TUGAS AKHIR – TE 141599

SISTEM PENJEJAK PIPA PADA BALON UDARA DENGAN MENGGUNAKAN KAMERA DAN KONTROL LOGIKA *FUZZY*

Dion Hayu Fandiantoro
NRP 2212 100 077

Dosen Pembimbing
Dr. Muhammad Rivai, S.T., M.T.
Rudy Dikairono, S.T., M.T.

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

FINAL PROJECT – TE 141599

PIPE TRACKING SYSTEM USING CAMERA AND FUZZY LOGIC CONTROL FOR AIR BALLOON

Dion Hayu Fandiantoro
NRP 2212 100 077

Supervisor
Dr. Muhammad Rivai, S.T., M.T.
Rudy Dikairono, S.T., M.T.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industrial Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

SISTEM PENJEJAK PIPA PADA BALON UDARA DENGAN MENGGUNAKAN KAMERA DAN KONTROL LOGIKA FUZZY

TUGAS AKHIR

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

Menyetujui :

Dosen Pembimbing I,	Dosen Pembimbing II,
Dr. Muhammad Rivai, S.T., M.T.	Rudy Dikairono, S.T., M.T.
NIP. 196904261994031003	NIP. 198103252005011002

SURABAYA
JUNI, 2016

SISTEM PENJEJAK PIPA PADA BALON UDARA DENGAN MENGGUNAKAN KAMERA DAN KONTROL LOGIKA FUZZY

Nama : **Dion Hayu Fandiantoro**
Pembimbing : **Dr. Muhammad Rivai, S.T., M.T.**
Rudy Dikairono, S.T., M.T.

ABSTRAK

Balon udara merupakan salah satu jenis *Unmaned Aerial Vehicle* (UAV) yang mampu bergerak secara otomatis, salah satu aplikasinya adalah sebagai penjejak pipa. Pada dasarnya digunakan sistem navigasi dengan bantuan *global positioning system* (GPS) dan kontrol PID untuk mengatur arah tujuan. Tetapi penggunaan GPS pada daerah yang tidak memiliki ruang terbuka bebas tidaklah memungkinkan dikarenakan pada daerah tersebut sinyal GPS menjadi lemah ataupun hilang. Sehingga dibutuhkan sebuah sistem yang dapat bekerja secara independen tanpa menggunakan GPS dalam proses penjejakan pipa. Digunakan proses pengolahan citra dengan *contour finding* dan *region of interest* yang terpadu dalam sebuah sistem yang terdiri dari Raspberry Pi dan Arduino Mega dalam memandu balon udara untuk menyusuri pipa, sehingga balon udara dapat berjalan secara otomatis menyusuri pipa. Serta digunakan kontrol logika *fuzzy* untuk menentukan kecepatan motor untuk mempertahankan keseimbangan dan untuk menyusuri pipa.

Hasil dari pengujian yang dilakukan dengan simulasi menggunakan pipa fleksibel berwarna biru pada tugas akhir ini menunjukan bahwa balon udara dapat menyusuri pipa dengan panduan pipa fleksibel. Selain menyusuri pipa fleksibel, balon udara juga dipertahankan kondisi *roll*-nya untuk selalu setimbang. Didapatkan kesalahan ukur dalam proses penjejakan pipa sebesar 4,7%, sedangkan untuk kondisi *roll* didapatkan kesalahan sebesar 0,76%.

Kata Kunci : UAV, Balon Udara , *Tracking, OpenCV, IMU*

PIPE TRACKING SYSTEM USING CAMERA AND FUZZY LOGIC CONTROL FOR AIRSHIP

Name : **Dion Hayu Fandiantoro**
Supervisor : **Dr. Muhammad Rivai, S.T., M.T.**
Rudy Dikairono, S.T., M.T.

ABSTRACT

Helium ballon is one type of Unmaned Aerial Vehicle (UAV) which capable of moving automatically, one of application of this kind of UAV is as pipe follower. Basically, global positioning system (GPS) is used for navigation system with PID control to adjust balloon's heading. However, GPS usage within area without line of sight to GPS sattelite is impossible due to weak signal reception. Therefore, a system which can work independently without GPS assist is needed to follow pipeline. Image processing is used with contour finding and region of interest which is integrated within Raspberry Pi and Arduino Mega to guide the balloon to follow pipeline, which the ballon can be run automatically.

The result of test performance with simulation using blue colored flexible pipe in this final project shows that helium ballon can fly above flexible pipe using as guideline. Appart from following flexible pipe, ballon's roll condition maintained to stable at 0°. Error reading obtained from gas pipe tracking is 4,7%, whilst for roll error obtained when testing is 0.76%.

Kata Kunci : UAV, Helium Balloon, Tracking, OpenCV, IMU

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Tugas Akhir merupakan penelitian yang dilakukan oleh mahasiswa S1 Institut Teknologi Sepuluh Nopember Surabaya. Tugas Akhir ini merupakan salah satu syarat wajib untuk menyelesaikan studi dalam program sarjana teknik.

Pada bab ini, akan dibahas mengenai hal-hal yang mendahului pelaksanaan Tugas Akhir. Hal-hal tersebut meliputi latar belakang, perumusan masalah, tujuan, batasan masalah, sistematika penulisan, dan relevansi.

Latar Belakang

Pada dasarnya, balon udara adalah sebuah alat yang mampu terbang dengan kecepatan sangat rendah maupun mengambang di udara dengan perubahan ketinggian yang kecil. Dengan perkembangan teknologi robotik bidang penerbangan, penggunaan balon udara juga semakin berkembang. Saat ini balon udara banyak dipakai sebagai media komersial [1], selain itu penggunaan balon udara juga banyak digunakan untuk memonitoring cuaca dengan menerbangkan balon pada ketinggian sekitar 18-37 km [2]. Pengembangan balon sebagai pesawat tanpa awak banyak dilakukan pada bidang transportasi kargo [3], monitoring lingkungan [4], monitoring trafik kendaraan [5], dan juga pada bidang telekomunikasi [6].

Penggunaan sistem posisi dan penjejak menjadi komponen utama dalam sistem kontrol dan sistem navigasi balon udara. Sistem yang paling banyak dipakai saat ini merupakan sistem sederhana yang menggunakan bantuan *global positioning satellite* (GPS) dengan titik lintang dan bujur yang sudah didefinisikan [7] dan sistem kontrol PID untuk mengatur ketinggian dan arah tujuan balon udara saat ini [8].

Tetapi pada realisasinya, dengan hanya menggunakan GPS untuk sistem navigasi masih kurang. Dikarenakan penyimpangan yang terjadi masih besar dengan orde meter [9]. Selain itu, GPS tidak dapat digunakan pada daerah dengan gedung-gedung tinggi atau pada tempat yang tidak memiliki ruang terbuka bebas (*line-of-sight), d*ikarenakan sinyal GPS yang tidak dapat diterima pada daerah yang disebutkan sebelumnya.

Oleh karena itu diajukan sebuah metode navigasi baru dengan penggunaan pipa sebagai media penjejak dengan menggunakan kamera. Metode ini dapat bekerja secara independen tanpa bantuan GPS dalam

navigasinya, dan dapat bekerja otomatis, serta balon dapat dikendalikan dari jarak jauh.

Permasalahan

Bedasarkan latas belakang di atas, dapat dirumuskan beberapa masalah, antara lain :

1. Bagaimana merancang sistem penjejak pipa untuk untuk balon udara?
2. Bagaimana mengontrol balon udara?
3. Bagaimana menstabilkan balon udara

Tujuan

Tujuan dari pembuatan tugas akhir ini adalah:

1. Penjejak pipa dengan menggunakan kamera
2. Balon udara dikontrol oleh mikrokontroller STM32F1 berdasarkan kondisi kamera.
3. Sensor IMU dapat menstabilkan laju balon udara.

Batasan Masalah

Batasan masalah pada tugas akhir ini adalah:

1. Menjejak pipa yang berada di atas tanah dengan menggunakan kamera.
2. Sensor IMU yang digunakan berisi akselerometer, giroskop, dan magnetometer.
3. Takeoff dan landing balon udara dilakukan secara manual.

Metodologi

Metodologi yang digunakan pada penelitian tugas akhir ini adalah sebagai berikut:

1. Studi Literatur

   Studi literatur berguna untuk mencari informasi atau data mengenai balon udara, kontroller, atau sistem secara keseluruhan. Studi literatur diperlukan sebagai landasan dalam mengerjakan Tugas Akhir agar diperoleh teori penunjang yang memadai, baik ilmu dasar, analisis, maupun metode Penelitian. Hal ini dapat dilakukan dengan melihat acuan dari jurnal, buku teks, internet, dan lain-lain. Dengan adanya studi literatur, penelitian dapat dilakukan berdasarkan teori-teori yang telah ada sebelumnya.

2. Perancangan Software

   Perangkat lunak dirancang dengan pembuatan source code melakukan penjejakan pipa, dan kontrol kendali balon. Untuk pemilihan warna pipa dilakukan oleh operator secara manual.untuk penjenjakan dapat dilakukan dengan melakukan pencarian titik tengah dari kontur pipa yang dijejak. Lalu dilakukan perhitungan agar pipa selalu berada pada tengah pipa.
3. Perancagan Hardware

   Perancangan hardware pada tahap ini meliputi kamera dan kontroller untuk menggerakan hardware berupa motor untuk pergerakan kiri dan kanan serta roll.
4. Pengujian Sistem

   Pengujian alat dilakukan untuk menentukan keandalan dari sistem yang telah dirancang. Pengujian dilakukan untuk melihat apakah software dan hardware dapat bekerja secara baik. Pengujian sistem dilakukan dengan penjejakan pipa yang telah dipilih. Apakah sistem dapat mengikuti pergerakan pipa
5. Analisa

   Analisa dilakukan terhadap hasil dari pengujian sehingga dapat ditentukan karakteristik pelacakan dan penguncian dari software dan hardware yang telah dibuat.
6. Penulisan Tugas Akhir

   Tahap penulisan laporan tugas akhir adalah tahapan terakhir dari proses pengerjaan tugas akhir ini. Laporan tugas akhir berisi seluruh hal yang berkaitan dengan tugas akhir yang telah dikerjakan yaitu meliputi pendahuluan, teori dasar, perancangan sistem, pengujian alat, dan kesimpulan dan saran.

Sistematika Penulisan

Sistematika pembahasan Tugas Akhir ini terdiri atas lima bab, yaitu pendahuluan, teori penunjang, perancangan sistem, pengujian dan analisis, serta penutup.

BAB I PENDAHULUAN

Bab ini meliputi penjelasan latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, sistematika penulisan, dan relevansi.

BAB II TEORI PENUNJANG

Bab ini menjelaskan tentang teori penunjang dan literature yang dibutuhkan dalam pengerjakan tugas akhir ini. Bagian ini memaparkan mengenai beberapa teori penunjang dan beberapa literatur yang berguna bagi pembuatan Tugas Akhir ini.

BAB III PERANCANGAN SISTEM

Bab ini menjelasakan tentang perencanaan sistem baik perangkat keras (hardware) maupun perangkat lunak (software) untuk sistem pelacakan dan penjejakan secara otomatis.

BAB IV PENGUJIAN DAN ANALISIS

Pada bab ini akan menjelaskan hasil uji coba sistem beserta analisanya.

BAB V PENUTUP

Bagian ini merupakan bagian akhir yang berisikan kesimpulan yang diperoleh dari pembuatan Tugas Akhir ini, serta saran-saran untuk pengembangan lebih lanjut.

Relevansi

Hasil yang diharapkan dari tugas akhir ini diharapkan mampu meringankan pekerjaan manusia khususnya pada pengecekan pipa gas. Pengembangan lebih lanjut dari sistem ini adalah penambahan pemilihan pipa otomatis tanpa operator, sehingga dapat secara adaptif mengubah target pipa di tengah perjalanan.

BAB II
TEORI PENUNJANG

Dalam penyusunan Tugas Akhir ini terdapat beberapa teori dasar yang menjadi acuan untuk merumuskan dan menyelesaikan masalah yang akan dibahas. Pada bagian awal diberikan tinjauan pustaka yang menggambarkan landasan teori secara umum yang akan digunakan pada tugas akhir ini. Pada bagian selanjutnya membahas tentang teori-teori pendukung.

2.1 Balon Udara

Balon Udara Blimps adalah tipe kendaraan udara yang termasuk dalam *Ligther-Than-Air* (LTA) craft. Disebut *Lighter-Than-Air* dikarenakan blimps menggunakan komponen helium yang mempunyai massa jenis lebih ringan dari udara untuk membantu mengangkat pesawat.

Balon Udara blimps menggunakan helium sebagai gaya angkat untuk terbang yang dikandung dalam envelope. Kontrol utama dari balon blimps pada dasarnya menggunakan motor yang berfungsi sebagai pendorong ke depan, elevate (mendorong keatas atau kebawah), dan rudder (untuk membelokkan kemudi). Penggunaan Balon Udara dalam UAV untuk aplikasi tertentu mempunyai beberapa kelebihan dimana tingkat keamanan, kemudahan penggunaan, dan kemampuan dalam terbang lebih baik daripada helikopter maupun quadkopter [10].

