TUGAS AKHIR

ANALISA *INVERSE KINEMATICS* PADA PROTOTYPE *3-DOF ARM ROBOT* DENGAN METODE *ADAPTIVE NEURO FUZZY INFERENCE SYSTEM* (ANFIS)

Diajukan guna melengkapi sebagian syarat dalam mencapai
gelar Sarjana Strata Satu (S1)

Disusun Oleh:

Nama	: Indra Sulaeman
N.I.M.	: 41419110020
Pembimbing	: Akhmad Wahyu Dani, ST., MT

PROGRAM STUDI TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS MERCU BUANA
JAKARTA
2021

HALAMAN PENGESAHAN

ANALISA *INVERSE KINEMATICS* PADA PROTOTYPE *3-DOF ARM ROBOT* DENGAN METODE *ADAPTIVE NEURO FUZZY INFERENCE SYSTEM* (ANFIS)

UNIVERSITAS
MERCU BUANA

Disusun Oleh:

Nama	: Indra Sulaeman
N.I.M.	: 41419110020
Program Studi	: Teknik Elektro

Mengetahui,
Pembimbing Tugas Akhir

(Akhmad Wahyu Dani, S.T., M.T.)

Kaprodi Teknik Elektro	Koordinator Tugas Akhir
(Dr. Setiyo Budiyanto, S.T., M.T.)	(Muhammad Hafizd Ibnu Hajar, S.T., M.Sc)

HALAMAN PERNYATAAN

Yang bertanda tangan di bawah ini,

Nama : Indra Sulaeman

NIM : 41419110020

Jurusan : Teknik Elektro

Judul : Analisa *Inverse Kinematics* pada Prototype 3-DOF *Arm Robot* dengan Metode *Adaptive Neuro Fuzzy Inference System* (ANFIS)

Dengan ini menyatakan bahwa hasil penulisan Skripsi yang telah saya buat ini merupakan hasil karya saya sendiri dan benar keasliannya. Apabila ternyata dikemudian hari penulisan Skripsi ini merupakan plagiat atau penjiplakan terhadap karya orang lain, maka saya bersedia mempertanggungjawabkan sekaligus bersedia menerima sanksi berdasarkan aturan di Universitas Mercu Buana.

Demikian pernyataan ini saya buat dalam keadaan sadar dan tidak dipaksakan.

Penulis,

(Indra Sulaeman)

NIM. 41419110020

KATA PENGANTAR

Pertama saya mengucapkan terima kasih kepada Allah S.W.T yang telah memberikan rahmat dan karunia-nya kepada penulis sehingga dapat menyelesaikan skripsi yang berjudul "Analisa *Inverse Kinematics* pada Prototype 3-DOF *Arm Robot* dengan Metode *Adaptive Neuro Fuzzy Inference System* (ANFIS)". Skripsi ini merupakan syarat untuk memperoleh gelar sarjana pada Program Studi Teknik Elektro Fakultas Teknik Universitas Mercu Buana.

Dalam penyusunan skripsi ini penulis menyadari sebagai manusia biasa tidak lepas dari kesalahan dan kekurangan akibat keterbatasan pengetahuan serta pengalaman. Penyusunan skripsi ini tidak lepas dari bimbingan, bantuan, dan dukungan berbagai pihak. Oleh karena itu, dalam kesempatan ini penulis mengucapkan puji syukur atas berkat dan karunia Tuhan Yang Maha Esa yang telah mencurahkan karunianya serta ingin berterima kasih kepada semua pihak yang telah membantu dalam penyusunan skripsi ini terutama kepada:

1. Bapak Akhmad Wahyu Dani, S.T., M.T. selaku dosen pembimbing yang telah membantu dan meluangkan waktu untuk masukan dan bimbingan.
2. Bapak Dr. Setiyo Budiyanto, S.T., M.T. selaku dosen mata kuliah tugas akhir yang telah memberikan kesempatan dan petunjuk dalam penyusunan serta analisa penelitian ini.
3. Kepada orang tua tercinta yang telah memberi semangat, do'a dan dukungan moral yang tiada henti-hentinya kepada penulis serta nasehat yang membangun.
4. Kepada istriku tercinta, Ikbar Ar-rumaisha yang telah memberikan semangat dan do'a serta dukungan yang tiada henti.
5. Sahabat seperjuangan yang telah banyak membantu dalam menyelesaikan laporan tugas akhir ini terutama Sondy Marsubedi Wisyudha yang selalu mendukung dan memberi semangat dalam penyusunan tugas akhir ini agar dapat menyelesaikan tepat pada waktunya.
6. Dan semua teman-teman angkatan 35 Universitas Mercu Buana yg telah berjuang bersama-sama. -

Dalam penulisan laporan ini masih jauh dari sempurna, maka dari itu kritik dan saran sangat membangun penulis untuk penyempurnaan laporan ini. Akhir kata penulis ucapkan terima kasih dan semoga laporan ini berguna bagi pengembangan teknologi di masa depan.

Jakarta, 28 Januari 2021

Penulis,

(Indra Sulaeman)

NIM. 41419110020

ABSTRAK

Perkembangan teknologi yang sudah semakin maju memberi dampak pada perkembangan sistem kendali lengan robot. Sistem kendali lengan robot yang digunakan bermacam-macam. Beberapa teknik yang umum digunakan adalah metode *forward kinematics* dan *inverse kinematics*. Namun, metode *inverse kinematics* memiliki kompleksitas yang tinggi karena diperlukan perhitungan fungsi turunan dari *forward kinematics*.

Metode terbaru yang saat ini dapat digunakan yaitu metode *Adaptive Neuro Fuzzy Inference System* (ANFIS). Pada umumnya ANFIS dilakukan untuk memprediksi data keluaran berdasarkan pelatihan hubungan data masukan dan data keluaran yang dimuat dalam *dataset*. Penerapan metode ANFIS yang dilakukan pada penelitian ini sebagai solusi *inverse kinematics* untuk mengurangi kompleksitas dari metode tersebut.

Berdasarkan hasil pengujian yang dilakukan bahwa RMSE pada *inverse kinematics* bernilai 25.36137736 sedangkan RMSE pada metode ANFIS bernilai 42.18690271. Meskipun nilai *error* yang dihasilkan dari metode ANFIS cukup besar akibat beberapa faktor *error* yang mempengaruhinya, metode ANFIS ini terbukti dapat digunakan untuk kontrol lengan robot 3-DOF.

Kata kunci: lengan robot 3-DOF, *inverse kinematics*, ANFIS

ABSTRACT

Increasingly advanced technology has an impact on the development of the robot arm control system. Various robotic arm control systems are used. Some of the techniques commonly used are the forward kinematics and inverse kinematics methods. However, the inverse kinematics method has high complexity because it requires calculating the derived function of the forward kinematics.

The newest method currently available is the Adaptive Neuro Fuzzy Inference System (ANFIS) method. In general, ANFIS is carried out to predict the output data based on training on the relationship between input data and output data contained in the dataset. The ANFIS method applied in this study can be a solution to inverse kinematics to reduce the complexity of the method.

Based on the results of tests conducted, the RMSE in the inverse kinematics is 25.36137736 while the RMSE in the ANFIS method is 42.18690271. Although the error value generated from the ANFIS method is quite large due to several error factors that influence it, this ANFIS method is proven to be used for 3-DOF robot arm control.

Keywords: *3-DOF robot arm, inverse kinematics, ANFIS*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Robot sudah dikenal masyarakat luas akan kegunaannya dan kemampuannya. Dalam dunia industri, robot yang khususnya lengan robot, digunakan untuk beberapa macam pekerjaan. Contohnya pada proses pengecatan, pengelasan, pemindahan barang, dan sebagainya. Tingkat akurasi dan presisi yang tinggi ini yang menjadi daya tarik perusahaan untuk menggunakan lengan robot sebagai bagian dalam proses produksi.

Kontrol pergerakan lengan robot dapat dilakukan dengan cara mengatur sudut *joint* atau menentukan koordinat yang diinginkan. Analisa kinematika lengan robot dapat menggunakan parameter Denavit-Hartenberg. Kinematika tersebut terdiri dari *Forward Kinematics* dan *Inverse Kinematics*.

Forward kinematics merupakan kinematika lengan robot dengan *input* berupa nilai sudut tiap *joint* sehingga didapatkan nilai output berupa koordinat *end-effector*. Sedangkan *inverse kinematics*, merupakan kinematika pergerakan robot dengan *input* berupa nilai koordinat *end-effector* sehingga didapatkan *output* berupa nilai sudut masing-masing *joint* robot. Kinematika ini sudah lazim dalam kontrol pergerakan robot.

Kekurangan *forward kinematics* yaitu operator akan kesulitan untuk mengatur posisi *end-effector* pada koordinat yang diinginkan karena pengaturan berfokus pada nilai sudut masing-masing *joint*. Sedangkan pada *inverse kinematics*, walaupun posisi *end-effector* dapat berada pada posisi koordinat yang diinginkan, proses perhitungan kinematikanya akan lebih kompleks.

Pada tahun 1993, Jyh-Shing R Jang melakukan penelitian mengenai perpaduan antara *fuzzy logic* dengan *neural network* yang dikenal sebagai *Adaptive Neuro Fuzzy Inference System* (ANFIS). ANFIS dapat digunakan untuk memprediksi suatu data berdasarkan dataset yang telah dilatih sebelumnya.

Beberapa jurnal penelitian saat ini sudah menerapkan metode ANFIS untuk kontrol pergerakan robot sebagai pengganti *inverse kinematics*. Namun, penelitian tersebut masih berupa simulasi dalam program Matlab dan belum diterapkan pada robot yang sesungguhnya.

Maka dari itu, penulis bermaksud menulis penelitian ini dengan judul "**ANALISA *INVERSE KINEMATICS* PADA *PROTOTYPE 3-DOF ARM ROBOT* DENGAN METODE *ADAPTIVE NEURO FUZZY INFERENCE SYSTEM* (ANFIS)**". Penelitian ini dilakukan untuk menganalisa kontrol pergerakan lengan robot menggunakan metode ANFIS yang diterapkan pada prototype robot 3-DOF.

1.2. Rumusan Masalah

Berdasarkan latar belakang, adapun perumusan masalah dalam penyusunan penelitian ini adalah sebagai berikut:

1. Bagaimana membuat *inverse kinematics* berdasarkan parameter Denavit-Hartenberg pada robot lengan 3-DOF?
2. Bagaimana mengaplikasikan *Adaptive Neuro Fuzzy Inference System* sebagai solusi dari *inverse kinematics* untuk lengan robot 3-DOF?
3. Bagaimana nilai *error* yang dihasilkan dalam tiap metode?

1.3. Tujuan Penelitian

Adapun tujuan dari penulisan tugas akhir ini adalah sebagai berikut:

1. Membuat perhitungan *inverse kinematics* berdasarkan parameter Denavit-Hartenberg pada robot lengan 3-DOF.
2. Mengaplikasikan *Adaptive Neuro Fuzzy Inference System* (ANFIS) pada lengan robot sebagai solusi kinematika robot.

1.4. Batasan Masalah

Agar pembahasan penelitian ini lebih terarah, penulis melakukan pembatasan masalah sebagai berikut:

1. Perhitungan *inverse kinematics* hanya menggunakan konvensi Denavit-Hartenberg.
2. Penelitian ini tidak membahas derajat kebebasan selain 3-DOF.

1.5. Metodologi Penelitian

Dalam melakukan sebuah penelitian, diperlukan metode serta tahapan-tahapan yang jelas dan sistematis untuk dapat memperoleh suatu hasil penelitian yang baik. Tahapan dalam melakukan penelitian ini dapat dijabarkan sebagai berikut:

1. Studi literatur

   Studi literatur dilakukan untuk memahami konsep dasar ANFIS, perhitungan *inverse kinematics*, pengaplikasian ANFIS pada lengan robot, dan pemrograman python pada Raspberry Pi.

2. Perancangan dan pembuatan robot

   Perancangan menggunakan software 3D SolidWorks untuk membantu pembuatan gambaran dan penentuan komponen yang akan dipakai. Kemudian, rancangan tersebut direalisasikan dengan menggnakan 3D printer agar memudahkan pada pembuatan dan dapat menyesuaikan rancangan yang telah dibuat.

3. Perancangan program

   Perancangan program dilakukan agar memudahkan dalam pembuatan program dan memudahkan analisa program.

4. Simulasi

   Simulasi dilakukan menggunakan software RoboAnalyzer sebagai alat pembanding hasil dari pengujian.

5. Pengujian dan analisa

   Pengujian dilakukan dengan membandingkan *input* dengan *output* koordinat *end-effector* berdasarkan metode *inverse kinematics* serta metode ANFIS.

1.6. Sistematika Penulisan

Untuk mendapatkan gambaran secara umum tentang pokok pembahasan Tugas Akhir ini, penulis membaginya dalam beberapa bab yang secara garis besar adalah sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini akan dibahas mengenai latar belakang masalah, perumusan masalah, tujuan penelitian, batasan masalah, metode penelitian, serta sistematika penulisan yang jadi latar belakang Tugas Akhir ini disusun.

BAB II LANDASAN TEORI

Pada bab ini akan diuraikan landasan teori mengenai jurnal – jurnal penelitian mengenai aplikasi ANFIS dalam robotika, penjelasan mengenai Raspberry Pi, serta komponen pada robot.

BAB III PEMODELAN DAN PERANCANGAN SISTEM

Pada bab ini, penulis akan menguraikan perancangan ANFIS, parameter Denavit-Hartenberg berdasarkan robot yang telah dibuat, rangkaian keseluruhan pada robot, dan perancangan ANFIS.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini dijelaskan mengenai hasil pengujian penerapan *inverse kinematics* serta ANFIS pada lengan robot 3-DOF.

BAB V PENUTUP

Pada bab ini, penulis memberikan kesimpulan akhir dari penelitian serta saran yang diajukan untuk pengembangan sistem yang diteliti untuk perbaikan proses pengujian selanjutnya.

