MOTION CAPTURE SYSTEM BERBASIS *FLEX SENSOR* DAN *INERTIAL MEASUREMENT UNIT* UNTUK GERAKAN PERGELANGAN TANGAN ROBOT

SKRIPSI

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

SIROJUL HADI
NIM. 125060301111048

UNIVERSITAS BRAWIJAYA
FAKULTAS TEKNIK
MALANG
2016

LEMBAR PERSETUJUAN

MOTION CAPTURE SYSTEM BERBASIS *FLEX SENSOR* DAN *INERTIALMEASUREMENT UNIT* UNTUK GERAKAN PERGELANGAN TANGAN ROBOT

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI ELEKTRONIKA

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

SIROJUL HADI
NIM. 125060301111048

Skripsi ini telah direvisi dan disetujui oleh dosen pembimbing
Pada tanggal 18 Maret 2016

Dosen Pembimbing I,	Dosen Pembimbing II,
Dr.Eng. Panca Mudjirahardjo., S.T., M.T	Ir. Nanang Sulistyanto, M.T.
NIP. 19700329 200012 1 001	NIP. 19700113 199403 1 002

RINGKASAN

Sirojul Hadi, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya, maret 2016, Motion Capture System Berbasis Flex Sensor dan Inertial Measurement Unit untuk Gerakan Pergelangan Tangan Robot, Dosen Pembimbing: Panca Mudjirahrdjo dan Nanang Sulistyanto.

Robot digunakan sebagai pengganti tugas manusia yang dapat membantu memudahkan pekerjaan manusia. Idealnya sebuah robot memiliki daya tahan yang baik terhadap lingkungan, memiliki fleksibilitas yang tinggi, dan diharapakn memiliki kecerdasan buatan. Robot dapat digunakan pada beberapa bidang yang sulit atau berbahaya dilakukan oleh manusia, seperti suatu ruangan yang memiliki tingkat radiasi tinggi, tempat yang memiliki limbah beracun, tempat yang memiliki suhu yang sangat tinggi atau sangat rendah, dan keadaan dimana diharuskan memindahkan benda-benda yang sangat berat, maka tidak dimungkinkan pekerjaan dilakukan oleh manusia sehingga dibutuhkan robot. Perlu adanya robot yang dapat di kontrol oleh pengguna dan dapat melakukan gerakan sesuai keinginan pengguna. Salah satu metode pengontrolan yang dapat digunakan yaitu metode motion capture system berbasis flex sensor dan inertial measurement unit (IMU), flex sensor digunakan untuk merekam gerakan jari-jari tangan pengguna, dan sensor IMU digunakan untuk merekam sudut rotasi dari pergelangan tangan. Hasil perekaman dari tangan pengguna dikirim secara nirkabel kemudian diterjemahkan oleh robot menjadi gerakan pada sendi-sendi robot.

Kata kunci – Pergelangan dan jari-jari robot, Motion capture system, flex sensor, IMU.

SUMMARY

Sirojul Hadi,Department of Electrical Engineering, Faculty of Engineering University of Brawijaya, march 2016, Motion Capture System Based Flex Sensor and Inertial Measurement Unit for Movement Robot Wrist, Academic Supervisor : Panca Mudjirahrdjo dan Nanang Sulistyanto.

Robot is use as human replacement to help them ease their jobs.Ideally a robot has a good durability to environment, high flexibility and also artificial intelegent skill.Robot can be used to some difficult field or dangerous for human to do, for example a room or place which has high radiaton, toxic waste, low or high temperature and a condition where we have to move heavy stuff, So it is impossible for human to do the jobs and therefore the robot is needed. We need robots which can be controlled by user and can make moves as the user want.One of the controlling methode is motion capture system by flex censor and inertial measurement unit basis, flex censor is use to record finger moves of user, and censor IMU is use to record rotation corner of the wrist user. The result of recording from user hands will be send by nirkabel and then will be translate by robot to be moves on the joints of robot.

Keyword - wrist and fingers of robot, motion capture system, flex sensor, IMU

PENGANTAR

Alhamdulillah, puji syukur ke hadirat Allah SWT atas rahmat dan karunia-Nya dan perkenan-Nya penulis dapat menyelesaikan penulisan skripsi ini.

Karya ini tidak mungkin selesai tanpa restu dan dukungan dari berbagai pihak. Untuk itu penulis mengucapkan terima kasih sedalam-dalamnya yang tidak terhingga kepada:

1. Kedua orang tua penulis, Bapak Sahabudin dan Ibu Siti Maemanah atas dukungan, pengorbanan, motivasi dan doa restunya sehingga penulis dapat menuntut ilmu sampai jenjang sarjana. Serta, kakak penulis Nisyatul Fahmi dan adik penulis Herman Firdaus dan Sudawan Herdin Prmana Hadi atas doa, dukungan, dan motivasi yang tiada henti kepada penulis.
2. Bapak M. Aziz Muslim, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Elektro Universtias Brawijaya.
3. Bapak Hadi Suyono, S.T., M.T., Ph.D selaku Sekretaris Jurusan Teknik Elektro Universtias Brawijaya.
4. Bapak Ali Mustofa, S.T., M.T., selaku ketua Program Studi Sarjana Teknik Elektro Universtias Brawijaya.
5. Ibu Ir. Nurussa'adah, M.T. selaku Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya yang selalu memberi semangat dan motivasi untuk cepat menyelesaikan skripsi.
6. Bapak Dr.Eng. Panca Mudjirahardjo., S.T., M.T dan Bapak Ir. Nanang Sulistyanto, M.T., sebagai pembimbing pertama dan sebagai pembimbing kedua, ditengah kesibukan beliau berdua selalu memberikan waktu untuk diskusi dengan tulus, sabar memberikan masukan yang sungguh berharga.
7. Laboran laboraturium Komputasi dan Jaringan Bapak Nugroho Madiantoko., A.md. atas semua fasilitas dan bantuan yang telah di sediakan selama pengerjaan skripsi.
8. Para Dosen Pengajar Program Studi Teknik Elektro Universitas Brawijaya, yang tidak dapat penulis sebutkan satu per satu yang telah memberikan bekal ilmu pada penulis dalam menyelesaikan studi.

9. Teman-teman seperjuagan konsentrasi Teknik Elektronika angkatan 2012 atas segala dukungan dan bantuan dalam pengerjaan skripsi.
10. Kakak-kakak, teman-teman dan adik-adik asisten laboratorium Komputasi dan Jaringan, Mas Hitzba, Mas firman, Mas fahad, Embak Ayak, Embak Liza, mas Fahmi, mas Abdur, Akbar, Ikhfal, Riyan, Ain, Gladys, Ulya, Muslichin, Octa, Frido, Septi atas segala dukungan dalam pengerjaan skripsi.
11. Teman-teman Workshop Divisi Robotika, terimakasih atas pengalaman, semangat, dan kerjasama yang telah terjalin didalam organisasi.
12. Sahabat dan teman-teman, Riski Amalia, Hadi Abdurrahman, Fadly, Dodi, Gias, Abid, Idin, Muja terimakasih untuk segala diskusi, keceriaan, semangat, motivasi, dan dukungan dalam pengerjaan skripsi.

Sekiranya Allah SWT membalas kebaikan semua pihak yang turut membantu skripsi ini terselesaikan. Akhirnya, penulis menyadari bahwa skripsi ini masih jauh dari sempurna, namun semoga skripsi ini dapat memberikan manfaat bagi kita semua. Amin, Terima kasih.

Malang, 18 Maret 2016

Penulis

DAFTAR ISI

Halaman

UNIVERSITAS BRAWIJAYA

DAFTAR GAMBAR

No Judul Halaman

DAFTAR TABEL

No Judul Halaman

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Manusia merupakan makhluk hidup yang paling cerdas dimuka bumi, akan tetapi tubuh manusia tidak bisa melakukan pekerjaan secara terus menerus tanpa melakukan istirahat. Oleh karena itu, robot digunakan sebagai pengganti tugas manusia yang dapat membantu memudahkan pekerjaan manusia. Idealnya sebuah robot memiliki daya tahan yang baik terhadap lingkungan, memiliki fleksibilitas yang tinggi, dan diharapakn memiliki kecerdasan buatan yang dapat mempermudah pekerjaan manusia.

Pengaplikasian robot pada kehidupan sebenarnya didesain dengan bentuk yang sesuai dengan kebutuhan. Di bidang industri misalnya, robot yang digunakan untuk memindahkan barang pada medan yang rata didesain menggunakan roda untuk melakukan mobilitas, namun untuk medan yang tidak rata atau tidak beraturan menggunakan kaki, sedangkan robot yang digunakan untuk memindahkan barang dari *conveyor* didesain sebagai lengan dengan sensor-sensor dan tidak dilengkapi dengan roda atau kaki karena memang tidak membutuhkan mobilitas (Nalwan, 2012:6).

Robot yang memiliki teknologi tinggi digunakan pada beberapa bidang yang sulit atau berbahaya dilakukan oleh manusia, seperti suatu ruangan yang memiliki tingkat radiasi tinggi yang sangat berbahaya bagi kesehatan, tempat yang memiliki limbah beracun, tempat yang memiliki suhu yang sangat tinggi atau sangat rendah, dan keadaan dimana diharuskan memindahkan benda-benda yang sangat berat, maka tidak dimungkinkan pekerjaan dilakukan oleh manusia sehingga dibutuhkan robot untuk menggantikan pekerjaan manusia dengan ketelitian dan produktivitas yang tinggi. Untuk robot yang digunakan memindahkan barang dalam kondisi yang sangat berbahaya dan tidak membutuhkan mobilitas digunakan robot lengan.

Lengan robot didesain menyerupai tangan manusia agar dapat berfungsi sama atau menyerupai organ lengan manusia. Lengan robot dilengkapi dengan motor DCservo sebagai*actuator* yang berfungsimelakukan perubahan sudut-sudut pada sendi-sendi lengan robot. Lengan robot dapat memiliki tiga*Degree of Freedom* (DOF) atau lebih. Semakin

banyak DOF dari robot maka semakin banyak juga gerakan yang dapat dilakukan. Pengendalian lengan robot yang membutuhkan fleksibelitas yang tinggi dapat menggunakan metode perekaman atau yang disebut dengan motion capture system.

Dalam skripsi ini dirancang pergelangan tangan robot yang dapat mengikuti pergerakan tangan pengguna (*motion capture system*). Agar robot dapat mengikuti gerakan tangan pengguna maka digunakan beberapa sensor pada jari-jari dan punggung tangan pengguna. Sensor yang digunakan yaitu*flex sensor* pada jari-jari tangan, dan sensor IMU (*Inertial Measurement Unit*) diletakkan pada punggung telapak tangan. Sensor IMU yang digunakan sudah dilengkapi oleh sensor *accelerometer* dan sensor *gyroscope* yang terintegrasi dalam satu *chip*. Untuk pengiriman data digunakan *bluetooth* sebagai media transmisi data tanpa kabel agar lebih mudah dan dapat mengurangi kabel penghubung antar *device*.

1.2. Identifikasi Masalah

Dalam penelitian ini, dirancang alat yang dapat merekam gerakan pergelangan tangan dan jari-jari manusia yang digunakan untuk mengontrol gerakan pergelangan tangan robot dan jari-jari robot. Perekaman gerakan menggunakan *flex sensor* dan sensor *Inertial Measurenment Unit* (IMU) untuk mengidentifikasi gerakan-gerakan yang dilakukan oleh manusia (pengguna).

1.3. Rumusan Masalah

Berdasarkan latar belakang tersebut, maka dapat disusun rumusan masalah sebagai berikut :

1) Bagaimana merancang dan membuat alat *motion capture system* yang digunakan untuk merekam gerakan jari-jari dan pergelagan tangan pengguna.
2) Bagaimana merancang dan membuat pergelangan tangan dan jari-jari robot yang dapat menirukan perekaman dari alat *motion capture system*.
3) Bagaimana menerjemahkan data sudut dari alat *motion capture system* menjadi sudut-sudut pada sendi robot.
4) Bagaimana metode pengiriman data dari alat *motion capture* ke pergelangan tangan robot.

1.4. Batasan Masalah

Berdasarkan permasalahan yang telah dirumuskan, maka hal-hal yang berkaitan dengan penelitian dan perancangan alat diberi batasan sebagai berikut :

1) Gerakan pergelangan tangan robot dibatasi hanya dari pergelangan tangan sampai jari-jari tangan dari robot.
2) Pembuatan pergelangan tangan robot mengguanakan motor DC servo dengan 3*Degree of Freedom* (DOF) dengan rincian satu DOF pada pergelangan tangan dan dua DOF pada jari-jari tangan robot.
3) Pengujian tiga DOF keseluruhan sistem ditampilkan pada demonstrasi *live* dan *video.*

1.5. Tujuan Penelitian

Tujuan penelitian ini adalah merancang dan membuat alat *motion capture system*pada pergelangan tangan dan jari-jari pengguna berbasis *flex sensor*dan *Inertial Measurenment Unit* (IMU) untuk mengontrol gerakan pergelangan tangan dan jari-jari robot dengan komunikasi data menggunakan *Bluetooth.*

1.6. Manfaat Penelitian

Dapat terciptanya perancangan dan realisasi alat yang dapat digunakan untuk mengendalikan gerakan pergelangan tangan dan jari-jari robot dengan menirukan gerakan pergelangan tangan dan jari-jari manusia secara *real time* dan *continue*.

1.7. Sistematika Penulisan

Sistematika pembahasan proposal penelitian ini adalah sebagai berikut :

BAB I Pendahuluan

Berisi pendahuluan yang meliputi latar belakang, rumusan masalah, batasan, tujuan, manfaat dan sistematika pembahasan.

BAB II Landasan Teori

Membahas teori-teori yang mendukung dalam perancangan dan pembuatan alat.

BAB III Metodologi Penelitian, Pembuatan dan Perancangan Alat

Berisi metode-metode dalam penelitian yang meliputi metode perancangan, pengujian, dan analisis.Perancangan dan pembuatan alat berisi tentang perancangan dan analisis terhadap spesifikasi alat, meknik, *hardware*, dan *software* dari alat.

BAB IV Pengujian dan Analisis

Berisi tentang pengujian dan analisis dari masing-masing blok alat yang akan direalisasikan dan pengujian dan analisis secara keseluruhan.

BAB V Kesimpulan dan Saran

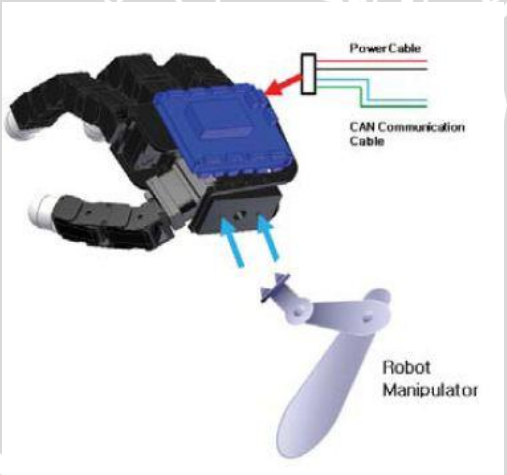
Berisi tentang kesimpulan dari penelitian yang telah dilakukan berdasarkan masalah yang telah dirumuskan sebelumnya.

BAB II
TINJAUAN PUSTAKA

Pengetahuan dan pemahaman yang dibutuhkan peneliti yang mendukung perencangan dan realisasi alat meliputi pengetahuan tentang tangan robot, *motion capture system*, *flex sensor*, sensor IMU,*Bluetooth*, Mikrokontroler, dan motor DC servo.

2.1. Tangan Robot

Kemampuan yang harus dimiliki oleh tangan robot yaitu tidak hanya dapat memahami perintah dalam bentuk kode program akan tetapi dapat juga memanipulasi gerakan. Gerakan tangan robot pada dasarnya harus dipertimbangkan, untuk gerakan yang terampil, dianjurkan agar jari-jari dari tangan robot memiliki banyak sendi yang aktif dan masing-masing sendi dapat dikendalikan. Banyaknya jumlah sendi pada tangan robot memiliki efek pada ukuran, berat dan tingkat kesulitan dalam pengendalian tangan robot. Pada umumnya jari-jari robot tidak digunakan dengan sendirian, melainkan dapat digabung dengan lengan robot manipulator seperti ditunjukan pada Gambar 2. 1

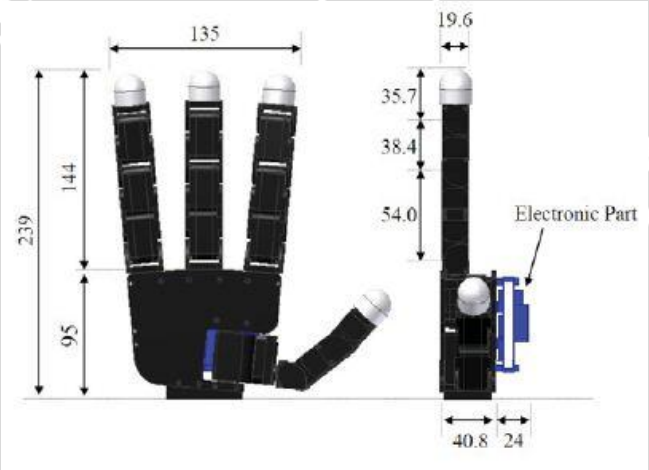


Gambar 2. 1Tangan Robot
Sumber : (Park,2012:2).

Tangan robot digunakan untuk melakukan tugas tugas seperti manusia, mengikuti gerakan seperti yang dilakukan oleh manusia. Untuk mendapatkan informasi tentang suatu objek, manusia menggunakan organ mata sebelum memahami prilaku objek, sehingga

nantinya dapat memutuskan tanggapan apa yang akan dilakukan berdasarkan prilaku tersebut. Tidak seperti halnya pada manusia, robot sulit untuk mengambil informasi. Oleh karena itu, yang dapat dilakukan adalah memberikan informasi secara langsung oleh pengguna kepada tangan robot. Pemberian informasi bisa langsung beruapa kode-kode program yang sudah ditanamkan terlebih dahulu ke dalam tangan robot ataupun dengan menirukan gerakan tangan pengguna dengan menambahkan sensor-sensor tertentu pada tangan pengguna (*motion capture system*)(Park, 2012:1).

Jumlah jari pada tangan robot yang paling banyak digunakan yaitu berkisar antara tiga sampai lima seperti ditunjukan dalamGambar 2. 2. Tangan robot digunakan sebagai sarana penelitian di berbagai bidang ilmu. Untuk membuat tangan robot peneliti perlu mempelajari struktur dan prilaku tubuh manusia (biomekanik). Simulasi pergerakan tubuh manusia mengarah pada pemahaman yang lebih baik mengenai hal tersebut(Park, 2012:2).



Gambar 2. 2Jari-jari Tangan Robot.
Sumber : (Park, 2012:2).

Secara keseluruhan, sebuah robot terdiri dari sistem mekanik, elektrik, dan program. Ditinjau dari segi kontrol, sebuah robot biasanya merupakan sistem loop tertutup, yang umumnya terdiri dari sensor, pengontrol, dan actuator. Sensor pada robot jika diibaratkan manusia berfungsi sebagai sistem pengindra, dan kontroller sebagai otak yang mengatur segala fungsi dari robot, serta aktuator sebagai sistem otot dan persendian. Robot tangan memiliki beberapa persendian yang disebut sebagai *Degree of Freedom* (DOF) yang berupa aktuator, seperti motor, pneumatik, dan sebagainya(Park, 2012:2).

2.2. *Degree of Freedom*

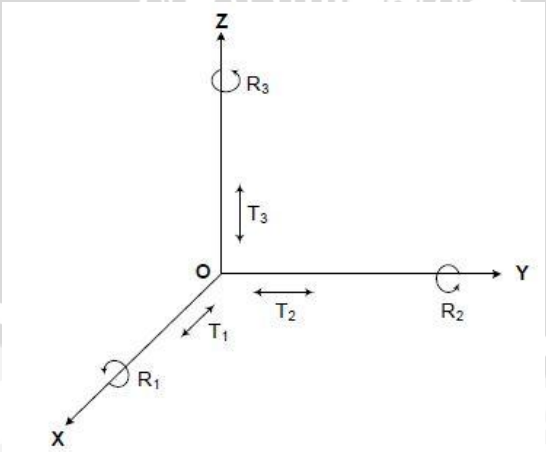
Derajat kebebasan (*Degree of Freedom*) adalah sambungan pada lengan, dapat dibengkokkan, diputar, maupun digeser. Derajat kebebasan digunakan untuk mengetahui cara robot bergerak, tingkat kerumitan algoritma kendali dan jumlah motor yang digunakan. Penentuan jumlah DOF dilakukan berdasarkan jumlah gerakan yang dapat dilakukan oleh lengan robot atau jumlah aktuator lengan robot.

Kaidah Denavit-Hartenberg merupakan aturan yang digunakan dalam perancangan robot yang diperkenalkan oleh Jaques Denavid dan Ricard S Hartenberg. Aturan tersebut menyatakan hanya terdapat dua gerakan yang mungkin terjadi yaitu bergeser dan berputar serta hanya terdapat 3 sumbu yang dapat terjadi yaitu sumbu x,y, dan z. Pada *end-effector* dapat ditambahkan empat derajat kebebasan berupa jari-jari tangan robot yang masing-masing jari dapat berputar 120^o sesuai dengan fungsi dan aktuator yang diinginkan(Nugraha 2011:22).

Derajat kebebasan (*Degree of Freedom*) suatu robot dapat diartikan sebagai jumlah gerakan independen yang dapat dibuat suatu objek terhadap sistem kordinat yang dapat menyebabkan perubahan posisi atau orientasi. Terdapat enam gerakan independen yang dapat dibuat suatu objek :

1) Tiga gerakan translasi T_1, T_2, T_3 sepanjang aksis OX, OY dan OZ.
2) Tiga gerakan rotasional R_1, R_2, R_3 pada aksis OX, OY, dan OZ.

Seperti ditunjukan padaGambar 2. 3.

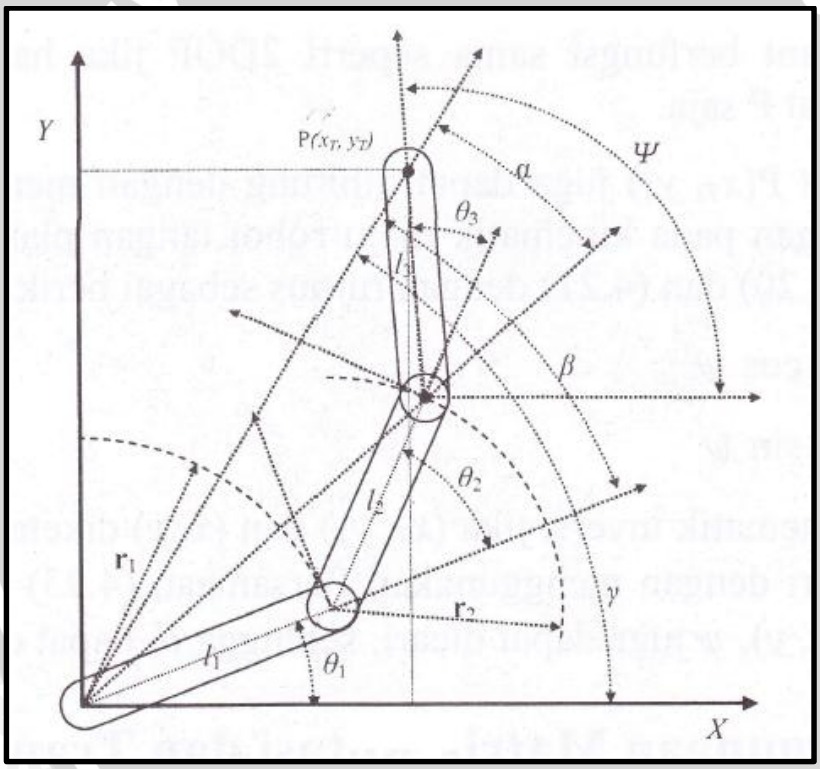


Gambar 2. 3 Derajat Kebebasan pada suatau Objek
Sumber : (Nugraha, 2011:23).

Didalam menentukan jumlah derajat kebebasan yang dimiliki oleh sebuah robot, tidak dapat dilakukan hanya dengan menghitung jumlah persendian (*joint*) yang dimiliki robot tersebut, karena tidak semua gerakan independen yang dibuat oleh persendian dapat dikategorikan sebagai derajat kebebasan (Nugraha, 2011:23).

2.3. Kinematika Tiga Sendi Dengan Persamaan Trigonometri.

Persamaan trigonometri adalah persamaan yang paling dasar dalam penyelesaian analisis kinematika sendi. Setiap kordinat (x,y,z) dapat dinyatakan sebagai transformasi dari setiap ruang sendi (r,θ). Jari-jari dapat dituliskan sebagai panjang dari setiap lengan (*l*). dalam bidang dua dimensi maka untuk sumbu z dapat tidak dituliskan. Konfigurasi tiga sendi dapat ditunjukan padaGambar 2. 4.



Gambar 2. 4 Konfigurasi Kinematika Tiga Sendi
Sumber : (Endra Pitowarno, 2006:23).

Kedudukan ujung lengan dinyatakan sebagai P(X,Y)

$$P(X,Y) = f(\theta_1, \theta_2, \theta_3) \quad (2\text{-}1)$$

Jika P diasumsikan sebagai vektor penjumlahan yang terdiri dari vektor $\mathbf{r_1}$lengan-1, vetor$\mathbf{r_2}$ lengan-2, dan vektor $\mathbf{r_3}$ lengan-3 maka :

$$r_1 = [l_1 \cos\theta_1, l_1 sin\,\theta_1] \quad (2\text{-}2)$$

$$r_2 = [l_2 \cos(\theta_1 + \theta_2), l_2 sin\,(\theta_1 + \theta_2)] \quad (2\text{-}3)$$

$$r_3 = [l_3 \cos(\theta_1 + \theta_2 + \theta_3), l_3 sin\,(\theta_1 + \theta_2 + \theta_3)] \quad (2\text{-}4)$$

Maka dengan menggunakan analisis kinematika maju maka diperoleh persamaan sebagai berikut :

$$X_T = l_1 cos\theta_1 + l_2 cos(\theta_1 + \theta_2) + l_3 cos(\theta_1 + \theta_2 + \theta_3) \quad (2\text{-}5)$$

$$Y_T = l_1 sin\theta_1 + l_2 sin(\theta_1 + \theta_2) + l_3 sin(\theta_1 + \theta_2 + \theta_3) \quad (2\text{-}6)$$

Ψ merupakan arah sudut lengan tiga terhadap sumbu x sehingga diperoleh persamaan

$$\psi = (\theta_1 + \theta_2 + \theta_3) \quad (2\text{-}7)$$

Hukum identitas trigonometri digunakan untuk menyederhanakan penjumlahan dalam trigonometri sehingga dapat digunakan persamaan sebagai berikut,

$$\cos(a + b) = \cos(a)\cos(b) - \sin(a)\sin(b) \quad (2\text{-}8)$$

$$\sin(a + b) = \sin(a)\cos(b) + \sin(b)\cos(a) \quad (2\text{-}9)$$

Sehingga Persamaan (2.5) dan Persamaan (2.6) dapat ditulis kembali sebagai berikut,

$$X_T = l_1 cos\theta_1 + l_2 cos\theta_1 cos\theta_2 - l_2 sin\theta_1 sin\theta_2 + l_3 cos(\theta_1 + \theta_2 + \theta_3) \quad (2\text{-}10)$$

$$Y_T = l_1 sin\theta_1 + l_2 sin\theta_1 cos\theta_2 + l_2 cos\theta_1 sin\theta_2 + l_3 sin(\theta_1 + \theta_2 + \theta_3) \quad (2\text{-}11)$$

2.4. Motion Capture System

Motion capture adalah proses perekaman gerakan dari manusia atau objek lainnya. *Motion capture device* bertugas mentransformasi pergerakan secara aktual dari subjek menjadi data posisi, orientasi, atau sudut persendian yang bisa digunakan. Aplikasi medis termasuk dalam kawasan dari studi pergerakan, analisa gaya berjalan, dan estimasi gaya persendian. Industri *entertainment* telah menunjukan minat yang tinggi terhadap seluruh *body motion capture*, terutama karena menyediakan cara cepat dan mudah untuk mendapatkan gerakan manusia yang realistis untuk animasi komputer (Kolosz, 1998:1).

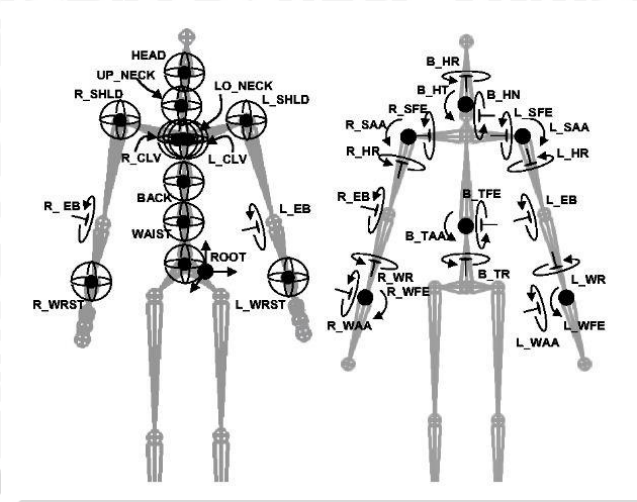
Animasi komputer tradisional dan pemrograman robot membutuhkan trajektori tersebut didesain untuk setiap derajat kebebasan individual. *Motion capture* memberikan pemrograman dari setiap derajat kebebasan secara simultan, sering kali secara *real time*. Pengontrolan *real time* dari komputer atau figur robot sangatlah sulit tanpa *motion capture*.

Data *motion capture* bersifat natural dan terlihat manusiawi, sejak data langsung diambil dari organ tubuh pengguna. *Motion capture* telah digunakan untuk pemrograman dan animasi komputer untuk kartun, iklan, dan*film* (Kolosz, 1998:1).

Sensor sensor yang digunakan dalam *motion capture system* bermacam-macam, seperti optikal, magnetik, inersial, serta elektromekanikal. Masing-masing sensor menentukan tipe sistem yang digunakan dalam proses *motion capture*.

Elektromekanikal merupakan salah satu yang dapat digunakan untuk melakukan *motion capture*, seperti untuk menirukan gerakan jari-jari tangan manusia bisa menggunakan sarung tangan yang dilengkapi beberapa sensor seperti *flex sensor*, *gyro* dan *accelerometer* yang dikoneksikan menjadi satu dalam suatu kontroller, sedangkan untuk menirukan gerakan lengan tangan manusia dapat digunakan sensor posisi putaran yaitu potensiometer, *hall effect* maupun *rotary encoder.Flex sensor* berfungsi untuk menentukan sudut tekuk dari jari-jari tangan, sedangkan *gyro* dan *accelerometer* berfungsi untuk mengetahui sudut kemiringan telapak tangan, dan potensiometer berfungsi untuk mengukur sudut persendian secara langsung dengan ditempatkan disekitar persendian pada salah satu lengan pengguna.

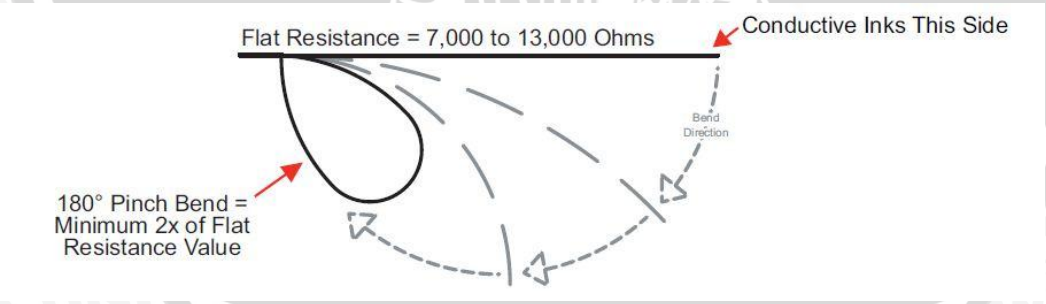
Metode yang paling sederhana dalam *motion capture* adalah menggunakan elektromekanis potensiometer yaitu dengan mengukur perpindahan orientasi dari organ tubuh pengguna. Pendekatan ini efektif dalam beberapa kasus karena tidak dipengaruhi oleh keadaan sekitar pengguna. Pengukuran terhadap sendi pengguna dapat langsung digunakan dan dapat secara portebel. Akan tetapi selain kemudahan tersebut, kelemahan utama pada metode ini yaitu gerakan yang dapat dilakukan oleh pengguna terbatas dan kekakuan dari alat *motion capture* ini berpengaruh terhadap kenyaman pengguna. Kerangka pada alat *motion capture* dapat membebankan pengguna dan membatasi gerak karena sendi pada manusia lebih fleksibel daripada mekanik alat *motion capture* tersebut. Selain itu mekanisme *motion capture* ini tidak dapat mendeteksi pengguna yang sedang melompat atau sedang memutar. Namun, metode ini memiliki tingkat kekuatan dan ketepatan dalam meniru gerakan setiap sendi-sendi pengguna. Derajat kebebasan untuk sendi-sendi manusia dapat ditunjukkan dalamGambar 2. 5.



