TUGAS AKHIR – TE 141599

RANCANG BANGUN SENSOR PEMINDAI GERAK TANGAN MANUSIA MENGGUNAKAN SENSOR AKSELEROMETER DAN SENSOR GIROSKOP UNTUK MENGENDALIKAN LENGAN ROBOT

MOCHAMAD FAJAR RINALDI UTOMO
NRP 2212 100 198

Dosen Pembimbing
Ronny Mardiyanto, ST., MT., Ph.D.
Ir. Tasripan, MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

TUGAS AKHIR – TE 141599

RANCANG BANGUN SENSOR PEMINDAI GERAK TANGAN MANUSIA MENGGUNAKAN SENSOR AKSELEROMETER DAN SENSOR GIROSKOP UNTUK MENGENDALIKAN LENGAN ROBOT

MOCHAMAD FAJAR RINALDI UTOMO
NRP 2212 100 198

Dosen Pembimbing
Ronny Mardiyanto, ST., MT., Ph.D.
Ir. Tasripan, M.T.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

Halaman ini sengaja dikosongkan

FINAL PROJECT – TE141599

DESIGN OF HUMAN HAND MOTION USING ACCELEROMETER AND GYROSCOPE TO CONTROL ARM ROBOT

MOCHAMAD FAJAR RINALDI UTOMO
NRP 2212 100 198

Advisor
Ronny Mardiyanto, ST., MT., Ph.D.
Ir. Tasripan, M.T.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Rancang bangun sensor pemindai gerak tangan manusia menggunakan sensor akselerometer dan sensor giroskop untuk mengendalikan lengan robot" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya,Juni 2016

Mochamad Fajar Rinaldi Utomo
NRP.2212 100 198

Halaman ini sengaja dikosongkan

RANCANG BANGUN SENSOR PEMINDAI GERAK TANGAN MANUSIA MENGGUNAKAN SENSOR AKSELEROMETER DAN SENSOR GIROSKOP UNTUK MENGENDALIKAN LENGAN ROBOT

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan Untuk
Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopemeber

Menyetujui

Dosen Pembimbing I,	Dosen Pembimbing II,
Rosuy Mardiyanto, S.T., M.T., Ph.D.	Ir. Tasripan, M.T.
NIP. 198101182003121003	NIP. 196204181990031004

SURABAYA

JANUARI, 2017

Halaman ini sengaja dikosongkan

RANCANG BANGUN SENSOR PEMINDAI GERAK TANGAN MANUSIA MENGGUNAKAN SENSOR AKSELEROMETER DAN SENSOR GIROSKOP UNTUK MENGENDALIKAN LENGAN ROBOT

Mochamad Fajar Rinaldi Utomo
2212100198

DosenPembimbing I : Ronny Mardiyanto. S.T., M.T., Ph.D.
DosenPembimbing II : Ir. Tasripan, M.T.

ABSTRAK

Sensor pemindai gerakan tangan adalah perangkat yang digunakan untuk mengukur gerak tangan dalam sumbu Cartesian. Robot tangan yang dapat dikendalikan secara jarak jauh sudah dikembangkan dan digunakan untuk membantu dalam misi berbahaya. Robot dengan kendali jarak jauh konvensional biasanya dikendalikan dengan mengatur sudut untuk setiap sendi robot secara individual. Penggunaan remote control konvensional memerlukan operator berpengalaman dan membutuhkan waktu yang relatif besar bagi operator untuk mengontrol lengan.

Dalam tugas akhir ini, sistem untuk mengukur gerakan tangan dikembangkan untuk mendapatkan nilai perpindahan untuk setiap gerakan. Accelerometer, giroskop dan magnetometer digunakan untuk mendapatkan setiap sudut sendi. metode kinematika maju digunakan untuk mendapatkan posisi dan gerakan end effector. Data gerakan yang diperoleh digunakan sebagai informasi untuk mengontrol gerakan lengan dengan metode kinematika balik. Genggaman telapak tangan diukur oleh flex sensor untuk mengontrol gripper pada end effector robot tangan. Lengan robot dikendalikan secara jarak jauh menggunakan koneksi nirkabel hingga jarak 100 meter.

hasil perhitungan kinematika maju menghasilkan kesalahan kurang dari 3%. hasil perhitungan kinematika balik menghasilkan kesalahan hingga 12%. Koneksi nirkabel menambahkan latensi 0,1 detik pada jarak 50 meter dan latensi 0,2 detik pada jarak 100 meter. Tegangan output sensor flex berkorelasi secara linier dengan sudut tekukan jari tangan.

Kata kunci: robot tangan, kendali nirkabel, akselerometer, giroskop, magnetometer, flex, *forward kinematic*, *inverse kinematic*.

Halaman ini sengaja dikosongkan

DESIGN OF HUMAN HAND MOTION USING ACCELEROMETER AND GYROSCOPE TO CONTROL ROBOT ARM

Mochamad Fajar Rinaldi Utomo
2212100198

Advisor : Ronny Mardiyanto. S.T., M.T., Ph.D.
Co-Advisor : Ir. Tasripan, M.T.

ABSTRACT

Hand motion scanning sensor is a device that is used to measure hand motion in the cartesian axes. Remotely controlled arm robot is already developed and used to help in dangerous mission. Conventional remotely controlled robot usually controlled by individually controlling the angle for each robot joints. The use of conventional remote control require experienced operator and require comparatively large time for the operator to control the arm.

In this final project, a system to sense hand movement is developed to obtain displacement value for every movement. Accelerometer, gyroscope and magnetometer is used to obtain each joint angle. Forward kinematic methods is used to obtain end effector position and movement. The obtained movement data is used as source information to control arm movement by inverse kinematics methods. Palm grip sensed by flex sensor is used to control robotic gripper at robotic arm's end effector. The robotic arm is cotrolled remotely using wireless connection up to 100 meters.

Forward kinematic calculation results in error of less than 3%. Inverse kinematics calculation results in error up to 12%. Wireless connection adds 0.1 second latency at 50 meters and 0.2 second latency at 100 meters. Flex sensor voltage output is linearly correlated with fingers bend angle.

***Keyword**: robot arm, wireless control, accelerometer, gyroscope, magnetometer, flex, forward kinematic, inverse kinematic*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Alhamdulillahirobbilaalamin, segala puji syukur kepada Allah SWT, atas segala nikmat, berkah, dan hidayah-Nya yang tak terkira kepada penulis, hingga penulis mampu menyelesaikan Tugas Akhir dengan judul :

RANCANG BANGUN SENSOR PEMINDAI GERAK TANGAN MANUSIA MENGGUNAKAN SENSOR AKSELEROMETER DAN SENSOR GIROSKOP UNTUK MENGENDALIKAN LENGAN ROBOT

Tujuan utama tugas akhir ini adalah sebagai salah satu persyaratan untuk menyelesaikan jenjang pendidikan pada Bidang Studi Elektronika Teknik Elektro Institut Teknologi Sepuluh Nopember.

Atas selesainya penyusunan tugas akhir ini, penulis ingin mengucapakan terima kasih kepada :

1. Kedua orang tua yang tiada hentinya memberikan doa restu, dukungan moril dan materil.
2. Bapak Ronny Mardiyanto, ST., MT., Ph.D. selaku dosen pembimbing 1 atas gagasan topik tugas akhir serta bimbingan dan arahan untuk penulis selama mengerjakan tugas akhir ini.
3. Bapak Ir. Tasripan, MT. selaku dosen pembimbing 2 atas bimbingan dan arahan untuk penulis selama mengerjakan tugas akhir ini. Elektro ITS Surabaya.
4. Seluruh dosen bidang studi elektronika jurusan teknik elektro ITS.
5. Teman-teman bidang studi elektronika yang tidak dapat penulis sebutkan satu-persatu yang telah memberikan penulis semangat dan canda tawa selama menjalani perkuliahan di teknik elektro ITS.

Penulis berharap para pembaca Tugas Akhir ini bersedia memberikan kritik, saran, dan masukan agar selanjutnya menambah manfaat Tugas Akhir.

Semoga laporan Tugas Akhir ini dapat bermanfaat dan bisa dijadikan referensi bagi Tugas Akhir selanjutnya.

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

TABLE OF CONTENTS

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada bulan maret 2011 terjadi peristiwa terorisme berupa bom berbentuk buku di Utan Kayu yang sempat meledak di tangan polisi ketika hendak menjinakkan bom tersebut. Hal ini disebabkan kecerobohan terhadap benda yang berpotensi membahayakan keselamatan manusia. Saat ini, telah banyak realisasi pemanfaatan robot manual untuk membantu kerja manusia. Salah satunya adalah robot dengan kendali *remote control* konvensional untuk mendeteksi serta menjinakkan bom.

TNI dan POLRI saat ini mengembangkan robot manual untuk menyelesaikan misi-misi khusus yang tidak dapat dilakakukan karena dapat mengancam keselamatan nyawa. Sebagai contoh robot manual kendali jarak jauh menggunakan *remote control* yang digunakan oleh POLRI adalah Morolipi V.1 dan Morolipi V.2 yang dikembangkan oleh Lembaga Ilmu Pengetahuan Indonesia (LIPI) [1]. Robot ini bertugas untuk menjinakkan bom dengan cara memotong kabel atau memindahkannya ke lokasi aman.

Untuk mengendalikan lengan pada robot Morolipi V.1 dan Morolipi V.2 adalah dengan mengatur sudut tiap-tiap *joint*. Padahal, lengan robot tersebut memiliki 5 *joint*, maka untuk menggerakkan menuju posisi tertentu akan menghabiskan waktu serta membutuhkan ketelitian yang lebih karena harus memutar setiap *joint* yang dibutuhkan.

Pengendalian menggunakan *remote control* konvensional memerlukan pelatihan dan pengalaman bagi pengguna. Perbedaan bentuk dan tombol pada perangkat *remote control* konvensional yang berbeda mengakibatkan pengguna memerlukan latihan yang berbeda. Hal ini mengakibatkan kurangnya nilai guna dari *remote control* konvensional bagi pemula. Pentingnya latihan dan pengalaman diperlukan untuk menghindari hal-hal yang tidak diinginkan seperti salah kendali arah dan salah menekan tombol yang dapat berdampak mengganggu sistem yang dikendalikan bahkan berdampak timbulnya bahaya.

Melihat dampak yang disebabkan berpengaruh pada keselamatan manusia, diperlukan pengembangan *remote control*

yang lebih bersahabat dengan pengguna. Untuk itu diperlukan sebuah pengendali baru yang dapat menggantikan *remote control* konvensional.

Sensor akselerometer dan sensor giroskop dibutuhkan untuk dapat memindai perubahan posisi pada lengan manusia. Dengan mengetahui nilai-nilai sudut pada masing-masing *joint* yang diamati, kita dapat mengetahui posisi perubahan ketika lengan terjadi pergerakan. Data ini dapat kita gunakan untuk mengsinkronisasi gerakan lengan robot agar sesuai dengan lengan penggunanya.

Dalam penelitian ini, rancangan yang akan dibuat antara lain alat pemindai gerak yang dapat dipakai manusia dengan integrasi beberapa sensor untuk memindai posisi lengan pengguna. Dengan sistem ini, pengguna akan meringkas waktu kerja serta mengurangi kesalahan yang dilakukan menggunakan *remote control*.

1.2 Perumusan Masalah

Berdasarkan latar belakang di atas, dapat dirumuskan beberapa masalah, antara lain :

1. Bagaimana mengintegerasikan sensor akselerometer, sensor giroskop dan sensor magnetometer?
2. Bagaimana sistem mekanik alat yang akan dipasang di lengan manusia?
3. Bagaimana cara mengolah data sudut-sudut pergerakan manusia menjadi vektor posisi lengan manusia?
4. Bagaimana mekanisme gerak lengan robot yang akan dibuat?
5. Bagaimana mengsinkronisasi data vektor lengan manusia ke pergerakan lengan robot?
6. Bagaimana menggerakkan bagian *gripper* dari lengan robot?

1.3 Tujuan Penelitian

Penelitian pada tugas akhir ini bertujuan sebagai berikut :

1. Sistem integrasi sensor akselerometer, sensor giroskop, dan sensor magnetometer untuk menghasilkan sudut di masing-masing *joint* lengan manusia.
2. Alat pemindai gerak yang cocok untuk dipasang di lengan manusia.
3. Implementasi metode *forward kinematic* untuk mengubah nilai-nilai sudut pada masing2 joint menjadi vektor posisi lengan manusia.

4. Implementasi lengan robot 3 dof dengan gripper.
5. Implementasi metode *inverse kinematic* untuk mengubah nilai posisi lengan manusia menjadi sudut-sudut lengan robot.
6. Implementasi *flex sensor* dengan kontrol proporsional.

1.4 Batasan Masalah

Batasan masalah dari tugas akhir ini adalah sebagai berikut:

1. Alat pemindai gerak yang dibuat hanya pada bagian tangan kanan manusia.
2. Sensor giroskop, sensor akselerometer, dan magnetometer yang digunakan masing-masing 3 buah.

1.5 Metodologi Penelitian

Dalam penyelesaian tugas akhir ini digunakan metodologi sebagai berikut:

1. Studi literatur

Pada tahap ini dilakukan pengumpulan dasar teori yang menunjang dalam penulisan tugas akhir. Dasar teori ini dapat diambil dari buku-buku, jurnal, *proceeding*, dan artikel-artikel di internet. Dengan adanya studi literatur, penelitian dapat dilakukan berdasarkan teori-teori yang telah ada sebelumnya.

2. Perancangan sistem

Setelah mempelajari literatur yang ada, selanjutnya akan dilakukan perancangan sistem. Perancangan sistem terbagi sebagai berikut:

a. Perancangan Perangkat Keras

Sistem perangkat keras yang akan dikerjakan adalah pembuatan mekanisme peletakan alat pemindai gerak dengan memperhatikan kenyamanan pengguna. Alat ini akan dibuat dengan *3D printer* dengan bantuan *software autocad*.

b. Perancangan perangkat lunak

Perancangan perangkat lunak meliputi proses pembacaan sensor akselerometer, sensor giroskop, sensor kompas, dan *flex sensor*. Realisasi program *forward kinematic* dan *inverse kinematic.* Algoritma program untuk mengsinkronisasi gerak lengan manusia dengan lengan robot. Algoritma menggerakkan gripper dengan kontrol proporsional.

3. Pengujian sistem

Pengujian alat ini dilakukan untuk menetukan keandalan dari sistem yang telah dirancang. Pengujian dilakukan untuk melihat apakah *software* dan *hardware* dapat berkerja dengan baik.

Pengujian dilakukan dalam beberapa tahap. Pertama adalah pengujian sensor-sensor yang digunakan. Kedua adalah pengujian algoritma *forward kinematic* sehingga menghasilkan posisi lengan manusia. Ketiga adalah pengujian sensor flex untuk menggerakkan *gripper*. Keempat adalah pengujian inverse kinematic pada robot. Yang terakhir adalah pengujian sistem alat pemindai gerak lengan untuk menggerakkan lengan robot.

4. Analisa

Analisa dilakukan terhadap hasil dari pengujian sehingga dapat ditentukan karakteristik dari *software* dan *hardware* yang telah dibuat. Apabila karakteristik dari *software* dan *hardware* pada sistem belum belum sesuai, maka perlu dilakukan perancangan ulang sistem dan diuji kembali.

5. Penyususan laporan tugas akhir

Tahap penulisan laporan tugas akhir adalah tahapan terakhir dari proses pengerjaan tugas akhir ini. Laporan tugas akhir berisi seluruh hal yang berikatan dengan tugas akhir yang telah dikerjakan yaitu meliputi pendahuluan, teori penunjang, perancangan sistem, pengujian, dan penutup.

1.6 Sistematika Penulisan

Laporan tugas akhir ini terdiri dari Lima Bab dengan sistematika penulisan sebagai berikut:

Bab 1 :PENDAHULUAN

Bab ini meliputi latar belakang, perumusan masalah, tujuan penelitian, batasan masalah, metodologi, sistematika penulisan, dan relevansi.

Bab 2 :DASAR TEORI

Bab ini menjelaskan tentang dasar-dasar teori yang dibutuhkan dalam pengerjaan tugas akhir ini, yang meliputi teori kinematika robot, motor servo, sensor IMU, AHRS.

Bab 3: PERANCANGAN SISTEM

Bab ini menjelaskan tentang perencanaan,sistem elektrik, mekanik, serta perangkat lunak. Bab ini juga berisi menjelaskan tentang prosedur pengujian yang dilakukan dalam penelitian.

Bab 4 :PENGUKURAN DAN ANALISIS SISTEM

Bab ini menjelaskan tentang hasil yang didapat dari pengujian tiap blok sistem secara keseluruhan.

Bab 5 : PENUTUP

Bab ini menjelaskan tentang kesimpulan meliputi kekurangan-kekurangan pada kerja alat dari hasil analisa serta saran untuk pengembangan ke depan.

1.7 Relevansi

Hasil dari tugas akhir ini diharapkan dapat memberikan manfaat Sebagai inovasi teknologi pengendali lengan robot menggunakan kendali lengan manusia.

Halaman ini sengaja dikosongkan

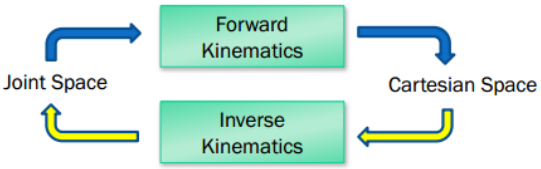
BAB II
TEORI PENUNJANG

Teori penunjang dalam bab ini menjelaskan tentang teori yang berhubungan dengan keseluruhan sistem yang akan dibuat pada tugas akhir ini.

2.1 Kinematika Robot

Kinematika adalah ilmu yang mempelajari gerak tanpa mengindahkan gaya yang menyebabkan gerakan tersebut. Kinematika robot adalah suatu disiplin ilmu yang mengaplikasikan geometri untuk mempelajari gerakan suatu titik yang menyusun struktur sistem robot [2]. Geometri yang dimaksud adalah setiap bagian robot dimodelkan sebagai *rigid bodies* (benda tegar) dan setiap sambungan bagian robot menghasilkan gerakan translasi atau rotasi secara murni. *Rigid bodies* (benda tegar) adalah suatu kesatuan benda secara ideal yang bentuk dan ukurannya tidak berubah ketika mendapatkan suatu gaya tertentu. Setiap robot bagian robot secara teori dianggap sebagai benda tegar, meskipun pada kenyatannya tidaklah sama. Robot kinematics mempelajari hubungan antara dimensi dan setiap sambungan (*joint*) dari beberapa bagian (*link*) kinematika dan posisi, kecepatan dan akselerasi dari setiap bagian robot untuk merencanakan kontrol pergerakan dan menghitung gaya dan torsi aktuator.

