum diskusi dan penelitian-penelitian sebelumnya.
2. **Perancangan Sistem**
   Perancangan sistem ini meliputi proses desain dan pembuatan mekanik, elektronik, algoritma program. Untuk desain mekanik dan elektronik mengacu pada aturan Kontes Robot Pemadam Api (KRPAI) divisi berkaki. Setelah semua proses pembuatan body dan elektronik selesai maka tahap selajutnya adalah proses

pengaksesan sensor IMU dan perumusan sistem kontrol pada robot tersebut.

3. **Pengujian Sistem**
   Pada bagian ini dilakukan untuk mengetahui hasil perancangan sistem yang telah dibuat sebelumnya. Pengujian dalam tugas akhir ini meliputi pengujian karakteristik sensor IMU, pengujian gerak berjalan lurus tanpa tanpa sistem kontro PID, dan pengujian berjalan lurus dengan sistem kontrol PID.
4. **Analisa**
   Pada bab ini dilakukan analisa dari hasil yang didapatkan dari pengujian sistem sebelumnya. Analisa tersebut meliputi perbandingan gerak robot berjalan lurus dengan dan tanpa sistem kontrol PID.
5. **Penyusunan Laporan Tugas Akhir**
   Penyusunan laporan tugas akhir adalah bagian terakhir dalam proses pengerjaan setelah dilakukan pengujian dan analisa data. Dalam hal ini akan berisikan hasil kegiatan yang telah dikerjakan yang meliputi pendahuluan, dasar teori penunjang, aplikasi perancangan, pengujian dan analisa serta kesimpulan.

1.6. Sistematika Penulisan

Pembahasan Tugas Akhir ini akan dibagi menjadi lima bab dengan sistematika sebagai berikut :

1. **BAB I Pendahuluan**
   Bab ini berisi uraian tentang latarbelakang permasalahan yang akan dikerjakan dalam tugas akhir ini, tujuan, batasan masalah yang akan dikerjakan, metodologi penelitian, sistematika penulisan dan relevansi.
2. **BAB II Dasar Teori**
   Pada bab ini berisi tentang uraian mengenai teori penunjang dan hal-hal yang nantinya dibutuhkan dalam proses penyelesaian tugas akhir.
3. **BAB III Perancangan Sistem**
   Bagian ini berisi tentang penjelasan terkait perencanaan sistem yang nantinya akan dikerjakan yang berupa perencanaan mekanik robot dan elektroniknya, perencanaan sensor IMU terhadap robot serta sistem kontrol yang akan diterapkan.
4. **BAB IV Pengujian dan Analisa**

Bab ini akan menjelaskan tentang pengujian dari sistem yang telah dirancang pada bab sebelumnya. Pengujian ini dilakukan untuk mendapatkan data yang kemudian di analisa hasil tersebut.

5. **BAB V Penutup**
   Bagian ini merupakan penutup yang berisi kesimpulan yang diambil dari pengujian dan analisa yang telah dilakukan. Saran dan kritik yang membangun untuk pengembangkan lebih lanjut juga dituliskan pada bab ini.

1.7. Relevansi dan Manfaat

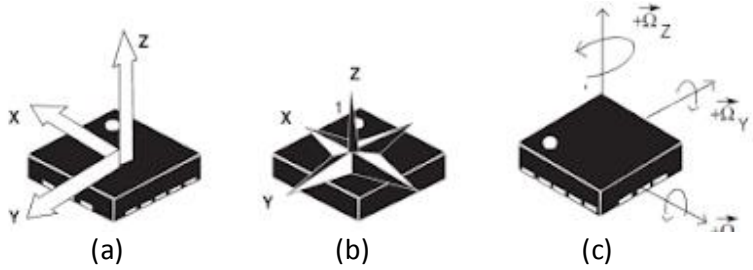
Hasil yang diharapkan dari tugas akhir ini adalah terbentuknya langkah kecil dalam riset untuk *mendigitalisasikan sebuah sistem mekanik* ke dalam suatu algoritma dan mengimplementasikannya dalam dunia nyata melalui sistem elektronik dan mekanik, sehingga sistem tersebut dapat bekerja secara otomatis dengan perintahnya masing-masing sebaik mungkin. Pada tugas akhir ini sensor IMU digunakan untuk membaca offset sudut yaw pada robot kerena kemencengan gerak berjalan lurus robot yang selanjutnya digunakan untuk sebagai feedback dalam sistem kontrol PID untuk memperbaiki gerak berjalan lurus tersebut agar tetap lurus.

BAB II
DASAR TEORI

Tinjauan Pustaka ini berisi tentang kumpulan informasi penunjang yang berhubungan dengan alat maupun sistem yang akan dibuat pada tugas akhir ini yang berasal dari berbagai sumber. Kajian teori yang dibahas pada bagian ini terdiri dari unit pengukuran inertial (IMU), kinematika gerak langkah robot, modul sensor IMU CMPS-11, modul sensor IMU GY-521, dan aktuator servo dynamixel RX-28.

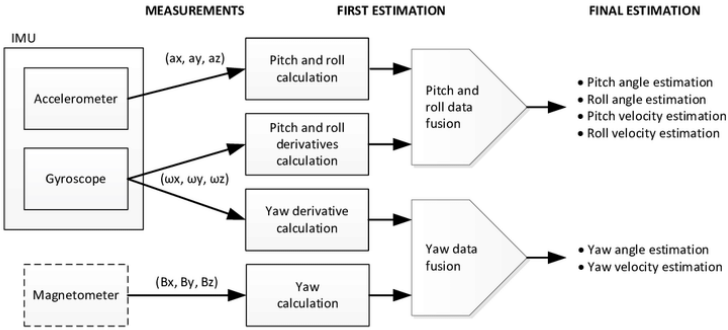
2.1. Unit Pengukuran Inersia

Pengertian Unit Pengukuran Inersia atau Inertial Measurement Unit (IMU) adalah suatu alat atau sensor MEMS yang mengkombinasikan sistem pengukuran seperti akselerometer, giroskop dan magnetometer yang dimana masing-masing memiliki tiga sumbu utama. Fungsi dari IMU sendiri adalah untuk memperkirakan posisi relatif, kecepatan, akselerasi, dan kemiringan dari objek terhadap orientasi tertentu.



Gambar 2.1. Ilustrasi Sensor IMU pada (a) akselerometer, (b) magnetometer, (c) giroskope [12]

Prinsip kerja dari IMU sendiri adalah memanfaatkan sifat inersia atau kelembaman (secara numerik diwakili oleh massa) sistem terhadap orientasi tertentu, dan memperhitungkan setiap perubahan nilai inersia secara terus menerus. Pada gambar 2.2 adalah Penggabungan data antara akselerometer, gyro, dan magnetometer untuk memperkirakan roll, pitch dan yaw.

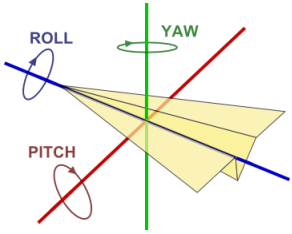


Gambar 2.2. Blok Diagram dari Kombinasi Tiga Sensor IMU

Berikut penjelasan block diagram pada gambar 2.2 di atas. Pada kolom blok paling awal (IMU) terdiri dari accelerometer yang berfugsi untuk membaca percepatan **a** (m/s^2) dan gyroscope yang berfungsi membaca kecepatan sudut **w** (rad/s) dan ada sensor magnetometer yang berfungsi membaca posisi deraajat arah mata angin yang semua nilai tersebut merupakan *raw data* (data mentah). Pada kolom blok kedua yaitu proses pengoloahan *raw data* menjadi nilai posisi sudut pada sumbu yaw, pitch, dan roll dari turunan pertama (kecepatan **a**) dan kedua (percepatan sudut **w**), untuk accelerometer hanya dapat menghasilkan nilai *pitch* dan *roll* saja, pada gyroscope dapat menghasilkan nilai *yaw*, *pitch*, dan *roll*, sedangkan magnetometer hanya menghasilkan nilai *yaw* saja yang berfungsi untuk memperbaiki nilai yaw gyroscope jika terjadi offset, karena pembacaan nilai sudut yaw oleh magnetometer memiliki acuan medan magnet bumi. Selanjutnya pada kolom yang ketiga adalah proses estimasi pertama yaitu proses penggabungan atau pengkombinasian nilai *yaw, pitch,* dan *roll* yang merupakan hasil dari beberapa sensor. Dimana untuk nilai posisi sudut *pitch* dan *roll* merupakan kombinasi dari sensor IMU accelerometer dan IMU gyroscope, dan posisi sudut *yaw* merupakan kombinasi dari sensor IMU accelerometer dan sensor magnetometer. Dan pada kolom block yang terakhir adalah hasil dari proses penggabungan tadi yaitu berupa posisi dan kecepatan sudut. Perhitungan matematika yang digunakan dalam pengolahan *raw data* ini adalah dengan implementasi *quarternion* dengan menggunakan *sudut euler* dan *matrix rotasi*.

Pada dasarnya sensor IMU dapat diaplikasikan untuk menghitung pergerakan dari sebuah sistem terhadap lingkungan seperti posisi, kecepatan, dan percepatan linier maupun anguler dari sebuah benda. Pada tugas akhir ini sensor IMU diaplikasikan untuk menghitung posisi angular (sudut) dari robot quadruped pada sumbu yaw, pitch, dan roll, berikut ini penjelasan mengenai ketiga sumbu tersebut :

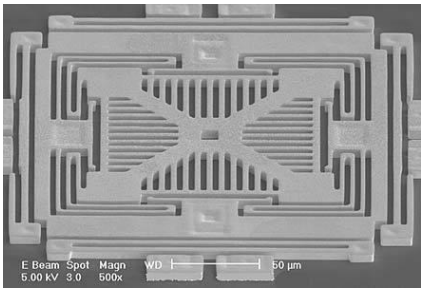
a) Sumbu Normal atau *Yaw*
   Sumbu dengan titik asal di pusat gravitasi yang ditarik dari atas ke bawah dan tegak lurus dengan dua sumbu lainnya. Gerak rotasi melalu sumbu ini adalah gerakan *yaw*.
b) Sumbu Lateral atau *Pitch*
   Sumbu dengan titik asal di pusat gravitasi yang ditarik dari samping kanan ke samping kiri dan sejajar dengan bidang horizontal. Gerak rotasi pada sumbu ini disebut gerakan *pitc*.
c) Sumbu logitudinal atau *Roll*
   Sumbu dengan titik asal di pusat gravitasi yang ditarik dari depan ke belakang dan sejajar dengan bidang horizontal. Gerak rotasi pada sumbu ini disebut gerakan *roll*.



Gambar 2.2. Ilustrasi Sumbu Utama Pada Sistem Koordinat Kartesius

2.2. Micro Electro Mechanical System

Micro Electro Mechanical System (MEMS) didefenisikan sebagai sebuah miniatur perangkat atau susunan perangkat yang menggabungkan elemen miniatur elektronik dan elemen miniatur mekanik dengan menggunakan teknik microfabrication dalam pembuatannya. Proposal istilah dan pengembangan sistem MEMS ini pertamakali diajukan pada tahun 1986 oleh University of Utah yang ditujukan kepada Defense Advanced Reseach Project Agency (DARPA).



Gambar 2.3. Salah Satu Struktur dari MEMS IMU

MEMS terdiri dari empat komponen utama, yaitu :

- Microsensor
  Berfungsi untuk mendeteksi perubahan sistem atau lingkungan dengan pengukuran mekanik, therma, magnetik, kima, informasi elektromagnetik, dan lain sebgaianya. Contohnya : sensor suhu, sensor radiasi, sensor mekanik, sensor magnetik, dsb.

- Microelectronic
  Bertugas untuk memproses informasi dan memberikan sinyal, dan memindahkan mikro untuk bekerja sama dengan satu sama lain dan untuk melakukan tugas uang diberikan. Contohnya : control system, signa transmission, signal processing, driver circuit, dsb.

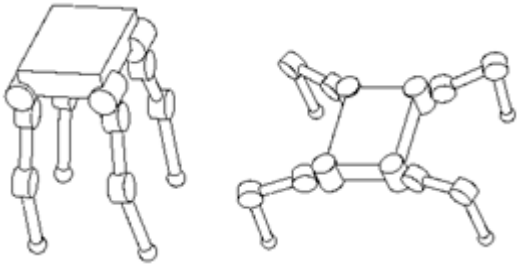
- Microactuator
  Berfungsi untuk bereaksi dan memberikan beberapa bentuk perubahan lingkungan seteleh mendapatkan sinyal dari microelectronics. Contohnya : actuator magnetic, actuator suhu, actuator piezelectric, dsb.

- Mechanical Microstructure
  Pada komponen ini berfungsi untuk mengontrol pergerakan didalam MEMS. Contohnya : micormirror, microvalve, microfluid, dan elemen mekanik lainnya.

Untuk proses fabrikasi atau pembuatan MEMS sendiri dilakukan degan teknik bulk micromachining, surface micromachining, high aspec ratio (HAR) silicon micromachining.

2.3. Quadruped Robot

Quadruped robot adalah tipe robot berkaki yang memiliki empat kaki dengan postur ponograd atau quadrupedalisme. Bentuk tersebut adalah bentuk lokomosi terestrial pada hewan memakai empat paha atau lutut. Qudaruped robot ini dibedakan menjadi dua jenis yaitu tipe mammal dan tipe sprawling. Pada tipe mamalia kaki robot menempel secara vertikal ke bawah dari pangkal kaki sebagai postur standar, sedangkan pada tipe sprawling kaki pertama (paha) berada dalam posisi arah horizontal, dan kaki kedual (betis) berada dalam arah vertikal sebagai postur standar. [6]



Gambar 2.4. Tipe Mammal (kiri) dan Tipe Sprawling (kanan) [6]

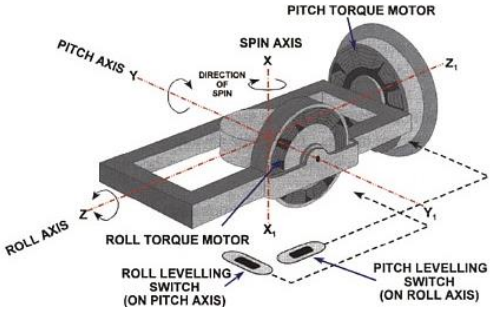
2.4. Giroskop

Giroskop secara umum adalah perangkat yang mengukur baik perubahan sudut atau tingkat perputaran sudut dari orientasi perangkat terhadap acuan tertentu. Prinsip kerja dari giroskop adalah berdasarkan hukum kekekalan momentum angular yang berbunyi "dalam sistem partikel apa pun, momentum sudut total sistem relatif terhadap titik mana pun yang tetap dalam ruang tetap konstan, asalkan tidak ada gaya eksternal yang bekerja pada sistem".

Kebanyakan giroskop hanya mengukur sepanjang satu sumbu sensitif dan tidak dipengaruhi oleh gravitasi bumi. Oleh karena itu, kombinasi dari tiga giroskop yang dipasang secara orthogonal memerlukan pengartian kembali gerakan sudut tiga dimensi. Untuk nilai gyroscopes, orientasi suatu objek dapat diperoleh dengan integrasi dari ukuran kecepatan sudut dalam satuan rotasi per menit (RAW). Secara umum gyroscope dibagi menjadi dua jenis yaitu :

2.4.1. Giroskop Mekanik

Giroskop tingkat mekanis menggunakan motor listrik untuk memutar disk kecil atau roda gila yang bisa berputar (pivot) pada satu sumbu dan memiliki mata angin untuk kembali ke pusat. Ketika giroskop dipindahkan dari sumbu yang sensitif, disk berputar miring dan tingkat kemiringan ini dijadikan umpan balik secara elektronik oleh sebuah ptentiomenter. Semakin cepat putaran giroskop, semakin besar lendutan yang ada dan berdasarkan lendutan tersebut, sinyal perbaikan dapat dimasukkan ke dalam motor listrik. Pada gambar 2.5 adalah salah satu contoh skematik giroskop mekanik dengan motor listrik.



Gambar 2.5. Giroskop Mekanik dengan Motor DC

2.4.2. Giroskop MEMS

Giroskop MEMS memiliki fungsi untuk mengukur kecepatan sudut pada sumbu acuan yang tetap (sumbu x, y, dan z) terhadap ruang inersia berukuran mikro. Prinsip kerja dari giroskop MEMS ini didasarkan pada elemen getar pada tingkat yang tinggi untuk merasakan adanya kecepatan sudut pada masing masing sumbu. Teori yang diterapan untuk mengukur kecepatan sudut tersebut adalah dengan menggunakan Effect Corilois, Effect Corilois adalah pembelokkan arah benda yang bergerak ketika dilihat dari kerangka acuan yang berputar, secara matematis dapat dituliskan sebagai berikut :

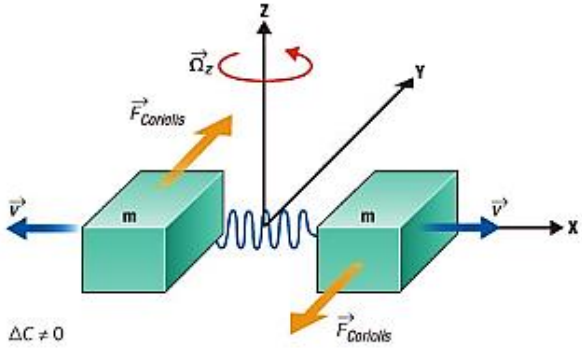
$$\boldsymbol{F}_{Coriolis} = -2m\boldsymbol{\Omega} \times \mathbf{v} \qquad (2.1)$$

Dimana :

- $\mathbf{F_{corilois}}$: Gaya Coirolis (N)
- **m** : Massa obyek yang diterapkan (Kg)

- **v** : Kecepatan (m/s)
- **Ω** : Kecepatan angular yang terjadi (rad/s)

Dari rumus tersebut, maka untuk implementasinya pada MEMS giroskop dapat dilihat pada ilustrasi di bawah ini (gambar 2.6)



Gambar 2.6. Ilustrasi Efek Coirolis Pada MEMS giroskop [16]

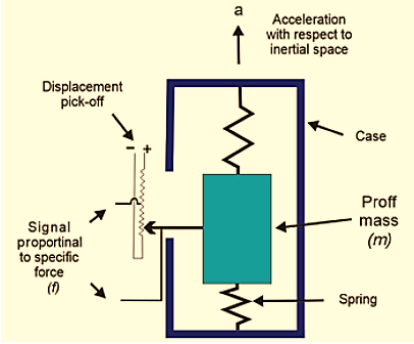
Dari ilustrasi di atas dapat dijelaskan bahwa ketika sebuah massa uji **m** yang saling bergetar dan berpindah secara konstan pada arah yang berlawanan yang bergerak dengan kecepatan dan arah **v** yang selanjutnya diterapkan perpindahan rotasi sebesar **Ω**, maka akan mengalami suatu gaya (arah panah oranye), gaya tersebut adalah *gaya coirolis*. [16]

2.5. Akselerometer

Akselerometer adalah suatu alat yang digunakan untuk mengukur percepatan suatu sistem terhadap lingkungan, sehingga dapat mendeteksi adanya perubahan posisi dan kecepatan sistem dan seberapa banyak perubahan yang terjadi pada sistem tersebut. Prinsip kerja alat ini yaitu dengan memanfaatkan hukum hukum kedua Newton tentang gerak (F = m x a), sehingga dalam akselerometer terdapat massa yang diterapkan pada alat tersebut dan suatu instrument khusus untuk mengukur perpindahan massa tersebut. Secara umum akselerometer dibagi menjadi dua jenis, yaitu akselerometer mekanik, dan MEMS akselerometer.

2.4.1. Akselerometer Mekanik

Akselerometer mekanik umumnya terdiri dari tiga komponen utama, yaitu massa uji, spring, dan potensiometer (alat instrument pengukur) pada alat dalam area tertutup (Gambar 2.6). Hukum fisika yang menjadi dasarnya adalah hukum Hooke dan hukum gerak Newton yang kedua.



Gambar 2.7. Model Akselerometer Mekanik
(diadopsi dari Titterton & Weston,2004) [7]

Pada gambar 2.6 dalam mengukur akselerasi dalam arah vertikal dapat menggunakan rumus **ap = a – g** (**ap** hasil akselerasi yang terjadi, **a** akselerasi, dan **g** percepatan gravitasi). Hukum Hooke menyatakan bahwa gaya yang diperlukan untuk memperpanjang atau menekan pegas sebanding dengan jarak **x** dari ekstensi atau kompresi yang diinginkan. Akibatnya, pegas akan menciptakan gaya dengan besaran yang sama untuk mengembalikan keadaan aslinya, bertindak dalam arah berlawanan dari gaya yang diterapkan. Gaya pemulih **F** dapat dijelaskan oleh **F = k * x**, di mana **k** adalah kekakuan pegas, hukum gerak Newton yang kedua menyatakan bahwa objek massa **m** yang dikenakan akselerasi, memberikan gaya **F** sehingga **F = m * a**, Karena itu dapat disimpulkan bahwa karena **F = m*a = k*x**, massa akselerometer **m** dipindahkan oleh $\mathbf{x = (m*a)*k^{-1}}$. Jika perpindahan, kekakuan pegas dan massa diketahui, maka akselerasi dapat dihitung menggunakan $\mathbf{a = (k* x)* m^{-1}}$**.** [7]

Sehingga, jika terjadi perubahan jarak terhadap massa **m** (gambar 2.7) maka pada alat instrumet (potensiometer) akan terjadi medan

magnet akibat arus yang melali kumparan. Arus ini sebanding dengan akselerasi, sehingga didapat akselerasi yang terjadi.

2.4.2. MEMS Akselerometer

MEMS Akselerometer memiliki fungsi untuk mengukur percepatan yang terjadi pada sebuah sistem dengan menggunakan unit pengukuran berukuran mikro. MEMS Akseleremoeter umumnya adalah tipe MEMS capacitive akselerometer. Struktur dari MEMS akselerometer ini terdiri dari jangkar (*anchor*) yang merupakan komponen tetap dan elemen sensing yang merupakan komponen yang bergerak (*proof mass*, *comb capacitor*, dan *springs*) yang terbuat dari kristal silikon tunggal.

Prinsip kerja dari kapasistansi akselerometer adalah dengan prinsip *self-balacing bridge* seperti pada gambar 2.8, saat akselerasi terjadi, akan menyebabkan gaya inersia pada *proof mass* yang akhirnya menyebabkan ia bergerak, yang kemudian diimbangi oleh gaya spring sebagai peneimbang saat akselerasi mulai menurun. Perpindahan pada *proof mass* tadi menyebabkan terjadinya perubahan kapasistansi pada C_1 dan C_2. Pada pengendali (*driver*) tersebut selalu mengirimkan sinyal AC kepada elektrode tetap (pada *anchor*) yang selanjutnya pada elektrode tidak tetap (pada *proof mass*) yang terhubung pada *terminal output* akan membaca sinyal perbandingan sinyal tersebut akibat perbedaan kapasitansi pada C_1 dan C_2. [8][3]

Maka persamaan untuk masing masing kapasitor dalam menghitung perubahan posisi tetap $\mathbf{x_0}$ terhadap $\mathbf{x}$ adalah :

$$C_1 = \varepsilon \frac{A}{x_0 + x} = \varepsilon \frac{A}{x_0.\left(1 + {}^{x}/_{x_0}\right)} = {}^{C_0}/_{1+\delta} \tag{2.2}$$

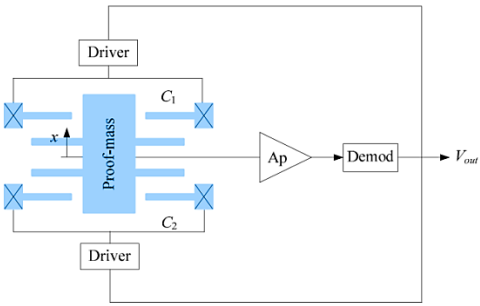
$$C_2 = \varepsilon \frac{A}{x_0 - x} = \varepsilon \frac{A}{x_0.\left(1 - {}^{x}/_{x_0}\right)} = {}^{C_0}/_{1-\delta} \tag{2.3}$$

Dalam jembatan *Wheastone* dimana jika $\delta << 1$, maka :

$$V_{out} = {}^{V_{in}}/_{2} \times \delta \tag{2.4}$$

Sehingga dari persamaan 2.2, 2.3, dan 2.4 didapa percepatan **a** :

$$a = {}^{K}/_{M}.x = {}^{K}/_{M}.x_0\delta = {}^{K}/_{M}.x_0.2.{}^{V_{out}}/_{V_{in}} \tag{2.5}$$



Gambar 2.8. Skema MEMS Akselerometer Kapasitif [8]

2.6. Magnetometer

Magnetometer adalah alat instrumen yang digunakan untuk mengukur kekuatan suatu medan magnet akibat material ferromagnetik, arah medan magnet, dan juga perubahan relatif medan magnet pada lokasi tertentu akibat efek induksi. Satuan internasional (SI) yang digunakan untuk mengukur kekuatan medan magnet adalah Tesla.