BAB II
LANDASAN TEORI

2.1. Tinjauan Pustaka

Sebelum melakukan penelitian lebih lanjut pada tugas akhir ini, penulis melakukan observasi terhadap penelitian terdahulu dengan melihat jurnal internasional yang berkaitan dengan implementasi *Adaptive Neuro Fuzzy Inference System* pada robot lengan. Berikut ini disajikan beberapa peelitian dalam jurnal-jurnal internasional sebagai berikut:

1. Pada tahun 2019, jurnal penelitian telah dibuat oleh Jacket Demby's, Yixiang Gao, dan G.N. DeSouza dalam *IEEE International Conference on Fuzzy Systems* (FUZZ-IEEE) berjudul "***A Study on Solving the Inverse Kinematics of Serial Robots using Artificial Neural Network and Fuzzy***". Pada jurnal ini disebutkan bahwa penelitian telah dilakukan untuk menganalisa robot serial 4, 5, 6, dan 7-DOF dengan kombinasi sambungan perismatik dan revolute. Tujuannya bukan untuk memprediksi pose berdasarkan lintasan tertentu (linier atau sebaliknya), namun untuk mempelajari seluruh ruang kerja robot. Tujuan ini dengan lebih baik membahas penggunaan *inverse kinematics* robot di dunia nyata, di mana setiap pose *end-effector* harus dapat dijangkau dari pose apapun saat ini. Dari percobaan tersebut dapat disimpulkan bahwa baik ANN dan ANFIS menyatu sampai tingkat tertentu ke fungsi *inverse kinematics*.

2. Pada tahun 2018, jurnal penelitian telah dibuat oleh Muhammad Gaafar, Ahmed Maged, M. Magdy, Nader A. Mansour, dan Ahmed El-Betar dalam *International Japan-Africa Conference on Electronics, Communications, and Computations* (JAC-ECC) berjudul "***Design and Analysis of Experiments in ANFIS Modeling of 3-DOF Planner Manipulator***". Pada jurnal ini penelitian menggunakan metodologi *Design of Experiment* (DoE) untuk mengoptimalkan parameter ANFIS yang signifikan ketika diterapkan pada solusi *inverse kinematics*.
3. Pada tahun 2017, jurnal penelitian telah dibuat oleh Ihtisham Qasim Kalimullah, Arvind Desikan, dan V. Kalaichelvi dalam *International Conference on Control, Automation and Robotics* berjudul "***Analysis of A Proposed Algorith for Point to Point Control of A 3-DOF Robot Manipulator***". Pada jurnal ini algoritma telah dikembangkan untuk model robot 3-DOF dengan kontrol *point – to – point* dari robot spasial. Selanjutnya, studi simulasi dilakukan dan juga diuji dalam realita. Serangkaian persamaan *inverse kinematics* telah dikembangkan yang memungkinkan diperolehnya hubungan antara sudut joint, panjang links, dan posisi yang diinginkan terhadap posisi awal *(homing position)*. Hal ini kemudian dibandingkan dengan algoritma yang dirancang menggunakan matrik rotasi dengan prediksi *incremental* yang kecil untuk mencapai kesalahan minimum.
4. Pada tahun 2017, jurnal penelitian telah dibuat oleh Pannawit Srisuk, Adna Sento, dan Yuttana Kitjaidure dalam *14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology* (ECTI-CON) berjudul "***Forward Kinematic-like Neural Network for Solving the 3D Reaching Inverse Kinematics Problems***". Pada jurnal ini disajikan solusi *inverse kinematics* berdasarkan *neural networks*. Pada umumnya pendekatan *neural networks* menggunakan data posisi *end-effector* sebagai *input* dan sudut joint sebagai output untuk melatih *neural networks* memetakan masukan ke keluaran. Namun, metode yang diusulkan membuat jaringan khusus dari persamaan *forward kinematics*. Struktur khusus ini membuat jaringan seperti pencari posisi dengan

kemampuan untuk menyesuaikan sudut joint secara otomatis hingga *end-effector* mencapai posisi yang diinginkan melalui *backpropagation* dengan algoritma kecepatan pembelajaran variabel. Kemudian, solusi sudut dapat ditemukan dari bobot akhir dan nilai bias. Selain itu, jaringan yang diusulkan menggunakan lebih sedikit jumlah *neuron* dan jumlah ruang solusi tidak bergantung pada data pelatihan. Terakhir, untuk mengevaluasi algoritma kinerja, Program MATLAB digunakan untuk mendemonstrasikan gerakan lengan robotik 4-DOF dalam 3 dimensi. Alhasil, algoritma yang diusulkan dapat membantu lengan robotik bergerak ke posisi yang diinginkan (jangkauan 3D) dengan cepat dan tepat.

5. Pada tahun 2016, jurnal penelitian telah dibuat oleh Mihai Crenganis, Radu Breaz, Gabriel Racz, dan Octavian Bologa dalam *International Conference on Computers Communications and Control* (ICCCC) berjudul "***Adaptive Neuro-Fuzzy Inference System for Kinematics Solutions of Redundant Robots***". Pada jurnal ini disajikan aspek-aspek tentang implementasi *Adaptive Neuro-Fuzzy Inference System* (ANFIS) dalam robot serial redundan. Untuk mendapatkan solusi optimal untuk *invers kinematics* robot redundan, persamaan matematika didasarkan pada metode lingkaran redundansi. Model ANFIS digunakan untuk menentukan posisi siku robot pada lingkaran redundansi sehingga robot dapat menghindari rintangan yang berbeda.

2.2. *Adaptive Neuro Fuzzy Inference System* (ANFIS)

Model *fuzzy* Sugeno telah diusulkan untuk pendekatan sistematis untuk menghasilkan aturan *fuzzy* dari kumpulan data *input-output* tertentu. Aturan *fuzzy* Sugeno yang khas dapat diekspresikan dalam bentuk berikut:

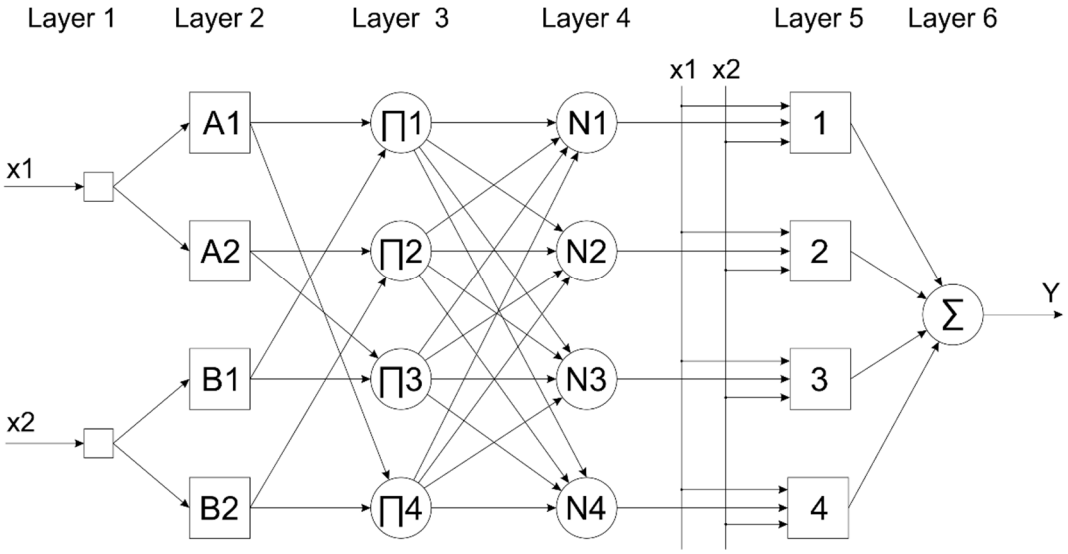
IF x_1 is A_1
AND x_2 is A_2
...
AND x_m is A_m
THEN $y = f(x_1, x_2, \dots, x_m)$

di mana $(x_1, x_2, \ldots, x_m)$ adalah variabel *input*; $A_1, A_2, \ldots, A_m$ adalah set *fuzzy*; dan y adalah baik konstanta atau fungsi linier dari variabel *input*. Jika y adalah konstanta, maka didapatkan model *fuzzy* Sugeno orde-nol di mana konsekuensi aturan ditentukan oleh singleton. Jika y adalah polinomial orde pertama, misal:

$$y = k_0 + k_1 x_1 + k_2 x_2 + \cdots + k_m x_m \quad (2.1)$$

didapatkan model *fuzzy* Sugeno orde pertama.

ANFIS Jang biasanya diwakili oleh jaringan neural feedforward enam layer. Gambar di atas ini menunjukkan arsitektur ANFIS yang sesuai dengan model *fuzzy* Sugeno orde satu.



Gambar 2.1 Struktur ANFIS

(Michael Negnevitsky, 2005)

Untuk mempermudah, asumsikan bahwa ANFIS memiliki dua *input* - x1 dan x2 - dan satu output - y. Setiap *input* diwakili oleh dua set *fuzzy*, dan *output* oleh polinomial orde pertama. Empat aturan implementasi ANFIS:

Rule 1:
IF x1 is A1
AND x2 is B1
THEN $y = f_1 = k_{10} + k_{11}x1 + k_{12}x2$

Rule 2:
IF x1 is A2
AND x2 is B2
THEN $y = f_2 = k_{20} + k_{21}x1 + k_{22}x2$

Rule 3:
IF x1 is A2
AND x2 is B1
THEN $y = f_3 = k_{30} + k_{31}x1 + k_{32}x2$

Rule 4:
IF x1 is A1
AND x2 is B2
THEN $y = f_4 = k_{40} + k_{41}x1 + k_{42}x2$

di mana x1, x2 adalah variabel *input*; A1 dan A2 adalah himpunan *fuzzy* dalam X1; B1 dan B2 adalah himpunan *fuzzy* dalam X2; dan k_{i0}, k_{i1} dan k_{i2} adalah seperangkat parameter yang ditentukan untuk rule i.

Layer 1 adalah **layer *input***. Neuron di lapisan ini hanya meneruskan sinyal *crisp* eksternal ke Layer 2. Yaitu,

$$y_i^{(1)} = x_i^{(1)} \tag{2.2}$$

di mana $x_i^{(1)}$ adalah *input* dan $y_i^{(1)}$ adalah output dari *input* neuron i pada Layer 1.

Layer 2 adalah **layer fuzzifikasi**. Neuron pada lapisan ini melakukan fuzzifikasi. Dalam model Jang, fuzzifikasi neuron memiliki ***bell activation function***. *Bell Activation Function*, yang memiliki bentuk lonceng biasa, ditetapkan sebagai:

$$y_i^{(2)} = \frac{1}{\left(1+\left(\frac{x_i^{(2)}-a_i}{c_i}\right)\right)^{2b_i}} \tag{2.3}$$

dimana $x_i^{(2)}$ adalah *input* dan $y_i^{(2)}$ adalah output dari neuron i pada Layer 2; dan a_i, b_i dan c_i adalah parameter yang mengontrol, masing-masing, pusat, lebar dan kemiringan *bell activation function* neuron i.

Layer 3 adalah **rule layer**. Setiap neuron pada lapisan ini sesuai dengan aturan *fuzzy* tipe Sugeno tunggal. Sebuah neuron aturan menerima *input* dari masing-masing neuron fuzzifikasi dan menghitung kekuatan menembak dari aturan yang diwakilinya. Dalam ANFIS, konjungsi dari anteseden aturan dievaluasi oleh produk operator. Dengan demikian, output neuron i pada Layer 3 diperoleh sebagai,

$$y_i^{(3)} = \prod_{j=1}^{k} x_{ji}^{(3)} \tag{2.4}$$

di mana $x_{ji}^{(3)}$ adalah *input* dan $y_i^{(3)}$ adalah output dari *input* neuron i pada Layer 3. Contohnya,

$$y_{\Pi 1}^{(3)} = \mu_{A1} \times \mu_{B1} = \mu_1 \tag{2.5}$$

di mana nilai μ_1 mewakili kekuatan tembak, atau nilai kebenaran, dari Rule 1.

Layer 4 adalah **layer normalisasi**. Setiap neuron di lapisan ini menerima *input* dari semua neuron di lapisan aturan, dan menghitung ***normalized firing strength*** dari aturan yang diberikan.

Normalized firing strength atau kekuatan menembak dinormalisasi adalah rasio kekuatan menembak dari aturan yang diberikan dengan jumlah kekuatan menembak semua aturan. Ini merupakan kontribusi dari aturan yang diberikan pada hasil akhir.

Dengan demikian, output neuron i pada Layer 4 ditentukan sebagai,

$$y_i^{(4)} = \frac{x_{ii}^{(4)}}{\sum_{j=1}^{n} x_{ji}^{(4)}} = \frac{\mu_i}{\sum_{j=i}^{n} \mu_j} = \overline{\mu}_i \tag{2.6}$$

di mana $x_{ji}^{(4)}$ adalah *input* dari neuron *j* yang terletak di Layer 3 ke neuron i di Layer 4, dan n adalah jumlah total neuron aturan. Sebagai contoh,

$$y_{N1}^{(4)} = \frac{\mu_1}{\mu_1 + \mu_2 + \mu_3 + \mu_4} = \bar{\mu}_1 \tag{2.7}$$

Layer 5 adalah **layer defuzzifikasi**. Setiap neuron pada layer ini terhubung ke masing-masing neuron normalisasi, dan juga menerima *input* awal, x1 dan x2. Neuron defuzzifikasi menghitung nilai konsekuensi tertimbang dari aturan tertentu sebagai,

$$y_i^{(5)} = x_i^{(5)}[k_{i0} + k_{i1}x1 + k_{i2}x2] = \bar{\mu}_i[k_{i0} + k_{i1}x1 + k_{i2}x2] \tag{2.8}$$

di mana $x_i^{(5)}$ adalah *input* dan $y_i^{(5)}$ adalah output dari defuzzifikasi neuron i di Layer 5, dan k_{i0}, k_{i1} dan k_{i2} adalah seperangkat parameter konsekuen dari aturan i.

Layer 6 diwakili oleh **neuron penjumlahan tunggal**. Neuron ini menghitung jumlah output dari semua neuron defuzzifikasi dan menghasilkan output ANFIS keseluruhan, y,

$$y = \sum_{i=1}^{n} x_1^{(6)} = \sum_{i=1}^{n} \bar{\mu}_i[k_{i0} + k_{i1}x1 + k_{i2}x2] \tag{2.9}$$

Dengan demikian, ANFIS yang ditunjukkan pada Gambar 2.1 memang fungsional setara dengan model orde satu *fuzzy* Sugeno.