Gambar 2. 5 Derajat kebebasan Setiap Sendi Bagian Atas Tubuh Manusia
Sumber : (Pollard, 1999:3).

2.5. *Flex Sensor*

Flex sensor adalah sensor yang nilai resistansinya berubah ketika sensor tersebut ditekuk, seperti ditunjukkan pada Gambar 2. 6. Perubahan resistansi pada sensor dapat meningkat atau menurun tergantung pada jenis flex sensor yang digunakan. Konsep ini menunjukkan bahwa jika flex sensor ditempatkan pada sendi jari, maka dapat ditentukan ketika jari tersebut menekuk atau tidak. Sehingga gerakan pada jari dengan mudah diberikan kode numerik yang akan digunakan sebagai sinyal perintah pada perangkat kontroller dan simulasi *virtual*.



Gambar 2. 6Cara Kerja *Flex Sensor*.
Sumber : (Spectra Symbol, 2014:1).

Dengan konsep ini, *flex sensor* dapat digunakan bersamaan dengan digital *gyroscope* dan *accelerometer*, yang dapat digunakan untuk mendeteksi sisi miring dari gerakan, sensor tersebut dapat ditempatkan pada sarung tangan yang memungkinkan

gerakan tangan dapat ditiru dan digunakan sebagai perintah untuk perangkat kontroller dan simulasi *virtual* dengan biaya rendah.

Fitur-fitur yang terdapat pada *flex sensor* adalah sebagai berikut :

1) Pengukuran perpindahan sudut.
2) *Bends and Flexes physically* dengan *motion device*.
3) Sensor ini dapat digunakan untuk robot, *gamming* (*virtual motion*), *medical device*, *computer peripheral*, instrumen musik, terapi fisik.
4) Kontruksi sensor sederhana.
5) *Low profile.*

Spesifikasi mekanik *flex sensor* adalah sebagai berikut :

1) Tinggi flex sensor 0,43 mm (0,017 inci).
2) Kisaran suhu yang dapat diterima sensor -35^{o} C sampai 80^{o} C.

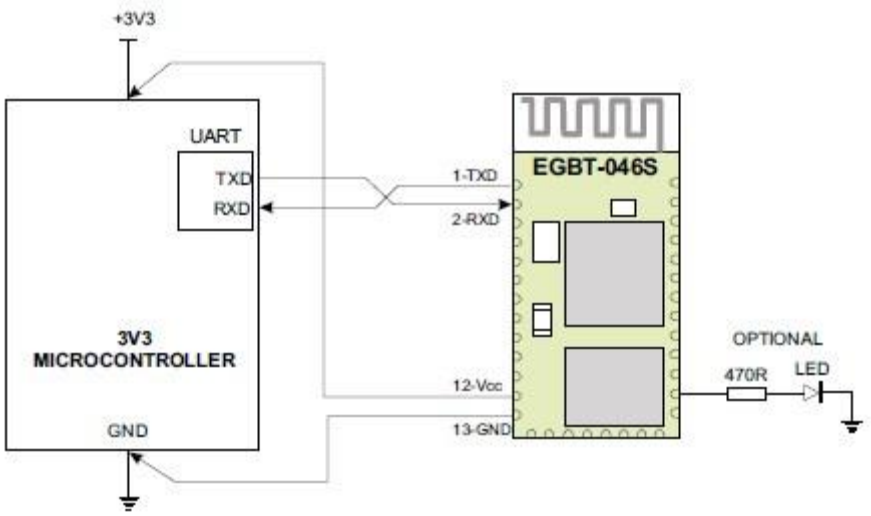
Spesifikasi elektrik *flex sensor* adalah sebagai berikut :

1) Resistansi flex sensor sebesar 10K Ohm dengan toleransi ±30%.
2) Daya yang dibutuhkan flex sensor 0.5 Watts continuous dan 1 Watt Peak
3) Dapat bekerja sampai sudut tekukan 180^{o}.

(Spectra Symbol, 2014:1).

2.6. *Bluetooth* HC-05

Bluetooth tersebut merupakan modul *bluetooth Serial Port Protocol* (SPP) yang berfungsi untuk melakukan komunikasi secara nirkabel (*serial wireless*) yang mudah digunakan dengan mengkonversi *serial port* ke *bluetooth*. *Bluetooth* tersebut memiliki dua mode konfigurasi yaitu *communication mode* yang berfungsi untuk melakukan komunikasi *bluetooth* dengan piranti lain atau sesama modul *bluetooth*. Selain itu terdapat *AT-mode* yang berfungsi untuk melakukan pengaturan konfigurasi pada modul bluetooth. Untuk komunikasi data, modul *bluetooth* menggunakan frekuensi sebesar 2.4 GHz. Komunikasi data menggunakan *bluetooth* dapat dilakukan secara *real time* antara *host-host bluetooth* dengan jarak-jarak yang terbatas. *Bluetooth* ini dapat bekerja sebagai *master*, *slave*, dan *loopback*. Daya yang digunakan oleh *bluetooth* sebesar 3,3 volt (*low power voltage*). Konfigurasi pin *bluetooth*dapat ditunjukan padaGambar 2. 7.



Gambar 2. 7*Interface* Mikrokontroler dengan *Bluetooth*.
Sumber : (e-Gizmo Mechatronix Central, 2011:3)

Mode kerja pada modul ini dapat diaktifkan dengan mengontrol PIN (PIO11). Pada modul ini terdapat pin serial sebagai berikut :

1) PIO8 yaitu terhubung dengan LED. Ketika modul tersebut diberikan daya maka LED akan menyala. LED yang berkedip akan menunjukan mode bekerja, dan perbedaan interval waktu berkedip menunjukan mode yang berbeda pada modul bluetooth.
2) PIO9 terhubung dengan LED. PIO9 menunjukan apakah modul sudah terkoneksi atau tidak. Apabila serial bluetooth telah terpasang maka LED akan menyala, yang berarti koneksi telah tersambung.
3) PIO11 adalah untuk mengalihkan mode kerja modul.
4) Modul dapat direset jika terjadi *re-powered* karena ada *reset circuit* pada modul.

Secara umum, cara kerja dari modul *bluetooth* yaitu pada keadaan awal, modul ini berada pada *mode standby*, pada mode ini *bluetooth* akan memerikasa setiap 1.28 detik dengan waktu pemeriksaan 10 ms, waktu pemeriksaan ini disebut dengan *inquiry*. Jika ingin melakukan komunikasi dengan piranti lain atau sesama *bluetooth* HC-05 maka harus diketahui *system clock* dan alamat untuk menentukan paket data (*access code*). Untuk menjalin komunikasi antar piranti maka salah satu dari piranti ada yang sebagai *master* dan ada yang sebagai *slave*, yang disebut sebagai *piconet*. *Bluetooth master* dapat berkomunikasi dengan tujuh *slave*.

Setelah *bluetooth* terhubung, terdapat empat mode operasi, yaitu mode *active*, mode *sniff*, mode *hold*, mode *park*. Mode *active* yaitu dapat berkomunikasi secara aktif dalam transmisi data. Mode *sniff* yaitu komunikasi dilakukan pada waktu tertentu. Pada mode *hold*, komunikasi berada pada frekuensi yang lebih rendah daripada mode *sniff*. Sedangkan pada mode *park* tidak berpartisipasi dalam hubungan *bluetooth master* dan *bluetooth slave*, tetapi tetap mempertahankan sinkronisasi dengan kanal komunikasi.

2.7. Motor Servo

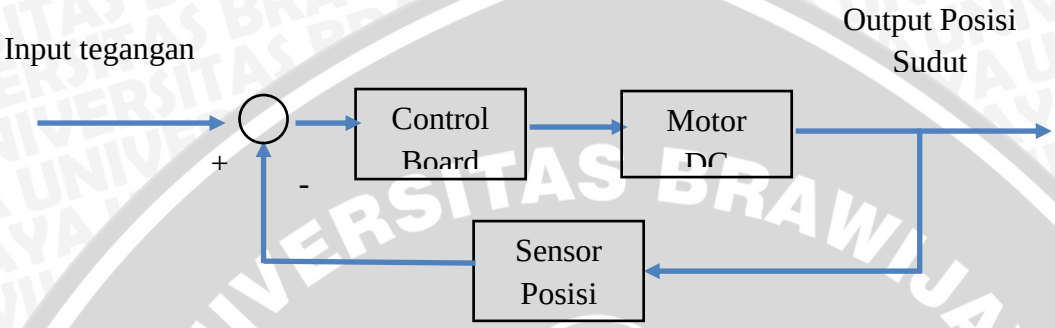
Berbeda dengan motor DC, motor servo tidak bergerak *continue*, melainkan menuju sudut tertentu saja dan berhenti di sudut tersebut. Motor ini digunakan untuk aplikasi gerakan-gerakan sudut dari robot, contohnya gerakan lengan robot, gripper menjepit benda, atau gerakan kaki melangkah. Motor servo ini terdiri atas beberapa bagian, yaitu:

1) Jangkar, untuk menghubungkan motor servo dengan objek-objek yang akan digerakkan.
2) Lubang jangkar, bagian ini untuk menepatkan sekrup yang mengaitkan jangkar ke objek-objek yang akan digerakkan. Lubang jangkar dihubungkan ke objek dengan sekrup untuk gerakan memutar.
3) Lubang sekrup, berfungsi untuk mengkaitkan motor servo dengan tubuh robot.
4) *Housing* servo, didalam bagian ini terdapat motor DC, *gearbox*, dan rangkain pengatur sudut servo.
5) Kabel, yang menghubungkan rangkaian servo dengan pengendali servo.
6) Konektor, terdapat tiga pin yang terdiri atas input tegangan positif (+), input tegangan negatif (GND), dan input pulsa (sinyal).

(Nalwan, 2012:13).

Tidak seperti motor DC, pengendalian motor servo harus dilakukan dengan terlebih dahulu melalui rangkaian pengatur sudut dimana rangkaian ini membutuhkan masukan berupa pulsa. Pulsa-pulsa ini biasanya dibangkitkan oleh komponen cerdas seperti mikrokontroller atau modul elektronik dekat servo kontroller. Modul delta servo kontroller berfungsi membangkitkan pulsa-pulsa untuk mengendalikan motor servo dari PC (*personal computer*) (Nalwan, 2012:13).

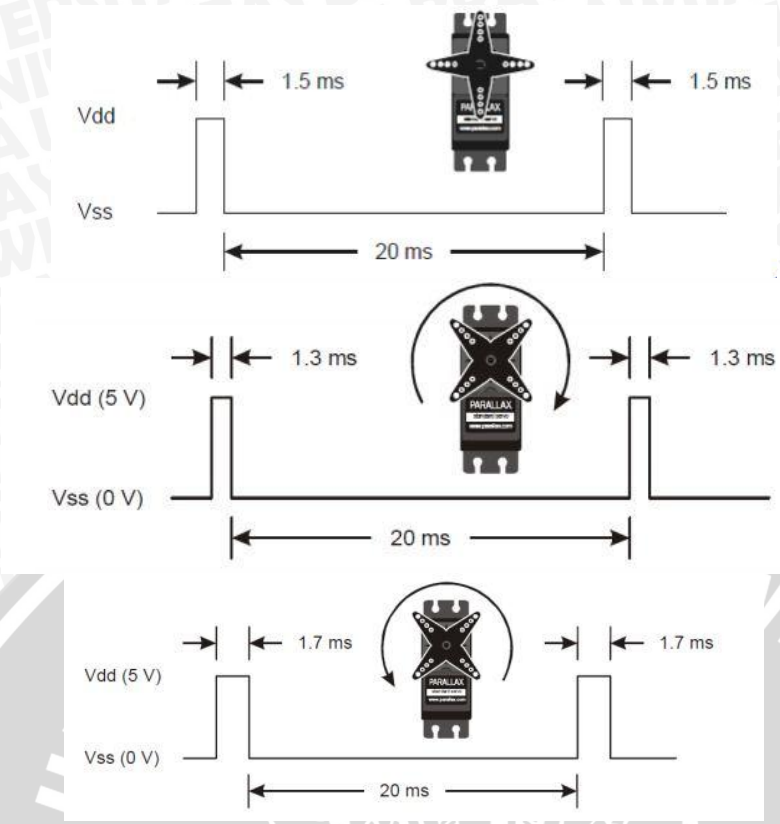
Motor servo menggunakan umpan balik tertutup dimana posisi sudut dari motor akan dinformasikan kembali ke rangkaian kontroller yang ada di dalam motor servo tersebut. Poros motor dihubungkan dengan rangkaian kendali, sehingga jika putaran poros belum sampai pada posisi yang diperintahkan maka rangkaian kendali akan terus mengoreksi posisi hingga mencapai posisi yang diperintahkan, seperti ditunjukan dalamGambar 2. 8.



Gambar 2. 8 Diagram Blok Sistem Kontrol Tertutup pada Motor Servo.

Motor dipasangkan dengan beberapa jenis *encoder* untuk memberikan posisi dan kecepatan umpan balik. Dalam kasus yang paling sederhana, hanya posisi yang diukur. Posisi diukur dari output dibandingkan dengan posisi perintah, *input eksternal* ke *controller*. Jika posisi keluaran berbeda dari yang diperlukan, sinyal *error* yang dihasilkan yang kemudian menyebabkan motor berputar pada kedua arah, yang diperlukan untuk membawa poros output ke posisi yang sesuai. Sebagai pendekatan posisi, sinyal *error* tereduksi menjadi nol dan motor berhenti.

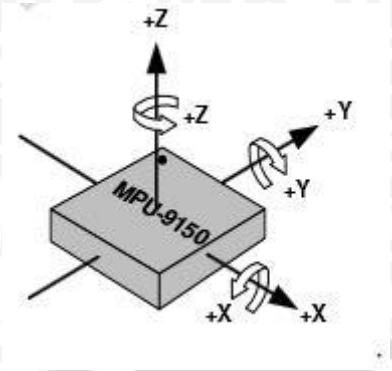
Motor Servo akan bekerja secara baik jika pada bagian pin kontrolnya diberikan sinyal PWM dengan frekuensi 50Hz, yang berarti sinyal PWM memiliki periode 20 ms. Pada saat sinyal dengan frekuensi 50Hz tersebut dicapai pada kondisi *ton duty cycle* 1.5 ms, maka rotor dari motor akan berhenti tepat di tengah-tengah (sudut 0° atau netral). Ketika diberikan sinyal PWM pada kontrol yang berada didalam motor servo maka kontrol akan merepon dengan syarat jika lebar pulsa 0,7 ms - 1 ms maka akan memutar motor searah jarum jam. Sedangkanlebar pulsa 1,7 ms – 2 ms akan memutar motor berlawanan arah jarum jam, sudut pada servo ketika diberikan pulsa PWM dapat ditunjukan pada Gambar 2. 9.



Gambar 2. 9 Diagram waktu PWM motor servo
Sumber : (Parallax:5-6).

2.8. Sistem Inertial Measurement Unit (IMU)

Inertial Measurement Unit (IMU) adalah modul elektronika yang dapat mengumpulkan data percepatan *angular* dan akselarasi linier. IMU terdiri dari kombinasi antara *gyroscope* dan *accelerometer*. *Gyroscope* digunakan untuk mengukur rotasi, kecepatan sudut yang dinamis dari suatu benda dan *accelerometer* digunakan untuk mengukur kecepatan, mendeteksi dan mengukur getaran (vibrasi) suatu benda. Sensor *accelerometer* mengukur percepatan akibat gerakan perpindahan benda yang melekat padanya. Akselarasi pada sensor dapat dideteksi pada sumbu x,y,z, seperti ditunjukkan pada Gambar 2. 10.



Gambar 2. 10 Orientasi Sumbu, Sensitivitas dan Polaritas Rotasi dari Accelerometer dan Gyroscope
Sumber : (InvenSense, 2011:6)

Dengan adanya kombinasi antara *gyroscope* dan *accelerometer* maka kekurangan dari masing-masing sensor akan saling melengkapi. *Accelerometer* dapat memberikan pengukuran sudut pada saat sensor dalam keadaan diam, sedangkan pada saat berotasi *accelerometer* tidak dapat bekerja secara maksimal dan akan memberikan respon yang lambat. Kelemahan yang terjadi pada *accelerometer* dapat diatasi oleh *gyroscope* karena *gyroscope* dapat membaca kecepatan sudut yang dinamis. *Gyroscope* juga memiliki kelemahan pada saat terjadi perpindahan kecepatan sudut dalam interval waktu yang panjang yang menyebabkan pembacaan data tidak akurat, hal ini disebabkan oleh efek bias yang dihasilkan oleh *gyroscope*.

Sensor IMU dapat diterapkan sebagai sistem navigasi robot. Untuk robot tangan, *gyroscope* pada IMU dapat digunakan untuk mengukur sudut kemiringan dari robot tangan dan *accelerometer* digunakan untuk mengukur percepatan perpindahan sudut robot tangan. Dalam pembuatan alat, sensor IMU yang akan digunakan yaitu MPU-6050.

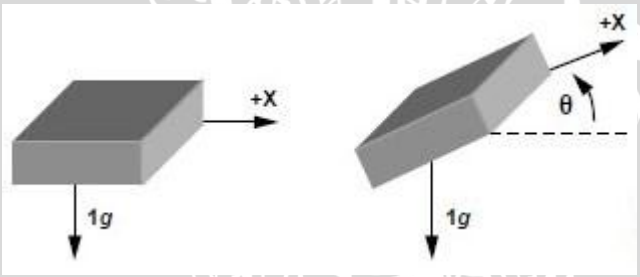
Sensor adalah chip IC yang diproduksi oleh Invense.ltd yang di dalamnya terdapat sensor *accelerometer* dan *gyroscope* yang telah terintegrasi. Sensor tersebut memilki *hardware* 16-bit yang dapat mengkonversi dari analog ke digital dari setiap jalurnya, oleh karena itu sensor ini mendapatkan nilai x, y, dan z pada waktu yang sama. Sensor tersebut memiliki spesifikasi sebagai berikut :

1) Berbasis *Chip* MPU-6050
2) *Supply* tegangan berkisar 3V-5V
3) *Gyroscope range* + 250 500 1000 2000 °/s
4) *Acceleration range*: ± 2 ± 4 ± 8 ± 16 g

5) *Communication standard* I2C
6) *Chip built-in* 16 bit *Analog Digital converter*, 16 bits data output
7) Jarak antar pin *header* 2.54 mm
8) Dimensi modul 20.3mm x 15.6mm

2.8.1. Penentuan Sudut *Accelerometer* dan *Gyroscope*

Sensor IMU terdiri atas 3 *axis accelerometer* dan 3 *axis gyroscope* yang telah terintegrasi dalam satu *chip*. Sehingga gerakan yang dapat dideteksi oleh sensor ini yaitu 6 DOF (*Degree Of Freedom*), untuk mengetahui bagaimana konversi data menjadi sudut maka penting untuk memahami bagaimana proses kerja 1 *axis*, 2 *axis*, dan 3 *axis* dalam sensor *accelerometer*. 1 *axis* memiliki kecendrungn keterbatasan dalam gerakan dengan resolusi yang agak kasar. Dalam menentukan persamaan dimisalkan *single-axis* dalam sumbu x dan penentuan arah gerakan sumbu x selalu berorientasi pada gravitasi (1g), sehingga dalam persamaan mengacu pada dua sumbu yaitu sumbu x dengan sumbu 1g. vektor dari 1 axis dapat ditunjukkan padaGambar 2. 11.

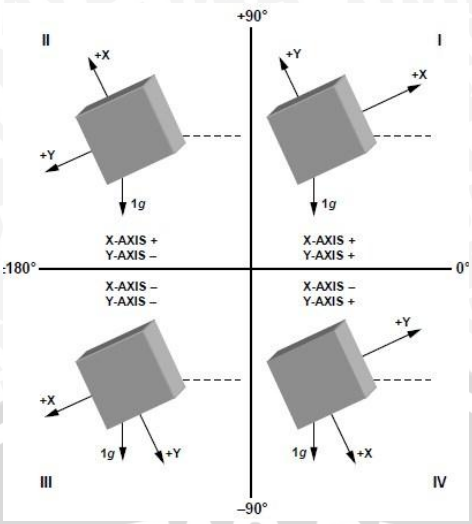


Gambar 2. 11Vektor Satu Axis
Sumber : (Analog Device,2010:3)

Dari Gambar 2.11 didapatkan persamaan yaitu :

$$Ax,out\ [g] = 1g \times sin\ (\theta) \qquad (2\text{-}12)$$

Mengacu pada trigonometri dasar yaitu proyeksi vektor gravitasi (1g) terhadap vektor sumbu x menghasilkan percepatan sama dengan sinus dari sudut pada sumbu x. keterbatasan dari *single-axis* yaitu keterbatasan dalam gerakan yang tidak bisa mencapai sudut 360^o tetapi hanya mencapai 180^o. Untuk mengatasi permasalahan dari *single-axis* maka dapat digunakan 2-*axis*. 2-*axis* memiliki konsep yang sama seperti *single-axis*, bedanya ada penambahan vektor sumbu y, seperti ditunjukkan padaGambar 2. 12.



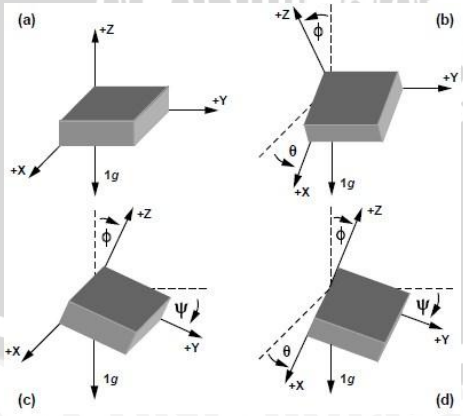
Gambar 2. 12Sudut Untuk Medeteksi Quadran.
Sumber : (Analog Device,2010:6)

Pada Gambar 2.12 didapatkan persamaan seperti berikut :

$$\frac{Ax,out}{Ay,out} = \frac{1g \times \sin(\theta)}{1g \times \cos(\theta)} = \tan(\theta) \quad (2\text{-}13)$$

$$\theta = \tan^{-1} \frac{Ax,out}{Ay,out} \quad (2\text{-}14)$$

Sudut kemiringan θ adalah dalam radian, pergerakan dari 2-*axis* dapat menentukan pergerakan 360º. agar memiliki pergerkan yang lebih halus dan lebih banyak pergerakan yang bisa dijelajah dapat digunakan sumbu 3-*axis*, seperti yang ditunjukkan padaGambar 2. 13.



Gambar 2. 13Vektor Sumbu 3-Axis
Sumber : (Analog Device,2010:7)

Pada Gambar 2.13 didapatkan persamaan yaitu :

$$\varphi = \cos^{-1} \frac{Az,out}{\sqrt{A^2x,out+A^2y,out+ A^2z,out}} \quad (2\text{-}15)$$

Sumbu (x,y,z) dirubah kedalam bentuk bola yaitu (ρ,θ, φ), karena memiliki kesamaan antara *single-axis* dan *dual-axis* maka untuk analisis dapat menggunakan metode *dual-axis* yaitu seperti pada persamaan berikut :

$$\theta = \tan^{-1} \frac{Ax,out}{\sqrt{A^2y,out+ A^2z,out}} \quad (2\text{-}16)$$

$$\psi = \tan^{-1} \frac{Ay,out}{\sqrt{A^2x,out+ A^2z,out}} \quad (2\text{-}17)$$

$$\Phi = \tan^{-1} \frac{\sqrt{A^2y,out+ A^2z,out}}{Az,out} \quad (2\text{-}18)$$

selain sensor *accelerometer* terdapat juga sensor *gyroscope* yang merupakan sensor untuk mengukur kecepatan sudut (ω). Untuk mendapatkan sudut dari sensor ini yaitu dengan melakukan integrasi terhadap data yang telah didapatkan dari sensor IMU. Persamaan kecepatan sudut adalah sebagai berikut :

$$\omega = \frac{d\theta}{dt} \quad (2\text{-}19)$$

$$\theta = \int \omega \, dt \quad (2\text{-}20)$$

Tetapi kelemahan dari persamaan tersebut yaitu apabila ada *noise* pada saat proses integrasi maka *noise* tersebut akan ikut terintegrasi walaupun *noise* tersebut kecil. *Gyroscope* pada penerapan membutuhkan untuk dikalibrasi sebelum digunakan, proses kalibrasi dari gyro dapat ditunjukkan dari persamaan berikut :

$$gyro_x = \frac{d}{dt}\theta x \quad (2\text{-}21)$$

$$gyro_y = \frac{d}{dt}\theta y \quad (2\text{-}22)$$

$$gyro_z = \frac{d}{dt}\theta z \quad (2\text{-}23)$$

Setelah melakukan kalibrasi maka akan didapatkan *gyro offset*, agar nilai *gyroscope* nol pada saat dalam keadaan diam maka nilai *gyro* sekarang dikurangi dengan nilai *gyrooffset*. Percepatan sudut terjadi ketika terjadi rotasi. Setelah mendapat nilai gyro rate (x,y,z) maka nilai tersebut dikurangin dengan nilai *sensitivity* yaitu 131. Untuk mendapat sudut keluaran dari gyroscope maka digunakan metode integrasi *forward-Euler* dengan persamaan sebagai berikut:

$$gyro_x = \frac{d}{dt}\theta_x^{gyro} \quad (2\text{-}24)$$

$$gyro_y = \frac{d}{dt}\theta_y^{gyro} \quad (2\text{-}25)$$

$$gyro_z = \frac{d}{dt}\theta_z^{gyro} \quad (2\text{-}26)$$

Untuk menghitung sudut gyrocope harus di integrasi sehingga persamaan untuk sumbu x sebagai berikut:

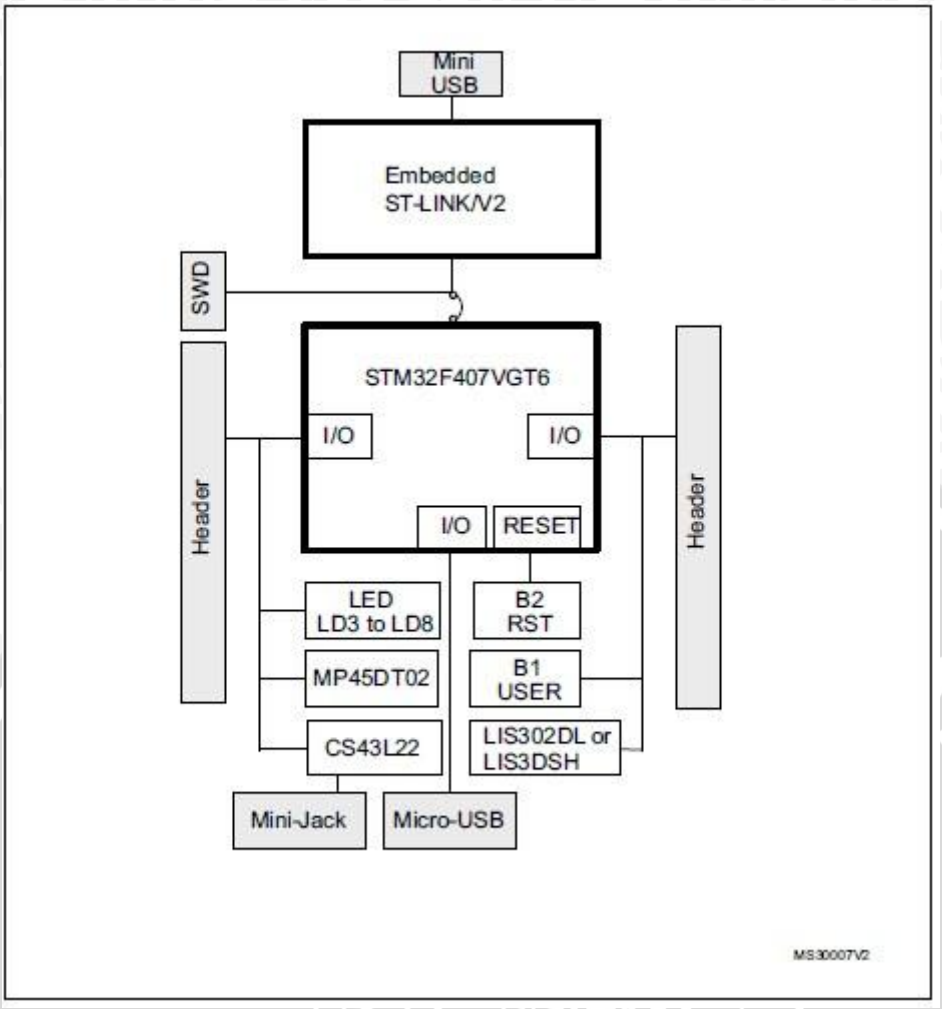
$$\theta_x^{gyro} = \int gyro_x \, dt \quad (2\text{-}27)$$

Sehingga persamaan yang didapatkan dalam metode integrasi forward-Euler adalah sebagai berikut:

$$\theta_x^{gyro}(t_n) = gyro_x \times T + \theta_x^{gyro}(t_n - 1) \quad (2\text{-}28)$$

2.9. Sistem Minimum STM32F407VG

Mikrokontroler tersebut merupakan sistem minimum yang didalamnya terdapat memori, mikroprosessor, *debugger Onboard*, dan perangkat lain yang terdapat dalam satu *chip*. Mikrokontroler tersebut merupakan buatan dari STMicroelectronic keluarga dari ARM Cortex-M4 yang merupakan *mikrokontroller* CMOS 32 bit dengan arsitektur RISC dan memiliki *clock* sampai dengan 168 MHz sehingga mampu mengeksekusi perintah sampai dengan 210 MIPS (*Milion Instruction per Second*). Mikrokontroler ini memiliki 100 pin, 5 port *input* atau *output* yang masing-masing dapat di program, memiliki 1 MB (*Mega Bytes*) *Flash Programable and Eraseble Read Only Memory* (*Flash* PEROM), memiliki 12 *timer*/*counter* 16 bit dan 2 buah *timer*/*counter* 32 bit, memiliki 192 KB (*Kilo Bytes*) SRAM, memiliki 24 channel ADC (*Analog to Digital Converter*), dan memiliki antarmuka kamera secara parallel 8-14 bit dengan kecepatan 54 MB (Mega Byte). Diagram blokmikrokontroller dapat ditunjukkan pada Gambar 2. 14.



Gambar 2. 14 Diagram Blok Mikrokontroler
Sumber : (STMicroelectronic :40).

Didalam STM32F4 terdapat dua DMA (*Direct Memory Acces*) yang masing-masing dapat menangani delapan perpindahan data dari suatu memori ke memori yang lain. DMA merupakan proses perpindahan data dari saatu alamat ke alamat lainnya yang tidak mengganggu kerja dari mikrokontroller. DMA dapat dintegrasikan dengan peripheral lain yaitu:

1) USART (*Universal Synchronous Asynchronous Receiver Transmitter*)
2) I2C (*Inter Integrated Circuit*)
3) SPI (*Serial Peripheral Interface*)
4) SDIO (*Secure Digital Input/Output*)
5) ADC (*Analog to Digital Converter*)
6) *Timer*/*counter*
7) DAC (*Digital to Analog Converter*)

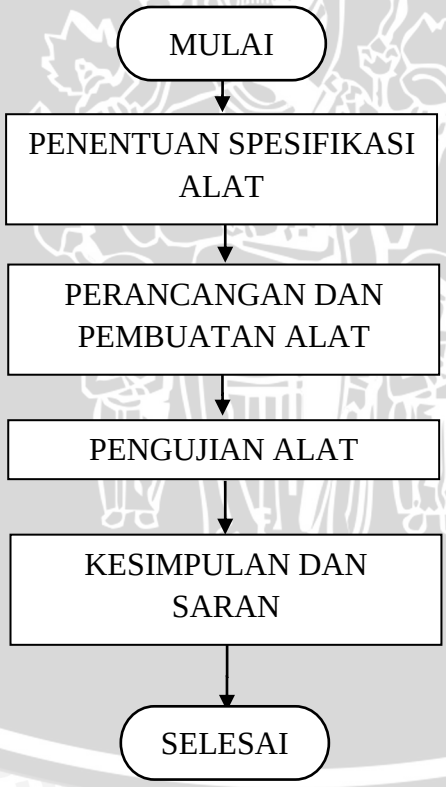
8) PWM (*Pulse Widh Modulation*)

Peripheral yang akan digunakan dalam penelitian ini adalah SDIO, USART, I2C, ADC, *Timer*/*Counter*, dan PWM (STMicroelectronic:1).