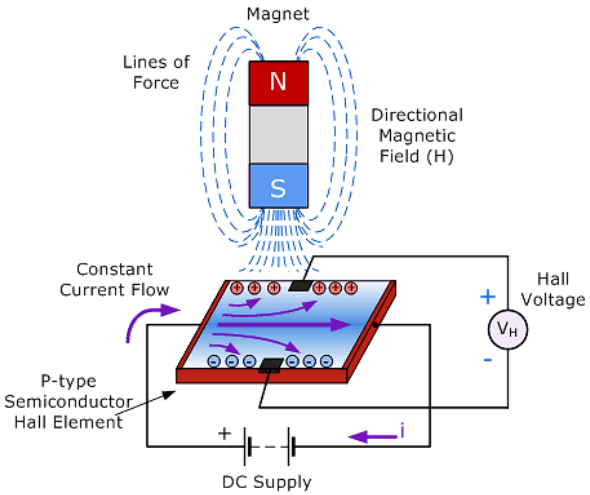
Gambar 2.9. Compass adalah tipe magnetometer yang paling sederhana

Medan magnet adalah jumlah vektor yang dicirikan oleh kekuatan dan arah. Dalam satuan SI kekuatan medan magnet diukur dalam satuan tesla , dan dalam gauss dalam sistem satuan cgs 10.000 gauss setara dengan satu tesla. Pengukuran medan magnet Bumi sering dikutip dalam satuan nanotesla (nT), juga disebut gamma. Medan magnet Bumi dapat bervariasi dari 20.000 hingga 80.000 nT tergantung pada lokasi, fluktuasi medan magnet Bumi berada pada urutan 100 nT, dan variasi medan magnet karena anomali magnetik dapat berada dalam rentang picotesla (pT).

Secara garis besar, bredasarkan pengukurannya magnetometer terdiri dari dua tipe yaitu magnetometer *vektor* (mengukur komponen vektor medan magnet) dan magnetometer *total field* atau magnetometer skalar (mengukur besar dari medan magnet vektor).

2.5.1. MEMS Magnetometer

MEMS Magnetometer adalah perangkat mikro-elektrikmekanikal sistem yang berfungsi untuk mengukur besar vektor medan magnet. Umumnya sistem ini beroprasi dengan mendeteksi dan mengukur efek *gaya lorenz*. Pada MEMS Magnetometer ini umumnya adalah magnetometer tipe *Hall effect*, yaitu dalam mengukur vektor medan magnet menggunakan Efek Hall.



Gambar 2.10. Ilustrasi Hall Effect Pada Elemen Semikonduktor [9]

Prinsip kerja dari efek hall ini adalah dimulai dari elemen hall yang terdiri dari sepotong tipis bahan semikonduktor tipe-P yang kemudian dilewatkan arus secara kontinue. Selanjutnya ketika elemen tersebut didekatkan dengan dengan medan magnet, maka mengakibatkan munculnya garis gaya fluks magnetik pada elemen, sehingga membelokan muatan dan elektron ke salah satu sisi pada elemen.

Tegangan output dari elemen tersebut disebut tegangan Hall (V_H) yang berbanding lurus dengan kekuatan medan magnet yang melewati

bahan semikonduktor (output $\propto$ H). Tegangan output ini sangatlah kecil, hanya berrukuran microvolts, sehingga ditambahkan amplifier DC, rangkaian switching logika dan regulator tegangan untuk meningkatkan sensitivitas sensor, dan output tegangan. Besar tengangan output ini sebanding dengan besarnya medan magnet yang melewati elemen Hall. Tegangan keluaran tersebut dapat dihitung berdasarkan persamaan berikut ini [9] :

$$V_H = R_H \left(\frac{I}{t} \times \frac{B}{10.000} \right) \quad (2.6)$$

Dimana :

- **V_H** : Tegangan Hall (Volt)
- **R_H** : Koefesien Efek Hall
- **I** : Arus yang melewati elemen Hall (A)
- **t** : ketebalan elemen Hall (mm)
- **B** : Densitas flux magnetik (gauss)

2.7. Kinematika Gerak Kaki Robot

Kinematika adalah sebuah studi dan analisa tentang pergerakan, kecepata, dan percepatan sebuah objek tanpa memperhatikan faktor-faktor yang menyebabkan gerak tersebut terjadi seperti gaya dan daya yang diperlukan untuk menggerakan objek tersebut. Dalam kinematika pergerak kaki robot ini dibagi menjadi tiga bagian utama yaitu, invers kinematik, trajektori linear, dan transformasi geometri. Pada tiap kaki robot terdiri dari tiga derajat kebebasan.

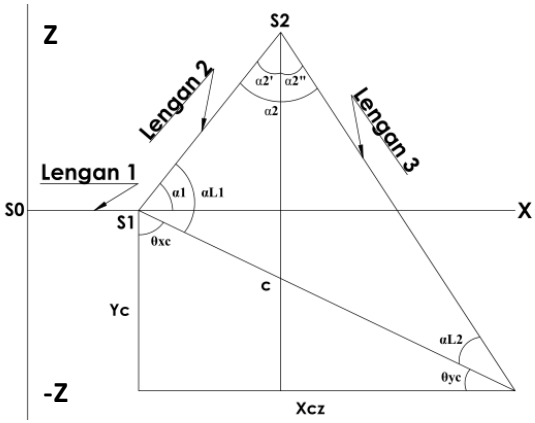
Gambar 2.11. Blok Diagram Susunan Algoritma Pergerakan Kaki

Pada blok diagram di atas adalah urutan proses dalam menghasilkan jenis gerak langkah yang diinginkan seperti maju dan belok. Proses diawali dengan input koordinat jenis gerak yang akan dilakukan, lalu diproses dalam algoritma transformasi geometri untuk mentransformasi posisi awal dari ujung kaki robot ke posisi yang diinginkan, setelah itu masuk ke fungsi trayektori untuk menghasilkan lintasan kaki yang diinginkan sesuai jarak antar koordinat awal dan tujuan, dan terakhir posisi pada tiap koordinat yang dihasilkan pada

ujung kaki di invers supaya menghasilkan besaran sudut servo pada tiap sambungan lengan.

2.7.1. Inverse Kinematik Pada Tiap Kaki Robot

Invers Kinematik adalah sebuah metode untuk mengubah data posisi koordinat pada *end-effector* kartesian menjadi data berupa sudut pada masing - masing *joint*. Dengan menggunakan metode inverse kinematics maka bisa didapatkan sudut-sudut tiap *joint* yang selanjutnya dikirmkan ke servo pada kaki robot. Pada gambar 2.12 di bawah ini adalah postur lengan pada tiap kaki robot.



Gambar 2.12. Sketsa Invers Kinematik Pada Tiap Kaki Robot

Dari gambar tersebut, didapatkan rumus inverse kinematik pada tiap kaki robot sebagai berikut :

$$X_C = L_{total\ X} - L_0$$

$$C = \sqrt{Xc^2 + Yc^2}$$

$$\alpha L2 = cos^{-1}\left(\frac{-L2^2 + L1^2 + C^2}{2.L_1.C}\right)$$

$$\alpha 0 = tan^{-1}\left(\frac{L_{total\ x}}{L_{total\ y}}\right) \quad (2.7)$$

$$\alpha 1 = \alpha L2\ + \theta c - 90° \quad (2.8)$$

$$\alpha 2 = cos^{-1}\left(\frac{L1^{2} + L2^{2} - C^{2}}{2.L1.L2}\right) \quad (2.9)$$

Dari rumus invers kinematik diatas, didapat alfa 0 adalah sudut servo untuk lengan satu, alfa 1 adalah sudut servo untuk lengan dua, dan alfa 3 adalah sudut servo untuk lengan ke tiga.

2.7.2. Trajektori Linear Pada Tiap Ujung Kaki

Trayektori ujung kaki merupakan sebuah perencanaan path (lintasan) gerak pada ujing kaki robot. Lintasan tersebut terbentuk dari kumpulan titik yang sudah direncanakan dan diperbarui pada tiap waktu tertentu yaitu waktu sampling, dengan waktu sampling yang digunakan disini adalah 0,04 detik. Perencanaan trayektori atau lintasan yang diterapkan pada ujung kaki robot ini adalah jenis kartesian trayektori berbasis kecepatan. Sehingga pembaruan titik posisi kaki ke titik berikutnya adalah berdasarkan kombinasi kecepatan dikali dengan waktu sampling.[5]

Rumus dari perencanaan kartesian trajektori berbasis kecepatan adalah sebagai berikut :

$$q(n.T) = V.T + q((n-1).T) \quad (2.10)$$

Dimana :

- q(n) : Posisi ke-n
- V : Kecepatan linear (cm/s)
- T : Waktu Sampling (s)

Pada tiap kaki robot ini terdapat tiga sumbu aksis, yaitu x, y, dan z sehingga dalam pembaruan posisi dari hasil kecepatan trayektori tersebut dilakukan penskalaan pada masing-masing sumbu, sehingga diapat kecepatan Vx, Vy, dan Vz yang merupakan kecepatan diagonal yang diharapkan posisi ujung tiap kaki akan bergerak dari posisi awal dan sampai posisi tujuan dengan waktu yang sama.

Rumus dari kecepatan diagonal pada masing-masing sumbu adalah sebagai berikut :

$$Vx = V \times \left(\Delta x/_S\right) \quad (2.11)$$

$$Vy = V \times \left(\Delta y/_S\right) \quad (2.12)$$

$$Vz = V \times \left(\Delta z/_S\right) \quad (2.13)$$

Dimana :

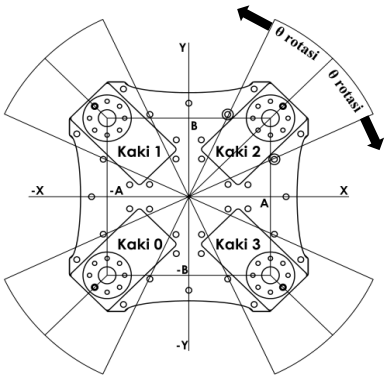
- $V_{(X/Y/Z)}$: Kecepatan pada sumbu X, Y, dan Z.
- $\Delta_{(X/Y/Z)}$: Perpindahan posisi pada sumbu X, Y, dan Z.
- S : Jarak dari perpindahan yang terjadi

2.7.3. Transofrmasi Geometri Pada Tiap Ujung Kaki

Transformasi Geometri adalah sebuah proses penentuan titik koordinat baru dari sebuah bangun pada sebuah bidang. Transformasi yang digunakan pada robot ini terdiri dari dua jenis yaitu transformasi rotasi dan transformasi translasi.

2.7.3.1. Transformasi Rotasi

Pada proses transformasi rotasi ini akan menentukan sudut putar badan robot yang menghasilkan koordinat putar pada masing masing ujung kaki robot. Masukan rotasi disimbolkan dengan ω (sudut putar), Nilai tersebut selanjutnya dimasukan ke dalam algoritma transformasi, yang selanjutnya untuk ω positif robot akan berputar ke kiri, dan ω negatif robot akan berputar ke kanan Pada gambar 3.13 adalah ilustrasi transformasi rotasi dari kaki robot. [5]



Gambar 2.13. Ilustrasi Transformasi Rotasi Pada Robot

Berikut ini adalah rumus transformasi rotasi pada robot adalah sebagai berikut :

$$x' = (cos\theta\,(x - a) - sin\theta\,(y - b)) + a \quad (2.14)$$

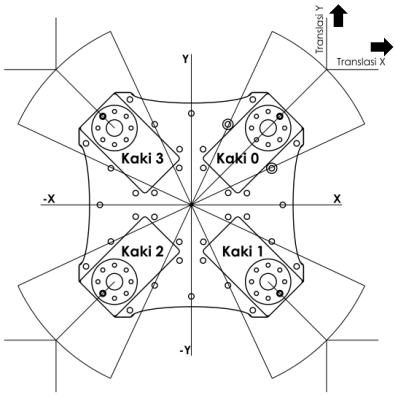
$$y' = (sin\theta\,(x - a) - cos\theta\,(y - b)) + b \quad (2.15)$$

Dimana :

- x : koordinat x awal
- y : koordinat y awal
- x' : koordinat x akhir
- y' : koordinat y akhir
- θ : sudut putar
- a : sumbu x pusat putar
- b : sumbu y pusat putar

2.7.3.2. Transformasi Translasi

Pada proses transformasi translasi ini akan menentukan lebar langkah robot yang menghasilkan koordinat translasi pada masing masing ujung kaki robot. Masukan translasi disimbolkan dengan x dan y (lebar langkah) dan z (tinggi langkah), Nilai tersebut selanjutnya dimasukan ke dalam algoritma transfotmasi, yang selanjutnya untuk x positif robot bergerak maju, x negatif robot bergerak mundur, y positif robot bergerak ke kanan, dan y negatif robot bergerak ke kiri, Pada gambar 3.14 adalah ilustrasi transformasi rotasi dari kaki robot. [5]



Gambar 2.14. Ilustrasi Transformasi Rotasi Pada Robot

Berikut ini adalah rumus transformasi translasi pada robot adalah sebagai berikut :

$$x' = x + a \quad (2.16)$$

$$y' = y + b \quad (2.17)$$

Dimana :

- x : koordinat x awal
- y : koordinat y awal
- x' : koordinat x akhir
- y' : koordinat y akhir
- a : pergeseran sumbu x (+ ke kanan, - ke kiri)
- b : pergeseran sumbu y (+ ke atas, - ke bawah)

2.7.4. Algoritma Fungsi Gerak Kaki Robot

Pada algoritma ini merupakan fungsi dimana akan menjalankan ketiga rumus kinematika di atas, dengan masukan pada fungsi gerak dinamis ini adalah berupa ω (sudut putar), x dan y (lebar langkah), z (tinggi langkah), dan speed sebagai kecepatan trayektori. Berikut dibawah ini adalah bentuk fungsi tersebut dalam bahasa C. [5]

```
void gerak(double x,double y,double w, double z, double
speed){
static int step = 0;
setSpeed(speed);
transformasiGerak(x,y,w);

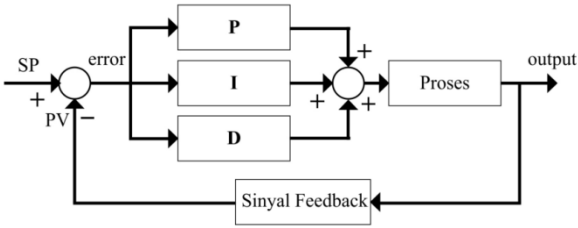
if(!statusGerak[0] && !statusGerak[1] && !statusGerak[2] &&
!statusGerak[3]){
 switch(step)
 {
  case 0:
  setPergeseranKaki(0,walkX1[0],walkY1[0],z);
  setPergeseranKaki(2,walkX1[2],walkY1[2],z);
  setPergeseranKaki(1,walkX2[1],walkY2[1],0);
  setPergeseranKaki(3,walkX2[3],walkY2[3],0);
  break;
  case 1:
  setPergeseranKaki(0,walkX1[0],walkY1[0],0);
  setPergeseranKaki(2,walkX1[2],walkY1[2],0);
  setPergeseranKaki(1,walkX2[1],walkY2[1],0);
  setPergeseranKaki(3,walkX2[3],walkY2[3],0);
  break;
  case 2:
  setPergeseranKaki(0,walkX2[0],walkY2[0],0);
```

```
 setPergeseranKaki(2,walkX2[2],walkY2[2],0);
 setPergeseranKaki(1,walkX1[1],walkY1[1],z);
 setPergeseranKaki(3,walkX1[3],walkY1[3],z);
 break;
 case 3:
 setPergeseranKaki(0,walkX2[0],walkY2[0],0);
 setPergeseranKaki(2,walkX2[2],walkY2[2],0);
 setPergeseranKaki(1,walkX1[1],walkY1[1],0);
 setPergeseranKaki(3,walkX1[3],walkY1[3],0);
 break;
}
step++;
if(step>3){step=0;hitung_langkah++;hitung_zigzag++;}
}
```

Untuk pola gerakan kaki diatas tersebut menggunakan pola algoritma langkah (gait). Algoritma ini diperlukan untuk mengatur waktu kapan sebuah kaki berada pada fase support (*stance*) yaitu posisi dimana kaki diam untuk mengangga badan robot, dan fase transfer (*swing*) yaitu posisi dimana kaki robot bergerak dari satu titik ke titik lainnya tanpa menyangga badan robot. Pada robot quadruped disini jenis algoritma gait yang digunakan adalah jenis *dualpod*, yaitu dinama jumlah kaki penopang (*stance*) dua buah dan kaki bergerak (*swing*) dua buah.

2.8. Sistem Kontrol Proportional Integral Derivatif

PID (ProportionaL Integral Derivative controller) merupakan kontroler untuk menentukan presisi suatu sistem instrumentasi dengan karakteristik adanya umpan balik pada sistem tesebut. Sistem kontrol PID terdiri dari tiga buah cara pengaturan yaitu kontrol P (Proportional), I (Integral), dan D (Derivative) dengan masing-masing memiliki kelebihan dan kekurangan masing-masing terdapat dalam tabel 2.1. Dalam implementasinya masing-masing cara dapat bekerja sendiri maupun gabungan diantaranya. Dalam perancangan sistem kontrol PID yang perlu dilakukan adalah mengatur parameter P, I atau D agar tanggapan sinyal keluaran system terhadap masukan tertentu sebagaimana yang diinginkan. [4]



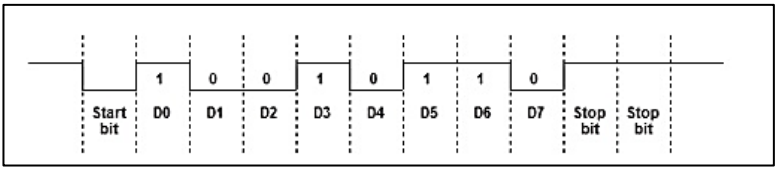
Gambar 2.15. Blok diagram sistem kontrol PID

Tabel 2.1. Efek pada masing-masing konroller Kp, Ki, dan Kd terhadap kinerja sistem kontrol PID

Parameter	Rise Time	Overshoot	Settling Time	Steady-State Error	Stability
Kp	Menurun	Meningkat	Sedikit Berubah	Menurun	Menurun
Ki	Menurun	Meningkat	Meningkat	Tereleminasi	Menurun
Kd	Sedikit Berubah	Menurun	Menurun	Tidak Berpengaruh	Meningkat jika Kd kecil

2.9. Komunikasi Serial

Komunikasi serial adalah komunikasi yang pengiriman datanya per-bit secara berurutan dan bergantian. Komunikasi ini mempunyai suatu kelebihan yaitu hanya membutuhkan satu jalur dan kabel yang sedikit dibandingkan dengan komunikasi paralel. Pada prinsipnya komunikasi serial merupakan komunikasi dimana pengiriman data dilakukan per bit sehingga lebih lambat dibandingkan komunikasi parallel, atau dengan kata lain 16 komunikasi serial merupakan salah satu metode komunikasi data di mana hanya satu bit data yang dikirimkan melalui seuntai kabel pada suatu waktu tertentu. [4][2]



Gambar 2.16. Format Frame Pada Komunikasi Serial

Dengan :

St	Start bit, selalu nol
D0 ~ D7	Data bits (0 sampai 8)
P	Parity bit. Can be odd or even
Sp	Stop bit, selalu satu

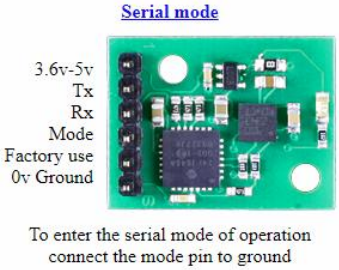
Terdapat dua macam komunikasi serial yaitu asynchronous serial dan synchronous serial.

1. Komunikasi Serial Sinkron
   Synchronous serial adalah komunikasi dimana hanya ada satu pihak (pengirim atau penerima) yang menghasilkan clock dan mengirimkan clock tersebut bersama-sama dengan data. Contoh pengunaan synchronous serial terdapat pada transmisi data keyboard.

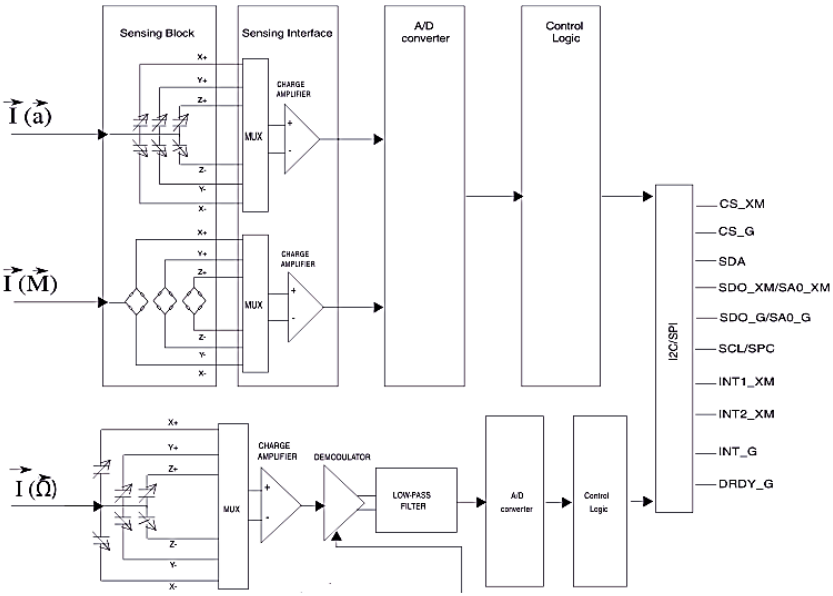
2. Komunikasi Serial Ansinkron
   Asynchronous serial adalah komunikasi dimana kedua pihak (pengirim dan penerima) masing-masing menghasilkan clock namun hanya data yang ditransmisikan, tanpa clock. Agar data yang dikirim sama dengan data yang diterima, maka kedua frekuensi clock harus sama dan harus terdapat sinkronisasi. Setelah adanya sinkronisasi, pengirim akan mengirimkan datanya sesuai dengan frekuensi clock pengirim dan penerima akan membaca data sesuai dengan frekuensi clock penerima. Contoh penggunaan asynchronous serial adalah pada Universal Asynchronous Receiver Transmitter (UART) yang digunakan pada serial port (COM) komputer.

2.10. Modul Sensor CMPS11

Modul sensor CMPS11 adalah modul sensor kompas generasi ke 3 yang dikembangkan oleh perusahaan Devantech, sedangkan untuk IC pada modul ini adalah LSM9DS0 dari perusahaan STMicroelectronic.



Gambar 2.17. Modul CMPS11 dan Pin Outnya dalam Mode Komunikasi Serial [13]



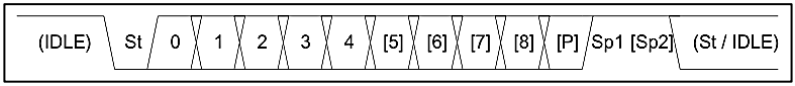
Gambar 2.18. Blok Diagram Modul Sensor IMU CMPS11 [12]

Modul sensor CMPS11 ini merupakan modul sensor IMU yang menggabungan tiga sensor IMU, yaitu tiga sumbu sensor akselerometer, tiga sumbu sensor giroskop, dan tiga sumbu sensor magnetometer. Untuk mengurangi eror yang disebabkan oleh kemiringan PCB maka digunakan sebuah Kalman Filter dalam mengkombinasikan sensor gyro dan sensor percepatan, fitur ini cocok diterapkan pada kacamata tuna netra, karena pergerakan kepala pengguna akan mengakibatkan kemiringan kompas menjadi tidak selalu stabil. CMPS11 menghasilkan output berupa nilai dari 0-3599 desimal untuk data 16 bit atau 0-255 desimal untuk data 8 bit yang mewakili 0°-359,9°, sehingga mikrokontroller dengan lebar data 8 bit dapat digunakan dalam melakukan perhitungan.