Namun, seringkali sulit atau bahkan tidak mungkin untuk menentukan aturan akibat dalam bentuk polinomial. Mudahnya, tidak perlu memiliki pengetahuan sebelumnya tentang parameter konsekuensi aturan untuk ANFIS

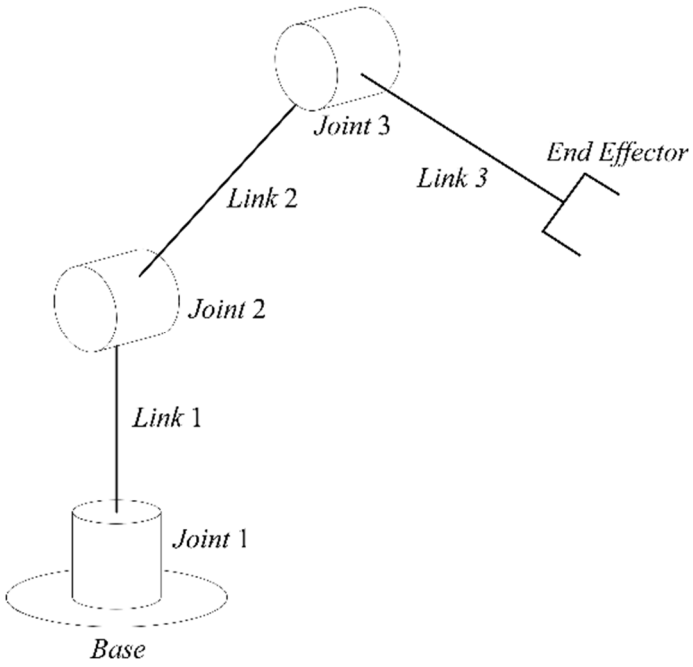
untuk menangani masalah. ANFIS mempelajari parameter ini dan mengatur fungsi keanggotaan (Michael Negnevitsky, 2005).

2.3. Lengan Robot

Lengan robot merupakan lengan mekanis, yang dapat diprogram, dengan fungsi yang mirip dengan lengan manusia. Lengan robot terdiri dari *link*, *joint*, dan *end-efector*. *Joint* merupakan titik terjadinya pergerakan robot, dapat berupa gerakan rotasi atau perismatik (gerak lurus). Sedangkan *link* merupakan sambungan antara *joint* dengan *joint* atau *joint* dengan *end-effector*. *End-effector* merupakan pangkal dari lengan robot yang dapat dipasangkan dengan *gripper* atau bahkan alat pengelas seperti pada robot industri.

2.3.1. Lengan Robot 3-DOF

Degree of Freedom (DOF) merupakan derajat independensi yang diperlukan untuk menyatakan posisi suatu sistem pada setiap saat (Widodo, 2017). Pada lengan robot, jumlah DOF menunjukkan banyaknya *joint* yang diterapkan pada lengan robot. Gambar 2.2 menunjukkan masing-masing joint berjenis rotasi.



Gambar 2.2 Lengan Robot 3-DOF

2.4. Denavit-Hartenberg Parameter

Untuk menjelaskan lokasi setiap link relatif terhadap bagian sebelahnya, perlu ditentukan frame yang dilampirkan ke setiap link. *Frame link* diberi nama dengan nomor sesuai dengan *link* yang dilampirkan. Artinya, *frame* {*i*} dipasang secara kuat ke *link i*.

2.4.1 *Link* Perantara dalam Deretan

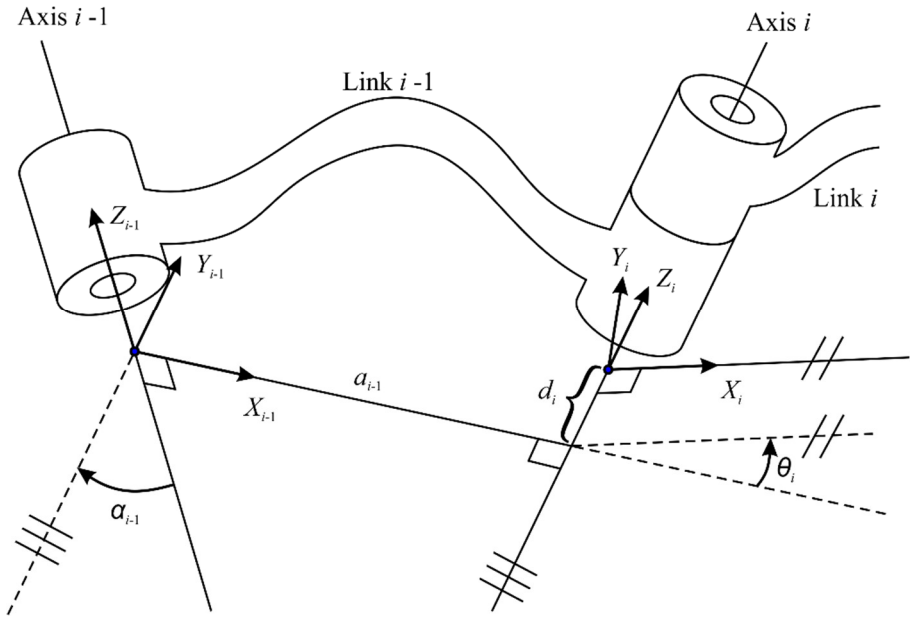
Kaidah yang akan digunakan untuk menentukan *frame* pada *link* adalah sebagai berikut: Sumbu Z dari *frame* {*i*}, disebut Z_i, bertepatan dengan *joint* sumbu *i*. Pangkal *frame* {i} terletak di mana garis tegak lurus memotong sumbu *i*. X_i menunjuk sepanjang a_i ke arah joint *i* sampai joint *i* + 1.

Dalam kasus $a_i = 0$, X_i normal untuk bidang Z_i dan Z_{i+1}. a_i didefinisikan sebagai bagian yang terukur dalam kaidah tangan kanan terhadap X_i dan dapat dilihat bahwa kebebasan memilih tanda dalam kasus ini sesuai dengan dua pilihan untuk arah X_i. Y_i dibentuk oleh kaidah tangan kanan untuk melengkapi *frame* ke-*i*.

2.4.2 *Link* Pertama dan Terakhir dalam Deretan

Frame yang terpasang ke dasar robot, atau *link* 0, yang disebut *frame* {0}. *Frame* ini tidak bergerak; untuk masalah kinematika lengan dapat dianggap sebagai kerangka acuan.

Frame {*0*} bersifat arbitrer, sehingga selalu menyederhanakan masalah untuk memilih Z_0 di sepanjang sumbu 1 dan untuk menemukan *frame* {*0*} sehingga bertepatan dengan *frame* {1} saat variabel gabungan 1 adalah nol. Menggunakan konvensi ini akan selalu memiliki $a_0 = 0{,}0$, $a_0 = 0{,}0$. Selain itu, ini memastikan bahwa $d_1 = 0{,}0$ jika sambungan 1 berputar, atau 01 = 0,0 jika sambungan 1 prismatik. Untuk joint n revolute, arah X_N dipilih sehingga sejajar dengan X_N_j jika = 0.0, dan asal frame {N} dipilih sehingga = 0.0. Untuk sambungan n prismatik, dipilih arah X_N sehingga = 0.0, dan asal rangka {N} dipilih pada perpotongan X_N.1 dan sumbu sambungan n bila d = 0.0. (John J. Craig, 2005)



Gambar 2.3 *Frame Link.*

(John J. Craig, 2005)

Berdasarkan penjelasan yang telah dijabarkan sebelumnya serta anotasi yang telah ditunjukkan pada gambar 2.4. maka dapat ditarik kesimpulan sebagai berikut:

- a_i = jarak dari sumbu Z_i sampai sumbu Z_{i+1} diukur sepanjang sumbu X_i
- α_i = sudut dari sumbu Z_i sampai sumbu Z_{i+1} diukur terhadap sumbu X_i
- d_i = jarak dari sumbu X_{i-1} sampai sumbu X_i diukur sepanjang sumbu Z_i
- θ_i = sudut dari sumbu X_{i-1} sampai sumbu X_i diukur terhadap sumbu Z_i

Notasi matriks berdasarkan parameter Denavit-Hartenberg adalah sebagai berikut:

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1} d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.10) $$

2.5. Raspberry Pi

Menurut halaman website resmi, Raspberry Pi merupakan komputer berbiaya rendah berukuran kartu kredit yang dapat dihubungkan ke monitor komputer atau TV dan menggunakan keyboard dan mouse standar. Ini adalah

perangkat kecil yang mumpuni yang memungkinkan orang-orang dari segala usia untuk menjelajahi komputasi dan untuk mempelajari cara memprogram dalam bahasa Scratch dan Python. Ia mampu melakukan semua yang diharapkan dari komputer desktop, dari menjelajahi internet dan memutar video definisi tinggi hingga membuat spreadsheet, pengolah kata, dan bermain game.

Raspberry Pi ini layaknya penggabungan anatara komputer dan mikrokontroler karena Raspberry Pi memiliki *General Purpose Input Output* (GPIO). GPIO ini dapat dipakai sebagai *input* sensor atau bahkan sebagai output pada aktuator tergantung kebutuhan penggunanya.

Beberapa pin power pada Raspberry Pi:

1. **GND** merupakan pin ***ground*** atau ***negative***.
2. **+5V** merupakan pin output yang mengalirkan tegangan positif 5 Volt yang telah melalui regulator.
3. **3V3** merupakan pin output yang mengalirkan tegangan positif 3,3 Volt yang telah melalui regulator.

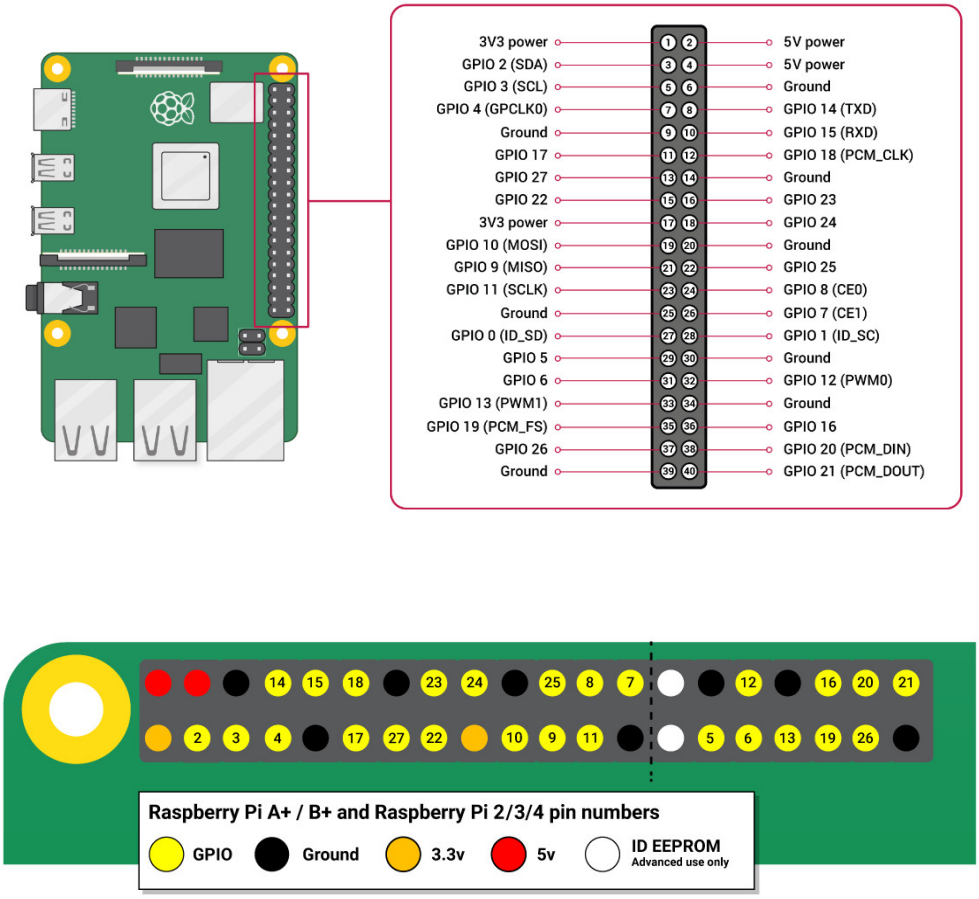
Raspberry Pi memiliki 28 General Purpose *Input* Output (GPIO) yang dapat digunakan sebagai *input* atau *output* digital. Beberapa fungsi khusus pada Raspberry Pi:

1. 6x *Universal Asynchronous Receiver-Transmitter* (UART)
2. 6x *Inter Integrated Circuit* (I2C)
3. 5x *Serial Pheriperal Interface* (SPI)
4. 1x *Secure Digital Input Output* (SDIO)
5. 1x *Display Paraller Interface* (DPI)
6. 1x *Pulse Code Modulation* (PCM)
7. 2x *Pulse Width Modulation* (PWM)
8. 3x *General Purpose Clock* (GPCLK)

Tabel 2.1 Fungsi Alternatif GPIO pada Raspberry Pi 4

(Raspberry Pi 4 Model B datasheet, 2019)

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	SPI3_CE0_N	TXD2	SDA6
1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
4	High	GPCLK0	SA1	DPI_D0	SPI4_CE0_N	TXD3	SDA3
5	High	GPCLK1	SA0	DPI_D1	SPI4_MISO	RXD3	SCL3
6	High	GPCLK2	SOE_N	DPI_D2	SPI4_MOSI	CTS3	SDA4
7	High	SPI0_CE1_N	SWE_N	DPI_D3	SPI4_SCLK	RTS3	SCL4
8	High	SPI0_CE0_N	SD0	DPI_D4	-	TXD4	SDA4
9	Low	SPI0_MISO	SD1	DPI_D5	-	RXD4	SCL4
10	Low	SPI0_MOSI	SD2	DPI_D6	-	CTS4	SDA5
11	Low	SPI0_SCLK	SD3	DPI_D7	-	RTS4	SCL5
12	Low	PWM0	SD4	DPI_D8	SPI5_CE0_N	TXD5	SDA5
13	Low	PWM1	SD5	DPI_D9	SPI5_MISO	RXD5	SCL5
14	Low	TXD0	SD6	DPI_D10	SPI5_MOSI	CTS5	TXD1
15	Low	RXD0	SD7	DPI_D11	SPI5_SCLK	RTS5	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	SPI6_CE0_N	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	SPI6_MISO	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	SPI6_MOSI	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	SPI6_SCLK	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	SDA6
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	SCL6
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT1	ARM_TDO	SPI3_CE1_N
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	SPI4_CE1_N
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	SPI5_CE1_N
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	SPI6_CE1_N