Robot terdiri dari link yang dihubungkan oleh joint. Biasanya setiap joint dilengkapi dengan instrument untuk mengukur posisinya relative terhadap posisi link yang diukur. Pada joint yang berputar (*rotary atau revolute*) perpindahan poisi link disebut dengan sudut joint (*joint angles*). Beberapa robot juga terdiri dari sambungan yang bergerak secara translasi atau prismatic. Perpindahan gerakan pada joint yang seperti ini biasanya disebut *joint offset.* Setiap bagian joint akan menghasilkan tingkat kebebasan robot dalam bergerak. Dengan bertambahnya maka bertambah pula gerakan robot. Jumlah yang menunjukkan tingkat keluesan robot biasanya disebut sebagai *Degree of Freedom (Dof)* [2].



Gambar 2. 1 *Forward Kinematic* dan *inverse kinematic*

Ada dua persamaan yang biasa digunakan pada kinematika robot, yaitu *forward kinematic* dan *inverse kinematic*. *Forward* kinematic digunakan untuk mendapatkan persamaan kinematika untuk menghitung posisi dari *end-effector* dengan mengetahui nilai sudut di setiap parameter *joint*.

Permasalahan pergerakan lengan robot adalah tiap-tiap joint robot hanya dapat diakses dengan memberikan nilai berupa sudut, sedangkan yang diinginkan pengendali robot adalah lengan robot dapat menjangkau posisi ujung lengan. *Inverse kinematic* adalah metode yang digunakan untuk mendapatkan persamaan kinematika untuk menghitung sudut dengan diketahui posisi akhir dari lengan robot yang diinginkan.

2.2 Motor Servo

Servo Motor adalah suatu aktuator yang dapat dikontrol dengan suatu sudut perputaran atau perpindahan linear. Motor servo mempunyai konfigurasi gear sehingga motor servo dapat mudah dikontrol. Motor servo pada umumnya dapat dikontrol mulai dari sudut 0 hingga 180 derajad. Ada pula motor servo yang dapat dikontrol secara kontinyu dengan kecepatan yang dapat diatur pula.

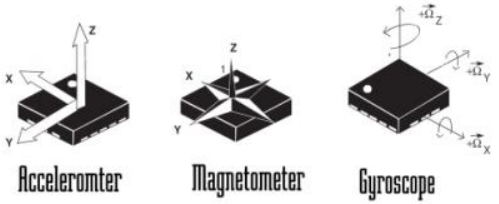
Servo motor dikontrol menggunakan sinyal PWM dengan frekuensi 50Hz. Panjang sinyal pulse yang mengontrol bermacam-macam. Standard panjang sinyal pulse untuk mengontrol servo adalah 1000 microsecond sampai dengan 2000 microsecond, namun biasanya sinyal pulse yang mengontrol motor servo berkisar 700 microsecond sampai dengan 2300 microsecond.

Pada penelitian ini digunakan beberapa motor servo sebagai penggerak lengan robot. Dengan motor servo, setiap sudut bagian lengan robot dapat dihitung dengan mudah. Kelemahan dari motor servo adalah harganya yang mahal seiring dengan bertambanhnya

torsi yang dapat di topang motor servo. Keterbatasan torsi dan ketidak terjangkauan harga motor servo seringkali mengakibatkan berubahnya konstruksi lengan robot.

2.3 Sensor IMU (*Inertial Measurement Unit)*

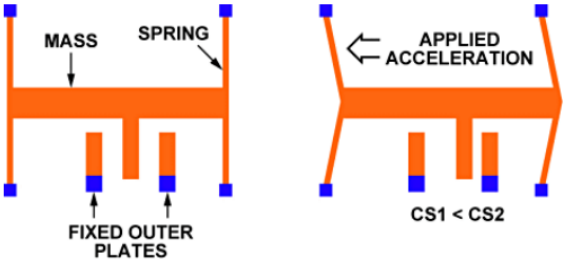
IMU adalah singkatan dari *Inertial Measurement Unit,* adalah sebuah sensor yang digunakan untuk mengukur sudut *roll*, sudut *pitch*, dan sudut *yaw*. Implementasi yang paling sering digunakan ketiga sudut ini adalah untuk menentukan orientasi dari sebuah pesawat udara saat terbang relatif terhadap bumi. Dengan pendekatan ketiga sudut (*roll, pitch, yaw)* ke sudut Euler maka akan didapatkan nilai kartesian dari sudut-sudut pada sumbu x, sumbu, dan sumbu z. Dalam mengukur orientasinya, IMU memerlukan tiga sensor pendukung yaitu magnetometer, akselerometer, dan giroskop.



Gambar 2. 2 Komponen sensor IMU

2.3.1 Akselerometer

Akselerometer adalah sensor yang digunakan untuk mengukur percepatan suatu objek. Akselerometer dapat mengukur percepatan dinamis dan statis. Pengukuran percepatan dinamis adalah pengukuran percepatan pada objek yang bergerak, sedangkan pengkuruan percepatan statis adalah pengukuran percepatan gravitasi bumi. akselerometer dapat digunakan untuk mengukur sudut kemiringan (*tilt*) [3].



Gambar 2. 3 MEMS Akselerometer [3]

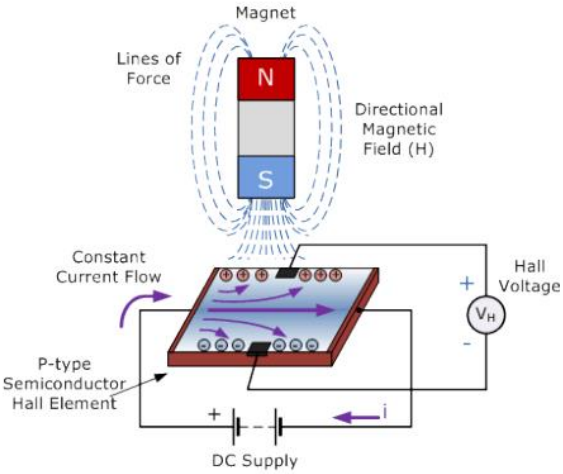
Data yang diperoleh modul sensor akselerometer adalah data mentah percepatan gravitasi yang direpresentasikan dlaam *signed integer* 16 bit. Pada perancangan tugas akhir ini, skala penuh yang dipilih adalah 8g artinya data mentas sejumlah 65536 mewakili kecepatan sudut mulai 0g hingga 8g. Sensitifitas modul sensor adalah saat skala penuh adalah 4 mg/digit.

2.3.2 Magnetometer

Magnetometer pada sensor IMU dapat mendeteksi medan magnet bumi sehingga dapat dijadikan acuan pada kompas digital. Sensor kompas digital dan sensor magnetometer pada hakikatnya memiliki prinsip kerja yang sama. Sensor kompas digital merupakan modul sensor magnetometer dengan keluaran berupa sudut yang menyatakan arah hadap. Sedangkan magnetometer keluarannya berupa besar medan magnet bumi yang diukur dalam tiga sumbu yang dapat digunakan untuk menentukan sudut arah hadap dengan rumusan tertentu.

Data yang diperoleh modul sensor magnetometer adalah data mentah gaya medan magnet bumi yang direpresentasikan dalam *signed integer* 16 bit. Di perancangan, skala penuh yang dipilih adalah 8, 1 gauss artinya data mentah sejumlah 65536 mewakili gaya medan magnet bumi mulai 0 gauss hingga 8, 1 gauss.

Pada sensor magnetometer, untuk menghindari kesalahan pengukuran pada keadaan sensor miring, dapat digunakan perhitungan kompensasi kemiringan dengan menggunakan keluaran sensor akselerometer. Jenis sensor yang digunakan pada tugas akhir ini adalah menggunakan *hall effect* [4].



Gambar 2. 4 Magnetometer dengan *Hall Effect* [4]

2.3.3 Giroskop

Giroskop digunakan untuk mengukur orientasi berdasarkan prinsip momentum sudut. Giroskop mengukur kecepatan sudut kerangka acuan inersia. Sudut orientasi berupa gerak *roll, pitch,* dan *yaw* didapatkan dengan mengintegerasikan kecepatan sudut. Sehingga hasil dari giroskop adalah percepatan sudut.

Giroskop berbeda dengan akselerometer dan kompas. Akselerometer mengukur gerakan linier acuan gravitasi. Akselerometer dapat memberikan pengukuran sudut kemiringan (*tilt*). Saat sistem berotasi atau bergerak, akselerometer tidak dapat mengikuti pergerakan yang cepat dikarenakan responnya lambat dan memiliki noise, sehingga tidak dapat digunakan untuk pengukuran sudut orientasi dalam pergerakan lengan manusia.

Sedangkan kompas mengukur gerakan linier dengan acuan medan magnet bumi. kompas dapat mengukur gerak yaw atau arah mata angin namun tidak dapat mengukur gerak *roll* dan *pitch*.

Keluaran giroskop berupa data kecepatan sudut. Kecepatan sudut adalah besaran vektor yang menyatakan frekuensi sudut suatu

benda terhadap sumbu putarnya. Satuan untuk kecepatan sudut adalah radian per detik.

2.4 AHRS (*Attitude Heading Reference System*)

Attitude Heading Reference System terdiri dari sensor pada tiga sumbu yang dapat menunjukkan informasi orientasi pesawat termasuk *roll*, *pitch*, dan *yaw*. AHRS dirancang untuk menggantikan instrumen penerbangan yang berupa giroskop mekanis tradisional dan memberikan kehandalan dan akurasi.

AHRS terdiri dari sensor *gyroscope*, *accelerometer*, *magnetometer* pada ketiga sumbu. Perbedaan utama antara IMU dan AHRS adalah pada AHRS ditambahkan pemroses informasi *attitude* dan *heading* dalam satu perangkat dibandingkan IMU yang memberikan data sensor pada perangkat tambahan untuk menghitung *attitude* dan *heading*. dapat Keluaran dari AHRS adalah berupa sudut Euler. IMU juga dapat digunakan sebagai perangkat AHRS dengan tambahan pemroses data mentah sensor.

Estimasi non linear seperti Kalman Filter dapat digunakan untuk menghitung informasi AHRS dari sensor IMU. Namun perhitungan kompleks pada Kalman filter sangat sulit diterapkan pada mikrokontroller karena keterbatasan pemrosesan data.

Pada tugas akhir ini digunakan filter kuarternion untuk mendapatkan data AHRS dan kemudian dikonversi menjadi sudut Euler. Perhitungan pada filter kuarternion lebih sederhana dari pada Kalman filter sehingga pemrosesan pada mikrokontroller dapat dilakukan dengan lebih cepat.

2.4.1 Kuaternion

Dalam matematika, Kuaternion merupakan perluasan dari bilangan-bilangan kompleks yang tidak komutatif, dan diterapkan dalam mekanika tiga dimensi. Kuaternion ditemukan oleh ahli matematika dan astronomi Inggris, William Rowan Hamilton, yang menurunkan aritmatika kompleks ke kuaternion [5].

Sebagai himpunan, kuaternion berlambang **H** (dinotasikan sesuai orang yang menemukannya Hamilton), sama dengan **R**4 yang merupakanruang vektor bilangan riil empat dimensi. **H** memiliki tiga macam operasi: pertambahan, perkalian skalar dan perkalian kuaternion. Elemenelemen kuaternion ditandakan sebagai *1*, *i*, *j* dan

k (*i*, *j* dan *k* adalah komponen imaginer), dan dapat ditulis sebagai kombinasi linear,*a* + *bi* + *cj* + *dk* (*a*, *b*, *c*, dan *d* adalah bilangan riil).

Tabel 2. 1 Tabel kuaternion

x	l	i	j	K
l	l	i	j	k
i	i	-l	k	-j
j	j	-k	-l	i
k	k	j	-i	-l

Konsep dari kuaternion adalah membagi sumbu kordinat yang semula 3 dimensi menjadi 4 dimensi.Kuaternion adalah matrik yang terdiri atas vektor dan skalar. Kenapa di sebut kuaternion karena pada dasarnya kuarternion mempunyai 4 elemen "q". Terdiri q0 – q3, dimana q0 = indikasi dimensi dari arah vector, q1 = mewakili vektor sumbu x, q2 =mewakili vektor sumbu y, q3 = mewakili vektor sumbu z. dapat dilihat pada tabel 2.1 aturan perkalian bilangan kompleks pada kuaternion.

Keluaran dari perhitungan kuaternion adalah berupa satuan kuaternion [6], yaitu:

$$\boldsymbol{q} = [q_0\ q_1\ q_2\ q_3]^T \tag{2.1}$$

Kuaternion dapat diturunkan menjadi rotasi pada sumbu dengan persamaan berikut ini:

$$\boldsymbol{q}_0 = \cos\left(\frac{\alpha}{2}\right) \tag{2.2}$$

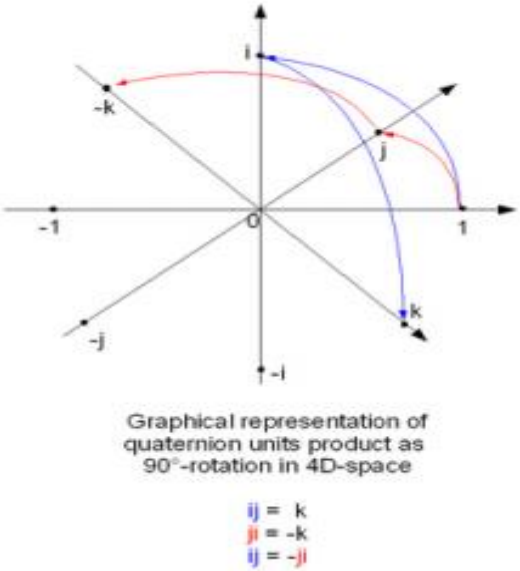
$$\boldsymbol{q}_1 = \sin\left(\frac{\alpha}{2}\right) \times \cos(\beta_x) \tag{2.3}$$

$$\boldsymbol{q}_2 = \sin\left(\frac{\alpha}{2}\right) \times \cos(\beta_y) \tag{2.4}$$

$$\boldsymbol{q}_3 = \sin\left(\frac{\alpha}{2}\right) \times \cos(\beta_z) \tag{2.5}$$

Dimana α adalah sudut rotasi sederhana (nilai dalam radian dari sudut rotasi) dan cos (βx), cos (βy), dan cos (βz) adalah "arah cosinus" pada sumbu rotasi (Teorema Euler). Representasi

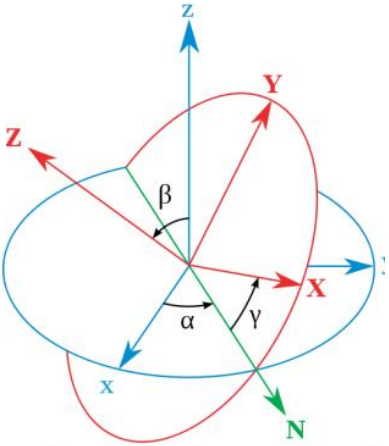
kuaternion dalam ruang 4 dimensi dapat terlihat pada gambar berikut



Gambar 2. 5 Representasi Kuaternion [6]

2.4.2 Sudut Euler

Sudut Euler merepresentasikan orientasi tiga dimensi dari suatu objek menggunakan kombinasi tiga rotasi sumbu yang berbeda. Informasi orientasi menggunakan sudut Euler lebih sederhana daripada kuaternion. Rotasi sudut Euler pada sumbu x, y, dan z dilambangkan dengan nama *roll* (Φ), *pitch* (θ), dan *yaw* (ψ) serperti terlihat pada gambar.



Gambar 2. 6 Pendekatan kuaternion dengan sudut Euler [7]

Berikut ini adalah persamaan untuk mengkonveri satuan kuaternion menjadi sudut Euler susai pendekatan kuaternion dengan sudut Euler [7]

$$\begin{bmatrix}\Phi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix}atan2(2(q0q1+q2q3), 1-2(q1^2+q2^2)\\ \arcsin(2(q0q1-q2q3))\\ atan2(2(q0q3+q1q2), 1-2(q2^2+q2^2\end{bmatrix} \quad (2.6)$$

Ada nilai pada saat orientasi sensor tidak dapat direpresentasikan menggunakan sudut Euler. Orientasi tersebut terjadi pada saat posisi *pitch* 90 derajat, sehingga *yaw* dan *roll* menjadi satu sumbu. Kondisi tersebut dinamakan *Gimbal Lock* [8].

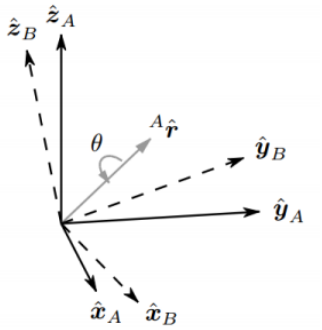
Semua orientasi atau AHRS yang menggunakan sudut Euler akan selalu gagal untuk menghasilkan informasi ketika sudut *pitch* mendekati 90 derajat. Ini adalah masalah mendasar sudut Euler dan hanya dapat diselesaikan dengan beralih ke metode representasi yang berbeda.