Tabel 2.2. Spesifikasi Teknik Modul CMPS11

Tegangan Supply	3.6V ~ 5V
Protokol komunikasi	I2C dan Serial UART (no pairty, dan 2 stop bits) dengan default baudrate 9600 bps
Chip sensor IMU	LSM9DS0 dari STMicroelectronic
Chip pengolah raw data	24FJ64GA002 dari Microchip

Pengaksesan data sensor IMU pada CMPS11 dengan komunikasi serial memiliki strruktur sebagai berikut :



Dimana maksud dari masing – masing bit data diatas adalah sebagai berikut :

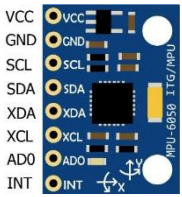
- St adalah Start bit, selalu bernilai nol
- 0 ~ 8 adalah bit perintah, terdapat pada tabel 2.2 dalam bentuk kode heksa. Pada tabel tersebut terdapat perintah dan juga respon dari perintah tersebut. Untuk mengakses data 16 bit, maka data akan dibagi menjadi dua yaitu MSB dan LSB yang masing masing 8 bit kemudian digabung menjadi satu.
- P adalah Pairty bit. Pada mode serial ini, bit ini dikosongkan
- Sp adalah Stop bit dengan nilai selalu tinggi dan terdiri dari dua bit

Tabel 2.3. Daftar Perintah dan data yang diperoleh pada CMPS11

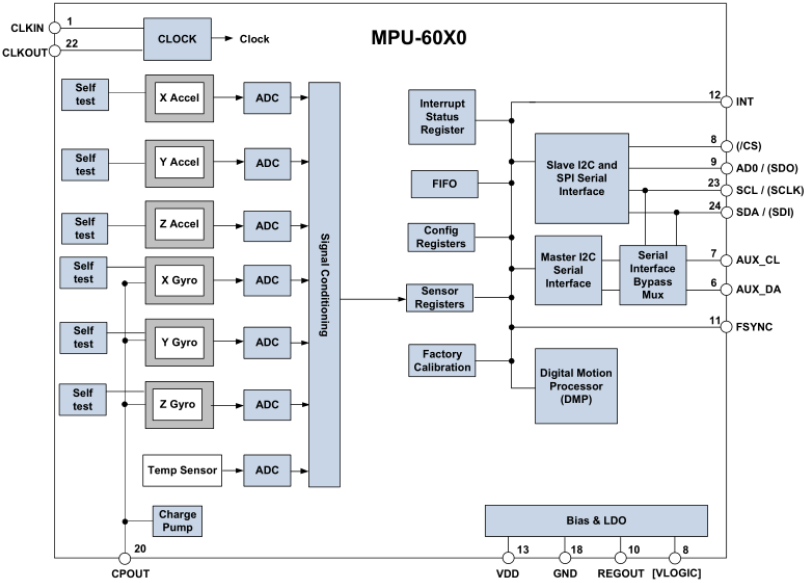
Command	Name	Bytes returned	Returned data description
0x11	GET VERSION	1	Software version
0x12	GET BEARING 8 BIT	1	Bearing as a single byte 0-255
0x13	GET BEARING 16 BIT	2	Bearing (16 bit), high byte first 0-3599
0x14	GET PITCH	1	Pitch angle +/- 0-90°
0x15	GET ROLL	1	Roll angle +/- 0-90°
0x19	GET MAG RAW	6	Raw magnetic data, 16 bit signed: X high, X low, Y high, Y low, Z high, Z low
0x20	GET ACCEL RAW	6	Raw accelerometer data, 16 bit signed: X high, X low, Y high, Y low, Z high, Z low
0x21	GET GYRO RAW	6	Raw gyro data, 16 bit signed: X high, X low, Y high, Y low, Z high, Z low
0x22	GET TEMP	2	BNO055 reported temperature as two bytes, high byte first and scaled in °C
0x23	GET ALL	4	Angle high, angle low (0-3599), pitch (+/- 0-90), roll (+/- 0-90)
0x24	GET CALIBRATION STATE	1	Bits 0 and 1 reflect the calibration status (0 un-calibrated, 3 fully calibrated)
0x25	GET BOSCH BEARING 16 BIT	2	Bearing (16 bit), high byte first (0-5759), divide by 16 for degrees
0x26	GET PITCH 180	2	Pitch angle (16 bit) high bytes first +/- 0-180°
0xF0	STORE CALIBRATION BYTE 1	1	Returns ok (0x55)
0xF5	STORE CALIBRATION BYTE 2	1	Returns ok (0x55)
0xF6	STORE CALIBRATION BYTE 3	1	Returns ok (0x55)
0xE0	DELETE CALIBRATION BYTE 1	1	Returns ok (0x55)
0xE5	DELETE CALIBRATION BYTE 2	1	Returns ok (0x55)
0xE2	DELETE CALIBRATION BYTE 3	1	Returns ok (0x55)
0xA0	BAUD 19200	1	Returns ok (0x55)
0xA1	BAUD 38400	1	Returns ok (0x55)

2.11. Modul Sensor ITG/GY-521

Modul sensor GY-521 ini adalah modul sensor IMU yang menggunakan unit mikroprosesor MPU6050 yang didalamnya terdiri dari tiga sensor IMU akselerometer dan tiga sensor IMU giroskop. Sensor ini adalah salah satu sensor cerdas karena dapat menyediakan data gerakan yang telah diolah melalui sebuah pemrosesan internal yang lebih dikenal dengan DMP(Digital Motion Processor). Protokol yang digunakan dalam modul sensor ini adalah dengan menggunakan komunikasi I2C.



Gambar 2.19. A Modul ITG/GY-521 beserta Pin Outnya



Gambar 2.20. Blok Diagram Modul Sensor IMU ITG/GY-521 [11]

Tabel 2.4. Spesifikasi Teknik ITG/GY-521

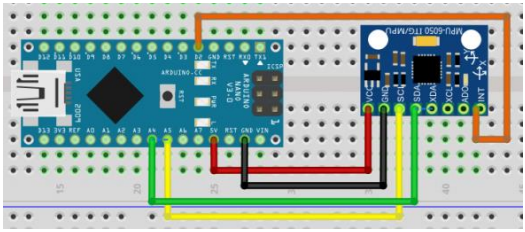
Tegangan Supply	2.375V ~ 3.46V
Protokol komunikasi	I2C
Chip sensor IMU	MPU6050

Sensor IMU giroskop pada modul MPU-6050 ini terdiri dari tiga giroskop MEMS vibrasi, yang mendeteksi rotasi pada sumbu x, y, dan z. Ketika gyroskop diputar pada salah satu sumbu, Efek Coriolis menyebabkan getaran yang dideteksi oleh pickoff kapasitif yang menghasilkan sinyal tegangan. Sinyal yang ini dihasilkan diperkuat, didemodulasi, dan difilter untuk menghasilkan tegangan yang sebanding dengan tingkat sudut. Tegangan ini didigitalkan menggunakan on-chip 16-bit Analog-to-Digital Converters (ADC) untuk sampel setiap sumbu. Rentang skala penuh sensor gyro dapat diprogram secara digital hingga ± 250, ± 500, ± 1000, atau ± 2000 derajat per detik (dps).

Untuk sensor IMU akselerometer pada modul MPU-6050 ini terdiri dari tiga aksis proff massa (massa uji), yaitu padda sumbu x, y, dan z. Akselerasi sepanjang sumbu tertentu menginduksi perpindahan pada proff mass yang sesuai, dan sensor kapasitif mendeteksi perpindahan secara diferensial. Ketika perangkat ditempatkan pada permukaan yang datar, ia akan mengukur 0g pada sumbu x dan y dan + 1g pada sumbu-z. Setiap sensor memiliki delta sigma khusus untuk menyediakan output digital. Rentang skala penuh output digital dapat disesuaikan hingga ± 2g, ± 4g, ± 8g, atau ± 16g, g disini adalah percepatan gravitasi yaitu 9,81 m/s^2.

2.11.1. Wiring Modul Sensor MPU6050 Pada Arduino

Pada pengkabelan modul sensor MPU6050 dan Arduino ini dilakukan dengan menggunakan komunikasi I2C, yaitu dengan menghububgkan pin SDA dan SCL pada modul dengan pin A4 (SDA) dan A5 (SCL) pada Arduino, dan pin INT pada modul dengan pin 2 pada Arduino. Untuk tegangan supply, modul ini bekerja pada tegangan 2.375V ~ 3.46V.



Gambar 2.21. Modul ITG/GY-521 beserta Pin Outnya lustrasi Wiring MPU6050 dengan Arduino Nano

2.11.2. Program Akses Modul MPU6050 Pada Arduino IDE

Pada tugas akhir ini, proses pengolahan raw data menjadi output yaw data menggunakan bantuan library dari Jeff Rowberg pada software Arduino IDE. Untuk memula mengakses data, pertama aktifkan dahulu fungsi inisialisasi `OUTPUT_READABLE_YAWPITCHROLL` (gambar 2.17) yang terdapat di luar fungsi void, lalu fungsi untuk menampilkan data yaw, pitch, dan roll terdapat pada gambar 2.18.

```
95  // uncomment "OUTPUT_READABLE_YAWPITCHROLL" if you want to see the yaw/
96  // pitch/roll angles (in degrees) calculated from the quaternions coming
97  // from the FIFO. Note this also requires gravity vector calculations.
98  // Also note that yaw/pitch/roll angles suffer from gimbal lock (for
99  // more info, see: http://en.wikipedia.org/wiki/Gimbal_lock)
100 #define OUTPUT_READABLE_YAWPITCHROLL
```

Gambar 2.22. Fungsi Inisial Akses Data Yaw, Pitch, dan Roll

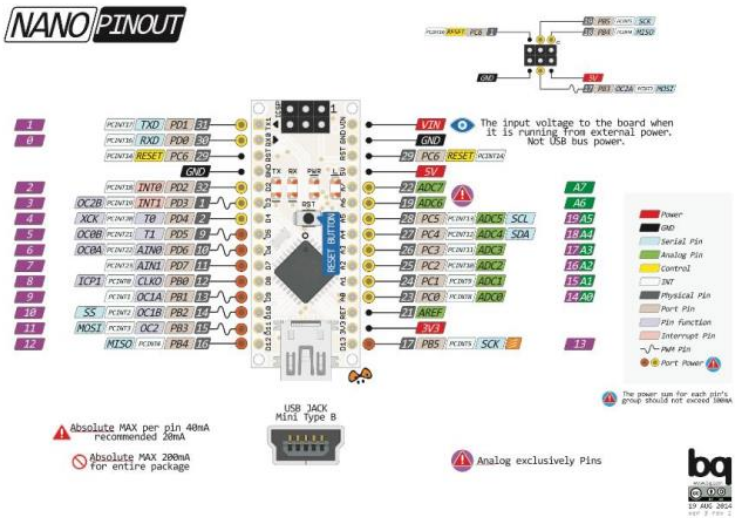
```
313         #ifdef OUTPUT_READABLE_YAWPITCHROLL
314             // display Euler angles in degrees
315             mpu.dmpGetQuaternion(&q, fifoBuffer);
316             mpu.dmpGetGravity(&gravity, &q);
317             mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
318             Serial.print("ypr\t");
319             Serial.print(ypr[0] * 180/M_PI);
320             Serial.print("\t");
321             Serial.print(ypr[1] * 180/M_PI);
322             Serial.print("\t");
323             Serial.println(ypr[2] * 180/M_PI);
324         #endif
```

Gambar 2.23. Fungsi Tampil Data Yaw, Pitch, Roll

2.12. Arduino Nano

Arduino merupakan sebuah mikrokontroler single-board yang bersifat open-source, diturunkan dari Wiring platform, dirancang sedemikian hingga agar mudah digunakan dalam berbagai bidang.

Arduino memiliki prosesor Atmel AVR dan memiliki bahasa pemrograman yang mudah diterjemahkan, bahasa pemrograman Arduino yang memiliki kemiripan syntax dengan bahasa pemrograman C++. Arduino menggunakan keluarga mikrokontroler ATMega yang dirilis oleh Atmel sebagai basis. Pada kondisi lain terdapat clone arduino dengan menggunakan mikrokontroler berbeda dan tetap kompatibel dengan arduino pada level hardware.



Gambar 2.24. Arduino Nano Beserta Pin Outnya

Tabel 2.5. Spesifikasi Teknik Arduino Nano

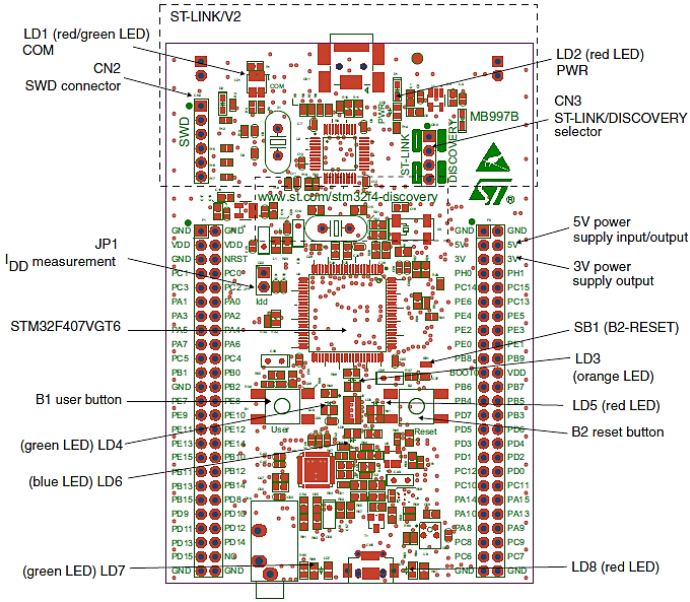
Mikrokontroller	ATmega 328P U16
Tegangan Operasi	5V
Input Voltage (disarankan)	7V ~ 12V
Komunikasi	I2C, Serial, SPI
Pin Digital I/O	13 (digital PWM)
Pins Input Analog	8
Arus DC per pin I/O	40 mA
Arus DC untuk pin 3.3V	50 mA
Flash Memory	
SRAM	
EEPROM	
Clock Speed	16000 KHz

2.13. STM32F4 Discovery

STM32F4 Discovery adalah sebuah modul mikrokontroler yang di dalamnya menggunakan IC STM32F407VGT6 ARM Cortex M4 yang mempunyai kecepatan sampai dengan 168 MHz, serta mampu mengeksekusi perintah hingga 210 MIPS (Million Instruction per Second). STM32F4 Discovery merupakan mikrokontroler 32 bit dengan arsitektur ARM. Mikrokontroller ini memiliki kapasitas 1 MByte Flash PEROM (Flash Programmable and Eraseble Read Only Memory), 192 Kbyte SRAM. Dilengkapi dengan 100 buah pin input output yang mempunyai karakteristik masing-masing yaitu USART, TIMER, ADC dan I2C.

Fitur dari STM32F4 Discovery antara lain yaitu :

- On-board ST-LINK/V2 yang digunakan untuk proses uploadi file hex ke dalam mikrokontroler.
- ST MEMS motion sensor, 3-axis accelerometer.
- Audio DAC dengan driver speaker kelas D
- Dua push button (user dan reset)
- USB OTG



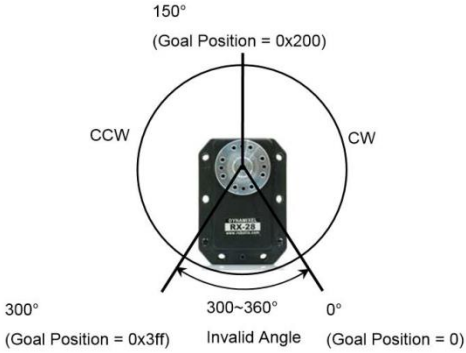
Gambar 2.25 STM32F4 Discovery [14]

2.14. Servo Dynamixel RX-28

Aktuator servo Dynamixel RX-28 merupakan sebuah servo yang mengintegrasikan speed reducer, controller, driver, dan network function menjadi satu modul. Setiap servo mempunyai ID yang dapat ditentukan dan diubah-ubah. Servo ini dikendalikan dengan paket komunikasi pada sebuah bus dengan menggunakan RS485 yang kemudian disambungkan pada mikrokontroller.

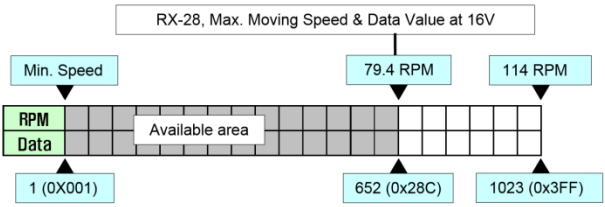
Gambar 2.26 Servo RX-28 [15]

Pada servo ini mikrokontroller dapat mengontrol kecepatan, posisi, torsi, dll dengan satu paket data, serta dapat mengontrol banyak servo dengan satu paket data. Terdapat penanda dalam bentuk status LED saat temperatur, torsi, tegangan supply menyimpang dari yang telah di tetapkan oleh user, serta terdapat fasilitas shutdown dimana servo akan otomatis mati sendiri pada keadaan tertentu sesuai yang user tetapkan.



Gambar 2.27 Rentan Goal Position Servo RX-28 [15]

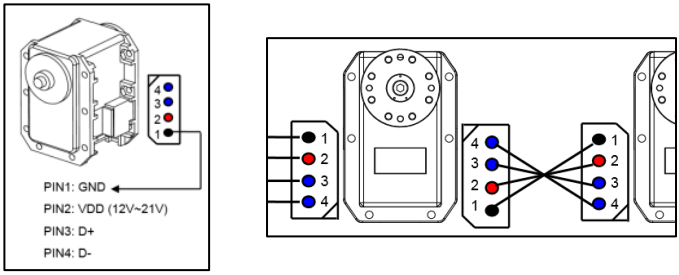
Rentang goal position pada servo dynamixel RX-28 adalah 0 ~ 300°, sehingga servo tidak dapat digerakkan pada sudut servo 300 ~ 360° atau pada sudut ini disebut sebagai invalid angle. Untuk menentukan goal position digunakan skala 0 sampai 1023 (10 bit) dengan 0 (0x00) = 0° dan 1023 (0x3FF) = 360°.



Gambar 2.28 Rentan & Hubungan Nilai Data & Kecepatan (RPM) [15]

Rentang kecepatan pada servo RX-28 adalah 0 sampai 1023. Kecepatan pada servo RX-28 dapat dikonversikan menjadi besaran RPM dengan mengalikan data kecepatan dengan 0.111. Contohnya jika data kecepatan yang dikirimkan adalah 1023 maka kecepatan dalam

RPM adalah 1023 x 0.111 = 113.6. Tetapi kecepatan riil pada servo ditentukan oleh banyak faktor luar. Kecepatan maksimal dari servo RX-28 berbanding lurus dengan ukuran tegangan suplai. Semakin besar tegangan suplai maka semakin besar area kecepatan yang dapat dikendalikan. Contohnya ketika RX-28 disuplai dengan 16V maka dapat mengendalikan kecepatan 0 sampai 79.4 RPM, sedangkan apabila disuplai dengan tegangan 12V maka keceptan maksimal akan berkurang menjadi 59.9RPM.



Gambar 2.29 Susunan Pin RX-28 (kiri), dan susunan koneksi RX-28 (kanan) [15]

Pada servo RX-28 terdapat empat buah port yang terdiri dari GND, VDD, D+ dan D-. Menghubungkan beberapa RX-28 dapat dilakukan melalui BUS.

Untuk mengontrol pergerakan RX-28, data yang dikirimkan harus sesuai dengan protokol RX-28. RX-28 digerakaan dengan menerima data biner. Mikrokontroller dan RX-28 berkomunkasi satu sama lain (dua arah) dengan mengirimkan dan meneruma data yang disebut paket. paket. Paket terbagi menjadi dua yaitu :

- Instruction packet, dimana mikrokontroller mengirimkan data kontrol ke RX-28
- Status packet, dimana RX-28 merespon data ke Mikrokontroler.

Paket instruksi yang dikirimkan dari mikrokontroler ke RX-28 mempunyai struktur sebagai berikut :

OXFF	0XFF	ID	LENGTH	INSTRUCTION	PARAMETER1	...	PARAMETER N	CHECK SUM

Dimana *instruction* adalah jenis perintah yang diberikan dan *parameter* adalah data parameter yang dikirim dan ingin di akses. Berikut pada tabel 2.5 dan tabel 2.6 adalah daftar intruksi dan parameter servo rx-28.

Tabel 2.6. Daftar Tipe Instruksi Dynamixel RX-28

Value	Name	Function	No. of Parameters
0x01	PING	No execution. It is used when controller is ready to recevie Status Packet	0
0x02	READ DATA	This command reads data from RX-64	2
0x03	WRITE DATA	This command writes data to RX-64	2 or more
0x04	REG WRITE	It is similar to WRTE_DATA, but it remains in the standby state without being executed until the ACTION command arrives.	2 or more
0x05	ACTION	This command initiates motions registered with REG WRITE	0
0x06	RESET	This command restores the state of RX-64 to the factory default setting.	0
0x83	SYNC WRITE	This command is used to control several RX-64s simultaneously at a time.	4 or more

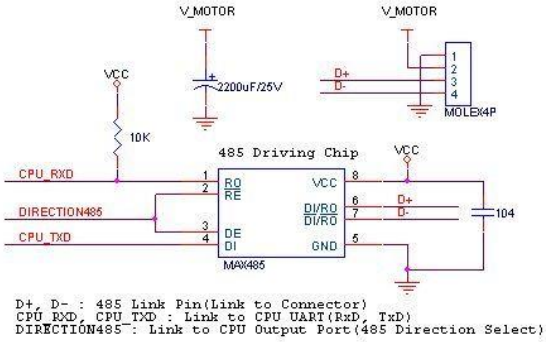
Tabel 2.7. Daftar Tipe Parameter Dynamixel RX-28

	Address (hexadecimal)	Name	Description	Access	Initial Value (Hexadecimal)
EEPROM Area	0 (0X00)	Model Number(L)	Lowest byte of model number	R	28 (0X1C)
	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	34 (0X22)
	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	80 (0X50)
	12 (0X0C)	the Lowest Limit Voltage	Lowest Limit Voltage	RW	60 (0X3C)
	13 [0X0D)	the Highest Limit Voltage	Highest Limit Voltage	RW	240 (0XF0)
	14 (0X0E)	Max Torque(L)	Lowest byte of Max. Torque	RW	255 (0XFF)
	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)
	18 (0X12)	Alarm Shutdown	Shutdown for Alarm	RW	36 (0X24)
RAM Area	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	0 (0X00)
	27 (0X1B)	CCW Compliance Margin	CCW Compliance margin	RW	0 (0X00)
	28 (0X1C)	CW Compliance Slope	CW Compliance slope	RW	32 (0X20)
	29 (0X1D)	CCW Compliance Slope	CCW Comliance 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	RW	-
	33 (0X21)	Moving Speed(H)	Highest byte of Moving Speed	RW	-
	34 (0X22)	Torque Limit(L)	Lowest byte of Torque Limit	RW	ADD14
	35 (0X23)	Torque Limit(H)	Highest byte of Torque Limit	RW	ADD15
	36 (0X24)	Present Position(L)	Lowest byte of Current Position	R	-
	37 (0X25)	Present Position(H)	Highest byte of Current Position	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 Instruction	Means if Instruction is registered	RW	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)

2.15. IC RS-485

Untuk memberikan perintah pada servo RX-28 dengan mikrokontroller, data pada USART (Universial Asynchronous Receiver/Transmitter) harus dikonversikan terlebih dahulu melalui RS-485 lalu menuju servo.



Gambar 2.30 Rangkaian RS-485 [15]

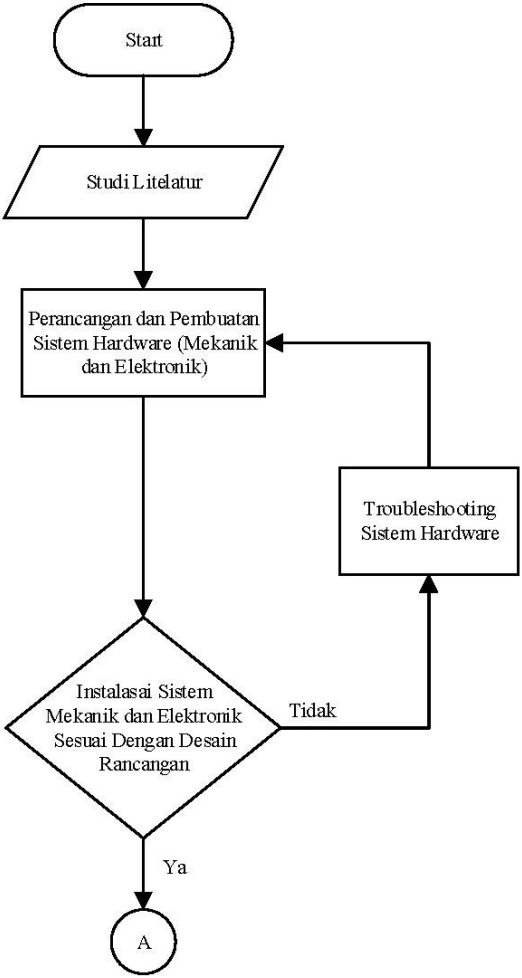
Berdasarkan datasheet, rangkaian RS486 untuk servo RX-28 pada STM32F4 Disc adalah sebagai berikut :

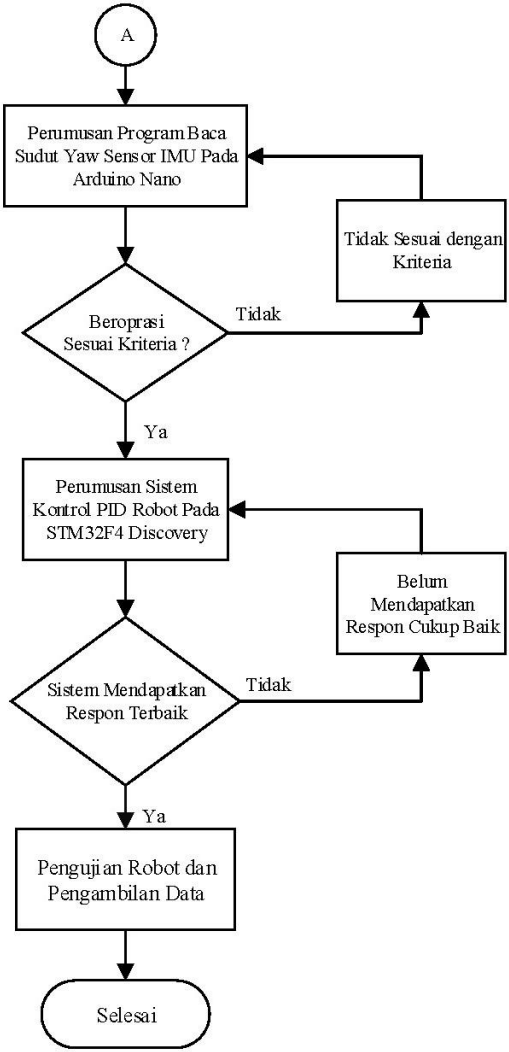
Supply dari RX-28 lewat Pin1(-) dan Pin2 (+). Port DIRECTION485 akan menentukan arah dari sinyal data pada TxD dan RxD. Jika DIRECTION485 bernilai 1 maka sinyal pada TxD akan keluar pada D+ dan D-, sedangkan apabila DIRECTION485 bernilai 0 maka sinyal D+ dan D- akan keluar pada RxD, sehingga port DIRECTION485 akan menentukan arah sinyal yaitu sebagai transmitter atau receiver. [5]

Halaman ini sengaja dikosongkan

BAB III
METODOLOGI

3.1 Diagram Alir (Flow Chart)





Gambar 3.1 Diagram Alir Perancangan Sistem

3.2 Studi Literatur

Mencari, mengumpulkan, dan mempelajari dasar dasar teori yang menunjang dalam penulisan tugas akhir ini. Dimulai dengan pengertian dan cara kerja sensor IMU, bagaimana mengolah data mentah (raw) sensor IMU hingga menjadi data posisi sudut, hingga bagaimana cara mereapkanya pada sistem kontrol PID dan terakhir menerapkan sistem kontrol tersebut pada robot.