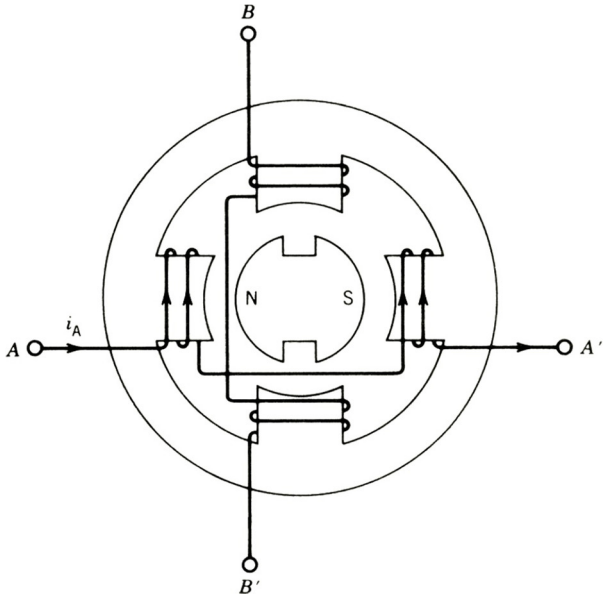
Gambar 2.4 Raspberry Pi *Pinout*
(www.raspberrypi.org)

2.6. *Stepping Motor*

Stepping motor berbeda dari motor DC standar sedemikian rupa sehingga memiliki dua kumparan independen yang dapat dikontrol secara independen. Akibatnya, *motor stepper* dapat digerakkan oleh impuls untuk melanjutkan satu langkah maju atau mundur, alih-alih gerakan kontinu yang mulus dalam motor DC standar. Jumlah tipikal langkah per revolusi adalah 200, menghasilkan ukuran langkah 1,8 °. Beberapa motor stepper memungkinkan setengah langkah, menghasilkan ukuran langkah yang lebih halus. Ada juga jumlah langkah maksimum per detik, bergantung pada beban, yang membatasi kecepatan motor stepper. (Thomas Bräunl, 2006)

2.6.1 Permanent Magnet Stepper Motor

Motor stepper magnet permanen memiliki konstruksi stator yang mirip dengan jenis reluktansi variabel tumpukan tunggal, tetapi rotor terbuat dari bahan magnet permanen. Gambar 2.5 menunjukkan motor stepper magnet permanen dua kutub. Kutub rotor sejajar dengan dua gigi stator (atau kutub) sesuai dengan eksitasi belitan. Gambar 2.5 menunjukkan kesejajaran jika belitan fase A tereksitasi. Jika eksitasi dialihkan ke fase B, rotor bergerak selangkah 90.



Gambar 2.5 *Permanen Magnet Stepper Motor*

(Paresh C. Sen, 2014)

Perhatikan bahwa polaritas arus penting dalam motor stepper magnet permanen, karena menentukan arah pergerakan motor. Gambar 2.5 mengilustrasikan posisi rotor untuk arus positif pada fase A. Sakelar ke arus positif pada belitan fase B akan menghasilkan langkah searah jarum jam, sedangkan arus negatif pada belitan fase B akan menghasilkan langkah berlawanan arah jarum jam. Sulit untuk membuat rotor magnet permanen kecil dengan sejumlah besar kutub, dan oleh karena itu motor stepper jenis ini dibatasi untuk ukuran langkah yang lebih besar dalam kisaran 30 hingga 90. Motor stepper magnet permanen memiliki inersia yang lebih tinggi dan oleh karena itu akselerasinya lebih lambat daripada motor stepper dengan reluktansi variabel. Laju langkah maksimum untuk motor stepper magnet permanen adalah 300 pulsa per detik, sedangkan untuk motor stepper

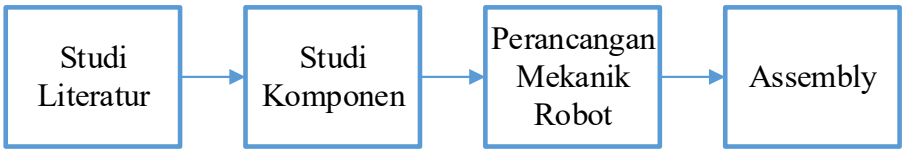
reluktansi variabel bisa setinggi 1200 pulsa per detik. Motor stepper magnet permanen menghasilkan lebih banyak torsi per arus stator ampere daripada motor stepper reluktansi variabel (Paresh C. Sen, 2014).

BAB III
PERANCANGAN ALAT

Bab ini menjelaskan mengenai perancangan lengan robot 3-DOF dengan metode ANFIS yang dikendalikan menggunakan Raspberry Pi 4 dengan input data kartesian posisi *end-effector*. Perancangan sistem pada bab ini terdiri dari dua bagian, yaitu perancangan perangkat keras (*hardware)* dan perancangan sistem.

3.1. Perancangan Mekanika Robot 3-DOF

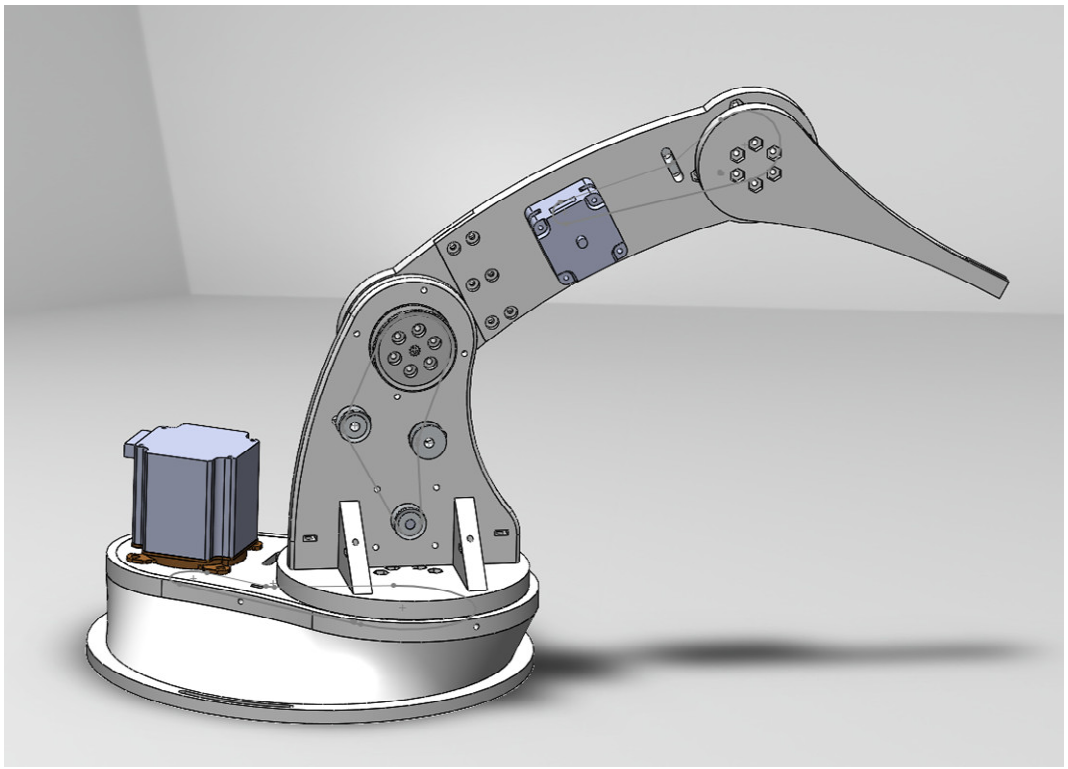
Perancangan mekanika robot dilakukan berdasarkan pada studi literatur serta studi komponen yang diperlukan untuk memudahkan dalam proses pembuatan. Kemudian dilakukan perancangan mekanik menggunakan 3D *software* untuk membantu pembuatan mekanika robot tersebut. Setelah pembuatan selesai maka dilakukan *assembly* atau perakitan komponen beserta bagian-bagian robot. Penjelasan mengenai alur pembuatan mekanika robot ini ditunjukkan pada gambar 3.1.



Gambar 3.1 Alur Perancangan Mekanika Robot

Perancangan mekanika robot dilakukan menggunakan software 3D SolidWorks. Pada mulanya komponen-komponen robot dibuat design 3D dengan software tersebut menggunakan dimensi yang sama dengan technical drawing.

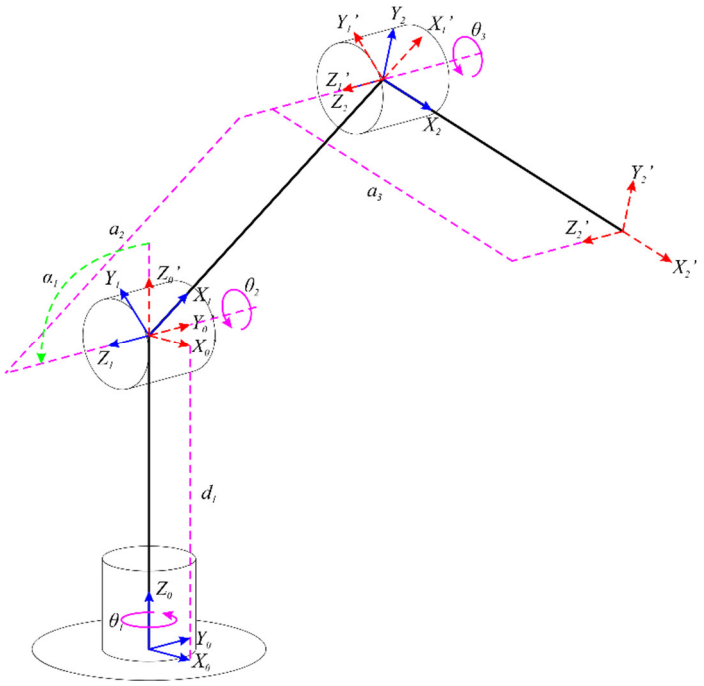
Kemudian rancangan mekanika robot disesuaikan dengan dimensi komponen-komponen serta massa komponen tersebut untuk menyesuaikan kemampuan masing-masing komponen. Hasil rancangan mekanika robot ditunjukkan pada gambar3.2.



Gambar 3.2 Rancangan 3D Mekanika Robot 3-DOF Menggunakan Software SolidWorks

3.2. Pemodelan Struktur Robot 3-DOF dengan DH Parameter

Pemodelan struktur dilakukan berdasarkan Parameter Denavit-Hartenberg. Tahap ini sangat diperlukan untuk mempermudah perhitungan rumus secara matematis. Notasi-notasi matematis ditentukan menggunakan kaidah tangan kanan serta aturan-aturan pada konvensi Denavit-Hartenberg.



Gambar 3.3 Pemodelan Robot 3-DOF berdasarkan Parameter DH

Tabel 3.1 Parameter Denavit pada Robot 3-DOF

No.	d_i	θ_i	a_i	α_i	$cos\theta_i$	$sin\theta_i$	$sin\alpha_i$	$sin\alpha_i$
1	d_1	θ_1	-	90°	C_1	S_1	0	1
2	-	θ_2	a_2	-	C_2	S_2	1	0
3	-	θ_3	a_3	-	C_3	S_3	1	0

Keterangan pada tabel 3.1 adalah sebagai berikut:

- d_1= jarak dari sumbu X_0 sampai sumbu X_1 diukur sepanjang sumbu Z_0 dengan nilai 139.2 mm.
- a_2 = jarak dari sumbu Z_1 sampai sumbu Z_2 diukur sepanjang sumbu X_1 dengan nilai 200 mm.
- a_3 = jarak dari sumbu Z_2 sampai sumbu Z_3 diukur sepanjang sumbu X_2 dengan nilai 179.2 mm.
- α_1 = sudut dari sumbu Z_0 sampai sumbu Z_1 diukur terhadap sumbu X_0 dengan nilai 90°.
- θ_1 = sudut dari sumbu X_0 sampai sumbu X_1 diukur terhadap sumbu Z_0 dengan batasan $0° \leq \theta_1 \leq 180°$.

- θ_2 = sudut dari sumbu X_1 sampai sumbu X_2 diukur terhadap sumbu Z_1 dengan batasan $-90° \leq \theta_2 \leq 90°$.
- θ_3 = sudut dari sumbu X_2 sampai sumbu X_3 diukur terhadap sumbu Z_2 dengan batasan $-90° \leq \theta_3 \leq 90°$.