2.4.3 Konsep Magdwick AHRS

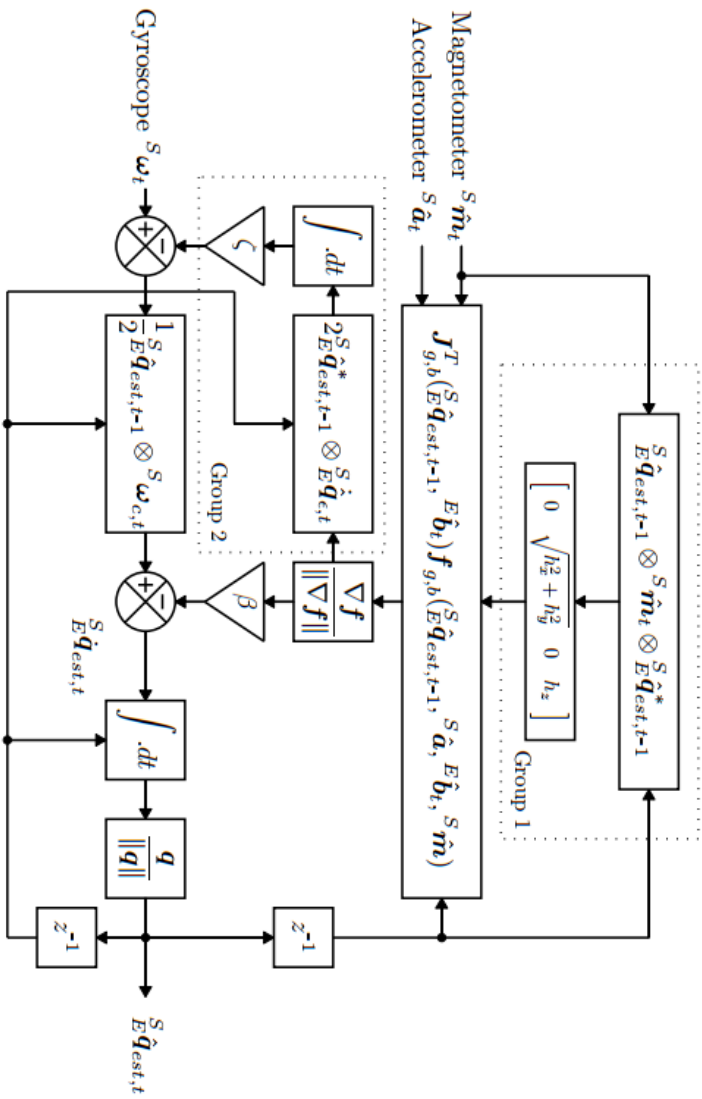
Konsep dasar dari *magdwick* AHRS adalah mencari antara relatif *frame* atau yang biasa disebut perpindahan *frame*. Contoh dapat dilihat pada gambar, *frame* A relatif terhadap *frame* B. Dimana nilai perpindahan *frame* A ke *frame* B dapat direpresentasikan dalam bentuk kuaternion seperti persamaan.

$$\frac{A}{B}\hat{q} = [q0\ q1\ q2\ q3] = [\cos\frac{\theta}{2} - rx \sin\frac{\theta}{2} - ry \sin\frac{\theta}{2} - rz \sin\frac{\theta}{2}] \quad (2.7)$$

$$B\,A\hat{q} * = B\,A\hat{q} = [q0 - q1 - q2 - q3] \quad (2.8)$$



Gambar 2. 7 Frame Kuaternion



Gambar 2. 8 Diagram Blok Algoritma AHRS Madgwick

Untuk menari nilai matrik yang baru, kedua matrik tersebut dikalikan dengan aturan *cross product* dimana dengan menggunakan aturan Hamilton. Seperti persamaan.

$$a \otimes b = [a0\ a1\ a2\ a3] \otimes [b0\ b1\ b2\ b3]$$
$$= \begin{bmatrix} a0b0 & -a1b1 & -a2b2 & -a3b3 \\ a0b1 & +a1b0 & +a2b3 & -a3b2 \\ a0b2 & -a1b3 & +a2b0 & +a3b1 \\ a0b3 & +a1b2 & -a2b1 & +a3b0 \end{bmatrix} \quad (2.9)$$

Pada gambar 2.8 dijelaskan alur dari metoe madgwick. Untuk mengetahui representasi pergeseran *frame* berdasarkan sumbu garis normal gravitasi bumi menggunakan persamaan *gradient decent* [9]. Dimana algoritma *gradient decent* mempunyai fungsi seperti persamaan. Dengan demikian apabila di *cross product* ketiga kuaternion tersebut akan menghasilkan matrik baru.

$$f(E\,\hat{Sq}, E\hat{d}, S\hat{s}) = E\,\hat{Sq} *\otimes E\hat{d} \otimes E\,\hat{Sq} - S\hat{s} \quad (2.10)$$

2.26

$$\hat{Sq} = [q1 - q2 - q3 - q4] \quad (2.11)$$

$$Eg = [0\ 0\ 0\ 1] \quad (2.12)$$

$$\hat{Sa} = [0\ ax\ ay\ az] \quad (2.13)$$

$$\mathrm{fg}(\tfrac{S}{E}\hat{q}, \hat{Sa}) = \begin{bmatrix} 2(q2q4 - q1q3) - ax \\ 2(q1q2 + q3q4) - ay \\ 2\left(\frac{1}{2} - q2^2 - q3^2\right) - az \end{bmatrix} \quad (2.14)$$

Setelah proses untuk mencari nilai *f*, kemudian dicari hasil dari matrik *jacobi* sesuai persamaan.

$$\mathrm{Jg}(\tfrac{S}{E}\hat{q}) = \begin{bmatrix} -2q3 & 2q4 & -2q1 & 2q2 \\ 2q2 & 2q1 & 2q4 & 2q3 \\ 0 & -4q2 & -4q3 & 0 \end{bmatrix} \quad (2.15)$$

Kemudian setelah kedua matrik *f* dan *j* telah didapatkan hasil. Untuk mencari nilai dari matrik *gradien decent* adalah sesuai dengan persamaan.

$$\nabla f = jT \times f \tag{2.16}$$

Untuk mencari nilai estimasi dari kuaternion, kuaternion pada giroskop dikurangkan dengan hasil kuaternion pada proses algoritma *gradien decent* yang sudah dibagi dengan nilai normalisasi dan dikalikan dengan nilai pembobotan (*beta*), sesuai dengan persamaan.

$$\dot{{}_{E}^{S}q}est, t = \dot{{}_{E}^{S}q}\ \omega,t - \beta \frac{\nabla f}{\|\nabla f\|} \tag{2.17}$$

Untuk mencari nilai β atau pembobotan dapat menggunakan persamaan.

$$\beta = \|\tfrac{1}{2}\hat{q} \otimes [0\ \bar{\omega}max\ \ \bar{\omega}max\ \ \bar{\omega}max]\| \sqrt{\frac{3}{4}} \tag{2.18}$$
$$\bar{\omega}max$$

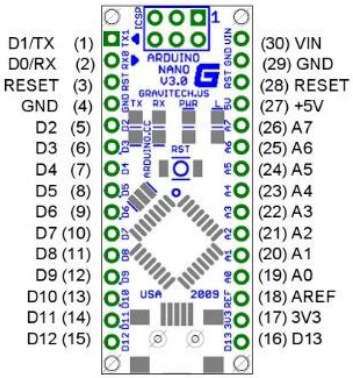
2.5 *Arduino Nano*

Arduino Nano adalah sebuah *mini board* berbasis mikrokontroler Atmega328. *Arduino UNO* mempunyai 14 pin digital *input/output* (pin 0-13) yang terdiri dari 8 pin *input* analog (pin 0-7) yang biasa digunakan untuk membaca tegangan dari sensor dan mengkonversikannya menjadi nilai 0 dan 1023, 6 pin *output* analog (pin 3, 5, 6, 9, 10, 11) yang digunakan untuk pengaturan PWM *(Pulse Width Modulation)*, sebuah osilator Kristal 16 MHz, sebuah koneksi USB, ssebuah ICSP *header*, dan sebuat tombol *reset. Arduino NANO* dapat dioperasikan dengan menggunakan *port* USB komputer, USB *charger*, atau adaptor AC-DC dengan tegangan yang direkomendasikan 7-12 Volt [9]. Spesifikasi Arduino Uno dijelaskan pada tabel 2.5.

Tabel 2. 2 Spesifikasi Arduino NANO Atmega 328

Microcontroller	Atmega328
Operating Voltage	5 V
Input Voltage	7-12 V
Digital I/O Pins	14 (6 PWM Outputs)
Analog Input Pins	8

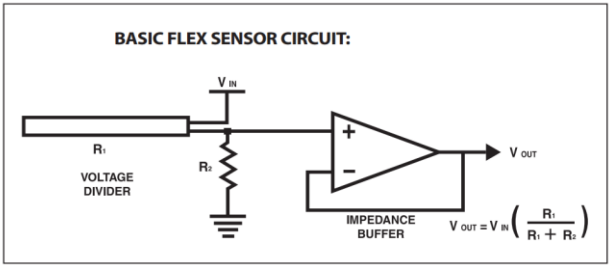
DC Current per I/O Pin	40 mA
DC Current for 3.3 V pin	50 mA
Flash Memory	32 Mbyte (Atmega328)
SRAM	2 KB (Atmega328)
EEPROM	512 byte (Atmega328)
Clock Speed	16 MHz
Length, Width	4.55 mm x 18 mm
Weigth	5 gram



Gambar 2. 9 Arduino Nano

2.6 Sensor Flex

Flex sensor adalah sensor yang dapat mendeteksi kelengkungan. Prinsip kerjanya sama seperti resistor variabel. Untuk menggunakan sensor flex harus menggunakan rangkaian pembagi tegangan agar nilai perubahan kelengkungan dapat terbaca di mikrokontroler. Kelengkungan dari *flex* sensor berbanding lurus dengan kenaikan hambatan yang dihasilkan flex sensor.



Gambar 2. 10 Rangkaian untuk mengakuisisi data sensor flex

Dengan perubahan nilai tegangan tersebut dapat dikonversi ke data ADC (*Analog to Digital Conversion*) dengan mengikuti persamaan

$$\text{Data ADC} = \frac{Vin\ Analog}{5} \text{ x } 1024 \qquad (2.19)$$

Pada tegangan nilai Vi bernilai 5 volt sesuai keluaran arduino yang digunakan, nilai R1 adalah 10 K sesuai dengan resistor pembagi tegangan yang digunakan dan R2 adalah nilai dari sensor flex yang terukur. Pada data ADC VinAnalog adalah nilai tegangan luaran dari sensor flex yang terukur.

Gambar 2. 11 Sensor flex

2.7 Telemetri

Telemetri adalah perangkat komunikasi nirkabel dengan gelombang radio berfrekuensi tinggi yang digunakan untuk

pengiriman data. Terdapat dua komponen utama dalam telemetri yaitu *transmitter* dan *receiver*. *Transmitter* berfungsi untuk mengirimkan data dari mikrokontroler utama menuju mikrokontroler lain. Sedangkan *Receiver* berfungsi untuk menerima data dari *transmitter*.

Gambar 2. 12 Produk telemetri dari RC Timer

Pada tugas akhir ini, telemetri digunakan untuk mengirimkan data dari mikrokontroler pemindai gerak menuju mikrokontroler pada robot. Telemetri tersebut menggunakan frekuensi sebesar 433 MHz dengan prediksi jarak maksimum pada 1,60934 kilometer.

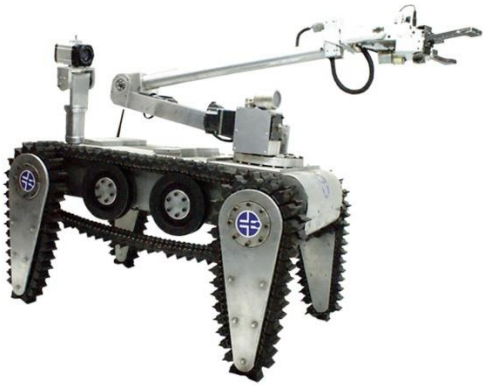
Sistem komunikasi yang digunakan adalah berupa *full duplex* (komunikasi dua pihak) dengan Tx/Rx yang dirangkai *cross*. Baud rate yang digunakan sebesar 38400. Fungsi Baud rate adalah untuk mengatur kecepatan transmisi, karena menggunakan 38400 maka kecepatan transmisi sebesar 38400 bps (bit per skon).

2.8 Tinjauan Pustaka

Tinjauan pustaka bertujuan untuk membandingkan perangkat teknologi yang telah ada dengan perangkat teknologi yang dirancang pada tugas akhir ini. Tinjauan pustaka mencakupi *mobile robot* yang telah ada, sistem pengendali robot tersebut, dan sensor yang digunakan.

2.8.1 Robot Morolipi V.1 dan Morolipi V.2

Robot sebelumnya yang telah digunakan adalah robot morolipi v.1 dan morolipi v.2. kedua robot tersebut memiliki dimensi 1 meter x 1 meter x 0.9 meter (panjang x lebar x tinggi). Dengan bobot robot seberat 100 kg.



Gambar 2. 13 Robot morolipi yang dikembangkan oleh LIPI

Mobile robot ini memiliki lengan dengan 5 derajat kebebasan yang dapat dikontrol secara manual tiap-tiap *joint*. Dilengkapi dengan *gripper* yang dapat memotong kabel setebal 2 mm. Kemampuan untuk mengangkat objek yang terbilang rendah, hanya mampu mengagkat 150 gram. Sedangkan jarak transmisi untuk mengengalikan robot adalah 6 km. Ketelitian pergerakan robot dan lengannya sangat bergantung pada keahlian dari operator.

Gambar 2. 14 Pengendali robot morolipi V.2 [1]

Beberapa kekurangan yang dimiliki oleh robot morolipi adalah sebagai berikut:

1. Bobot robot 100 kg dapat memperlambat manuver pergerakan robot.
2. Dimensi robot terbilang besar sedangkan pada pengujian hanya mampu mengangkat 150 gram dan memotong kabel setebal 2mm.
3. Sistem kendali robot dengan remote kontrol yang memiliki intrepretasi yang sangat buruk dengan kendali manual tiap-tiap *joint.* Sehingga sangat terpengaruh pada keahlian operator dalam mengendalikan robot.

2.8.2 *Robot-arm controller using LEAP motion controller*

Teknologi yang sudah dikembangkan untuk menghitung pergerakan tangan manusia dan menggerakkan lengan robot adalah perangkat yang diciptakan oleh Microsoft Research dengan nama *leap motion controller*. Didalam perangkat tersebut, untuk memindai pergerakan tangan menggunakan sensor LED (*Light Emitting Diode*) infra merah dengan cakupan sudut 140-150 derajat dan dua buah kamera dengan kecepatan 200 FPS (*Frame per Second*) [12].

Gambar 2. 15 *leap motion controler*

Perbedaan perangkat tersebut dengan alat yang diciptakan pada tugas akhir ini adalah sebagai berikut:

1. Sensor untuk memindai berupa kamera dan sensor LED infra merah. Sedangkan pada tugas akhir ini menggunakan sensor IMU.
2. Untuk mendapatkan posisi gerakan lengan menggunakan *image processing* dengan membutuhkan perangkat laptop. Sedangkan pada tugas akhir ini menggunakan analisa geometri berdasarkan sudut-sudut setiap sendi lengan manusia.
3. Sistem komunikasi pengiriman data menggunakan kabel. Sedangkan pada tugas akhir ini, dirancang untuk dapat dikendalikan pada jarak 100 meter.
4. Sensor tidak diletakkan di tangan penggunanya. Sedangkan pada tugas akhir ini, sensor diletakkan pada setiap sendi yang mempengaruhi perpindahan gerak.

Halaman ini sengaja dikosongkan

BAB III
PERANCANGAN SISTEM

Pada bab ini akan dijelaskan perancangan sistem mulai dari perancangan perangkat keras hingga perancangan perangkat lunak. Perancagan perangkat keras meliputi perancangan mekanik dan perancangan komponen elektrik. Perancangan mekanik terdiri dari pembuatan mekanisme peletakan sensor di lengan manusia dan mekanisme pergerakan lengan robot dan gripper. Perancangan komponen elektrik antara lain memadukan sensor dengan mikrokontroler, pembuatan supply daya untuk robot dan mikrokontroler pada robot dan juga pada lengan manusia. Perancangan perangkat lunak meliputi proses akuisisi data sensor IMU dan flex, perancangan algoritma dengan kuaternion, perancangan algoritma *forward kinematic,* perancangan algoritma *inverse kinematic* pada lengan robot, dan algoritma mengsinkronisasi gerak lengan manusia ke gerak lengan robot.

3.1 Diagram Blok Sistem

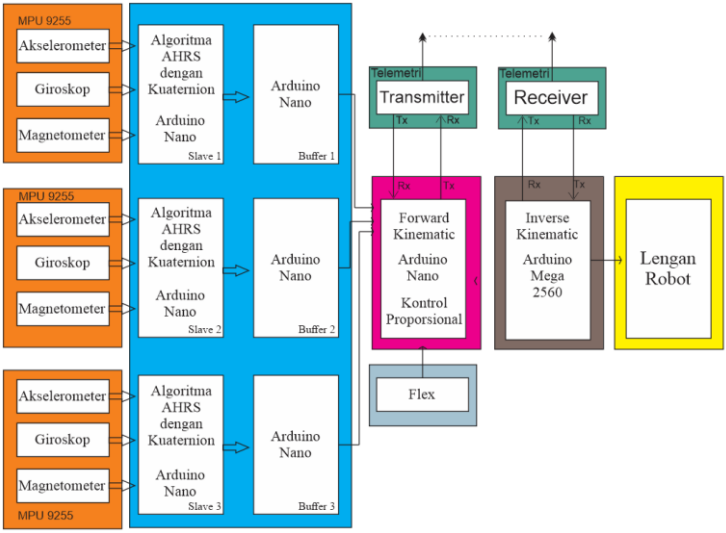
Pada tugas akhir ini, sensor IMU dengan komunikasi I2C (*Inter-Integrated Circuit*) yang diletakkan di masing-masing joint lengan manusia digunakan untuk mengetahui besar vektor perpindahan lengan manusia. Informasi yang dihasilkan sensor IMU di masing-masing joint berupa besaran sudut *roll*, sudut *pitch,* dan sudut *yaw.* Dengan pendekatan sudut Euler, sudut-sudut tersebut ditransformasikan ke sudut-sudut dalam koordinat kartesian.

Dikarenakan pada tugas akhir ini sensor IMU yang digunakan memiliki *address* I2C yang sama dan tidak dapat diganti maka pengiriman data dari mikrokontroler *slave* ke mikrokontroler *master* tidak dapat langsung menggunakan I2C atau juga serial. Hal ini terjadi karena ketika menggunakan I2C ataupun serial dapat mengganggu pengiriman semua mikrokontroler *slave* yang digunakan.

Informasi sudut-sudut tersebut lalu diolah dengan metode *forward kinematic* sehingga menghasilkan nilai posisi *end-effector*. Perpindahan posisi *end-effector* ini lalu dikirimkan ke robot sehingga robot mengetahui terjadinya perpindahan gerak. Pergerakan lengan robot akan mengikuti posisi *end-effector* dengan telah diketahuinya

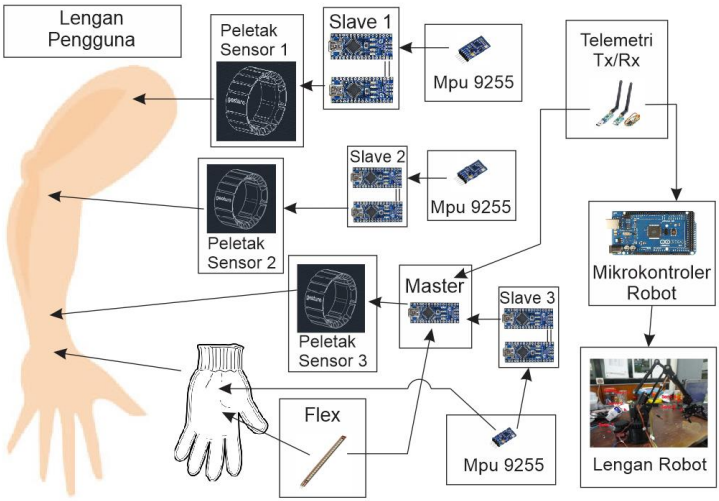
besarnya nilai perpindahan. Metode untuk menggerakkan lengan robot adalah dengan inverse kinematic.