3.3 Desain Sistem Hardware Robot

Pada proses desain desain robot ini terdiri dari desain mekanik robot yang terdiri dari desain frame kaki robot dan desain body robot dengan menggunakan bantuan software CAD SolidWorks dan selanjutnya adalah desain PCB Board elektronik robot yang terdiri dari desain PCB Shield mikrokontroller slave Arduino NANO dan mikrokontroller master STM32F4 Discovery dengan menggunakan bantuan software penggambar PCB yaitu EAGLE.

3.2.1 Desain Mekanik Robot

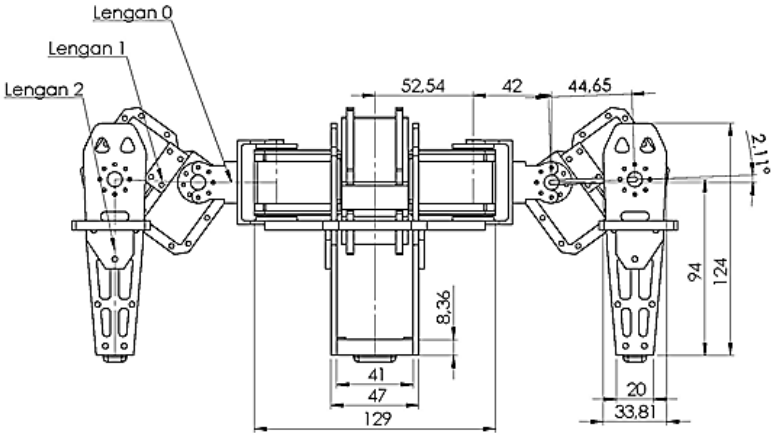
Proses perencanaan mekanik dibagi menjadi dua bagian utama yaitu perencanaan desain frame robot dan perencanaan desain body robot. Pada proses ini pendesainan dilakukan dengan menggunakan software CAD SolidWorks. Penggunaan software tersebut didasari dengan berba-

Gambar 3.2 Desain 3D Keseluruhan Robot

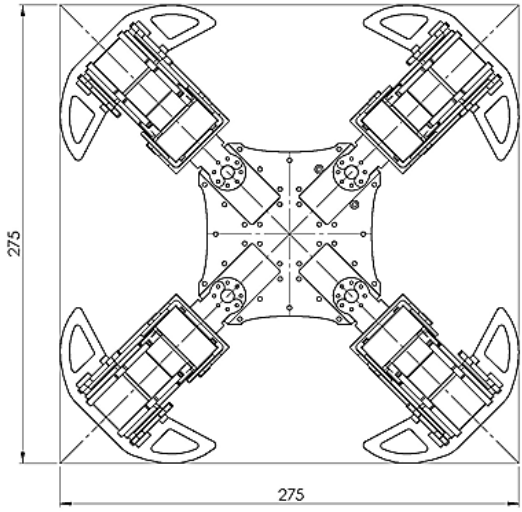
gai tujuan, antara lain agar didapatkan model 3D segingga dapat meminimalisir terjadinya kesalahan saat assembly terutama dengan board elektronik, agar didapatkan simulasi gerakan kaki sehingga mampu meminimalisir terjadinya tabrakan kaki dengan bagian lain, dan agar dapat dianalisa lebih lanjut sebelum dilakukan penyetakan.

3.2.1.1 Desain Frame Robot

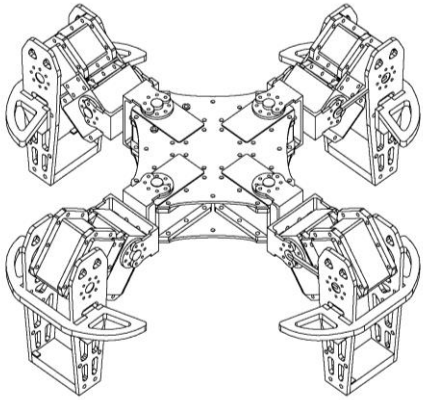
Desain dan pembuatan bagian frame terdiri dari lengan kaki dan base kaki. Dua faktor utama dalam mendesain frame dari robot adalah dimensi servo RX-28 dan dimensi maksimum mengacu pada aturan Kontes Robot Pemadam Api Indonesia (KRPAI) yaitu dengan panjang x lebar adalah 31x31 cm. Untuk begian frame ini terbuat dari aluminium agar frame ini kuat terhadap beban dan getaran yang berlangsung secara kejutan maupun kontinyue saat melakukan gerakan jalan, jenis bahan yang digunakan adalah aluminium alloy tipe 5052 dengan ketebalan dua mm. Berikut ini adalah gambar tampak samping dan atas dari bagian frame robot.



Gambar 3.3 Frame robot tampak samping



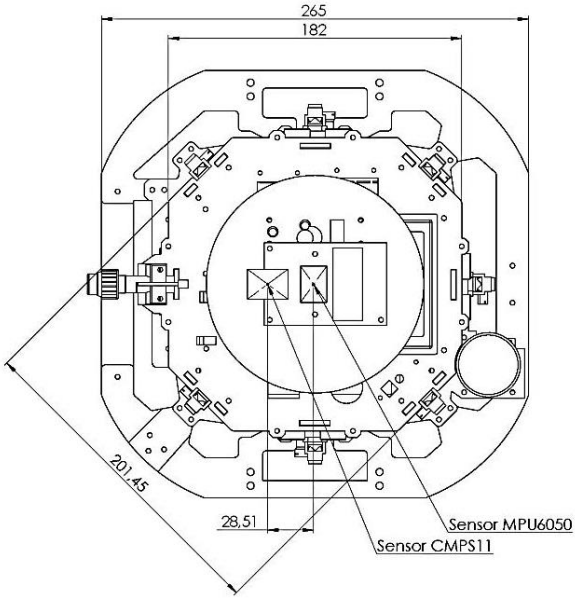
Gambar 3.4 Frame robot tampak atas



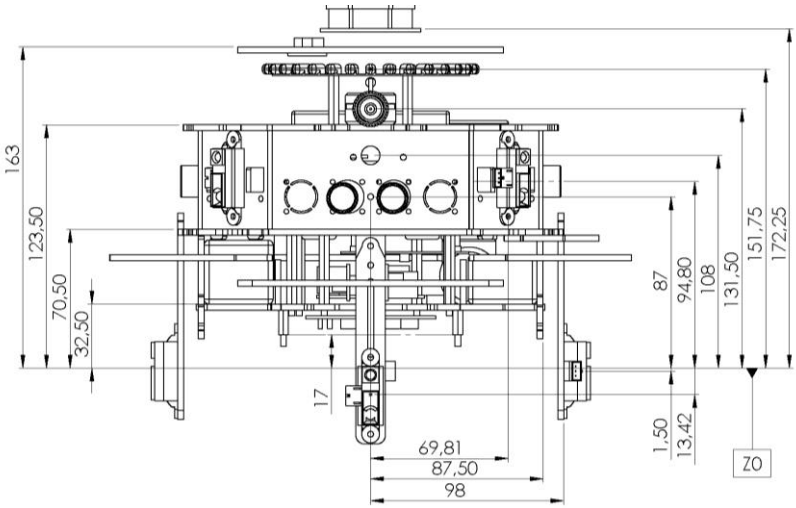
Gambar 3.5 Frame robot tampak isometri

3.2.1.2 Desain Body Robot

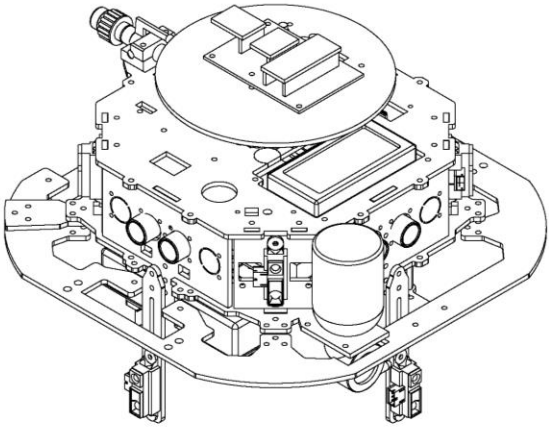
Bagian body robot merupakan tempat dari komponen elektronik seperti mikrokontroller, sensor, dan juga baterai. Faktor utama dalam mendesain body robot adalah mengkonsep peletakan posisi dari sensor terhadap titik pusat geometry body robot, dan juga peletakan posisi shield PCB mikrokontroller. Bagian body ini seluruhnya terbuat dari akrilik dengan tujuan agar beban robot menjadi ringan dan mudah direvisi jika ada kesalahan. Berikut ini gambar tampak samping dan atas body robot.



Gambar 3.6 Desain body robot tampak atas



Gambar 3.7 Desain Body Robot Tampak Depan

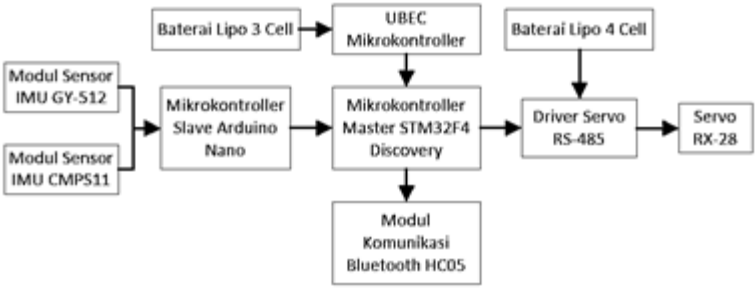


Gambar 3.8 Desain Body Robot Tampak Atas

3.2.2 Desain Sistem Elektronik Robot

Pada sub sub bab 3.2.2 ini dijelaskan mengenai rangkaian dari sistem elektronik robot yang kemudian dilakukan proses pendesainan shield PCB untuk mikrokontroller berdasarkan kebutuhan dari sistem komponen elektronik tersebut.

Proses pendesainan shield PCB untuk mikrokontroller ini dilakukan dengan menggunakan software eagle. Yang selanjutnya hasil desain tersebut di cetak dalam PCB dengan menggunakan proses. Tujuan pembuatan PCB disini adalah untuk mengekspansi pin yang ada pada mikrokontroller agar lebih mudah dihubungkan dengan modul sensor, aktuator servo, dan juga slave mikrokontroller. Dibawah ini adalah mengenai rangkaian dari sistem elektronik robot.



Gambar 3.9 Blok diagram dari Sistem Elektronik Robot

Rangkaian dari Sistem elektronik robot ini terdiri dari komponen modul sensor, modul mikrokontroller, dan aktuator servo. Pada Tugas Akhir ini modul sensor terdiri dari dua sensor IMU, yaitu sensor CMPS11 dan GY-521. Pada sensor CMPS11 terdapat sensor magnetometer yang berfungsi untuk mengakses posisi derajat kompas untuk menentukan setpoint posisi derajat yaw robot terhadap lingkungan. Sedangkan sensor GY-521 berfungsi sebaga sensor feedback kontrol PID untuk mendeteksi error pada robot, dengan menggunakan kombinasi sensor akselerometer dan giroskop pada sensor tersebut.

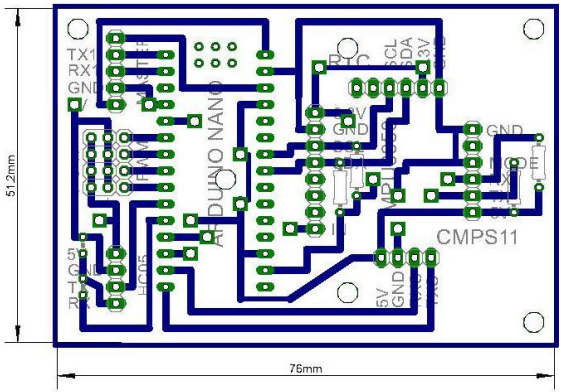
Selanjutnya, pada Tugas Akhir ini, modul mikrokontroller yang digunakan adalah mikrokontroller Arduino NANO dan STM32F4 Discovery. Mikrokontroller Arduino dengan basis ATmega328P berfungsi sebagai pengakses kedua sensor IMU, juga sebagai

mikrokontrller slave. Sedangkan mikrokontroller STM32F4 Discovery merupakan mikrokontroller master yang berfungsi sebagai pusat pemrosesan data, mulai dari mengolah data yang dikirim oleh slave, memprosesnya kedalam kontrol PID, hingga mengirinya ke konverter RS-485 yang selanjutnya dikonversi dan dikirim ke servo Dynamixel RX-28.

Selanjutnya, pada Tugas Akhir ini, aktuator yang digunakan adalah servo Dynamixel RX-28. Servo RX-28 ini berfungsi untuk menghasilkan gerakan mekanik berupa gerak langkah robot dari sinya input yang dikirimkan. Karena sistem komunikasi yang terdapat pada servo berbeda dengan yang terdapat pada mikrokontroller maka data dikonversi dahulu dengan menggukan IC RS-485.

3.2.2.1 PCB Shield Arduino Nano

PCB Shield slave Arduino Nano disini digunakan untuk memuat mikrokontroller Arduino Nano, yang berfungsi untuk meng-ekspansi pin pin yang ada pada mikrokontroller tersebut sehingga mudah dihubungkan dengan modul sensor IMU. Komponen penyusun pada shield ini adalah resistor dan lampu LED sebagai indikator.

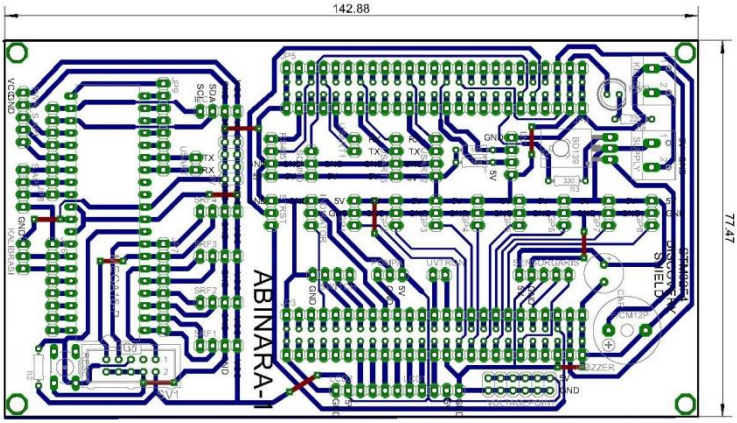


Gambar 3.10 Arduino Nano Slave PCB Shield

Alasan digunakannya mikrokontroller tambahan atau slave Arduino Nano ini adalah sebagai asisten dari mikrokontroller master, dikarenakan data yang dikeluarkan oleh sensor MPU6050 tersebut masih berupa *raw data*, sehingga memerlukan waktu untuk memproses data hingga data tersebut menjadi sudut derajat *yaw, pitch*, dan *roll*.

3.2.2.2 PCB Shield STM32F4 Discovery

PCB Shield Master STM32F4 Discovery disini digunakan untuk memuat mikrokontroller STM32F4 Discovery, yang digunakan untuk meng-ekspansi pin pin yang ada pada mikrokontroller tersebut sehingga mudah dihubungkan dengan modul Sensor IMU, PCB Shield Slave, PCB Converter RS-485 dan juga sumber tegangan.



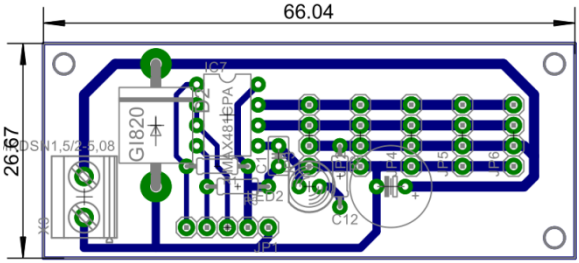
Gambar 3.11 STM32F4 Discovery Master PCB Shield

Komponen penyusun pada PCB shield ini terdiri dari :

- LED (3mm)
- Kapasitor (10Uf)
- Dioda (1N4001)
- Resistor (10K, 560)
- Transistor BD139
- Buzzer
- Pin Header dan Pin Screw
- Push Button

3.2.2.3 PCB Konverter RS-485

PCB Konverter RS-485 disini digunakan untuk memuat IC RS-485 yang merupakan sebuah protocol komunikasi yang digunakan oleh servo Dynamixel RX-28 untuk berkomunikasi dua arah dengan mikrokontroler master STM32F4 Discovery. RS-485 merupakan protokol komunikasi bekerja secara half duplex.



Gambar 3.12 PCB Board RS-485

Komponen penyusun pada PCB Board RS-485 ini terdiri dari :

- IC Max 485
- Dioda (1N4007)
- Resistor (10K, 560)
- Kapasitor (100uf)
- Pin Header dan Pin Screw

3.4 Perancangan Sistem Sensor IMU

Pada perancangan sistem sensor *inersia measuremen unit* (IMU) ini dibagi menjadi dua bagian utama, yaitu sistem sensor IMU sebagai kompas dengan sensor CMPS11 dan sistem sensor IMU sebagai *feedback* posisi derajat kemiringan yaw robot dengan sensor MPU6050.

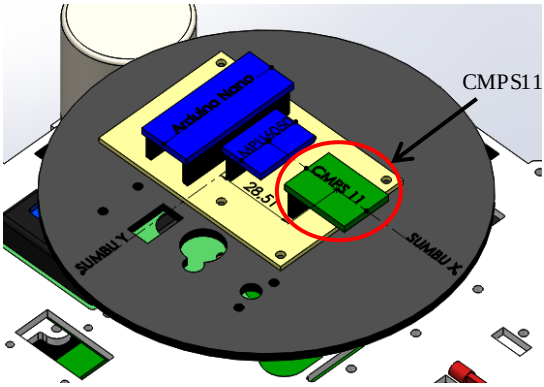
3.4.1 Perancangan Sistem Modul Sensor CMPS11 Pada Robot

Modul sensor CMPS 11 ini menggunakan chip LSM9DS0 dari STMicroelectronics yang terdiri dari tiga jenis sensor IMU yaitu MEMS giroskop, MEMS akselerometer, dan MEMS magnetometer yang masing masing memiliki tiga sumbu axis x,y, dan z.

Pada Tugas Akhir ini modul sensor CMPS11 digunakan untuk mengakses arah mata angin atau arah kompas dengan menggunakan kombinasi sensor MEMS magnetometer dan MEMS giroskope pada sensor untuk menentukan *setpoint* arah posisi derajat yaw body robot terhadap arah mata angin yang digunakan untuk proses navigasi robot.

3.4.1.1 Penempatan Modul CMPS11 Pada Robot

Desain rancangan penempatan *modul sensor* CMPS11 pada body robot ada pada gambar 3.13 dibawah ini.

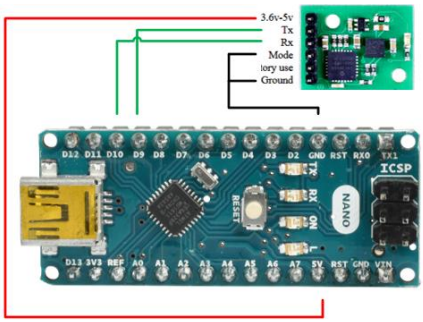


Gambar 3.13 Desain Peletakan Sensor CMPS11 Pada Body Robot

Sensor CMPS11 ini diletakan searah dengan salah satu sumbu horizontal dari pusat geometry robot, disini sensor CMPS11 diletakan searah dengan sumbu x. Tujuan peletakan searah dengan salah satu sumbu dikarenakan agar hasil arah posisi derajat yaw body robot dapat searah dengan arah mata angin, sehingga mampu menghasilkan posisi *setpoint* yang sesuai untuk proses navigasi.

3.4.1.2 Pengkabelan Modul CMPS 11 Pada Robot

Pengkabelan modul sensor CMPS11 ini dilakukan dengan menggunakan komunikasi serial UART, yaitu dengan menghubungkan pin TX dan RX pada modul dengan pin PWM 9 dan PWM 10 yang dijadikan pin serial dengan menggunakan library "`new.softwareserial`" pada mikrokontroller slave Arduino Nano, dan pin MODE pada modul CMPS11 dihubungkan dengan GND untuk mode komunikasi serial. Untuk tegangan supply, modul ini bekerja pada tegangan 5V. Untuk selanjutnya data yang diterima pada Arduino Nano ini sudah dalam bentuk data yaw, sehingga selanjutnya dikirm menuju mikrokontroller Master STM32F4 Discovery melalui komunikasi UART.



Gambar 3.14 Wiring Diagram CMPS11 dengan Arduino Nano

3.4.1.3 Pengaksesan Data Bearing CMPS 11

Pengaksesan data bearing atau kompas dari sensor CMPS11 dilakukan pada mikrokontroller Arduino Nano dengan menggunakan komunikasi serial, untuk mode komunikasi serial pada sensor CMPS11, pin mode harus dihubungkan ke ground. Default baud rate komunikasi serial pada sensor CMPS11 ini adalah 9600 bit/s dan level tegangan 3.3V – 5V.

Berikut dibawah ini adalah format pengiriman perintah mengakses nilai bearing dari Arduino Nano agar sensor CMPS11 mengirimkan data kompas nya :

```
/* Perintah mengakses data bearing 16 bit */
cmps11.write(0x13);
/* maksudnya "< 2" adalah karena data 16 bit
tersebut dipecah menjadi 8bit MSB dan 8 bit LSB
jika terpenuhi maka mikrokontroller akan membaca
data dari sensor */
if(cmps11.available() < 2);
/* membaca data 8 bit MSB */
high_byte = cmps11.read();
/* membaca data 8 bit LSB */
low_byte = cmps11.read();
/* Calculate 16 bit angle */
angle16 = high_byte;
/* menggeser data 8 bit MSB agar bisa menambahkan
8 bit LSB */
angle16 <<= 8;
/* menggabungkan data MSB dengan LSB */
angle16 += low_byte;
```

- ```
  /* meneruskan data sensor ke master */
  Serial.println(angle16/10);
  ```

3.4.1.4 Kalibrasi Arah Mata Angin CMPS11

Kalibrasi arah mata angin ini dilakukan dikarenakan terdapat ketidak samaan selisih di dalam range 0 – 359 derajat posisi sudut yaw pada sensor CMPS11. Maka dari itu dilakukan *proses kalibrasi* dengan mengacu pada empat arah mata angin yaitu : utara, barat, selatan, dan timur. Dalam hal ini utara dipilih sebagai posisi sudut acuan utama yaitu 0 atau 360 derajat .

Proses kalibrasi dilakukan dengan cara memposisikan terlebih dahulu sensor menghadap sumbu utara, setelah itu dengan menggunakan bantuan penggaris siku sensor dihadapkan menyiku dari sumbu acuan sebelumnya dengan arah berlawanan arah jarum jam dan dilakukan hingga sumbu acuan terakhir yaitu timur. Setelah itu mengambil data posisi derajat kompas pada masing masing acuan.

Setelah proses diatas selesai, maka selanjutnya dengan bantuan aplikasi exel data ke empat posisi masing masing acuan akan dilakukan proses pengolahan data kalibrasi hingga keluar range batas pada masing masing arah mata angin, seperti pada gambar 3.x dibawah ini.