Pada tabel 3.2 di bawah ini merupakan penjelasan mengenai penyederhanaan notasi bilangan trigonometri agar memudahkan dalam proses perhitungan matriks dengan penjelasan sebagai berikut:

Tabel 3.2 Penjelasan Penyederhanaan Notasi Trigonometri

$C_1 = cos\theta_1$	$S_1 = sin\theta_1$
$C_2 = cos\theta_2$	$S_2 = sin\theta_2$
$C_3 = cos\theta_3$	$S_3 = sin\theta_3$
$C_{23} = \cos(\theta_2 + \theta_3)$	$S_{23} = \sin(\theta_2 + \theta_3)$

Pemodelan kinematika menggunakan matriks Denavit-Hartenberg yang ditunjukkan pada persamaan (3.1)

$$ {}^{i-1}_{i}T = \begin{bmatrix} cos\theta_i & -sin\theta_i . cos\alpha_i & sin\theta_i . sin\alpha_i & a_i . cos\theta_i \\ sin\theta_i & cos\theta_i . cos\alpha_i & -cos\theta_i . sin\alpha_i & a_i . sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.1) $$

Matriks transofrmasi tersebut kemudian digunakan untuk mencari matriks transformasi masing-masing *joint* pada lengan robot 3-DOF, yaitu matriks transformasi ${}^{0}_{1}T$, ${}^{1}_{2}T$, ${}^{2}_{3}T$ dengan cara memasukkan nilai dari empat parameter DH yang ditunjukkan pada Tabel 3.1. ke dalam persamaan (3.1) sehingga didapat persamaan (3.2), (3.3), dan (3.4):

$$ {}^{0}_{1}T = \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2) $$

$$ {}^{1}_{2}T = \begin{bmatrix} C_2 & -S_2 & 0 & a_2 . C_2 \\ S_2 & C_2 & 0 & a_2 . S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3) $$

$$ {}^{2}_{3}T = \begin{bmatrix} C_3 & -S_3 & 0 & a_3.C_3 \\ S_3 & C_3 & 0 & a_3.S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4) $$

Setelah didapat matriks transformasi tiap sendi kemudian dicari matriks transformasi dari keseluruhan robot lengan lewat aturan perkalian matriks yang ditunjukkan pada persamaan (3.5) sampai dengan persamaan (3.18):

$$ {}^{0}_{3}T = {}^{0}_{1}T \times {}^{1}_{2}T \times {}^{2}_{3}T \quad (3.5) $$

$$ {}^{0}_{3}T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.6) $$

Matriks persamaan (3.6) merupakan matriks homogen yang berguna untuk mendeskripsikan sistem koordinat XYZ.

$$ {}^{0}_{3}T = \begin{bmatrix} C_1C_{23} & -C_1S_{23} & S_1 & C_1(a_2C_2 + a_3C_{23}) \\ S_1C_{23} & -S_1S_{23} & -C_1 & S_1(a_2C_2 + a_3C_{23}) \\ S_{23} & C_{23} & 0 & a_2S_2 + a_3S_{23} + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.7) $$

Dari persamaan (3.6) dan (3.7) didapatkan:

$$ p_x = C_1(a_2C_2 + a_3C_{23}) \quad (3.8) $$

$$ p_y = S_1(a_2C_2 + a_3C_{23}) \quad (3.9) $$

$$ p_z = a_2S_2 + a_3.S_{23} + d_1 \quad (3.10) $$

Selanjutnya, dari persamaan (3.5) sampai dengan persamaan (3.10) diturunkan guna memperoleh persamaan *inverse kinematics*. Persamaan inilah yang menjadi acuan pada perhitungan *inverse kinematics* oleh ANFIS pada nilai dari *variable set joint* yang perlu diraih tiap sendi untuk memposisikan dan mengorientasikan *end-effector* pada tempatnya. Persamaan (3.11) sampai dengan persamaan (3.14) menunjukkan persamaan *inverse kinematics* hasil penurunan persamaan *forward kinematics* untuk solusi *joint* pertama θ_1.

$$ {}^{0}_{1}\mathrm{T}^{-1}.\,{}^{0}_{3}\mathrm{T} = {}^{1}_{2}\mathrm{T}.\,{}^{2}_{3}\mathrm{T} \quad (3.11) $$

$$\begin{bmatrix} C_1.n_x + S_1.n_y & C_1.o_x + S_1.o_y & C_1.a_x + S_1.a_y & C_1p_x + S_1.p_y \\ n_z & o_z & a_z & p_z - d_1 \\ S_1.n_x - C_1.n_y & S_1o_x. - C_1.o_y & S_1.a_x - C_1.a_y & S_1.p_x - C_1.p_y \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} C_{23} & -S_{23} & 0 & a_3.C_{23} + a_2.C_2 \\ S_{23} & C_{23} & 0 & a_3.S_{23} + a_2.S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.12)$$

$$S_1.p_x - C_1.p_y = 0 \quad (3.13)$$

$$\theta_1 = tan2^{-1}(p_y, p_x) \quad (3.14)$$

Dengan tetap menggunakan persamaan (3.12) proses pencarian solusi dilajutkan untuk mencari solusi sudut $\theta_3$yang ditunjukkan pada persamaan (3.15) sampai dengan persamaan (3.20).

$$C_1p_x + S_1p_y = a_3C_{23} + a_2C_2 \quad (3.15)$$

$$p_z - d_1 = a_3S_{23} + a_2S_2 \quad (3.16)$$

$$S_1px - C_1p_y = 0 \quad (3.17)$$

$$\left(C_1p_x + S_1p_y\right)^2 + (p_z - d_1)^2 + \left(S_1px - C_1p_y\right)^2 = (a_3C_{23} + a_2C_2)^2 + (a_3S_{23} + a_2S_2)^2 + (0)^2 \quad (3.18)$$

$$C_3 = \frac{p_x^2+p_y^2+p_z^2+d_1^2-a_2^2-a_3^2-2d_zd_1}{2a_2a_3} \quad (3.19)$$

$$S_3 = \mp\sqrt{1 - (C_3)^2} \quad (3.20)$$

$$\theta_3 = tan2^{-1}(S_3, C_3) \quad (3.21)$$

Melihat persamaan (3.12) maka sudah tidak ditemukan lagi solusi langsung untuk mencari variable joint tersisa untuk sudut θ_2. Maka dilakukan lagi inverse matrix yang ditunjukkan pada persamaan (3.22) dan diringkas menjadi persamaan (3.24) dan (3.33)

$${}^1_2T^{-1}.\,{}^0_1T.^{-1}\,{}^0_3T = {}^2_3T \quad (3.22)$$

$$\begin{bmatrix} C_1C_2 & S_1C_2 & S_2 & -S_2d_1 - a_2 \\ -C_1S_2 & -S_1S_2 & C_2 & -C_2d_1 \\ S_1 & -C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} C_3 & -S_3 & 0 & a_3.C_3 \\ S_3 & C_3 & 0 & a_3.S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.23)$$

Hasil perkalian ruas kiri persamaan (3.23) kemudian dicarikan solusi langsungnya yang berkorespondensi letak pada ruas kanan persamaan (3.24) dan (3.26) menunjukkan solusi langsung yang dimaksud.

$$C_1C_2p_x + S_1C_2p_y + S_2p_z - S_2d_1 - a_2 = a_3C_3 \quad (3.24)$$

$$-C_1S_2p_x - S_1S_2p_y + C_2p_z - C_2d_1 = a_3S_3 \quad (3.25)$$

$$S_1p_x - C_1p_y = 0 \quad (3.26)$$

Dengan menggunakan metode eliminasi yang diterapkan pada persamaan (3.24) hingga (3.26) diperoleh persamaan (3.27) dan solusi untuk sudut θ_2 ditunjukkan pada persamaan (3.28).

$$C_2\left(-2a_2\left(C_1p_x + S_1p_y\right)\right) + S_2\left(2a_2(d_1 - p_z)\right) = -\left(p_x^2 + p_y^2 + p_z^2 + d_1^2 + a_2^2\right) + a_3^2 + 2p_zd_1 \quad (3.27)$$

Untuk memudahkan pencarian solusi, maka dibuat permisalan seperti pada persamaan (3.28) hingga (3.30).

$$A = \left(2a_2(d_1 - p_z)\right) \quad (3.28)$$

$$B = \left(-2a_2\left(C_1p_x + S_1p_y\right)\right) \quad (3.29)$$

$$C = -\left(p_x^2 + p_y^2 + p_z^2 + d_1^2 + a_2^2\right) + a_3^2 + 2p_zd_1 \quad (3.30)$$

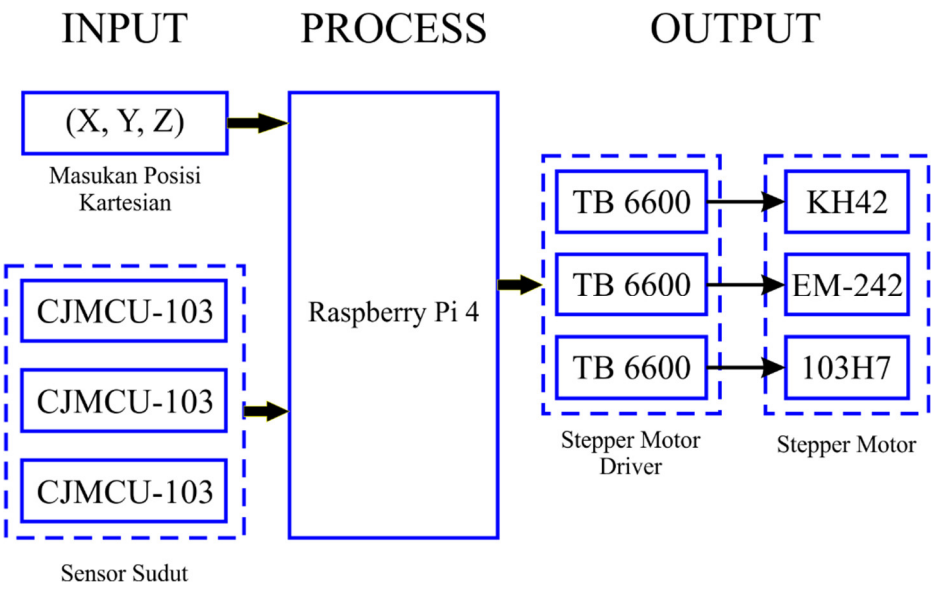
Maka, solusi untuk persamaan 3.27 adalah persamaan berikut,

$$A.S_2 + B.C_2 = C \quad (3.31)$$

$$\theta_2 = tan2^{-1}(B, A) - tan2^{-1}(\sqrt{A^2 + B^2 - C^2}, C) \quad (3.32)$$

3.3. Perancangan Sistem

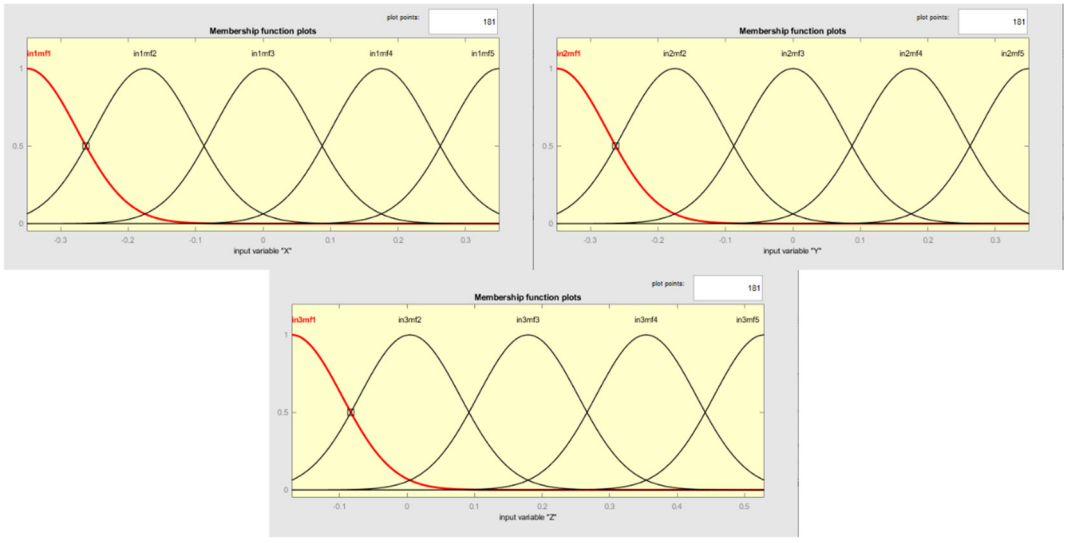
Secara garis besar, rancangan sistem terdiri dari beberapa komponen input, komponen proses, serta komponen output. Komponen input pada sistem terdiri dari 3 buah sensor sudut serta data masukan berupa posisi *end-effector* dalam ruang kartesian. Komponen proses berupa Raspberry Pi 4 yang telah diprogram untuk dapat mengolah data input sesuai dengan logika ANFIS. Kemudian hasil olahan data berupa nilai sudut joint dijadikan dalam bentuk frekuensi pulsa yang dikirimkan ke modul TB6600. TB6600 merupakan driver motor stepper sehingga posisi *end-effector* dapat sesuai dengan data input.



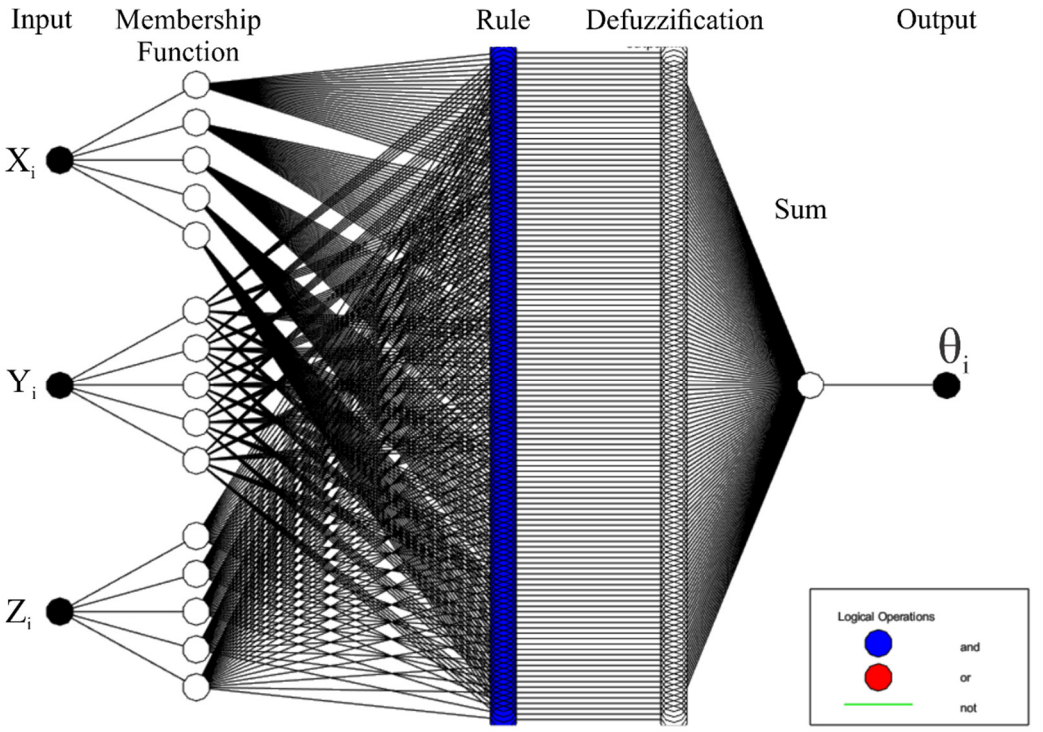
Gambar 3.4 Skema Perancangan Sistem Kontrol

3.3.1. Perancangan Struktur ANFIS

Perancangan struktur ANFIS diperlukan untuk menentukan komponen-komponen pada masing-masing layer ANFIS. Besarnya jumlah membership function dapat mempengaruhi tingkat akurasi hasil proses.