UNIVERSITAS BRAWIJAYA

BAB III
METODE PENELITIAN

Pada bab ini akan diuraikan tentang metode penelitian yang akan digunakan dalam proses perancangan dan realisasi alat *motion capture system* berbasis *flex sensor* untuk gerakan pergelangan tangan robot dengan komunikasi data *bluetooth*. Hasil luaran yang diinginkan dalam penelitian ini yaitu dapat merealisasikan alat *motion capture system* yang berbasis *flex sensor* yang dapat digunakan untuk menggerakkan pergelangan tangan dan jari-jari robot sesuai dengan gerakan yang di tangkap oleh alat *motion capture system*. Komunikasi antara alat *motion capture* dengan robot dilakukan dengan menggunakan *bluetooth* sebagai media transmisi data dari alat *motion capture* ke robot. Metode yang digunakan secara umum dapat ditunjukkan pada Gambar 3. 1.



Gambar 3. 1 Diagram Alir Metodologi Penelitian

3.1. Penentuan Spesifikasi Alat

Penetapan spesifikasi alat secara umum dilakukan sebagai acuan dalam perancangan berikutnya. Spesifikasi alat yang direncanakan adalah sebagai berikut :

1) Robot berbahan dasar alumunium pada pergelangan tangan robot dan berbahan dasar mika *acrylic* pada jari-jari dan telapak tangan robot.
2) Robot yang akan dirancang memiliki 3 DOF (*Degree of Freedom*) dengan rincian 2 DOF pada jari-jari tangan robot dan 1 DOF pada pergelangan tangan robot.
3) *Actuator* yang digunakan pada robot yaitu motor DC Servo.
4) Motor DC servo yang digunakan bertipe *Tower Pro* MG995 pada pergelangan tangan robot dan EMAX ES08MD pada jari-jari robot.
5) Menggunakan sistem minimum (mikrokontroler) sebagai pengendali pergerakan robot dan pengolah data sensor.
6) Transmisi data menggunakan *bluetooth*.
7) Untuk merekam gerakan dari setiap jari-jari pengguna pada alat *motion capture* digunakan *flex sensor*.
8) Untuk mendeteksi kemiringan dari pergelangan tangan pengguna digunakan sensor IMU yang dalam chip ini terintegrasi sensor *accelerometer* dan *gyroscope*.
9) Sumber tegangan yang digunakan yaitu baterai *Lithium Polymer3 Cell* 11,1 volt 2200mAh.
10) Regulator DC-DC *step-down* yang digunakan yaitu XL4005 5A, LM2596, dan LM1117-3,3V.

3.2. Perancangan dan Pembuatan Alat.

Perancangan dan realisasi alat dalam penelitian ini dibagi menjadi tiga bagian yaitu mekanik, *hardware*,dan *software*.

3.2.1. Perancangan Mekanik

Perancangan mekanik pada penelitian ini diperlukan untuk membuat desain alat *motion capturesystem* dan desain pergelangan tangan robot. Bentuk mekanik secara umum dirancang menggunakan perangkat lunak *Corel Draw*.

3.2.2. Perancangan Perangkat Keras

Perancangan rangkaian elektronika lebih ditekankan pada perancangan rangkaian antarmuka *flex sensor* dengan mikrokontroler, sensor IMU dengan mikrokontroler, *bluetooth* dengan mikrokontroler, dan motor DC servo dengan mikrokontroler. *Shieldboard* mikrokontroller. Papan rangkaian tercetak (PCB) dirancang dengan menggunakan perangkat lunak EAGLE (*Easily Applicable Graphical Layout Editor*).

3.2.3. Perancangan Perangkat Lunak

Setelah melakukan perancangan dan pembuatan perangkat keras baru dilakukan perancangan dan pembuatan perangkat lunak. Perangkat lunak akan dibuat menggunakan CooCox CoIDE yang merupakan salah satu IDE yang *open source* untuk membuat *project*, melakukan *editing*, *debugging*, dan *compiling* untuk ARM *Cortex Mseries*. Perangkat lunak yang akan dibuat berfungsi untuk memberikan perintah ke *actuator* motor DC servo dengan data masukan dari *flex sensor* dan Sensor IMU yang digunakan.

Perangkat lunak dirancang dari pembuatan diagram alir (*flowchart*) sistem secara keseluruhan. Kemudian dilakukan penulisan program menggunakan bahasa pemrograman C pada *software* CooCox CoIDE.

3.3. Pengujian Alat

Untuk menguji alat apakah sudah sesuai dengan yang telah direncanakan atau tidak maka diperlukan pengujian sistem. Pengujian keseluruhan sistem dilakukan dengan menghubungkan blok-blok perangkat keras (*hardware*) dan mengoprasikan sistem kemudian dapat dilakukan analisis apakah perangkat keras sudah bekerja sesuai yang diharapkan. Setelah perangkat keras sudah beroprasi sesuai yang diharapkan, maka perangkat lunak dapat diuji. Apabila perangkat keras dan perangkat lunak bekerja dengan baik dan saling bersinergi maka keseluruhan sistem sudah bekerja sesuai dengan spesifikasi rancangan.

3.3.1. Pengujian Sub Sistem

Pengujian ini dilakukan untuk masing-masing blok pada alat. Pengujian-pengujian yang dilakukan adalah sebagai berikut :

1) Pengujian Rangkaian Motor DC Servo.

Pengujian ini bertujuan untuk mengetahui kesesuaian gerak motor servo. Tiap-tiap servo dilakukan pengujian yang sama untuk mengetahui kecepatan, akurasi sudut, dan torsi.

2) Pengujian Rangkaian *Flex Sensor*.

Pengujian *flex sensor* dilakukan untuk mengukur tegangan keluaran terhadap tekukan yang terjadi pada flex sensor.

3) Pengujian Rangkaian Sensor IMU.

Pengujian Sensor IMU dilakukan untuk mengetahui sudut kemiringan dari pergelangan tangan robot, menguji sudut dari sensor accelerometer, menguji percepatan sudut dari sensor gyroscope dan menguji time constant pada complementary filter.

4) Pengujian Rangkaian *Bluetooth*.

Pengujian bluetooth dilakukan untuk mengetahui proses komunikasi dan pengiriman data antara *bluetooth master* dengan *bluetooth slave* dan mengetahui jarak maksimal pengiriman data.

5) Pengujian Rangkaian Minimum sistem.

Pengujian mikrokontroller bertujuan untuk mengetahui apakah mikrokontroller sudah dapat mengolah data dan menjalankan instruksi berupa kode program yang telah didefinisikan atau tidak.

3.3.2. Pengujian Keseluruhan Sistem

Pengujian keseluruhan sistem dilakukan dengan menghubungkan blok-blok sistem yang sesuai dengan diagram blok sistem dan memasukkan program kedalam mikrokontroller untuk mengontrol *hardware* dari keseluruhan sistem yang telah dibuat. Sistem akan berjalan dengan baik apabila sudah berjalan sesuai dengan *flowchart* yang telah dibuat.

BAB IV
HASIL DAN PEMBAHASAN

4.1. Perancangan dan Pembuatan Alat

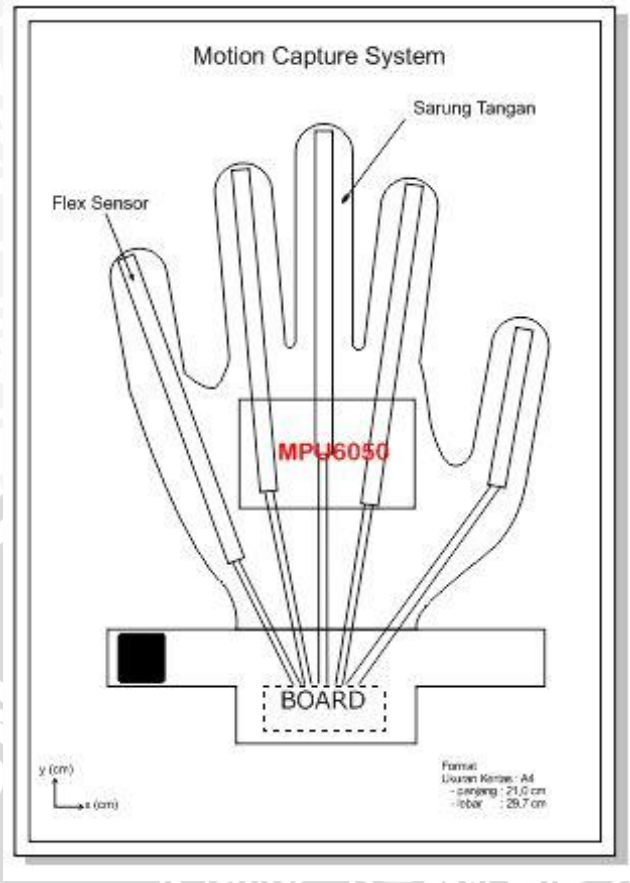
Penyusunan penelitian ini berkaiatan dengan perancangan dan pembuatan gerakan pergelangan tangan robot dengan *motioncapture system* menggunakan *bluetooth* yang di dasarkan pada rumusan masalah yang telah di rumuskan. Perancangan di lakukan untuk merealisasikan alat yang telah direncanakan dengan langkah-langkah sebagai berikut yaitu perancangan mekanik, perancangan perangkat keras (*hardware*), dan perancangan perangkat lunak (*software*).

4.1.1. Perancangan Mekanik

Pada perancangan mekanik, ada dua macam sistem yang harus dirancang yaitu perancangan sistem *motion capture* dan perancangan mekanik pergelangan tangan dan jari-jari robot.

4.1.1.1. Perancangan Mekanik Motion Capture System

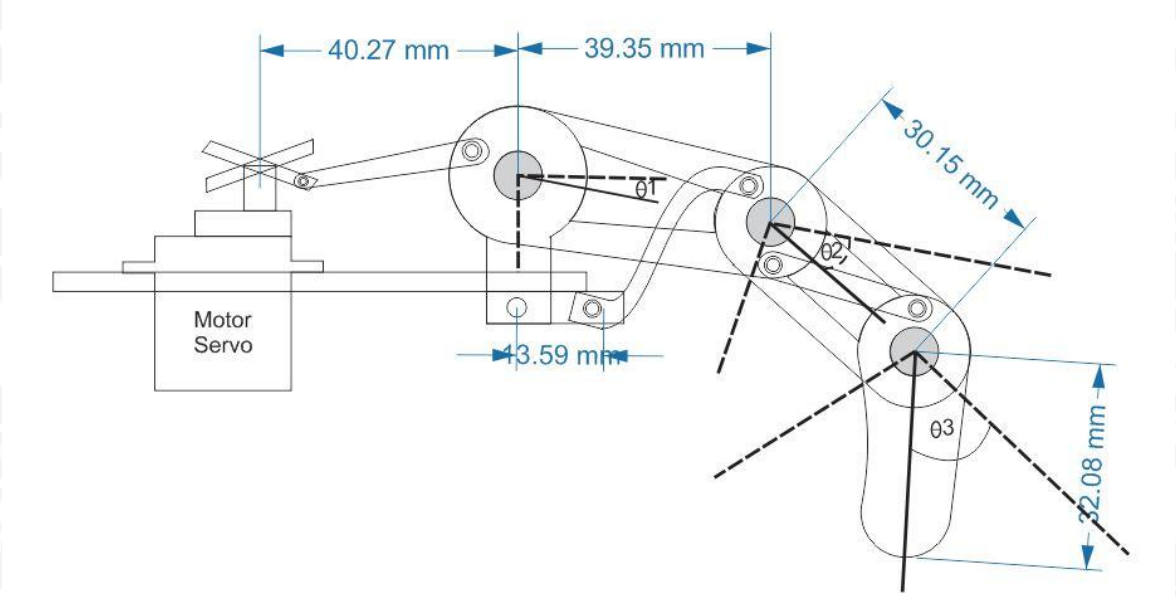
Pada perancangan mekanik *motion capture system* menggunakan bahan-bahan seperti sarung tangan, *flex sensor*, sensor IMU, *mainboard*, dan *casing mainboard*. Perancangan dilakukan dengan menghubung setiap blok dari komponen mekanik dan *hardware* dari sistem *motion capture system*. Perancangan sistem *motion capture* dapat ditunjukkan pada Gambar 4.1.



Gambar 4. 1 Mekanik Sistem *Motion Capture*.

4.1.1.2. Perancangan Mekanik Robot

Pada perancangan ini, bahan bahan yang digunakan yaitu mika *acrylic* 3 mm, dan aluminium. Mekanik tersebut digunakan untuk menampilkan hasil perekaman dari alat *motion capture*. Mekanik robot didesain menyerupai jari-jari manusia sehingga peniruan gerakan semakin mudah dilakukan. Perancangan mekanik jari-jari robot dapat ditunjukkan pada Gambar 4.2.



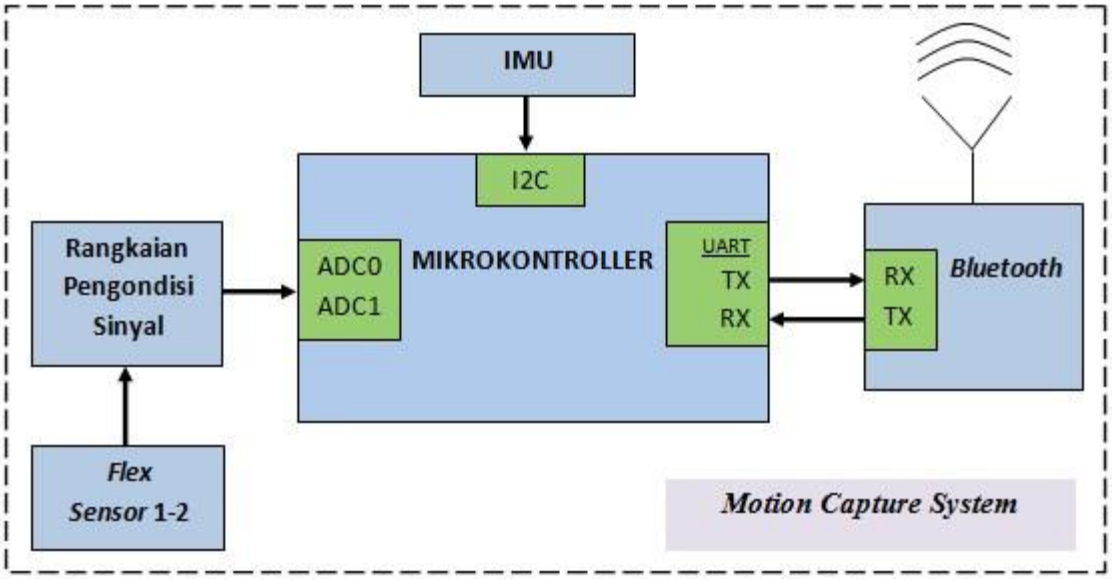
Gambar 4. 2 Perancangan Mekanik Jari-Jari Robot.

Pada Gambar 4.2 motor sevo akan menggerakkan mekanik dari jari-jari robot. Sehingga setiap *joint*akan membentuk sudut tertentu sesuai dengan sudut dari motor servo yang akan diberikan. Pembuatan mekanik jari-jari robot semuanya berbahan mika *acrylic* 3 mm.

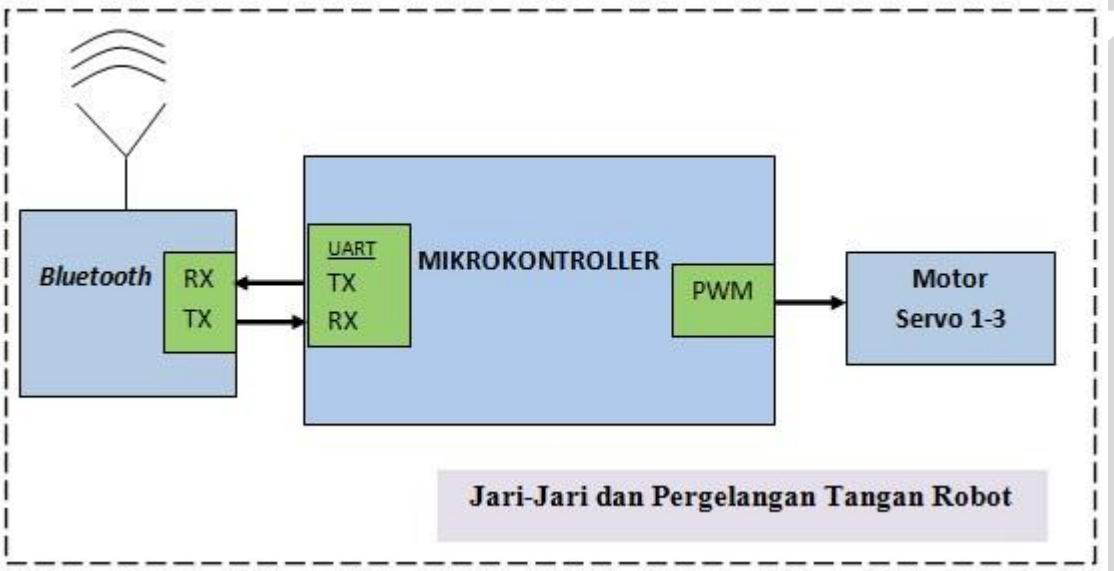
Terdapat tiga sudut yang dihasilkan oleh jari-jari robot yaitu θ1 yang merupakan sudut dari lengan-1, θ2 merupakan sudut dari lengan-2, dan θ3 merupakan sudut dari lengan-3. Jika masing-masing lengan dihubungkan maka akan membentuk jari-jari robot dengan titik ujung pada jari-jari robot akan menunjukan kordinat (x,y) atau bisa juga menunjukan sudut. Perhitungan kordinat (x,y) dapat dilakukan dengan dengan menggunakan persamaan kinematika tiga sendi (3 DOF) trigonometri, seperti ditunjukan pada persamaan (2-10) dan persamaan (2-11).

4.1.2. Perancangan Perangkat Keras

Perancangan perangkat keras lebih ditekankan kepada perancangan antarmuka dari masing-masing blok seperti antarmuka sensor dengan mikrokontroller dan antarmuka *actuator* dengan mikrokontroller. Selain itu, dilakukan juga perancangan *main board* mikrokontroller pada alat *motion capture* dan *main board* mikrokontroller pada robot. Diagram blok sistem secara keseluruhan dapat ditunjukkan pada Gambar 4.3 dan Gambar 4.4



Gambar 4. 3 Diagram Blok Alat Motion Capture



Gambar 4. 4 Diagram Blok Jari-jari dan Pergelangan Tangan Robot

Diagram blok sistem secara garis besar terbagi dalam dua bagian yaitu diagram blok alat *motion capture system* yang digunakan untuk merekam pergerakan dari pergelangan tangan pengguna sampai jari-jari pengguna dan diagram blok alat jari-jari dan pergelangan tangan robot yang digunakan sebagai objek yang menampilkan hasil perekaman gerakan yang dilakukan oleh alat *motion capture*.

Dalam penelitian ini akan dirancang alat *motion capture system* berbasis *flex sensor* dan sensor IMU.*Flex sensor* dan sensor IMU diletakkan pada sebuah sarung tangan

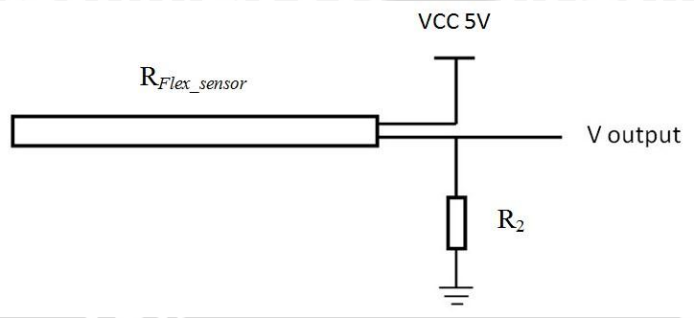
sehingga memudahkan dalam proses perekaman gerakan dari pengguna. Rangkaian *Flex sensor* yang menghasilkan tegangan analog akan dirubah menjadi tegangan digital dengan bantuan *peripheral* ADC (*Analog to Digital Converter*) yang terdapat pada mikrokontroller. Data ADC yang telah didapatkan digunakan untuk menentukan sudut dari pergarakan jari-jari pengguna. Pada punggung sarung tangan di letakkan sensor IMU yang digunakan untuk merekam kemiringan dari punggung tangan, kemiringan yang direkam yaitu ketika punggung tangan melakukan gerkan rotasi ke arah kanan dan rotasi ke arah kiri sehingga menghasilkan sudut tertentu. Perekaman kemiringan bisa dilakukan oleh sensor IMU yaitu sensor *gyroscope* dan *accelerometer* yang sudah terintegrasi dalam satu *chip*. *Peripheral* yang digunakan dalam antarmuka sensor tersebut yaitu I2C (*Inter Integrated Circuit*) dengan pin SCL (*Serial Clock*) dan pin SDA (*Serial Data*) yang membawa informasi dari sensor ke mikrokontroller. Pengolahan data sensor IMU menjadi sudut kemiringan digunakan *complementary filter* yang merupakan gabungan dari sensor *gyroscope* dan *accelerometer*. Setelah mendapatkan sudut dari masing-masing sensor, mikrokontroller pada *motion capture*akan mengirimkan data sudut tersebut menggunakan *bluetooth*.

Dari diagram blok robot pada Gambar 4.4, *bluetooth* HC-05 sebagai *slave* akan menerima data sudut dari *bluetooth* master. Data sudut yang telah didapatkan akan di terjemahkan menjadi sinyal PWM (*Pulse Width Modulation*) untuk mengontrol putaran sudut dari motor DC servo sehingga apabila sumbu putar dari motor servo berputar dalam sudut tertentu maka akan menggerakkan mekanik dari jari-jari robot sehingga menghasilkan sudut sesuai dengan sudut jari-jari pengguna.

4.1.2.1. Perancangan Rangkaian *Flex Sensor*

Flex sensor merupakan sensor yang dapat memiliki perubahan resistansi ketika sensor tersebut ditekuk pada sudut tertentu. Resistansi yang dihasilkan oleh sensor berbanding lurus terhadap sudut tekuk dari sensor. Semakin besar sudut tekuk dari sensor maka resistansi yang dihasilkan oleh sensor akan semakin besar, begitupun sebaliknya semakin kecil sudut tekuk dari sensor maka resistansi dari sensor juga akan semakin kecil. Prinip kerja *flex sensor* ini memiliki kesamaan dengan prinsip kerja potensiometer yaitu dapat menghitung sudut *wiper* potensiometer dengan memperhatikan perubahan resistansi yang dihasilkan potensiometer ketika *wiper* diputar ke arah kanan atau kiri. *Flex sensor* memiliki desain yang sederhana, memiliki dua kaki seperti resistor, dan memiliki resistansi 10KΩ ± 30%. Karena memiliki karakteristik seperti resistor maka sensor ini membutuhkan

rangkaian tambahan untuk menghasilkan *output* sesuai yang diinginkan. Rangkaian pengondisi sinyal *flex sensor* yang akan digunakan yaitu rangkaian pembagi tegangan dengan memparalel *flex sensor* dengan resistor terhadap output rangkaian sensor sehingga menghasilkan tegangan analog pada sisi keluaran rangkaian *flex sensor*. Rangkaian *flex sensor* dapat ditunjukkan pada Gambar 4.5.



Gambar 4. 5 Rangkaian Flex Sensor

Untuk menghasilkan tegangan output yang diinginkan, maka perlu untuk merancang resistor yang tepat agar ketika *flex sensor* dalam keadaan 0^o dengan resistansi minimum dan pada saat 180^o dengan resistansi maksimum didapatkan *range* tegangan *output* yang besar sehingga proses untuk mengkonversi sudut dari *flex sensor* terhadap perubahan tegangan *output*akan semakin mudah. Hasil pengukuran resistansi terhadap sudut tekuk *flex sensor* dapat ditunjukkan pada Table 4.1. Pengukuran resistansi menggunakan alat ukur multimeter.

Tabel 4. 1 Pengukuran Resistansi Sensor Terhadap Sudut

Sudut Flex Sensor (^{o})	Resistansi (Ω)
0	9850
180	27700

Berdasarkan hasil pengukuran tersebut, resistansi minimum dan resistansi maksimum dapat dijadikan salah satu dasar dalam penentuan resistor yang akan digunakan. Selain itu, tegangan refrensi analog yang dapat dikonversi oleh mikrokontroler menjadi sinyal digital yaitu sebesar 3 volt. Oleh karena itu, resistansi dan tegangan refrensi akan dijadikan dasar perancangan resistor dalam rangkaian pembagi tegangan tersebut. Persamaan yang digunakan yaitu :

$$Vout = \frac{R2}{R\ flex\ sensor+R2} \times Vcc \quad (4\text{-}1)$$

Vout adalah tegangan *output* dari rangkaian *flex sensor*, R_2 adalah resistor yang akan dicari, $R_{flex\ sensor}$ adalah resistansi yang dihasilkan oleh *flex sensor*, dan Vcc adalah tegangan input dari rangkaian sensor. Hasil perhitungan resistor menggunakan rumus pada persamaan (4-1) dapat ditunjukkan pada Tabel 4.2.

Tabel 4. 2 Perhitungan Nilai Resistansi Pada Rangkaian Flex Sensor

Vcc(V)	R2(Ω)	Resistansi Flex Sensor		vout (Rmin) $(\frac{R2}{Rmin+R2} xVcc)$ (V)	Vout(Rmax) $(\frac{R2}{Rmax+R2} xVcc)$ (V)	Range Vout (Vout(Rmin)-Vout(Rmax)) (V)
		Rmin(Ω)	Rmax(Ω)			
5	10000	9850	27700	2.5189	1.3263	1.1926
5	11000	9850	27700	2.6379	1.4212	1.2167
5	12000	9850	27700	2.7460	1.5113	1.2347
5	13000	9850	27700	2.8446	1.5971	1.2476
5	14000	9850	27700	2.9350	1.6787	1.2564
5	**15000**	**9850**	**27700**	**3.0181**	**1.7564**	**1.2617**
5	16000	9850	27700	3.0948	1.8307	1.2641
5	17000	9850	27700	3.1657	1.9016	1.2642
5	18000	9850	27700	3.2316	1.9694	1.2622
5	19000	9850	27700	3.2929	2.0343	1.2586
5	20000	9850	27700	3.3501	2.0964	1.2536
5	21000	9850	27700	3.4036	2.1561	1.2475
5	22000	9850	27700	3.4537	2.2133	1.2404
5	23000	9850	27700	3.5008	2.2682	1.2325
5	24000	9850	27700	3.5451	2.3211	1.2240
5	25000	9850	27700	3.5868	2.3719	1.2149

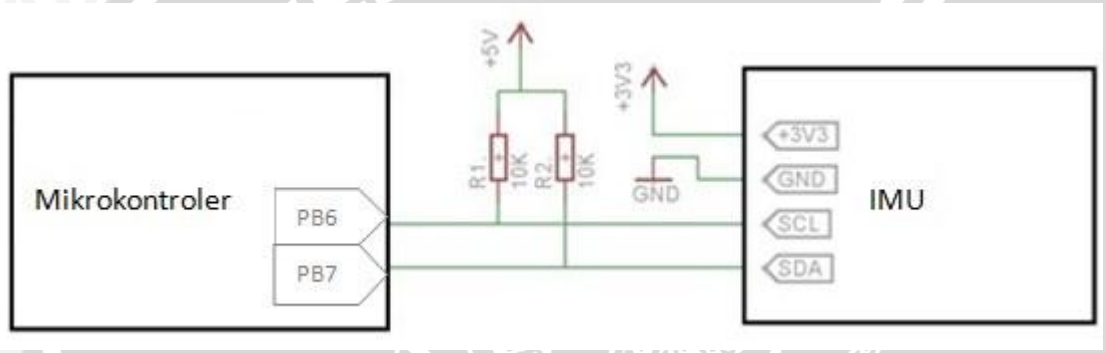
Resistor (R_2) yang akan digunakan dalam rangkaian *flex sensor* ini yaitu resistor 15 kΩ. pemilihan resistor tersebut berdasarkan tegangan keluaran yang dihasilkan sensor tidak jauh melebihi tegangan analog yang dapat dikonversi oleh mikrokontroler yaitu 3.0181 volt sampai 1,7564 volt, selain itu resistor tersebut mudah di dapatkan dipasaran dan merupakan salah satu yang memiliki *range* tegangan keluaran yang besar yaitu 1,2617 volt.

Keluaran dari rangkaian *flex sensor* adalah tegangan analog, sehingga untuk memudahkan perhitungan sudut perekaman dari pergerakan jari-jari pengguna digunakan *peripheral* ADC (*Analog to Digital Converter*) yaitu yang mengubah sinyal analog

menjadi sinyal digital yang terdapat pada mikrokontroler. ADC yang digunakan dalam mikrokontroler tersebut adalah ADC3 yaitu *Channel* 10 yang terdapat pada PC0 (*Pin* C0) dan *Channel* 11 yang terdapat pada PC1 (*Pin* C1).

4.1.2.2. Perancangan Rangkaian IMU (*Inertial Measurement Unit*)

Sensor ini memiliki kaki VCC (sumber tegangan), GND (*ground*), SCL (*serial clock*), SDA (*serial data*), XDA, XCL, AD0, dan INT (*interrupt*). Pin yang akan digunakan dalam perancangan ini yaitu VCC, GND, SCL, SDA, dan AD0. Sumber tegangan yang dibutuhkan oleh sensor IMU adalah 3,3 volt yang dapat diperoleh dari output regulator 3,3V. Antarmuka sensor IMU dengan mikrokontroler dapat ditunjukkan pada Gambar 4.6.



Gambar 4. 6 Antarmuka Sensor IMU dengan mikrokontroler

Antarmuka sensor IMU dengan mikrokontroler menggunakan pin SCL (serial clock) dan SDA (serial data). Pada mikrokontroler pin SCL yang digunakan yaitu pin B6 (PB6) dan pin SDA yang digunakan yaitu pin B7 (PB7), kedua pin ini merupakan *peripheral* dalam I2C1. Dalam antarmuka pada Gambar 3.7 terdapat *pull-up* resistor yang digunakan untuk menghilangkan *floating* (mengambang) atau keadaan tidak tentu yang menyebabkan data yang dikirm tidak utuh. *Pull-up* resistor terhubung ke sumber tegangan 5 volt. Digunakan resistor 10 KΩ agar *pull-up* yang digunakan berfungsi sebagai *weak pull-up* yang artinya *pull-up* yang digunakan sedikit mengalirkan arus.

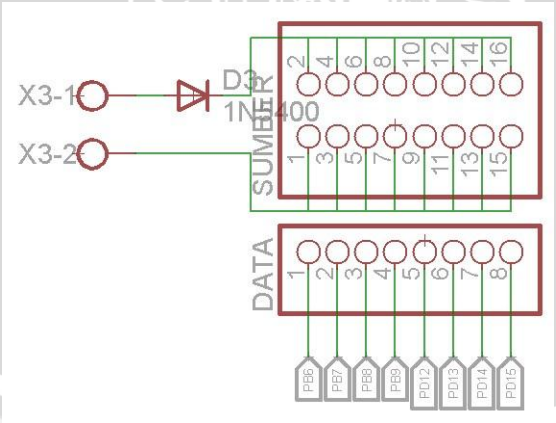
4.1.2.3. Rangkaian *Bluetooth*

Pin yang akan digunakan dalam perancangan ini yaitu pin RXD, TXD, GND, dan VCC. Modul *bluetooth* ini merupakan modul *bluetooth* yang siap pakai sehingga tidak membutuhkan rangkaian tambahan dalam melakukan perancangan. Antarmuka modul *bluetooth* dengan mikrokontroler hanya menggunakan *header female* 1×6 dalam antarmuka

modul *bluetooth* HC-05 dengan mikrokontroler karena modul *bluetooth* ini didesain sudah siap pakai. Dalam perancangan digunakan *header female* 1×6 agar modul *bluetooth* dapat langsung dihubungkan ke *shield board* sehingga tidak memerlukan kabel lagi dalam antarmuka tersebut. Sumber tegangan 5V *bluetooth* dihubungkan ke keluaran dari *regulator* 5V (2A) DC-DC *step-down*, *ground* dari *bluetooth* dihubungkan ke *ground* dari mikrokontroler. RXD dihubungkan ke pin D8 (PD8) yang berfungsi sebagai *transmitter* (TX) dari mikrokontroler, dan TXD dihubungkan ke pin D9 (PD9) yang berfungsi sebagai *receiver* (RX) dari mikrokontroler.