Step 1 : Memasukan Arah Acuan			Step 2 : Mendefinisikan Range 45 Derajat				
// Input Arah Mata Angin				North	West	South	East
int nol north =	0	;	Normal	0	96	196	276
int nol west =	96	;	Up Limit	45	141	241	321
int nol south =	196	;	Rev. 1 UL	45	141	241	321
int nol east =	276	;	Down Lim	-45	51	151	231
Range	45		Rec.1 DL	315	51	151	231
Step 4 : Output Masing 2 Range			Step 3 : Menyesuaikan Range Agar tdk Konflik & Kosng				
// Result Range Arah Mata Angin			6	48	North	319	-6
int upn =	48	;	49		0		318
int upw =	146	;					
int ups =	226	;					
int upe =	318	;	West				East
int dwnn =	319	;	96				276
int dwnw =	49	;					
int dwns =	147	;					
int dwne =	227	;	146		South		227
			10	147	196	226	-10

Gambar 3.15 Aplikasi Pengolah Range Arah Mata Angin
  ```

Pada tahap berikutnya, data posisi derajat masing masing acuan beserta range nya akan dimasukan ke algoritma pengolahan data yang seluruhnya dilakukan di STM32F4 Discovery.

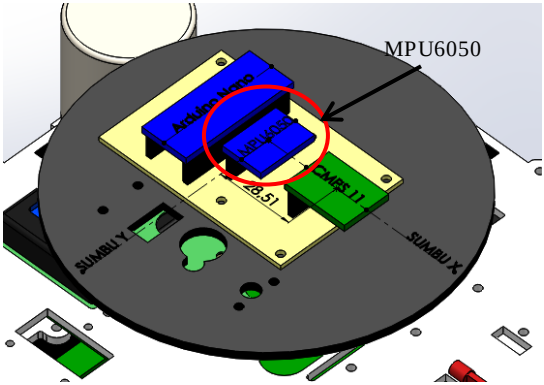
3.4.2 Perancangan Sistem Modul Sensor GY-521 Pada Robot

Modul sensor GY-521 ini menggunakan chip MPU6050 dari InvensSense yang terdiri dari dua sensor IMU yaitu MEMS giroskop dan MEMS akselerometer yang masing masing memiliki tiga sumbu axis x,y, dan z.

Pada Tugas Akhir ini *modul sensor* MPU6050 digunakan sebaga *feedback* posisi derajat yaw dari body terhadap lingkungan pada saat robot melakukan gerak berjalan lurus. Sensor IMU yang diakses dari modul MPU6050 adalah sensor giroscope dan sensor akselerometer. Dengan menggunakan library dari *"Jeff Rowberg"* yang terdapat pada aplikasi Arduino IDE, maka *raw data* dari kombinasi kedua sensor tersebut dapat diolah hingga menjadi data posisi derajat *yaw*, *pitch*, dan *roll*, yang selanjutnya data *yaw* digunakan sebagai *feedback* posisi derajat kemiringan yaw body robot terhadap lingkungan.

3.4.2.1 Penempatan Modul GY-521 Pada Robot

Desain rancangan penempatan modul sensor MPU6050 pada body robot ada pada gambar 3.16 dibawah ini.

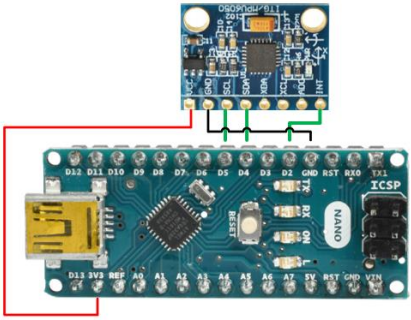


Gambar 3.16 Desain Peletakan Sensor MPU6050 Pada Body Robot

Modul sensor MPU6050 ini diletakan pada pusat sumbu horizontal dari pusat geometry robot. Tujuannya agar menghindari terjadinya perubahan posisi translasi pada sumbu x dan y, karena dapat mengakibatkan terjadinya pembacaan akselerasi yang dapat mengganggu pembacaan sudut yaw sensor.

3.4.2.2 Pengkabelan Modul GY-521 Pada Robot

Pengkabelan modul sensor MPU6050 ini dilakukan dengan menggunakan komunikasi I2C, yaitu dengan menghububgkan pin SDA dan SCL pada modul dengan pin A4 (SDA) dan A5 (SCL) pada mikrokontroller slave Arduino Nano, dan pin INT pada modul dengan pin 2 pada Arduino. Untuk tegangan supply, modul ini bekerja pada tegangan 2.375V ~ 3.46V. Untuk selanjutnya data yaw hasil pengolahan pada Arduino Nano ini dikirm menuju mikrokontroller Master STM32F4 Discovery melalui komunikasi UART.



Gambar 3.17 Wiring Diagram Modul GY-521 dengan ArduinoNano

3.4.2.3 Pengaksesan Data Yaw Modul GY-521

Pengaksesan data posisi derajat yaw dari modul sensor MPU6050 dilakukan pada mikrokontroller Arduino Nano dengan menggunakan komunikasi serial I2C dan dengan bantuan library dari *"Jeff Rowberg"*. Proses pengolahan data pada library tersebut dilakukan dengan mengolah data mentah (*raw data*) dari modul MPU6050 menggunakan metode perhitungan *matrix rotasi* dan *euler matrix* yang selanjutnya diolah kembali menggunkan *quarternion* hingga menjadi data posisi derajat kemiringan *yaw, pitch,* dan *roll*.

Berikut pada gambar 3.18 adalah cuplikan program pada Arduino IDE dengan menggunakan library untuk mengakses nilai *yaw*, *pitch*, dan *roll* dari modul sensor MPU6050.

```
313 #ifdef OUTPUT_READABLE_YAWPITCHROLL
314   // display Euler angles in degrees
315     mpu.dmpGetQuaternion(&q, fifoBuffer);
316     mpu.dmpGetGravity(&gravity, &q);
317     mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
318     Serial.print("ypr\t");
319     Serial.print(ypr[0] * 180/M_PI); // => YAW
320     Serial.print("\t");
321     Serial.print(ypr[1] * 180/M_PI); // => PITCH
322     Serial.print("\t");
323     Serial.print(ypr[2] * 180/M_PI); // => ROLL
324 #endif
```

Gambar 3.18 Cuplikan Program Pada Arduino IDE

3.4.2.4 Kalibrasi Pada Modul Sensor GY-521

Proses kalibrasi pada modul MPU6050 ini dilakukan karena pada saat proses pembacaan data yaw pertama kali data tersebut masih belum konstan sehingga masih selalu bertambah hingga kurun waktu tertentu dan menjadi nilai offset sudut yaw.

Maka tujuan dilakukannya proses ini adalah agar proses penambahan offset sudut yaw tersebut dapat dibatasi oleh waktu tertentu dengan menggunakan fungsi *"millis()"* pada Arduino IDE. Selanjutnya hasil offset sudut pada kurun waktu yang sudah ditentukan tersebut digunakan untuk untuk mengurangi sudut yaw output sehingga didapat hasil sudut yaw yang baru sebesar nol derajat.

Berikut dibawah ini adalah format penulisan kalibrasi modul MPU6050 pada aplikasi Arduino IDE :

```
/* => proses kalibrasi */
if(statusKalibrasi == 0){
dataYaw = ypr[0] * 180/M_PI;
tSekarang = millis();
dT = tSekarang - tSebelum;
/* a. mengecek apakah "waktu cek" sudah terpenuhi,
jika belum, maka menghitung yawSekarang */
 if(dT < WAKTU_CEK){
 yawSekarang=ypr[0];
 }
/* b. Jika waktu cek sdh melebihi waktu minimal */
 else{
 yawSebelum = yawSekarang;
 dYaw = abs(yawSekarang - yawSebelum); // error
  if(dYaw < ERROR_MINIMAL){
  counterKalibrasi++ ;
  }
  else counterKalibrasi = 0;
  if (counterKalibrasi>2){
  statusKalibrasi = 1;
  offsetSudut = yawSekarang; // kalibrasi beres
  }
 tSebelum = tSekarang; // rekayasa ulang millis
 }
}
/* jika kalibrassi telah selesai */
else{
digitalWrite(led, HIGH);
yaw2 = (ypr[0]-offsetSudut) * 180/M_PI + 180;
Serial.print("xy")
Serial.println(yaw2); // mengirim ke STM32F4 Disc.
}
```

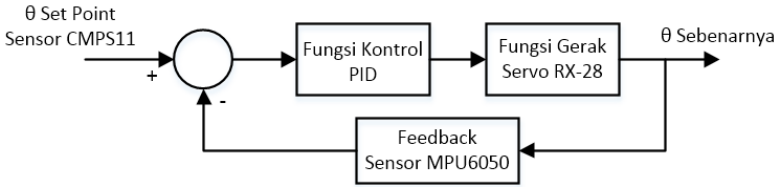
Pada proses kalibrasi di atas, seluruhnya dilakukan pada mikrokontroller slave Arduino Nano, sehingga yang dikirim ke mikrokontroller masterSTM32F4 adalah data yaw yang sudah terkalibrasi.

3.5 Perancangan Sistem Kontrol Pada Robot

Dalam pengaplikasian *sensor IMU* pada robot *quadrupet* ini diperlukan susatu sistem kontrol yang diterapkan pada robot agar dapat bekerja sesuai dengan keinginan.

Dalam Tugas Akhir ini digunakan sistem kontrol PID untuk dapat mengaplikasikan sensor IMU pada robot quadrupet tersebut. Tujuan pengaplikasian sensor IMU dan sistem kontrol PID sendiri sebenarnya adalah untuk memperbaiki gerak berjalan lurus pada robot dikarenakan ketidak sempurnaan struktur mekanik robot maupun lingkungan yang tidak rata, sehingga gerak langkah robot yang ditargetkan lurus menjadi sedikit berbelok-belok.

Kontrol PID digunakan agar mendapat respot yang cepat untuk mencapai keadaan *steady state*. Berikut pada gambar 3.19 adalah blok diagram kontrol PID yang diterapkan pada robot untuk memperbaiki gerak berjalan lurus pada robot.



Gambar 3.19 Blok Diagram Kontrol PID

Dikarenakan sistem kontrol PID ini diterapkan pada mikro-kontroller yang dimana pengolahan sinyal dilakukan hanya pada waktu diskrit dalam sistem, maka sistem kontrol PID yang digunakan adalah sistem kontrol PID Digital. Dalam hal ini, konversi sinyal dari kontinyu ke digital, pengolahan sinyal error, sampai konversi balik digital ke kontinyu dilakukan pada interval atau waktu sampling tertentu.

Untuk dapat mengimplementasikan PID digital di mikrokontroler, maka kontroler PID kontinue harus diubah terlebih dahulu ke bentuk digital. Penurunan kontroler PID digital dapat dilihat pada Persamaan 3.1 sampai dengan Persamaan 3.4. Ketelitian PID digital yang di dapat dari diskretisasi ini sangat tergantung dari lebar waktu sampling yang digunakan.

Berikut ini persamaan matematis dari kontrol PID adalah sebagai berikut :

$$u(t) = K_P \times e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \text{ ... (3.1)}$$

Sehingga, untuk bentuk integral dan diferensial dapat ditulis dalam bentuk diskrit seperti pada Persamaan 3.2 dan Persamaan 3.3 berikut :

$$\int_0^t e(t)dt \approx T \sum_{K=0}^{K} e(k) \text{ ... (3.2)}$$

$$\frac{de(t)}{dt} \approx \frac{e_k - e_{k-1}}{T} \text{ ... (3.3)}$$

Sehingga diperoleh dalam bentuk kontroler PID diskrit ialah sebagai berikut :

$$u_{(k)} = K_P \times e_k + K_i \times T \times \sum_0^K e_k + \frac{1}{T} \times K_d \times (e_k - e_{k-1}) \text{ ... (3.4)}$$

Maka dalam bentuk pseudocode, persamaan kontrol PID digital tersebut dapat disusun sebagai berikut :

```
error_sblm = 0
Ts = 1;

start :
error = Sp - θ
pid =  Kp*error + Ki*Ts*(error+error_sblm) +
       (Kd/Ts)*(error - error_sblm)
error_sblm = error
goto start
```

Dimana :

- Kp, Ki, Kd adalah parameter kontrol PID
- Sp adalah parameter set point kontrol PID
- θ (theta) adalah sudut aktual posisi derajat yaw
- error adalah nilai kesalahan terbaru
- error_sblm adalah nilai kesalahan sebelumnya
- Ts adalah waktu sampling (waktu cuplik)

Dalam menentukan nilai Kp, Ki, dan Kd dalam kontrol PID dilakukan dengan menggunakan metode *manual tuning* atau bisa disebut *tuning eksperiment*. Inti dari metode tuning ini adalah menentukan

ketiga kontrol PID yaitu Kp, Ki, dan Kd dengan cara bereksperiment hingga didapat ketiga kombinasi tercepat menuju ke keadaan *steady state*.

Sebagai langkah atau acuan dalam penentuan parameter Kp, Ki, dan kd dengan metode manual tuning, diadopsi dari *Williams, C* dalam *"feedback and temperature control"*, Langkah metode tersebut ialah sebagai berikut :

- Langkah awal gunakan kontrol proporsional terlebih dahulu, abaikan konstanta integratif dan derivatifnya dengan memberikan nilai nol pada integratif dan derivatif.
- Tambahkan terus konstanta proporsional maksimum hingga keadaan stabil namun robot masih berosilasi.
- Untuk meredam osilasi, tambahkan konstanta diferensial dengan membagi dua nilai proporsional, amati keadaan sistem robot hingga stabil dan lebih responsif.
- Jika sistem robot telah stabil, kontrol integral dapat menjadi opsional, dalam artian jika ingin mencoba-coba tambahkan kontrol integral tersebut, namun pemberian nilai integral yang tidak tepat dapat membuat sistem robot menjadi tidak stabil.
- Nilai sampling time (waktu cuplik) juga mempengaruhi perhitungan PID, tentunnya saat penggunaan kontrol integral dan diferensial.
- Periksa kembali perfoma sistem hingga mendapatkan hasil yang memuaskan

Berikut dalam tabel 4.1 ini, tertera efek dari masing masing kontroler terhadap sistem, yang diadopsi dari (williams, 2006). [10]

Tabel 3.1. Efek pada masing-masing konroller Kp, Ki, dan Kd terhadap kinerja sistem kontrol PID

Parameter	Rise Time	Overshoot	Settling Time	Steady-State Error	Stability
Kp	Menurun	Meningkat	Sedikit Berubah	Menurun	Menurun
Ki	Menurun	Meningkat	Meningkat	Tereleminasi	Menurun
Kd	Sedikit Berubah	Menurun	Menurun	Tidak Berpengaruh	Meningkat juka Kd kecil

Dari hasil metode *tuning PID* di atas tersebut didapatkan nilai Kp sebesar 1,75, Ki sebesar 0, dan Kd sebesar 0,75. Maka selanjutnya adalah membuat fungsi kontroller PID pada program dengan menggunakan bahasa C. Berikut dibawah ini adalah cuplikan dari program fungsi PID.

```
void PIDmpu(){
 // valmpu6050 => NILAI MPU6050
 // dalam kasus ini void getspmpu harus sudah di
panggil untuk menentukan setpint
  errormpu = (spmpu - valmpu6050);
 // pembagi antara error min dan plus dari data 360
derajat
  if (errormpu > 180) errormpu -= 360;         //
untuk -179
  else if (errormpu < -180)errormpu += 360;    //
untuk 179
 // kotroller PID
  pidmpu = - (kpmpu*errormpu + kimpu*(errormpu +
errorsmpu) + kdmpu*(errormpu + errorsmpu));
  sudutBelok = pidmpu;
 // Pembatas Sudut
  if (sudutBelok >= 3)sudutBelok = 3;          //
sudut + untuk belok kiri
  if (sudutBelok <= -3)sudutBelok = -3; //   sudut   -
untuk belok kanan
 // fungsi mengirim error melalui bluetooth HC05
  USART_Puts(USART2,tampil);
  errorsmpu = errormpu;}
```

Setelah fungsi kontrol PID tersebut dibuat, maka selanjutnya adalah mengirimkannya ke fungsi gerak pada servo, disini nilai kontroller PID adalah besaran sudut, yaitu untuk meluruskan gerak berjalan pada robot, sehingga di dalam fungsi gerak kontroller PID berfungsi sebagai masukan omega (sudut belok). Berikut dibawah ini adalah culikan susunan fungsi gerak pada program dengan menggunakan bahasa C.

```
void navigasilurus(){
  kecepatan = 50;    // Kecepatan Trayektori
  langkah = 3;       // lebar langkah
  tinggi = 1;        // tinggi langkah dari tanah