Mekanisme untuk menggerakkan *gripper* robot adalah dengan menggunakan *feedback* dari sensor *flex*. Ketika tangan mengepal, sensor *flex* dalam kondisi melengkung, terjadinya perubahan nilai resistansi ini lalu diolah menjadi gerakan *gripper*. Sistem pengiriman data menggunakan metode nirkabel dengan telemetri yang bertujuan untuk mengirim data pada jarak lebih dari 100 meter. Berikut ini adalah diagram blok rancangan dasar sistem:



Gambar 3. 1 Diagram blok rancangan sistem

3.2 Perancangan Perangkat Keras



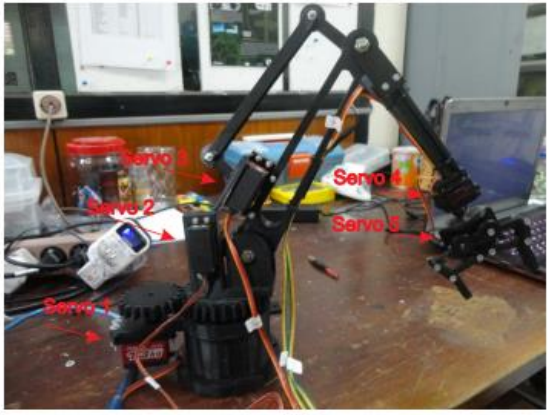
Gambar 3. 2 Perancangan perangkat keras

Perangkat keras pada tugas akhir ini antara lain, lengan robot, mekanik peletak sensor, sensor IMU, sensor flex, arduino nano, arduino Mega.

Gambar 3. 3 Robot secara keseluruhan

3.2.1 Lengan Robot

Robot yang digunakan adalah robot 3 DOF (*Degrees of Freedom)* dengan kata lain memiliki 3 derajat kebebasan dapat bergerak searah sumbu x, sumbu y, sumbu z dalam koordinat kartesian. Robot memiliki bagian *base*, *elbow*, hasta, dan *gripper*. Robot dengan 5 motor servo disusun dengan 3 servo pada bagian lengan dan 2 servo pada bagian *gripper*. 2 servo pada bagian *gripper* berguna untuk menggerakkan *tools* untuk mencapit dan memutar pergelangan. Massa dari lengan robot tersebut adalah 1.15 kg.



Gambar 3. 4 Penjelasan *joint* pada robot

Dengan spesifikasi ukuran panjang lengan serta panjang *end-effector* pada tabel dibawah ini

Tabel 3. 1 Ukuran panjang robot

Bagian Robot	Panjang
Lengan 1	9.5 cm
Lengan 2	19.5 cm
Lengan 3	26.4 cm
End-effector	7 cm

Panjang lengan 1 adalah panjang dari pusat *horn* motor servo 1 sampai pusat *horn* motor servo 2. Panjang lengan 2 adalah panjang dari pusat *horn*motor servo 2 sampai *horn* motor servo 3. Panjang lengan 3 adalah panjang dari pusat *horn* motor servo 3 hingga akhir *end-effector*. Panjang *end-effector* saat terbuka maksimum adalah 7 cm.

Spesifikasi sudut yang dapat diakses setiap servo pada robot dijelaskan pada tabel dibawah ini

Tabel 3. 2 Sudut maksimum yang dapat diakses pada robot

Servo ke-	Sudut Minimum	Sudut Maximum
1	0 derajat	160 derajat
2	150 derajat	180 derajat
3	20 derajat	100 derajat
4	0 derajat	180 derajat
5	45 derajat	75 derajat

Pada lengan *joint* antara servo 1 dan *base* robot terdapat roda gigi dengan perbandingan 20:30 (20 pada servo 1 dan 30 pada *base* robot). Sedangkan *joint* antara servo 2 dan lengan 3 terdapat roda gigi dengan perbandingan 11:16. Adanya roda gigi dengan perbandingan semakin besar tersebut mampu mengurangi beban yang dipikul setiap *joint*. Sehingga perhitungan sudut hasil pengolahan dari *inverse kinematic* harus dikalikan dengan perbandingannya, agar ketelitian perhitungan sudut semakin baik.

3.2.2 Mekanik Untuk Meletakkan Sensor

Perangkat mekanik ini diletakkan pada lengan manusia berguna untuk meletakkan sensor dan juga mikrokontroler. Mekanik peletak sensor ini dibuat menjadi bagian lengan dekat bahu, lengan hasta , dan pergelangan tangan.

3.2.2.1 Bagian tangan

Pada bagian ini akan menggunakan sarung tangan sebagai tempat peletakan sensor dan mikrokontroler. Dengan bahan kain yang lembut bertujuan agar nyaman dikenakan, ringan dan tidak mengganggu pergerakan pergelangan tangan.

3.2.2.2 Bagian lengan dekat bahu dan hasta

Pada bagian hasta dan lengan dekat bahu dibuat seperti pada gambar dibawah ini

Gambar 3. 5 Perancangan mekanik peletak sensor

Perangkat ini dibuat menyerupai gelang dengan bahan dasar plastik filamen dengan memperhatikan aspek kenyamanan. Dengan menggunakan *software* untuk menggambar 3*D*, alat tersebut didesain sebagai tempat untuk meletakkan sensor IMU dan juga mikrokontroler arduino nano. Dengan tambahan kain karet membuat perangkat ini menjadi elastis dan dapat diubah ukurannya. Perangkat ini dibuat sendiri.

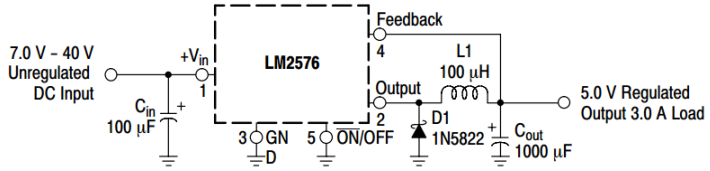
Tabel 3. 3 Bobot mekanik untuk meletakkan sensor

Wearable device pada *elbow*	156,25 gram
Wearable device pada hasta	93,75 gram

3.2.3 Rangkaian Suplai Daya

Suplai daya adalah perangkat elektronika yang mensuplai sumber listrik ke perangkat elektronika lainnya. Dalam suatu rangkai suplai daya terdapat sebuah regulator tegangan dimana digunakan untuk menurunkan tegangan dari satu level tertentu ke level yang diinginkan. Dalam tugas akhir ini menggunakan sebuah regulator tegangan berupa UBEC turnigy yang mampu meregulasi tegangan input dari rentang 2 sel lipo – 3 sel lipo (7,4 volt – 12,6 volt) menjadi 5 volt atau 6 volt dan mampu menyuplai beban sampai batas arus 3A.

Dalam tugas akhir ini, besar tegangan yang dibutuhkan adalah 6 V untuk suplai daya mikrokontroler, sensor, dan motor servo.

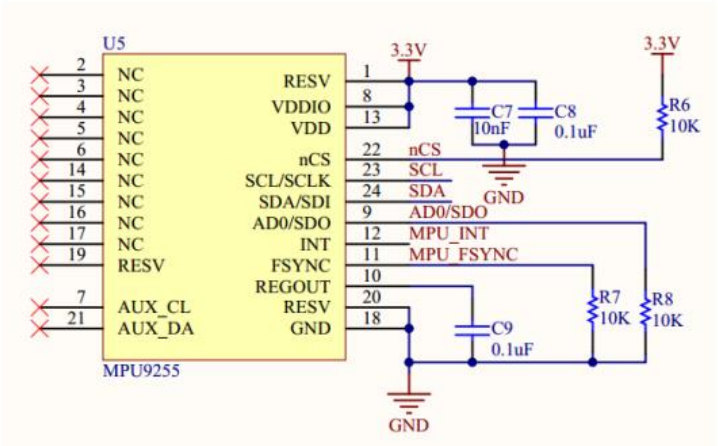


Gambar 3. 6 Rangkaian regulator tegangan 5-6V

Pada gambar 3.3 kapasitor berfungsi sebagai pengaman pada tegangan input maupun pada output agar tegangan yang akan masuk maupun yang akan dikeluarkan tidak terinterferensi oleh noise akibat adanya loncatan arus. Untuk mendapatkan keluaran tegangan yang dinginkan menggunakan perbandingan besar R yang berbeda-beda.

3.2.4 Sensor IMU

Inertial Measurement Unit (IMU) adalah sebuah sensor yang menggabungkan sensor giroskop, akselerometer dan magnetometer dalam sebuah devais. Sensor IMU yang digunakan pada tugas akhir ini adalah *Waveshare* 10 DoF IMU yang terdiri dari sebuah sensor IMU MPU 9255 dan sebuah barometer BMP180. Tegangan kerja sensor berada pada rentang 3,3 - 5 V.

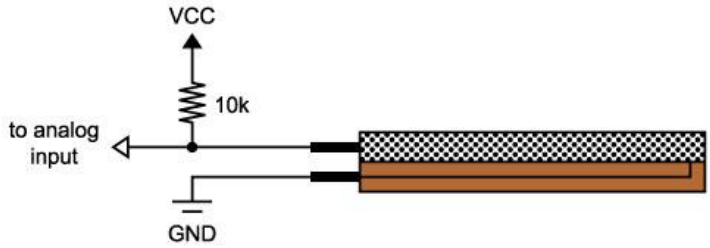


Gambar 3. 7 Rangkaian Skematik MPU 9255

MPU 9255 adalah sensor IMU yang terdiri dari akselerometer dan giroskop buatan *Invensys*, serta magnetometer buatan *Asahi Kasei Microdevices* dalam satu chip. Sedangkan BMP180 adalah sensor barometer buatan *Bosch*. Antarmuka sensor ini menggunakan I2C.

3.2.5 Sensor Flex

Pada tugas akhir ini, sensor *flex* yang akan digunakan adalah sensor buatan *sparkfun* dengan spesifikasi nilai resistansi 0-50 KΩ. namun pada realisasinya, nilai resistansi saat tidak melengkung adalah 10KΩ. Sensor yang digunakan pada tugas akhir ini memili panjang 4.4 inchi. Untuk penggunaan sensor *flex* adalah seperti pada gambar dibawah ini



Gambar 3. 8 Rangkaian pembagi tengangan untuk flex sensor

Dengan perubahan nilai tegangan tersebut dapat dikonversi ke data ADC (*Analog to Digital Conversion*) dengan mengikuti persamaan 2.19.

Pada tegangan nilai Vi bernilai 5 volt sesuai keluaran arduino yang digunakan, nilai R1 adalah 10 K sesuai dengan resistor pembagi tegangan yang digunakan dan R2 adalah nilai dari sensor flex yang terukur. VinAnalog adalah nilai tegangan luaran dari sensor flex yang terukur.

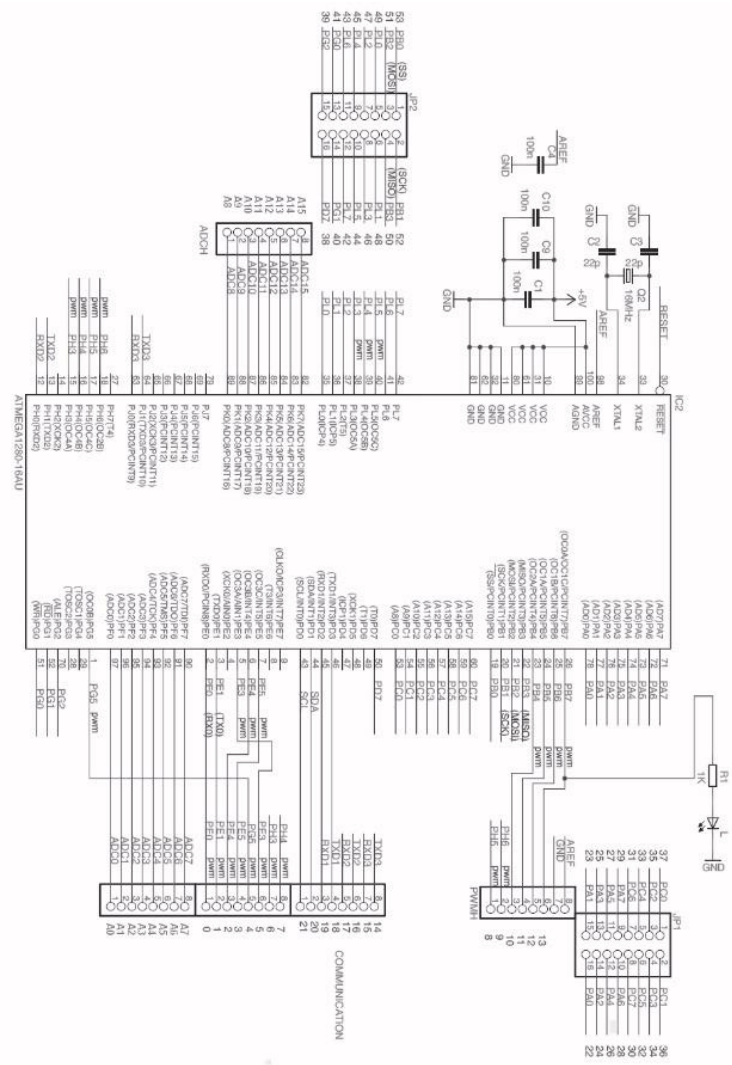
Dengan telah diketahuinya *feedback* gerak berupa data ADC, nilai tersebut dijadikan acuan untuk pergerakan *gripper*. Data nilai minimum dan maximum disinkronkan dengan pergerakan *gripper* dengan kontroler proporsional.

3.2.6 Arduino Nano

Processing unit yang digunakan pada tugas akhir ini sebagai pemroses data sensor dan mengkonversi data-data tersebut menjadi nilai pergerakan adalah Arduino Nano. Arduino Nano adalah sebuah *mini board* berbasis mikrokontroler Atmega328. Arduino Nano mempunyai 14 pin digital *input/output* (pin 0-13) yang terdiri dari 8 pin *input* analog (pin 0-7) yang biasa digunakan untuk membaca tegangan dari sensor dan mengkonversikannya menjadi nilai 0 dan 1023, 6 pin *output* analog (pin 3, 5, 6, 9, 10, 11) yang digunakan untuk pengaturan PWM *(Pulse Width Modulation)*, sebuah osilator Kristal 16 MHz, sebuah koneksi USB, ssebuah ICSP *header*, dan sebuat tombol *reset.* Arduino NANO dapat dioperasikan dengan menggunakan *port* USB komputer, USB *charger*, atau adaptor AC-DC dengan tegangan yang direkomendasikan 7-12 Volt.

3.2.7 Arduino Mega

Processing unit yang digunakan pada tugas akhir ini sebagai pemroses data *inverse kinematic* dan menggerakkan robot adalah Arduino Mega 2560. Arduino Mega 2560 merupakan sebuah board mikrokontroler berbasis ATMega2560. Modul ini memiliki 54 digital input/output dimana 15 digunakan untuk PWM output dan 16 digunakan sebagai analog input, 4 port serial, 16 MHz osilator Kristal, ICISP Header, dan tombol reset. Arduino Mega 2560 memiliki flash memory sebesar 256KB. Arduino Mega 2560 tidak memerlukan *flash* program external karena di dalam chip mikrokontroler Arduino telah diprogram dengan *bootloader* yang membuat proses upload program yang dibuat menjadi lebih sederhana dan cepat.



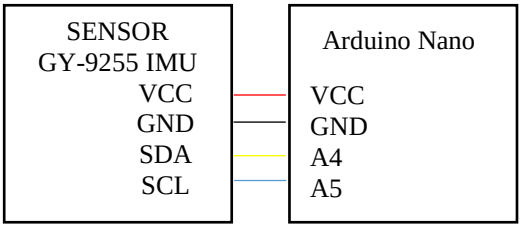
Gambar 3. 9 Skematika rangkaian arduino Mega 2650

3.3 Perancangan Perangkat Lunak (*Software)*

Pada tahap ini, algoritma pemrograman dirancang untuk melakukan beberapa proses yang dibagi menjadi lima bagian utama. Bagian pertama yaitu akuisisi data sensor yaitu IMU dan flex. Bagian kedua adalah perancangan akuisisi data perpindahan lengan pengguna dengan metode *forward kinematic*. Bagian ketiga adalah perancangan gerak *gripper* dengan mengolah data dari sensor flex dan sudut *roll* yang terletak pada tangan pengguna. Bagian keempat adalah perancangan gerak lengan robot menggunakan metode *inverse kinematic*. Bagian kelima adalah perancangan *software* untuk mengsinkronisasi pergerakan lengan pengguna dengan lengan robot.

3.3.1 Akuisisi Data Sensor IMU

Akuisisi data sensor IMU menggunakan komunikasi I2C dimana *address* sensor IMU untuk MPU6050 (sensor akselerometer dan sensor giroskop) adalah 0x68 saat *address device* AD0 =0. Sedangkan *address* untuk megnetometer adalah 0x0C. Dibawah ini adalah ilustrasi *wiring* antara sensor dan mikrokontroler pembaca sensor



Keluaran sensor ini menghasilkan nilai mentah yang biasa disebut *raw data* yang terdiri dari ax, ay, az, gx, gy, dan gz (a untuk akselerometer dan g untuk giroskop). Sedangkan untuk magnetometer nilai luaran yang dihasilkan berupa mx, my, dan mz. Nilai-nilai tersebut belum searah dengan sumbu-sumbu dalam derajat kartesian, untuk itu diperlukan sebuah algoritma AHRS yang akan menghasilkan nilai dalam derajat Euler yang akan dijelaskan dalam sub bab selanjutnya.

3.3.2 Kalibrasi Magnetometer

Pada tugas akhir ini, kalibrasi magnetometer adalah proses yang dilakukan setiap sensor magnetometer pada masing-masing *joint*. Proses ini dilakukan saat awal menggunakan sensor. Dengan menyimpan nilai *offset* pada EEPROM (*Electrically Erasable Programmable Read-Only Memory*).

Ketika mengakuisisi data magnetometer untuk dikonversi menjadi sudut dalam sumbu *yaw*, perlu diketahui terdapat jenis kesalahan yang mempengaruhi pembacaan sensor. Kalibrasi dibutuhkan untuk melihat *offset* dari sensor dan meningkatkat akurasi nilai output sensor agar mendekati nilai sebenarnya.

Nilai sumbu *yaw* dapat ditentukan melalui penggunaan giroskop, akselerometer, dan magnetometer tiga sumbu yang terserdia di sensor IMU. Namun sistem kompas harus diterapkan dengan benar dan dapat mengkompensasi pengaruh elevasi dan kemiringan sudut.