Gambar 3.5 *Membership Function* pada ANFIS untuk *Input* X, Y, dan Z



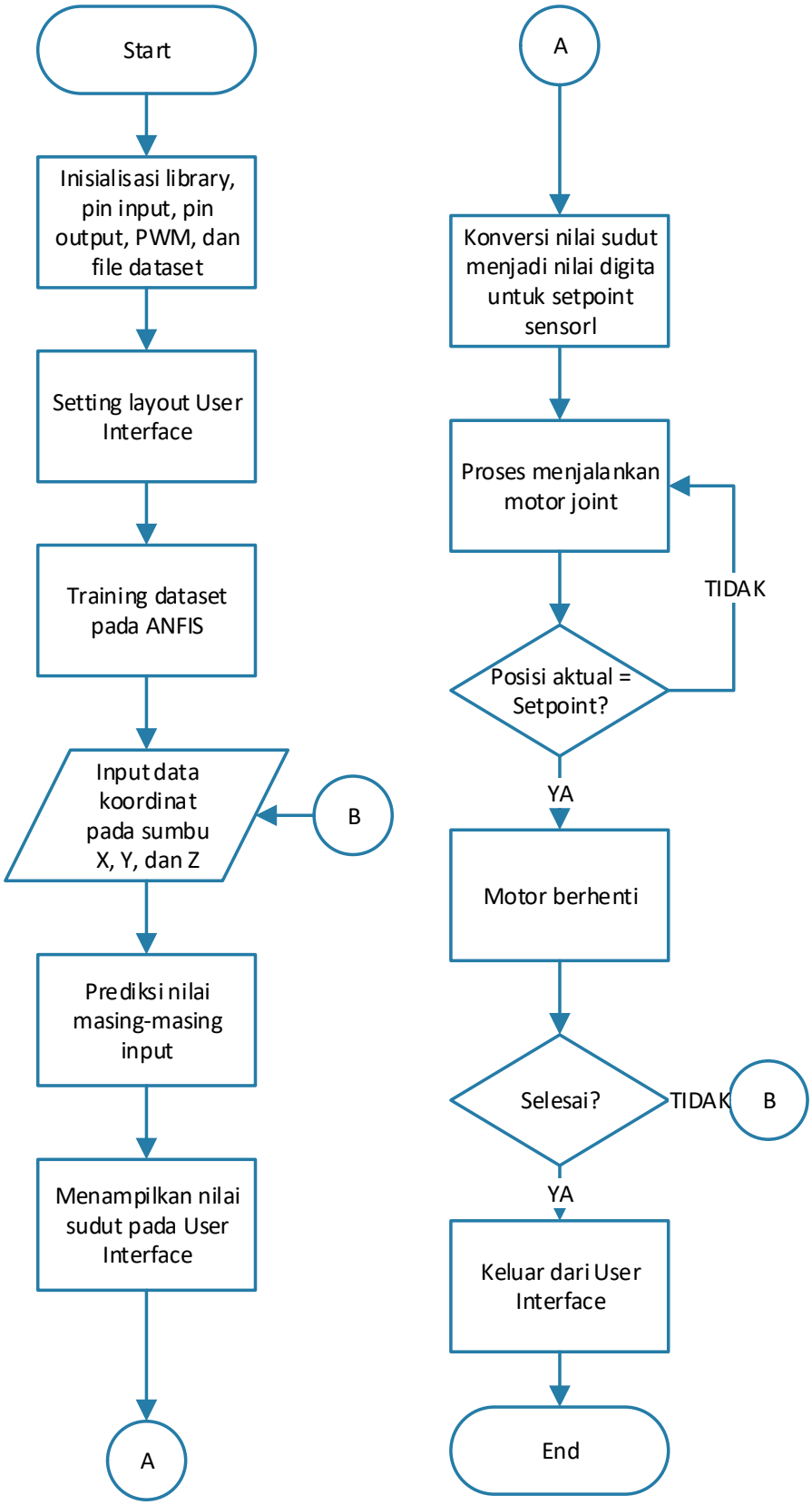
Gambar 3.6 Rancangan Struktur ANFIS

Masing-masing layer memiliki fungsi tersendiri dengan penjelasan sebagai berikut:

1. Layer 1 merupakan input koordinat posisi yang diinginkan.
2. Layer 2 merupakan *membership function fuzzy*
3. Layer 3 merupakan rule berdasarkan membership function
4. Layer 4 merupakan proses defuzzifikasi
5. Layer 5 merupakan proses normalisasi berdasarkan proses defuzzifikasi
6. Layer 6 merupakan output berupa sudut *joint*

3.4. *Flowchart*

Flowchart perancangan sistem sangat diperlukan agar memudahkan dalam proses pembuatan program serta memudahkan pemahaman mengenai sistem.

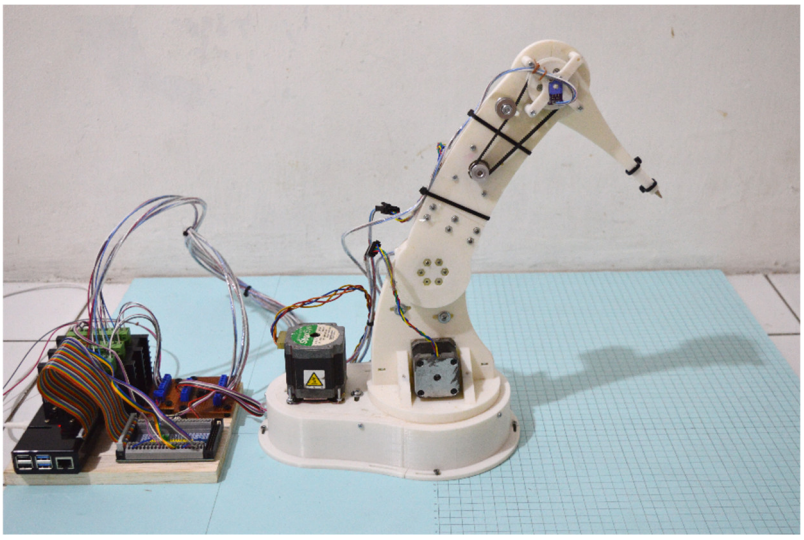


Gambar 3.7 *Flowchart* Sistem

BAB IV
HASIL DAN PEMBAHASAN

4.1. Hasil Perancangan Mekanik Robot 3-DOF

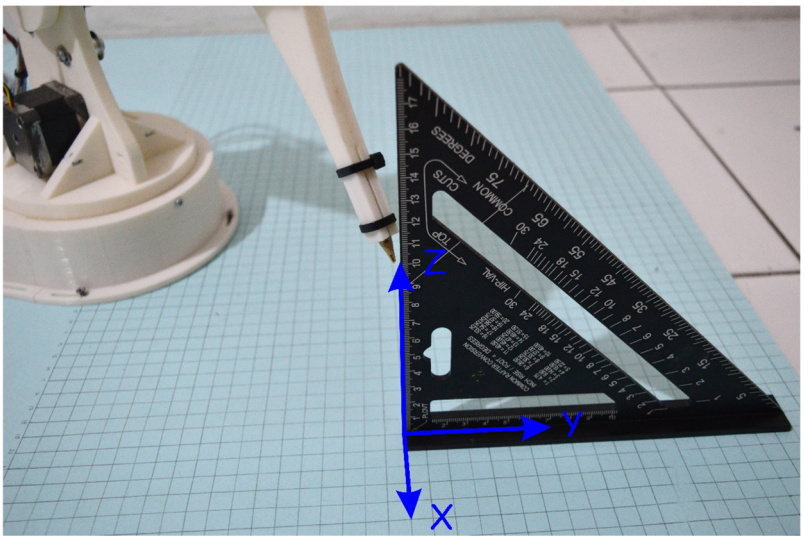
Pembuatan fisik robot menggunakan bahan plastik agar memudahkan dalam pembuatannya. Komponen penggerak menggunakan stepper motor serta *pulley* dan *belt* sebagai transmisi penggerak. Gambar 4.1 merupakan hasil pembuatan fisik robot tersebut.



Gambar 4.1 *Protoype* Robot 3-DOF

4.2. Pengujian Sistem

Pengujian dilakukan dengan cara membandingkan nilai input dan output posisi *end-effector* pada koordinat kartesian. Pengujian ini membandingkan tingkat ketelitian antara metode *inverse kinematics* dengan metode ANFIS.



Gambar 4.2 Pengukuran Posisi *End-effector*

Adapun teknik pengujian dilakukan dengan cara mengukur jarak posisi *end-effector* pada sumbu X, Y, dan Z menggunakan alat ukur yang ditunjukkan pada Gambar 4.2.

4.2.1. Pengujian Metode *Inverse Kinematics*

Untuk melakukan pengujian tingkat ketelitian metode *inverse kinematics* maka digunakan persamaan yang telah didapatkan sebelumnya.

$$\theta_1 = tan2^{-1}(p_y, p_x)$$

$$\theta_2 = tan2^{-1}(B, A) - tan2^{-1}(\sqrt{A^2 + B^2 - C^2}, C)$$

$$\theta_3 = tan2^{-1}(S_3, C_3)$$

Dengan rincian sebagai berikut:

$$A = \left(-2a_2\left(C_1 p_x + S_1 p_y\right)\right)$$

$$B = \left(2a_2(d_1 - p_z)\right)$$

$$C = -\left(p_x^2 + p_y^2 + p_z^2 + d_1^2 + a_2^2\right) + a_3^2 + 2p_z d_1$$

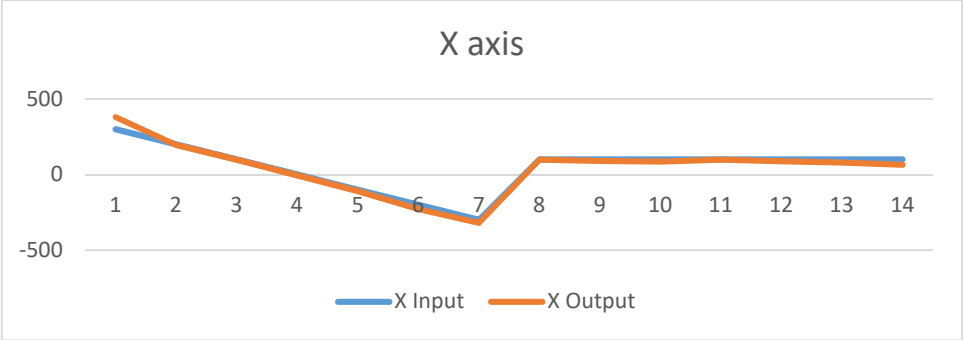
$$C_3 = \frac{p_x^2 + p_y^2 + p_z^2 + d_1^2 - a_2^2 - a_3^2 - 2p_z d_1}{2a_2 a_3}$$

$$S_3 = \sqrt{1 - (C_3)^2}$$

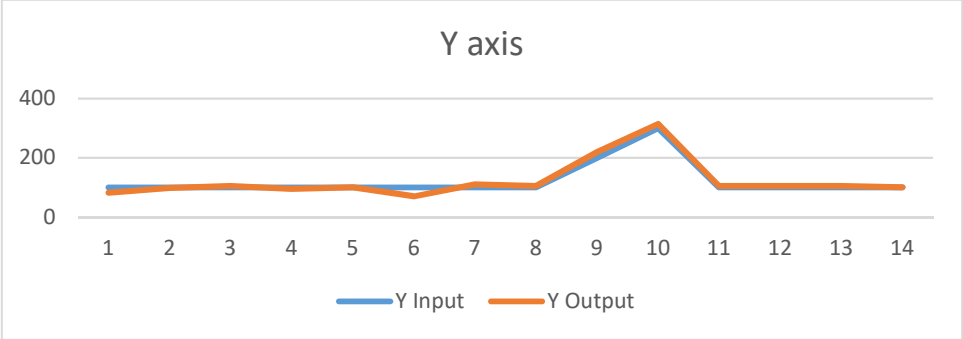
Data hasil pengujian dengan metode *inverse kinematics* disajikan dalam Tabel 4.1.