  // memanggil fungsi kontrol PID untuk mendapatkan
sudut belok
  PIDmpu();
  // mengirimkan perintah untuk gerak servo
  gerak(0, langkah, sudutBelok, tinggi, kecepatan);
}
```

Halaman ini sengaja dikosongkan

BAB IV
PENGUJIAN

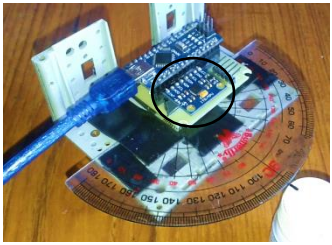
Pada bab ini akan dibahas mengenai hasil pengujian dan analisa terhadap sisitem yang telah didesain dan dirancang sebelumnya. Tujuan dari pengujian ini adalah untuk mengetahui apakah perancangan sistem, seperti pengaplikasian sensor dan sistem kontrol sudah dapat berjalan cukup baik atau belum dan akan dilakukan analisa data dari hasil yang diperoleh sebagai referensi untuk percobaan dan pengembangan berikutnya. Pengujian yang dibahas dalam hal ini terdiri dari pengujian karakterisasi sensor IMU, pengujian gerak robot berjalan lurus tanpa PID dengan variable kecepatan, dan pengujian gerak robot berjalan lurus dengan PID.

4.1. Pengujian Karakteristik Sensor IMU

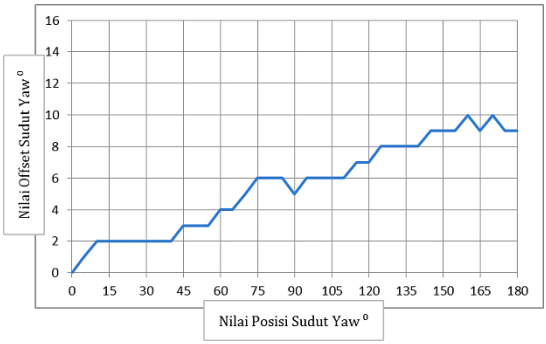
Pada pengujian ini dilakukan untuk mengetahui krakteristik modul sensor IMU dalam pembacaan nilai posisi derajat yaw terhadap nilai posisi sudut yaw yang sebenarnyadan untuk mendapatkan nilai rata-rata kesalahan. Dalam pengujian ini, alat bantu yang digunakan adalah penggaris busur, dengan ketelitian pembacaan posisi derajat yaw satu derajat.

4.1.1. Pengujian Modul Sensor ITG-521/MPU6050

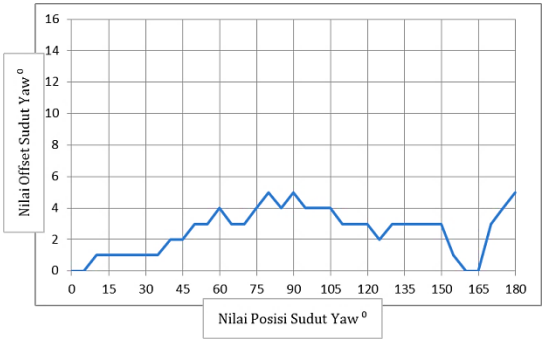
Pada pengujian ini, sensor IMU yang diakses dari modul MPU6050 adalah kombinasi sensor akselerometer dan giroskop, dan dengan menggunakan library dari Jeff Rowberg pada software Arduino IDE sehingga didapatkan nilai posisi sudut yaw dari modul tersebut. Range sudut yaw yang akan diuji adalah 0° sampai 179° dan -179° sampai 0° dengan selisih penambahan adalah 5°.



Gambar 4.1 Pengujian sudut yaw mpu6050 dengan penggaris busur



Gambar 4.2 Grafik hasil pengujian offset sudut yaw terhadap range 0° sampai 180° dengan sampling pengambilan data tiap 5°



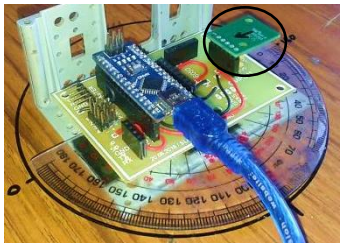
Gambar 4.3 Grafik hasil pengujian offset sudut yaw terhadap range 0° sampai -180° dengan sampling pengambilan data tiap 5°

Dari hasil pengujian yang telah dilakukan pada modul sensor MPU6050, pada grafik di atas didapat kesalahan pembacaan posisi sudut yaw rata-rata oleh sensor untuk sudut 0° sampai 180° adalah 5,514° dan untuk sudut 0° sampai -180° adalah 2,586°, dengan kesalahan tertinggi adalah 10°. Tentunya hasil ini akan berbeda untuk modul sensor MPU6050 yang lainnya.

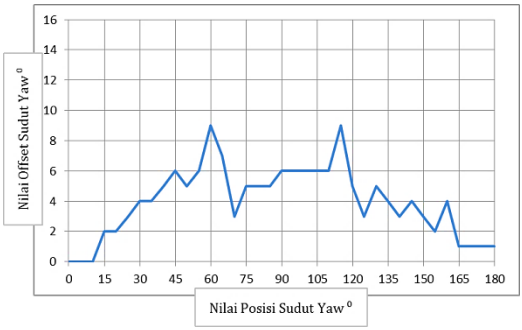
4.1.2. Pengujian Modul Sensor CMPS11

Pada pengujian ini, sensor IMU yang diakses dari modul CMPS11 adalah kombinasi sensor giroskop dan magnetometer. Untuk mengakses nilai sudut yaw compas, menggunakan komunikasi serial digunakan

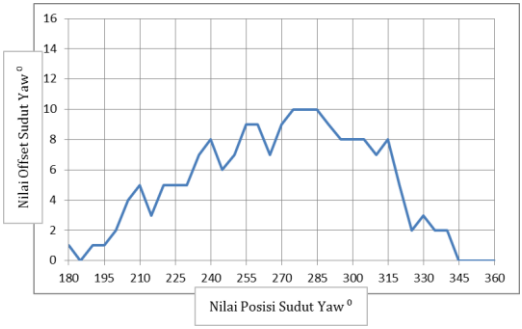
perintah 0x13. Pada pengujian ini, range sudut yaw yang akan diuji adalah 0° sampai 360° dengan selisih penambahan adalah 5°.



Gambar 4.4 Pengujian sudut yaw mpu6050 dengan penggaris busur a



Gambar 4.5 Grafik hasil pengujian offset sudut yaw terhadap range 0° sampai 180° dengan sampling pengambilan data tiap 5°



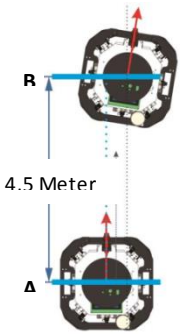
Gambar 4.6 Grafik hasil pengujian offset sudut yaw untuk sudut 180° sampai 360° dengan sampling pengambilan data tiap 5°

Dari hasil pengujian yang telah dilakukan pada modul sensor CMPS11, pada grafik di atas didapat kesalahan pembacaan posisi sudut yaw rata-rata oleh sensor untuk sudut 0° sampai 180° adalah 3,973° dan untuk sudut 0° sampai -180° adalah 5,139°, dengan kesalahan tertinggi adalah 10°. Tentunya hasil ini akan berbeda untuk modul sensor CMPS11 yang lainnya.

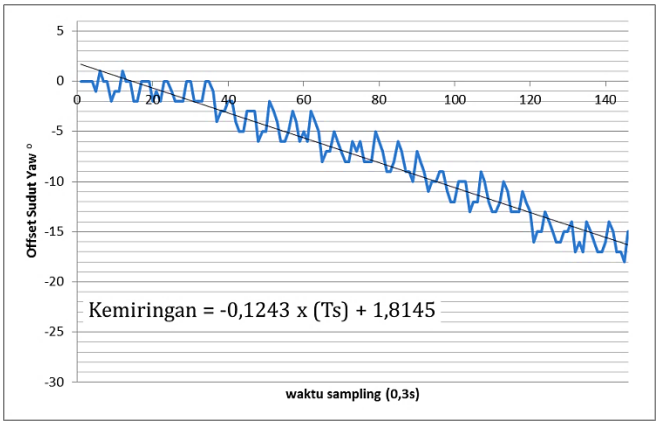
4.2. Pengujian Gerak Robot Berjalan Lurus

Pegujian ini bertujuan untuk menguji sistem mekanik yaitu pada bagian frame kaki robot pada saat melakukan gerak berjalan lurus, apakah sudah dapat berfungsi dengan baik pada saat berjalan lurus sehingga dapat mempertahankannya ataukah masih terdapat kesalahan (seperti perbedaan panjang lengan, dan bentuk karet pada ujung kaki) sehingga robot melenceng dari jalan yang lurus.

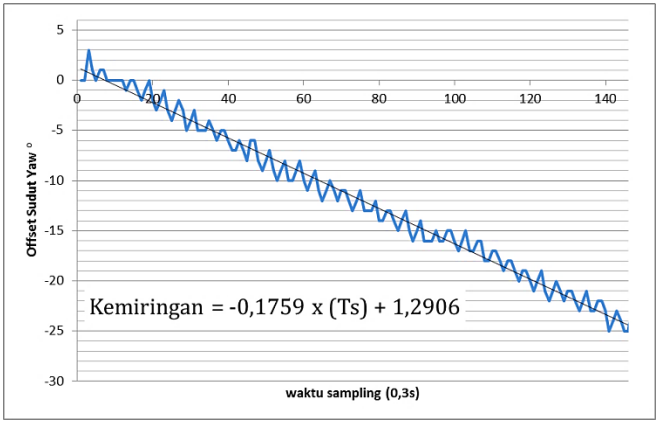
Pengujian ini dilakukan dengan menguji robot berjalan lurus pada jarak 4,5 meter, dengan variable kecepatan trayektori dari 30 cm/s sampai 70 cm/s dengan selisih penambahan 10 cm/s, ditentukannya jarak 4,5 meter tersebut adalah agar hasil pengujian dapat cuku akurat. Dalam membaca besar kesalayan kemiringan sudut yaw robot menggunakan bantuan sensor IMU MPU6050, dan dalam mengirim kesalahan dari robot menuju PC menggunakan bantuan perangkat nirkabel bluetooth HC-05 pada STM32F4 Disc. Waktu sampling yang digunakan dalam mengirim data kesalahan tersebut melalui bluetooth adalah 0,3 detik.



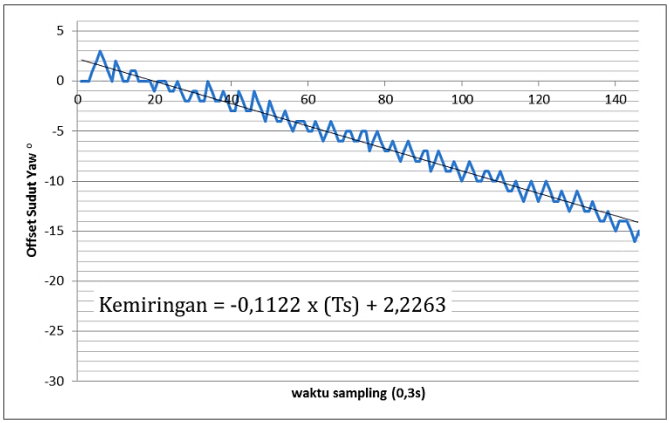
Gambar 4.7 Ilustrasi pengujian robot berjalan lurus dari posisi awal A ke posisi akhir B



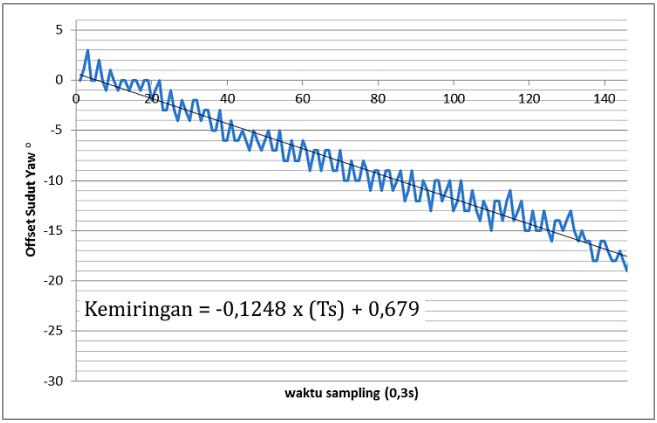
Gambar 4.8 Hasil kesalahan kemiringan yaw robot terhadap waktu sampling 0,3 second pada kecepatan trayektory 30 cm/s



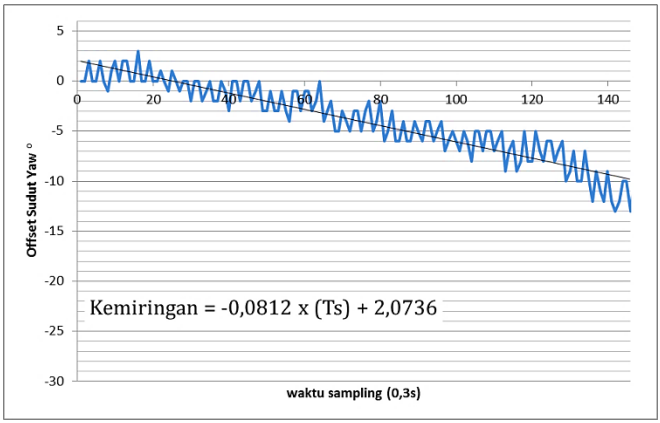
Gambar 4.9 Hasil kesalahan kemiringan yaw robot terhadap waktu sampling 0,3 second pada kecepatan trayektory 40 cm/s



Gambar 4.10 Hasil kesalahan kemiringan yaw robot terhadap waktu sampling 0,3 second pada kecepatan trayektory 50 cm/s

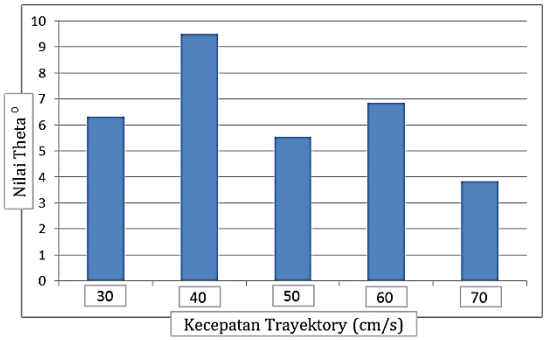


Gambar 4.11 Hasil Kesalahan Kemiringan Yaw Robot terhadap Waktu Sampling 0,3 second Pada Kecepatan Trayektory 60 cm/s



Gambar 4.12 Hasil Kesalahan Kemiringan Yaw Robot terhadap Waktu Sampling 0,3 second Pada Kecepatan Trayektory 70 cm/s

Dari hasil pengujian yang telah dilakukan, pada grafik di atas dapat disimpulkan bahwa kesalahan kemiringan yaw robot akan selalu bertambah setiap waktunya dan lebih cenderung berbelok ke arah kanan pada jarak tempuh 4,5 meter, dengan kesalahan terbesar ke kanan adalah 25° dan ke kiri adalah 3°. Hal ini disebabkan karena kurang presisinya frame kaki robot sehingga tiap langkah pada kaki robot memiliki jarak yang berbeda-beda. Faktor lain yang menyebabkan ini terjadi juga dikarenakan posisi *default* selalu berubah pada saat penggantian servo.



Gambar 4.13 Nilai *Theta* Penambahan Offset Yaw Pada Masing-Masing Kecepatan Trayektory Robot

Dari gambar 4.8 sampai gambar 4.12 didapat persamaan linier penambahan kemiringan (nilai offset sudut yaw robot) dari masing-masing kecepatan trayektory, sehingga didapat nilai *theta* kemiringan penambahan nilai offset sudut yaw robot terhadap waktu sampling hingga 146 adalah seperti pada gambar 4.13.

Maka dari itu diperlukan suatu sistem kontrol sebagai salah satu solusi untuk mengatasi kesalahan kemiringan robot akibat kurang presisinya geometri frame dari kaki robot mamupun karakteristik elektronik robot yang tidak dapta dihilangkan. Sistem kontrol tersebut adalah dengan menggunakan sistem kontrol PID, fungsi dari sistem kontrol ini adalah untuk memanipulasi nilai kesalahan keimiringan sudut yaw robot menjadi nilai yaw sudut belok robot, sehigga sebenarnya robot tetap pembelokan, namun pembelokan tersebut diatasi dengan dengan memasukan sudut belok hasil kontrol PID, namun dengan arah yang berlawanan, sehingga robot dapat tetap berjalan lurus.

4.3. Penentuan dan Analisa *Center of Gravity* Body Robot

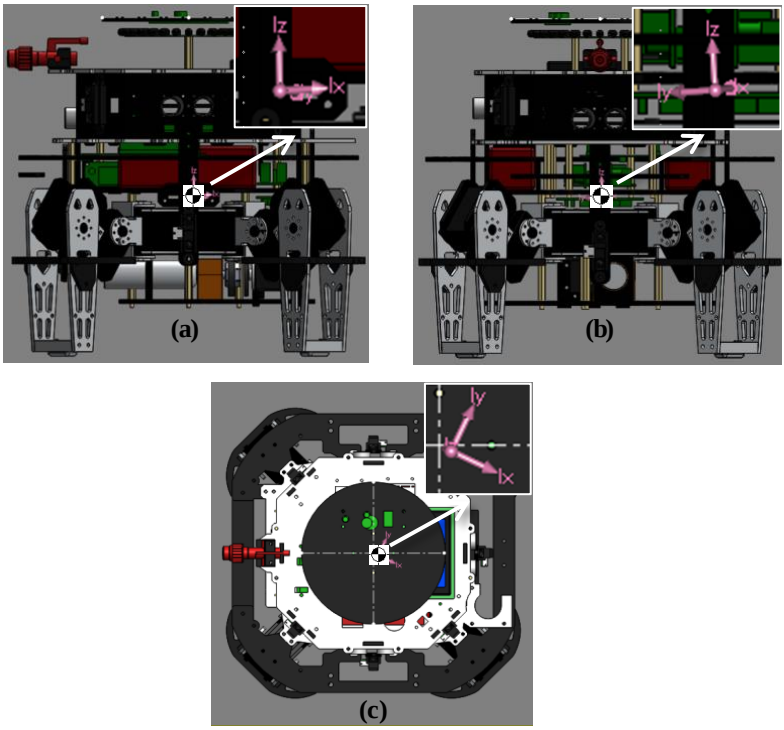
Penentuan titik berat atau center of gravity dari body robot (body atas dan frame kaki) dilakukan dengan bantuan software SolidWorks dengan menggunakan fitur Mass Propertis. Sistem sumbu kartesian yang digunakan berpusat pada pusat geometry robot dengan sumbu x adalah sumbu longitudinal, sumbu z adalah sumbu lateral, dan y adalah sumbu normal. Kecuali pada sumbu y berpusat di titik tertinggi geometry robot sebagai acuan, hal ini dikarenakan posisi kaki robot tidak tetap. Maka dari hasil perhitungan dengan software SolidWorks didapat massa total robot dan titik berat adalah sebagai berikut :

- $\mathbf{m_{total}}$ = 2,6213 Kg
- **x** = 0,004327 m (4,33 mm)
- **y** = -0,13575 m (-135,75 mm)
- **z** = 0,0021074 m (2,11 mm)

Dari hasil tersebut, didapat terjadi offset dari posisi nol pusat geometry robot terhadap center of gravity, sehingga menyebabkan terjadinya moment inersia yang tidak seimbang pada robot yang berefek pada kecenderungan robot untuk berbelok saat melakukan gerak berjalan lurus. Maka dilakukan perhitungan moment inersia pada masing-masing sumbu dengan menggunakan bantuan software SolidWorks dan didapat :

- **Ix** = 0.019158 Kg.m^2
- **Iz** = 0.020277 Kg.m^2
- **Iy** = 0.023658 Kg.m^2

Dari hasil tersebut, didapat nilai dari masing-masing moment inersia tidak sama dengan nilai terbesar adalah pada sumbu yaw sebesar 0.023658 Kg.m^2 yang berefek pada gerak berjalan lurus pada robot. Pada gambar 4.14 adalah ilustrasi letak posisi titik berat dan moment inersia body robot terhadap pusat geomrrtry robot, tampak depan, tampak samping, dan tampak atas.

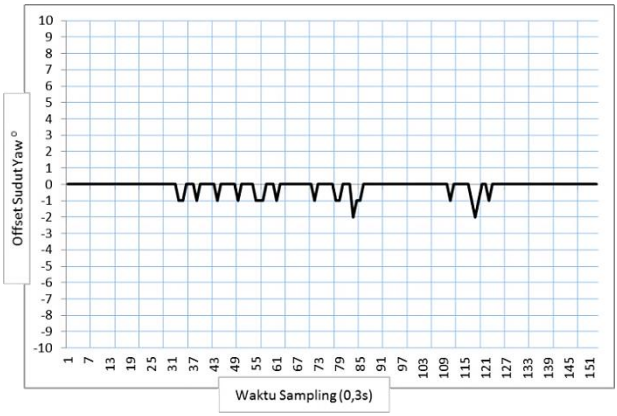


Gambar 4.14 Center of gravity body robot (a) tampak samping, (b) tampak depan, (c) tampak atas

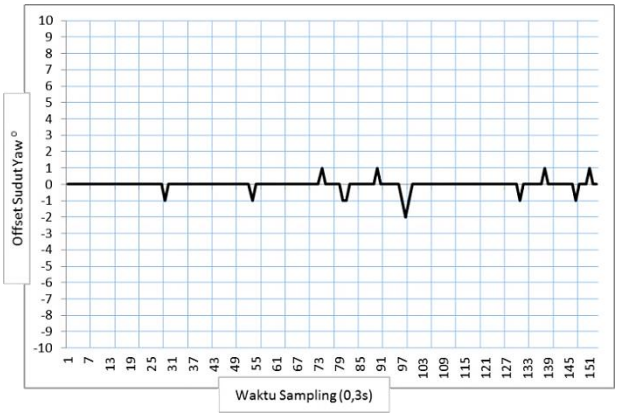
4.4. Pengujian Gerak Robot Berjalan Lurus Dengan Kontrol PID

Pada pengujian ini bertujuan untuk menguji peforma (respon dalam mengatasi error) dari sistem kontrol PID yang mengaplikasikan sensor IMU MPU6050 sebagai feedback dari sistem kontrol tersebut yang selanjutnya diterapkan pada robot untuk mengatasi kesalahan kemiringan sudut yaw robot saat gerak berjalan lurus akibat karakteristik geometri mekanik maupun karakteristik dari sistem elektronik. Kontroller PID tersebut terdiri dari *kontrol proporsional, kontrol integral, dan kontrol derivative* yang selanjutnya dari ketiga kontroler tersebut dijumlahkan. Dalam menentukan masing- masing nilai konstanta Kp, Ki, dan Kd dilakukan dengan menggunakan metode *manual tuning* PID. Dari hasil tuning tersebut didapat nilai Kp sebesar 1.75, Ki sebesar 0, dan Kd sebesar 0.75 yang mendapatkan respon relatif stabil.

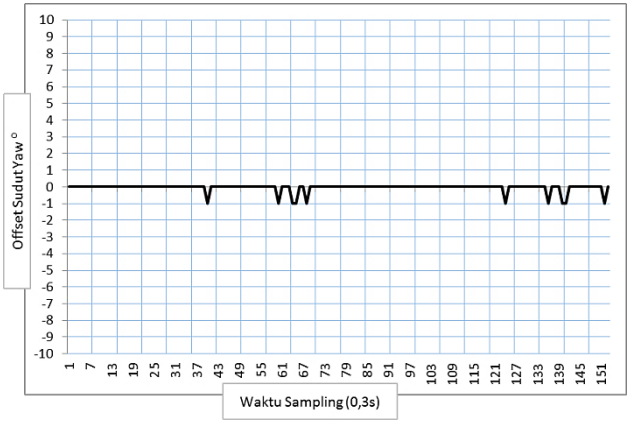
Pada pengujian ini dilakukan pada range jarak 4,5 meter dengan variabel kecepatan trayektori dari 30 cm/s sampai 70 cm/s dengan selisih 10 cm/s, untuk membandingkannya dengan gerak berjalan lurus tanpa PID Dalam pengujian ini digunakan modul sensor MPU6050 untuk mendeteksi *feedback* kesalahan dari robot, dan bluetooth HC-05 untuk mengirimkan nilai kesalahan tersebut ke PC.



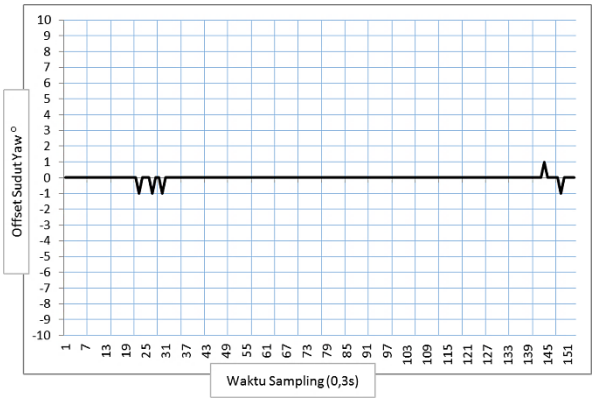
Gambar 4.15 Hasil Kemiringan Yaw Robot dengan waktu sampling 0,3 second pada kecepatan trayektori 30 cm/s



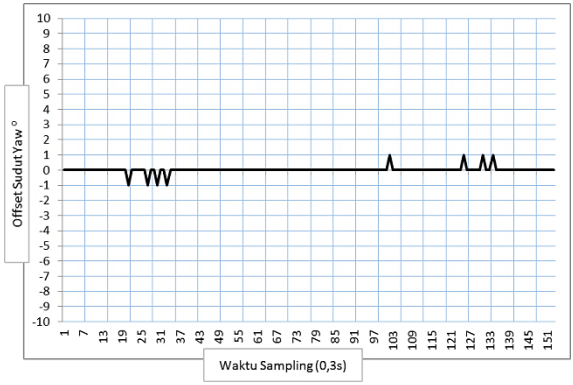
Gambar 4.16 Hasil Kemiringan Yaw Robot dengan waktu sampling 0,3 second pada kecepatan trayektori 40 cm/s



Gambar 4.17 Hasil Kemiringan Yaw Robot dengan waktu sampling 0,3 second pada kecepatan trayektori 50 cm/s



Gambar 4.18 Hasil Kemiringan Yaw Robot dengan waktu sampling 0,3 second pada kecepatan trayektori 60 cm/s

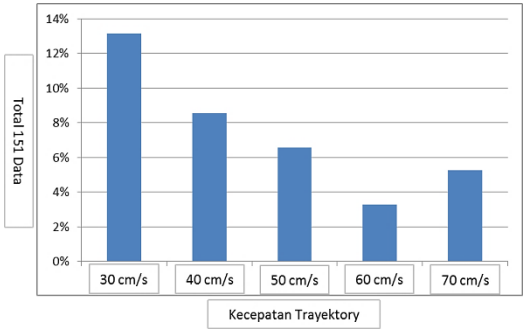


Gambar 4.19 Hasil Kemiringan Yaw Robot dengan waktu sampling 0,3 second pada kecepatan trayektori 70 cm/s

Dari hasil pengujian yang telah dilakukan, dengan menerapkan kontroller PID pada robot tersebut didapatkan kontroller tersebut sudah berfungsi dengan baik, sehingga robot sudah dapat melakukan gerak berjalan lurus dengan lurus, walaupun terkadang terdapat kesalahan namun robot dapat meluruskan gerak langkahnya lagi. Dari pengujian ini didapat kesalahan terbesar dengan jarak 4,5 meter adalah 2°.

Dari Gambar 4.14 sampai Gambar 4.18 tersebut didapat presentase terjadinya kesalahan dari 151 sampling pengambilan data pada

pengujian robot gerak berjalan lurus dengan kontrol PID terhadap masing-masing kecepatan trayektory adalah sebagai berikut :



Gambar 4.20 Presentase Terjadinya Kesalahan Terhadap Variasi Kecepatan Trayektory

Halaman ini sengaja dikosongkan

BAB V
PENUTUP

5.1. Kesimpulan

Pada tugas akhir dari aplikasi sensor IMU pada robot quadruped yang telah dilakukan pengujian ini, bisa diambil kesimpulan sebagai berikut :

1. Karakteristik kesalahan pembacaan nilai sudut yaw oleh kedua modul sensor IMU adalah sebagai beriku :
   a. Pada pembacaan nilai sudut yaw oleh modul sensor GY-521 didapat kesalahan rata-rata untuk sudut 0° sampai 179° adalah 5,514° dan untuk sudut -179° sampai 0° adalah 2,586°.
   b. Pada pembacaan nilai sudut yaw oleh modul sensor CMPS11 didapat kesalahan rata-rata untuk sudut 0° sampai 180° adalah 3,973° dan untuk sudut 180° sampai 360° adalah 5,139°.
2. Pada perancangan desain sistem sensor IMU GY-521 (mekanik dan elektronik) pada robot quadruped didapat :
   a. Untuk desain mekanik, posisi sensor IMU pada bidang horizontal diletakan pada pusat gravitasi robot yaitu pada koordinat (4.33 mm, 2.11 mm) dari pusat geometry robot (0 mm, 0 mm).
   b. Untuk desain elektronik, menggunakan mikrokontroller Arduino Nano sebagai pengolah data IMU dengan komunikasi serial I2C untuk mengirim dan menerima data.
3. Dengan bantuan sensor IMU dan sistem kontrol PID yang diterapkan pada robot, didapat robot sudah dapat berjalan lurus dengan presentase kesalahan untuk 151 smapling data, terkecil adalah pada kecepatan trayektory 60 cm/s sebesar 3% dan terbesar pada kecepatan trayektory 70 cm/s sebesar 13%.

5.2. Saran

Untuk pengembangan lebih lanjut mengenai tugas akhir ini, disarankan untuk melakukan beberapa langkah lanjutan :

1. Dikarenaan dalam pengolahan data sensor IMU disini masih menggunakan library oleh *Jeff Rowberg*, maka untuk melanjutkan riset mengenai sensor ini lebih mendasar, bisa dengan menggunakan implementasi *quarternion* dengan *sudut euler* untuk

mendapat-kan nilai yaw, pitch, dan roll dengan mengolah data mentah sensor tersebut.

2. Masih terdapat kesalahan utama dalam pembacaan yaw oleh sensor IMU yang digunakan, antara lain :
   a. Sensor IMU yang digunakan belum mampu mendeteksi kesalahan robot akibat terjadinya pergeseran ke arah samping kanan maupun kiri, sehingga diperlikannya kombinasi dengan menggunakan sensor GPS.
   b. Pada pembacaan nilai nol posisi sudut yaw, masih terjadi offset setelah menempuh jarak tertentu akibat banyaknya getaran dan kecepatan yang tidak konstan, sehingga diperlukan sensor Compass dan Kalman Filter untuk mendeteksi kesalahan pembacaan nol tersebut, dan mengaturnya tiap jarak tertentu.

DAFTAR PUSTAKA

[1] Direktorat Kemahasiswaan, Ristekdikti, "Kontes Robot Pemadam Api Indonesia (KRPAI) - 2018", Jakarta, 2018.

[2] P. Barry, "Fundamentals of Digital Electronics", National Instrument Corporate Headquartes, Texas, 1998.

[3] F. Jacob, "Handbook of Modern Sensors", Advanced Motions Corporation, San Diego, 2003, hal.278 dan 325

[4] M. Arifin, "Implementasi Pusat Tekanan (COP) Untuk Kontrol Keseimbangan Postur Pada Robot Humanoid", Departemen Teknik Elektro ITS, Surabaya, 2017.

[5] W. Wahyu Tri, "Implementasi Invers Kinematik Pada Robot Qudruped", Departemen Teknik Elektro ITS, Surabaya, 2017

[6] K. Satoshi, "TITAN-XIII: sprawling-type quaduped robot with ability of fast and energy effecient walking", Tokyo Institute of Tehnology, 2016.

[7] O. Kai Daniel, "Inertial Measurement Unit (IMU) Technology", Institute of Biomechanics and Orthopeadics, 2015

[8] H. Jiangbo, "Structural Designing of a MEMS Capacitive Accelerometer for Low Temperature Coeffecient and High Linearity, Xihua University, China, 2018

[9] "Hall Effect Sensor". https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html. Diakses 16 Juli 2018

[10] William, C. "Feedback and Temperature Control". http://newton.ex.ac.uk/teaching/CDHW/feedback/setup-PID.html. Diakses 16 Juli 2018

[11] InvenSense, "MPU-6000 and MPU-6050 Product Specification Revision 3.4", 2013.

[12] STMicroelectronics, "LSM9DS0, iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer", 2013.

[13] Devantech, "CMPS11 – Tilt Compensated Compas Module", 2015.

[14] Robotis, "User's Manual Dynamixel RX-28", V1.10.

[15] STMicroelectronics, UM1472 User manual Discovery kit for STM32F407/417 lines, 2014.

[16] SolidState Technology, "Introduction to MEMS gyroscope", 2010.

[17] "PID Controller". http://staffnew.uny.ac.id/upload/132206815/pendidikan/pid-controller.pdf. Diakses 16 Juli 2018

Halaman ini sengaja dikosongkan

LAMPIRAN

A. Program Akses Compass CMPS11 Pada Arduino NANO

```
cmps11.write(CMPS_GET_ANGLE16);
if(cmps11.available() < 2);
high_byte = cmps11.read();
low_byte = cmps11.read();
angle16 = high_byte; // Calculate 16 bit angle
angle16 <<= 8;
angle16 += low_byte;
yaw3 = angle16 / 10; // fungsi baca
Serial.println(yaw3);
```

B. Program Kalibrasi MPU6050 Pada Arduino NANO

```
#ifdef OUTPUT_READABLE_YAWPITCHROLL // display Euler angles°
mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
if(statusKalibrasi == 0) // proses kalibrasi
{
dataYaw = ypr[0] * 180/M_PI;
//tSekarang = millis();
tSekarang = ardu_time;
//Serial.println(tSekarang); // => just for test
dT = tSekarang - tSebelum;
//Serial.println(dT);
/* mengecek apakah "waktu cek" sudah terpenuhi
dan jika belum, maka menghitung yawSekarang */
if(dT < WAKTU_CEK){
yawSekarang=ypr[0];
//Serial.print("yaw sekarang"); // => menampilkan yaw
realtime
//Serial.println(yawSekarang);
}
/* Jika "waktu cek" sudah melebihi waktu minimal */
else{
yawSebelum = yawSekarang;
dYaw = abs(yawSekarang - yawSebelum);
//Serial.print("yaw Sebelum"); // => menampilkan nilai yaw
sebelum
//Serial.println(yawSebelum);
//Serial.print("dYaw"); // => menampilkan nilai delta untuk
mengecek error yang terjadi
```

```
//Serial.print(dYaw);
/* mengecek apakah "dYaw" sudah lebih kecil dari
"ERROR_MINIMAL" */
if(dYaw < ERROR_MINIMAL){
counterKalibrasi++ ; // => Kalibrasi +1 sampai nilainya
menjadi 2
//Serial.println(counterKalibrasi);
}
/* jika nilai error "dYaw" masilh lebih besar dari nilali
"ERROR_MINIMAL" */
else counterKalibrasi = 0;
/* Jika sudah lolos dari fungsi "(dYaw < ERROR_MINIMAL)" */
if (counterKalibrasi>2){
statusKalibrasi = 1;
offsetSudut = yawSekarang; // => kalibrasi beres
// get_time(); // => !! when this use, for the second time
mpu will calibrate again
}
 tSebelum = tSekarang; // => fungsinya adakah rekayasa untuk
mengulang nilai millis jika terjadi gagal dalam kalibrasi
}
}
else{
digitalWrite(led, HIGH);
//Serial.print("xy"); // => untuk stm32f4
// => define as universal
//Serial.println(ypr[0] * 180/M_PI); //+180,0);
//yaw2 = ypr[0] * 180/M_PI;
// => after calibration (only work alone)
yaw2 = ((ypr[0] - offsetSudut) * 180/M_PI); // + 180;
if (yaw2 > 180) yaw2 -= 360;    // untuk -179
else if (yaw2 < -180)yaw2 += 360;   // untuk 179
//yaw2 = (ypr[0] * 180/M_PI) + 180;
//dataYaw = (ypr[0]-offsetSudut) * 180/M_PI; // karena jika
dalam dua bagian, hasilnya tidak presisi
} // dikarenakan ada offset hasil kalibrasi
#endif
}
}
```

C. Program Master STM32F4 Discovery

➢ /* Fungsi Program Utama */