Tanpa kalibrasi, data magnetometer akan terdistorsi sehingga pembacaan menjadi tidak tepat. Untuk melakukan kalibrasi dilakukan penghitungan *offset* yaitu dengan persamaan :

$$\alpha = \frac{x_{max} + x_{min}}{2} \tag{3.1}$$

$$\beta = \frac{y_{max} + y_{min}}{2} \tag{3.2}$$

$$\gamma = \frac{z_{max} + z_{min}}{2} \tag{3.3}$$

Hasil dari persamaan diatas adalah besar *offset* pada sumbu x, y, dan z. kemudian pembacaan setiap sumbu magnetometer dikurangkan dengan hasil tersebut.

$$Mx = Mx - \alpha \tag{3.4}$$

$$My = My - \beta \tag{3.5}$$

$$Mz = Mz - \gamma \tag{3.6}$$

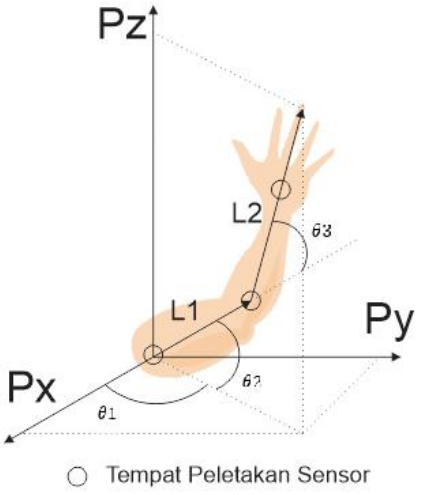
3.3.3 Kompensasi Kemiringan Kompas

Tujuan digunakannya kompas dalam tugas akhir ini karena terjadinya *drifting* pada sumbu *yaw*. *Drifting* yang dimaksud adalah nilai sudut *yaw* bergeser dan besarnya pergeseran tersebut bersifat *random*. Giroskop dapat mengukur perubahan kecepatan sudut (dalam hal ini yaw). Akselerometer akan mengukur percepatan gravitasi bumi sehingga derajat kemiringan. Penggunaan kompas pada tugas akhir ini akan membantu sistem untuk mendapatkan nilai sumbu *yaw* yang lebih akurat. Namun ketika sensor magnetometer dimiringkan, hasil pembacaan akan berubah karena terpengaruh oleh medan magnet bumi yang berubah terhadap gaya gravitasi bumi.

Dengan kompensasi kemiringan, maka arah dari kompas akan tetap pada arah sebenarnya dan tidak mengalami *offset*. Algoritma yang digunakan untuk mengkompensasi data kompas adalah AHRS (*Attitude Heading Reference System*). AHRS akan mengkonversi data keluaran magnetometer, akselerometer, dan giroskop yang telah dikalibrasi menjadi representasi kuaternion. Hasil dari konversi tersebut kemudian diproses pada filter kuaternion Magwick (GAMBAR). Filter tersebut menghasilkan empat elemen kuaternion yaitu q_1, q_2, q_3, dan q_4. Elemen kuaternion tersebut lalu dikonversi menjadi sudut Euler menggunakan persamaan yang menghasilkan keluaran berupa *roll* (Φ), *pitch* (θ), dan *yaw* (ψ). Sehingga didapatkan nilai *heading* dari hasil konversi kuarternion yang berupa *yaw*.

3.3.4 Analisa Geometri Pergerakan Tangan

Tujuan analisa geometri pergerakan tangan adalah untuk mengetahui besar perpindahan terhadap besaran panjang dalam derajat kartesian. Umpan balik dari sensor IMU adalah berupa sudut di masing-masing *joint*. Dengan mengetahui besar sudut di masing-masing *joint* dan panjang masing-masing lengan maka akan diketahui panjang lengan dan vektor perpindahan lengan. Terdapat 3 sudut berupa *roll* (Φ), *pitch* (θ), dan *yaw* (ψ) yang memiliki nilai mendekati sumbu x, y, dan z Dibawah ini adalah gambar analisa sumbu-sumbu pada tangan dalam derajat kartesian.



Gambar 3. 10 Analisa Tangan Pada Sumbu Kartesian

Pada lengan 1 terdapat sensor IMU 1 yang menghasilkan nilai sudut *roll_1* (Φ1), *pitch_1* (θ1), dan *yaw_1* (ψ1), pada lengan 2 terdapat sensor IMU 2 menghasilkan nilai sudut *roll_2* (Φ2), *pitch_2* (θ2), dan *yaw_2* (ψ2), dan pada pergelangan tangan terdapat sensor IMU 3 menghasilkan nilai sudut *roll_3* (Φ3), *pitch_3* (θ3), dan *yaw_3* (ψ3).

Dibawah ini adalah persamaan untuk mendapatkan panjang setiap sumbu x, y, z.

$$Px = (\, l1 * \cos\theta1 + l2 * \cos(\theta2 + \theta3) * \cos\theta1 \qquad (3.7)$$

$$Py = (\, l1 * \cos\theta1 + l2 * \cos(\theta2 + \theta3) * \sin\theta1 \qquad (3.8)$$

$$Pz = (l1 \sin\theta1 + \theta2) * \sin(\theta2 + \theta3) \qquad (3.9)$$

Dengan ketentuan $\theta1$ adalah nilai sudut *roll_1* (Φ1), $\theta2$ adalah nilai sudut *pitch_2* (θ2), dan $\theta3$ adalah nilai sudut *pitch_3* (θ3). Sudut roll setiap sensor tidak mempengaruhi panjang dari masing-masing sumbu.

Pada tugas akhir ini panjang dari perpindahan Px, Py, dan Pz menggunakan persamaan 3.7 sampai 3.9. Dalam sekali perhitungan pada mikrokontroler pada lengan pengguna menghitung dengan

waktu rata-rata 1812 µs. Untuk mengetahui pemrosesan rata-rata adalah dengan cara menghitung waktu akhir pemrosesan dikurangi waktu awal pemrosesan.

3.3.5 *Forward Kinematic*

Pada tahap ini metode *forward kinematic* digunakan untuk mendapatkan posisi *end-effector* berdasarkan posisi sudut-sudut pada *joint* dan struktur mekanik robot. Metode yang bisa digunakan dalam tugas akhir ini menggunakan *forward kinematic* dengan DH parameter.

Analisa DH parameter untuk pergerakan lengan pengguna adalah seperti pada tabel dibawah ini

Tabel 3. 4 Analisa DH parameter

Joint i	α_i	a_i	d_i	θ_i
1	-90	20	0	θ_1
2	0	27	0	θ_2
3	0	14	0	θ_3

Dengan penjelasan DH pada tabel dibawah ini

a_i	Panjang lengan antara dua *joint* axis
d_i	Jarak translasional antara dua sumbu *joint*
α_i	Sudut memutar antara dua *joint* axis yang berdekatan
θ_i	Sudut *joint* antara dua buah *joint* axis

Hasil dari transformasi matrik antara ujung lengan sampai tangan adalah sebagai berikut:

$$A_3^0 = \begin{bmatrix} c\theta_1 c_{23} & -c\theta_1 s\theta_{23} & s\theta_1 & c\theta_1(a_1 + a_2 c\theta_2 + a_3 c\theta_{23} \\ s\theta_1 c\theta_{23} & -s\theta_1 s\theta_{23} & -c\theta_1 & s\theta_1(a_1 + a_2 c\theta_2 + a_3 c\theta_{23} \\ s\theta_{23} & c\theta_{23} & 0 & a_3 s\theta_{23} + a_2 s\theta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.10)$$

Dari matrik A_3^0 diatas dikalikan dengan *transpose* matriknya. Sehingga mengasilkan persamaan *forward kinematic* dari sistem ini adalah

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} c\theta_1(a_1 + a_2c\theta_2 + a_3c\theta_{23} \\ s\theta_1(a_1 + a_2c\theta_2 + a_3c\theta_{23} \\ a_3s\theta_{23} + a_2s\theta_2 \end{bmatrix} \quad (3.11)$$

3.3.6 Perancangan Gerak *Gripper*

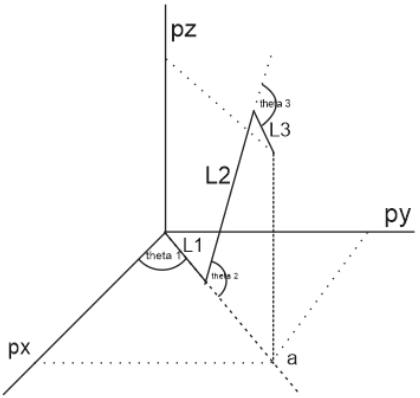
Pada tahap ini, pergerakan *gripper* menggunakan *feedback* nilai dari sensor IMU (sumbu *roll*) dan sensor flex.

Dengan menggunakan *Analog to Digital Conversion* data dari sensor flex diproses menjadi perbesaran sudut. Nilai awal sudut pada capit robot diatur pada sudut 35 derajat. Ketika tangan pengguna mengepal, terjadi perubahan sudut pada sensor flex. Dengan metode kontrol proporsional, perubahan sudut pada tangan manusia di aplikasikan ke capit robot.

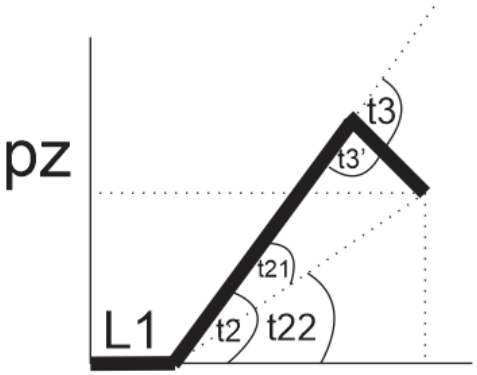
Pergerakan pada pergelangan robot disinkronisasi dengan sudut *roll* pada sensor IMU yang diletakkan di tangan pengguna. Nilai awal pergelangan tangan robot diatur pada sudut 90 derajat. Perubahan nilai sudut pada tangan pengguna akan disinkronisasi dengan pergelangan robot dengan metode *mapping* pada arduino.

3.3.7 *Inverse Kinematic*

Dibawah ini merupakan ilustrasi gambar lengan 3 DOF yang digunakan pada tugas akhir ini



Gambar 3. 11 Ilustrasi gambar lengan robot pada sumbu kartesian 3 dimensi



Gambar 3. 12 Ilustrasi gambar lengan robot pada sumbu kartesian 2 dimensi

Pada *inverse kinematic* tidak perlu menghitung gerak rotasi capit dan juga gerakan mencapit karena tidak memberikan pengaruh terhadap posisi ujung lengan.

Pergerakan robot diatur oleh *inverse kinematic* dengan metode geometri yang menghasilkan persaman dibawah ini

$$\theta_1 = \text{acos}\left(\frac{(-{a_2}^2 - {a_3}^2)}{2a_2a_3}\right) \tag{3.12}$$

$$\theta_2 = \alpha_1 + \alpha_2 \tag{3.13}$$

$$\theta_3 = atan\left(\frac{y}{x}\right) \tag{3.14}$$

Dengan ketentuan tambahan persamaan dibawah ini

$$c = \left(\left(\sqrt{x^2 + y^2}\right) - a_1\right)^2 + z^2 \tag{3.15}$$

$$\alpha_1 = acos\left(\frac{{a_2}^2 + c^2 - {a_3}^2}{2a_2c}\right) \tag{3.16}$$

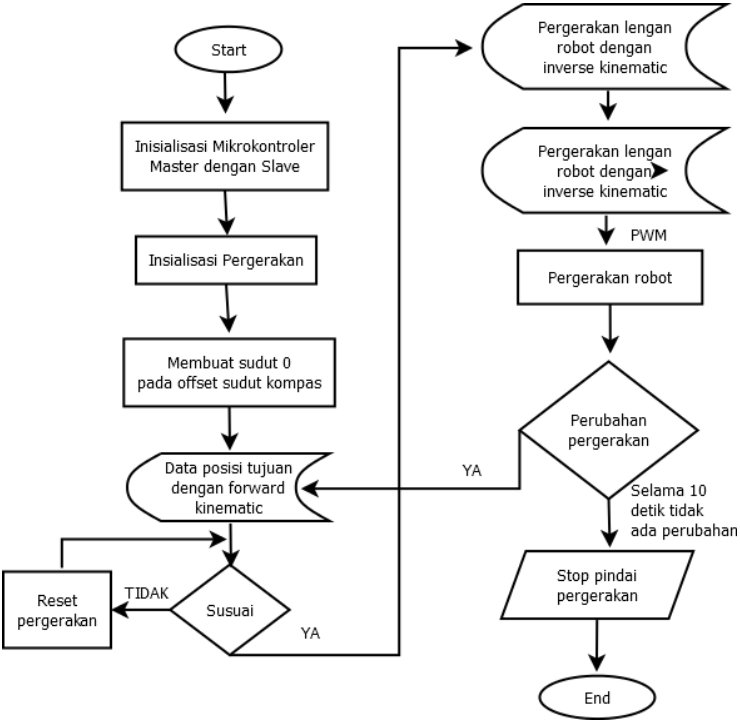
$$\alpha_2 = atan\left(\frac{z}{\sqrt{x^2 + y^2} - a_1}\right) \tag{3.17}$$

Karena nilai-nilai tersebut masih dalam radian, sedangkan yang kita butuhkan adalah nilai dalam satuan derajat, maka perlu dikonversi ke derajat dengan mengalikan hasil $\theta_1, \theta_2, dan\ \theta_3$dengan $\frac{180}{\pi}$.

3.3.8 Sinkronisasi Pergerakan

Langkah awal yang dilakukan oleh sistem adalah inisialisasi sensor. Sensor yang digunakan adalah sensor IMU dan flex. Kemudian setelah itu dilakukan inisialisasi komunikasi I2C dan Serial antara mikrokontroler *master* dan *slave*. Kemudian pengguna memberikan informasi *feedback* berupa *offset* nilai kompas. Hal ini diperlukan untuk membuat acuan arah kompas dimanapun menghadap akan menjadi sudut 0 (dalam sumbu *yaw*). Kemudian setelah pergerakan pengguna telah dipindai, nilai-nilai sudut di proses menjadi posisi *end-effector* dengan *forward kinematic*. Lalu data gerakan capit oleh sensor flex juga diproses. Kemudian data perpindahan lengan dan tanga pengguna lalu dikirimkan ke mikrokontroler robot untuk diproses dengan *inverse kinematic.* Yang

terakhir adalah algoritma untuk menjadikan gerak robot serupa dengan gerakan pengguna. Robot dapat mengetahui pergerakan lengan manusia (dalam sumbu kartesian) serta besar pergerakan yang dilakukan. Diagram blok serta flow chart dari skema alir algoritma sinkronisasi dijelaskan dengan gambar dibawah ini



Gambar 3. 13 Diagram alir proses keseluruhan

Akan tetapi lengan robot memiliki batas pergerakan derajat rotasi setiap servo, hal ini mengakibatkan adanya batasan pergerakan robot, sehingga tidak dapat mengikuti lengan manusia ketika terjadi simpangan perpindahan lengan yang terlalu besar.

Proses sinkronisasi dikatakan telah selesai jika pengguna tidak memberikan *feedback* perpindahan lengan lebih dari 10 detik. Kemudian sistem akan kembali memproses pergerakan jika tombol untuk memindai pergerakan ditekan.

3.4 Spesifikasi Robot

Tabel 3. 5 Spesifikasi robot

Jumlah *DoF*	3
Masukan Tegangan	7 V – 12 V
Jarak Jangkauan Lengan	40 cm
Dimensi Robot	75 cm x 55 cm x 64 cm
Bobot Lengan Robot	1.15 kg
Jarak Kendali	0 – 100 meter
Latency Kendali	~ 0.2 s saat lebih dari 100 m
Beban Maksimum	150 gram
Akurasi Pergerakan Lengan	90%
Akurasi Pergerakan Capit	95%

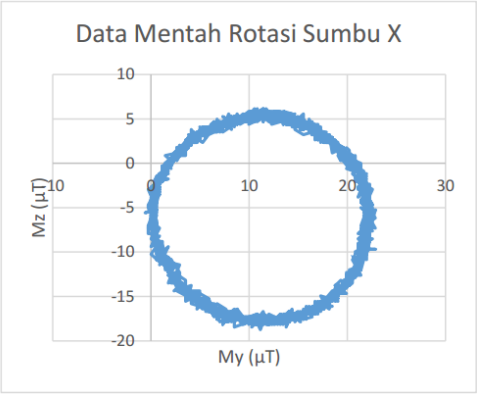
Tabel diatas adalah pemaparan spesifikasi lengan robot yang dirancang. Jarak kendali akurat adalah pada jarak 0 hingga 100 meter dengan latency kurang dari 0.2 detik. Beban maksimum yang dapat diangkat pada lengan robot adalah 150 gram. Akurasi perhitungan pergerakan lengan robot sebesar 90% dan akurasi pergerakan capit sebesar 95%.

BAB IV
PENGUJIAN DAN ANALISIS

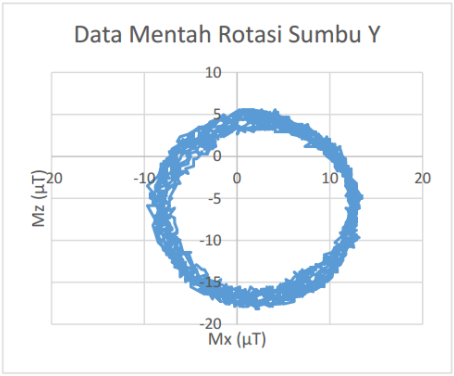
Pada bab ini akan dibahas mengenai pengujian dari sistem yang telah dirancang. Bab ini bertujuan untuk mengetahui apakah tujuan dalam perancagan sistem pada tugas akhir ini telah terlaksana atau tidak.

4.1 Pengujian Kalibrasi Sensor Magnetometer

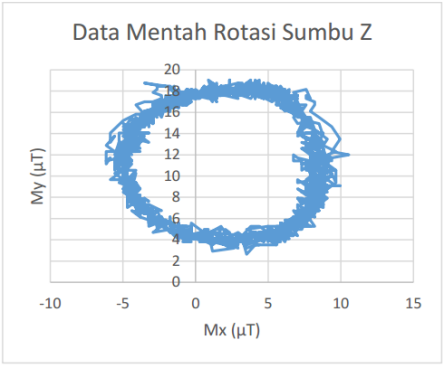
Kalibrasi sensor magnetometer sangat diperlukan untuk mengetahui besar galat (kekeliruan) pembacaan medan magnet bumi. Kalibrasi juga digunakan untuk meningkatkan akurasi pembacaan sensor. Kalibrasi dilakukan dengan memutar sensor pada setiap sumbu untuk mendapatkan data mentah sensor.