4.1.2.4. Perancangan Rangkaian Motor DC Servo

Motor DC servo memiliki tiga pin yaitu pin data yang digunakan untuk mengontrol pergerakan dari motor DC servo dengan memberikan pulsa pada periode 20 ms, pin Vcc yang digunakan sebagai sumber tegangan dari motor DC servo, dan pin *ground*. Ada dua tipe motor DC servo yang digunakan yaitu *Tower Pro* MG995 dan EMAX ES08MD. Sumber tegangan yang dibutuhkan oleh motor DC servo MG995 yaitu 4,8 volt sampai 7,2 volt dan motor DC servo ES08MD yaitu 4,8 volt sampai 6,0 volt, sehingga sumber tegangan yang akan digunakan pada kedua motor servo tersebut yaitu 6,0 volt yang didapatkan dari regulator *adjustable* DC-DC *step down* dengan arus keluaran 5A. Rangkaian antarmuka motor DC servo dengan mikrokontroler dapat ditunjukkan pada Gambar 4.7.



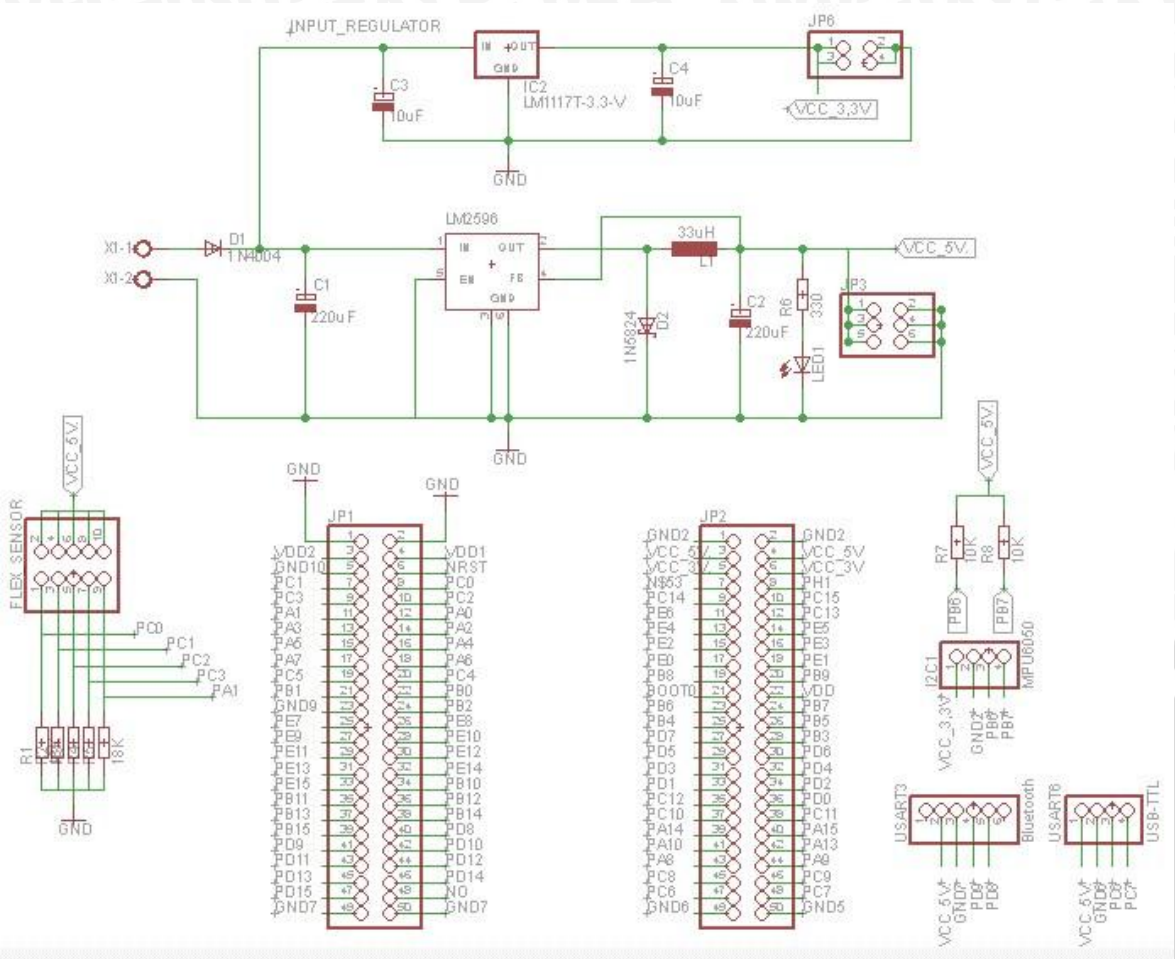
Gambar 4. 7 Antarmuka Motor DC Servo dengan Mikrokontroler

Diode pada rangkaian Gambar 4.7 digunakan untuk menjaga rangkaian tetap aman ketika tejadi pertukaran polaritas antara positif dengan negatif, *header male* 2×8 yang terhubung dengan *regulator* dengan arus keluaran 5A yang digunakan sebagai sumber tegangan kedua motor DC servo, dan *header male* 1×8 yang terhubung ke pin B6, B7, B8,

B9, D12, D13, D14, dan D15 pada mikrokontroler. Pin pada data tersebut merupakan pin pada timer 4.

4.1.2.5. Perancangan Board Utama *Motion Capture System*

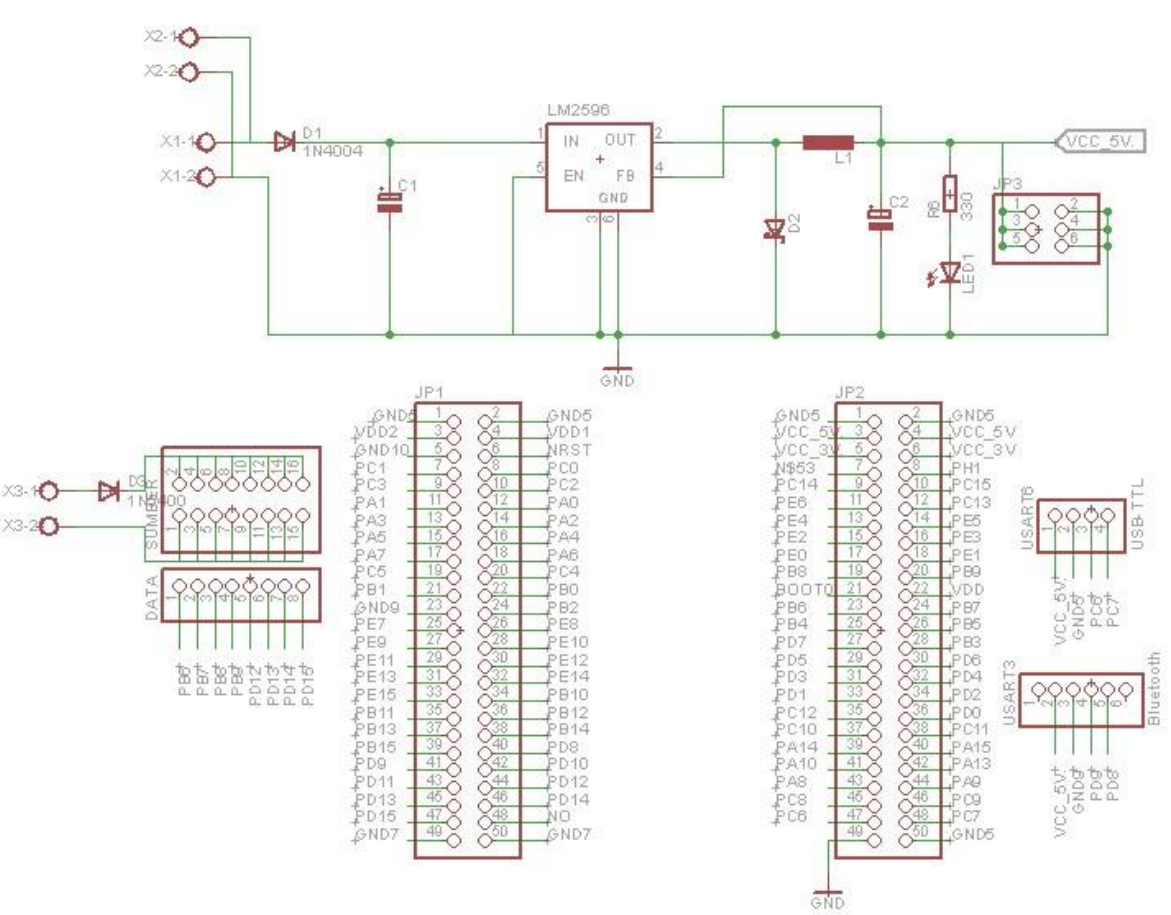
Main board pada perancangan ini memiliki fitur-fitur yang digunakan untuk melakukan *interface* dengan perangkat lain. Fitur fitur yang disediakan dalam *main board* tersebut yaitu satu *terminal screw* yang digunakan untuk menghubung baterai lipho (*lithium polimer*) 11,1 V sebagai sumber tegangan ke rangkaian *main board*, satu rangkaian *regulator* DC-DC *fixed step-down* 5 volt, satu rangkaian *regulator* DC-DC *step-down* 3,3V, pin *male header* 2×8 digunakan untuk berkomunikasi dengan *flex sensor*, pin *male header* 1×4 yang terhubung dengan *pull-up* resistor digunakan untuk berkomunikasi dengan sensor IMU, pin *female header* 1×6 untuk berkomunikasi dengan *bluetooth*, pin *male header* 1×4 untuk berkomunikasi dengan perangkat lain yang menggunakan komunikasi UART (*Universal Asynchronous Receiver/Transmitter*), dan dua *female header* 2×25 untuk menghubungkan mikrokontroler dengan semua perangkat yang ada di *main board*. Fungsi dari masing-masing blok dari rangkaian telah dijelaskan pada perancangan sebelumnya. Skematik rangkaian *motion capture system main board* dapat ditunjukkan pada Gambar 4.8.



Gambar 4. 8 Skematik Rangkaian Main Board*Motion Capture System.*

4.1.2.6. Perancangan Board Utama Pergelangan Tangan Robot

Fitur-fitur yang tersedia dalam *main board* pergelangan tangan ini yaitu dua *terminal screw* yang menghubungkan baterai lipho (*lithium polimer*) 11,1 V dengan rangkaian *main board*, satu rangkaian *regulator* DC-DC *step-down*, satu pin *male header* 2×8 untuk catu seluruh motor DC *servo* yang digunakan, satu pin *male header* 1×8 untuk data motor DC *servo*, satu pin *female header* 1×6 untuk *interface* dengan *bluetooth*, satu *male header* 1×4 digunakan untuk berkomunikasi dengan perangkat tambahan dengan komunikasi UART (*Universal Asynchronous Receiver/Transmitter*), dan dua pin *female header* 2×25 digunakan untuk menghubungkan mikrokontroler dengan perangkat yang terhubung dengan *main board* pergelangan dan jari-jari tangan robot. Skematik rangkaian dari main board pergelangan tangan robot dapat ditunjukan pada Gambar 4.9.



Gambar 4. 9 Sekamtik Rangkaian Board Utama Pergelangan Tangan Robot

4.1.3. Perancangan Perangkat Lunak

4.1.3.1. Perancangan Perangkat Lunak Utama

Perancangan perangkat lunak secara keseluruhan dibagi menjadi dua bagian yaitu perangkat lunak *motion capture system* dan perangkat lunak pergelangan tangan robot. Setelah penentuan spesifikasi sistem, maka tahap selanjutnya yaitu pengembangan perangkat lunak yang merupakan proses pengubahan spesifikasi sistem menajadi sistem yang dapat dijalankan. Tahap ini yaitu mencangkup perancangan dan pemrograman perangkat lunak. Perancangan perangkat lunak merupakan deskripsi struktur yang akan diimplemetasikan berupa data-data yang diolah dalam sistem, interface antar komponen elektronik pendukung, dan penentuan algoritma yang tepat dalam sistem. Proses perancangan perangkat lunak secara umum akan membahas tiga hal yaitu :

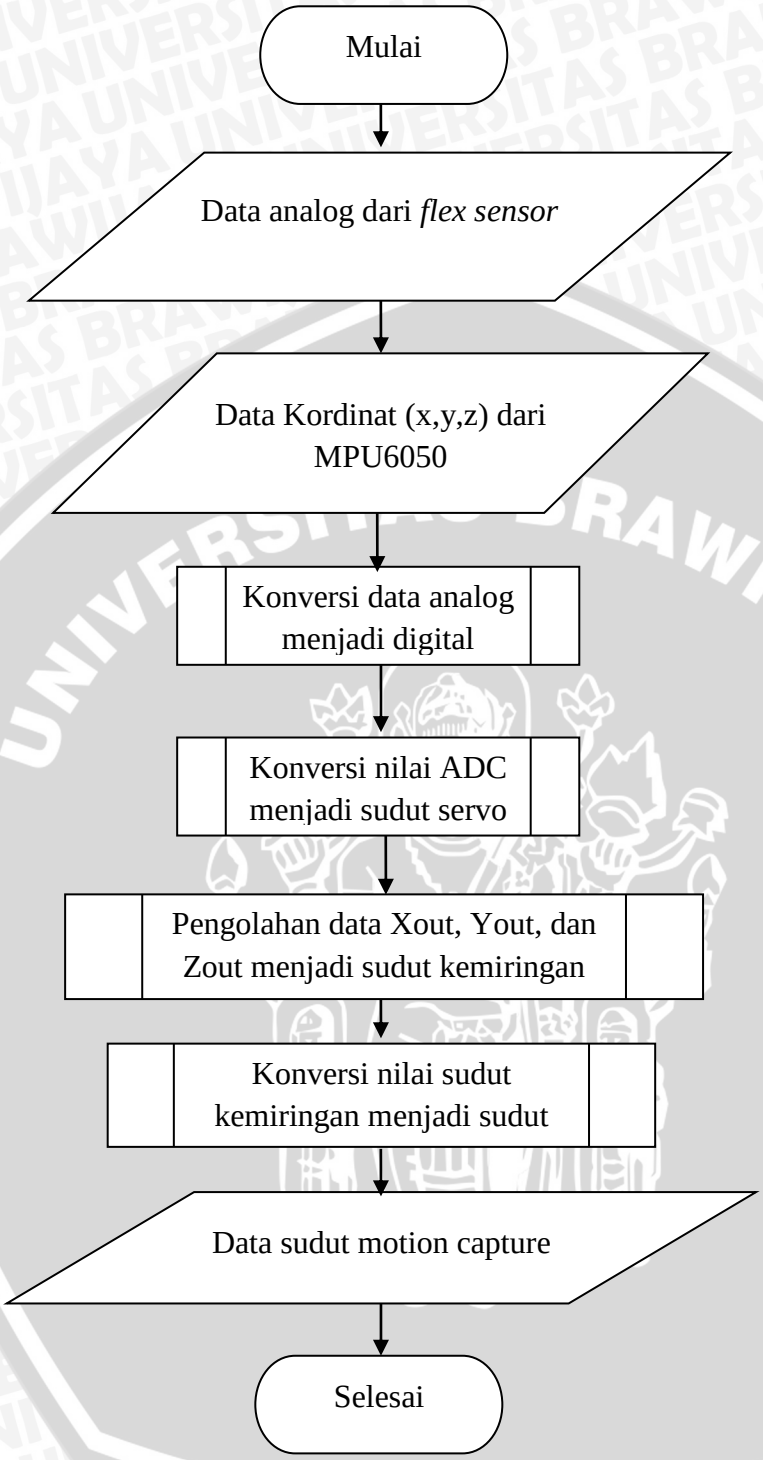
1. Perancangan *interface*, yaitu perancangan komunikasi antara mikrokontroller dengan sub-sub sistem seperti sensor, aktuator, dan alat transmisi data.

2. Perancangan struktur data, yaitu proses pengolahan data secara terprogram dengan melakukan perhitungan data yang didapatkan dari sensor agar dapat dirubah menjadi sudut.
3. Perancangan algoritma, yaitu untuk memberikan perincian terhadap sistem yang akan dibangun.

Metode yang digunakan dalam perancangan perangkat lunak ini yaitu metode berorientasi objek yaitu dengan menentukan model hubungan statis dan dinamis objek, dan bagaimana objek berintraksi satu sama lain ketika objek sedang dijalankan. Dengan metode ini, memungkinkan perancangan dan pengolahan data dapat dilakuakn berdasarkan data yang didapatkan dari objek. Data yang didapatkan juga dapat dibandingkan dengan spesifikasi yang telah dituliskan oleh perusahaan komponen tersebut.

4.1.3.1.1.Perancangan Perangkat Lunak Utama *Motion Capture System*.

Perancangan ini dilakukan untuk mengimplemetasikan secara terprogram setiap sub-sub diagram blok yang telah dirancang pada Gambar 4.3. perancangan perangkat lunak *motion capture* secara garis besar meliputi pengenalan port yang akan digunakan dengan melakukan sfesifikasi port tersebut sesuai dengan kebutuhan yang diinginkan misalnya port ADC (*Analog to Digital Converter*), port I2C (*Inter Integrated Circuit*), dan port UART (*Universal asynchronous Receiver/Transmitter*), melakukan konversi nilai ADC ke sudut servo, melakukan konversi sudut kemiringan ke sudut servo, dan melakukan perhitungan dalam setiap konversi yang dilakukan. Diagram alir perangkat lunak utama *motion capture system* dapat ditunjukkan pada Gambar 4.10.



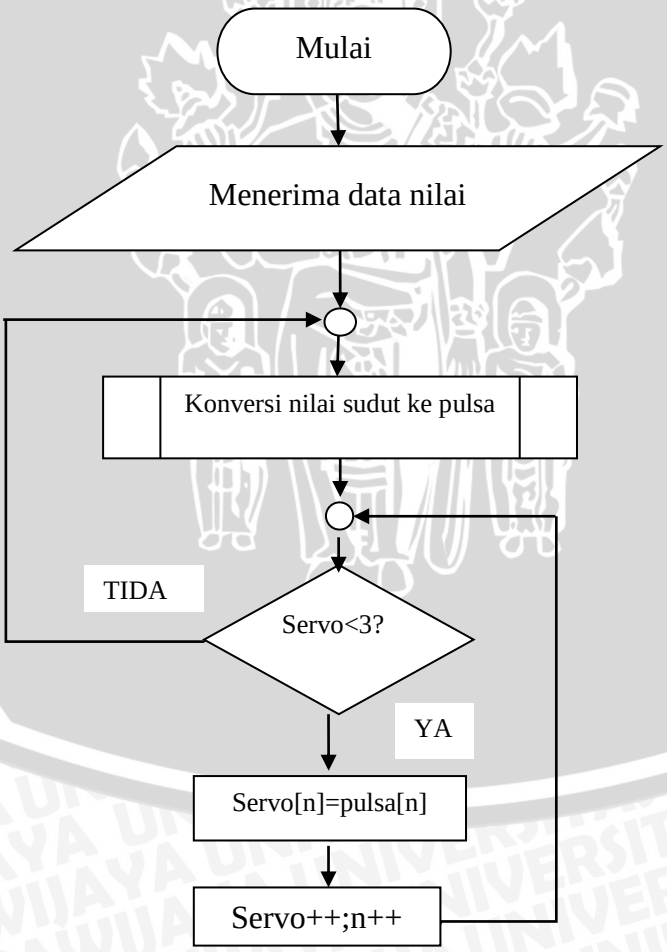
Gambar 4. 10 Diagram alir perangkat lunak utama motion capture system.

Pada Gambar 4.10 secara umum keluaran dari rangkaian *flex sensor* merupakan data analog sehingga dalam *interface* dengan mikrokontroller diguanakan *peripheral* ADC untuk mengubah data analog menjadi data digital. Nilai ADC yang telah didapatkan akan dirubah menjadi sudut servo yang digunakan untuk menggerakkan jari-jari dari robot.

Sensor IMU memberikan data kepada mikrokontroler dengan komunikasi I2C (*Inter Integrated Circuit*) berupa data (x,y,z) yang kemudian diolah menjadi sudut kemiringan dengan menggunakan metode *complementary filter* yang merupakan gabungan dari sensor *gyroscope* dan sensor *accelerometer*. Setelah mendapatkan sudut kemiringan maka akan dirubah menjadi sudut servo yang digunakan untuk menggerakkan servo pergelangan tangan robot. Setelah mendapatkan sudut servo dari konversi nilai ADC dan konversi sudut kemiringan maka data akan dikirimkan ke robot untuk menggerakkan robot sesuai dengan hasil perekaman dari *motion capture system*.

4.1.3.1.2.Perancangan Perangkat Lunak Utama Robot

Pada perancangan perangkat lunak utama robot meliputi *interface* antara mikrokontroler dengan *bluetooth* dan *interface* mikrokontroler dengan motor DC servo. Diagram perancangan perangkat lunak utama robot dapat ditunjukkan pada Gambar 4.11.



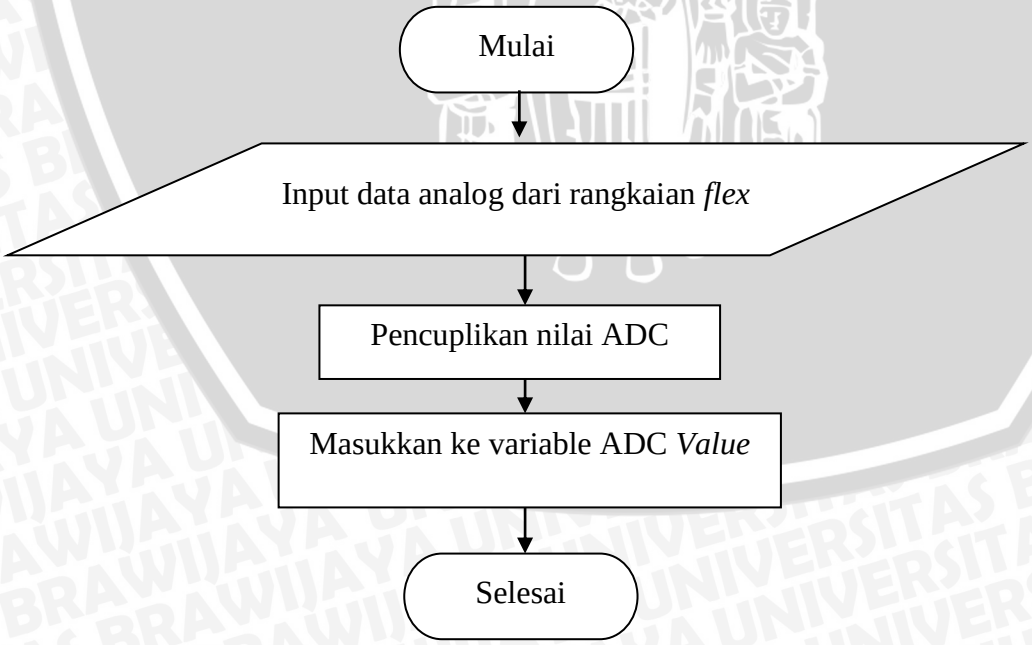
Gambar 4. 11 Diagram Alir Pergelangan dan Jari-Jari Tangan Robot.

Pada perangkat lunak robot, proses diawali dengan penerimaan data sudut oleh *bluetooth slave* dari *alat motion capture system*. *Interface* dengan bluetooth menggunakan

komunikasi UART (*Universal Asynchronous Receiver*/*Transmitter*). Data sudut yang diterima oleh robot digunakan untuk menggerakkan aktuator dari robot, aktuator yang digunakan yaitu motor DC servo. Aktuator tersebut dapat dikontrol menggunakan timer PWM pada periode 20 ms. Untuk mengontrol motor DC servo nilai sudut yang telah didapatkan akan dikonversi menjadi lebar pulsa yang dapat menggerakkan motor DC servo. Ada tiga motor DC servo yang digunakan yaitu satu motor DC servo yang terletak pada pergelangan tangan robot dan dua motor DC servo yang terletak pada jari-jari tangan robot. Proses penerimaaan data dilakukan secara *continue* sehingga robot akan terus mengikuti perekaman gerakan dari alat *motion capture system*.

4.1.3.2. Perancangan Perangkat Lunak ADC

ADC (*Analog to Digital Converter*) merupakan *peripheral* dalam mikrokontroler yang dapat merubah data analog menjadi data digital. ADC digunakan pada rangkaian *flex sensor* karena keluaran dari rangkaian *flex sensor* merupakan tegangan analog. Ada tiga ADC 16 bit yang disediakan oleh mikrokontroler dan setiap ADC memiliki 16 *external channel*. ADC pada mikrokontroler juga disediakan menggunakan *timer interrupt* yaitu menggunakan *timer* 1, *timer* 2, *timer* 3, *timer* 4, *timer* 5, dan *timer* 8. Diagram alir perancangan perangkat lunak ADC dapat ditunjukkan pada Gambar 4.12.



Gambar 4. 12 Diagram alir perangkat lunak ADC.

Pada Gambar 4.12 dimulai dengan menerima data analog dari keluaran rangkaian *flex sensor*, data analog akan dicuplik dengan *timer* pada frekuensi 2 MHz, frekuensi dapat berubah sesuai dengan kecepatan cuplik yang diinginkan, namun pada perancangan ini dinginkan pencuplikan sebesar 2 MHz sehingga periode pencuplikan data analog yaitu sebesar 0,5 ms. *Timer* yang digunakan yaitu *timer* 2, dengan *prescaler* 999 dan periode 41. Hasil pencuplikan akan dimasukkan kedalam variable ADC value. ADC yang digunakan dalam perancangan ini yaitu ADC3 10-bit sehingga nilai sampling maksimal yaitu 1024. Proses sampling dari ADC dapat diketahui melalui persamaan (4-2) berikut :

$$ADC = \frac{Vout\ sensor}{Tegangan\ refrensi} \times 2^{n-1} \quad (4\text{-}2)$$

Tegangan refrensi yang digunakan yaitu 3 volt karena tegangan maksimal yang dapat dikonversi oleh mikrokontroler menjadi nilai ADC adalah 3 volt. Pada Persamaan (4-2), n merupakan jumlah bit yang digunakan, karena bit yang digunakan yaitu 10 bit maka n sama dengan 10 sehingga 2^{n-1} sama dengan 1023. Maka persamaannya menjadi :

$$ADC = \frac{Vout\ sensor}{3} \times 1023 \quad (4\text{-}3)$$

Vout sensor merupakan keluaran dari rangkaian *flex sensor* berupa tegangan analog. Nilai n dapat berubah sesuai dengan jumlah bit yang digunakan.

4.1.3.3. Perancangan Konversi Nilai ADC Menjadi Sudut Servo.

Flex sensor memiliki resistansi yang berubah-ubah dalam sudut tertentu. Perubahan sudut dapat dilakukan dari 0^o sampai 180^o. sudut 0^o dapat dijadikan sebagai batas bawah dan sudut 180^o dapat dijadikan sebagai batas atas dari suatu persamaan. Diasumsikan bahwa diantara range 0^o sampai 180^o memiliki perubahan resistansi yang linier tehadap sudut *flex sensor* tersebut. Oleh karena itu untuk melakukan konversi nilai ADC menjadi sudut servo dapat digunakan persamaan garis linier yaitu :

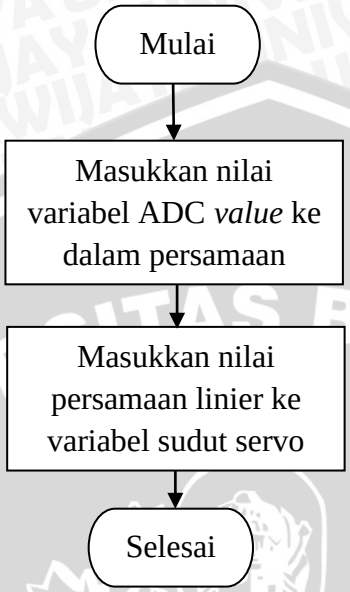
$$y = m.x + c \quad (4\text{-}4)$$

Pada Persamaan (4-4), y merupakan keluaran dari proses konversi yaitu sudut servo, x merupakan masukan dari proses konversi yaitu nilai ADC, m merupakan gradient, dan c merupakan konstanta. Dengan adanya batas bawah dan batas atas dari persamaan maka batas bawah dan batas atas dapat langsung digunakan dalam persamaan yaitu :

$$Y = mx_o + c \quad (4\text{-}5)$$

$$Y = mx_1 + c \quad (4\text{-}6)$$

Diagram alir perancangan perangkat lunak dalam proses konversi nilai ADC menjadi sudut servo dapat ditunjukkan pada Gambar 4.13.



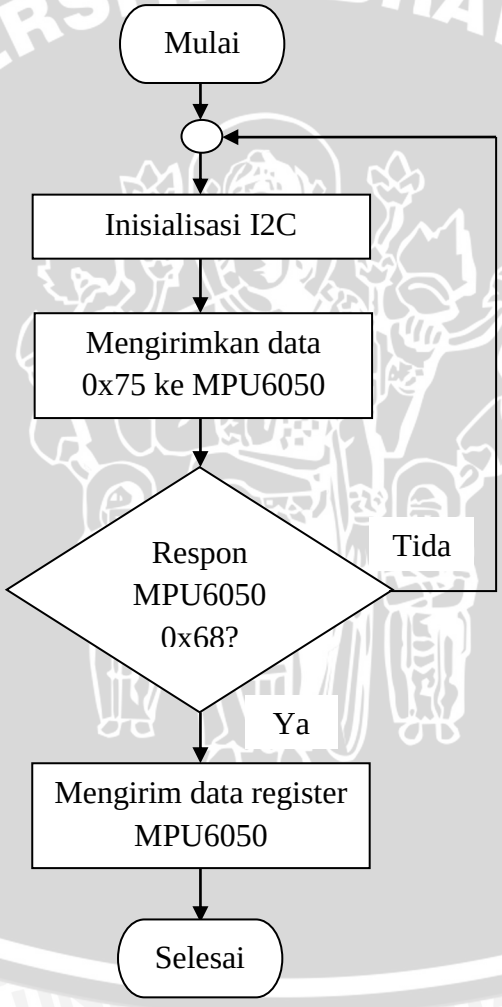
Gambar 4. 13 Diagram Alir Perangkat Lunak Konversi Nilai ADC menjadi Sudut Servo

Pada Gambar 4.13 dimulai dengan memasukkan nilai dari variabel ADC *value* kedalam persamaan (4-4) kemudian setelah dilakukan perhitungan maka nilai dari hasil perhitungan dimasukkan ke variabel sudut servo yang nantinya akan dikirim ke robot sebagai sudut dari actuator motor DC servo.

4.1.3.4. Perancangan *Interface* IMU

IMU merupaka perangkat yang mengguanakan komunikasi I2C (*Inter Integrated Circuit*). I2C menggunakan dua jalaur komunikasi yaitu SCL (*Serial Clock*) yang merupakan jalur yang digunakan untuk mensinkronisasi transfer data pada jalur I2C dan SDA (*Serial Data*) yang merupakan jalur dari data yang dikirim. Frekuensi yang digunakan dalam modul ini yaitu frekuensi *fast mode* 400 KHz. I2C pada modul ini selalu bertindak sebagai *slave* ketika melakukan komunikasi dengan perangkat lain, artinya modul ini akan merespon jika diminta oleh *processor* master. Jalur SCL hanya bisa dikontrol oleh *master* sehingga inisialisasi frekuensi jalur SCL berada pada *processor master*. Jalur pada I2C membutuhkan *pull-up* resistor sehingga jalur pada I2C menjadi *open drain* yang artinya perangkat hanya perlu memberikan logia output *low* untuk membuat jalur menjadi *low* karena untuk logika *high* sudah di *cover* oleh *pull-up* resistor.