```
void ta_kompas_bagoes()
{
   switch (case_mpu)
   {
      case 0:
         hitung_langkah = 0; // new
         lcd_clear();
         case_mpu = 1;
         break;
      case 1:
         sendmpu2 = '3'; // SWITCH AKSES MPU6050 CONTINUED
         sprintf(tampil, "%c", sendmpu2);
         USART_Puts(USART3,tampil);
         case_mpu = 2;
         break;
      case 2:
         navigasilurus();
         break;
   }
}
```

➢ /* Fungsi berjalan lurus dengan PID */

```
void navigasilurus()
{
   kecepatan = 50; // => Kecepatan Trayektori
   langkah = 3;
   tinggi = 1;
   PIDmpu();
   gerak(0, langkah, sudutBelok, tinggi, kecepatan);
/* Tampil Error */
   sprintf(tampil, "%3i\n", errormpu);
/* Tampil Langkah dan Error */
   //sprintf(tampil, "%3i %3i\n", hitung_langkah, errormpu);
}
```

➢ /* Fungsi Kontrol PID */

```
void PIDmpu()
{
   // dalam kasus ini void getspmpu harus sudah di panggil
   errormpu = (spmpu - valmpu6050);
   // pembagi antara error min dan plus dari 360 derajat
   if (errormpu > 180) errormpu -= 360; // untuk -179
   else if (errormpu < -180)errormpu += 360; // untuk 179
   pidmpu = - (kpmpu*errormpu + kimpu*(errormpu + errorsmpu)
            + kdmpu*(errormpu + errorsmpu));
   sudutBelok = pidmpu;
```

```
   if (sudutBelok >= 3)sudutBelok = 3; // + belok kiri
   if (sudutBelok <= -3)sudutBelok = -3;// - belok kanan
   lcd_clear(); // independent lcd
   lcd_gotoxy(0,0);
   lcd_puts("UJI PID MPU6050");
   lcd_gotoxy(0,1);
   errorsmpu = errormpu;
}
```

➢ /* Fungsi Mendapatkan SetPoint */

```
void getspmpu2()
{
   switch(case_baca2_mpu)
   {
      case 0:
         sendmpu2 = '1';
         sprintf(tampil, "%c", sendmpu2);
         USART_Puts(USART3,tampil);
         delay_ms(3000);
         case_baca2_mpu = 1;
         break;
      case 1:
         getspmpu(); // jika dalam case ini terjadi loop,
                     // maka case 2, harus dikunci
         delay_ms(10);
         case_baca2_mpu = 2;
         break;
      case 2:
         lcd_clear(); // independent lcd
         lcd_gotoxy(0,0);
         lcd_puts(">> Step 4");
         lcd_gotoxy(0,1);
         lcd_puts("SetPoint MPU6050");
         lcd_gotoxy(0,2);
         sprintf(tampil, ": %3d", spmpu);
         lcd_puts(tampil);
         delay_ms(175);
      break;
   }
}
```

➢ /* Fungsi Invers Kinematik */

```
void inversKinematic (double x, double y, double z)
{
   double theta1;
   double theta2;
   a = sqrt(x*x + y*y);
   c = sqrt((a-l1)*(a-l1) + z*z);
```

```
   alfa0 = radKeDerajat(acos((c*c - l2*l2 - l3*l3)/ (2*l2*l3
           )));
   theta1 = radKeDerajat(acos((l2*l2 + c*c - l3*l3)/(2*l2
           *c)));
   theta2 = radKeDerajat(atan(z/(a-l1)));
   alfa1 = theta1 + theta2;
   alfa2 = radKeDerajat(atan(y/x));
}
```

➢
```
/* Fungsi Trajektori Linear */
void trajectoriLinier(double speed)
{
   static double delta_x[jumlahKaki];
   static double delta_y[jumlahKaki];
   static double delta_z[jumlahKaki];

   static double speed_x[jumlahKaki];
   static double speed_y[jumlahKaki];
   static double speed_z[jumlahKaki];

   static double s[jumlahKaki];
   double ts = TIME_SAMPLING;

   int i;
   for(i=0;i<jumlahKaki;i++)
   {
      delta_x[i]=posisi_tujuan_x[i] - posisi_sekarang_x[i];
      delta_y[i]=posisi_tujuan_y[i] - posisi_sekarang_y[i];
      delta_z[i]=posisi_tujuan_z[i] - posisi_sekarang_z[i];

      s[i] = sqrt(delta_x[i]*delta_x[i] +delta_y[i]*
             delta_y[i] + delta_z[i]*delta_z[i]);

      if(fabs(s[i])>fabs(speed*ts))
      {
         speed_x[i] = speed*(delta_x[i]/s[i]);
         speed_y[i] = speed*(delta_y[i]/s[i]);
         speed_z[i] = speed*(delta_z[i]/s[i]);

         posisi_sekarang_x[i] = speed_x[i]*ts +
                                posisi_sekarang_x[i];
         posisi_sekarang_y[i] = speed_y[i]*ts +
                                posisi_sekarang_y[i];
         posisi_sekarang_z[i] = speed_z[i]*ts +
                                posisi_sekarang_z[i];
      }

      else if(fabs(s[i])<=fabs(speed*ts) && fabs(s[i]>0))
      {
```

```
            posisi_sekarang_x[i] = posisi_tujuan_x[i];
            posisi_sekarang_y[i] = posisi_tujuan_y[i];
            posisi_sekarang_z[i] = posisi_tujuan_z[i];
            statusGerak[i]=0;
         }
         else if(fabs(s[i]==0))statusGerak[i]=0;
            pergeseranPerkaki(i,posisi_sekarang_x[i]-
            koordinat_awal_x[i],posisi_sekarang_y[i]-
            koordinat_awal_y[i],posisi_sekarang_z[i]-
            koordinat_awal_z[i]);
      }
   }
```

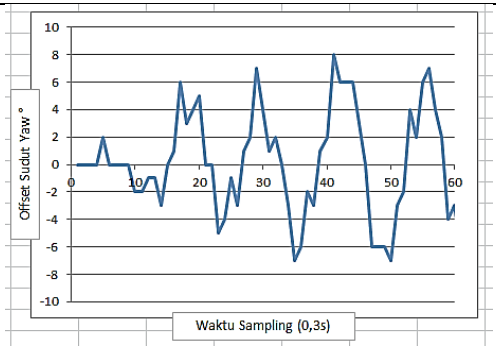
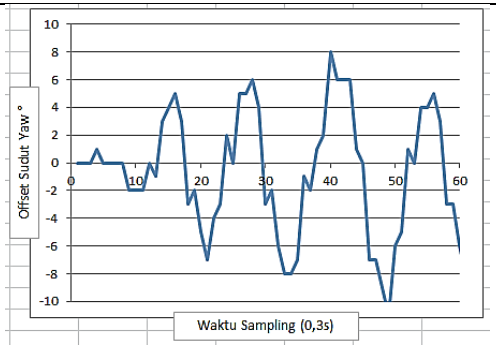
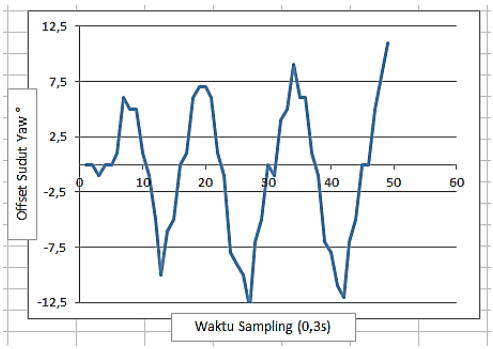
➢ /* Fungsi Transformasi Geometri */

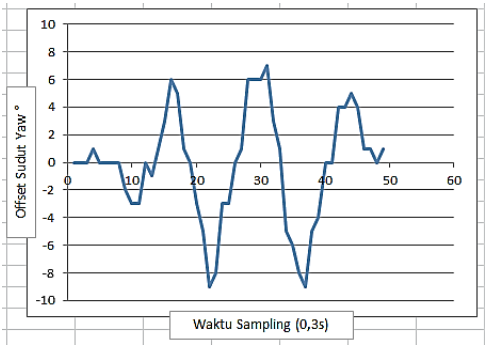
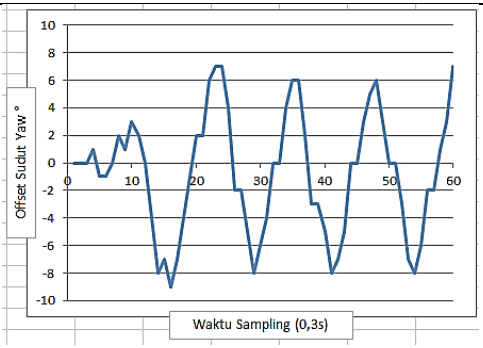
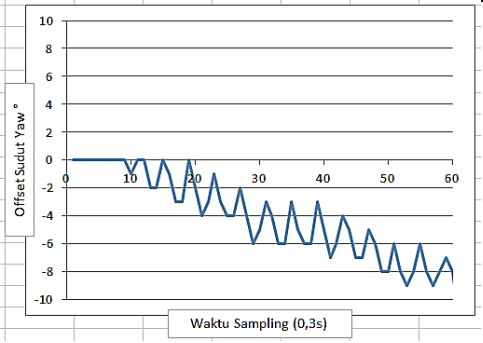
```
void transformasiGerak(double x,double y,double w)
{
       int i;
       w = w*PI/180; // => dijadikan radian
       for(i=0;i<jumlahKaki;i++)
       {
              //transformasi rotasi
              if (i==0)
              {
                     walkX1[i]   =   (koordinat_awal_x[i]-(-
center_a))  *  cos(w)  -  (koordinat_awal_y[i]-(-center_b))  *
sin(w) + (-center_a);
                     walkX2[i]   =   (koordinat_awal_x[i]-(-
center_a))  *  cos(-w)  -  (koordinat_awal_y[i]-(-center_b))  *
sin(-w) + (-center_a);
                     walkY1[i]   =   (koordinat_awal_x[i]-(-
center_a))  *  sin(w)  +  (koordinat_awal_y[i]-(-center_b))  *
cos(w) + (-center_b);
                     walkY2[i]   =   (koordinat_awal_x[i]-(-
center_a))  *  sin(-w)  +  (koordinat_awal_y[i]-(-center_b))  *
cos(-w) + (-center_b);
              }
              else if (i==1)
              {
                     walkX1[i]   =   (koordinat_awal_x[i]-(-
center_a))  *  cos(w)  -  (koordinat_awal_y[i]-(center_b))  *
sin(w) + (-center_a);
                     walkX2[i]   =   (koordinat_awal_x[i]-(-
center_a))  *  cos(-w)  -  (koordinat_awal_y[i]-(center_b))  *
sin(-w) + (-center_a);
                     walkY1[i]   =   (koordinat_awal_x[i]-(-
center_a))  *  sin(w)  +  (koordinat_awal_y[i]-(center_b))  *
cos(w) + (center_b);
```

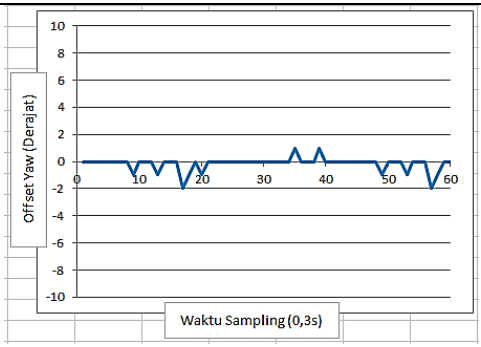
```
                        walkY2[i]   =   (koordinat_awal_x[i]-(-
center_a))  *  sin(-w)  +  (koordinat_awal_y[i]-(center_b))  *
cos(-w) + (center_b);
                }
                else if (i==2)
                {
                        walkX1[i]    =    (koordinat_awal_x[i]-
(center_a))  *  cos(w)  -  (koordinat_awal_y[i]-(center_b))  *
sin(w) + (center_a);
                        walkX2[i]    =    (koordinat_awal_x[i]-
(center_a))  *  cos(-w)  -  (koordinat_awal_y[i]-(center_b))  *
sin(-w) + (center_a);
                        walkY1[i]    =    (koordinat_awal_x[i]-
(center_a))  *  sin(w)  +  (koordinat_awal_y[i]-(center_b))  *
cos(w) + (center_b);
                        walkY2[i]    =    (koordinat_awal_x[i]-
(center_a))  *  sin(-w)  +  (koordinat_awal_y[i]-(center_b))  *
cos(-w) + (center_b);
                }
                else if (i==3)
                {
                        walkX1[i]    =    (koordinat_awal_x[i]-
(center_a))  *  cos(w)  -  (koordinat_awal_y[i]-(-center_b))  *
sin(w) + (center_a);
                        walkX2[i]    =    (koordinat_awal_x[i]-
(center_a))  *  cos(-w)  -  (koordinat_awal_y[i]-(-center_b))  *
sin(-w) + (center_a);
                        walkY1[i]    =    (koordinat_awal_x[i]-
(center_a))  *  sin(w)  +  (koordinat_awal_y[i]-(-center_b))  *
cos(w) + (-center_b);
                        walkY2[i]    =    (koordinat_awal_x[i]-
(center_a))  *  sin(-w)  +  (koordinat_awal_y[i]-(-center_b))  *
cos(-w) + (-center_b);
                }