Gambar 2.1 Balon Udara (techspot.com)

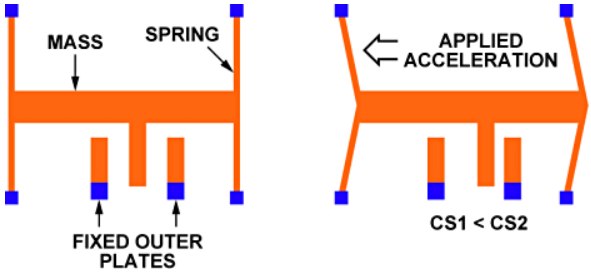
2.2 Akselerometer

Akselerometer adalah alat yang digunakan untuk mengukur percepatan, mendeteksi dan mengukur getaran (vibrasi), dan mengukur percepatan akibat gravitasi (inklinasi) [11]. Akselerometer dapat digunakan untuk mengukur getaran pada mobil, mesin, bangunan, dan instalasi pengamanan. Akselerometer juga dapat diaplikasikan pada pengukuran aktivitas gempa bumi dan peralatan-peralatan elektronik, seperti permainan 3 dimensi, mouse komputer, dan telepon. Akselerometer yang sering digunakan untuk aplikasi umum merupakan akselerometer jenis MEMS, dengan memanfaatkan sebuah massa yang ditopang oleh pegas sesuai dengan gambar 2.2. Untuk aplikasi yang lebih lanjut, sensor ini banyak digunakan untuk keperluan navigasi. Percepatan merupakan suatu keadaan berubahnya kecepatan terhadap waktu. Bertambahnya suatu kecepatan dalam suatu rentang waktu disebut percepatan (akselerasi). Namun jika kecepatan semakin berkurang daripada kecepatan sebelumnya, disebut perlambatan (deakselerasi). Percepatan juga bergantung pada arah/orientasi karena merupakan penurunan kecepatan yang merupakan besaran vektor. Berubahnya arah pergerakan suatu benda akan menimbulkan percepatan pula. Untuk memperoleh data jarak dari sensor akselerometer, diperlukan proses integral ganda terhadap keluaran sensor [12].

2.3 Giroskop

Menurut Kamus Besar Bahasa Indonesia (KBBI), giroskop adalah alat berupa cakram yang sumbunya berputar antara dua penopang dan tetap dalam posisinya apabila tidak ada pengaruh kekuatan luar; alat pengendali roket.

Giroskop digunakan untuk mengukur orientasi berdasarkan prinsip momentum sudut. Giroskop konvensional adalah giroskop mekanikal, yang terdiri dari sebuah piringan yang berputar di sumbu putar. Sumbu putar ini terpasang pada suatu rangka yang disebut gimbal. Jumlah gimbal menentukan jumlah aksis giroskop. Giroskop yang memiliki satu gimbal hanya dapat berputar dengan satu poros, giroskop yang memiliki dua gimbal dapat berputar dengan dua poros, dan giroskop yang memiliki tiga gimbal dapat berputar dengan tiga poros [13].

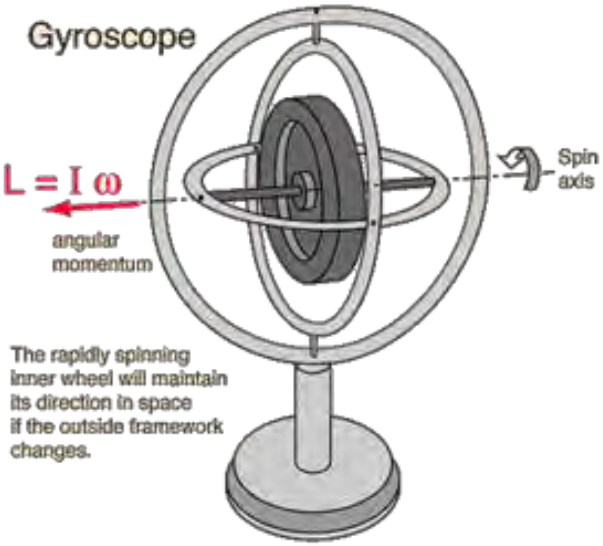


Gambar 2.2 Gambaran Dasar Cara Kerja Akselerometer (analog.com)

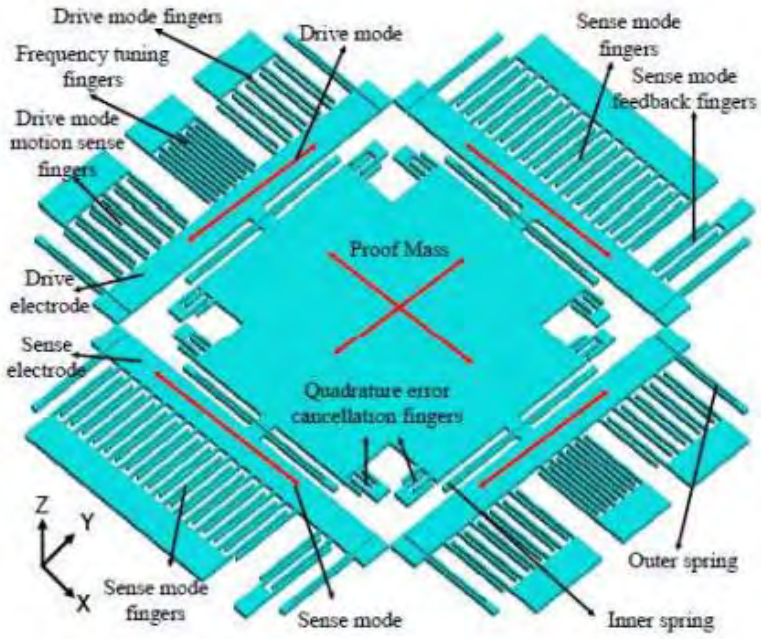
Prinsip rotor giro adalah kekakuan dalam ruang atau inersia giroskopik. Hukum pertama Newton menyatakan bahwa gaya total suatu benda adalah nol, maka gerak benda tidak akan terjadi.

$$\sum F = 0 \text{ (dalam kesetimbangan)} \quad (2.1)$$

Rotor berputar dalam giroskop mempertahankan sikap konstan dalam ruang selama tidak ada gaya luar yang mengubah gerakannya. Stabilitas ini meningkat jika rotor memiliki massa yang besar dan berputar dengan cepat [14].



Gambar 2.3 Giroskop (hyperphysic.phy-astr.gsu.edu)



Gambar 2.4 Struktur Dasar Giroskop MEMS (analog.com)

Giroskop MEMS (Micro Electro Mechanical System) adalah sebuah giroskop yang terbuat dari semikonduktor. Piringan massa pada giroskop MEMS dibuat bergetar pada tempatnya dengan 2 drive mode fingers yang simetris. Ketika terkena rotasi eksternal, percepatan Coriolis menyebabkan piringan massa bergerak dan menggeser sense mode finger dan menciptakan perubaha kapasitansi differensial. Sebuah rangkaian interface dengan noise yang rendah merasakan perubaha differensial dari kapasintansi dan merubahnya menjadi sinyal tegangan.

2.4 Algoritma Magdwick AHRS

Madgwick AHRS adalah metode untuk proses orientasi dari sensor IMU dengan mengunakan alogaritma gradient decent untuk mengestimasi nilai orientasi dari ketiga sumbu yang ada pada giroskop,akselerometer dan magnetometer. Dimana diketahui bahwa giroskop pada IMU mempunyai nilai drift. Drift pada giroskop adalah pergeseran nilai derajat

dari sumbu awal. Pada metode ini sudut-sudut euler dirubah menjadi Kuaternion [15].

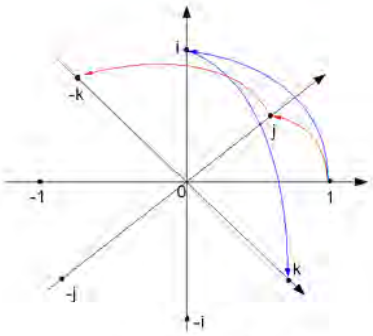
2.4.1 Kuaternion

Dalam matematika kuaternion merupakan perluasan dari bilangan-bilangan kompleks yang tidak komutatif, dan diterapkan dalam mekanika tiga dimensi. Kuaternion ditemukan oleh ahli matematika dan astronomi inggris, William Rowan Hamilton, yang memperpanjang aritmatika kompleks nomor ke kuaternion [16].

Persamaan elemen kuaternion *i*, *j*, dan *k* adalah:

$$i^2 = j^2 = k^2 = ijk = -1 \quad (2.2)$$

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



Gambar 2.5 Representasi Kuaternion (aosrobot.com)

Tabel 2.1 Aturan Kuaternion

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

2.4.2 Konsep Magdwick AHRS

Konsep dasar dari *Magwick AHRS* adalah mencari antara relatife *frame* atau yang biasa disebut perpindahan *frame*. Contoh dapat dilihat pada gambar di atas *frame* A relatife terhadap *frame* B. Dimana nilai perpindahan *frame* A ke *frame* B dapat direpresentasikan dalam bentuk kuaternion seperti persamaan 2.3.

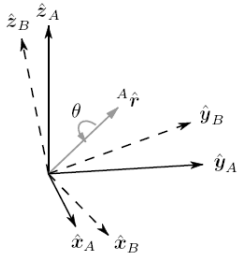
$$ {}^{A}_{B}\hat{q} = [q0\ q1\ q2\ q3] = \left[\cos\frac{\theta}{2} - rx\sin\frac{\theta}{2} - ry\sin\frac{\theta}{2} - rz\sin\frac{\theta}{2}\right] \quad (2.3) $$

$$ {}^{A}_{B}\hat{q} * = {}^{B}_{A}\hat{q} = [q0 - q1 - q2 - q3] \quad (2.4) $$

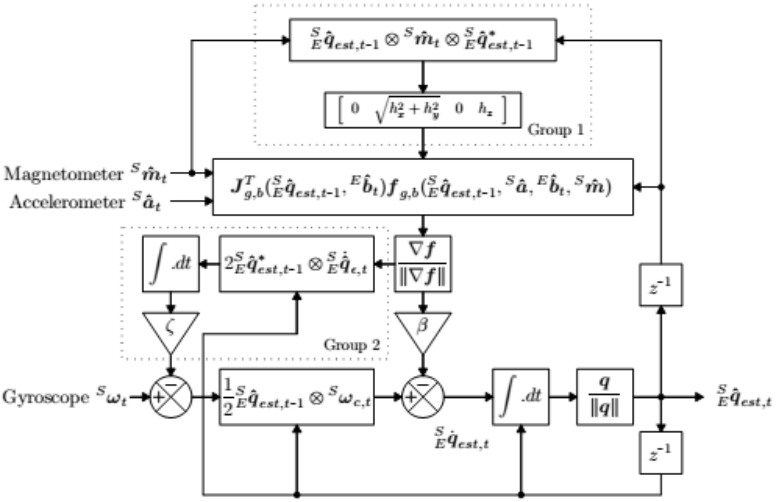
Bahwa untuk mencari nilai matrik yang baru, kedua matrik tersebut dikalikan dengan aturan *cross product* dimana dengan mengunakan aturan Hamilton. Seperti persamaan 2.5.

$$ a \otimes b = [a0\ a1\ a2\ a3] \otimes [b0\ b1\ b2\ b3] $$

$$ = \begin{bmatrix} a_0b_0 & -a_1b_1 & -a_2b_2 & -a_3b_3 \\ a_0b_1 & +a_1b_0 & +a_2b_3 & -a_3b_2 \\ a_0b_2 & -a_1b_3 & +a_2b_0 & +a_3b_1 \\ a_0b_3 & +a_1b_2 & -a_2b_1 & +a_3b_0 \end{bmatrix}^T \quad (2.5) $$



Gambar 2.6 Frame Kuaternion (intechopen.com)



Gambar 2.7 Diagram Blok Algoritma AHRS Madgwick [15]

Pada gambar 2.6 dijelaskan alur dari metode madgwick. Untuk mengetahui representasi pergeserah *frame* berdasarkan sumbu garis normal gravitasi bumi menggunakan persamaan *gradient decent*. Dimana alogaritma *gradient decent* mempunyai fungsi seperti persamaan (2.6). Dengan demikian apabila dicross product ketiga quaternion tersebut akan menghasilkan matrik baru (2.10).

$$f(\hat{q}, E_{\hat{d}}, S_{\hat{s}}) = \hat{q} * \otimes E_{\hat{d}} \otimes \hat{q} - S_{\hat{s}} \quad (2.6)$$

$$\hat{q} = [q1 - q2 - q3 - q4] \quad (2.7)$$

$$E_{\hat{g}} = [0 \;\; 0 \;\; 0 \;\; 1] \quad (2.8)$$

$$S_{\hat{a}} = [0axayaz] \quad (2.9)$$

$$\mathrm{f_g}(\hat{q}, S_{\hat{a}}) = \begin{bmatrix} 2(q2q4 - q1q3) - ax \\ 2(q1q2 + q3q4) - ay \\ 2(\frac{1}{2} - {q_2}^2 - {q_3}^2) - az \end{bmatrix} \quad (2.10)$$

Setelah proses untuk mencari nilai f , kemudian dicari hasil dari matrik *jacobi* sesuai degan persamaan(2.11).

$$\mathrm{J_g}(\hat{q}) = \begin{bmatrix} -2q3 & 2q4 & -2q1 & 2q2 \\ 2q2 & 2q1 & 2q4 & 2q3 \\ 0 & -4q2 & -4q3 & 0 \end{bmatrix} \quad (2.11)$$

Kemudian setelah kedua matrik f dan J telah didapatkan hasil.Untuk mencari nilai dari matrik *gradien decent* adalah sesuai dengan persamaan (2.12).

$$\nabla f = j^T \times f \tag{2.12}$$

Untuk mencari nilai estimasi dari kuaternion, kuaternion pada giroskop dikurangkan dengan hasil kuaternion pada proses alogaritma *gradien decent* yang sudah dibagi dengan nilai normalisasi dan dikalikan dengan nilai pembobotan(beta), sesuai dengan persamaan (2.13).

$$\dot{q}est, t = \dot{q}\omega, t - \beta \frac{\nabla f}{\|\nabla f\|} \tag{2.13}$$

Untuk mencari nilai β atau pembobotan dapat menggunakan persamaan(2.14).

$$\beta = \left\| \frac{1}{2} \hat{q} \otimes [0 \ \bar{\omega}_{max} \ \bar{\omega}_{max} \ \bar{\omega}_{max}] \right\| \sqrt{\frac{3}{4}} \bar{\omega}_{max} \tag{2.14}$$

2.5 *Computer Vision*

Computer Vision adalah pencitraan komputer dimana aplikasi tidak melibatkan manusia dalam sebagai salah satu cabang ilmu pengetahuan yang mempelajari bagaimana komputer dapat mengenali obyek yang diamati atau diobservasi. Cabang ilmu ini bersama kecerdasan buatan (Artificial Intelligence) akan mampu menghasilkan sistem kecerdasan visual (Visual Intelligence System) [17].