Tabel 4.1 Hasil Pengujian Metode *Inverse Kinematics* pada Robot 3-DOF

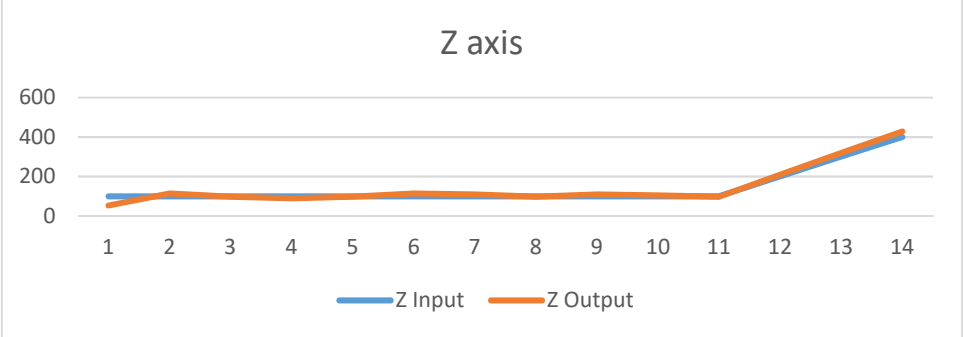
Input (mm)			Output (mm)		
X	Y	Z	X	Y	Z
300	100	100	380	82	53
200	100	100	195	98	115
100	100	100	99	105	98
0	100	100	-5	95	90
-100	100	100	-110	100	98
-200	100	100	-230	70	115
-300	100	100	-320	110	110
100	100	100	99	105	98
100	200	100	90	220	110
100	300	100	85	315	105
100	100	100	99	105	98
100	100	200	88	105	210
100	100	300	80	105	320
100	100	400	65	100	428



Gambar 4.3 Grafik Perbandingan *Input* dan *Output* pada Sumbu X dengan Metode *Inverse Kinematics*



Gambar 4.4 Grafik Perbandingan *Input* dan *Output* pada Sumbu Y dengan Metode *Inverse Kinematics*

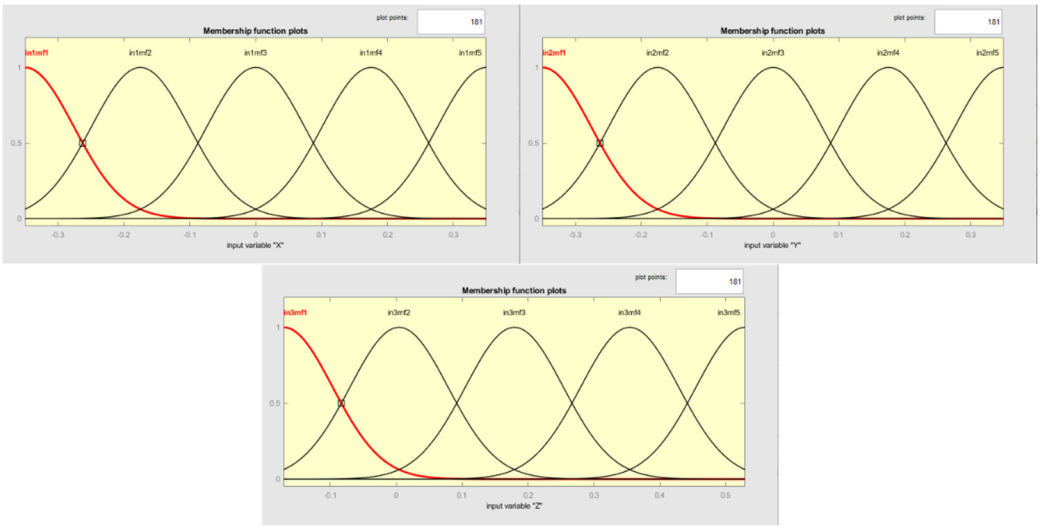


Gambar 4.5 Grafik Perbandingan *Input* dan *Output* pada Sumbu Z dengan Metode *Inverse Kinematics*

Nilai RMSE pada pengujian menggunakan metode *inverse kinematics* adalah 25.36137736.

4.2.2. Pengujian Metode ANFIS

Pengujian metode ANFIS dilakukan dengan cara melatih dataset yang telah dibuat dan diterapkan pada program. Adapun *membership function* pada *fuzzy* yang dibuat ditunjukkan pada Gambar 4.6:

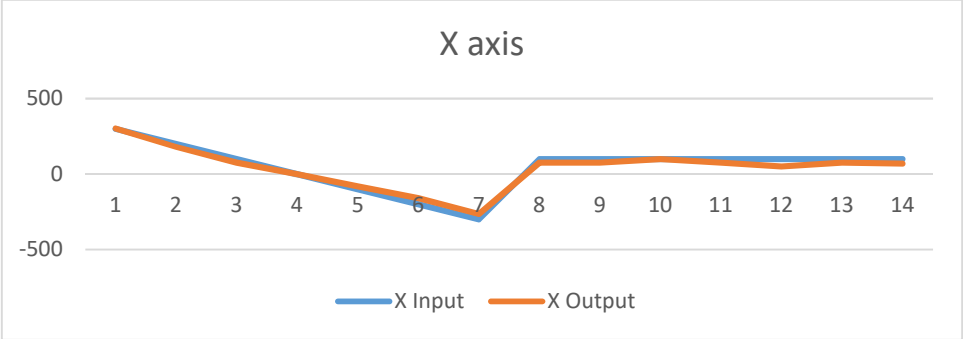


Gambar 4.6 *Membership Function Fuzzy* pada ANFIS

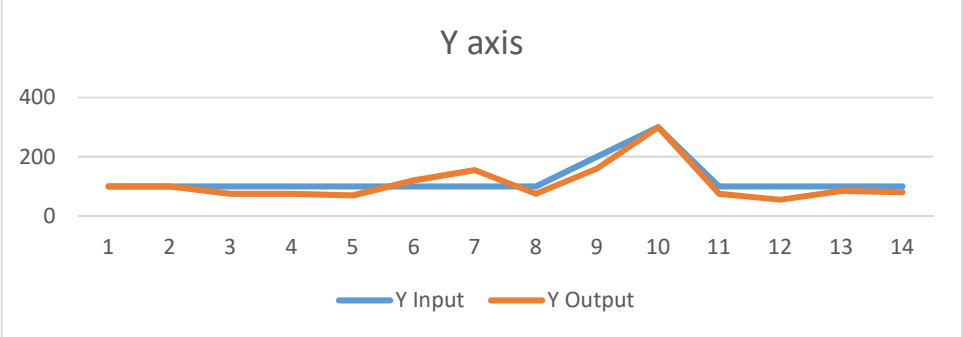
Berdasarkan hasil yang diperoleh dari proses *training* dengan *dataset* berdasarkan *forward kinematics*, maka didapatkan hasil pengujian dengan data sebagai berikut:

Tabel 4.2 Hasil Pengujian ANFIS pada Robot 3-DOF

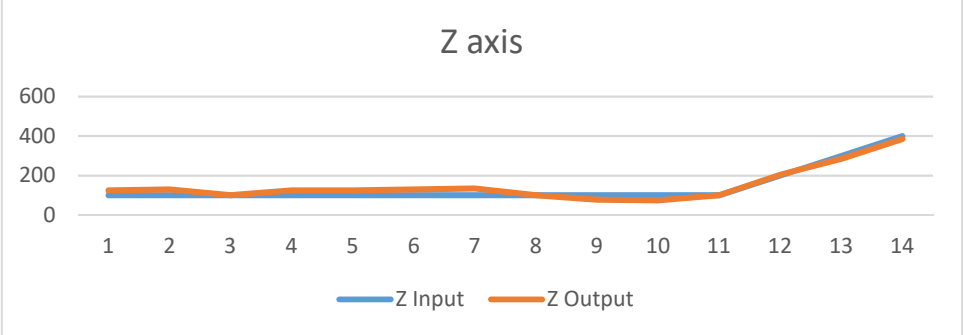
Input (mm)			Output (mm)		
X	Y	Z	X	Y	Z
300	100	100	302	100	125
200	100	100	180	100	130
100	100	100	75	75	100
0	100	100	0	75	125
-100	100	100	-80	70	125
-200	100	100	-160	120	130
-300	100	100	-265	155	135
100	100	100	75	75	100
100	200	100	75	160	78
100	300	100	100	300	75
100	100	100	75	75	100
100	100	200	50	55	205
100	100	300	75	85	285
100	100	400	70	80	385



Gambar 4.7 Grafik Perbandingan *Input dan Output* pada Sumbu X dengan Metode ANFIS



Gambar 4.8 Grafik Perbandingan *Input dan Output* pada Sumbu Y dengan Metode ANFIS



Gambar 4.9 Grafik Perbandingan *Input dan Output* pada Sumbu Z dengan Metode ANFIS

Nilai RMSE pada pengujian menggunakan metode ANFIS adalah 42.18690271.

4.3. Analisa Pengujian

Pengujian dengan metode *inverse kinematics* dan metode ANFIS dilakukan dengan cara membandingkan input posisi *end-effector* dengan output posisi *end-effector*. Nilai input berdasarkan uji perilaku robot terhadap perubahan nilai pada masing-masing sumbu. Terdapat *error* yang dihasilkan dari pengujian tersebut, baik dengan metode *inverse kinematics* maupun metode ANFIS. Beberapa faktor yang mempengaruhi *error* tersebut, yaitu:

1. Konstruksi robot

   Konstruksi dapat berpengaruh terhadap nilai *error* yang dihasilkan. Semakin *rigid* konstruksi robot maka semakin kuat pula robot bertahan terhadap momen yang diterima akibat beban (*actuator* dan *body* robot) itu sendiri.

2. Kemampuan sensor

   Kemampuan sensor pun mempengaruhi tingkat kepresisian posisi *end-effector*. Pada penelitian ini, sensor yang digunakan adalah potensiometer dengan *Analog to Digital Converter* (ADC) sebagai pengubah sinyal analog menjadi sinyal digital. ADC yang digunakan adalah MCP 3008 10 bit.

3. *Noise*

   Error yang besar pun sangat dipengaruhi oleh *noise* yang dihasilkan dari sensor. Raspberry akan membaca nilai target dari sensor yang harus dicapai setiap pergerakannya, namun bila terdapat *noise* yang didapat dari sinyal yang diberikan maka posisi *end-effector* pun akan berubah tergantung dari besarnya *noise*.

4. Jumlah dataset pada ANFIS

   Jumlah dataset pada ANFIS pun akan mempengaruhi nilai *error* yang dihasilkan. Semakin banyak jumlah *dataset* yang diberikan untuk proses pelatihan data, maka semakin kecil pula nilai *error* yang dihasilkan.

5. Jenis actuator

   Aktuator yang digunakan dalam penelitian ini menggunakan *stepping motor*. Tingkat akurasi tiap motor ini adalah 1.8°/step dan duhubungkan pada *belt* dan *pulley* dengan perbandingan 1:3 sehingga tingkat akurasinya pun semakin besar yaitu 0.8°/step. Namun, hal ini pun akan tdak berpengaruh apabila

terdapat *noise* pada sinyal yang dihasilkan *motor driver* maupun dari motor itu sendiri.

BAB V
PENUTUP

5.1. Kesimpulan

Perancangan, pembuatan, dan pengujian sistem pada *prototype* lengan robot menggunakan metode *inverse kinematics* dan metode ANFIS telah dilakukan. Dengan menganalisa hasil pengujian dengan metode tersebut, maka dapat ditarik kesimpulan dengan penjelasan sebagai berikut:

1. Metode *inverse kinematics* dilakukan dengan cara menurunkan fungsi dari *forward kinematics* sehingga didapatkan nilai tiap sudut *joint*.
2. Metode ANFIS dilakukan dengan cara membuat fungsi *forward kinematics*, *dataset*, dan *neural network*. Dataset dibuat berdasarkan rangkaian data yang diolah menggunakan fungsi dari metode *forward kinematics*. Kemudian membuat *membership function fuzzy* sebagai pengolah data *input* sebelum diterima oleh *node neural network*. Jumlah *node* yang dihasilkan pada *neural network* bergantung pada *rule* dan jumlah *membership function*-nya.
3. Tingkat *error* yang dihasilkan pada tiap metode yaitu 25.36137736 untuk metode *inverse kinematics* serta 42.18690271 untuk metode ANFIS.

5.2. Saran

Untuk perbaikan dan pengembangan selanjutnya pada penelitian ini, penulis memberikan saran dengan harapan dapat dijadikan referensi pada penelitian selanjutnya. Adapun saran tersebut adalah sebagai berikut:

1. Perbaikan pada faktor-faktor pembuat *error* sehingga nilai *error* yang dihasilkan diharapkan akan menjadi sangat kecil

2. Peningkatan jumlah *Degree of Freedom* (DOF) untuk penelitian selanjutnya dan pendekatan pada fungsi robot untuk kebutuhan industri.

DAFTAR PUSTAKA

Braunl, Thomas (2006). *Embedded Robotics: Mobile Robot Design and Application with Embedded Systems*. Jerman: Springer.

Craig, John J. (2005). *Introduction to Robotics Mechanics and Control*. Amerika: Pearson Education International.

Crenganis, Mihai, Radu Breaz, Gabriel Racz, dan Octavian Bologa. 2016. "Adaptive neuro-*fuzzy* inference system for kinematics solutions of redundant robots". *2016 6th International Conference on Computers Communications and Control (ICCCC)*, Oradea. Hlm. 271-276. DOI: 10.1109/ICCCC.2016.7496773.

Demby's, Jacket, Yixiang Gao, dan G. N. DeSouza. 2019. "A Study on Solving the Inverse Kinematics of Serial Robots using Artificial Neural Network and *Fuzzy* Neural Network". *IEEE Int. Conf. Fuzzy Syst*. Hlm. 1–6. DOI: 10.1109/FUZZ-IEEE.2019.8858872.

Gaafar, Muhammed, Ahmed Maged, M. Magdy, Nader A. Mansour, dan Ahmed El-Betar. 2018. "Design and Analysis of Experiments in ANFIS Modeling of a 3-DOF Planner Manipulator". *2018 International Japan-Africa Conference on Electronics, Communications and Computations (JAC-ECC)*. Hlm. 169-172, DOI: 10.1109/JEC-ECC.2018.8679552.

Jang, J. -. R. 1993. "ANFIS: adaptive-network-based *fuzzy* inference system". *IEEE Transactions on Systems, Man, and Cybernetics*. 23(3). Hlm. 665-685. DOI: 10.1109/21.256541.

Kalimullah, Ihtisham Qasim, Arvind Desikan dan V. Kalaichelvi. 2017. "Analysis of a proposed algorithm for point to point control of a 3 DOF robot manipulator". *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*. Hlm. 289-292. DOI: 10.1109/ICCAR.2017.7942705.

Negnevitsky, Michael (2015). *Artificial Intelligence: A Guide to Intelligence Systems*. Addison-Wesley, Harlow.

Prabantara, Saprindo Harun dan Agus Harjoko. 2013. "Analisis Kinematika Balik pada Kendali Robot Lengan Dental Light Berbasis Pengolahan Citra Digital Berdasarkan Isyarat Tangan". *Indonesian Journal of Electronics and Instrumentation Systems*. 3(2). Hlm. 207-218. DOI: 10.22146/ijeis.3895

Sen, Paresh C. (2014). *Principles of Electric Machines and Power Electronics*. Amerika: Wiley.