Pada Gambar 4.14 dimulai dengan inisialisasi program I2C pada mikrokontroler yang umumnya seperti I2C berapa yang digunakan, *port* untuk I2C yang digunakan, kecepatan pengiriman data, pengaturan proses tulis I2C, dan pengaturan proses baca I2C. setelah inisialisasi selesai, untuk mencoba apakah jalur I2C sudah bisa bekerja maka dilakukan pengiriman *register address* 0x75 ke IMU, apabila I2C bekerja maka sensor IMU akan mengirimkan data 0x68 kepada I2C *master*. Dalam proses tersebut pin *enable* pada sensor IMU dimasukkan ke *ground*. Jika tidak mendapatkan balasan dari sensor IMU maka perlu mengatur inisialisasi kembali dan jika jalur I2C sudah berhasil maka perlu mengirimkan data *register* untuk mengatur sensor IMU sesuai kinerja yang diinginkan.Diagram alir perancangan *interface* IMU dapat ditunjukkan pada Gambar 4.14.



Gambar 4. 14 Perancangan Interface IMU.

4.1.3.5. Perancangan Konversi Data IMU Menjadi Sudut Servo.

Dalam pembuatan program, untuk menentukan sudut dari *accelerometer* menggunakan Persamaan (2-16) dan Persamaan (2-17), keluaran dari persamaan tersebut berupa radian sehingga untuk merubah ke sudut menggunakan persamaan sebagai berikut :

$$sudut(derajat) = \frac{sudut\ (radian)}{\pi} \times 180 \quad (4\text{-}7)$$

$$sudut(derajat) = sudut\ (radian) \times 57{,}295 \quad (4\text{-}8)$$

Sehingga dalam kode program dapat dituliskan seperti berikut :

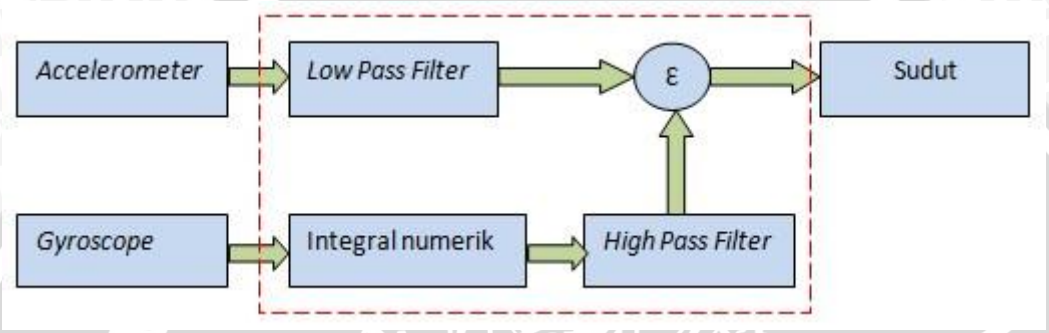
```
acc_x_angle = (float) 57.295 * atan((float)acc_y /
              sqrt(pow((float)acc_z,2)+pow((float)acc_x,2)));

acc_y_angle = (float) 57.295 * atan((float)acc_x/
              sqrt(pow((float)acc_z,2)+pow((float)acc_y,2)));
```

Penentuan sudut dari gyroscope menggunakan metode integrasi forward-Euler seperti pada Persamaan (2-28). Dalam kode program dapat dituliskan sebagai berikut :

```
Read_I2C(MPU6050_ADDRESS,MPU6050_RA_GYRO_XOUT_H,&giro_xh);
Read_I2C(MPU6050_ADDRESS,MPU6050_RA_GYRO_XOUT_L,&giro_xl);
Read_I2C(MPU6050_ADDRESS,MPU6050_RA_GYRO_YOUT_H,&giro_yh);
Read_I2C(MPU6050_ADDRESS,MPU6050_RA_GYRO_YOUT_L,&giro_yl);
Read_I2C(MPU6050_ADDRESS,MPU6050_RA_GYRO_ZOUT_H,&giro_zh);
Read_I2C(MPU6050_ADDRESS,MPU6050_RA_GYRO_ZOUT_L,&giro_zl);
giro_x = ((giro_xh<<8)|giro_xl)- giro_x_offset;
giro_y = ((giro_yh<<8)|giro_yl)- giro_y_offset;
giro_z = ((giro_zh<<8)|giro_zl)- giro_z_offset;
giro_rate_x = giro_x/giro_x_sens;
giro_rate_y = giro_y/giro_y_sens;
giro_rate_z = giro_z/giro_z_sens;
gyro_x_angle +=(float)(giro_rate_x*dt);
gyro_y_angle +=(float)(giro_rate_y*dt);
gyro_z_angle +=(float)(giro_rate_z*dt);
```

Dalam penerapannya sensor *accelerometer* dan *gyroscope* memiliki kelemahan jika digunakan secara terpisah-pisah. Dalam perhitungan sudut kemiringan oleh

*accelerometer*memiliki respon yang lambat sedangkan perhitungan sudut kemiringan yang dilakukan oleh *gyroscope* memiliki *noise* yang besar jika digunakan dalam jangka waktu yang lama. Oleh karena itu perlu adanya penggabungan untuk dapat menutupi kekurangan satu sama lain diantara sensor tersebut, maka digunakanlah salah satu metode filter sinyal yaitu *complementary filter*. Fungsi dari *complementary filter* ini yaitu dengan memanfaatkan *Low Pass Filter* yang disaring oleh *accelerometer* dan memanfaatkan sudut yang terintegrasi *gyroscope* dalam waktu yang singkat. Diagram blok dari complementary filter dapat ditunjukkan pada Gambar 4.15.



Gambar 4. 15 Diagram Blok Complementary Filter

Berdasarkan Gambar 4.15 maka didapatkan persamaan untuk complementary filter yaitu sebagai berikut :

$$Angle = Kf \times (angle + gyro\_rate \times dt) + (1-Kf) \times (accelero\_angle) \quad (4\text{-}9)$$

Keterangan :

Kf = Koefisien filter

dt = Waktu sampling

Gyro_rate = Kecepatan sudut *gyroscope*

Accelero_angle = Sudut *accelerometer*

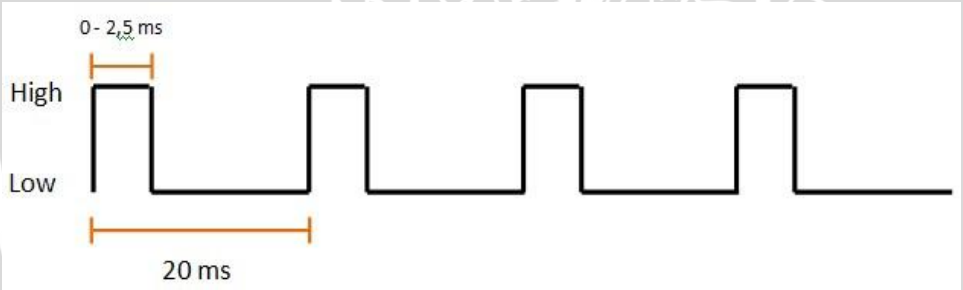
Dalam kode program dapat dituliskan seperti berikut :

```
Complementary_X_Angle = ((ax * (Complementary_X_Angle + (giro_rate_x*dt)))
                        + (acc_x_angle*(1-ax)));

Complementary_Y_Angle = ((ax * (Complementary_Y_Angle + (giro_rate_y*dt)))
                        + (acc_y_angle*(1-ax)));
```

Untuk merubah sudut dari sensor IMU menjadi sudut servo maka digunakan Persamaan (4-4), diasusmsikan bahwa perubahan sudut yang terjadi linier.

4.1.3.6. Perancangan Pengontrolan Motor DC Servo

Motor DC servo bekerja pada periode 20 ms dan diberikan secara periodik. Pembangkitan periode 20 ms menggunakan timer PWM (*Pulse Width Modulation*) yang telah tersedia pada mikrokontroller, karena motor DC servo hanya dapat membaca dalam bentuk pulsa maka periode timer PWM yang diberikan kepada motor DC servo akan diterjemahkan menjadi pulsa oleh motor DC servo. Agar motor DC servo dapat mempertahankan sudut putarannya maka periode diberikan secara terus menerus sebesar 20 ms. Jika pemberian pulsa berhenti maka akan menghentikan sudut putaran dari motor DC servo. Jika periode yang diberikan berubah-ubah maka posisi sudut dari motor DC servo juga akan berubah-ubah. Sehingga pemberian periode harus diberikan secara terus-menerus dan tidak berubah-ubah sehingga pulsa yang diberikan kepada motor DC servo tidak berubah-ubah juga. Pulsa periodik dapat ditunjukkan pada Gambar 4.16.



Gambar 4. 16 Pulsa Periodik Kontrol Motor DC Servo.

Untuk mengontrol motor DC servo membutuhkan periode 20 ms, sehingga jika dirubah dalam bentuk frekuensi yaitu 50 Hz. Untuk mendapatkan frekuensi sebesar 50 Hz maka dapat digunakan rumus seperti berikut :

$$Update\_event = TIM\_CLK / ((Psc + 1) \times (ARR + 1) \times (RCR + 1)) \quad (4\text{-}10)$$

Keterangan :

TIM_CLK = *Timer Clock Input*.
Psc = 16 bit *Prescaler Register*
ARR = 16/32 bit *Autoreload Register*.
RCR = 16 bit *Repetition Counter*.

Agar mendapatkan *update event* 50 Hz maka dapat ditentukan *prescaler* dan *autoreload* terlebih dahulu. Diketahui data *prescaler* dan *autoreload* yang digunakan adalah sebagai berikut :

TIM_CLK = *System Core Clock* / 2 = 164 MHz / 2 = 84 MHz
Prescaler = 21
Autoreload = 76362
Perhitungan frekuensi menggunakan persamaan (4-10).
Update_event = 84000000 / (21+1) × (76362 + 1) × (1)
= 84000000 / 1679986
= 50 Hz.

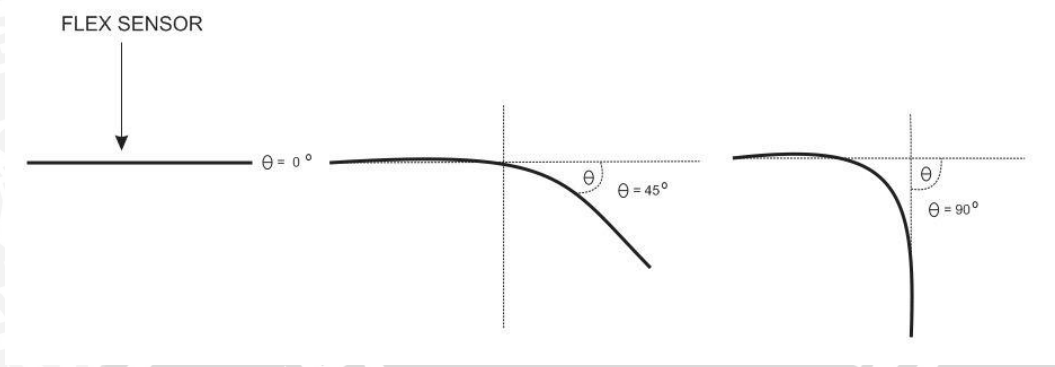
4.2. Pengujian Sistem

4.2.1. Pengujian *Flex Sensor*

Pengujian *flex sensor* bertujuan untuk menguji tegangan keluaran dari *flex sensor* terhadap posisi sudut dari *flex sensor*. Pengujian dari *flex sensor* dilakukan dengan menggunakan bahan-bahan seperti *flex sensor*, busur 180°, penjepit besi, rangkaian pembagi tegangan, dan *voltmeter*. *Flex sensor* merupakan objek yang akan diuji. Busur 180° adalah alat ukur yang digunakan untuk mengukur sudut dari *flex sensor*. Penjepit besi digunakan untuk menjepit busur derajat dan *flex sensor* agar pada saat pengukuran sudut tidak terjadi kesalahan. Rangkaian pembagi tegangan digunakan sebagai rangkain pengondisi sinyal oleh *flex sensor*.

Pada pengujian *flex sensor*, variabel yang diamati adalah tegangan keluaran dari rangkaian *flex sensor* terhadap sudut. Tegangan keluaran diukur setiap kelipatan 10°. Pengukuran dimulai dari 0° sampai dengan sudut maksimal 180°. Keluran dari *flex sensor* berupa resistansi sehingga membutuhkan rangkaian pengondisi sinyal berupa pembagi tegangan seperti pada Gambar 4.5 yang digunakan untuk mengatur tegangan keluaran dari rangkaian *flex sensor*. Untuk menguji sudut dari *flex sensor* digunakan busur 180° derajat dengan jari-jari 5 cm. Busur derajat dijepit secara vertikal menggunakan penjepit besi agar tidak terjadi banyak gerakan ketika pengukuran sudut. *Flex sensor* juga dijepit menggunakan penjepit besi pada salah satu ujung sumbu yang berdekatan dengan penghubung *flex sensor* dan kabel. *Flex sensor* dan busur derajat sejajar pada sudut 0°

terhadap *flex sensor*. Ujung *flex sensor*akan ditarik searah jarum jam pada kelipatan sudut 10° setelah itu akan diamati tegangan keluaran dari rangkaian *flex sensor*. Menurut refrensi dari data sheet *flex sensor*, pengukuran sudut dapat dilakukan dengan cara seperti pada Gambar 4.17.



Gambar 4. 17 Pengukuran Sudut Flex Sensor

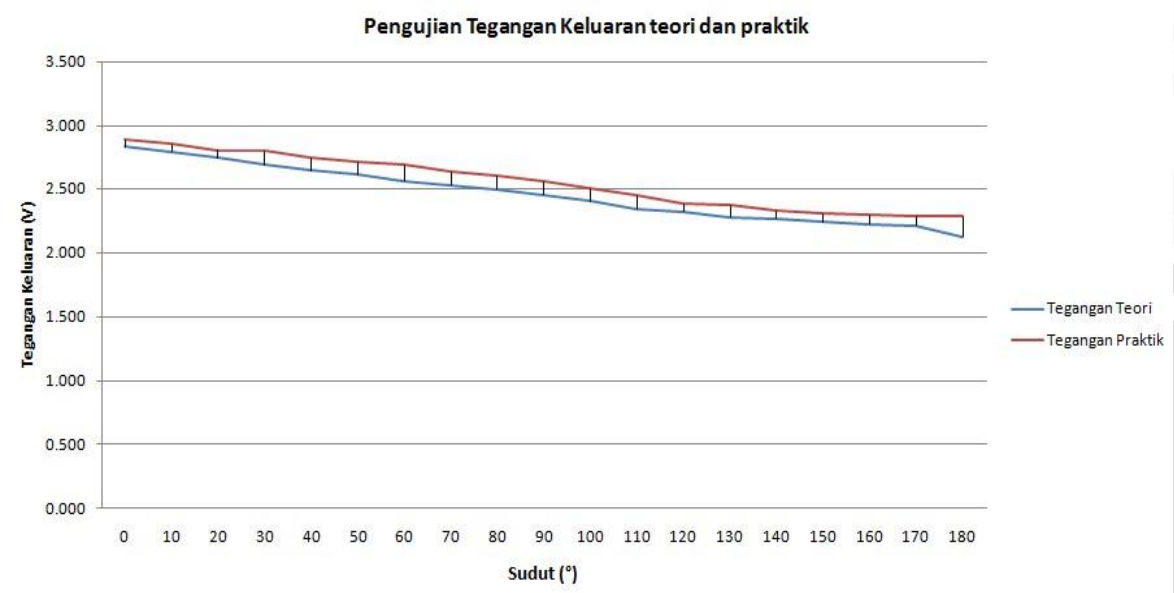
Rangkaian *flex sensor* dapat memiliki tegangan keluaran dari 2,258 volt sampai dengan 2,842 volt dengan tegangan sumber yang terukur pada voltmeter 4,70 volt. Penentuan tegangan keluaran dari rangkaian *flex sensor* menggunakan persamaan (4-1). Hasil percobaan *flex sensor* dapat ditunjukkan pada Tabel 4.4.

Tabel 4. 3 Hasil Pengujian Flex Sensor.

No	Sudut(°)	Tegangan Keluaran Flex Sensor (V)
1	0	2.886
2	10	2.856
3	20	2.804
4	30	2.795
5	40	2.748
6	50	2.716
7	60	2.686
8	70	2.633
9	80	2.601
10	90	2.557
11	100	2.504
12	110	2.446
13	120	2.381
14	130	2.372
15	140	2.328
16	150	2.308
17	160	2.296
18	170	2.287

19	180	2.287

Berdasarkan Tabel 4.3 sudut efektif dari *flex sensor* yaitu 170° dengan sudut batas bawah yaitu 0° dan sudut batas atas yaitu 170°. Kesalahan rata-rata dari pembacaan tegangan keluaran dari rangkaian flex sensor sebesar 0,087 volt. Grafik tegangan keluaran dari *flex sensor* dapat ditunjukkan pada Gambar 4.18.

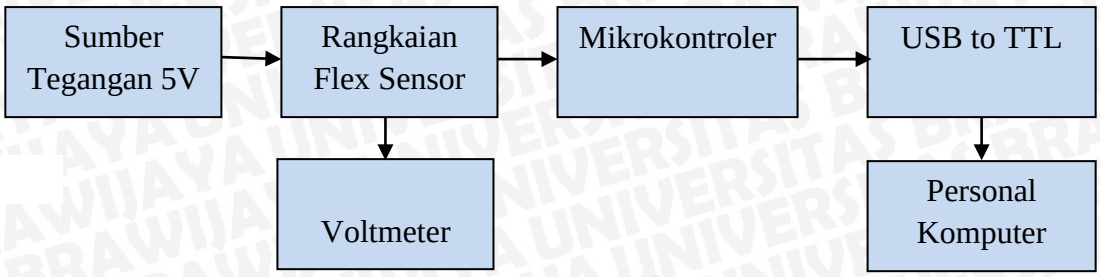


Gambar 4. 18 Grafik Tegangan Keluaran Flex Sensor.

Berdasarkan hasil pengujian pada Tabel 4.3, dapat disimpulkan bahwa tegangan keluaran dari rangkaian *flex sensor* yaitu linier. Semakin besar sudut dari *flex sensor* maka tegangan keluaran yang dihasilkan oleh *flex sensor* semakin kecil. Memiliki sudut efektif 170° dan memiliki error yang kecil yaitu sebesar 0,087 volt.

4.2.2. Pengujian ADC Mikrokontroler.

Rangkaian *flex sensor* memiliki keluran berupa tegangan analog, sehingga perlu dirubah menjadi tegangan digital dengan menggunakan *peripheral* ADC (*Analog to Digital Converter*). Pada pengujian ADC ini, variabel yang diamati yaitu perubahan nilai ADC terhadap masukan tegangan analog dari output rangkaian *flex sensor*. Bahan-bahan yang digunakan dalam pengujian ini yaitu mikrokontroller, *flex sensor*, rangkaian pembagi tegangan, multimeter SANWA CD800a, USB to TTL CP2102, dan *personal computer* (PC). Diagram blok pengujian ADC mikrokontroler dapat dilihat pada Gambar 4.19.



Gambar 4. 19 Diagram Blok Pengujian ADC Mikrokontroler.

Sumber tegangan 5 volt dihubungkan ke rangkaian *flex sensor* dan pembagi tegangan. Rangkaian *flex sensor* memiliki tegangan keluaran analog yang berubah-ubah sesuia dengan sudut dari *flex sensor* seperti pada Gambar 4.2. Keluaran dari rangkaian *flex sensor* dihubungkan dengan mikrokotroler untuk merubah tegangan analog menjadi sinyal digital dan rangkaian *flex sensor* juga dihubungkan dengan voltmeter untuk mengamati tegangan keluaran rangkaian flex sensor yang dijadikan tegangan input (Vin) oleh mikrokontroler. Untuk menampilkan nilai ADC digunakan personal komputer dengan prantara antarmuka USB to TTL. Pengiriman data dilakuakn secara serial dari mikrokontroler ke personal komputer dengan metode UART (*Universal Asynchronous Receiver/Transmitter*) yang terdapat pada *peripheral* mikrokontroler.

Pada mikrokontroler dibuat sebuah program yang dapat menterjemahkan sinyal analog menjadi sinyal digital kemudian data konversi dikirim ke personal komputer. Ada dua fungsi program yang dibuat dalam pengujian ADC mikrokontroler ini, yaitu fungsi untuk menerjemahkan sinyal analog menjadi sinyal digital (ADC) dan fungsi yang digunakan untuk mengirimkan data ADC ke personal komputer. ADC yang digunakan pada pengujian ini yaitu ADC 10 bit sehingga dapat dihitung resolusi sampling ADC terhadap tegangan input dapat dihitung menggunakan persamaan berikut.

$$Resolusi = \frac{tegangan\ skala\ penuh}{2^n - 1} \quad (4\text{-}11)$$

Keterangan :

Tegangan skala penuh = tegangan maksimal yang dapat dikonversi mikrokontroler

n = jumlah bit yang digunakan.

Pada mikrokontroler tegangan skala penuh yang dapat dikonversi menjadi sinyal digital yaitu 3 volt. Sehingga resolusi dari sampling ADC dapat dihitung menggunakan Persaman (4-1) sebagai berikut :

$$Resolusi = \frac{tegangan\ skala\ penuh}{2^n - 1}$$

$$Resolusi = \frac{3}{2^{10} - 1}$$

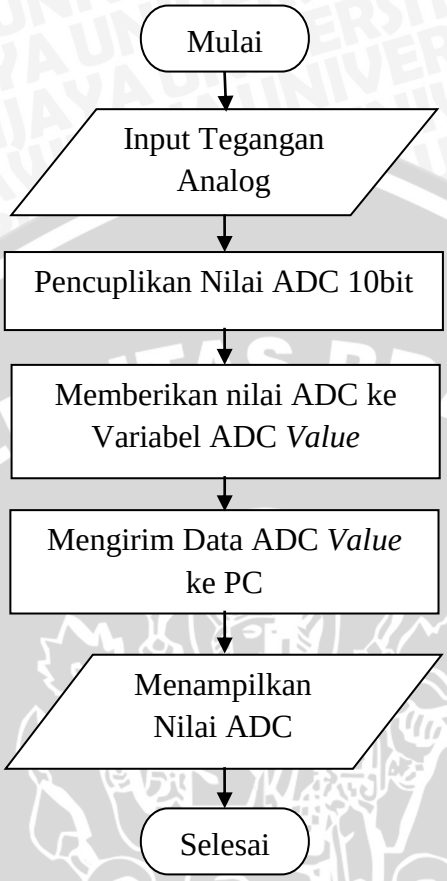
$$Resolusi = \frac{3}{1023}$$

$$Resolusi = 2{,}933\ mV$$

Pada hasil perhitungan didapatkan resolusi yang sangat kecil yaitu sebesar 2,933mV. Semakin kecil resolusi maka semakin kecil juga *error* yang didapatkan ketika proses sampling terjadi. Semakin kecil resolusi maka pencacahan sinyal analog yang dapat dikonversi menjadi nilai ADC semakin banyak sehingga sinyal digital yang dihasilkan mendekati sinyal analog yang disampling dengan *noise* yang relatif kecil.

Selain fungsi ADC, dibuat juga program dengan fungsi UART. UART yang digunakan pada mikrokontroler yaitu UART3 dengan *baudrate* 9600. *Baud rate* yang digunakan adalah baudrate standar komunikasi. Baudrate dapat diperbesar sesuai dengan yang diinginkan.

Pada Gambar 4.19, input tegangan analog dari rangkaian *flex sensor* di cuplik untuk mendapatkan nilai ADC pada mikrokontroler. Nilai ADC yang telah didapatkan diberikan kepada variabel ADC *value*. Kemudian nilai ADC *value* dikirim ke PC menggunakan perantara USB *to* TTL dengan *interface* menggunakan UART. Dan terakhir data di tampilkan ke personal komputer.Diagram alir pengujian ADC mikrokontroler dapat ditunjukan pada Gambar 4.20.



Gambar 4. 20 Diagram Alir Program Pengujian ADC Mikrokontroler.

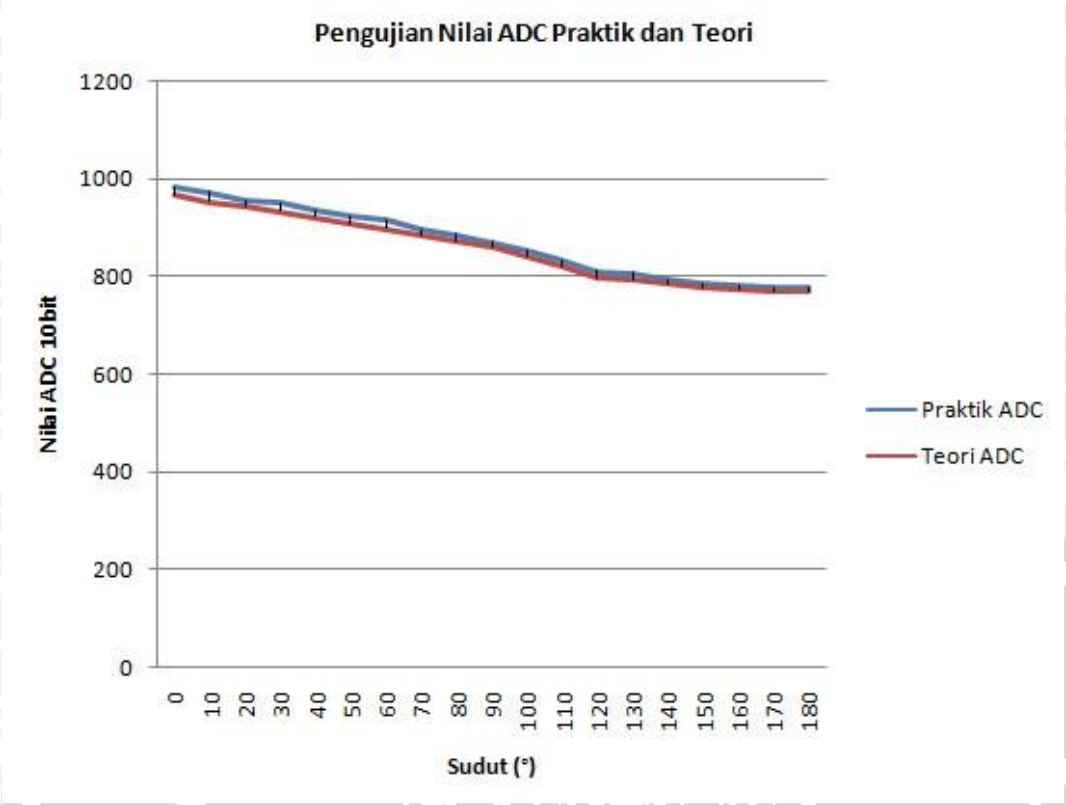
Pada pengujian ADC mikrokontroler, ada dua variabel yang akan diamati yaitu nilai ADC praktik yang merupakan nilai ADC yang didapatkan dari pengamatan secara langsung pada data yang ditampilkan di personal komputer, variabel yang kedua yaitu nilai ADC teori yang merupakan nilai ADC yang didapatkan dari perhitungan menggunakan Persamaan (3-2). Data hasil pengujian ADC praktik dan ADC teori dapat ditunjukkan pada Tabel 4.4.

Tabel 4. 4 Pengujian ADC mikrokontroler.

Sudut(°)	Vref(V)	Vin(V)	Praktik ADC
0	3	2.842	984
10	3	2.801	974
20	3	2.769	956
30	3	2.741	953

40	3	2.705	937
50	3	2.668	926
60	3	2.637	916
70	3	2.592	898
80	3	2.565	887
90	3	2.524	872
100	3	2.468	854
110	3	2.411	834
120	3	2.342	812
130	3	2.325	809
140	3	2.301	794
150	3	2.286	787
160	3	2.273	783
170	3	2.260	780
180	3	2.258	780

Nilai ADC yang diamati yaitu berdasarkan sudut dari *flex sensor*. Sudut dari *flex sensor* dimulai dari sudut 0° kemudian selajutnya naik dengan kelipatan 10° sampai sudut maksimal 180°. Kesalahan rata-rata dalam pengujian yaitu sebesar 1,271%. Kesalahan paling besar yaitu 1,843% dan kesalahan terkecil yaitu 0,731%. Nilai ADC sudah mulai stabil pada sudut 170° dengan nilai ADC praktik 780 sehingga sudut efektif berdasarkan nilai ADC yaitu pada sudut 0° sampai sudut 170°. Grafik nilai ADC pada pengujian nilai ADC mikrokontroler dapat ditunjukkan pada Gambar 4.21.



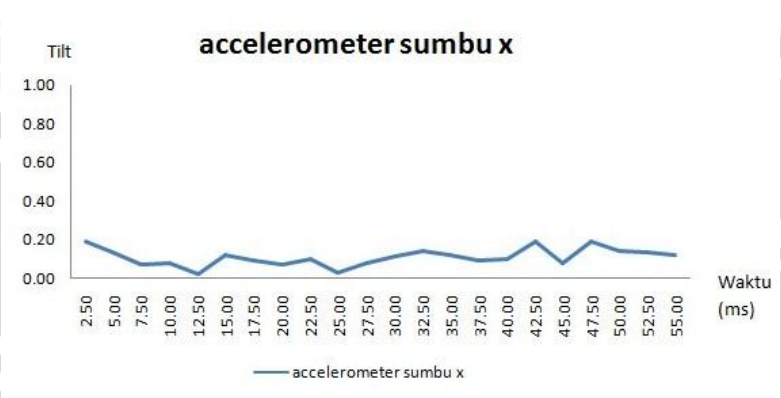
Gambar 4. 21 Grafik Nilai ADC Mikrokontroler.

Berdasarkan Gambar 4.21 dapat disimpulkan bahwa semakin besar sudut flex sensor maka nilai ADC yang dihasilkan semakin kecil. Nilai keluaran ADC yaitu linier terhadap sudut *flex sensor*. Nilai ADC sudah mendekati nilai sebenarnya berdasarkan *error* rata-rata yang relatif kecil. Nilai maksimal ADC yang dapat dikonversi yaitu 1024 karena menggunakan 10 bit, nilai maksimal ADC masih dapat ditambah karena pada mikrokontroler ADC dapat melakukan sampling samapai 16 bit.

4.2.3. Pengujian *Inertial Measurement Unit* (IMU)

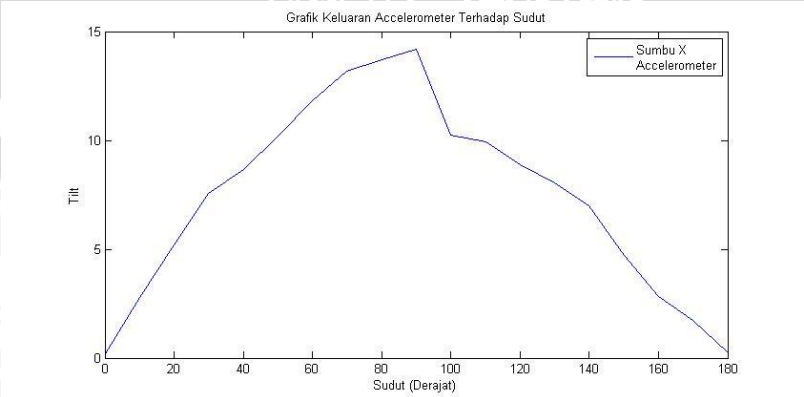
Pengujian IMU bertujuan untuk mengetahui perubahan nilai koefisien filter dan waktu sampling dari *complementary filter* terhadap penggabungan sensor *accelerometer* dan *gyroscope*. Penggabungan sensor *accelerometer* dan *gyroscope* dilakukan karena masing-masing sensor memiliki kelebihan dan kekurangan. Sensor *accelerometer* memiliki respon terhadap perubahan sudut yang sangat akurat jika sensor tersebut berada pada keadaan yang statis (diam), akan tetatapi apabila sensor dalam keadaan dinamis (bergerak) sensor tersebut tidak dapat memberikan sudut yang akurat karena memiliki respon yang lambat. Sensor *gyroscope* memiliki kelebihan dalam membaca kecepatan sudut (*angular*

rate) dalam keadaan dinamis. Penentuan sudut dari sensor tersebut dilakukan dengan metode integrasi terhadap sinyal keluaran dari sensor. Metode integrasi yang digunakan oleh sensor *gyroscope* menyebabkan *noise* pada *gyroscope* juga akan di integrasi sehingga jika dalam waktu yang lama maka *noise* akan semakin besar. Metode integrasi yang digunakan yaitu metode integrasi *forward-Euler* seperti ditunjukkan pada persamaan (2-28). Untuk keluaran *accelerometer* dapat ditunjukkan pada Gambar 4.22, dan Gambar 4.23.