        // transformasi linear
        walkX1[i] +=x;
        walkX2[i] -=x;
        walkY1[i] +=y;
        walkY2[i] -=y;
        //jadikan pergeseran
        walkX1[i] -=koordinat_awal_x[i];
        walkX2[i] -=koordinat_awal_x[i];
        walkY1[i] -=koordinat_awal_y[i];
        walkY2[i] -=koordinat_awal_y[i];
        }
}
```

D. Proses Manual Tuning PID Pada Robot dengan Kecepatan Trayektory 50 cm/s Pada Jarak 2,5 meter

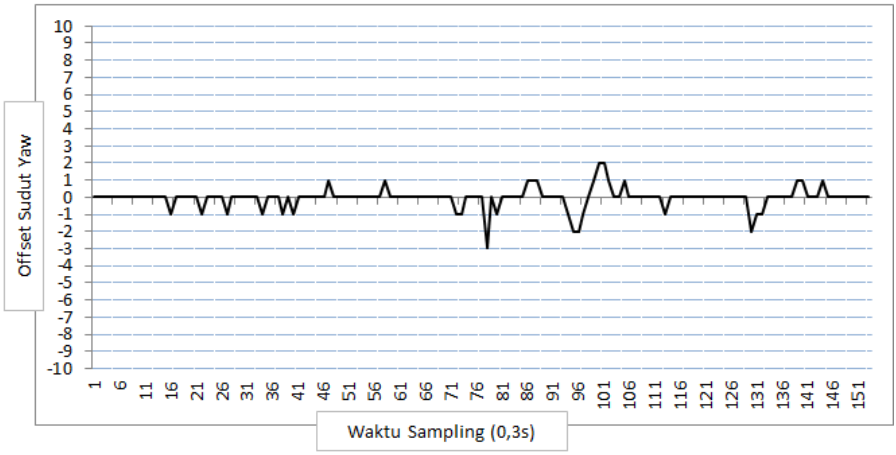
KP : 10 KI : 0 KD : 0	
KP : 10 KI : 0 KD : 5	
KP : 20 KI : 1.15 KD : 8.5	

KP : 20 KI : 0 KD : 10	
KP : 20 KI : 0 KD : 8.5	
KP : 0,5 KI : 0 KD : 0	

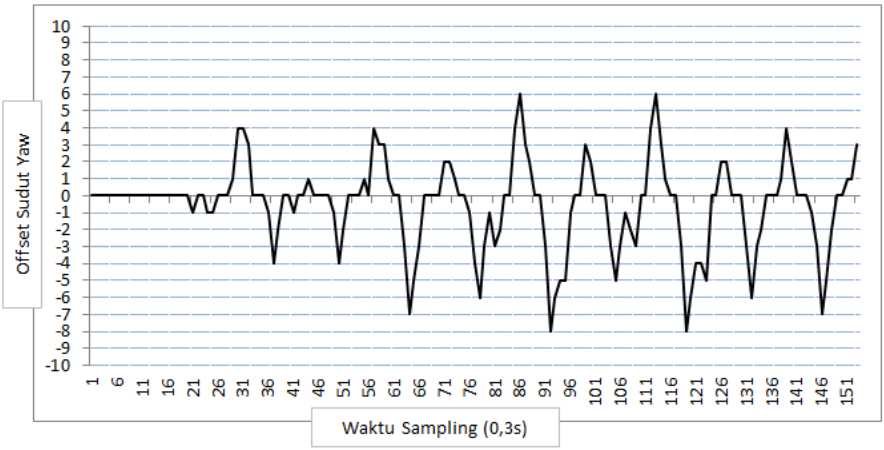
KP : 1,5 KI : 0 KD : 0,75	

E. Pengujian Gerak Robot Berjalan Lurus dengan Kontrol PID dengan Kp 20, Ki 0, dan Kd 8.5 Pada Variabel Kecepatan 30 cm/s sampai 70 cm/s

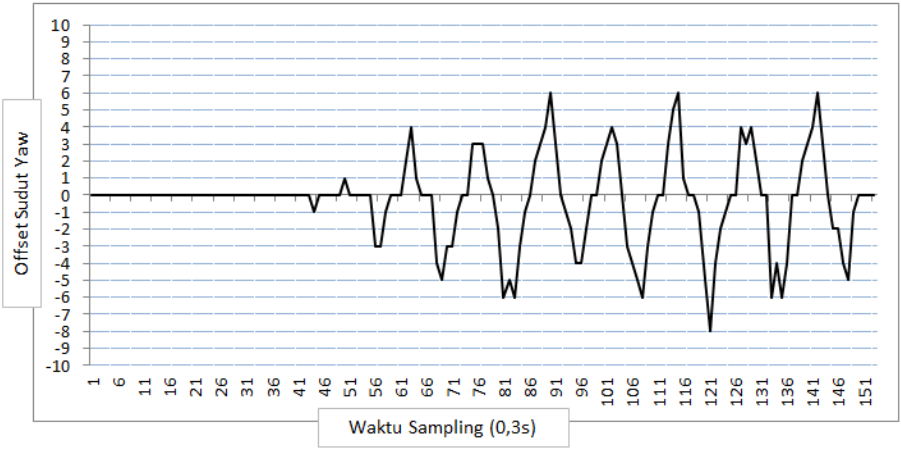
1. Kecepatan Trayektory 30 cm/s



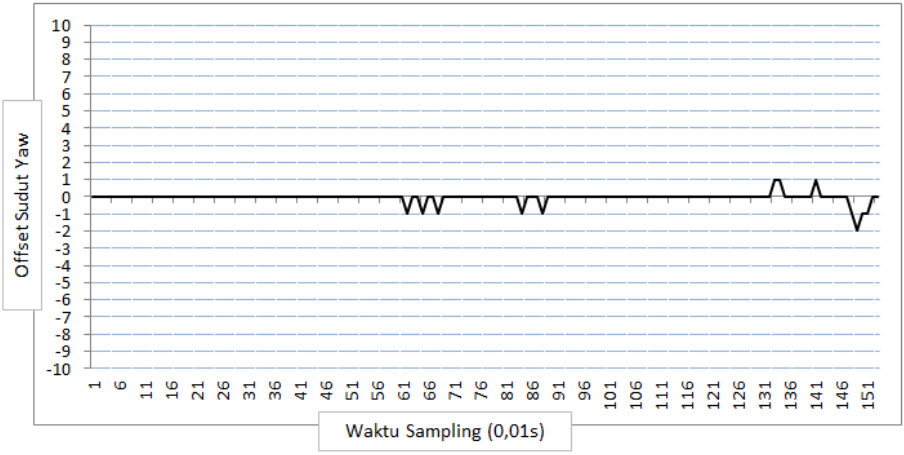
2. Kecepatan Trayektory 40 cm/s



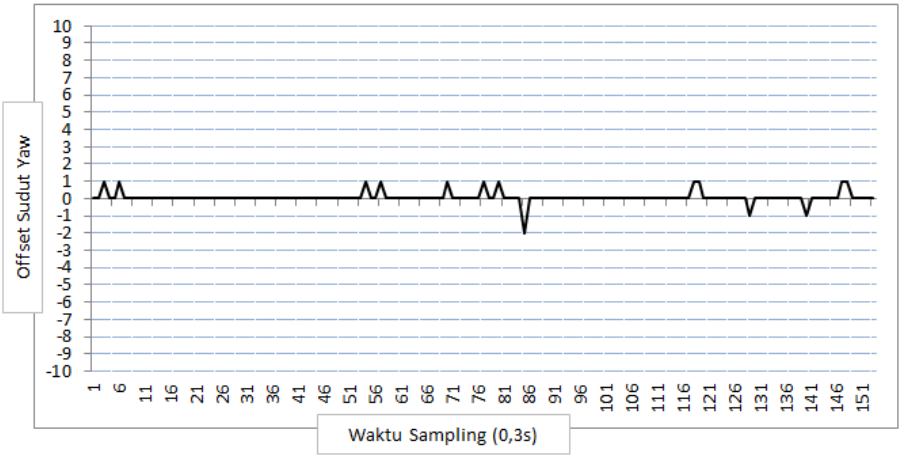
3. Kecepatan Trayektory 50 cm/s



4. Kecepatan Trayektory 60 cm/s



5. Kecepatan Trayektory 70 cm/s



F. Perbandingan Offset Sudut Yaw MPU5060 Terhadap Susdut Yaw Sebenarnya Pada Pengujian Gerak Berjalan Lurus Tanpa Kontrol PID dengan Jarak 4,5 meter

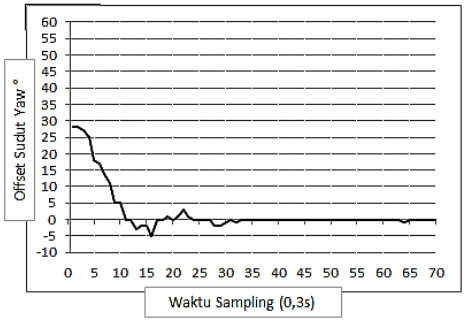
Kecepatan Trajectory	Offset Yaw Pada Sensor IMU	Offset Yaw Sebenarnya	Kesalahan Pembacaan Sensor IMU
30 cm/s	16,2019º	12,832º	3,370 º
40 cm/s	24,3908º	13,134º	11,257 º
50 cm/s	14,1549º	8,531º	5,624 º
60 cm/s	17,5418º	10,697º	6,845 º
70 cm/s	9,7816º	3,497º	6,285 º

G. Perbandingan Offset Sudut Yaw MPU5060 Terhadap Sudut Yaw Sebenarnya Pada Pengujian Gerak Berjalan Lurus Dengan Kontrol PID (Kp = 1.5, Ki = 0.001, Kd = 0.75) dengan Jarak 4,5 meter

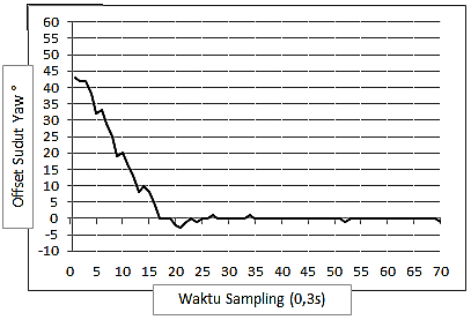
Kecepatan Trajectory	Offset Yaw Pada Sensor IMU	Offset Yaw Sebenarnya	Kesalahan Pembacaan Sensor IMU
30 cm/s	0º	0,255º	0,255º
40 cm/s	0º	0,637º	0,637º
50 cm/s	0º	0º	0º
60 cm/s	0º	0,255º	0,255º
70 cm/s	0º	0,382º	0,382º

H. Pengujian Respon Sistem Kontrol PID dengan Kp 1.5, Ki 0, dan Kd 0.75 Pada Kecepatan Trayektory 50 cm/s dengan Pemberian Kesalahan Posisi Sudut Yaw Saat Start

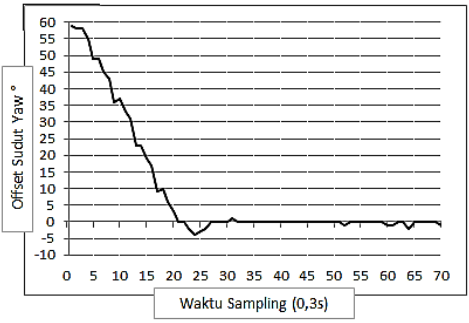
1. Offset Start 30 derajat



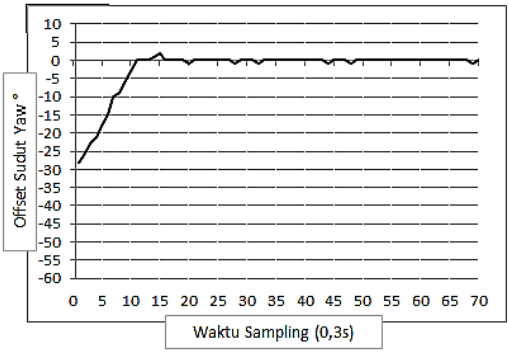
2. Offset Start 45 derajat



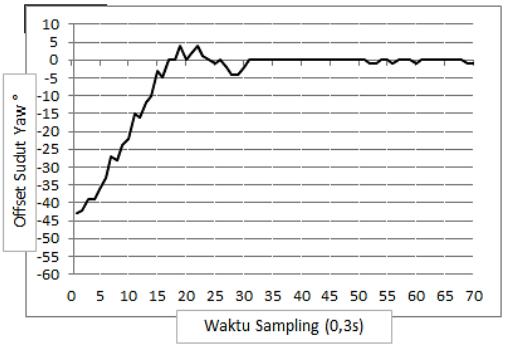
3. Offset Start 60 derajat



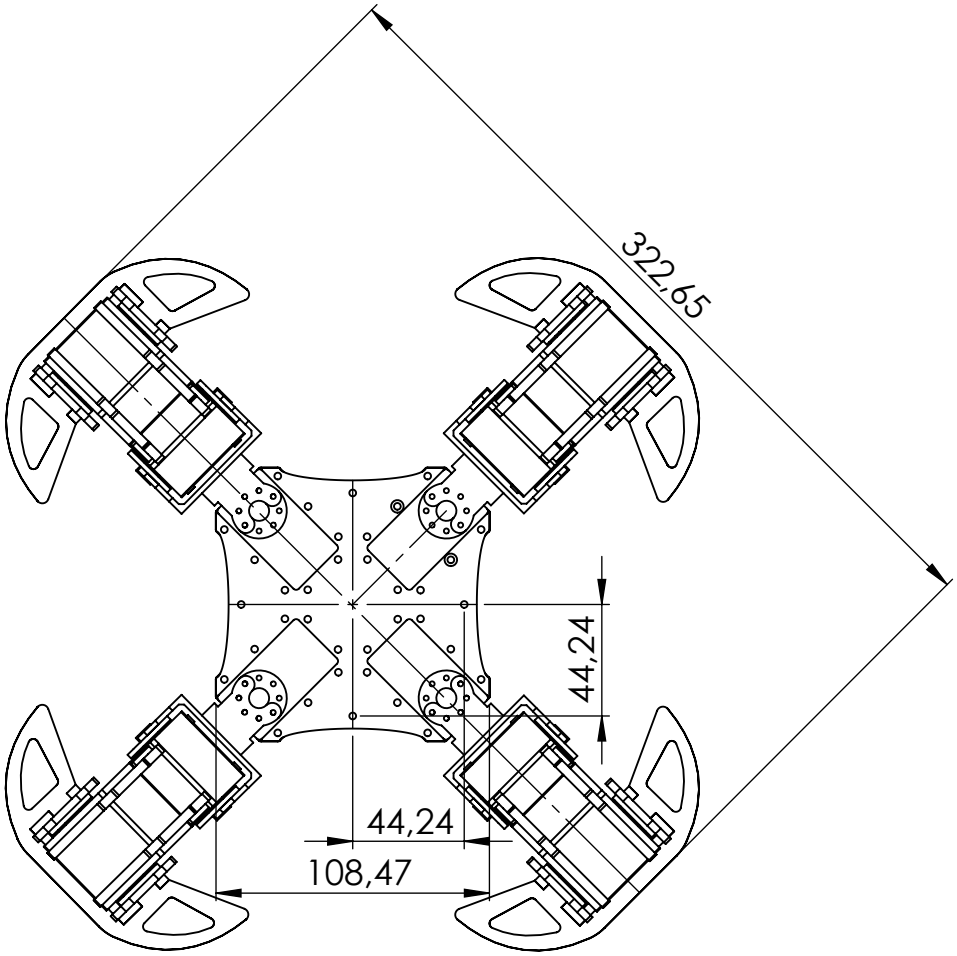
4. Offset Start -30 derajat

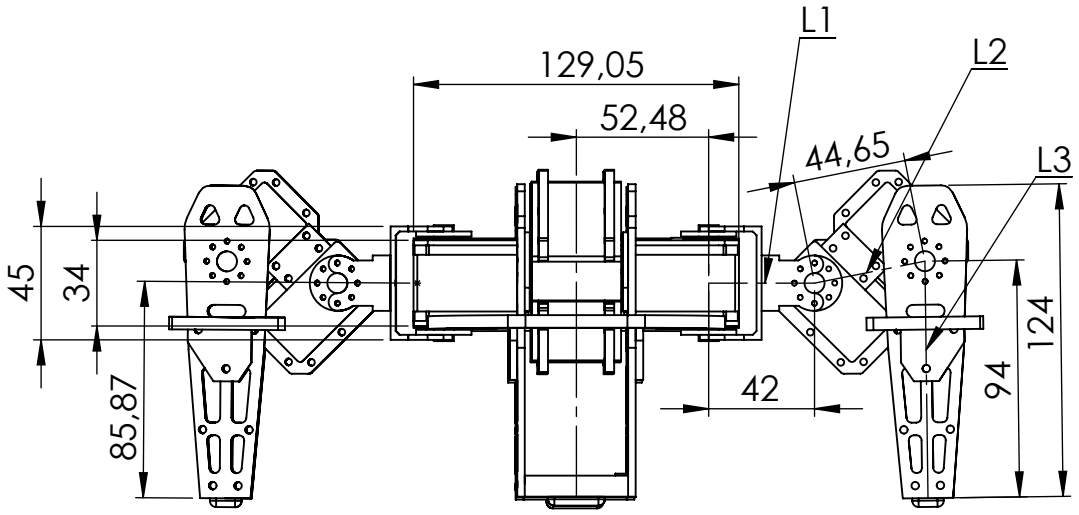


5. Offset -45 derajat

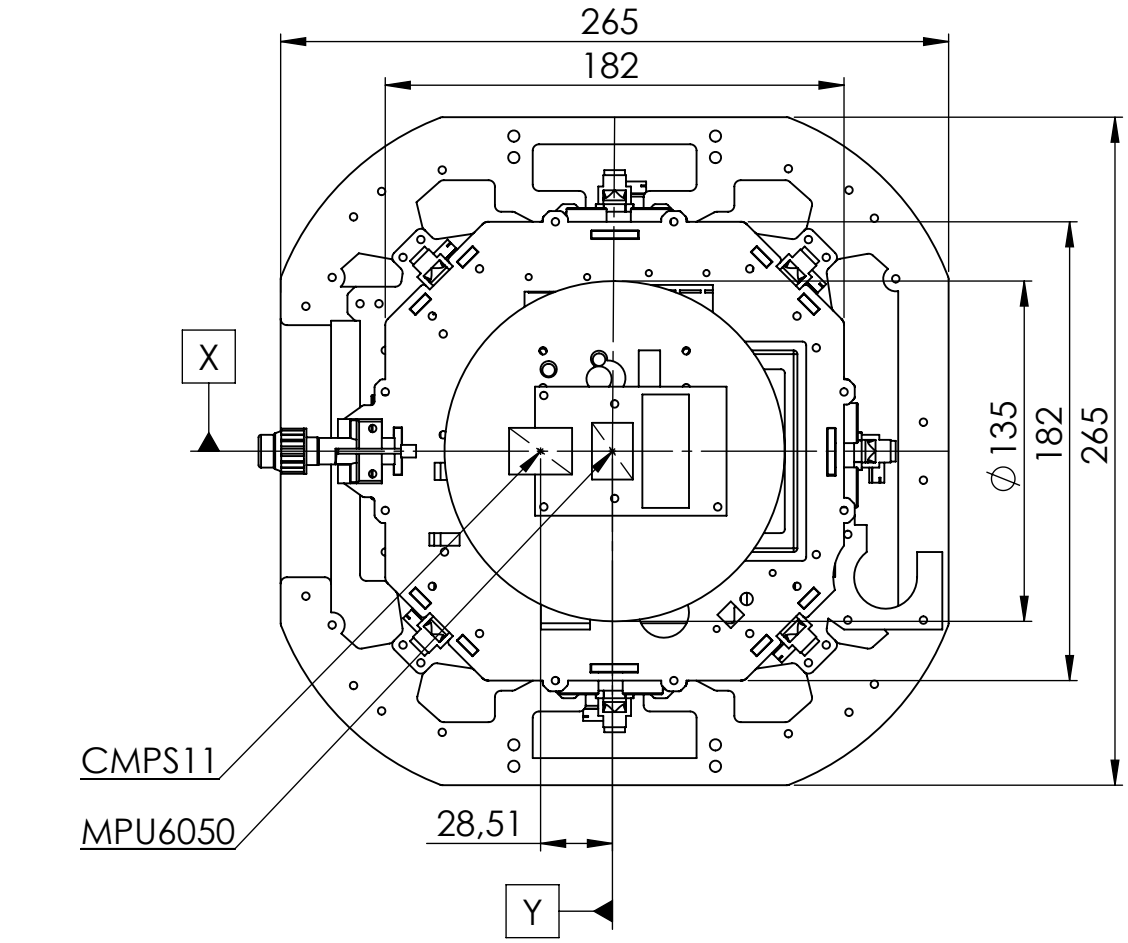


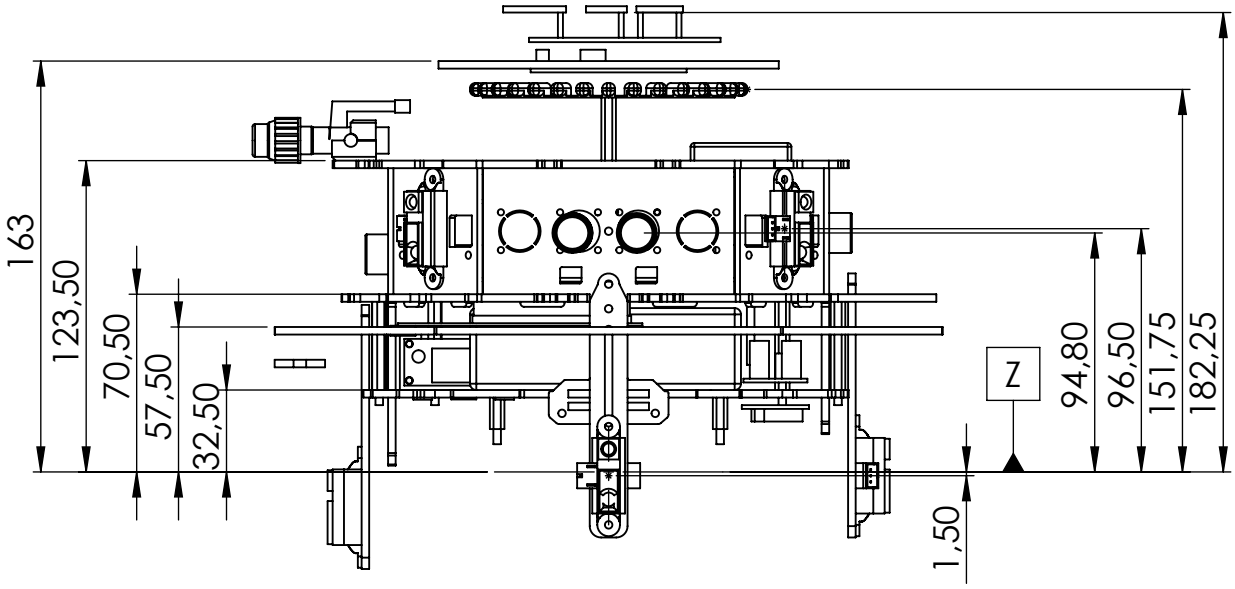
Halaman ini sengaja dikosongkan



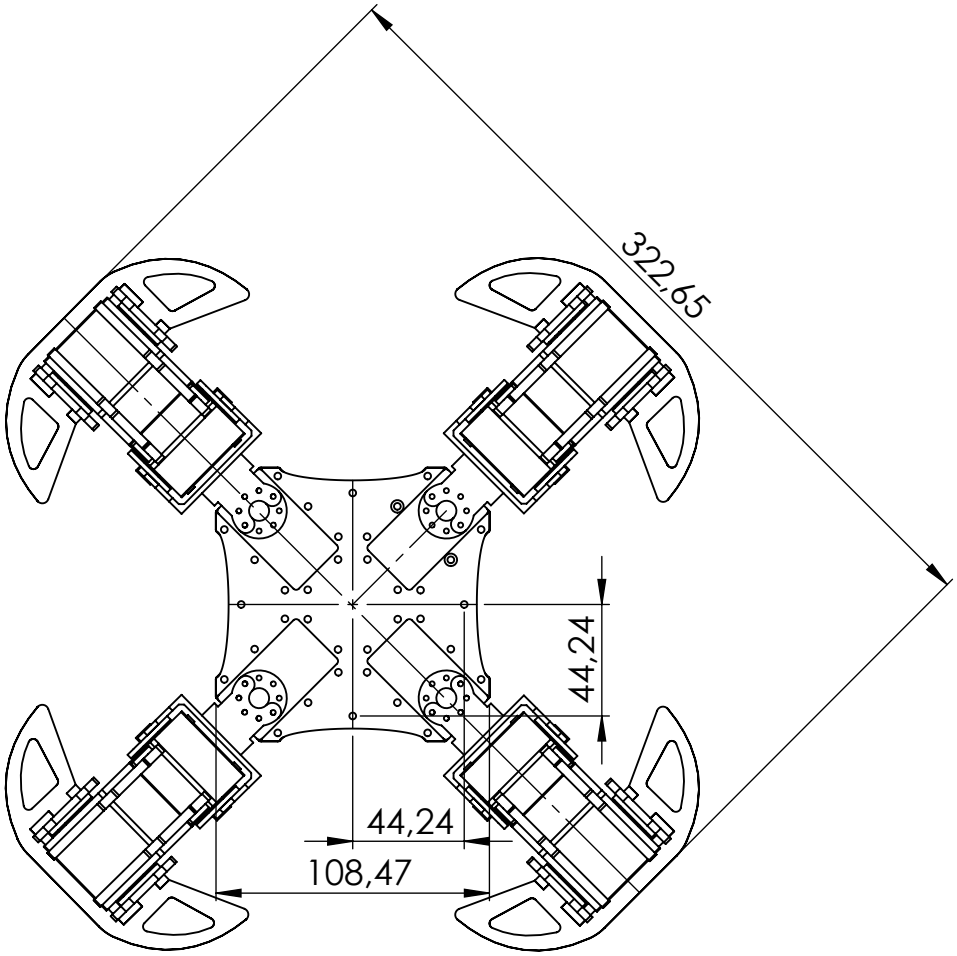


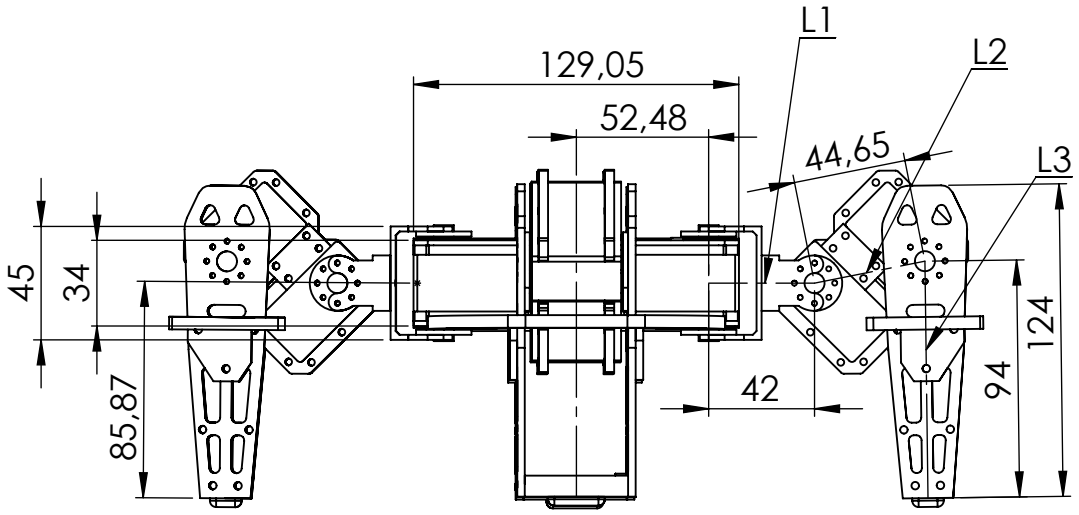
			Aluminium		
NO	JUMLAH	NAMA	BAHAN	UKURAN	
		SKALA : 1 : 3	DIGAMBAR : TIM ABINARA-1		KETERANGAN :
		UKURAN : mm	NRP : -		
		TANGGAL : 12/07/2018	DILIHAT : -		
-		FRAME KAKI ROBOT		NO.	A4



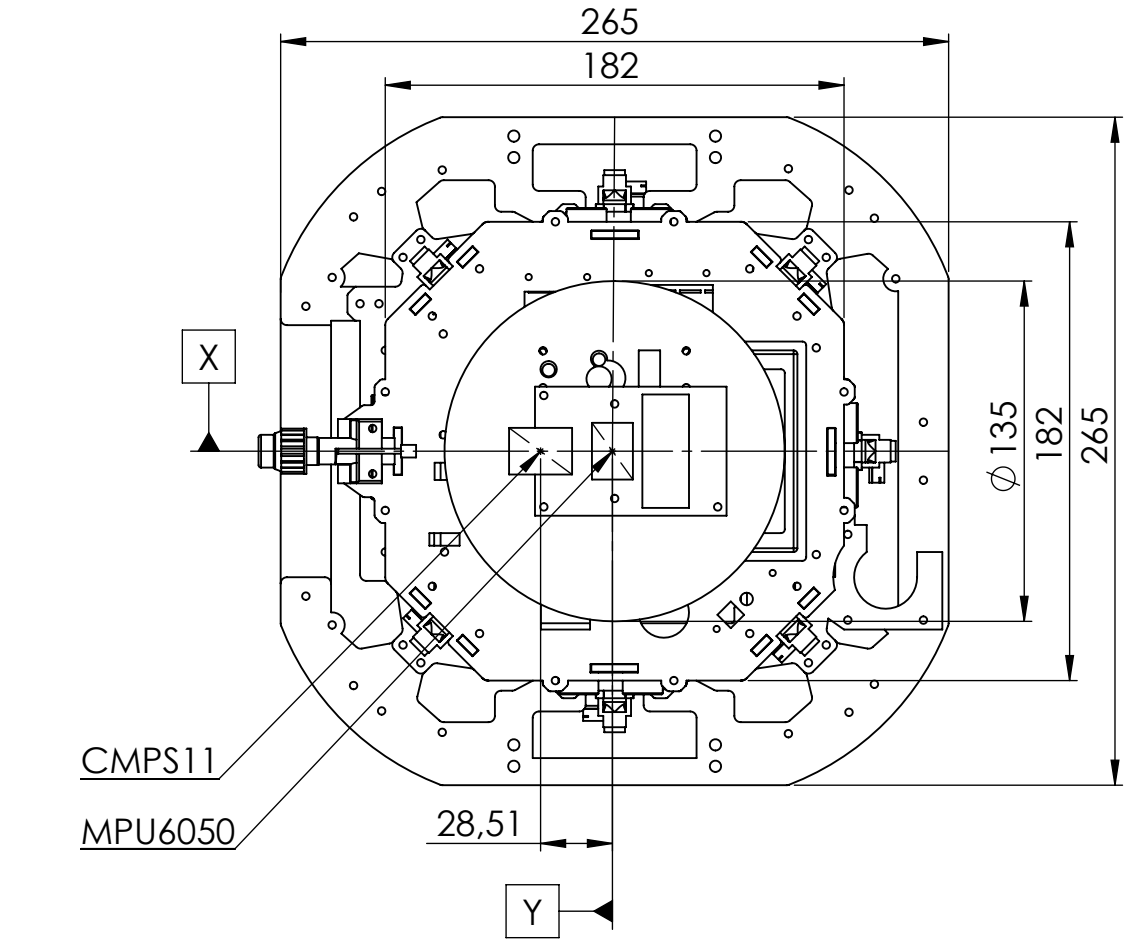


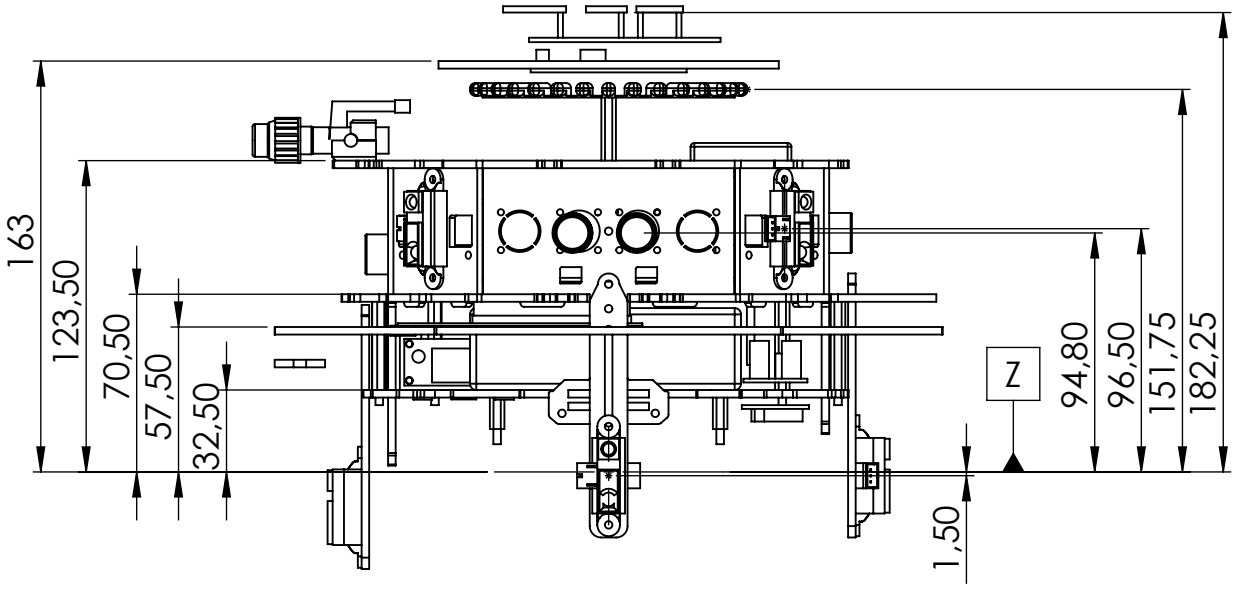
			AKRILIK		
NO	JUMLAH	NAMA	BAHAN	UKURAN	
		SKALA : 1 : 3	DIGAMBAR : TIM ABINARA-1		KETERANGAN :
		UKURAN : mm	NRP : -		
		TANGGAL : 12/07/2018	DILIHAT : -		
-		DESAIN BODY ROBOT		NO.	A4





			Aluminium		
NO	JUMLAH	NAMA	BAHAN	UKURAN	
		SKALA : 1 : 3	DIGAMBAR : TIM ABINARA-1		KETERANGAN :
		UKURAN : mm	NRP : -		
		TANGGAL : 12/07/2018	DILIHAT : -		
-		FRAME KAKI ROBOT		NO.	A4





			AKRILIK		
NO	JUMLAH	NAMA	BAHAN	UKURAN	
		SKALA : 1 : 3	DIGAMBAR : TIM ABINARA-1		KETERANGAN :
		UKURAN : mm	NRP : -		
		TANGGAL : 12/07/2018	DILIHAT : -		
-		DESAIN BODY ROBOT		NO.	A4

BIODATA PENULIS

Penulis, Bagoes Prawira Natakusuma lahir pada 26 April 1997 di Kabupaten Magetan, Jawa Timur, penulis merupakan anak pertama dari dua bersaudara. Penulis menempuh pendidikan dasar di SDIT Imambukhori Jatinangor Kab.Sumedang (2003 - 2009), pendidikan menengah di SMP Al-Ma'soem Cileunyi, Kab. Bandung (2009 - 2011), SMP Wachid Hasyim 7 (2011 - 2012) di Surabaya, dan pendidikan menengah atas di SMA Trimurti Surabaya (2012-2015). Saat ini penulis sedang menyelesaikan studi diploma III di Institut Teknologi Sepuluh Nopember (ITS) Suarabaya pada departemen Teknik Mesin Industri, Fakultas Vokasi dengan bidang studi manufaktur. Sejak pendidikan dasar penulis sudah mulai menggemari keahlian di bidang mekatronika, khususnya robotika dan pernah mengikuti perlombaan seperti pembuatan teknologi sederhana, dilanjut pada pendidikan menengah mulai mengikuti perlombaan robotika tingkat jawa barat, dan pada saat kuliah penulis melanjutkan keahliannya di bidang robotika dengan mengikuti unit kegiatan mahasiswa robotika dan menjadi perwakilan tim robot ITS pada divisi Kontes Robot Pemadam Api Indonesia (KRPAI) kategori berkaki pada tahun 2016 - 2018 (dua periode). Penulis juga sangat tertarik dengan kegiatan riset mengenai mekatronika, sistem kontrol, robotika, dan mekanika kuantum.

Email :
bagoespn@windowslive.com

Halaman ini sengaja dikosongkan