2.5.1 *Open Source Computer Vision*

Open Source Computer Vision (OpenCV) adalah sebuah API (Application Programming Interface) library yang sudah sangat familiar pada pengolahan citra computer vision. Computer vision itu sendiri adalah salah satu cabang dari bidang ilmu pengolahan citra (Image Processing) yang memungkinkan komputer dapat melihat seperti manusia. Dengan computer vision tersebut komputer dapat mengambil keputusan, melakukan aksi, dan mengenali terhadap suatu objek. Beberapa pengimplementasian dari computer vision adalah face recognition, face detection, face/object tracking, road tracking, dll.

OpenCV adalah library open source untuk computer vision untuk C/C++ , Python, Java dan Matlab/Octave. OpenCV didesain untuk aplikasi real-time, memiliki fungsi-fungsi akuisisi yang baik untuk image/video. OpenCV juga menyediakan interface ke Integrated Performance Primitives (IPP) Intel sehingga jika anda bisa mengoptimasi aplikasi computer vision anda jika menggunakan prosesor Intel [18]. Fitur yang dimiliki OpenCV antara lain :

a. Manipulasi data citra (alocation, copying, setting, convert).
b. Citra dan video I/O (file dan kamera based input, image/video file output).
c. Manipulasi Matriks dan Vektor beserta rutin-rutin aljabar linear (products, solvers, eigenvalues, SVD).
d. Data struktur dinamis (lists, queues, sets, trees, graphs).
e. Pemroses citra fundamental (filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids).
f. Analisis struktur (connected components, contour processing, distance Transform, various moments, template matching, Hough Transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation).
g. Kalibrasi kamera (calibration patterns, estimasi fundamental matrix, estimasi homography, stereo correspondence).
h. Analisis gerakan (optical flow, segmentation, tracking).
i. Pengenalan obyek (eigen-methods, HMM).
j. Graphical User Interface (display image/video, penanganan keyboard dan mouse handling, scroll-bars).

2.5.2 Warna

Dengan menggunakan 3 buah reseptor manusia dapat membedakan banyak warna. Warna tricromatic RGB dalam sistem grafis umumnya menggunakan 3 byte, atau sekitar 16 juta kode warna. Dikatakan kode warna dan bukan warna karena manusia tidak dapat membedakan warna sebanyak itu. Mesin dapat membedakan antara berbagai kode warna, namun perbedaan tersebut belum tentu dapat menunjukan perbedaan yang dapat ditangkap oleh mata manusia. Tiap 3 byte pada pixel RGB mencangkup 1 byte pixel untuk tiap warna Red (merah), Green (hijau), Blue (biru) [19].

Gambar direpresentasikan sebagai matrik dengan elemen integer atau pixel. Biasanya pixel tidak disimpan dalam bentuk matrik sederhana, namun menggunakan data yang lebih rumit. Dan terkadang memudahkan secara matematis jika kita beranggapan gambar berwarna sebagai matrik dari vektor tiga dimensi [20].

$$Y = 0.299\ R + 0.587\ G + 0.114\ B \qquad (2.15)$$

Bila dibandingkan dengan gambar biasa, gambar grayscale memilki akurasi yang kurang, hal ini terutama dikarenakan warna dapat membantu kita dalam membedakan objek dengan lebih baik. Namun untuk beberapa aplikasi penggunaan warna akan meningkatkan pengeluaran, sehingga

tidak diusulkan. Hal ini dikarenakan, tidak hanya karena kamera berwarna yang harganya lebih mahal, juga disebabkan karena dengan menggunakan warna maka diperlukan tiga buah digitizer, dan tiga buah frame untuk memasukkan gambar tersebut. Dan karena dengan menggunakan gambar berwarna maka diperlukan pemrosesan yang lebih banyak dalam menginterpretasikan gambar berwarna dengan sepenuhnya [21].

Sedangkan pada grayscale, tiap warna direpresentasikan oleh 1 bit pixel, dimana tiap bit dari pixel yang merepresentasikan gambar grayscale adalah suatu nilai yang menunjukkan nilai intensitas dari gambar yang berada pada posisi tersebut. Sehingga dapat diproses sebagai entitas tunggal pada komputer. Tidak seperti gambar RGB, yang merupakan suatu set yang berisi tiga bit data, maka diperlukan tiga buah angka untuk merepresentasikan informasi yang ada di dalamnya. Sehingga pada gambar grayscale diperlukan proses komputasi yang lebih sedikit di bila bandingkan dengan gambar berwarna.

2.5.3 Morfologi

Nama morfologi secara matematis didapatkan dari pembelajaran tentang bentuk. Pendekatan ini menggali fakta bahwa di dalam banyak aplikasi bahasa mesin adalah wajar, dan mudah untuk berfikir dalam bentuk saat merancang algoritma. Pendekatan morfologikal memfasilitasi pemikiran berbasiskan bentuk atau ikonis. Dalam pendekatan morfologi, unit yang paling dasar dari informasi bergambar adalah gambar biner.

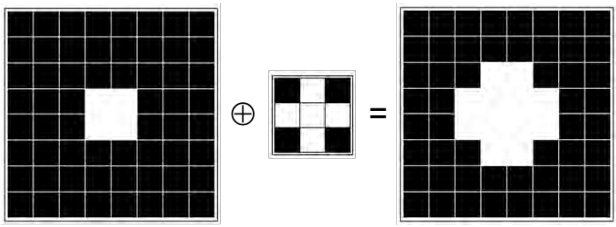
Morfologi matematis mendeskripsikan suatu aplikasi operator aljabar, yang dapat digunakan untuk mengambil suatu daerah, bentuk dan batas pada gambar.

2.5.3.1 Dilatasi

Dilatasi merupakan proses menambahkan atau mempertebal objek pada gambar biner. Hal ini dilakukan dengan memberikan tambahan pixel pada batasan suatu objek pada suatu gambar. Jumlah pixel yang ditambahkan tergantung dari ukuran dan bentuk dari stucturing element yang digunakan pada pengolahan gambar. Dimana stucturing element, merupakan sebuah matrik yang hanya berisi 1 dan 0, yang dapat memiliki berbagai bentuk dan ukuran.

Secara matematis dilatasi A oleh B, dinotasikan A⊕B, didefinisikan sebagai persamaan 2.15.

$$A \oplus B = \{Z \mid [\![(B)]\!]\_X \cap A\hat{}C \neq \varphi\} \quad (2.16)$$



Gambar 2.8 Ilustrasi Morfologi Dilatasi (aishack.in)

(a) merupakan gambar awal, dengan objek bernilai 1; (b) structuring element; (c) hasil akhir proses dilatasi.

Dimana φ merupakan set yang kosong dan B adalah stucturing element-nya. Atau dapat juga dikatakan sebagai dilatasi A oleh B merupakan suatu set yang didalamnya terdiri dari semua stucturing element dari lokasi awalnya, dimana B yang telah direfleksikan, dan ditranslasikan akan tumpang tindih setidaknya dengan sebagian dari gambar A.

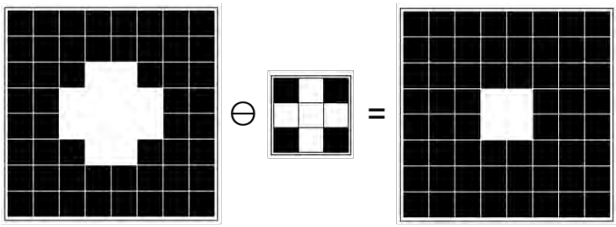
Secara algoritma operasi ini dapat didefinisikan sebagai : Anggaplah bahwa B merupakan sebuah mask. Kemudian taruhlah titik referensi dari stucturing element pada pixel yang memiliki nilai 1 pada gambar A. dan kemudian lakukan fungsi OR pada A dan B. Sehingga bila terdapat nilai yang berbeda pada pixel A dan B yang memiliki lokasi yang sama, maka nilai dari keluarannya adalah 1. Namun bila bilai pada pixel A dan B sama maka nilai keluarannya akan sama dengan nilai terebut. Proses ini kemudian akan diulang terus menerus, hingga nilai titik referensi B telah berada di semua titik A yang memiliki nilai 1. Untuk lebih jelasnya dapat dilihat pada gambar 2.7 .

2.5.3.2 Erosi

Erosi merupakan proses untuk mengecilkan ataupun menipiskan objek pada gambar. Sama seperti dilatasi, proses ini juga dipengaruhi oleh structuring element yang digunakan pada pengolahan gambar.

Definisi matematis erosi mirip dengan dilatasi. Dimana erosi A oleh B, dinotasikan $A \ominus B$, didefinisikan sebagai persamaan 2.16.

$$A \ominus B = \{Z \mid [\![(B)]\!]\_X \cap A\hat{}C \neq \varphi\} \qquad (2.17)$$



Gambar 2.9 Ilustrasi Morfologi Erosi (aishack.in)

(a) merupakan gambar awal, dengan objek bernilai 1; (b) structuring element; (c) hasil akhir proses erosi.

Secara algoritma kita dapat mendefinisikan erosi sebagai : Pertama-tama tiap nilai yang ada pada keluaran A⊖B akan dijadikan nilai 0 (zero). B kemudian akan ditempatkan pada tiap lokasi A yang memiliki nilai pixel 0 (dengan mengacu pada titik referensi B). Bila pada A terdapat B (pada proses A AND B tidak sama dengan zero) maka pada keluaran akan digunakan nilai B. Hasil keluaran akhir adalah setiap elemen dari A yang tidak terkena elemen B pada proses diatas. Untuk lebih jelasnya dapat dilihat pada gambar 2.8.

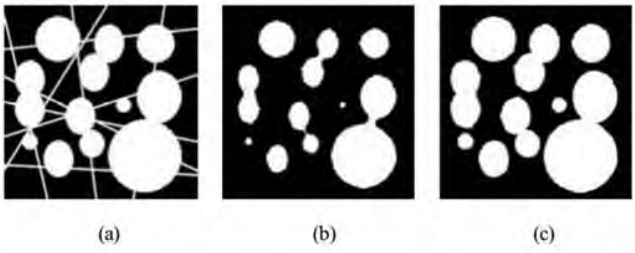
2.5.3.3 Opening

Opening merupakan suatu proses dimana suatu gambar yang terlebih dahulu mengalami proses erosi, disusul dengan proses dilatasi dengan menggunakan struktur elemen yang sama. Hal ini bertujuan untuk memulihkan sebesar mungkin data yang telah hilang karena erosi pada suatu gambar. Selain itu erosi juga dapat digunakan untuk menghilangkan objek yang terlalu kecil, memisahkan antar objek yang jaraknya berdekatan.

Opening A oleh B, yang didenotasikan dengan A◦B, didefinisikan sebagai persamaan 2.17.

$$A \circ B = (A \ominus B) \oplus B \tag{2.18}$$

Sehingga A◦B, merupakan gabungan dari semua translasi B yang termasuk dalam A. Opening membuang semua bagian dari objek yang tidak dapat menampung structuring element-nya. Sehingga memperhalus bagian tepi objek tanpa merusak bentuk aslinya, dan mengilangkan objek yang terlalu kecil atau tipis.



Gambar 2.10 Ilustrasi Morfologi *Opening* (aishack.in)

(a) gambar asli; (b) gambar asli yang telah mengalami erosi; (c) gambar (b) yang telah didilatasi dengan structuring element yang sama.