Gambar 4. 22 Keluaran Accelerometer Pada Saat Diam

Pada Gambar 4.22 keluaran dari *accelerometer* pada saat sudut kemiringan 0° tidak tepat menunjukan sudut 0° karena keluaran pada *accelerometer* masih memiliki *noise* sehingga keluaran sensor pada sudut kemiringan 0° memiliki nilai fluktuatif pada rentang 0,00 sampai 0,20. Keluaran sensor *accelerometer* terhadap sudut dapat ditunjukan pada Gambar 4.23.

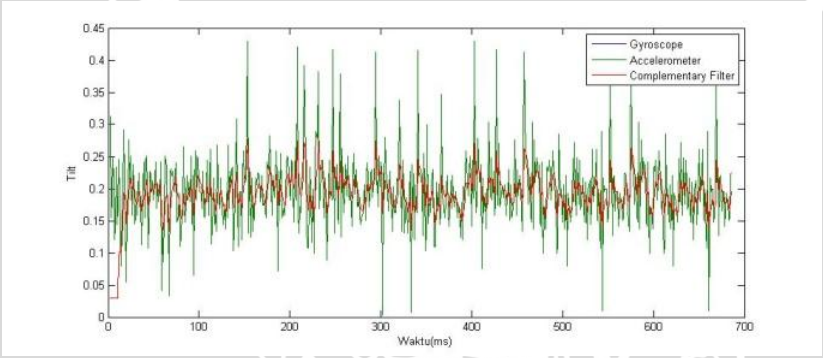


Gambar 4. 23 Keluaran Accelerometer Terhadap Sudut.

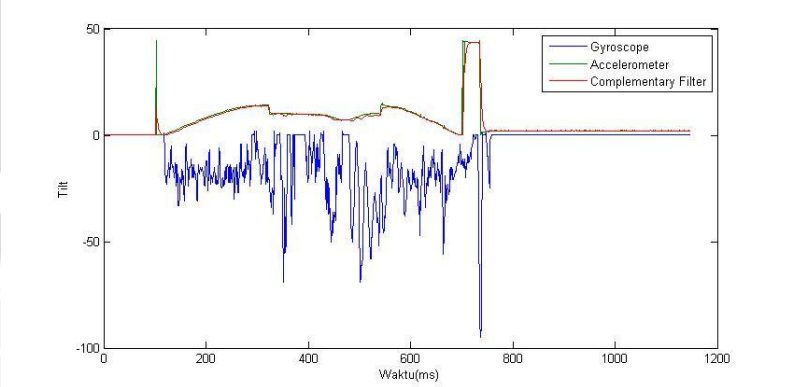
Pada Gambar 4.23 keluaran sensor accelerometer diukur pada sudut 0° sampai dengan sudut 180°. Pengukuran dilakukan dengan memutar sensor IMU berlawanan arah

jarum jam dengan sudut kelipatan 10°. Keluaran dari sensor terus naik ketika sudut 0° sampai dengan sudut 90° dengan nilai tertinggi 14 kemudian setelah melewati sudut 90° keluaran sensor terus turun secara linier hingga mendekati 0 pada saat sudut 180°. Hasil keluaran dari sensor tersebut menunjukan nilai 0. Hal tersebut terjadi karena nilai keluaran dari *gyroscope*sama dengan nilai *offset*-nya sehingga menghasilkan sudut kemiringan 0°. Nilai *offset* didapatkan dari proses kalibrasi.

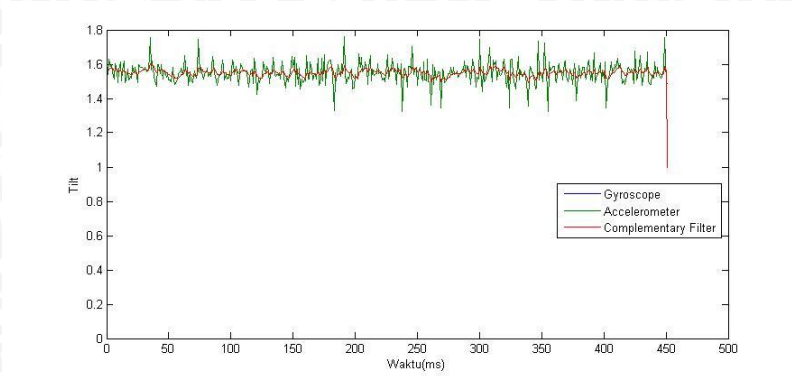
Pengujian selanjutnya pada sensor IMU yaitu mengamati *time constant* pada *complementary filter*. Pengujian dilakukan dengan melihat respon ketika koefisien filter dan waktu sampling dirubah-ubah pada persamaan (4-9). Percobaan pertama yaitu dengan merubah nilai koefisien filter sebesar 0,7, 0,8, dan 0,995 dengan waktu sampling tetap yaitu 10 ms.



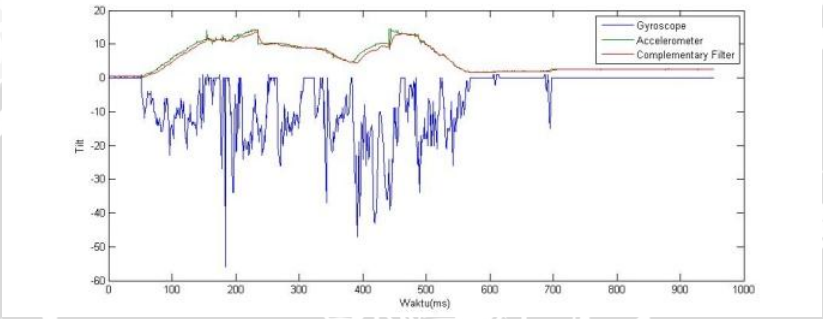
Gambar 4. 24 Koefisien Filter 0,7 dan Waktu Sampling 10ms Pada Saat Diam.



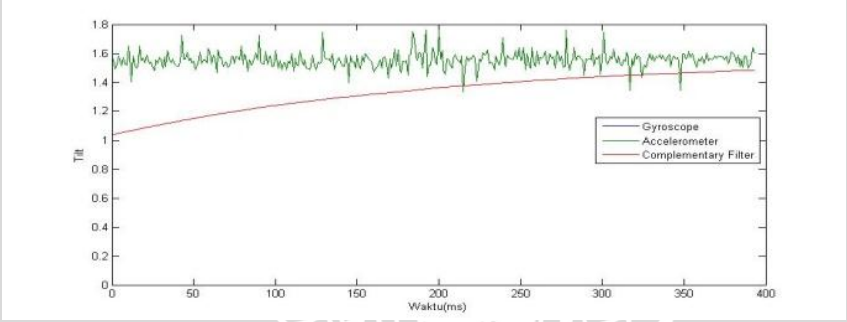
Gambar 4. 25 Koefisien Filter 0,7 dan Waktu Sampling 10ms Pada Saat Rotasi.



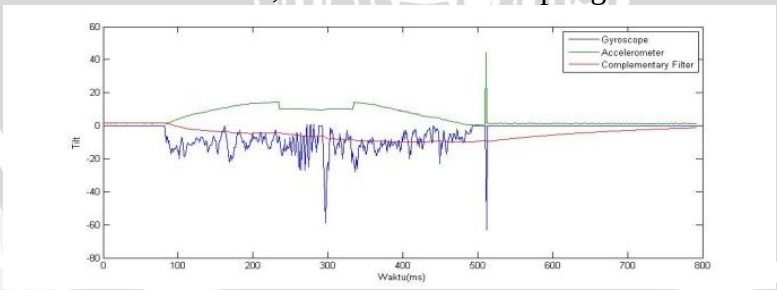
Gambar 4. 26 Koefisien Filter 0,8 dan Waktu Sampling 10ms Pada Saat Diam.



Gambar 4. 27 Koefisien Filter 0,8 dan Waktu Sampling 10ms Pada Saat Berotasi.



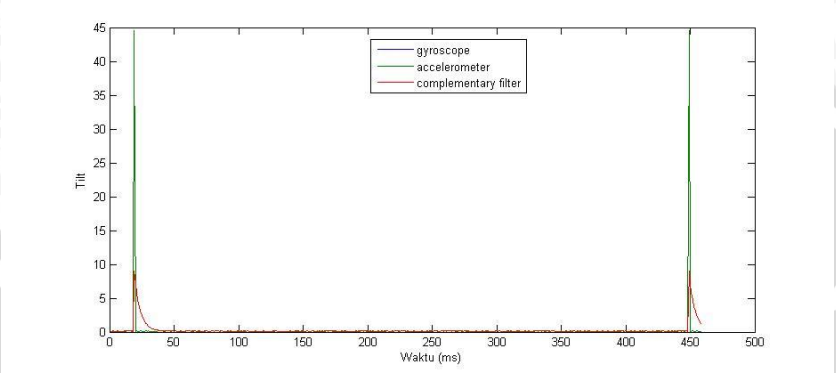
Gambar 4. 28 Koefisien Filter 0,995 dan Waktu Sampling 10ms Pada Saat Diam.



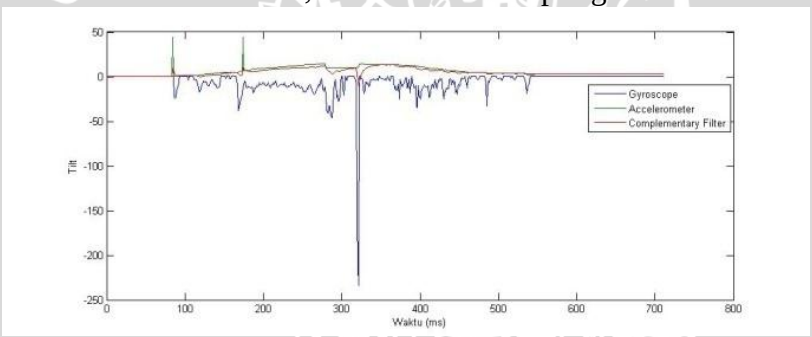
Gambar 4. 29 Koefisien Filter 0,995 dan Waktu Sampling 10ms Pada Saat Berotasi.

Pada pengujian dengan nilai koefisien filter yang berubah-ubah tersebut dapat ditarik kesimpulan bahwa semakin besar nilai koefisien filter maka sampling sinyal semakin akurat akan tetapi membutuhkan waktu yang lebih lama. Seperti ditunjukan pada Gambar 4.24 dan Gambar 4.25 yang memiliki respon yang lebih cepat dibandingkan

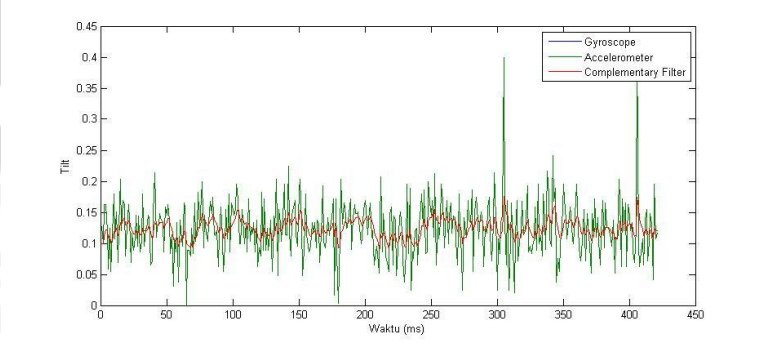
dengan Gambar 4.28 dan Gambar 4.29. Oleh karena itu, semakin besar nilai koefisien filter maka time constant akan semakin lambat. Sehingga untuk mendapatkan nilai sudut kemiringan yang akurat dengan waktu yang relatif tidak lama maka digunakan koefisien filter sebesar 0,8. Selanjutnya akan diuji nilai koefisen filter tetap sebesar 0,8 kemudian waktu samplingnya berubah-ubah sebesar 10Hz,50Hz, dan100Hz. Hal tersebut bertujuan untuk melihat respon filter terhadap perubahan dari waktu sampling (dt).



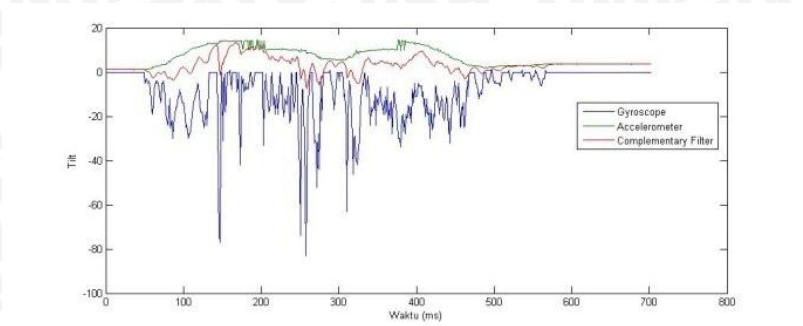
Gambar 4. 30 Koefisien Filter 0,8 dan Waktu Sampling 20ms Pada Saat Diam.



Gambar 4. 31 Koefisien Filter 0,8 dan Waktu Sampling 20ms Pada Saat Berotasi.



Gambar 4. 32 Koefisien Filter 0,8 dan Waktu Sampling 100ms Pada Saat Diam.



Gambar 4. 33 Koefisien Filter 0,8 dan Waktu Sampling 50ms Pada Saat Berotasi.

Untuk percobaan nilai dt = 0,01s sudah dilakukan pada Gambar 4.26 dan Gambar 4.27. Berdasarkan waktu sampling yang berubah-ubah maka dapat disimpulkan bahwa semakin cepat waktu sampling maka waktu yang dibutuhkan untuk mendapatkan sinyal yang lebih akurat semakin sedikit.

4.2.4. Pengujian PWM Mikrokontroler

Pengujian PWM mikrokontroler bertujuan untuk mengetahui nilai *update event* (waktu *high* sinyal) untuk mengontrol motor DC servo. Pengujian dilakukan pada periode 20ms dan menggunakan Velleman PCLAB 2000SE yang memiliki fitur untuk menampilkan *duty cycle*, positif *width*, periode, dan frekuensi. Sinyal keluaran dapat ditunjukkan pada Gambar 4.34.

Gambar 4. 34 Sinyal Keluaran Ketika Autoreload 1500us.

untuk membuktikan keakuratan sinyal maka terdapat perhitungan secara teori dengan menghitung *update event*. PWM yang digunakan yaitu 16MHz dengan *prescaler* 21 dan *autoreload* yang ditentukan. Hasil pengujian didapatkan seperti pada Tabel 4.5.

Tabel 4. 5 Pengujian PWM mikrokontroler

Autoreload (µs)	*Update Event* (s)		Kesalahan(%)
	Teori	Praktik	
500	0.000689	0.00068	0.044
600	0.000826	0.00082	0.057
700	0.000964	0.00097	0.006
800	0.001101	0.00110	0.007
900	0.001239	0.00124	0.006
1000	0.001376	0.00137	0.032
1100	0.001514	0.00150	0.069
1200	0.001651	0.00165	0.007
1300	0.001789	0.00179	0.006
1400	0.001926	0.00192	0.032
1500	0.002064	0.00206	0.019
1600	0.002201	0.00219	0.057
1700	0.002339	0.00233	0.044
1800	0.002476	0.00244	0.182
1900	0.002614	0.00260	0.069
2000	0.002751	0.00274	0.057
2100	0.002889	0.00289	0.006
2200	0.003026	0.00304	0.068
2300	0.003164	0.00316	0.019
2400	0.003301	0.00329	0.057
2500	0.003439	0.00344	0.006
Rerata			0.040

Berdasarkan hasil pengujian tersebut dapat disimpulkan bahwa PWM yang dihasilkan oleh mikrokontroler yang digunakan memiliki keakuratan yang tinggi dengan terbukti memiliki *error* yang sangat kecil yaitu sebesar 0,04%. *Error* tertinggi yaitu sebesar 0.182% dan *error* terkecil yaitu 0,006%.

4.2.5. Pengujian Motor DC Servo

Pengujian motor DC servo bertujuan untuk menganalisis perubahan sudut motor servo terhadap nilai PWM yang diberikan kepada motor DC servo. Ditampilkan juga nila *duty cycle* praktik dan teori. Perhitungan *duty cycle* teori yaitu dengan membagi nilai *autoreload* (*pulse*) terhadap *pulse* maksimal PWM yaitu 20ms kemudian dikalikan dengan 100. Pengalian 100 dilakukan karena satuan dari *duty cycle* yaitu dalam bentuk persen. Hasil pengujian dari motor DC servo dapat ditunjukkan pada Tabel 4.6.

Tabel 4. 6 Hasil Pengujian Motor Servo**.**

Autoreload (pulse)	Sudut Motor Servo (°)		Duty Cycle(%)	
	MG995	ES08MD	Teori	Praktik
400	0	0	2.00	3.41
500	0	12	2.50	4.09
600	10	24	3.00	4.84
700	20	36	3.50	5.51
800	30	48	4.00	6.21
900	40	60	4.50	6.87
1000	50	72	5.00	7.52
1100	60	84	5.50	8.25
1200	70	96	6.00	8.97
1300	80	108	6.50	9.62
1400	90	120	7.00	10.30
1500	100	132	7.50	11.00
1600	110	144	8.00	11.70
1700	120	156	8.50	12.20
1800	120	168	9.00	13.00

4.2.6. Pengujian Bluetooth

Pengujian ini bertujuan untuk mengetahui jarak maksimal *pairing* untuk pengiriman data. Selain itu pengujian juga dilakukan terhadap posisi dan *delay* dari *bluetooth*. Pengujian dilakukan dengan mengirimkan data sebanyak 50 data kemudian setiap 10 data di *delay* selama 1 detik. Pengujian juga dilakukan dalam dua posisi *bluetooth* yaitu horizontal dan vertikal. Baudrate yang digunakan yaitu 9600 bps. Hasil pengujian dapat ditunjukkan pada Tabel 4.7 dan Tabel 4.8.

Tabel 4. 7 Pengujian *Bluetooth* dengan Posisi Horizontal.

Jarak (m)	Pairing (Bluetooth Horizontal)	
	Delay per 10 Data (s)	Data Terima (Data)
1	1	1-50
2	1	1-50
3	1	1-50
4	1	1-50
5	1	1-50
6	1	1-50
7	1	1-50
8	1	1-50
9	-	-
10	-	-

Tabel 4. 8 Pengujian *Bluetooth* dengan Posisi Vertikal.

Jarak (m)	Pairing (Bluetooth Vertikal)	
	Delay per 10 Data (s)	Data Terima
5	1	1-50
10	1	1-50
15	1	1-50

20	1	1-50
25	1	1-50
30	1	1-50
35	1	1-50
40	-	-

Dari hasil pengujian dapat disimpulkan bahwa posisi dari *bluetooth* menentukan jarak *pairing bluetooth* tersebut. Ketika posisi horizontal jarak maksimal hanya 8 meter dan ketika posisi *bluetooth* vertikal maka jarak maksimal sampai 35 meter. Pada alat yang dirancang, akan digunakan pairing *bluetooth* secara vertikal karena jarak pairing *bluetooth* tersebut cukup jauh.

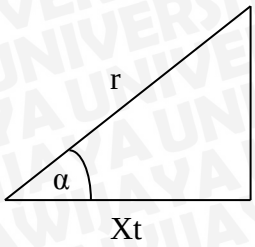
4.2.7. Pengujian Mekanik Jari-Jari Robot.

Pengujian tersebut bertujuan untuk mengetahui sudut masing masing dari lengan penyusun jari-jari robot. Lengan penyusun jari-jari robot dapat ditunjukan pada Gambar 3.3. selain itu, pengujian juga dilakukan untuk motor DC servo yang terhubung dengan lengan 1 pada jari-jari robot. Pengujian motor DC tersebut dilakukan untuk mengetahui perbandingan sudut antara motor DC servo dengan lengan 1 pada jari-jari robot. Perbandigan tersebut digunakan untuk menentukan sudut dari keseluruhan mekanik jari-jari robot. Pengujian dilakukan dengan menggunakan busur derajat 180°. Busur tersebut diletakan sejajar pada sumbu putar dari masing-masing lengan penyusun jari-jari robot. Kemudian motor DC servo yang terhubung dengan lengan 1 dari jari-jari robot diberikan sudut dengan kelipatan 1.7° untuk menggerakan 1° dari lengan 1. Kelipatan 1.7° didapatkan dari hasil pengujian sudut motor DC servo ES08MD. Sudut yang dihasilkan oleh lengan 1 pada jari-jari robot mengakibatkan lengan 2 dan lengan 3 jari-jari robot membentuk sudut tertentu. Pengujian mekanik dari jari-jari robot dapat ditunjukan pada tabel 4.9.

Tabel 4. 9 Pengujian Mekanik Jari-jari Robot.

Pulsa awal(0°)	Pulsa (PWM)	Sudut Servo(°)	Sendi Jari-Jari Robot			Kordinat		Sudut Keseluruhan
			Lengan1 (θ1)(°)	Lengan2 (θ2)(°)	Lengan3 (θ3)(°)	Xt	Yt	
	10	1.7	1	5.0	3.0	10.1014	0.8857	5.0108
	20	3.4	2	10.0	6.0	9.9327	1.7555	10.0230
	30	5.1	3	14.0	7.0	9.7435	2.3923	13.7945
	40	6.8	4	18.0	8.0	9.4991	3.0079	17.5706
	50	8.5	5	20.0	10.0	9.2804	3.4572	20.4317
	60	10.2	6	22.0	13.5	8.9782	3.9525	23.7605
	70	11.9	7	22.0	15.5	8.8308	4.1898	25.3821
	80	13.6	8	23.0	17.5	8.6067	4.5031	27.6190
	90	15.3	9	23.5	18.0	8.4699	4.7109	29.0825
	100	17.0	10	24.0	18.0	8.3498	4.8972	30.3919
	110	18.7	11	25.0	18.0	8.1875	5.1183	32.0111
	120	20.4	12	26.0	21.5	7.8530	5.4385	34.7037
1060	130	22.1	13	29.0	21.5	7.5061	5.7736	37.5667
	140	23.8	14	35.0	20.5	6.9196	6.2323	42.0083
	150	25.5	15	39.0	20.5	6.4304	6.5490	45.5234
	160	27.2	16	41.0	21.0	6.0916	6.7511	47.9396
	170	28.9	17	45.0	21.0	5.5695	6.9967	51.4795
	180	30.6	18	46.0	21.0	5.3437	7.1216	53.1174
	190	32.3	19	51.0	21.0	4.6958	7.3218	57.3260
	200	34.0	20	53.0	21.0	4.3554	7.4293	59.6191
	210	35.7	21	56.0	21.0	3.9054	7.5247	62.5702
	220	37.4	22	58.0	21.0	3.5599	7.5923	64.8790
	230	39.1	23	60.0	21.0	3.2135	7.6428	67.1948
	240	40.8	24	61.0	21.0	2.9733	7.6878	68.8554
	250	42.5	25	62.0	21.0	2.7328	7.7249	70.5180

Dari hasil perhitungan, pulsa yang harus diberikan untuk menggerakan 1° lengan 1 yaitu membutuhkan nilai pulsa 10. Pulsa yang diberikan pada saat sudut 0° pada jari-jari robot yaitu 1060. Sehingga pulsa yang sebenarnya yang diberikan yaitu 1060+10=1070 untuk menggerakan satu dejarajat dari lengan 1. Untuk menghitung sudut dari keseluruhan lengan jari-jari robot dapat digunakan persamaan trigonometri dasar pada persamaan (2-5) dan persamaan (2-6). Dari persamaan tersbut dihasilkan kordinat (x,y) sehingga perlu dilakukan konversi ke dalam bentuk sudut. Penentukan sudut dapat dilakukan dengan cara sebagai berikut :



$$r^2 = xt^2 + yt^2 \quad (4\text{-}12)$$

$$\tan \alpha = \frac{Yt}{Xt} \quad (4\text{-}13)$$

Perhitungan kordinat (Xt,Yt) dapat dilakukan sebaga berikut :

Xt $= l_1 cos\ \theta_1 + l_2 cos\ (\theta_1 + \theta_2) + l_3 cos\ (\theta_1 + \theta_2 + \theta_3)$

= 3.935 x cos(1) + 3.015 x cos(1+5) + 3.208 x cos(1+5+3)

= 10.1014

Yt $= l_1 sin\ \theta_1 + l_2 sin\ (\theta_1 + \theta_2) + l_3 sin\ (\theta_1 + \theta_2 + \theta_3)$

= 3.935 x sin(1) + 3.015 x sin(1+5) + 3.208 x sin(1+5+3)

= 0.8857

Kordinat (Xt,Yt) dapat dirubah menjadi sudut dengan persamaan trigonometri sederhana yaitu:

$$\propto = \arctan \frac{Yt}{Xt}$$

$$\propto = \arctan \frac{0.8857}{10.1014}$$

$$\propto = 5.0108°$$

Sudut keseluruhan mekanik jari-jari robot ketika lengan 1 diberikan sudut 1° yaitu 5.0108°. sudut keseluruhan mekanik ketika diberikan kelipatan sudut sebesar 1° dapat ditunjukan pada tabel 4.9.

4.2.8. Pengujian Keseluruhan Sistem Pergelangan Tangan

Pengujian tersebut bertujuan untuk mengetahui presentase kesalahan dari pergelangan tangan robot ketika sensor IMU (*Inertial Measurement Unit*) diberikan sudut tertentu. Pengujian dilakukan dengan menggunakan dua busur 180°. Masing-masing busur diletakan pada motor DC servo dan sensor IMU untuk mengukur sudut rotasi dari masing-masing objek tersebut. Pengujian dilakukan dengan memutar sensor IMU ke arah kanan dan ke arah kiri dengan sudut tertentu, setelah itu dilihat respon sudut dari pergelangan tangan. Karena sudut maksimal dari motor DC servo yang digunakan terbatas hanya

sampai 120°. Hasil pengujian keseluruhan sistem pergelangan tangan dapat ditunjukan pada tabel 4.10 dan tabel 4.11.

Tabel 4. 10 Pengujian Sudut Sensor IMU Rotasi ke Arah Kanan

Pengujian Sudut Rotasi ke Arah Kanan		
Sensor IMU (°)	Pergelangan Tangan (°)	Kesalahan (%)
10	7	5.0000
20	21	1.6667
30	27	5.0000
40	46	10.0000
50	55	8.3333
60	66	10.0000
Rerata		6.6667

Tabel 4. 11 Pengujian Sudut Rotasi Sensor IMU ke Arah Kiri

Pengujian Sudut Rotasi ke Arah Kiri		
Sensor IMU (°)	Pergelangan Tangan (°)	Kesalahan (%)
10	6	6.6667
20	18	3.3333
30	30	0.0000
40	47	11.6667
50	60	16.6667
60	68	13.3333
Rerata		8.6111

Berdasarkan hasil pengujian keseluruhan sistem pergelangan tangan dengan sudut rotasi sensor imu ke arah kanan menghasilkan kesalahan sebesar 6,6667% dengan kesalahan tertinggi 10% dan kesalahan terendah 0%. pengujian keseluruhan sistem pergelangan tangan dengan sudut rotasi sensor imu kea rah kiri menghasilkan kesalahan sebesar 8,6111% dengan kesalahan tertinggi 16,6667% dan kesalahan terendah sebesar 3,3333%. dari hasil pengujian tersebut, keseluruhan sistem pergelangan tangan sudah cukup baik dengan terbukti memiliki kesalahan yang sangat kecil yaitu di bawah 10%.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

1. Alat *motion capture system* dibuat menggunakan sensor *inertial measurement unit* (IMU) dan *flex sensor*. Sensor IMU digunakan untuk merekam sudut kemiringan dari pergelangan tangan pengguna. Sudut kemiringan pada sensor IMU ditentukan dengan persamaan *complementary filter*. *Flex sensor* digunakan untuk merekam pergerakan dari jari-jari tangan pengguna dengan mengkonversi tegangan keluaran dari rangkaian *flex sensor* menjadi nilai ADC. Nilai ADC tersebut digunakan untuk menentukan sudut jari-jari tangan pengguna. Berdasarkan hasil pengujian *flex sensor*, tegangan keluaran *flex sensor* yaitu linier. Semakin besar sudut tekuk *flex sensor* maka tegangan keluaran akan semakin kecil.
2. Pergelangan tangan dan jari-jari robot menggunakan *actuator* motor DC servo dengan bahan dasar *acrylic* dan alumunium. Pada pergelangan tangan robot, dirancang dengan sudut rotasi 60° ke kiri dan 60° ke kanan sedangkan jari-jari robot memiliki sudut maksimal 70.5180° dengan θ_1 (lengan 1) 25°, θ_2 (lengan 2) 62°, dan θ_3 (lengan 3) 21,5°.
3. Data sudut dari alat *motion capture system* diterima oleh robot kemudian dikonversi menjadi sinyal PWM (*Pulse Wide Modulation*) untuk menggerakan motor DC servo yang terhubung ke mekanik pergelangan tangan dan jari-jari robot. PWM yang digunakan sangat akurat karena memiliki kesalahan sebesar 0,04%.
4. Metode pengiriman data sudut dari alat *motion capture system* ke robot yaitu serial dengan baudrate 9600 bps. Pengiriman data sudut menggunakan *bluetooth*. posisi dari *bluetooth* menentukan jarak *pairing bluetooth* tersebut. Ketika posisi horizontal jarak maksimal hanya 8 meter dan ketika posisi *bluetooth* vertikal maka jarak maksimal sampai 35 meter.

5.2. Saran

1. Pengembangan selanjutnya dapat menggunakan *actuator* pada setiap lengan dari jari-jari robot sehingga sudut yang dihasilkan oleh lengan jari-jari robot lebih akurat dan konstruksinya lebih kuat.

2. Sensor yang digunakan untuk merekam pergerakan jari-jari pengguna dapat diletakan pada setiap ruas dari jari-jari tangan pengguna sehingga pergerakan setiap ruas dari tangan pengguna dapat direkam.

3. Untuk pengembangan *motion capture system* selanjutnya dapat digunakan *kinect sensor*.

DAFTAR PUSTAKA

Anonim.Bluetooth Modules Wireless UART Cable Repelacement Hardware Manual & AT-Commands Refrence Manual Rev 1r0. Philippines : e-Gizmo Mechatronix Central

Field, Matthew dkk.2009.*Motion Capture in Robotics Review*. Australia : University of Wollongong.

Kolozs, James.1998.*Design of An Exoskeletal Human Motion Capture SystemSensuit*™.Utah:The University of Utah.

Nalwan, Andi.2012.Teknik Rancang Bangun Robot.Yogyakarta :Penerbit ANDI.

Nugraha, D. Wiria.2011.Pengendalian Robot yang Memiliki Lima Derajat Kebebasan. Palu : UNTAD.