Gambar 4. 1 Data mentah sensor pada rotasi sumbu x



Gambar 4. 2 Data mentah sensor pada rotasi sumbu y



Gambar 4. 3 Data mentah sensor pada rotasi sumbu z

Gambar diatas adalah penjabaran data sensor yang diputar pada sumbu x, y, dan z. Terlihat posisi lingkaran tidak tepat di titik 0,0. Hal tersebut dapat mengakibatkan kesalahn dalam pembacaan derajat arah dari sensor. untuk menentukan besar *offset* yang diperlukan agar posisi lingakran menjadi titik 0,0 adalah menggunakan persamaan 3.1 sampai 3.3.

Program untuk melakukan proses tersebut adalah:

```
sample_count = 128;
for(ii = 0; ii < sample_count; ii++)
{
 readMagData(mag_temp);
 for (int jj = 0; jj < 3; jj++)
 {
   if(mag_temp[jj] > mag_max[jj]) mag_max[jj]
      = mag_temp[jj];
   if(mag_temp[jj] < mag_min[jj]) mag_min[jj]
      = mag_temp[jj];
 }
}
```

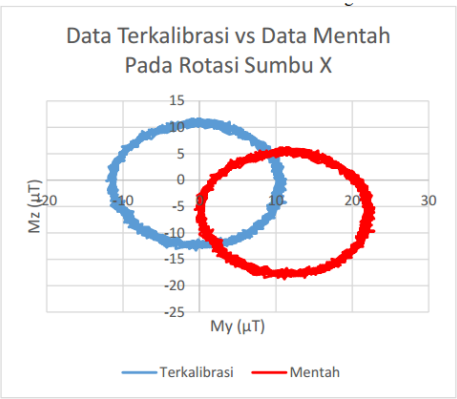
Hasil dari proses diatas adalah besar *offset* pada sumbu x, y, dan z. Kemudian pembacaan setiap sumbu kompas dikurangkan dengan hasil tersebut sesuai dengan persamaan 3.4 sampai 3.6.

Program untuk melakukan proses tersebut adalah:

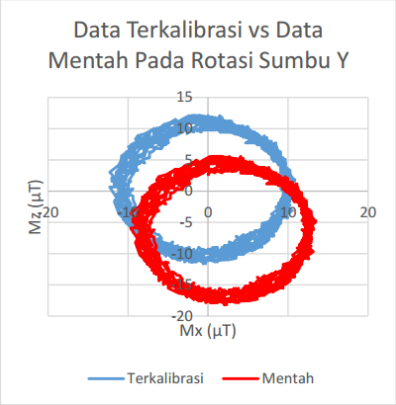
```
mx = (float)magCount[0]*mRes*magCalibration[0]
      - compCalib.magBias[0];
my = (float)magCount[1]*mRes*magCalibration[1]
      - compCalib.magBias[1];
mz = (float)magCount[2]*mRes*magCalibration[2]
      - compCalib.magBias[2];
```

Didapatkan data bahwa besar *offset* adalah:
X = 2,354 mT
Y= 12,795 mT
Z = -9,283 mT

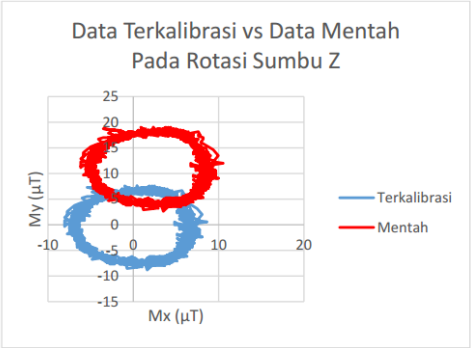
Berikut ini adalah hasil kalibrasi dari sensor magnetometer:



Gambar 4. 4 Data hasil kalibrasi pada sumbu x



Gambar 4. 5 Data hasil kalibrasi pada sumbu y



Gambar 4. 6 Data hasil kalibrasi pada sumbu y

Terlihat dari sebaran data sensor ketika diputar dalam sumbu x, y, dan z setelah dikalibrasi, posisi pusat lingkaran menjadi di titik 0,0. Dengan demikian jika data hasil kalibrasi tersebut dikonversi menjadi derajat arah akan menjadi lebih akurat.

4.2 Pengujian Kompensasi Kemiringan Kompas

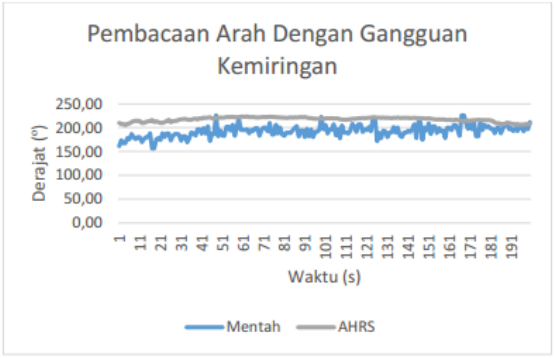
Pengujian ini bertujuan untuk mengetahui besar kekeliruan yang diakibatkan oleh error kemiringan pada sensor magnetometer seiring berjalannya waktu.

Pengujian kompensasi kemiringan kompas dilakukan dengan menggerakkan sensor pada kemiringan tertentu pada penunjukkan arah yang sama. Sehingga akan terlihat perbedaan antara arah sebenarnya dengan pembacaan sensor.

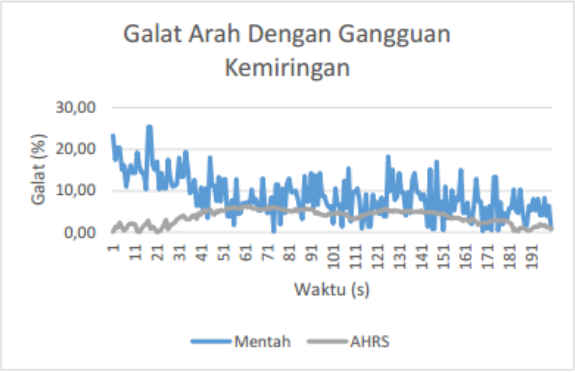
Algoritma kompensasi kemiringan kompas yang digunakan pada tugas akhir ini adalah algoritma kuaternon Magwick yang berbasis AHRS (*Attitude and Heading Reference System*). Algoritma kompensasi kemiringan kompas dilakukan pada proses dibawah ini:

```
1.  MadgwickQuaternionUpdate(-ay,-ax,az,gy*PI/180.0f,
    gx*PI/180.0f,-gz*PI/180.0f,mx,my,mz);
2.  yaw   = atan2(2.0f*(q[1]*q[2]+q[0]*q[3]),q[0]*q[0]
            +q[1]*q[1]-q[2]*q[2]-q[3]*q[3]);
3.  pitch = -asin(2.0f*(q[1]*q[3]-q[0]*q[2]));
4.  roll  = atan2(2.0f*(q[0]*q[1]+q[2]*q[3]),q[0]*q[0]
            -q[1]*q[1]-q[2]*q[2]+q[3]*q[3]);
5.
6.  pitch *= 180.0f / PI;
7.  yaw   *= 180.0f / PI;
8.  yaw   -= 1.3f; // Declination at Surabaya
9.  roll  *= 180.0f / PI;
10.
11. heading = yaw;
12. heading -= 180;
13.
14. while (heading < -180) heading += 360;
15. while (heading >  180) heading -= 360;
```

Gambar 4. 7 Pengujian Kompas



Gambar 4. 8 Output dari algoritma AHRS



Gambar 4. 9 Galat algoritma AHRS

Terlihat bahwa galat(kekeliruan) dari pembacaan arah sensor relatif kecil setelah algoritma mencapai *steady state*. Dari 200 data yang didapatkan, maksimum galat yang terukur adalah 6,41%. Terbukti bahwa filter kuaternion AHRS dapat meminimalisasi galat pembacaan arah dari sensor.

4.3 Pengujian Pindai Pergerakan Lengan Pengguna

Pengujian pindai pergerakan lengan dilakukan dengan memindai pergerakan pengguna dengan *feedback* dari sensor IMU. Metode yang dilakukan untuk mendapatkan posisi perpindahan adalah dengan metode *forward kinematic*. Dibawah ini adalah hasil perpindahan lengan

Tabel 4. 1 Pengujian hasil pindai pergerakan lengan pengguna

Posisi Sebenarnya			*Forward Kinematic*			Error
X (cm)	Y (cm)	Z (cm)	X (cm)	Y (cm)	Z (cm)	%
46	0	0	46.998	0.024	-0.150	2.17
44	6	2	43.812	5.911	1.789	2.5
40	10	7	40.022	9.888	7.911	2.2
35	16	-10	35.551	16.012	-10.90	1.6
34	15	-15	34.651	15.018	-15.88	1.9
37	4	-3	36.448	3.774	-3.31	1.5

Tabel diatas merupakan cuplikan data hasil pindai perhitungan posisi lengan pengguna yang sedang melakukan pergerakan. Tabel

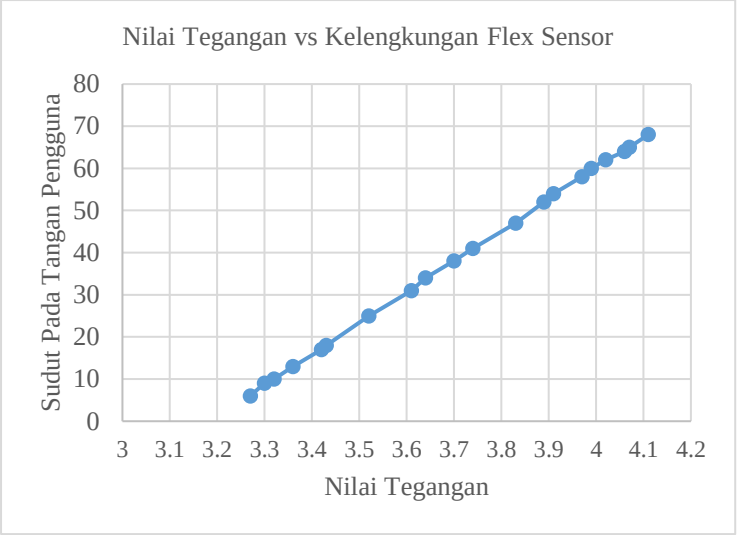
dibuat memiliki jangkauan yang jauh agar besar error dapat terlihat. Error pada tabel adalah error rata-rata dari semua sumbu x,y, dan z hasil perhitungan *forward kinematic*.

4.4 Pengujian Pindai Pergerakan Tangan Pengguna

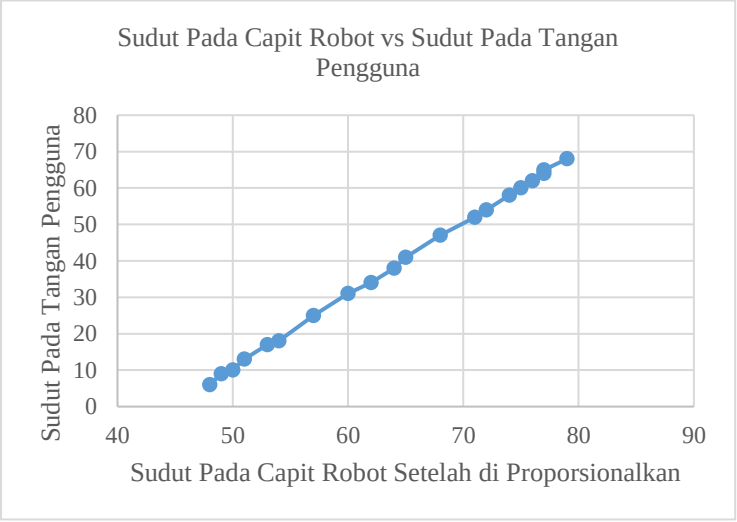
Pengujian pindai pergerakan tangan manusia dilakukan untuk mendapatkan pergerakan tangan saat kondisi menggenggam dan membuka. Pengujian dilakukan bersamaan dengan memindai pergerakan lengan pengguna. Dibawah ini adalah hasil dari pindai pergerakan tangan pengguna.

Tabel 4. 2 Data yang ditunjukkan oleh sensor flex

Nilai ADC	Tegangan (Volt)	Sudut di Tangan	Sudut Capit Robot
669	3.27	6	48
675	3.3	9	49
679	3.32	10	50
688	3.36	13	51
699	3.42	17	53
702	3.43	18	54
721	3.52	25	57
738	3.61	31	60
745	3.64	34	62
757	3.7	38	64
766	3.74	41	65
783	3.83	47	68
795	3.89	52	71
800	3.91	54	72
812	3.97	58	74
817	3.99	60	75
823	4.02	62	76
830	4.06	64	77
832	4.07	65	77
841	4.11	68	79



Gambar 4. 10 Tegangan banding kelengkungan flex sensor



Gambar 4. 11 Perbandingan sudut

Dari hasil pengujian beberapa karakteristik dari sensor flex adalah sebagai berikut:

1. Kelengkungan sensor flex yang digunakan hanya berkerja pada satu arah
2. Semakin besar sudut kelengkungan dari sensor flex (kondisi tangan mengepal) maka tegangan pembacaan pada sensor flex akan semakin meningkat.
3. Nilai resistansi dari sensor flex semakin berkurang saat kelengkungan dari sensor flex semakin bertambah

4.5 Pengujian Pergerakan Robot Dengan *Inverse Kinematic*

Pengujian *Inverse Kinematic* Robot bertujuan untuk mengetahui besar error pergerakan lengan robot berdasarkan pengolahan *inverse kinematic*. Perumusan *inverse kinematic* robot sesuai persamaan pada persamaan 3.9 hingga 3.14.

Pada pengujian ini, diberikan masukkan manual *inverse kinematic* robot berupa nilai posisi dalam derajat kartesian pada sumbu x, sumbu y, dan sumbu z.

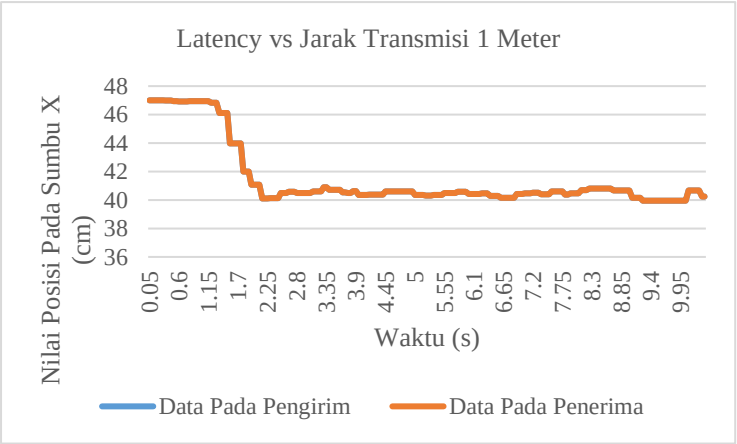
Tabel 4. 3 Pengujian perhitungan *inverse kinematic*

Posisi yang diinginkan			*Inverse Kinematic*			Error
X (cm)	Y (cm)	Z (cm)	X (cm)	Y (cm)	Z (cm)	%
35	0	0	32.40	0	0	7.4
35	0	2	32.40	0	1.8	8.7
37	0	2	35.15	0.13	1.91	10.5
37	2	2	35.55	2.14	1.84	12.3
37	2	0	35.15	2.22	0.1	11%

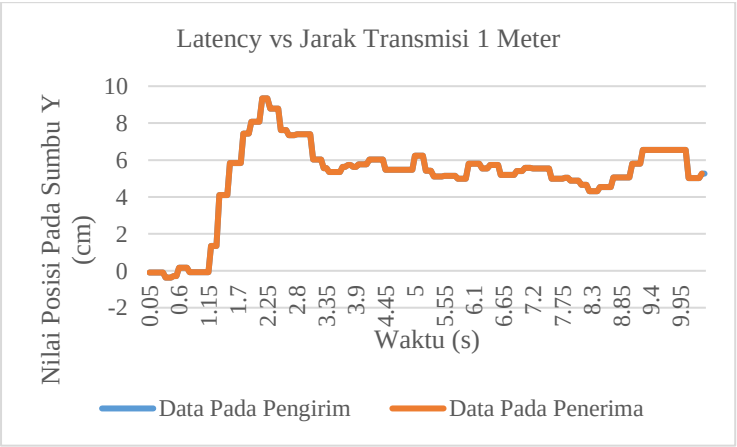
4.6 Pengujian *Time Latency* Terhadap Jarak Transmisi Pada Sinkronisasi Pergerakan

Pada pengujian ini, informasi *time latency* dibutuhkan untuk mengetahui besar tingkat kesalahan akibat transmisi jarak jauh dengan telemetri. Proses pengujian ini dilakukan dengan beberapa tahap jarak yaitu 1 meter, 10 meter, 25 meter, 50 meter, dan 100 meter. Untuk mengetahui besar *time latency* tersebut, proses pengiriman dan penerimaan data masing-masing menggunakan *interrupt timer* pada mikrokontroler agar dapat mengatur waktu

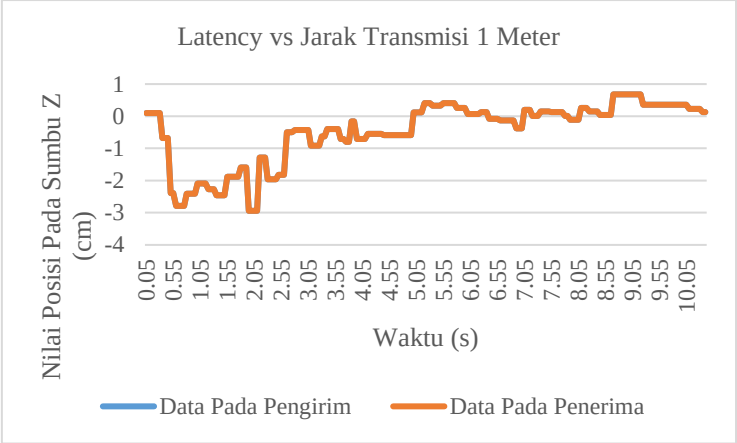
sampling. Setiap iterasi data yang terkirim memiliki besaran waktu yang sama dengan waktu sampling yang telah ditentukan.