Srisuk, Pannawit, Adna Sento, dan Yuttana Kitjaidure. 2017. "Forward kinematic-like neural network for solving the 3D reaching inverse kinematics problems" *2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Phuket. Hlm. 214-217. DOI: 10.1109/ECTICon.2017.8096211.

Widodo (2017). *Analisa Dinamika Struktur*. Yogyakarta: Pustaka Pelajar.

LAMPIRAN A

Program ANFIS pada python

```
#import msaing-masing library
import RPi.GPIO as GPIO
import Adafruit_GPIO.SPI as SPI
import Adafruit_MCP3008
import anfis
from membership import *
import numpy as np
import time
import threading
import PySimpleGUI as sg
import math

#############################################################
"""Subroutine pergerakan motor"""
class Motor:
    def __init__(self, joint, degree):
        self.joint = joint
        self.degree = degree

    def loop_motor(self):
        y = {'joint1':0, 'joint2':1, 'joint3': 2}
        x = y[self.joint]
        a = uv[x]
        b = lv[x]
        c = round(self.degree/dvd[x]*(a-b)+b)
        print(f'Nilai dari joint {x+1} adalah {c}\n')

        while True:
            if mcp.read_adc(x) ==c:
                pwm_[x].stop()
                break
            if mcp.read_adc(x) >c:
                GPIO.output(pin_dir[x], True)#Clockwise
                pwm_[x].start(25)
                print(f'Sensor {x+1} ',mcp.read_adc(x),'\t')
            if mcp.read_adc(x) <c:
                GPIO.output(pin_dir[x], False)#Counter Clockwise
                pwm_[x].start(25)
                print(f'Sensor {x+1} ',mcp.read_adc(x),'\t')


#############################################################
"""Perintah posisi default"""
#home position = 90, 130, -140
def home_pos():
    t1 = threading.Thread(target=Motor('joint1', 90).loop_motor)
    t2 = threading.Thread(target=Motor('joint2', 130).loop_motor)
    t3 = threading.Thread(target=Motor('joint3', -140).loop_motor)

    t1.start()
    t2.start()
    t3.start()

    t1.join()
    t2.join()
    t3.join()

#################################################################
#for anfis prediction value
def prediksi(joint, x, y, z):
    ls = {'joint1':0, 'joint2':1, 'joint3':2}
    teta = anfis.predict(anf[ls[joint]], np.asarray([[x, y, z]]))[0][0]
    return teta

#################################################################
```

```
"""Setting layout GUI"""
layout = [
[sg.Text('Please input coordinates')],
[sg.Text('X axis value ='), sg.Input(key='X')],
[sg.Text('Y axis value ='), sg.Input(key='Y')],
[sg.Text('Z axis value ='), sg.Input(key='Z')],
[sg.Text('Output angles')],
[sg.Text(size=(40,1), font=('Helvetica',12), text_color='white',key='output1')],
[sg.Text(size=(40,1), font=('Helvetica',12), text_color='white',key='output2')],
[sg.Text(size=(40,1), font=('Helvetica',12), text_color='white',key='output3')],
[sg.Button('OK'), sg.Button('Quit')]
]

######################################################################
"""Setting pin dan pwm Raspberry Pi"""

#pin = (pin1, pin2, pin3)
pin_en = (18, 25, 6)
pin_dir = (23, 16, 5)
pin_pul = (24, 26, 17)
freq = 1600
dvd = (180, 180, 90) #bilangan pembagi

#value = (joint1, joint2, joint3)
uv = (762, 772, 305) #upper value
lv = (212, 232, 584) #lower value
#middle value 486,524,590

GPIO.setwarnings(False) #untuk menghilangkan warning dalam GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(pin_en, GPIO.OUT) #setting pin enable->output
GPIO.setup(pin_dir, GPIO.OUT) #setting pin direction->output
GPIO.setup(pin_pul, GPIO.OUT) #setting pin pulse->output
GPIO.output(pin_en, False) #enable ON

pwm1 = GPIO.PWM(pin_pul[0], freq)
pwm2 = GPIO.PWM(pin_pul[1], freq)
pwm3 = GPIO.PWM(pin_pul[2], freq)
pwm_ = (pwm1, pwm2, pwm3) #untuk memudahkan dalam selection

######################################################################
"""Setting SPI untuk ADC"""
SPI_PORT = 0
SPI_DEVICE = 0
mcp = Adafruit_MCP3008.MCP3008(spi = SPI.SpiDev(SPI_PORT, SPI_DEVICE))

######################################################################
#training data ANFIS
ts = np.loadtxt("data_training_2.txt", usecols=[0,1,2,3,4,5])
X1 = ts[:,0:3]
Y1 = ts[:,3]

X2 = ts[:,0:3]
Y2 = ts[:,4]

X3 = ts[:,0:3]
Y3 = ts[:,5]

mf = [
[['gaussmf',{'mean':-35.01,'sigma':7.434}]
,['gaussmf',{'mean':-17.51,'sigma':7.434}]
,['gaussmf',{'mean':0,'sigma':7.434}]
,['gaussmf',{'mean':17.51,'sigma':7.434}]
,['gaussmf',{'mean':35.01,'sigma':7.434}]],

[['gaussmf',{'mean':-350.1,'sigma':7.434}]
,['gaussmf',{'mean':-175.1,'sigma':7.434}]
,['gaussmf',{'mean':0,'sigma':7.434}]
,['gaussmf',{'mean':175.1,'sigma':7.434}]
,['gaussmf',{'mean':350.1,'sigma':7.434}]],
```

```
[['gaussmf',{'mean':-21.088,'sigma':7.434}]
,['gaussmf',{'mean':-3.76,'sigma':7.434}]
,['gaussmf',{'mean':13.57,'sigma':7.434}]
,['gaussmf',{'mean':30.9,'sigma':7.434}]
,['gaussmf',{'mean':48.23,'sigma':7.434}]]
]

mfc = membershipfunction.MemFuncs(mf)
anf1 = anfis.ANFIS(X1, Y1, mfc)
anf2 = anfis.ANFIS(X2, Y2, mfc)
anf3 = anfis.ANFIS(X3, Y3, mfc)
anf = [anf1, anf2, anf3]

print('Training data teta 1')
anf1.trainHybridJangOffLine(epochs=10)
print('Training data teta 2')
anf2.trainHybridJangOffLine(epochs=10)
print('Training data teta 3')
anf3.trainHybridJangOffLine(epochs=10)

################################################################
home_pos()
window = sg.Window('Aplikasi inverse kinematics', layout)
while True:
    event, values = window.read()
    if event == sg.WINDOW_CLOSED or event == 'Quit':
        break

    input_gui = float(values['X'])/10, float(values['Y'])/10, float(values['Z']
    if event == 'OK':
        #menampilkan hasil pada window
        window['output1'].update('Nilai teta 1= '+ str(prediksi('joint1',
input_gui)/10)))
        window['output2'].update('Nilai teta 2= '+ str(prediksi('joint2',
input_gui)/10)))
        window['output3'].update('Nilai teta 3= '+ str(prediksi('joint3',
input_gui)/10)))

        #menjalankan motor pada sudut yang telah ditentukan
        t1 = threading.Thread(target=Motor('joint1', prediksi('joint1',
input_gui)/10)).loop_motor)
        t2 = threading.Thread(target=Motor('joint2', prediksi('joint2',
input_gui)/10)).loop_motor)
        t3 = threading.Thread(target=Motor('joint3', prediksi('joint3',
input_gui)/10)).loop_motor)
        t1.start()
        t2.start()
        t3.start()
        t1.join()
        t2.join()
        t3.join()

window.close() #window akan close
```

LAMPIRAN B

Program inverse kinematics pada python

```
#import masing-masing library
import RPi.GPIO as GPIO
import Adafruit_GPIO.SPI as SPI
import Adafruit_MCP3008
import numpy as np
import time
import threading
import PySimpleGUI as sg
import math

######################################################################
"""Subroutine pergerakan motor"""
class Motor:
    def __init__(self, joint, degree):
        self.joint = joint
        self.degree = degree

    def loop_motor(self):
        y = {'joint1':0, 'joint2':1, 'joint3': 2}
        x = y[self.joint]
        a = uv[x]
        b = lv[x]
        c = round(self.degree/dvd[x]*(a-b)+b)
        print(f'Nilai dari joint {x+1} adalah {c}\n')

        while True:
            if mcp.read_adc(x) ==c:
                pwm_[x].stop()
                break
            if mcp.read_adc(x) >c:
                GPIO.output(pin_dir[x], True)#Clockwise
                pwm_[x].start(25)
                print(f'Sensor {x+1} ',mcp.read_adc(x),'\t')
            if mcp.read_adc(x) <c:
                GPIO.output(pin_dir[x], False)#Counter Clockwise
                pwm_[x].start(25)
                print(f'Sensor {x+1} ',mcp.read_adc(x),'\t')

######################################################################
"""Subroutine persamaan Inverse Kinematics"""
class Inverse():
    def __init__(self, dx, dy, dz, d1=139.2, a2=200, a3=179.2):
        self.dx = dx
        self.dy = dy
        self.dz = dz
        self.d1 = d1
        self.a2 = a2
        self.a3 = a3

    def teta1(self):
        if (self.dx == 0 and self.dy == 0):
            self.teta1 = 90
        else:
            self.teta1 = math.degrees(math.atan2(self.dy, self.dx))
        return self.teta1

    def teta2(self):
        a = 2*self.a2*(self.d1-self.dz)
        b = -2*self.a2*(math.cos(math.radians(self.teta1))*self.dx +
math.sin(math.radians(self.teta1))*self.dy)
        c = (pow(self.a3,2) + 2*self.dz*self.d1) - (pow(self.dx, 2) + pow(self.dy,
2) + pow(self.dz, 2) + pow(self.d1, 2) + pow(self.a2, 2))
        akar = (math.sqrt(pow(a, 2) + pow(b, 2) - pow(c, 2)))
        teta2 = 90-(math.degrees(math.atan2(b, a)+math.atan2(akar, c)))
        return teta2
```

```
    def teta3(self):
        temp3 = (pow(self.dx, 2) + pow(self.dy, 2) + pow(self.dz, 2) +
pow(self.d1, 2) - pow(self.a2, 2) - pow(self.a3, 2)-2*self.dz*self.d1)
        temp4 = 2*self.a2*self.a3
        cos_temp3 = temp3/temp4
        sin_temp3 = math.sqrt(1- pow(cos_temp3, 2))
        temp3 = -math.degrees(math.atan(sin_temp3/cos_temp3))
        if temp3 < 0:
            teta3 =temp3
        if temp3 > 0:
            teta3 = -180 + temp3
        #teta3 = math.degrees(math.acos(cos_temp3))
        return teta3

######################################################################
"""Perintah posisi default"""
#home position = 90, 130, -140
def home_pos():
    t1 = threading.Thread(target=Motor('joint1', 90).loop_motor)
    t2 = threading.Thread(target=Motor('joint2', 130).loop_motor)
    t3 = threading.Thread(target=Motor('joint3', -140).loop_motor)

    t1.start()
    t2.start()
    t3.start()

    t1.join()
    t2.join()
    t3.join()

######################################################################
"""Setting layout GUI"""
layout = [
[sg.Text('Please input coordinates')],
[sg.Text('X axis value ='), sg.Input(key='X')],
[sg.Text('Y axis value ='), sg.Input(key='Y')],
[sg.Text('Z axis value ='), sg.Input(key='Z')],
[sg.Text('Output angles')],
[sg.Text(size=(40,1), font=('Helvetica',12), text_color='white',key='output1')],
[sg.Text(size=(40,1), font=('Helvetica',12), text_color='white',key='output2')],
[sg.Text(size=(40,1), font=('Helvetica',12), text_color='white',key='output3')],
[sg.Button('OK'), sg.Button('Quit')]
]

######################################################################
"""Setting pin dan pwm Raspberry Pi"""

#pin = (pin1, pin2, pin3)
pin_en = (18, 25, 6)
pin_dir = (23, 16, 5)
pin_pul = (24, 26, 17)
freq = 1600
dvd = (180, 180, 90) #bilangan pembagi

#value = (joint1, joint2, joint3)
uv = (762, 772, 305) #upper value
lv = (212, 232, 584) #lower value
#middle value 486,524,590

GPIO.setwarnings(False) #untuk menghilangkan warning dalam GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(pin_en, GPIO.OUT) #setting pin enable->output
GPIO.setup(pin_dir, GPIO.OUT) #setting pin direction->output
GPIO.setup(pin_pul, GPIO.OUT) #setting pin pulse->output
GPIO.output(pin_en, False) #enable ON

pwm1 = GPIO.PWM(pin_pul[0], freq)
pwm2 = GPIO.PWM(pin_pul[1], freq)
pwm3 = GPIO.PWM(pin_pul[2], freq)
pwm_ = (pwm1, pwm2, pwm3) #untuk memudahkan dalam selection
```

```
#################################################################
"""Setting SPI untuk ADC"""
SPI_PORT = 0
SPI_DEVICE = 0
mcp = Adafruit_MCP3008.MCP3008(spi = SPI.SpiDev(SPI_PORT, SPI_DEVICE))

################################################################

home_pos()
window = sg.Window('Aplikasi inverse kinematics', layout)
while True:
    event, values = window.read()
    if event == sg.WINDOW_CLOSED or event == 'Quit':
        break

    if event == 'OK':
        result1 = Inverse(float(values['X']), float(values['Y']),
float(values['Z']))
        result2 = Inverse(float(values['X']), float(values['Y']),
float(values['Z']))
        #menampilkan hasil pada window
        window['output1'].update('Nilai teta 1= '+ str(round(result1.teta1(), 2)))
        window['output2'].update('Nilai teta 2= '+ str(round(result1.teta2(), 2)))
        window['output3'].update('Nilai teta 3= '+ str(round(result1.teta3(), 2)))

        #menjalankan motor pada sudut yang telah ditentukan
        t1 = threading.Thread(target=Motor('joint1', result2.teta1()).loop_motor)
        t2 = threading.Thread(target=Motor('joint2', result2.teta2()).loop_motor)
        t3 = threading.Thread(target=Motor('joint3', result2.teta3()).loop_motor)
        t1.start()
        t2.start()
        t3.start()
        t1.join()
        t2.join()
        t3.join()
window.close() #window akan close
```