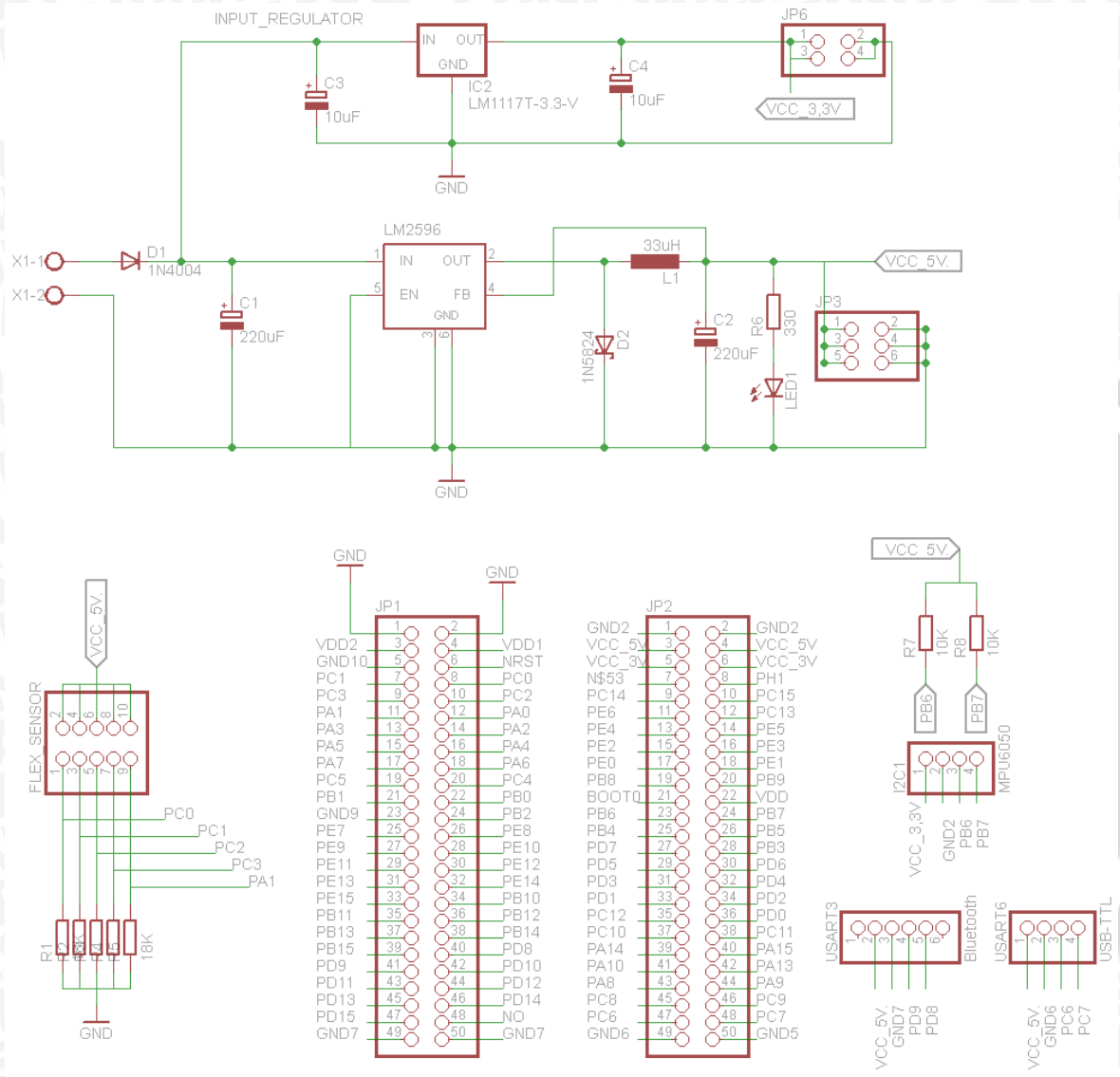
Park, Sung-Woo.2012.*Development of an Anthropomorphic Robot Hand Aimed at Practical Use for Wide Service Robot Application*. Seoul :Korea Institute of Industrial Technology,Ansan

Pitowarno, Endra. 2006. *Robotika: Desain, Kontrol dan Kecerdasan Buatan*. Yogyakarta: Penerbit ANDI.

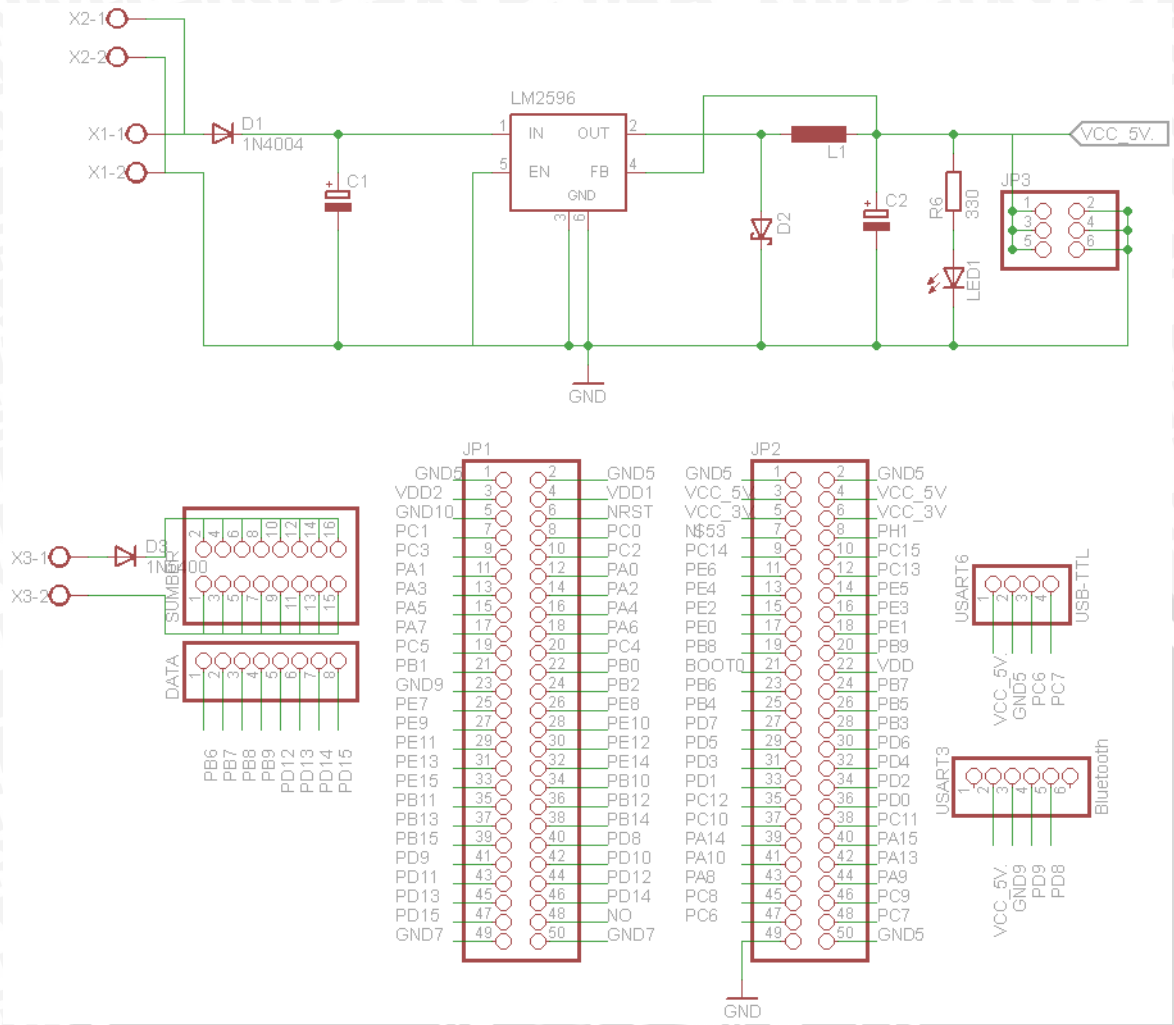
Pollard, Nancy S dkk.1999.*Adapting Human Motion for the Control of a Humanoid Robot*.Computer Science Department, Brown University.

LAMPIRAN

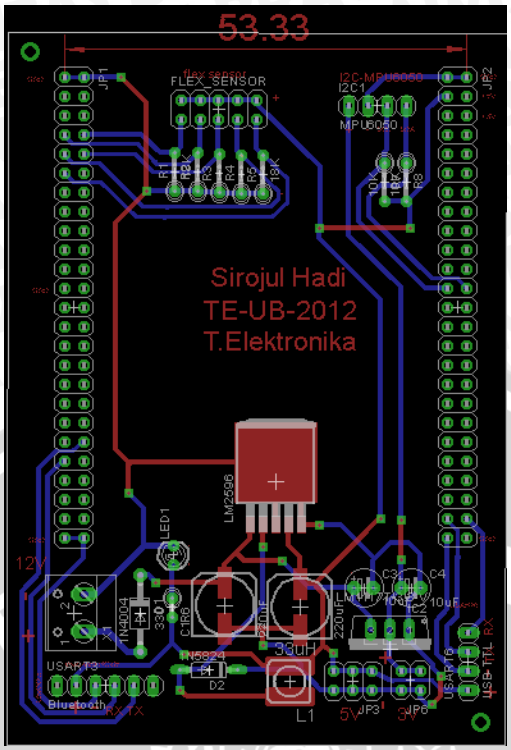
Lampiran 1. Skematik Rangkaian Motion Capture System dan Skematik Rangkaian Pergelangan Tangan Robot.



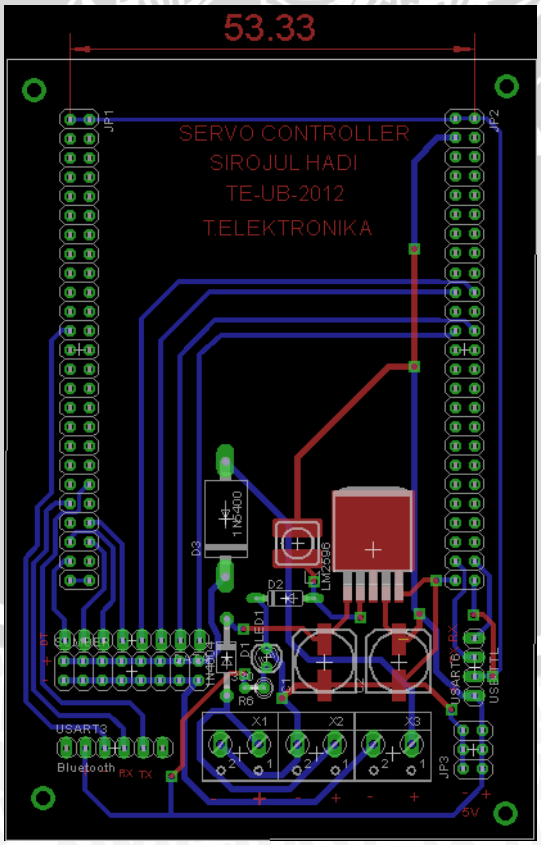
Gambar 1 Sekmatik Rangkaian Shield Board Alat Motion Capture System



Gambar 2 Skematik Rangkaian Shield Board Alat Pergelangan Tangan Robot

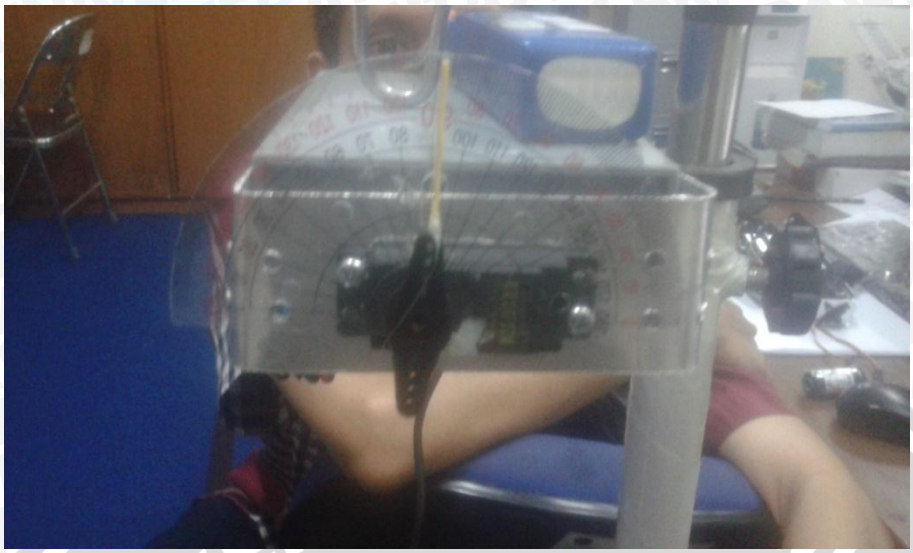


Gambar 3 PCB Shield Board Alat Motion Capture System



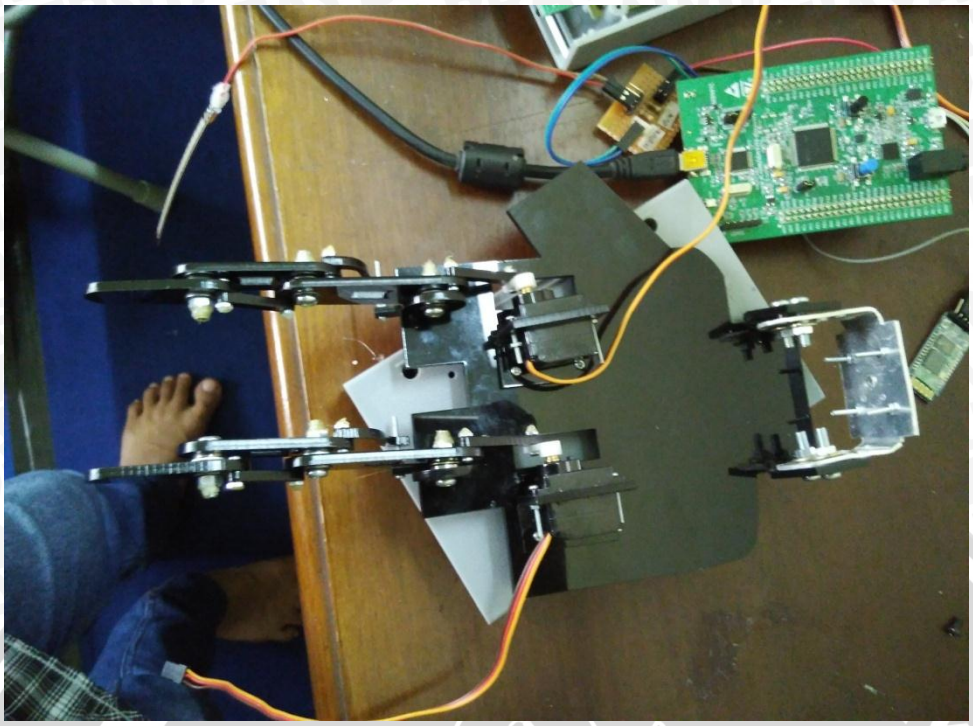
Gambar 4 PCB Shield Board Alat Pergelangan Tangan Robot