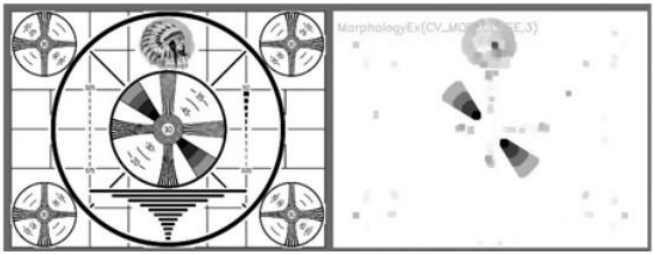
2.5.3.4 Closing

Closing merupakan kebalikan dari *opening*, dimana suatu gambar mengalami proses dilatasi terlebih dahulu sebelum kemudian dierosi dengan struktur elemen yang sama. Operasi ini biasanya digunakan untuk menghilangkan *noise* yang ada pada gambar.

Operasi *closing* A oleh B, yang didenotasi sebagai A·B, didefinisikan sebagai persamaan 2.18.

$$A \cdot B = (A \oplus B) \ominus B \tag{2.19}$$

Closing A·B merupakan komplemen dari semua gabungan dari semua translasi *B* yang tidak tumpang tindih dengan *A*. Hal ini memungkinkan operasi ini untuk menutup lubang kecil atau garis tipis pada suatu objek, dan menghubungkan 2 objek yang letaknya berdekatan, serta menghaluskan batas tiap objek tanpa merusak bentuk aslinya.



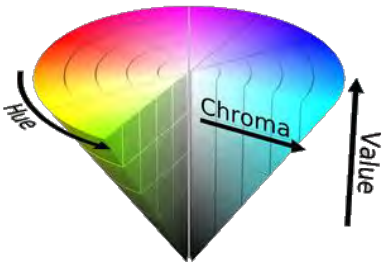
Gambar 2.11 Ilustrasi Morfologi *Closing* (aishack.in)

2.5.4 *Color Filtering Hue Saturation Value*

Model *Hue Saturation Value* (HSV) menunjukkan ruang warna dalam bentuk tiga komponen utama, yaitu *hue*, *saturation* dan *value* (atau disebut juga *brightness*). *Hue* adalah sudut dari 0 sampai 360 derajat. Biasanya 0 adalah merah, 60 derajat adalah kuning, 120 derajat adalah hijau, 180 derajat adalah *cyan*, 240 derajat adalah biru dan 300 derajat adalah magenta.

Hue menunjukkan jenis warna (seperti merah, biru atau kuning) atau corak warna, yaitu tempat warna tersebut ditemukan dalam spektrum warna. Merah, kuning dan ungu adalah kata-kata yang menunjukkan *hue*. Saturasi suatu warna adalah ukuran seberapa besar kemurnian dari warna tersebut. Sebagai contoh, suatu warna yang semuanya merah tanpa putih adalah saturasi penuh. Jika ditambahkan putih ke merah, hasilnya lebih berwarna-warni dan warna bergeser dari merah ke merah muda. *Hue* masih tetap merah tetapi nilai saturasinya berkurang. Saturasi biasanya bernilai 0 sampai 1 (atau 0% sampai 100%) dan menunjukkan nilai keabu-abuan warna dimana 0 menunjukkan abu-abu dan 1 menunjukkan warna primer murni. Komponen ketiga dari HSV adalah *value* atau disebut juga intensitas, yaitu ukuran seberapa besar kecerahan suatu warna atau seberapa besar cahaya datang dari suatu warna. Nilai *value* dari 0% sampai 100%. Pemetaan ruang warna HSV dalam bentuk *Hexcone* dapat dilihat pada gambar 2.11.

Suatu warna dengan nilai *value* 100% akan tampak sangat cerah dan suatu warna dengan nilai *value* 0 akan tampak sangat gelap. Sebagai contoh, jika *hue* adalah merah dan *value* bernilai tinggi maka warna akan terlihat cerah tetapi ketika nilai *value* bernilai rendah maka warna tersebut akan terlihat gelap.



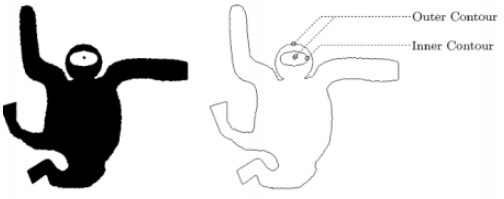
Gambar 2.12 Ruang Warna HSV (wikipedia.org)

Dengan sistem koordinat HSV, beberapa pengamatan dapat dilakukan sehubungan dengan daerah warna kubus RGB. Yang pertama adalah vertek-vertek *cyan*, magenta dan kuning dari kubus yang menunjukkan warna yang lebih cerah dibanding dengan warna merah, hijau dan biru karena warna merah hijau dan biru diproyeksikan lebih rendah ke sumbu netral. Dengan cara yang sama, semua warna dalam piramid yang ditunjukkan vertek C,Y,M dan W berhubungan ke warna-warna lebih terang dan piramid yang ditunjukkan oleh titik pusat dan vertek R,G dan B berhubungan ke warna-warna yang lebih gelap. Warna dekat sumbu netral dalam kubus akan mempunyai banyak warna karena saturasinya kurang dan warna yang lebih jauh dari sumbu ini akan tampak lebih hidup.

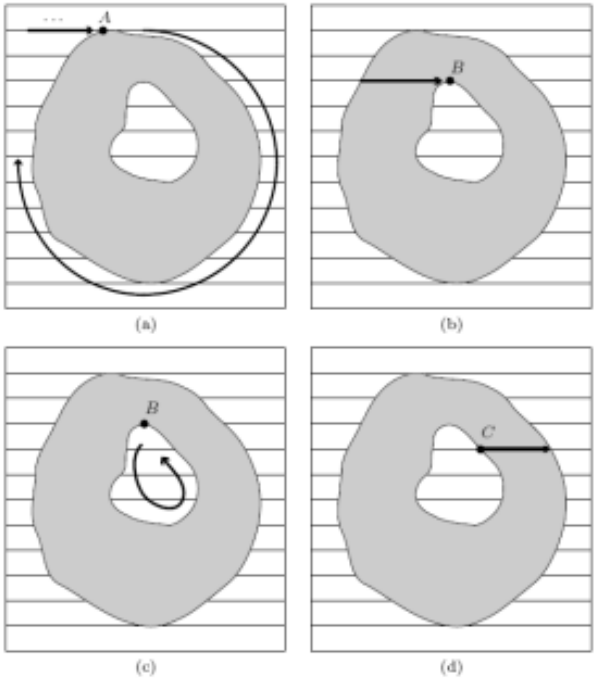
2.5.5 Contour Finding

Salah satu masalah yang paling umum pada analisis grafis dan gambar adalah menemukan interior dari sebuah *area* dimana kontur diberikan [21]. Masalah ini dapat diselesaikan dalam beberapa cara, dimana bisa dibagi menjadi dua kelas besar. Yang pertama, hasil mempunyai deskripsi dari kontur sebagai *polygon* dan menentukan bagian mana dari bidang yang terletak pada interoir dengan mempertimbangkan efek persamaan garis. Teknik tersebut biasanya dikatakan berbasis *polygon.* Metode dari kelas kedua, memetakan kontur kedalam bidang yang berlainan dan kemudian menentukan lokasi dari interior dengan mengamati nilai dari *pixel*.

uatu bagian gambar yang berhubungan hanya akan memiliki satu kontur luar, namun dapat memiliki beberapa kontur dalam akibat dari adanya “lubang”. Sehingga pada lubang tersebut masih dapat memiliki beberapa daerah lain didalamnya. Selain itu ada pula daerah yang terhubung dengan satu pixel yang sama. Sehingga dapat terjadi kesalahan dalam penentuan titik akhir suatu kontur.



Gambar 2.13 Deteksi Kontur (aishack.in)



Gambar 2.14 Proses Kerja Contour Finding (aishack.in)

Hal ini dapat dihindari dengan melakukan penggambungan antar konsep pemberian label daerah secara sekuensial, dengan konsep penentuan kontur kedalam sebuah algoritma. Hal ini dilakukan dengan pemberian label pada daerah gambar dan pada saat yang bersamaan melakukan penentuan kontur luar dan dalam.

Gambar 2.14 , memperlihatkan cara pengerjaan dari proses *contour finding* menggunakan metode ini, dimana :

(a) Ditemukan titik awal, dan penelusuran garis yang berhubungan.
(b) Ditemukannya titik awal kontur dalam.
(c) Penelusuran kontur awal.
(d) Penentuan bagian interor.

2.6 Logika Fuzzy

Konsep tentang logika fuzzy diperkenalkan oleh Prof. Lotfi Astor Zadeh pada 1962. Logika fuzzy adalah metodologi sistem kontrol pemecahan masalah, yang cocok untuk diimplementasikan pada sistem, mulai dari sistem yang sederhana, sistem kecil, embedded system, jaringan PC, multi- channel atau workstation berbasis akuisisi data, dan sistem kontrol. Metodologi ini dapat diterapkan pada perangkat keras, perangkat lunak, atau kombinasi keduanya. Dalam logika klasik dinyatakan bahwa segala sesuatu bersifat biner, yang artinya adalah hanya mempunyai dua kemungkinan, "Ya atau Tidak". Oleh karena itu, semua ini dapat mempunyai nilai keanggotaan 0 atau 1. Akan tetapi, dalam logika fuzzy kemungkinan nilai keanggotaan berada diantara 0 dan 1. Artinya,bisa saja suatu keadaan mempunyai dua nilai "Ya dan Tidak" secara bersamaan, namun besar nilainya tergantung pada bobot keanggotaan yang dimilikinya [22].

2.6.1 Operasi Himpunan Fuzzy

Operasi himpunan fuzzy diperlukan untuk proses inferensi atau penalaran. Dalam hal ini yang dioperasikan adalah derajat keanggotaanya. Derajat keanggotaan sebagai hasil dari operasi dua buah himpunan fuzzy disebut sebagai fire strength atau α-predikat [22].

Ada beberapa hal yang perlu diketahui dalam memahami sistem fuzzy, yaitu :

1. Variabel fuzzy merupakan variabel yang hendak dibahas dalam suatu sistem fuzzy.
2. Himpunan fuzzy merupakan suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel fuzzy.
3. Semesta pembicaraan adalah keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel fuzzy. Semesta pembicaraan merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai semesta pembicaraan dapat berupa bilangan positif maupun negatif. Adakalanya nilai semesta pembicaraan ini tidak dibatasi batas atasnya.
4. Domain himpunan fuzzy adalah keseluruhan nilai yang diizinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan fuzzy. Seperti halnya semesta pembicaraan, domain merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara

monoton dari kiri ke kanan. Nilai domain dapat berupa bilangan positif maupun negatif [23][24].

2.6.2 Fungsi Keanggotaan

Fungsi keanggotaan adalah grafik yang mewakili besar dari derajat keanggotaan masing-masing variabel input yang berada dalam interval antara 0 dan 1. Derajat keanggotaan sebuah variabel x dilambangkan dengan simbol μ(x). Rule- rule menggunakan nilai keanggotaan sebagai faktor bobot untuk menentukan pengaruhnya pada saat melakukan inferensi untuk menarik kesimpulan [22].

Ada beberapa fungsi yang bisa digunakan antara lain :

1. Representasi Linear, pada representasi linear pemetaan input ke derajat keanggotaannya digambarkan sebagai suatu garis lurus. Representasi Kurva Segitiga, Kurva segitiga pada dasarnya merupakan gabungan antara dua garis linear.
2. Representasi Kurva Trapesium, Kurva trapesium pada dasarnya seperti bentuk segitiga, hanya saja ada beberapa titik yang memiliki nilai keanggotaan 1.
3. Representasi Kurva Bentuk Bahu, Daerah yang terletak di tengah – tengah suatu variabel yang dipresentasikan dalam bentuk segitiga, pada sisi kanan dan kirinya akan naik dan turun. Tetapi terkadang salah satu sisi dari variabel tersebut tidak mengalami perubahan.
4. Representasi Kurva-S, Kurva PERTUMBUHAN dan PENYUSUTAN merupakan kurva-S atau sigmoid yang berhubungan dengan kenaikan dan penurunan permukaan secara tak linear.
5. Representasi Kurva Bentuk Lonceng (Bell Curve), Untuk mempresentasikan bilangan fuzzy, biasanya digunakan kurva berbentuk lonceng. Kurva berbentuk lonceng ini terbagi atas tiga kelas, yaitu kurva PI, kurva beta, dan kurva Gauss. Perbedaan ketiga kurva ini terletak pada gradientnya [24].

2.6.3 Logika Fuzzy Mamdani

Metode Mamdani paling sering digunakan dalam aplikasi-aplikasi karena strukturnya yang sederhana, yaitu menggunakan operasi MIN-MAX atau MAX-PRODUCT. Untuk mendapatkan output, diperlukan empat tahapan berikut [22] :

1. Pembentukan himpunan fuzzy. Pada proses fuzzifikasi langkah yang pertama adalah menentukan variable fuzzy dan

himpunan fuzzinya. Kemudian tentukan derajat kesepadanan (degree of match) antara data masukan fuzzy dengan himpunan fuzzy yang telah didefenisikan untuk setiap variabel masukan sistem dari setiap aturan fuzzy. Pada metode mamdani, baik variabel input maupun variabel output dibagi menjadi satu atau lebih himpunan fuzzy.