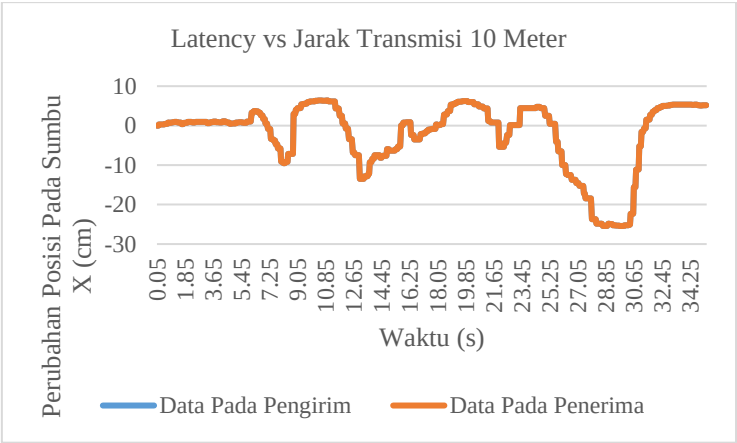
Gambar 4. 12 Tidak terjadi *latency* jarak 1 meter



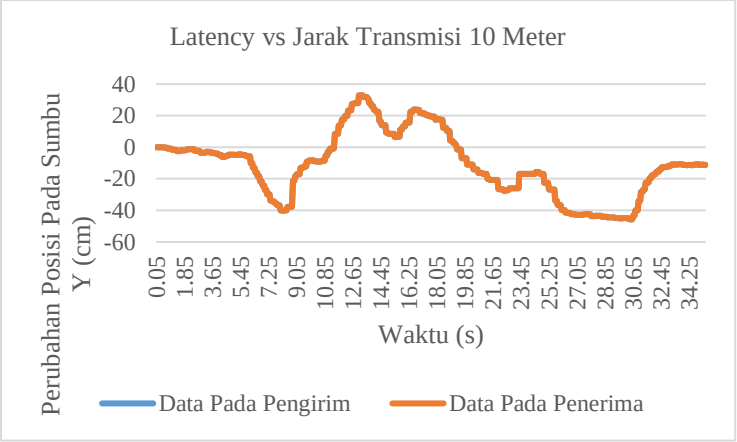
Gambar 4. 13 Tidak terjadi *latency* jarak 1 meter



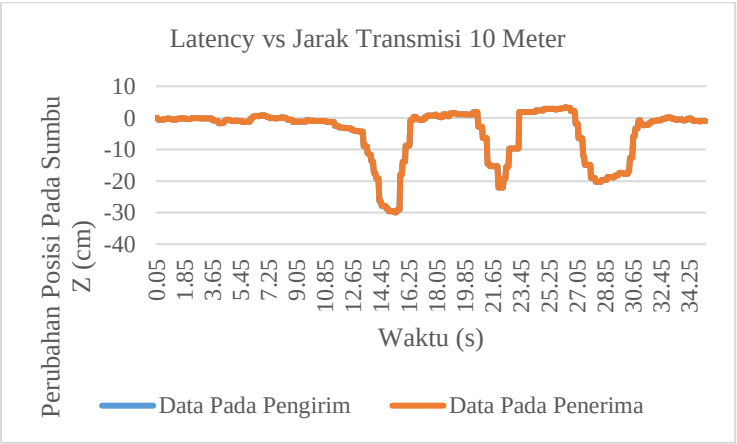
Gambar 4. 14 Tidak terjadi *latency* jarak 1 meter



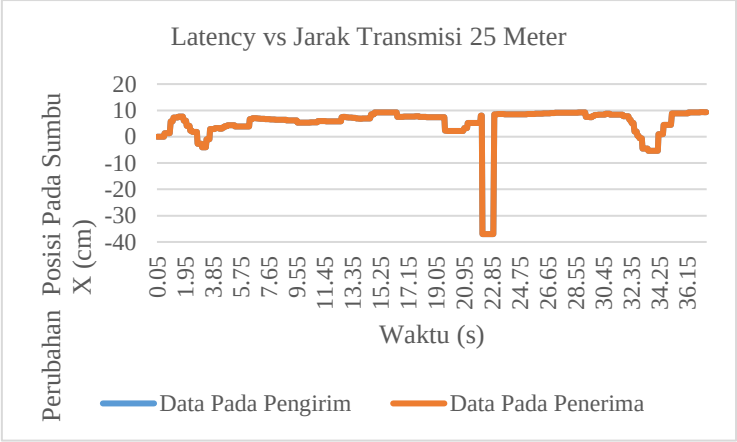
Gambar 4. 15 Tidak terjadi *latency* jarak 10 meter



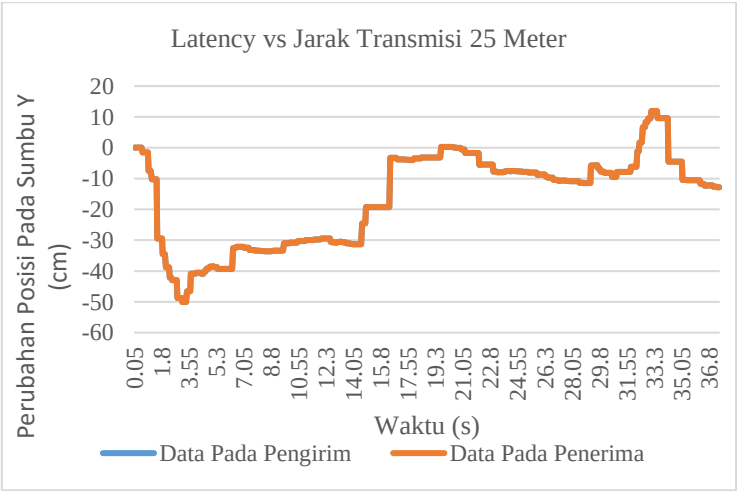
Gambar 4. 16 Tidak terjadi *latency* jarak 10 meter



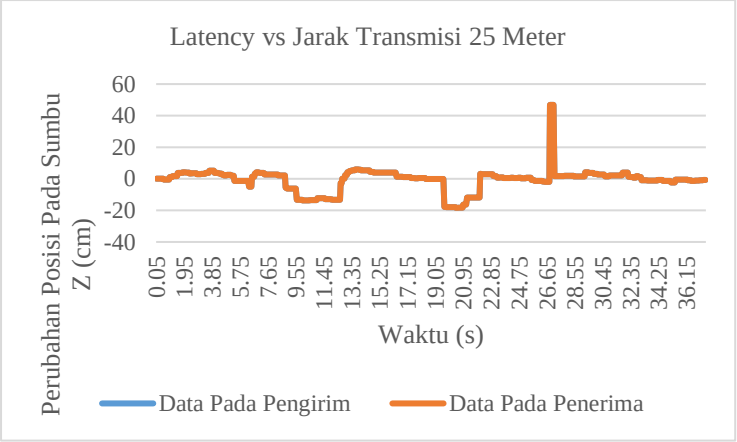
Gambar 4. 17 Tidak terjadi *latency* jarak 10 meter



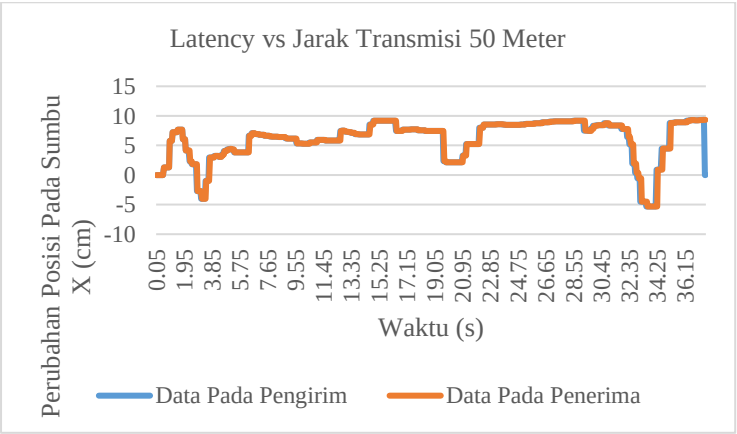
Gambar 4. 18 Tidak terjadi *latency* jarak 25 meter



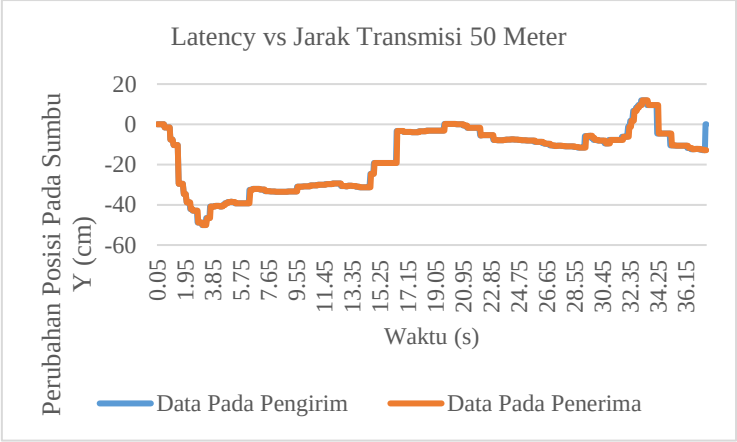
Gambar 4. 19 Tidak terjadi *latency* jarak 25 meter



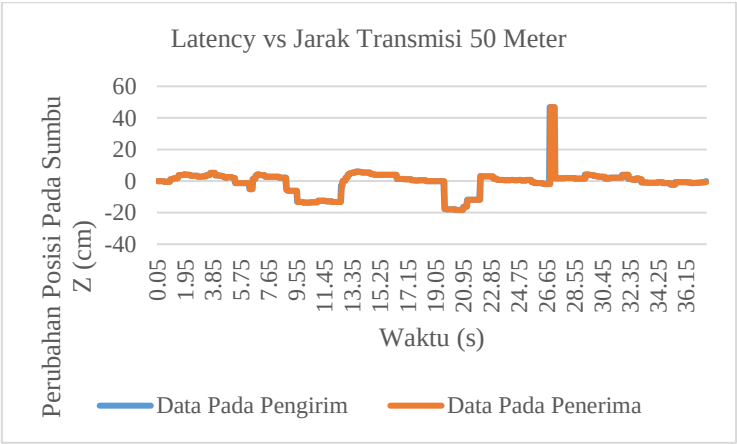
Gambar 4. 20 Tidak terjadi *latency* jarak 25



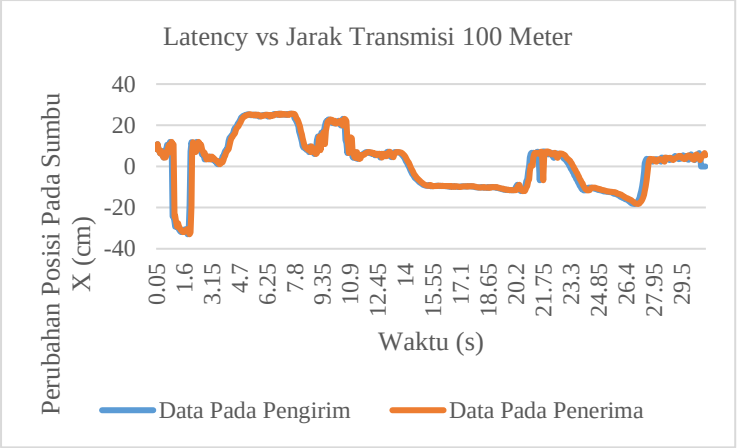
Gambar 4. 21 Jarak 50 meter terjadi *latency* 0.1 detik



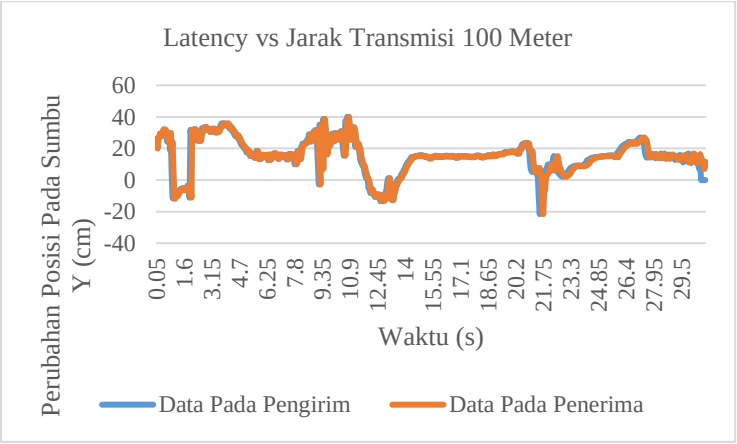
Gambar 4. 22 Jarak 50 meter terjadi *latency* 0.1 detik



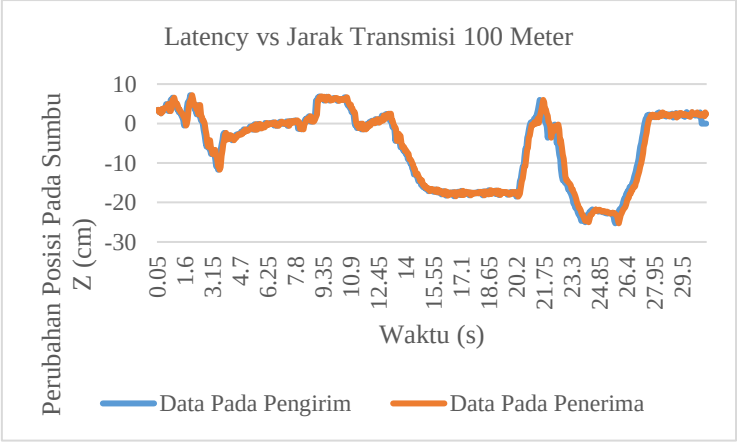
Gambar 4. 23 Jarak 50 meter terjadi *latency* 0.1 detik



Gambar 4. 24 Jarak 100 meter terjadi *latency* 0.2 detik



Gambar 4. 25 Jarak 100 meter terjadi *latency* 0.2 detik



Gambar 4. 26 Jarak 100 meter terjadi *latency* 0.2 detik

Dari gambar grafik diatas diketahui bahwa, sistem dapat berjalan baik hingga jarak kendali hingga 100 meter. Untuk pengendalian robot pada jarak 0 hingga 25 meter tidak terjadi *latency*. Pengendalian robot pada jarak 50 hingga 100 meter terjadi latency 0.1 hingga 0.2 detik. *Time sampling* pada penerima dan pengirim dibuat sama, yaitu 50 ms.

4.7 Pengujian *Time Latency* Antara Pergerakan Lengan Pengguna Sensor dan Pergerakan Lengan Robot

Pada tahap ini, pengujian dilakukan untuk mengetahui perbedaan waktu antara pergerakan lengan robot dengan pergerakan lengan pengguna. Sehingga pengujian berguna untuk mengetahui kemampuan lengan robot dalam mengikuti pergerakan lengan penggunanya. Pengujian dilakukan dengan beberapa sampel perpindahan jarak lalu dihitung waktu pergerakan lengan pengguna sensor dan lengan robot dengan menggunakan 2 *stopwatch*. Dengan perbandingan jarak dan waktu akan diketahui kualitas dari alat pada tugas akhir ini. Semakin kecil perbedaan waktu maka kualitas dari alat semakin baik. Dibawah ini adalah tabel hasil pengujian

Tabel 4. 4 Pengujian *time latency* pada sumbu x

Perpindahan lengan pengguna pada sumbu x		Perpindahan lengan robot pada sumbu x		Selisih waktu (detik)
Panjang (cm)	Waktu (detik)	Panjang (cm)	Waktu (detik)	
0 sampai 3	1,81	0 sampai 2,8	2,16	0.35
0 sampai 5	2,17	0 sampai 4,6	2,64	0.47
0 sampai 10	3.40	0 sampai 9, 5	4,19	0.79
0 sampai 13	3.7	0 sampai 12,3	4,55	0.85
0 sampai -5	2,23	0 sampai -4,7	2,51	0.21
0 sampai -8	2,94	0 sampai -7,4	3,38	0.44
Rata- rata selisih waktu				0.518

Tabel 4. 5 Pengujian *time latency* pada sumbu y

Perpindahan lengan pengguna pada sumbu y		Perpindahan lengan robot pada sumbu y		Selisih waktu (detik)
Panjang (cm)	Waktu (detik)	Panjang (cm)	Waktu (detik)	
0 sampai 3	1,79	0 sampai 2,7	2,03	0.24
0 sampai 5	2,01	0 sampai 4,8	2,51	0.5
0 sampai 10	3.23	0 sampai 9, 5	3,92	0.69
0 sampai 13	3.81	0 sampai 12,6	4,65	0.84
0 sampai -5	2,14	0 sampai -4,5	2,51	0.37
0 sampai -10	3,29	0 sampai -9,5	3,81	0.52
Rata- rata selisih waktu				0.527

Tabel 4. 6 Pengujian *time latency* pada sumbu z

Perpindahan lengan pengguna pada sumbu z		Perpindahan lengan robot pada sumbu z		Selisih waktu (detik)
Panjang (cm)	Waktu (detik)	Panjang (cm)	Waktu (detik)	
0 sampai 3	1,94	0 sampai 2,7	2,36	0.42
0 sampai 5	2,32	0 sampai 4,8	2,84	0.52
0 sampai 7	2,90	0 sampai 6,5	3,38	0.48
0 sampai -3	1.78	0 sampai -2,3	2,15	0.37
0 sampai -5	2,23	0 sampai -4,7	2,81	0.58
0 sampai -8	2,94	0 sampai -7,4	3,38	0.44
Rata- rata selisih waktu				0.468

Pada tabel 4.4 hingga tabel 4.6 menunjukkan selisih waktu berbanding lurus dengan jarak tempuh dari pergerakan. Hal ini dikarenakan terdapatnya *delay* dalam perhitungan yang memang sengaja diberikan agar perhitungan nilai dari data sensor dapat dilakukan dan tidak menumpuk. *Delay* pada program diberikan 10 ms untuk setiap data yang masuk dalam fungsi perhitungan *inverse kinematic*.

Halaman ini sengaja dikosongkan

BAB V
PENUTUP

5.1 Kesimpulan

Kesimpulan yang dapat diperoleh dari tugas akhir ini adalah:

1. Terdapat error rata-rata perhitungan untuk mengetahui besar posisi lengan manusia dengan metode *forward kinematic* sebesar kurang dari 3%
2. Error terbesar perhitungan untuk menggerakkan robot dengan metode *inverse kinematic* sebesar 12%
3. Kenaikan besar tegangan pada sensor flex berbanding lurus secara linear bertambahnya sudut kelengkungannya.
4. Terjadi *latency* saat transmisi data robot pada jarak 50 meter hingga 100 meter sebesari 0.1 detik hingga 0.2 detik.
5. Perbedaan waktu saat pengguna menggerakkan tangan dan lengan robot melakukan pergerakan rata-rata sebesar 0.4 detik.
6. Hasil pengujian menunjukkan banyak aspek pada robot sudah sesuai spesifikasi.