Lampiran 2. Dokumentasi Alat

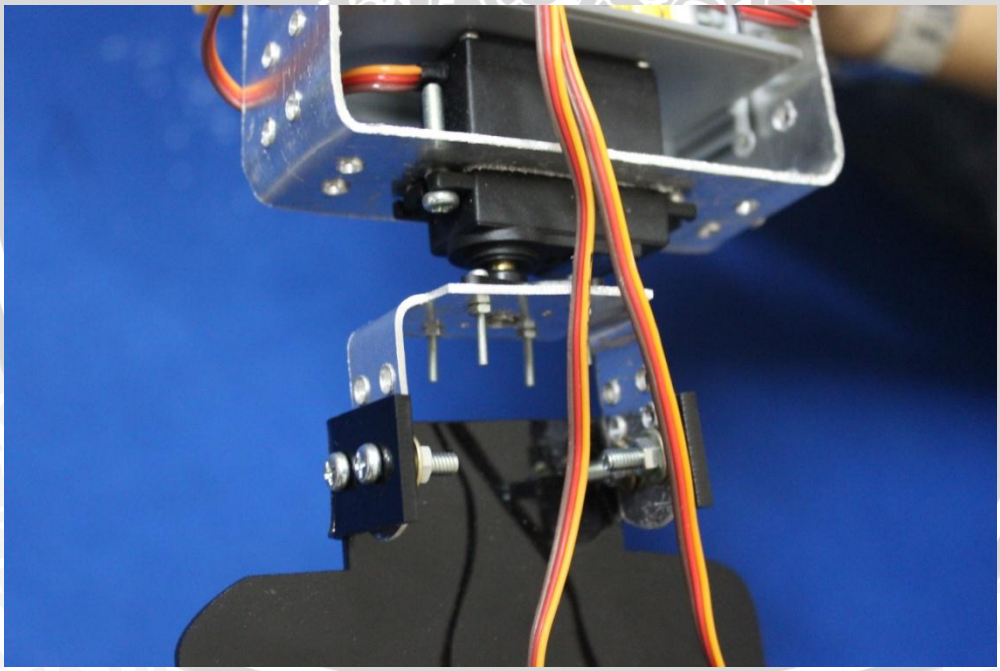


Gambar 5 Pengujian Motor Servo

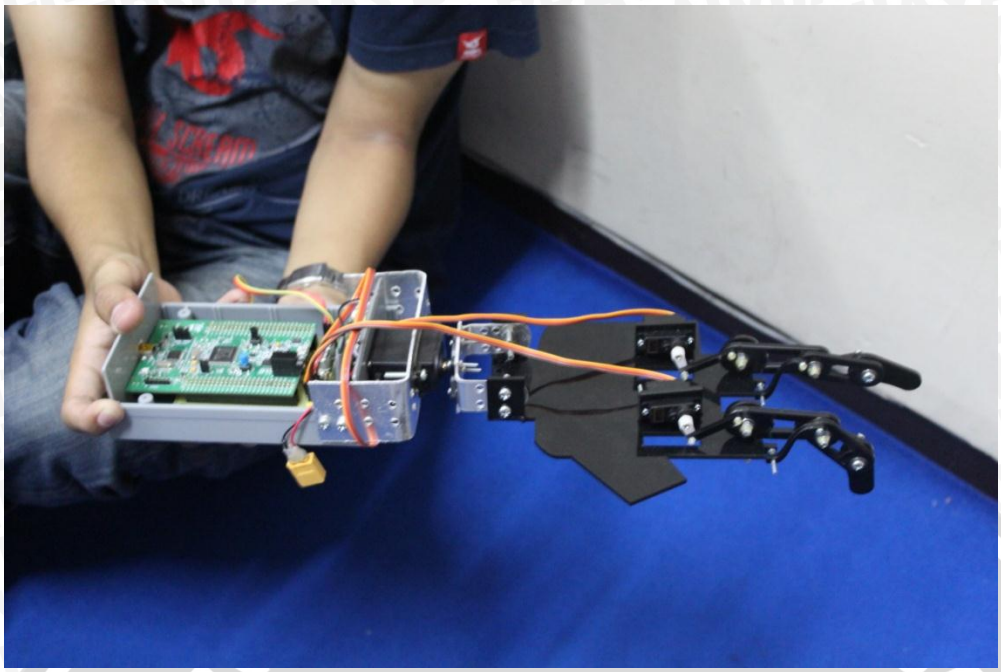
Gambar 6 Pengujian Flex Sensor



Gambar 7 Pengujian Mekanik Jari-Jari Robot



Gambar 8 Pergelangan Tangan Robot



Gambar 9 Alat Pergelangan Tangan Robot

Gambar 10 Alat Motion Capture System

Lampiran 3. Listing Program

A. Listing Program Alat Motion Capture System

```
#include <stm32f4xx.h>
#include <stm32f4xx_gpio.h>
#include <stm32f4xx_rcc.h>
#include <stm32f4xx_usart.h>
#include <stm32f4xx_i2c.h>
#include <stm32f4xx_tim.h>
#include <stm32f4xx_adc.h>
#include <stm32f4xx_dma.h>
#include <misc.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
GPIO_InitTypeDef     GPIO_Struct;
USART_InitTypeDef USART_Struct;
NVIC_InitTypeDef                NVIC_Struct;
TIM_OCInitTypeDef TIM_OCStruct;
TIM_TimeBaseInitTypeDef       TIM_TimeBaseStruct;
ADC_InitTypeDef                 ADC_Struct;
ADC_CommonInitTypeDef       ADC_CommonStruct;
DMA_InitTypeDef                 DMA_Struct;
I2C_InitTypeDef                 I2C_Struct;
//sudut
float sudut_srv=0;
//Variabel MPU6050
        float    dt=0.001,kf=0.8; //10ms
        int nilai_data,x,y,z;
        unsigned char giro_xh=0x00,giro_xl=0x00;
        unsigned char giro_yh=0x00,giro_yl=0x00;
        unsigned char giro_zh=0x00,giro_zl=0x00;
        float gyro_x_angle;
        float gyro_y_angle;
        float gyro_z_angle;
        signed long giro_z1000=0x00;
        signed long giro_x1000=0x00;
        signed long giro_y1000=0x00;
        signed int giro_x_ofset;
        signed int giro_y_ofset;
        signed int giro_z_ofset;
        float giro_rate_x;
        float giro_rate_y;
        float giro_rate_z;
        float sudut_giro_x;
        float sudut_giro_z;
        float sudut_giro_y;
        signed int giro_x;
        signed int giro_y;
        signed int giro_z;
        unsigned char acc_xh,acc_xl;
        unsigned char acc_yh,acc_yl;
        unsigned char acc_zh,acc_zl;
        signed int acc_x;
        signed int acc_y;
        signed int acc_z;
        float acc_x_angle,acc_y_angle,acc_z_angle;
        unsigned char temp_h,temp_l;
        signed int temp;
        float temperature;
        float ax=0.8;
        float filter_angle_x[3] = {0,0,0};
        float filter_angle_y[3] = {0,0,0};
        float filter_angle_z[3] = {0,0,0};
        float Complementary_X_Angle = 0x00;
        float Complementary_Y_Angle = 0x00;
        #define Time_Const 0.1
        #define MPU6050_ADDRESS 0x68 //AD0=0GND
        #define giro_x_sens 131
        #define giro_y_sens 131
        #define giro_z_sens 131
```

```
//Register MPU6050
#define XG_OFFS_TC                  0x00
#define YG_OFFS_TC                  0x01
#define ZG_OFFS_TC                  0x02
#define X_FINE_GAIN                 0x03
#define Y_FINE_GAIN                 0x04
#define Z_FINE_GAIN                 0x05
#define XA_OFFS_H                   0x06
#define XA_OFFS_L_TC                0x07
#define YA_OFFS_H                   0x08
#define YA_OFFS_L_TC                0x09
#define ZA_OFFS_H                   0x0A
#define ZA_OFFS_L_TC                0x0B
#define XG_OFFS_USRH                0x13
#define XG_OFFS_USRL                0x14
#define YG_OFFS_USRH                0x15
#define YG_OFFS_USRL                0x16
#define ZG_OFFS_USRH                0x17
#define ZG_OFFS_USRL                0x18
#define SMPLRT_DIV                  0x19
#define CONFIG                      0x1A
#define GYRO_CONFIG                 0x1B
#define ACCEL_CONFIG                0x1C
#define FF_THR                      0x1D
#define FF_DUR                      0x1E
#define MOT_THR                     0x1F
#define MOT_DUR                     0x20
#define ZRMOT_THR                   0x21
#define ZRMOT_DUR                   0x22
#define FIFO_EN                     0x23
#define I2C_MST_CTRL                0x24
#define I2C_SLV0_ADDR               0x25
#define I2C_SLV0_REG                0x26
#define I2C_SLV0_CTRL               0x27
#define I2C_SLV1_ADDR               0x28
#define I2C_SLV1_REG                0x29
#define I2C_SLV1_CTRL               0x2A
#define I2C_SLV2_ADDR               0x2B
#define I2C_SLV2_REG                0x2C
#define I2C_SLV2_CTRL               0x2D
#define I2C_SLV3_ADDR               0x2E
#define I2C_SLV3_REG                0x2F
#define I2C_SLV3_CTRL               0x30
#define I2C_SLV4_ADDR               0x31
#define I2C_SLV4_REG                0x32
#define I2C_SLV4_DO                 0x33
#define I2C_SLV4_CTRL               0x34
#define I2C_SLV4_DI                 0x35
#define I2C_MST_STATUS              0x36
#define INT_PIN_CFG                 0x37
#define INT_ENABLE                  0x38
#define DMP_INT_STATUS              0x39
#define INT_STATUS                  0x3A
#define ACCEL_XOUT_H                0x3B
#define ACCEL_XOUT_L                0x3C
#define ACCEL_YOUT_H                0x3D
#define ACCEL_YOUT_L                0x3E
#define ACCEL_ZOUT_H                0x3F
#define ACCEL_ZOUT_L                0x40
#define TEMP_OUT_H                  0x41
#define TEMP_OUT_L                  0x42
#define GYRO_XOUT_H                 0x43
#define GYRO_XOUT_L                 0x44
#define GYRO_YOUT_H                 0x45
#define GYRO_YOUT_L                 0x46
#define GYRO_ZOUT_H                 0x47
#define GYRO_ZOUT_L                 0x48
#define EXT_SENS_DATA_00  0x49
#define EXT_SENS_DATA_01  0x4A
#define EXT_SENS_DATA_02  0x4B
#define EXT_SENS_DATA_03  0x4C
#define EXT_SENS_DATA_04  0x4D
#define EXT_SENS_DATA_05  0x4E
#define EXT_SENS_DATA_06  0x4F
#define EXT_SENS_DATA_07  0x50
```

```
#define SENS_DATA_08                0x51
#define ENS_DATA_09                 0x52
#define EXT_SENS_DATA_10  0x53
#define EXT_SENS_DATA_11  0x54
#define EXT_SENS_DATA_12  0x55
#define EXT_SENS_DATA_13  0x56
#define EXT_SENS_DATA_14  0x57
#define EXT_SENS_DATA_15  0x58
#define EXT_SENS_DATA_16  0x59
#define EXT_SENS_DATA_17  0x5A
#define EXT_SENS_DATA_18  0x5B
#define EXT_SENS_DATA_19  0x5C
#define EXT_SENS_DATA_20  0x5D
#define EXT_SENS_DATA_21  0x5E
#define EXT_SENS_DATA_22  0x5F
#define EXT_SENS_DATA_23  0x60
#define MOT_DETECT_STATUS           0x61
#define I2C_SLV0_DO                 0x63
#define I2C_SLV1_DO                 0x64
#define I2C_SLV2_DO                 0x65
#define I2C_SLV3_DO                 0x66
#define I2C_MST_DELAY_CTRL          0x67
#define SIGNAL_PATH_RESET           0x68
#define MOT_DETECT_CTRL   0x69
#define USER_CTRL                       0x6A
#define PWR_MGMT_1                  0x6B
#define PWR_MGMT_2                  0x6C
#define BANK_SEL                    0x6D
#define MEM_START_ADDR              0x6E
#define MEM_R_W                     0x6F
#define DMP_CFG_1                       0x70
#define DMP_CFG_2                       0x71
#define FIFO_COUNTH                 0x72
#define FIFO_COUNTL                 0x73
#define FIFO_R_W                    0x74
#define WHO_AM_I                    0x75

//Variabel ADC
#define Max_Strlen 10000
char terima_string[Max_Strlen],buf[16];
uint16_t SdtFlex1=0,SdtFlex2=0,SdtFlex3=0,SdtFlex4=0,SdtFlex5=0;
int adc_1=0,adc_2=0,adc_3=0,adc_4=0,adc_5=0;
uint16_t ADC3ConvertedValue[5]={0,0,0,0,0};
int data,a=75,b=50;
uint16_t srv;
uint32_t multiplier;

//Makro Variabel
//ADC
void ADC_Channel(void);
void Timer_ADC(void);
//Sensor IMU-->MPU6050
void setup_I2C(void);
void Write_I2C(int Address, int Register, int Data);
void Read_I2C(int Address, int Register, unsigned char *Data);
void tes_MPU6050(void);
void MPU6050_Setting(void);
int Cek_Registers_MPU6050(void);
void Kalibrasi_Gyro(void);
void Get_Gyro_Rate(void);
void Show_Gyro_Rate(void);
void Get_Accel(void);
void Get_Accel_Angle(void);
void Show_Accel_Angle(void);
void get_temp(void);
void Complementary_Filter(void);
void show_all(void);
void konversi(void);
//Bluetooth-->USART3
void Bluetooth_Init(uint32_t baudrate);
void Kirim_DataBluetooth(USART_TypeDef* USARTx, const char *pFormat, ... );
//Kirim Data ke PC --> USART6
void USART6_Init(uint32_t baudrate);
void cek_data(USART_TypeDef* USARTx, const char *pFormat, ... );
void USART_puts(USART_TypeDef* USARTx, volatile char *s);
```

```
int main(void)
{
	SystemInit();
	Bluetooth_Init(9600);
	USART6_Init(9600);
	ADC_Channel();
	Timer_ADC();
	//KOSONGKAN REGISTER I2C
	setup_I2C();
	I2C1->DR = 0;
//	tes_MPU6050();
	MPU6050_Setting();
	Kalibrasi_Gyro();

   while(1)
    {
//	Show_Gyro_Rate();
//	show_all();
//	Show_Accel_Angle();
	konversi();
	//srv=(int)sudut_srv;
//	Kirim_DataBluetooth(USART3,"%ds%ds%ds",(int)sudut_srv,adc_1);
	Kirim_DataBluetooth(USART3,"%ds%ds",(int)sudut_srv,adc_1);
//	Kirim_DataBluetooth(USART3,"%ds\r",adc_1);
    }
}

//PROGRAM FLEX SEBSOR --> PERIPHERAL ADC
void ADC_Channel(void)
{
	//Mengaktifkan ADC3, DMA2 dan GPIO clocks
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_DMA2 | RCC_AHB1Periph_GPIOC | RCC_AHB1Periph_GPIOA, ENABLE);
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC3, ENABLE);//Mengaktifkan ADC3

	//Konfigurasi DMA Stream0 Channel
	DMA_Struct.DMA_Channel = DMA_Channel_2;
	DMA_Struct.DMA_PeripheralBaseAddr = (uint32_t)&ADC3->DR;//data register ADC3
	DMA_Struct.DMA_Memory0BaseAddr = (uint32_t)&ADC3ConvertedValue;
	DMA_Struct.DMA_DIR = DMA_DIR_PeripheralToMemory;
	DMA_Struct.DMA_BufferSize = 5;
	DMA_Struct.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
	DMA_Struct.DMA_MemoryInc = DMA_MemoryInc_Enable;
	DMA_Struct.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;//membaca data 16bit
	DMA_Struct.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;//penyimpanan 16bit
	DMA_Struct.DMA_Mode = DMA_Mode_Circular;
	DMA_Struct.DMA_Priority = DMA_Priority_High;
	DMA_Struct.DMA_FIFOMode = DMA_FIFOMode_Disable;
	DMA_Struct.DMA_FIFOThreshold = DMA_FIFOThreshold_HalfFull;
	DMA_Struct.DMA_MemoryBurst = DMA_MemoryBurst_Single;
	DMA_Struct.DMA_PeripheralBurst = DMA_PeripheralBurst_Single;
	DMA_Init(DMA2_Stream0, &DMA_Struct);
	DMA_Cmd(DMA2_Stream0, ENABLE);

	//Konfigurasi Pin GPIO
	GPIO_Struct.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3;// PC0, PC1, PC2, PC3
	GPIO_Struct.GPIO_Mode = GPIO_Mode_AN;//Mode Analog
	GPIO_Struct.GPIO_PuPd = GPIO_PuPd_NOPULL ;//Tidak ada Pull-up/pull-down resistor
	GPIO_Init(GPIOC, &GPIO_Struct);
	GPIO_Struct.GPIO_Pin = GPIO_Pin_1;//PA1
	GPIO_Init(GPIOA, &GPIO_Struct);

	//ADC Common Init
	ADC_CommonStruct.ADC_Mode = ADC_Mode_Independent;
	ADC_CommonStruct.ADC_Prescaler = ADC_Prescaler_Div2;
	ADC_CommonStruct.ADC_DMAAccessMode = ADC_DMAAccessMode_Disabled;
	ADC_CommonStruct.ADC_TwoSamplingDelay = ADC_TwoSamplingDelay_5Cycles;
	ADC_CommonInit(&ADC_CommonStruct);

	//ADC3 Init
	ADC_DeInit();
	ADC_Struct.ADC_Resolution = ADC_Resolution_10b;//konversi 10 bit
	ADC_Struct.ADC_ScanConvMode = ENABLE;//memeriksa channel
	ADC_Struct.ADC_ContinuousConvMode = ENABLE;//Melanjutkan Konversi
	ADC_Struct.ADC_ExternalTrigConv = 0;
	ADC_Struct.ADC_ExternalTrigConvEdge = ADC_ExternalTrigConvEdge_None;
	ADC_Struct.ADC_DataAlign = ADC_DataAlign_Right;//Konversi bergeser ke kanan
```

```
	ADC_Struct.ADC_NbrOfConversion = 5;
	ADC_Init(ADC3, &ADC_Struct);

	//Memilih Channel untuk Pembacaan ADC
	ADC_RegularChannelConfig(ADC3, ADC_Channel_10, 1, ADC_SampleTime_144Cycles);//PC0
	ADC_RegularChannelConfig(ADC3, ADC_Channel_11, 2, ADC_SampleTime_144Cycles);//PC1
	ADC_RegularChannelConfig(ADC3, ADC_Channel_12, 3, ADC_SampleTime_144Cycles);//PC2
	ADC_RegularChannelConfig(ADC3, ADC_Channel_13, 4, ADC_SampleTime_144Cycles);//PC3
	ADC_RegularChannelConfig(ADC3, ADC_Channel_1,  5, ADC_SampleTime_144Cycles);//PA1

	//Mengaktifkan DMA Request setelah transfer terakhir
	ADC_DMARequestAfterLastTransferCmd(ADC3, ENABLE);
	//Mengaktifkan DMA
	ADC_DMACmd(ADC3, ENABLE);
	//Mengaktifkan ADC3
	ADC_Cmd(ADC3, ENABLE);
	//Mulai konversi
	ADC_SoftwareStartConv(ADC3);
}

void Timer_ADC(void)
{
	RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

	//pengaturan waktu
	TIM_TimeBaseStruct.TIM_ClockDivision = TIM_CKD_DIV1;
	TIM_TimeBaseStruct.TIM_CounterMode = TIM_CounterMode_Up;
	TIM_TimeBaseStruct.TIM_Prescaler =999;//39999=TCNT ==>freq=2000000/999+1=2000Hz->0,0005s=0,5ms
	TIM_TimeBaseStruct.TIM_Period =41;//Tim_tick => 84000000Hz/41+1=2000000Hz
	TIM_TimeBaseStruct.TIM_RepetitionCounter = 0;
	TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStruct);

	//Mengaktifkan Interrupt
	TIM_ITConfig(TIM2,TIM_IT_Update,ENABLE);
	TIM_Cmd(TIM2,ENABLE);

	NVIC_Struct.NVIC_IRQChannel = TIM2_IRQn;
	NVIC_Struct.NVIC_IRQChannelCmd = ENABLE;
	NVIC_Struct.NVIC_IRQChannelPreemptionPriority = 0;
	NVIC_Struct.NVIC_IRQChannelSubPriority = 0;
	NVIC_Init(&		NVIC_Struct);
	}

void TIM2_IRQHandler(void)
{
	while( (TIM2->SR & (1<<0)) > 0 ) TIM2->SR &= ~(1<<0); // clear UIF flag
	//Memberikan nilai konversi ke Variabel adc_value
	SdtFlex1 = (984-ADC3ConvertedValue[0])*0.813;
//	adc_value2 = (984-ADC3ConvertedValue[1])/0.813;
//	adc_value3 = ADC3ConvertedValue[2];
//	adc_value4 = ADC3ConvertedValue[3];
//	adc_value5 = ADC3ConvertedValue[4];

	if (SdtFlex1<=70)
	{
		adc_1=(SdtFlex1+300)*(-1);
	}
}

//PROGRAM IMU --> KOMUNIKASI I2C
void setup_I2C(void) {
	//Pengaturan Pin
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
  GPIO_Struct.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
  GPIO_Struct.GPIO_Mode = GPIO_Mode_AF;
  GPIO_Struct.GPIO_OType = GPIO_OType_OD;
  GPIO_Struct.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Struct.GPIO_PuPd = GPIO_PuPd_UP;
  GPIO_Init(GPIOB, &GPIO_Struct);

  // Penaturan I2C
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_I2C1, ENABLE);
  I2C_Struct.I2C_Mode = I2C_Mode_I2C;
  I2C_Struct.I2C_ClockSpeed = 400000;
  I2C_Struct.I2C_DutyCycle = I2C_DutyCycle_2;
  I2C_Struct.I2C_OwnAddress1 = 0x00;
```

```
  I2C_Struct.I2C_Ack = I2C_Ack_Disable;
  I2C_Struct.I2C_AcknowledgedAddress = I2C_AcknowledgedAddress_7bit;

  I2C_Cmd(I2C1, ENABLE);
  I2C_Init(I2C1, &I2C_Struct);

  // Pin SCL I2C
  GPIO_PinAFConfig(GPIOB, GPIO_PinSource6, GPIO_AF_I2C1);
  // Pin SDA I2C
  GPIO_PinAFConfig(GPIOB, GPIO_PinSource7, GPIO_AF_I2C1);
}

void Write_I2C(int Address, int Register, int Data)
{
        while (I2C_GetFlagStatus(I2C1, I2C_FLAG_BUSY)) {;}

        I2C_GenerateSTART(I2C1, ENABLE);
        while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT)) {;}

        I2C_Send7bitAddress(I2C1,Address<<1,I2C_Direction_Transmitter);
        while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED)) {;}

        I2C_SendData(I2C1,Register);
        while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED)) {;}

        I2C_SendData(I2C1,Data);
        while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED)) {;}

        I2C_GenerateSTOP(I2C1, ENABLE);
        while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED));
}

void Read_I2C(int Address, int Register, unsigned char *Data)
{
        while (I2C_GetFlagStatus(I2C1, I2C_FLAG_BUSY)) {;}

        I2C_GenerateSTART(I2C1, ENABLE);
        while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT)) {;}

        I2C_Send7bitAddress(I2C1, Address<<1,I2C_Direction_Transmitter);
        while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_TRANSMITTER_MODE_SELECTED)) {;}

        I2C_SendData(I2C1,Register);
        while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_TRANSMITTED)) {;}

        I2C_GenerateSTART(I2C1, ENABLE);
        while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT)) {;}

        I2C_Send7bitAddress(I2C1,Address<<1,I2C_Direction_Receiver);
        //I2C_SendData(I2C1,0x44);
        while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_RECEIVER_MODE_SELECTED)) {;}
        while (!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED)) {;}

        *Data=I2C_ReceiveData(I2C1);
        I2C_GenerateSTOP(I2C1, ENABLE);
}

void tes_MPU6050(void)
{
  unsigned char datax=0x00;
  Read_I2C(0x68,0x75,&datax);
  cek_data(USART3," %d ",datax);
  if(datax==0x68) {
        USART_puts(USART3," BERHASIL \r");
  }
  else{
        USART_puts(USART3," TIDAK BERHASIL \r");
  }}
void MPU6050_Setting(void)
{
        //sample rate divider
        Write_I2C(MPU6050_ADDRESS,SMPLRT_DIV,0x07);
        //configuration
        Write_I2C(MPU6050_ADDRESS,CONFIG,0x00);
        //gyroscope configuration
        Write_I2C(MPU6050_ADDRESS,GYRO_CONFIG,0x08);
```

```
	//accelerometer configuration
	Write_I2C(MPU6050_ADDRESS,ACCEL_CONFIG,0x00);
	Write_I2C(MPU6050_ADDRESS,FF_THR,0x00);
	Write_I2C(MPU6050_ADDRESS,FF_DUR,0x00);
	//Motion detection threshold
	Write_I2C(MPU6050_ADDRESS,MOT_THR,0x00);
	Write_I2C(MPU6050_ADDRESS,MOT_DUR,0x00);
	Write_I2C(MPU6050_ADDRESS,ZRMOT_THR,0x00);
	Write_I2C(MPU6050_ADDRESS,ZRMOT_DUR,0x00);
	//FIFO Enable
	Write_I2C(MPU6050_ADDRESS,FIFO_EN,0x00);
	//I2C Master Control
	Write_I2C(MPU6050_ADDRESS,I2C_MST_CTRL,0x00);

	Write_I2C(MPU6050_ADDRESS,I2C_SLV0_ADDR,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV0_REG,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV0_CTRL,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV1_ADDR,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV1_REG,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV1_CTRL,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV2_ADDR,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV2_REG,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV2_CTRL,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV3_ADDR,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV3_REG,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV3_CTRL,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV4_ADDR,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV4_REG,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV4_DO,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV4_CTRL,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV4_DI,0x00);

	//Write_I2C(MPU6050_ADDRESS,0x36,0x00);
	Write_I2C(MPU6050_ADDRESS,INT_PIN_CFG,0x00);
	Write_I2C(MPU6050_ADDRESS,INT_ENABLE,0x00);

	Write_I2C(MPU6050_ADDRESS,I2C_SLV0_DO,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV1_DO,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV2_DO,0x00);
	Write_I2C(MPU6050_ADDRESS,I2C_SLV3_DO,0x00);

	Write_I2C(MPU6050_ADDRESS,I2C_MST_DELAY_CTRL,0x00);
	Write_I2C(MPU6050_ADDRESS,SIGNAL_PATH_RESET,0x00);
	Write_I2C(MPU6050_ADDRESS,MOT_DETECT_CTRL,0x00);
	Write_I2C(MPU6050_ADDRESS,USER_CTRL,0x00);
	Write_I2C(MPU6050_ADDRESS,PWR_MGMT_1,0x02);
	Write_I2C(MPU6050_ADDRESS,PWR_MGMT_2,0x00);
	Write_I2C(MPU6050_ADDRESS,FIFO_R_W,0x00);

	USART_puts(USART6," Setting MPU6050 selesai \r");
}

int Cek_Registers_MPU6050(void)
{
unsigned char Data = 0x00;
unsigned char Failed = 0;

Read_I2C(MPU6050_ADDRESS,SMPLRT_DIV,&Data);
if(Data != 0x01) { cek_data(USART6,"\rRegister 1 salah, seharusnya 0x01, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,CONFIG, &Data);
if(Data != 0x03) { cek_data(USART6,"\rRegister 2 salah, seharusnya 0x03, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,GYRO_CONFIG, &Data);
if(Data != 0b00001000) { cek_data(USART6,"\rRegister 3 salah, seharusnya 0b00001000, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,ACCEL_CONFIG, &Data);
if(Data != 0b00001000) { cek_data(USART6,"\rRegister 4 salah, seharusnya 0b00001000, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,FF_THR, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 5 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,FF_DUR, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 6 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,MOT_THR, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 7 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,MOT_DUR, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 8 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,ZRMOT_THR, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 9 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,ZRMOT_DUR, &Data);
```

```
if(Data != 0x00) { cek_data(USART6,"\rRegister 10 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,FIFO_EN, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 11 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_MST_CTRL, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 12 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV0_ADDR, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 13 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV0_REG, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 14 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV0_CTRL, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 15 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV1_ADDR, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 16 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV1_REG, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 17 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV1_CTRL, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 18 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV2_ADDR, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 19 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV2_REG, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 20 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV2_CTRL, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 21 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV3_ADDR, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 22 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV3_REG, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 23 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV3_CTRL, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 24 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV4_ADDR, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 25 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV4_REG, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 26 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV4_DO, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 27 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV4_CTRL, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 28 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV4_DI, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 29 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,INT_PIN_CFG, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 30 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,INT_ENABLE, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 31 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV0_DO, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 32 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV1_DO, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 33 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV2_DO, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 34 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_SLV3_DO, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 35 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,I2C_MST_DELAY_CTRL, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 36 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,SIGNAL_PATH_RESET, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 37 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,MOT_DETECT_CTRL, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 38 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,USER_CTRL, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 39 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,PWR_MGMT_1, &Data);
if(Data != 0x02) { cek_data(USART6,"\rRegister 40 salah, seharusnya 0x02, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,PWR_MGMT_2, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 41 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }
Read_I2C(MPU6050_ADDRESS,FIFO_R_W, &Data);
if(Data != 0x00) { cek_data(USART6,"\rRegister 42 salah, seharusnya 0x00, Bukan 0x%x", Data); Failed = 1; }

if (Failed == 0) { cek_data(USART6,"\rRegister benar"); }
else { cek_data(USART6,"\rRegister salah");}

return(Failed);
}

void Kalibrasi_Gyro(void)
{
        int x=0;
        giro_x1000=0x00;
```

```
	giro_y1000=0x00;
	giro_z1000=0x00;
	for (x=0;x<1000;x++)
	{
		Read_I2C(MPU6050_ADDRESS,GYRO_XOUT_H,&giro_xh);
		Read_I2C(MPU6050_ADDRESS,GYRO_XOUT_L,&giro_xl);
		Read_I2C(MPU6050_ADDRESS,GYRO_YOUT_H,&giro_yh);
		Read_I2C(MPU6050_ADDRESS,GYRO_YOUT_L,&giro_yl);
		Read_I2C(MPU6050_ADDRESS,GYRO_ZOUT_H,&giro_zh);
		Read_I2C(MPU6050_ADDRESS,GYRO_ZOUT_L,&giro_zl);

		giro_x1000 += ((giro_xh<<8)|(giro_xl));
		giro_y1000 += ((giro_yh<<8)|(giro_yl));
		giro_z1000 += ((giro_zh<<8)|(giro_zl));
		//delay_ms(1);
	}
	giro_x_ofset = giro_x1000/1000;
	giro_y_ofset = giro_y1000/1000;
	giro_z_ofset = giro_z1000/1000;

	cek_data(USART6," \rGyro_X_Offset_sum=%ld --> Gyro_X_Offset=%d ",giro_x_ofset,giro_x1000);
	cek_data(USART6," Gyro_Y_Offset_sum=%ld --> Gyro_Y_Offset=%d ",giro_y_ofset,giro_y1000);
	cek_data(USART6," Gyro_Z_Offset_sum=%ld --> Gyro_Z_Offset=%d \r",giro_z_ofset,giro_z1000);
}

void Get_Gyro_Rate(void)
{
	Read_I2C(MPU6050_ADDRESS,GYRO_XOUT_H,&giro_xh);
	Read_I2C(MPU6050_ADDRESS,GYRO_XOUT_L,&giro_xl);
	Read_I2C(MPU6050_ADDRESS,GYRO_YOUT_H,&giro_yh);
	Read_I2C(MPU6050_ADDRESS,GYRO_YOUT_L,&giro_yl);
	Read_I2C(MPU6050_ADDRESS,GYRO_ZOUT_H,&giro_zh);
	Read_I2C(MPU6050_ADDRESS,GYRO_ZOUT_L,&giro_zl);

	giro_x = ((giro_xh<<8)|giro_xl)- giro_x_ofset;
	giro_y = ((giro_yh<<8)|giro_yl)- giro_y_ofset;
	giro_z = ((giro_zh<<8)|giro_zl)- giro_z_ofset;

	giro_rate_x = giro_x/giro_x_sens;
	giro_rate_y = giro_y/giro_y_sens;
	giro_rate_z = giro_z/giro_z_sens;
	if (giro_rate_x<=-1&&giro_rate_x>=-400)
	{
		giro_rate_x =-giro_rate_x;
	}
	if (giro_rate_x<=-400)
	{
		giro_rate_x =-500-giro_rate_x;

	}

	gyro_x_angle +=(float)(giro_rate_x*dt);
	gyro_y_angle +=(float)(giro_rate_y*dt);
	gyro_z_angle +=(float)(giro_rate_z*dt);

}

void Show_Gyro_Rate(void)
{
	Get_Gyro_Rate();
	cek_data(USART6,"sudut_giro_x=%f     ",giro_rate_x);
	cek_data(USART6,"sudut_giro_y=%f     ",giro_rate_y);
	cek_data(USART6,"sudut_giro_z=%f \r",giro_rate_z);
	}

void Get_Accel(void)
{
	Read_I2C(MPU6050_ADDRESS,ACCEL_XOUT_H,&acc_xh);
	Read_I2C(MPU6050_ADDRESS,ACCEL_XOUT_L,&acc_xl);
	Read_I2C(MPU6050_ADDRESS,ACCEL_YOUT_H,&acc_yh);
	Read_I2C(MPU6050_ADDRESS,ACCEL_YOUT_L,&acc_yl);
	Read_I2C(MPU6050_ADDRESS,ACCEL_ZOUT_H,&acc_zh);
	Read_I2C(MPU6050_ADDRESS,ACCEL_ZOUT_L,&acc_zl);

	acc_x = ((acc_xh<<8)|acc_xl);
	acc_y = ((acc_yh<<8)|acc_yl);
```

```
	acc_z = ((acc_zh<<8)|acc_zl);

	}

void Get_Accel_Angle(void)
{
	Get_Accel();
	acc_x_angle =(float) 57.295*(atan((float)acc_y/ sqrt(pow((float)acc_z,2)+pow((float)acc_x,2))));
	acc_y_angle =(float) 57.295*(atan((float)acc_x/ sqrt(pow((float)acc_z,2)+pow((float)acc_y,2))));
	}

void Show_Accel_Angle(void)
{
	Get_Accel_Angle();
	cek_data(USART6,"\r accel_sudut_x=%.3f     ",acc_x_angle);
	cek_data(USART6,"accel_sudut_y=%.3f     ",acc_y_angle);
	}

void get_temp(void)
{
	Read_I2C(MPU6050_ADDRESS,TEMP_OUT_H,&temp_h);
	Read_I2C(MPU6050_ADDRESS,TEMP_OUT_L,&temp_l);
	temp=((temp_h<<8)|temp_l);
	temperature=(temp+12412)/340;

	cek_data(USART3,"temperature=%.3f     \r",temperature);
	}
// Complementary Filter Angle for MPU6050
void Complementary_Filter(void)
{
	Get_Gyro_Rate();
	Get_Accel_Angle();
	Complementary_X_Angle = (kf*(Complementary_X_Angle + (giro_rate_x*dt))) + (acc_x_angle*(1-kf));
	Complementary_Y_Angle = (kf*(Complementary_X_Angle + (giro_rate_x*dt))) + (acc_x_angle*(1-kf));
	}
void show_all(void)
{
	Complementary_Filter();
	cek_data(USART3,"%.2f\t %.2f\r",acc_x_angle,acc_y_angle);
	//cek_data(USART3,"%.3f gy=%.2f
gz=%.2f\tax=%.2f\tay=%.2f\tcfx=%f\tcfy=%f\r",giro_rate_x,giro_rate_y,giro_rate_z,acc_x_angle,acc_y_angle,Complementary_X_Ang
le,Complementary_Y_Angle);
	//cek_data(USART3,"%.3f\t%.3f\t%.3f\r",giro_rate_x,acc_x_angle,Complementary_X_Angle);
}

//konversi IMU ke sudut
void konversi(void)
{
	Get_Gyro_Rate();
	Get_Accel_Angle();
	if (acc_y_angle>=76.00)// rotasi ke kanan
	{
		if(acc_x_angle>=0.00 && acc_x_angle<=2.80){
			sudut_srv= -(((2.8-(2.8-acc_x_angle))*3.7735)+100);
		}
		else if (acc_x_angle>2.80 && acc_x_angle<=5.21){
			sudut_srv= -((((2.41-(5.21-acc_x_angle))*4.1493)+10)+100);
		}
		else if (acc_x_angle>5.21 && acc_x_angle<=7.55){
			sudut_srv= -((((2.34-(7.55-acc_x_angle))*4.2735)+20)+100);
		}
		else if (acc_x_angle>7.55 && acc_x_angle<=8.68){
			sudut_srv= -((((1.13-(8.68-acc_x_angle))*8.8495)+30)+100);
		}
		else if (acc_x_angle>8.68 && acc_x_angle<=10.20){
			sudut_srv= -((((1.52-(10.20-acc_x_angle))*6.5789)+40)+100);
		}
		else if (acc_x_angle>10.20 && acc_x_angle<=11.83){
			sudut_srv= -((((1.6-(11.83-acc_x_angle))*6.1349)+50)+100);
		}
		else if (acc_x_angle>11.83 && acc_x_angle<=14.17)
		{
			sudut_srv=-160;
	}}
	else if (acc_y_angle<=51.65)//rotasi ke kiri
	{
```

```
                if(acc_x_angle>=44.40 && acc_x_angle<=45.30){
                        sudut_srv= -(((0.9-(acc_x_angle-44.40)) * (float)11.11)+200);
                }
                else if (acc_x_angle>=43.43 && acc_x_angle<44.40){
                        sudut_srv= -(((0.9-(acc_x_angle-43.43)) * (float)10.31 + 10)+200);
                }
                else if (acc_x_angle>40.65 && acc_x_angle<43.43){
                        sudut_srv= -(((0.9-(acc_x_angle-40.65)) * (float)3.60 + 20)+200);
                }
                else if (acc_x_angle>39.74 && acc_x_angle<40.65){
                        sudut_srv= -(((0.9-(acc_x_angle-39.74)) * (float)10.99 + 30)+200);
                }
                else if (acc_x_angle>38.73 && acc_x_angle<39.74){
                        sudut_srv= -(((0.9-(acc_x_angle-38.73)) * (float)9.90 + 40)+200);
                }
                else if (acc_x_angle>38.03 && acc_x_angle<=38.73){
                        sudut_srv= -(((0.9-(acc_x_angle-38.03)) * (float)14.29 + 50)+200);
                }
                else{
                        sudut_srv=-260;
                }
        }}
//BLUETOOTH-HC-05 --> Komunikasi USART3
void Bluetooth_Init(uint32_t baudrate)
{
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART3, ENABLE);
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

        GPIO_Struct.GPIO_OType = GPIO_OType_PP;
        GPIO_Struct.GPIO_PuPd = GPIO_PuPd_NOPULL;
        GPIO_Struct.GPIO_Mode = GPIO_Mode_AF;

        GPIO_Struct.GPIO_Pin = GPIO_Pin_8|GPIO_Pin_9;
        GPIO_Struct.GPIO_Speed = GPIO_Speed_100MHz;
        GPIO_Init(GPIOD, &GPIO_Struct);

        GPIO_PinAFConfig(GPIOD, GPIO_PinSource8, GPIO_AF_USART3);
        GPIO_PinAFConfig(GPIOD, GPIO_PinSource9, GPIO_AF_USART3);

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

        USART_Cmd(USART3, ENABLE);
}

void Kirim_DataBluetooth(USART_TypeDef* USARTx, const char *pFormat, ... )
{
   va_list ap;
   char pStr[100];

   va_start(ap, pFormat);
   vsprintf(pStr, pFormat, ap);
   va_end(ap);

   int i=0;
   int n = strlen(pStr);
   for(i=0;i<n;i++)
   {
      USART_SendData(USARTx, (uint8_t)pStr[i]);
      while (USART_GetFlagStatus(USARTx, USART_FLAG_TC) == RESET);
   }
}

//komunikasi dengan komputer --> Komunikasi USART6
void USART6_Init(uint32_t baudrate)
{
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART6, ENABLE);
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
  /* Configure USART Tx as alternate function  */
        GPIO_Struct.GPIO_OType = GPIO_OType_PP;
        GPIO_Struct.GPIO_PuPd = GPIO_PuPd_NOPULL;
```

```
	GPIO_Struct.GPIO_Mode = GPIO_Mode_AF;

	GPIO_Struct.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7;
	GPIO_Struct.GPIO_Speed = GPIO_Speed_100MHz;
  GPIO_Init(GPIOC, &GPIO_Struct);

	GPIO_PinAFConfig(GPIOC, GPIO_PinSource6, GPIO_AF_USART6);
	GPIO_PinAFConfig(GPIOC, GPIO_PinSource7, GPIO_AF_USART6);

	USART_Struct.USART_BaudRate = baudrate;
	USART_Struct.USART_WordLength = USART_WordLength_8b;
	USART_Struct.USART_StopBits = USART_StopBits_1;
	USART_Struct.USART_Parity = USART_Parity_No;
	USART_Struct.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
	USART_Struct.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
	USART_Init(USART6, &USART_Struct);

	USART_Cmd(USART6, ENABLE);
}

void cek_data(USART_TypeDef* USARTx, const char *pFormat, ... )
{
  va_list ap;
  char pStr[100];

  va_start(ap, pFormat);
  vsprintf(pStr, pFormat, ap);
  va_end(ap);

  int i=0;
  int n = strlen(pStr);
  for(i=0;i<n;i++)
  {
    USART_SendData(USARTx, (uint8_t)pStr[i]);
    while (USART_GetFlagStatus(USARTx, USART_FLAG_TC) == RESET);
  }
}

void USART_puts(USART_TypeDef* USARTx, volatile char *s){

	while(*s){
		// wait until data register is empty
		while( !(USARTx->SR & 0x00000040) );
		USART_SendData(USARTx, *s);
		*s++;
	}
}
```

B. Listing Program Alat Pergelangan Tangan Robot

```
#include <stm32f4xx.h>
#include <stm32f4xx_gpio.h>
#include <stm32f4xx_usart.h>
#include <stm32f4xx_rcc.h>
#include <stm32f4xx_tim.h>
#include <misc.h>

#include "math.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "stdarg.h"

	GPIO_InitTypeDef	GPIO_InitStructure;
	USART_InitTypeDef	USART_InitStructure;
	NVIC_InitTypeDef	NVIC_InitStructure;

#define MAX_STRLEN 10000
volatile float servo_angle[4] = { 0.0, 0.0, 0.0, 0.0 };
volatile uint32_t count;
char received_string[MAX_STRLEN];
```

```
int terima[5],angka=0,DATA,sudut_servo=0;
int data[3]={0,0,0};
int i;

void delay_ms(__IO uint32_t nCount);
void gpio_set(void);
void TIM4_Config(int a);
void NVIC_Configuration(void);
void USART3_Init(uint32_t baudrate);
void USART6_Init(uint32_t baudrate);
void cek_data(USART_TypeDef* USARTx, const char *pFormat, ... );
void USART_puts(USART_TypeDef* USARTx, volatile char *s);

int main(void)
{
        USART3_Init(9600);
        USART6_Init(9600);
        gpio_set();
        TIM4_Config(1);
        NVIC_Configuration();


  while(1)
  {
        //jari-jari=0-140 --> pergelangan tangan 0 - 600 dan 0 - -600
        if (data[0]<=-100 && data[0]>-200)
        {
                servo_angle[0] = (data[0]+100)*(-10);
                delay_ms(10);
        }
        else if (data[0]<=-200 && data[0]>-300)
   {
        servo_angle[0] = (data[0]+200)*10;
        delay_ms(10);
   }
        if (data[1]<=-300 && data[1]>-400)
        {
    servo_angle[1] = (data[1]+300)*(-10);
    servo_angle[2] = (data[1]+300)*(-10);
    delay_ms(10);
        }
        cek_data(USART6,"%d\t%d\r",(int)servo_angle[0],(int)servo_angle[1]);
  }

}

void delay_ms(__IO uint32_t nCount)
{
        count=nCount*15272;
        while (count--)
        {;}
        }

void gpio_set(void)
{
        GPIO_InitTypeDef                GPIO_InitStructure;
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

        GPIO_InitStructure.GPIO_Pin             = GPIO_Pin_12 | GPIO_Pin_13 | GPIO_Pin_14 | GPIO_Pin_15;
        GPIO_InitStructure.GPIO_Mode            = GPIO_Mode_AF;
        GPIO_InitStructure.GPIO_Speed           = GPIO_Speed_100MHz;
        GPIO_InitStructure.GPIO_OType           = GPIO_OType_PP;
        GPIO_InitStructure.GPIO_PuPd            = GPIO_PuPd_NOPULL;
        GPIO_Init(GPIOD, &GPIO_InitStructure);

        //Timer Pin
        GPIO_PinAFConfig(GPIOD, GPIO_PinSource12, GPIO_AF_TIM4);
```

```
	GPIO_PinAFConfig(GPIOD, GPIO_PinSource13, GPIO_AF_TIM4);
	GPIO_PinAFConfig(GPIOD, GPIO_PinSource14, GPIO_AF_TIM4);
	GPIO_PinAFConfig(GPIOD, GPIO_PinSource15, GPIO_AF_TIM4);
}

void TIM4_Config(int a)
{
	TIM_OCInitTypeDef  	TIM_OCInitStructure;
	TIM_TimeBaseInitTypeDef   TIM_TimeBaseStructure;
	RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);

	TIM_TimeBaseStructure.TIM_Prescaler 	= 21;
	TIM_TimeBaseStructure.TIM_Period 	= 80075;
	TIM_TimeBaseStructure.TIM_ClockDivision 	= 0;
	TIM_TimeBaseStructure.TIM_CounterMode 	= TIM_CounterMode_Up;
	TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);

	 TIM_ARRPreloadConfig(TIM4, ENABLE);

	 TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
	 TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
	 TIM_OCInitStructure.TIM_Pulse = 0; // Servo Top-Center
	 TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;

	 /* Output Compare PWM1 Mode configuration: Channel1 PD.12 */
	 TIM_OC1Init(TIM4, &TIM_OCInitStructure);
	 TIM_OC1PreloadConfig(TIM4, TIM_OCPreload_Enable);

	 /* Output Compare PWM1 Mode configuration: Channel2 PD.13 */
	 TIM_OC2Init(TIM4, &TIM_OCInitStructure);
	 TIM_OC2PreloadConfig(TIM4, TIM_OCPreload_Enable);

	 /* Output Compare PWM1 Mode configuration: Channel3 PD.14 */
	 TIM_OC3Init(TIM4, &TIM_OCInitStructure);
	 TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable);

	 /* Output Compare PWM1 Mode configuration: Channel4 PD.15 */
	 TIM_OC4Init(TIM4, &TIM_OCInitStructure);
	 TIM_OC4PreloadConfig(TIM4, TIM_OCPreload_Enable);

	 /* TIM Interrupts enable */
	 TIM_ITConfig(TIM4, TIM_IT_Update, ENABLE);

	 /* TIM4 enable counter */
	 TIM_Cmd(TIM4, ENABLE);
}

void NVIC_Configuration(void)
{
  NVIC_InitTypeDef NVIC_InitStructure;

  /* Enable the TIM4 gloabal Interrupt */
  NVIC_InitStructure.NVIC_IRQChannel = TIM4_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}

void TIM4_IRQHandler(void)
{
  if (TIM_GetITStatus(TIM4, TIM_IT_Update) != RESET)
  {
    TIM_ClearITPendingBit(TIM4, TIM_IT_Update);

    TIM_SetCompare1(TIM4, 1100 + (int)(servo_angle[0]) ); // PD.12
    TIM_SetCompare2(TIM4, 1060 + (int)(servo_angle[1]) ); // PD.13
```

```
  TIM_SetCompare3(TIM4, 1060 + (int)(servo_angle[2]) ); // PD.14
  TIM_SetCompare4(TIM4, 400 + (int)(servo_angle[3]) ); // PD.15
  }
}

//bluetooth slave
void USART3_Init(uint32_t baudrate)
{
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART3, ENABLE);
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8 | GPIO_Pin_9; // Pins 8 (TX) and 9 (RX)
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
        GPIO_Init(GPIOD, &GPIO_InitStructure);

        GPIO_PinAFConfig(GPIOD, GPIO_PinSource8, GPIO_AF_USART3);
        GPIO_PinAFConfig(GPIOD, GPIO_PinSource9, GPIO_AF_USART3);

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

        USART_ITConfig(USART3, USART_IT_RXNE, ENABLE);

        NVIC_InitStructure.NVIC_IRQChannel = USART3_IRQn;
        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
        NVIC_Init(&NVIC_InitStructure);

        USART_Cmd(USART3, ENABLE);
}
void USART3_IRQHandler(void){

        if( USART_GetITStatus(USART3, USART_IT_RXNE) )
        {
                static uint8_t cnt = 0;
                char t = USART3->DR;

                //data akan di tampung sampai mendapatkan karakter s dan dibawah indek string yang didefinisikan
                if( (t !='s') && (cnt < MAX_STRLEN) ){
                                        received_string[cnt] = t; // data register ditampung dalam variabel
string
                                        cnt++; // counter up
                        }
                        else{ // setelah data di tampung maka data dapat di tampilkan (digunakan)
                        cnt = 0; // counter reset
                        switch(angka)
                        {
                        case 0 : data[angka]=atoi(received_string);angka+=1; break;
                        case 1 : data[angka]=atoi(received_string);angka=0; break;
                        //case 2 : data[angka]=atoi(received_string);angka=0; break;
                        }
                        }
                USART_ClearFlag(USART3, USART_FLAG_RXNE);
        }
}

//usart1
void USART6_Init(uint32_t baudrate){
```

```
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART6, ENABLE);

	RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);

	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7; // Pins 10 (TX) and 11 (RX)
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
	GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
	GPIO_Init(GPIOC, &GPIO_InitStructure);

	GPIO_PinAFConfig(GPIOC, GPIO_PinSource6, GPIO_AF_USART6);
	GPIO_PinAFConfig(GPIOC, GPIO_PinSource7, GPIO_AF_USART6);

	USART_InitStructure.USART_BaudRate = baudrate;
	USART_InitStructure.USART_WordLength = USART_WordLength_8b;
	USART_InitStructure.USART_StopBits = USART_StopBits_1;
	USART_InitStructure.USART_Parity = USART_Parity_No;
	USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
	USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
	USART_Init(USART6, &USART_InitStructure);

	USART_Cmd(USART6, ENABLE);
}

void cek_data(USART_TypeDef* USARTx, const char *pFormat, ... )
{
   va_list ap;
   char pStr[100];

   va_start(ap, pFormat);
   vsprintf(pStr, pFormat, ap);
   va_end(ap);

   int i=0;
   int n = strlen(pStr);
   for(i=0;i<n;i++)
   {
      USART_SendData(USARTx, (uint8_t)pStr[i]);
      while (USART_GetFlagStatus(USARTx, USART_FLAG_TC) == RESET);
   }
}
void USART_puts(USART_TypeDef* USARTx, volatile char *s){

	while(*s){
		// wait until data register is empty
		while( !(USARTx->SR & 0x00000040) );
		USART_SendData(USARTx, *s);
		*s++;
	}
}
```

Lampiran 4. Datasheet Komponen