2. Aplikasi fungsi implikasi pada metode mamdani. Fungsi implikasi yang digunakan adalah min. Lakukan implikasi fuzzy berdasar pada kuat penyulutan dan himpunan fuzzy terdefinisi untuk setiap variabel keluaran di dalam bagian konsekuensi dari setiap aturan. Hasil implikasi fuzzy dari setiap aturan ini kemudian digabungkan untuk menghasilkan keluaran infrensi fuzzy [23].
3. Komposisi Aturan. Tidak seperti penalaran monoton, apabila sistem terdiri dari beberapa aturan, maka infrensi diperoleh dari kumpulan dan korelasi antar aturan. Ada 3 metode yang digunakan dalam melakukan inferensi sistem fuzzy, yaitu: max, additive dan probabilistik OR.
4. Penegasan (defuzzy). Input dari proses defuzzifikasi adalah suatu himpunan fuzzy yang diperoleh dari komposisi aturan-aturan fuzzy, sedangkan output yang dihasilkan merupakan suatu bilangan pada domain himpunan fuzzy tersebut.

$$Z^* = \frac{\int u(z)zdz}{\int u(z)dz} \qquad (2.20)$$

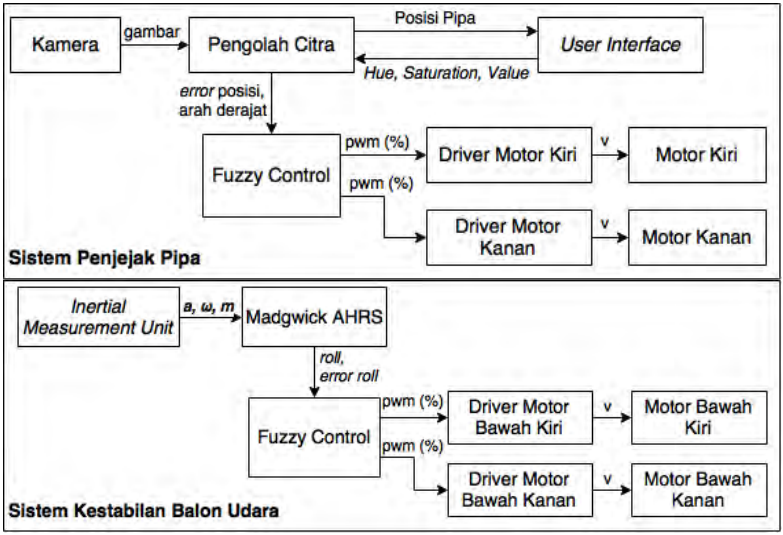
Halaman ini sengaja dikosongkan

BAB III
PERANCANGAN SISTEM

Perancangan sistem penjejak pipa dan kestabilan balon udara yang sudah terprogram terdiri dari beberapa bagian, yakni perancangan mekanik, perancangan elektrik, dan perancangan perangkan lunak.

3.1 Diagram Blok Sistem

Sistem yang dirancang terdiri dari lima bagian besar yaitu sensor, Raspberry Pi, Arduino Mega 2560, *H-Bridge*, dan Aktuator. Untuk sensor, terdiri dari sensor akselerometer, giroskop dan magnetometer yang tergabung menjadi satu pada sensor *Inertial Measurement Unit* (IMU). Serta kamera yang digunakan pada proses penjejakan pipa dengan bantuan pengolahan citra yang ada pada Raspberry Pi. Penggunaan Raspberry Pi, selain sebagai pengolah citra, juga sebagai *user interface* utama yang dapat diakses melalu layar atau *remote desktop*.



Gambar 3.1 Blok Diagram Sistem yang Dirancang

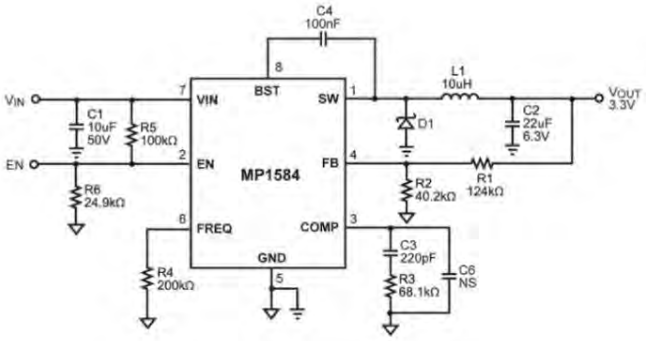
Seluruh data dari sensor IMU diolah langsung dengan algoritma Madgwick AHRS untuk didapatkan nilai *roll*-nya, yang kemudian diolah dengan logika *fuzzy* untuk menstabilkan gerak *roll* dari balon udara yang digunakan. Sedangkan data yang telah diolah oleh pengolah citra yang terdapat pada Raspberry Pi, ditransfer dengan komunikasi *serial* menuju Arduino Mega 2560 untuk diolah dengan logika *fuzzy* untuk memandu balon udara dalam mengikuti dan menjejak pipa.

3.2 Perancangan Perangkat Keras

3.2.1 Power Supply

Power Supply adalah perangkat elektronika yang mensuplai sumber listrik ke perangkat elektronika lainnya. Dalam suatu power supply terdapat sebuah regulator tegangan dimana digunakan untuk menurunkan tegangan dari satu level tertentu ke level yang diinginkan. Dalam tugas akhir ini menggunakan sebuah regulator tegangan berupa modul MP1584 yang mampu meregulasi tegangan dari rentang 4-28Vinput menjadi 0,8-25Voutput dan mampu menyuplai suatu beban sampai batas arus 3A. Diperlukan duah buah besaran tegangan yang dibutuhkan ialah 3,3V dan 5V.

Pada gambar 3.2 kapasitor berfungsi sebagai pengaman pada tegangan input maupun pada output agar tegangan yang akan masuk maupun yang akan dikeluarkan tidak terinterferensi oleh noise akibat adanya loncatan arus. Untuk mendapatkan keluaran tegangan yang dinginkan menggunakan perbandingan besar R yang berbeda-beda. Berikut perhitungan rumus untuk menentukan besar perbandingan niali resistor pada persamaan 3.1.



Gambar 3.2 Skematik *Buck Converter* (monolithicpower.com)

$$Vout = \frac{Vfb(R1+R2)}{R2} \quad (3.1)$$

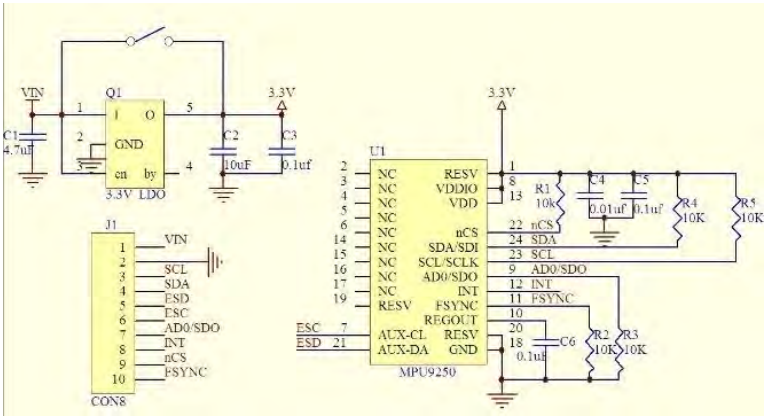
Pada beberapa aplikasi nilai R2 dijadikan 40,2 KOhm sehingga yang perlu dicari adalah besar nilai R1. Untuk nilai tegangan 3,3V nilai R1 adalah 124 KOhm da nilai R2 adalah 40,2 KOhm.

3.2.2 Inertial Measurement Unit

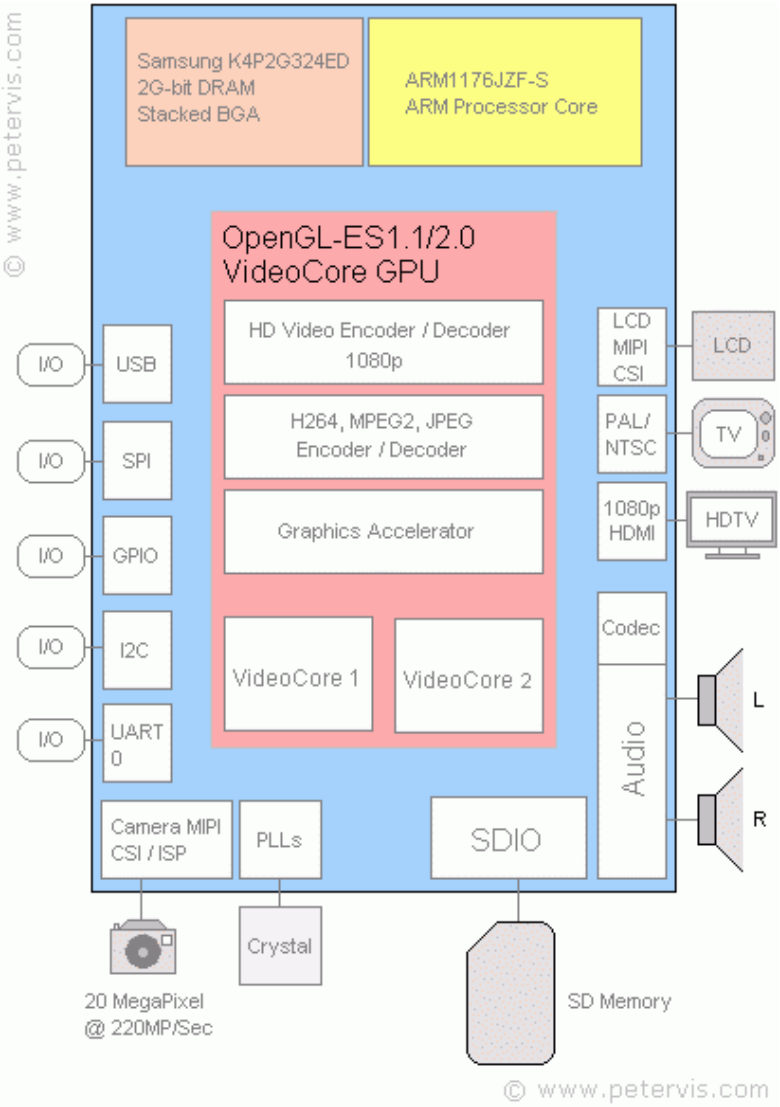
Inertial Measurement Unit (IMU) adalah sebuah sensor yang menggabungkan sensor giroskop, akselerometer dan magnetometer dalam sebuah devais. Dalam tugas akhir ini digunakan sensor IMU MPU9255 yang berupa modul GY-9255. Rangkaian dari sensor yang dipakai terlihat pada gambar 3.3.

3.2.3 *Raspberry Pi 3*

Raspberry Pi 3 digunakan sebagai pengolah citra utama, pada konektor CSI Raspberry akan dipasang sebuah kamera khusus untuk Raspberry Pi. Raspberry Pi 3 memiliki input tegangan 5V 2A. Agar dapat memenuhi suplai tegangan, maka sebuah *power bank* 4400 MAh dipasang pada balon udara. Diagram dasar dari *Raspberry Pi* dapat dilihat pada gambar 3.4.



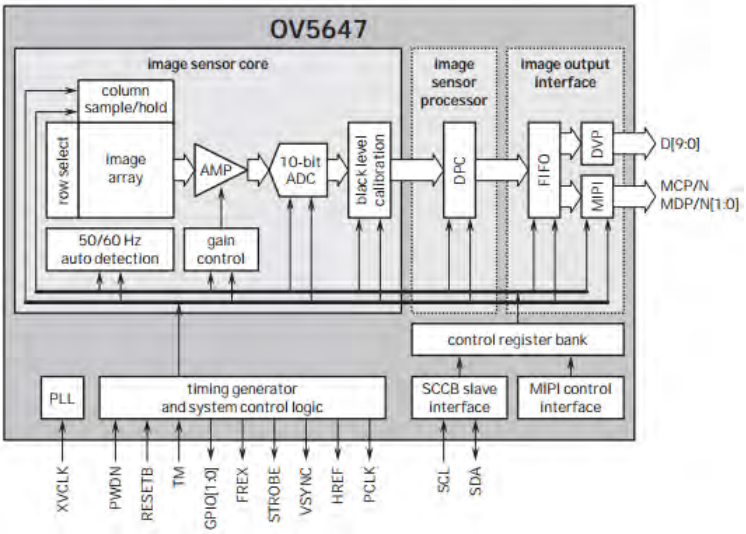
Gambar 3.3 Skematik Dasar MPU9250/MPU9255 (haoyuelectronics.com)



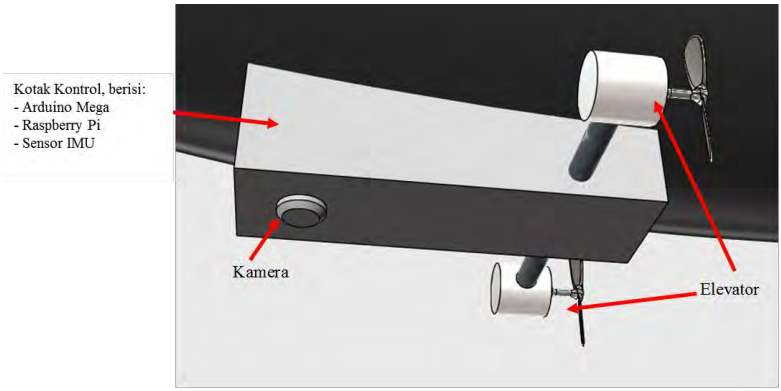
Gambar 3.4 Raspberry Pi 3 Model B Beserta Penjelasan *Port In* dan *Out* (petervis.com)

3.2.4 Modul Kamera Raspberry Pi

Modul kamera yang digunakan merupaka modul yang khusus dibuat untuk Raspberry Pi dengan kamera OV5647, konektor yang digunakan adalah CSI-2 (*Camera Serial Interface Gen-2*). Modul ini dipasang di bagian bawah titik masa balon untuk mempermudah kalkulasi pergerakan.