5.2 Saran

Adapun untuk perbaikan dan pengembangan alat kedepannya diantaranya adalah perlu perbaikan sistem mekanik, agar pergerakan lengan robot tidak terbatasi pada sudut tertentu sehingga lebih leluasa dalam bermanuver.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Lembaga Ilmu Pengetahuan Indonesia, morolipi v2 robot penjinak bom, Nopember, 2013. Available : http://lipi.go.id/risetunggulan/single/morolipi-v2-robot-penjinak-bom/6 [Diakses 12 Desember 2016]

[2] R M Crowder, "Automation and Robotics, Robot Kinematics", 10 Desember 2013 [Online]. Available: http://www.southampton.ac.uk/~rmc1/robotics/arkinematics.htm [Diakses 12 Desember 2016]

[3] K. H. Rob O'Reilly, "Sonic Nirvana: MEMS Accelerometers as Acoustic Pickups in Musical Instruments," Analog Devices Inc, 1 Juni 2009.[Online]. Available:http://www.sensorsmag.com/sensors/acceleration-vibration/sonicnirvana-mems-accelerometers-acoustic-pickups-musical-i-5852. [Diakses 12 Desember 2016].

[4] W. Storr, "Hall Effect Sensor and How Magnets Make It Works," Electronics-tutorials.ws, 14 April 2016. [Online]. Available: http://www.electronics-tutorials.ws/electromagnetism/halleffect.html. [Diakses 12 Desember 2016].

[5] W. R. Hamilton, Elements of Quaternions, Longmans: Green & Company, 1866.

[6] P. Hapala, "Quaternion," Wikimedia Foundation, Inc., 11 November 2007. [Online]. Available: https://en.wikipedia.org/wiki/Quaternion. [Diakses 12 Desember 2016]

[7] L. Brits, "Conversion between quaternions and Euler angles," Wikimedia Foundation, Inc, 11 November 2007. [Online]. Available:64https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles. [Diakses 12 Desember 2016].

[8] C. Chamberlain, "Understanding Euler Angles," CH Robotics LLC, 06 Juni 2009. [Online]. Available: http://www.chrobotics.com/library/understanding-euler-angles. [Diakses 12 Desember 2016].

[9] S. O. H. Madgwick, A. J. L. Harrison and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient

descent algorithm," in *2011 IEEE International Conference on Rehabilitation Robotics*, Zurich, 2011.

[10] Datasheet, *flex sensor* sparkfun, [Online]. Available: https://www.sparkfun.com/.../Sensors/Flex/FlexSensor.pdf [Diakses 12 Desember 2016].

[11] Ozzmaker, "BerryIMU-accelerometer, gyroscope, magnetometer," Tindie, Inc., 11 April 2014. [Online]. Available: https://www.tindie.com/products/ozzmaker/berryimuaccelerometer-gyroscope-magnetometer/. [Diakses 12 Desember 2016].

[**12**] Y. Pititeeraphab , P. Choitkunnan, N. Thongpance, K. Kullathum, Ch. Pintavirooj, "Robot-arm control system using LEAP motion controller" IEEE xplore., 15 december 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7782091/

LAMPIRAN

Program algoritma kuaternion Magdwick:

```
1.  void MadgwickQuaternionUpdate(float ax, float ay
    , float az, float gx, float gy, float gz, float
    mx, float my, float mz)
2.  {
3.      float q1 = q[0], q2 = q[1], q3 = q[2], q4 =
    q[3];
4.      float norm;
5.      float hx, hy, _2bx, _2bz;
6.      float s1, s2, s3, s4;
7.      float qDot1, qDot2, qDot3, qDot4;
8.
9.      float _2q1mx;
10.     float _2q1my;
11.     float _2q1mz;
12.     float _2q2mx;
13.     float _4bx;
14.     float _4bz;
15.     float _2q1 = 2.0f * q1;
16.     float _2q2 = 2.0f * q2;
17.     float _2q3 = 2.0f * q3;
18.     float _2q4 = 2.0f * q4;
19.     float _2q1q3 = 2.0f * q1 * q3;
20.     float _2q3q4 = 2.0f * q3 * q4;
21.     float q1q1 = q1 * q1;
22.     float q1q2 = q1 * q2;
23.     float q1q3 = q1 * q3;
24.     float q1q4 = q1 * q4;
25.     float q2q2 = q2 * q2;
26.     float q2q3 = q2 * q3;
27.     float q2q4 = q2 * q4;
28.     float q3q3 = q3 * q3;
29.     float q3q4 = q3 * q4;
30.     float q4q4 = q4 * q4;
31.
32.     norm = sqrt(ax * ax + ay * ay + az * az);
33.     if (norm == 0.0f) return; // handle NaN
34.     norm = 1.0f/norm;
35.     ax *= norm;
```

```
36.     ay *= norm;
37.     az *= norm;
38.
39.     norm = sqrt(mx * mx + my * my + mz * mz);
40.     if (norm == 0.0f) return; // handle NaN
41.     norm = 1.0f/norm;
42.     mx *= norm;
43.     my *= norm;
44.     mz *= norm;
45.
46.     _2q1mx = 2.0f * q1 * mx;
47.     _2q1my = 2.0f * q1 * my;
48.     _2q1mz = 2.0f * q1 * mz;
49.     _2q2mx = 2.0f * q2 * mx;
50.     hx = mx * q1q1 - _2q1my * q4 + _2q1mz * q3 +
             mx * q2q2 + _2q2 * my * q3 + _2q2 * mz
             * q4 - mx * q3q3 - mx * q4q4;
51.     hy = _2q1mx * q4 + my * q1q1 - _2q1mz * q2 +
             _2q2mx * q3 - my * q2q2 + my * q3q3 +
             _2q3 * mz * q4 - my * q4q4;
52.     _2bx = sqrt(hx * hx + hy * hy);
53.     _2bz = -
             _2q1mx * q3 + _2q1my * q2 + mz * q1q1 +
             _2q2mx * q4 - mz * q2q2 + _2q3 * my *
             q4 - mz * q3q3 + mz * q4q4;
54.     _4bx = 2.0f * _2bx;
55.     _4bz = 2.0f * _2bz;
56.
57.     s1 = -
             _2q3 * (2.0f * q2q4 - _2q1q3 - ax) + _2
             q2 * (2.0f * q1q2 + _2q3q4 - ay) - _2bz
              * q3 * (_2bx * (0.5f - q3q3 - q4q4) +
             _2bz * (q2q4 - q1q3) - mx) + (-
             _2bx * q4 + _2bz * q2) * (_2bx * (q2q3
             - q1q4) + _2bz * (q1q2 + q3q4) - my) +
             _2bx * q3 * (_2bx * (q1q3 + q2q4) + _2b
             z * (0.5f - q2q2 - q3q3) - mz);
58.     s2 = _2q4 * (2.0f * q2q4 - _2q1q3 - ax) + _2
             q1 * (2.0f * q1q2 + _2q3q4 - ay) - 4.0f
              * q2 * (1.0f - 2.0f * q2q2 - 2.0f * q3
             q3 - az) + _2bz * q4 * (_2bx * (0.5f -
             q3q3 - q4q4) + _2bz * (q2q4 - q1q3) - m
```

```
            x) + (_2bx * q3 + _2bz * q1) * (_2bx *
            (q2q3 - q1q4) + _2bz * (q1q2 + q3q4) -
            my) + (_2bx * q4 - _4bz * q2) * (_2bx *
             (q1q3 + q2q4) + _2bz * (0.5f - q2q2 -
            q3q3) - mz);
59.     s3 = -
            _2q1 * (2.0f * q2q4 - _2q1q3 - ax) + _2
            q4 * (2.0f * q1q2 + _2q3q4 - ay) - 4.0f
             * q3 * (1.0f - 2.0f * q2q2 - 2.0f * q3
            q3 - az) + (-
            _4bx * q3 - _2bz * q1) * (_2bx * (0.5f
            - q3q3 - q4q4) + _2bz * (q2q4 - q1q3) -
             mx) + (_2bx * q2 + _2bz * q4) * (_2bx
            * (q2q3 - q1q4) + _2bz * (q1q2 + q3q4)
            - my) + (_2bx * q1 - _4bz * q3) * (_2bx
             * (q1q3 + q2q4) + _2bz * (0.5f - q2q2
            - q3q3) - mz);
60.     s4 = _2q2 * (2.0f * q2q4 - _2q1q3 - ax) + _2
            q3 * (2.0f * q1q2 + _2q3q4 - ay) + (-
            _4bx * q4 + _2bz * q2) * (_2bx * (0.5f
            - q3q3 - q4q4) + _2bz * (q2q4 - q1q3) -
             mx) + (-
            _2bx * q1 + _2bz * q3) * (_2bx * (q2q3
            - q1q4) + _2bz * (q1q2 + q3q4) - my) +
            _2bx * q2 * (_2bx * (q1q3 + q2q4) + _2b
            z * (0.5f - q2q2 - q3q3) - mz);
61.     norm = sqrt(s1 * s1 + s2 * s2 + s3 * s3 + s4
    * s4);
62.     norm = 1.0f/norm;
63.     s1 *= norm;
64.     s2 *= norm;
65.     s3 *= norm;
66.     s4 *= norm;
67.
68.     qDot1 = 0.5f * (-
   q2 * gx - q3 * gy - q4 * gz) - beta * s1;
69.     qDot2 = 0.5f * (q1 * gx + q3 * gz - q4 * gy)
    - beta * s2;
70.     qDot3 = 0.5f * (q1 * gy - q2 * gz + q4 * gx)
    - beta * s3;
71.     qDot4 = 0.5f * (q1 * gz + q2 * gy - q3 * gx)
    - beta * s4;
```

```
72.
73.     q1 += qDot1 * deltat;
74.     q2 += qDot2 * deltat;
75.     q3 += qDot3 * deltat;
76.     q4 += qDot4 * deltat;
77.     norm = sqrt(q1 * q1 + q2 * q2 + q3 * q3 + q4
    * q4);
78.     norm = 1.0f/norm;
79.     q[0] = q1 * norm;
80.     q[1] = q2 * norm;
81.     q[2] = q3 * norm;
82.     q[3] = q4 * norm;
83. }
```

Program komunikasi I2C multi perangkat

➔ Pengirim

```
1.  #define SLAVE1_I2C_ADDRESS 0x5D
2.  #define SLAVE2_I2C_ADDRESS 0x5A
3.  #define SLAVE3_I2C_ADDRESS 0x5B
4.
5.  enum {
6.    IMU_NONE,
7.    IMU_COUNT,
8.    IMU_ANGLE,
9.    IMU_NONE2,
10.   IMU_COUNT2,
11.   IMU_ANGLE2,
12.   IMU_NONE3,
13.   IMU_COUNT3,
14.   IMU_ANGLE3
15. };
16.
17. struct IMUInfo {
18.   unsigned long readCount;
19.   unsigned long validCount;
20.   float yaw;
21.   float pitch;
```

```
  float roll;
  unsigned long readCount2;
  unsigned long validCount2;
  float yaw2;
  float pitch2;
  float roll2;
  unsigned long readCount3;
  unsigned long validCount3;
  float yaw3;
  float pitch3;
  float roll3;
};

IMUInfo imuInfo;

void i2c_slave_read_handler()
{
  switch (requestedInfo) {
  case IMU_COUNT:
    Wire.write((uint8_t*)&imuInfo.readCount, 8);
    break;
  case IMU_ANGLE:
    Wire.write((uint8_t*)&imuInfo.px, 16);
    break;
  }
}

void i2c_setup()
{
  Wire.begin(SLAVE1_I2C_ADDRESS); //tergantung
address yang digunakan
  Wire.onReceive(i2c_slave_write_handler);
  Wire.onRequest(i2c_slave_read_handler);
}
```

➔ Penerima

```
float readLongFromI2C() {
  unsigned long tmp = 0;

  for (int i = 0; i < 4; i++) {
    unsigned long tmp2 = Wire.read();
```

```
6.      tmp |= tmp2 << (i*8);
7.    }
8.
9.    return tmp;
10. }
11.
12. float readFloatFromI2C() {
13.
14.   float f = 0;
15.   byte* p = (byte*)&f;
16.
17.   for (int i = 0; i < 4; i++)
18.     p[i] = Wire.read();
19.
20.   return f;
21. }
22.
23. void i2c_loop()
24. {
25.   static unsigned long lastIMURead = 0;
26.   static unsigned long lastReadCount = 0;
27.   unsigned long now = millis();
28.
29.   if ( now - lastIMURead > 20 ) {
30.   lastIMURead = now;
31.
32.   static int delaykonstan = 5;
33.
34.   Wire.beginTransmission(SLAVE1_I2C_ADDRESS);
35.   Wire.write(IMU_COUNT);
36.   Wire.endTransmission();
37.   delay(delaykonstan);
38.
39.   Wire.requestFrom(SLAVE1_I2C_ADDRESS, 8);
40.   if (Wire.available() >= 8) {
41.     imuInfo.readCount = readLongFromI2C();
42.     imuInfo.validCount = readLongFromI2C();
43.   }
44.   else
45.    Serial.println("SLAVE 1 SALAH ADDRESS");
46.
47.   if ( imuInfo.readCount == lastReadCount )
```

```
48.     return;
49.   lastReadCount = imuInfo.readCount;
50.
51.   Wire.beginTransmission(SLAVE1_I2C_ADDRESS);
52.   Wire.write(IMU_ANGLE);
53.   Wire.endTransmission();
54.   delay(delaykonstan);
55.
56.   Wire.requestFrom(SLAVE1_I2C_ADDRESS, 16);
57.   if (Wire.available() >= 16)
58.   {
59.   imuInfo.yaw   = readFloatFromI2C();
60.   imuInfo.pitch = readFloatFromI2C();
61.   imuInfo.roll  = readFloatFromI2C();
62.   }
63.   else
64.   Serial.println("SLAVE 1 DATA TIDAK SAMPAI");
65.
66.   Wire.beginTransmission(SLAVE2_I2C_ADDRESS);
67.   Wire.write(IMU_COUNT2);
68.   Wire.endTransmission();
69.   delay(delaykonstan);
70.
71.   Wire.requestFrom(SLAVE2_I2C_ADDRESS, 8);
72.   if (Wire.available() >= 8) {
73.     imuInfo.readCount2 = readLongFromI2C();
74.     imuInfo.validCount2 = readLongFromI2C();
75.   }
76.   else
77.    Serial.println("SLAVE 2 SALAH ADDRESS");
78.
79.   if ( imuInfo.readCount2 == lastReadCount )
80.     return;
81.   lastReadCount = imuInfo.readCount2;
82.
83.   Wire.beginTransmission(SLAVE2_I2C_ADDRESS);
84.   Wire.write(IMU_ANGLE2);
85.   Wire.endTransmission();
86.   delay(delaykonstan);
87.
88.   Wire.requestFrom(SLAVE2_I2C_ADDRESS, 16);
89.   if (Wire.available() >= 16)
```

```
90.   {
91.   imuInfo.yaw2   = readFloatFromI2C();
92.   imuInfo.pitch2 = readFloatFromI2C();
93.   imuInfo.roll2  = readFloatFromI2C();
94.   }
95.   else
96.   Serial.println("SLAVE 2 DATA TIDAK SAMPAI");
97.
98.   Wire.beginTransmission(SLAVE3_I2C_ADDRESS);
99.   Wire.write(IMU_COUNT3);
100.  Wire.endTransmission();
101.   delay(delaykonstan);
102.
103.   Wire.requestFrom(SLAVE3_I2C_ADDRESS, 8);
104.   if (Wire.available() >= 8) {
105.     imuInfo.readCount3 = readLongFromI2C();
106.     imuInfo.validCount3 = readLongFromI2C();
107.   }
108.   else
109.    Serial.println("SLAVE 3 SALAH ADDRESS");
110.
111.   if ( imuInfo.readCount3 == lastReadCount )
112.     return;
113.   lastReadCount = imuInfo.readCount3;
114.
115.   Wire.beginTransmission(SLAVE3_I2C_ADDRESS);
116.   Wire.write(IMU_ANGLE3);
117.   Wire.endTransmission();
118.   delay(delaykonstan);
119.
120.   Wire.requestFrom(SLAVE3_I2C_ADDRESS, 16);
121.   if (Wire.available() >= 16)
122.   {
123.   imuInfo.yaw3   = readFloatFromI2C();
124.   imuInfo.pitch3 = readFloatFromI2C();
125.   imuInfo.roll3  = readFloatFromI2C();
126.   }
127.   else
128.   Serial.println("SLAVE 3 DATA TIDAK SAMPAI");
129.   }
130. }
131.
```

Program rumus *forward kinematic*

```
1.  void forward_kinematic2(double yaw1, double
    pitch1, double roll1, double yaw2, double
    pitch2, double roll2, double yaw3, double
    pitch3, double roll3)
2.  {
3.    yaw1 *= PI / 180;
4.    yaw2 *= PI / 180;
5.    yaw3 *= PI / 180;
6.    roll1 *= PI / 180;
7.    roll2 *= PI / 180;
8.    roll3 *= PI / 180;
9.    pitch1 *= PI / 180;
10.   pitch2 *= PI / 180;
11.   pitch3 *= PI / 180;
12.
13.   px = l1*cos(pitch1)*cos(yaw1) +
    l2*cos(pitch2)*cos(yaw2);
14.   py = l1*cos(pitch1)*sin(yaw1) +
    l2*cos(pitch2)*sin(yaw2);
15.   pz = l1*sin(pitch1) + l2*sin(pitch2);
16. }
```

Program rumus *inverse kinematic*

```
1.  void inverse_kinematic(double x, double y,
    double z)
2.  {
3.    double theta1;
4.    double theta2;
5.
6.    a = sqrt(x*x + y*y);
7.    c = sqrt((a-l1)*(a-l1) + z*z);
8.
9.    alfa0 = radKeDerajat(acos((c*c - l2*l2 -
    l3*l3) / (2*l2*l3)));
10.
```

```
11.   theta1 = radKeDerajat(acos((l2*l2 + c*c -
    l3*l3) /(2*l2*c)));
12.   theta2 = radKeDerajat(atan(z/(a-l1)));
13.
14.   alfa1 = theta1 + theta2;
15.
16.   alfa2 = radKeDerajat(atan(y/x));
17.  }
```








BIODATA PENULIS

Mochamad Fajar Rinaldi Utomo, lahir di Sidoarjo, 28 November 1994. Penulis memulai jenjang pendidikan di sekolah dasar di SDI Bakti Ibu pada tahun 2000 hingga 2006. Penulis melanjutkan pendidikan tingkat menengah di SMP Negeri 41 Jakarta pada tahun 2006 hingga 2009. Kemudian penulis melanjutkan jenjang pendidikan di SMA Negeri 28 Jakarta pada tahun 2009 hingga 2012. Pada tahun 2012, penulis memulai pendidikan strata I di jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Selama kuliah, penulis aktif mengikuti kegiatan perlombaan robotika nasional dan aktif menjadi asisten laboratorium elektronika dasar.

Email : mfajarrinaldi@yahoo.co.id

Halaman ini sengaja dikosongkan