Gambar 3.5 Kamera OV5467 (ovt.com)



Gambar 3.6 Gambaran Peletakan Kamera, Sensor IMU, dan Raspberry Pi

3.2.5 Driver H-Bridge L298

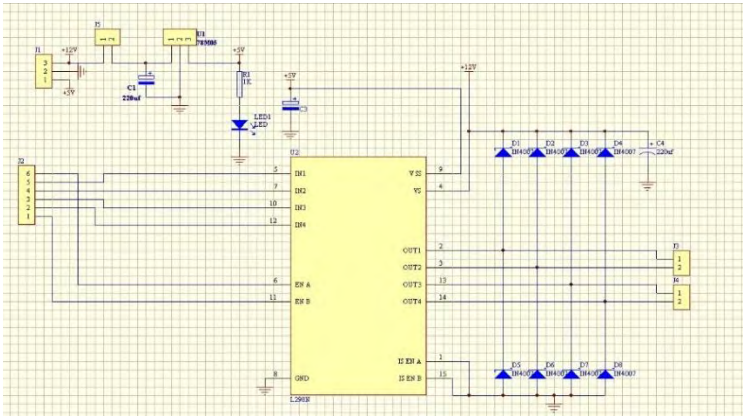
L298 adalah IC yang dapat digunakan sebagai driver motor DC. IC ini menggunakan prinsip kerja H-Bridge. Tiap H-Bridge dikontrol menggunakan level tegangan TTL yang berasal dari output mikrokontroler. L298 dapat mengontrol 2 buah motor DC. Tegangan yang dapat digunakan untuk mengendalikan robot bisa mencapai tegangan 46 VDC dan arus 2 A untuk setiap kanalnya. Berikut ini bentuk IC L298 yang digunakan sebagai motor driver:

Pengaturan kecepatan kedua motor dilakukan dengan cara pengontrolan lama pulsa aktif (mode PWM – Pulse width Modulation) yang dikirimkan ke rangkaian driver motor oleh mikrokontroller. Duty cycle PWM yang dikirimkan menentukan kecepatan putar motor DC.

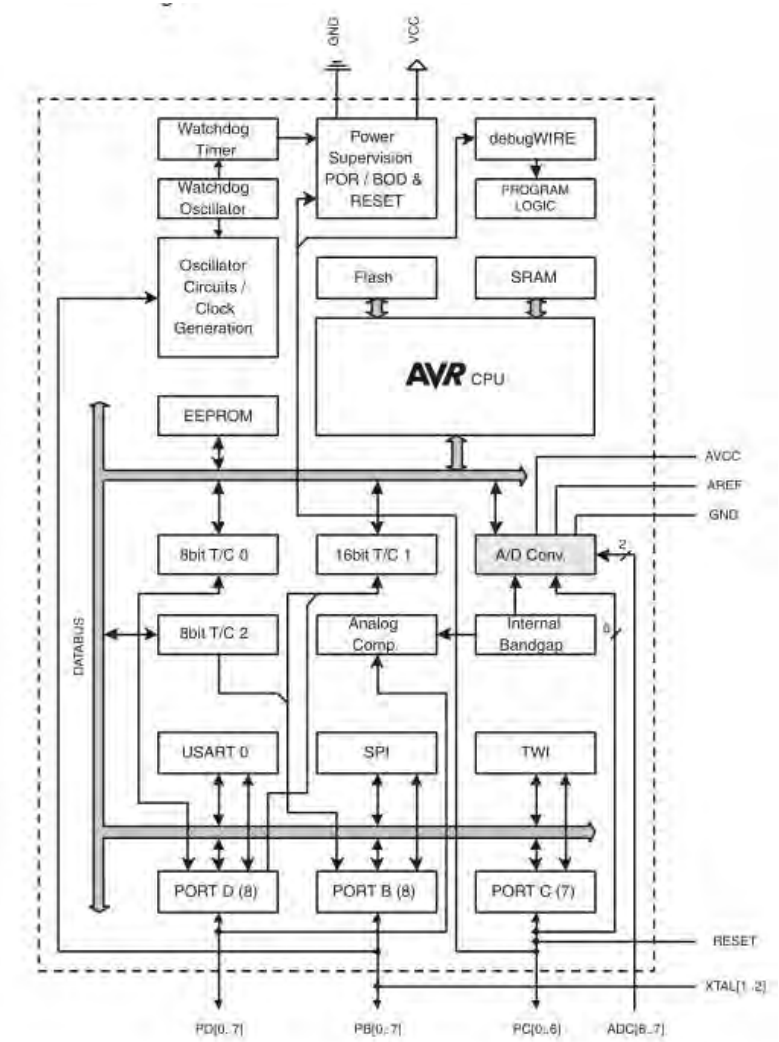
3.2.6 Arduino Mega 2560

Arduino Mini Pro merupakan sebuah board mikrokontroler berbasis ATMega2560. Modul ini memiliki 54 digital input/output dimana 15 digunakan untuk PWM output dan 16 digunakan sebagai analog input, 4 port serial, 16 MHz osilator Kristal, ICISP Header, dan tombol reset. Memiliki flash memory sebesar 256KB sangat cukup untuk menampung program yang banyak.

Arduino Mega 2560 tidak memerlukan flash program external karena di dalam chip mikrokontroler Arduino telah diisi dengan bootloader yang membuat proses upload program yang kita buat menjadi lebih sederhana dan cepat.



Gambar 3.7 Modul *Dual H-Bridge* L298N (instructables.com)



Gambar 3.8 Diagram dasar AtMega 2560 (arduino.cc)

3.2.7 Balon Udara

Balon udara dirancang untuk memiliki 4 buah motor, yaitu 2 motor *rudder* dan 2 motor *elevator*. Untuk posisi masing masing motor dapat dilihat pada gambar 3.9.

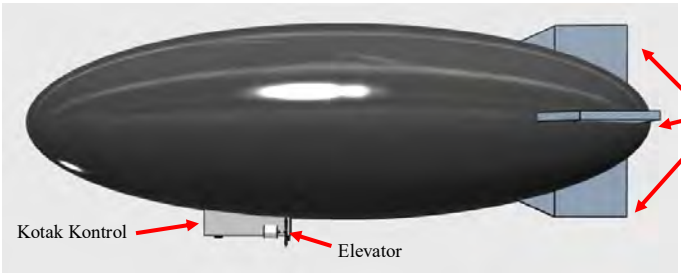
Posisi peletakan kamera berada di bagian bawah dari titik masa balon udara, hal ini dilakukan untuk mempermudah proses kalkulasi dari penjejakan pipa. Seharusnya posisi peletakan sensor IMU harus berada di titik masa dari balon. Tetapi pada tugas akhir ini posisi sensor IMU diletakan tepat diatas kamera, kerena yang digunakan hanya data *roll*.

3.3 Perancangan Perangkat Lunak

3.3.1 Proses Akuisisi Data IMU dan Madgwick AHRS

Sensor IMU diakses dengan menggunakan komunikasi I2C(*Inter-Integrated Circuit*), pengaksesan data dilakukan dengan menggunakan Arduino Mega. Sebelum proses akuisisi data, IMU terlebih dahulu dikalibrasikan dengan menggunakan nilai rata-rata dari pembacaan saat IMU dalam kondisi tetap.

Untuk mengubah data akselerasi, giroskopik, dan magnetis menjadi kuadran kuaternion, dan kemudian dapat dirubah menjadi data *roll, pitch,* dan *yaw*. Data yang digunakan dalam tugas akhir ini adalah *roll*, untuk menstabilkan posisi *roll* dari balon yang digunakan. Untuk menstabilkan balon digunakan operasi *fuzzy* dengan input berupa kondisi *roll* saat ini dan *error roll*. Nilai *error roll* didapatkan dengan mengurangkan nilai *roll* saat ini dan *roll* yang lalu. Untuk mendapatkan kestabilan balon, nilai *roll* harus 0°.



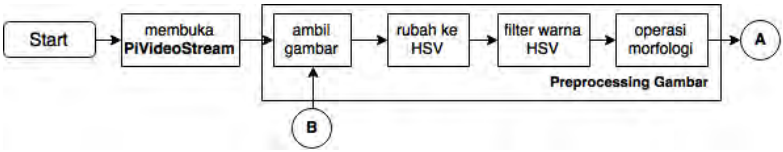
Gambar 3.9 Gambaran Peletakan Motor dan Kotak Kontrol

3.3.2 Pengambilan Citra dan Filter Warna

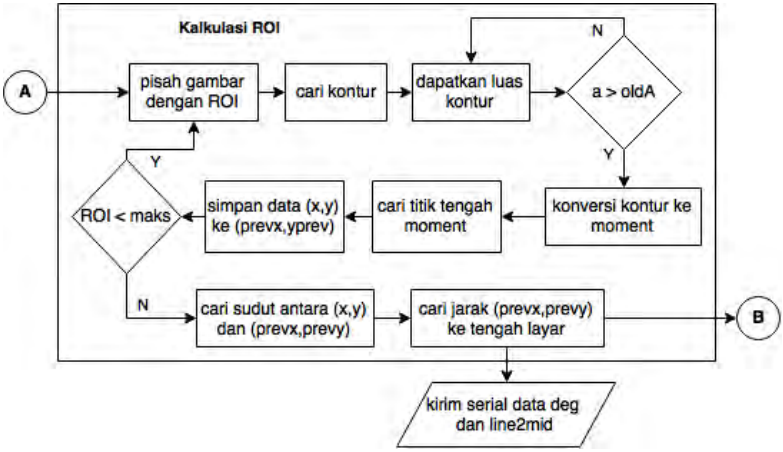
Proses pengambilan citra dan filter warna dilakukan pada Raspberry Pi 3, bahasa yang dipakai adalah bahasa python dengan versi 3.4. Proses pengambilan citra dan filter warna dibantu dengan library OpenCV 3.0. Untuk mengambil citra digunakan fungsi PiVideoStream dengan bantuan threading agar tidak memblok sistem saat dilakukan akuisisi citra.

Stream.read() digunakan untuk membaca citra dari buffer internal PiVideoStream, kemudian citra yang didapat dirubah menjadi HSV yang dilanjutkan pemfilteran warna.

Dari gambar 3.11 diatas dilakukan pengambilan gambar terlebih dahulu dengan format BGR, setelah itu gambar dirubah formatnya menjadi HSV yang kemudian gambar difilter dengan masukan dari pengguna berupa *Hue, Saturation,* dan *Value*.



Gambar 3.10 Diagram Alir Proses Pengambilan dan *Preprocessing* Gambar



Gambar 3.11 Diagram Alir Proses Kalkulasi Perbelokan dan Posisi

3.3.3 Kalkulasi Penjejak Pipa

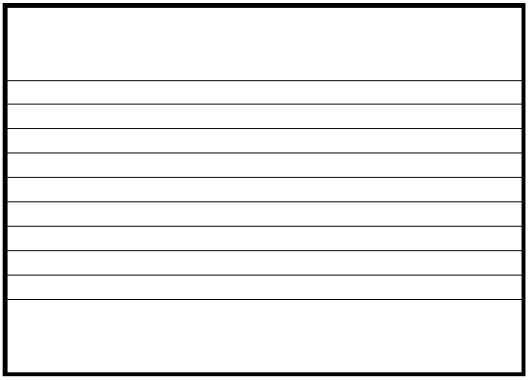
Proses penjejakan pipa dilakukan dengan menggunakan citra keluaran dari proses pengambilan citra dan filter warna, tetapi sebelum dilakukan proses penjejakan pipa, citra yang didapat dimorfologikan terlebih dahulu. Setelah citra termorfologi, citra yang didapat dapat dimasukan kedalam program penjejak pipa. Penjejakan pipa menggunakan metode ROI (*Region of Interest*) dan *contour finding*.

Metode ROI digunakan untuk mempermudah proses *contour finding*, untuk mengotak-kotakkan citra digunakan fungsi imageSource[y1:y2,x1:x2] dengan (x1,y1) merupakan titik awal dan (x2,y2) adalah titik akhir. Setiap kotak yang terbentuk dicari posisi konturnya dengan fungsi cv2.findContours setelah itu dicari kontur terluas dengan memanfaatkan fungsi cv2.contourArea. setelah didapatkan posisinya, dicari nilai tengah dari kontur yang ditemukan dengan mengubah kontur menjadi *moments* agar dapat diakses secara *array*. Nilai x dan y terletak pada kombinasi posisi array m10, m01 dan m00. Untuk mendapatkan nilai x dan y yang benar digunakan rumus:

$$x = \frac{m10}{m00} dan\ y = \frac{m01}{m00} \tag{3.2}$$

Setelah didapatkan nilai x dan y pada setiap kotak ditemukan, maka jalur dari pipa yang ingin dijejak dapat dikalkulasikan dengan rumus:

$$\deg = \tan^{-1} \frac{y1-y2}{x1-x2} * \left(\frac{180}{\pi}\right) \tag{3.3}$$

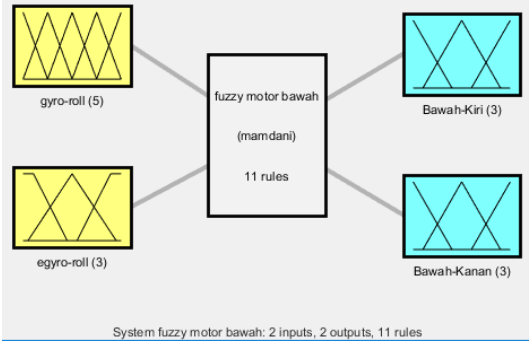


Gambar 3.12 Gambaran Lokasi ROI yang Diinginkan

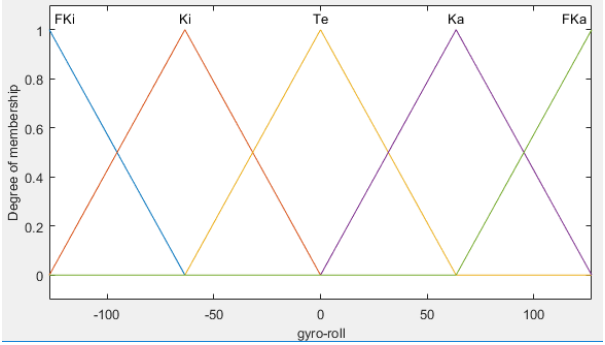
3.3.4 Perhitungan Kecepatan Motor

Perhitungan kecepatan motor dilakukan dengan menggunakan bantuan logika fuzzy untuk mempermudah proses penentuan. Pada sistem yang dirancang digunakan 2 *Fuzzy Interference System* (FIS), yaitu FIS untuk kontrol motor bawah dan FIS untuk kontrol motor samping.

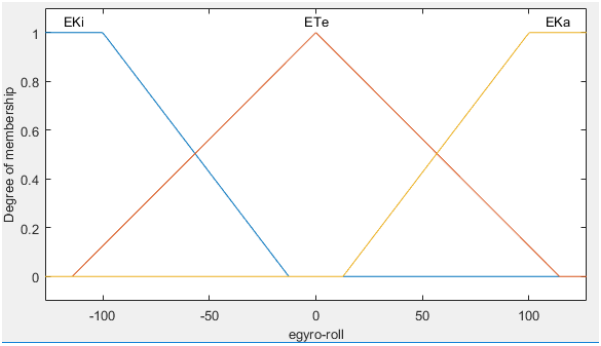
Untuk FIS motor bawah dirancang fuzzy 2 input 2 output, untuk lebih jelasnya dapat dilihat di gambar 3.12 – gambar3.16, dan untuk *fuzzy rule* dapat dilihat pada tabel 3.1.



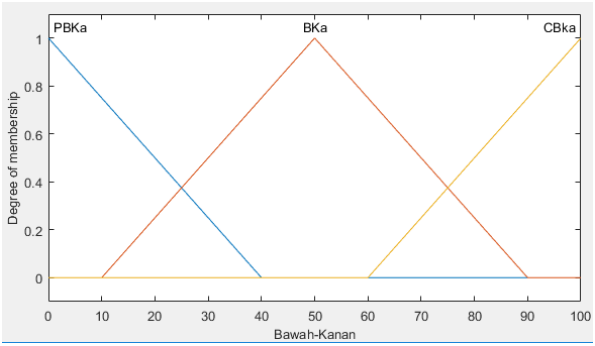
Gambar 3.13 Sistem Fuzzy yang Dirancang untuk Motor Bawah



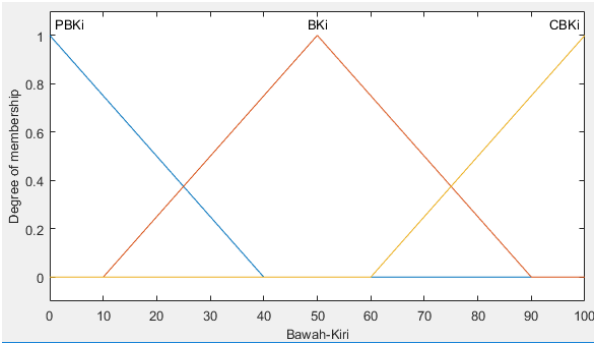
Gambar 3.14 Grafik Fungsi Keanggotaan untuk "*roll*"



Gambar 3.15 Grafik Fungsi Keanggotaan untuk "e-*roll*"



Gambar 3.16 Grafik Fungsi Keanggotaan untuk "Bawah-Kanan"



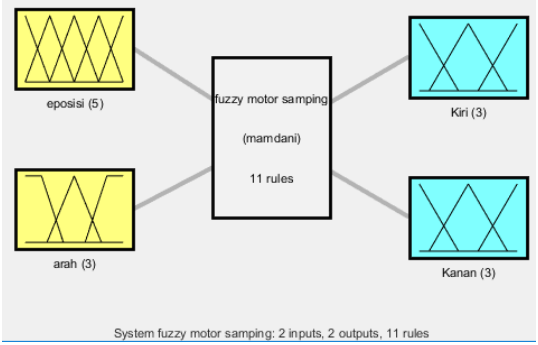
Gambar 3.17 Grafik Fungsi Keanggotaan untuk "Bawah-Kiri"

Tabel 3.1 *Fuzzy Rule* untuk Motor Bawah

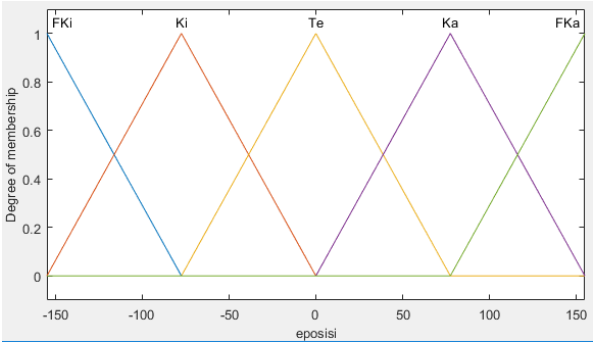
roll / e-roll	Fkiri	Kiri	Tengah	Kanan	Fkanan
eKiri	CBKi, PBKa	CBKi, PBKa	NBKi, PBKa	PBKi, NBKa	PBKi, CBKa
eTengah	CBKi, PBKa	NBKi, PBKa	PBKi, PBKa	PBKi, NBKa	PBKi, CBKa
eKanan	CBKi, PBKa	NBKi, PBKa	PBKi, NBKa	PBKi, CBKa	PBKi, CBKa

Keterangan:
CBKi = Cepat Bawah Kiri,
NBKi = Normal Bawah Kiri,
PBKi = Pelan Bawah Kiri,
CBKi = Cepat Bawah Kiri,
NBKi = Normal Bawah Kiri,
PBKa = Pelan Bawah Kanan.

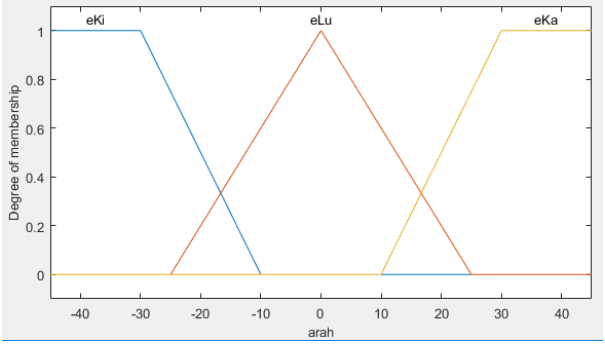
Untuk FIS motor samping dirancang fuzzy 2 input 2 output, untuk lebih jelasnya dapat dilihat di gambar 3.17 – gambar 3.21, dan untuk *fuzzy rule* dapat dilihat pada tabel 3.2.



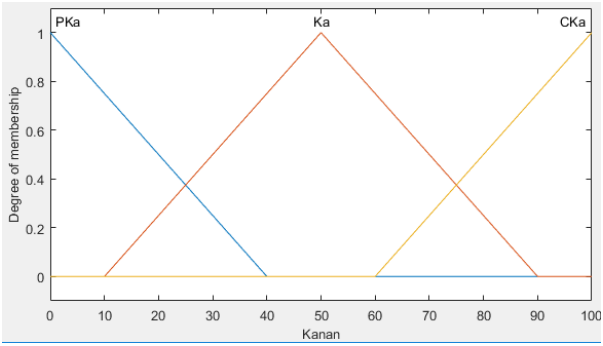
Gambar 3.18 Sistem Fuzzy yang Dirancang untuk Motor Samping



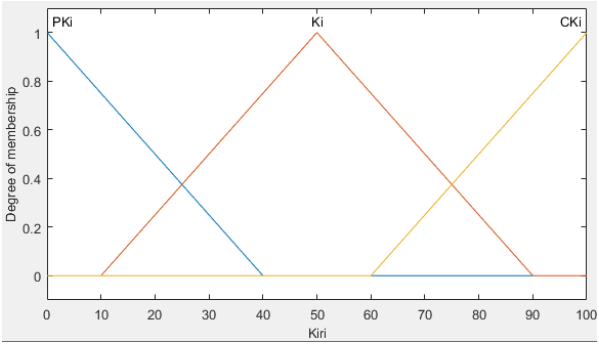
Gambar 3.19 Grafik Fungsi Keanggotaan untuk "e-posisi"



Gambar 3.20 Grafik Fungsi Keanggotaan untuk "arah"



Gambar 3.21 Grafik Fungsi Keanggotaan untuk "Motor-Kanan"



Gambar 3.22 Grafik Fungsi Keanggotaan untuk "Motor-Kiri"

Tabel 3.2 *Fuzzy Rule* untuk Motor Samping

e-posisi / arah	Fkiri	Kiri	Tengah	Kanan	Fkanan
eKiri	PKi, CKa	NKi, NKa	NKi, CKa	PKi, CKa	CKi, PKa
eLurus	PKi, CKa	NKi, CKa	CKa, CKa	NKi, CKa	CKi, PKa
eKanan	PKi, CKa	CKi, PKa	CKa, CKi	NKi, NKa	CKi, PKa

Keterangan:
CKi = Cepat Kiri,
NKi = Normal Kiri,
PKi = Pelan Kiri,
CKi = Cepat Kiri,
NKi = Normal Kiri,
PKa = Pelan Kanan.

Perhitungan *fuzzy* menggunakan metode mamdani dengan menggunakan operasi MIN-MAX dan untuk *defuzzyfication*-nya menggunakan operasi *centroid.* Semua output berupa pwm kecepatan motor. Dan semua operasi *fuzzy* dilakukan pada Arduino Mega.

Halaman ini sengaja dikosongkan

BAB IV
PENGUJIAN DAN ANALISIS

4.1 Realisasi Desain Balon Udara

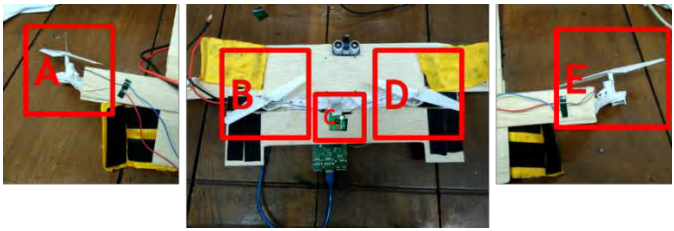
Pada tugas akhir ini digunakan dua balon udara air swimmer yang akan diisi dengan helium. Volume dari balon udara ini adalah 0.1274 m3. Dua balon udara air swimmer mempunyai volume untuk diisi helium sebesar 0.2548 m3.

Helium utamanya digunakan sebagai gas pengangkat pada kapal udara. Permintaan atas gas helium meningkat semasa Perang Dunia II. Spektrometer massa helium juga sangat vital dalam proyek bom atom Manhattan. Dengan isian 1 liter helium balon udara mampu mengangkat besaran 1 Kg. Sehingga balon udara *air swimmer* dengan volume 0.2548 m3 dapat mengangkat beban sekitar 254 gram.

Gambar 4.1 Realisasi Balon Udara yang Digunakan

Tabel 4.1 Daya Angkat Gas Hidrogen dan Helium

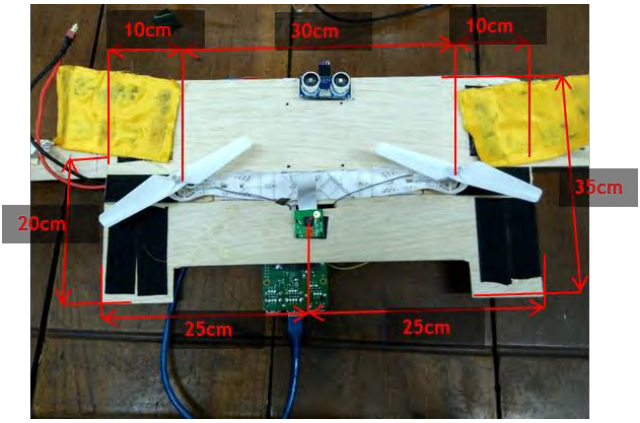
	Berat Gas (kg)	Berat Udara (kg)	Beban Maksimum (kg)
Hidrogen	0,085	1,223	1,138
Helium	0,169	1,223	1,054



Gambar 4.2 Realisasi Kotak Kontrol dan Lokasi Motor

Keterangan:
A = Motor Samping Kanan
B = Motor Bawah Kanan
C = Kamera
D = Motor Bawah Kiri
E = Motor Samping Kiri

Gambar 4.2 menunjukkan lokasi daripada kamera beserta motor yang digunakan. Pada alat yang dibuat, digunakan dua set motor yaitu motor bawah dan mtor samping. Hal ini digunakan untuk mempermudah perhitungan belok serta perhitungan dari keseimbangan. Gambar 4.3 menunjukkan ukuran dari lokasi penempatan kamera, motor bawah kiri dan kanan, serta ukuran kotak kontrol yang digunakan.



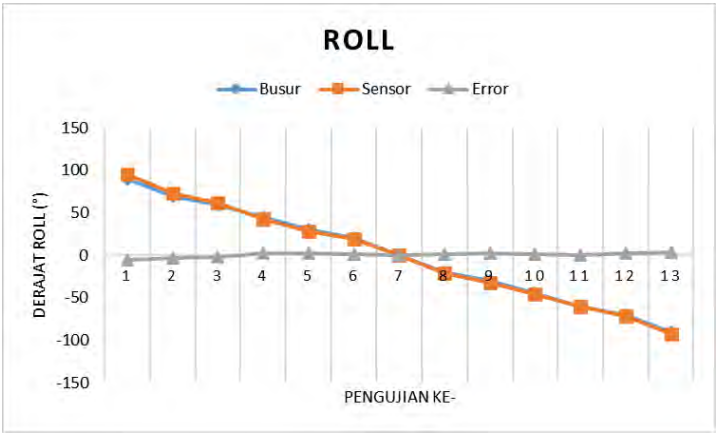
Gambar 4.3 Posisi Kamera dan Motor Bawah pada Kotak Kontrol

Gambar 4.4 Posisi Motor Samping terhadap Kotak Kontrol

4.2 Pengujian Hasil Pembacaan Roll pada Arduino

Pengujian ini dilakukan untuk mengukur performa dari filter Magdwick untuk Inertial Measurement Unit yang digunakan. Pengujian dilakukan dengan bantuan busur 360°, sensor diposisikan pada 0° pada saat kalibrasi awal. Kemudian data diambil dengan mengatur sudut posisi sensor dengan referensi 0° pada bidang horizontal. Didapatkan data sesuai pada tabel 4.2.

Seperti yang tampak pada Gambar 4.5 didapatkan hasil bahwa derajat *roll* hasil dari AHRS memiliki kesalahan rata-rata sebesar 0.3° dengan acuan busur derajat. Dengan kesalahan maksimum sebesar 5° dan minimum sebesar 0°.



Gambar 4.5 Grafik Pengujian Data *Roll* dengan referensi Busur Derajat.

Tabel 4.2 Hasil Pengujian *Roll* dengan Menggunakan Madgwick AHRS dengan Referensi Busur Derajat.

Roll (°)		
Busur derajat	AHRS	*Error*
90	95	-5
70	73	-3
60	62	-2
45	43	2
30	28	2
20	19	1
0	0	0
-20	-21	1
-30	-32	2
-45	-46	1
-60	-60	0
-70	-72	2
-90	-93	3

4.3 Pengujian Madgwick AHRS

Pengujian ini dilakukan untuk mengetahui kemampuan dari pada algoritma Madgwick AHRS yang digunakan, terdapat dua data yaitu data mentah dari giroskop dan data keluaran dari algoritma yang digunakan. Pengujian dilakukan dengan mengubah posisi dari sensor sebesar 90° ke kiri (-90) dan 90° ke kanan (90). Didapatkan data sesuai dengan tabel 4.3.

Tabel 4.3 Hasil Pengujian Madgwick AHRS

Detik	Sudut Uji (°)	Raw (°)	Madgwick (°)
1	0	0,1	0,2
2	0	0,1	-0,4
3	0	0,2	-0,4
4	0	0,1	0,1
5	0	0,1	-0,4

Detik	Sudut Uji (°)	Raw (°)	Madgwick (°)
6	0	0,1	0,0
7	0	0,2	-0,2
8	0	0,1	-0,2
9	0	0,2	-0,3
10	0	0,2	-0,3

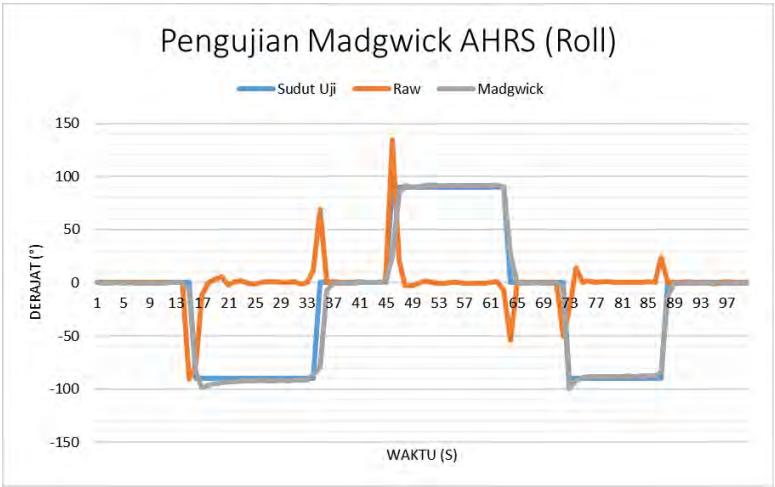
Detik	Sudut Uji (°)	Raw (°)	Madgwick (°)
11	0	0,1	-0,2
12	0	0,1	-0,1
13	0	0,0	-0,1
14	0	-0,1	-0,1
15	0	-90,7	-7,0
16	-90	-76,0	-84,0
17	-90	-11,4	-98,6
18	-90	0,2	-96,7
19	-90	3,3	-94,4
20	-90	5,9	-93,6
21	-90	-1,9	-93,1
22	-90	0,9	-93,0
23	-90	2,2	-92,1
24	-90	-0,8	-92,2
25	-90	-1,3	-92,2
26	-90	0,2	-91,8
27	-90	1,2	-92,2
28	-90	0,8	-92,2
29	-90	0,0	-91,8
30	-90	0,4	-92,0
31	-90	1,2	-91,7
32	-90	-1,6	-91,6
33	-90	0,6	-91,9
34	-90	11,4	-86,1
35	0	69,1	-79,9
36	0	0,3	-7,2

Detik	Sudut Uji (°)	Raw (°)	Madgwick (°)
37	0	0,8	-0,1
38	0	0,7	-0,3
39	0	0,5	-0,2
40	0	0,5	-0,2
41	0	0,7	0,0
42	0	0,3	0,1
43	0	0,6	0,0
44	0	0,4	0,2
45	0	1,2	0,2
46	90	134,8	24,5
47	90	21,1	84,2
48	90	-2,5	91,3
49	90	-2,9	89,8
50	90	-0,1	90,4
51	90	1,6	91,3
52	90	0,1	91,8
53	90	-0,2	91,7
54	90	-0,2	91,7
55	90	0,2	91,4
56	90	0,0	91,4
57	90	-0,3	91,7
58	90	-0,8	91,5
59	90	-0,5	91,6
60	90	-0,1	91,3
61	90	0,3	91,5
62	90	1,2	91,7

Detik	Sudut Uji (°)	Raw (°)	Madgwick (°)
63	90	-6,9	90,8
64	0	-53,9	25,8
65	0	0,6	-0,2
66	0	0,3	-0,2
67	0	0,2	-0,1
68	0	0,2	0,3
69	0	0,1	-0,4
70	0	0,1	-0,3
71	0	0,2	-0,2
72	0	-49,9	-3,4
73	-90	-25,9	-99,7
74	-90	14,1	-92,1
75	-90	0,6	-88,8
76	-90	1,8	-88,5
77	-90	0,6	-88,3
78	-90	0,9	-88,3
79	-90	1,0	-88,3
80	-90	0,4	-88,6
81	-90	0,0	-88,4

Detik	Sudut Uji (°)	Raw (°)	Madgwick (°)
82	-90	0,5	-87,8
83	-90	0,5	-88,0
84	-90	0,6	-87,9
85	-90	0,8	-87,9
86	-90	0,6	-87,4
87	-90	24,5	-84,2
88	0	0,4	-11,8
89	0	0,6	-0,3
90	0	0,6	-0,3
91	0	0,8	-0,2
92	0	0,7	-0,2
93	0	0,5	-0,4
94	0	0,5	-0,2
95	0	-1,4	0,1
96	0	0,3	-0,2
97	0	0,8	-0,3
98	0	0,1	-0,5
99	0	0,1	-0,5
100	0	0,1	-0,4

Dari data pada tabel 4.3, didapatkan data dari giroskop, akselerometer dan magnetometer yang bersih dari drift. Hal ini terbukti dengan hasil yang didapat pada saat giroskop diam, pada saat giroskop tidak bergerak terdapat nilai drift yang terlihat pada gambar 4.6 garis warna jingga. Nilai drift ini terkompensasi pada algoritma Madgwick sehingga didapatkan keluaran yang hampir menyerupai nilai sudut uji, untuk hasil algoritma Madgwick dapat dilihat pada gambar 4.6 garis abu-abu dan nilai sudut uji dengan garis biru.



Gambar 4.6 Grafik Pengujian Madgwick AHRS

4.4 Pengujian Jarak Perhitungan dibandingkan dengan Jarak Sesungguhnya

Pengujian ini dilakukan untuk mengetahui error dari perhitungan berdasarkan pengolahan citra dibandingkan dengan jarak sesungguhnya, pada percobaan dilakukan jarak antara pipa dengan kamera fix pada jarak 18cm, dengan lebar pipa sebesar 2,2cm. Cara pengujian dilakukan dengan menaruh penggaris di bawah pipa sebagai referensi dan dilakukan pembandingan antara hasil perhitungan dan jarak dari penggaris. Didapatkan data seperti yang terlihat pada tabel 4.4 dan gambar 4.8.

Seperti yang tampak pada tabel 4.4 dan gambar 4.7, didapatkan bahwa perhitungan yang dilakukan secara *software* untuk menentukan jarak pada tiap piksel dibandingkan dengan penggaris memiliki kesalahan maksimum sebesar 0.3 cm dan kesalahan minimum sebesar 0.1 cm.

Tabel 4.4 Hasil Pengujian Kamera dengan Referensi Penggaris.

Penggaris (cm)	Perhitungan (cm)	Error (cm)
10	9,9	0,1
9	8,9	0,1
8	7,8	0,2
7	6,9	0,1
6	5,8	0,2
5	4,7	0,3
4	3,8	0,2
3	2,7	0,3
2	1,8	0,2
0	0	0
-2	-1,7	-0,3
-3	-2,9	-0,1
-4	-3,7	-0,3
-5	-4,7	-0,3
-6	-5,8	-0,2
-7	-6,7	-0,3
-8	-7,9	-0,1
-9	-8,8	-0,2
-10	-9,8	-0,2

Gambar 4.7 Grafik Pengujian Perhitungan Kamera dengan Referensi Penggaris.

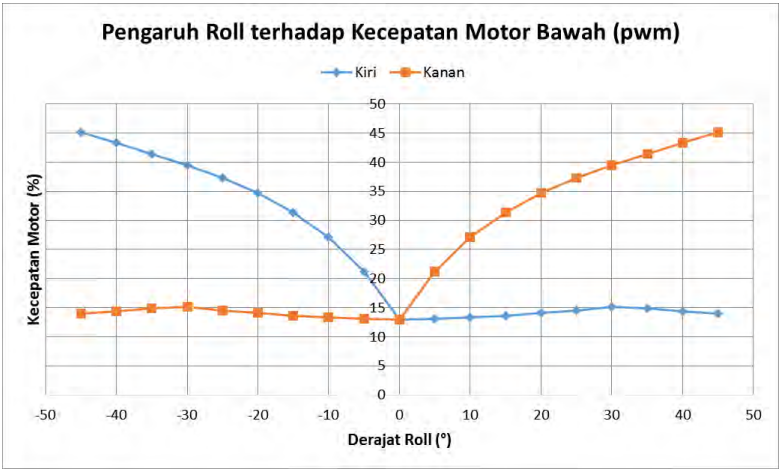
Gambar 4.8 Posisi Penggaris Terhadap Pipa.

4.5 Pengujian Respon Motor Bawah

Pengujian ini dilakukan untuk mengetahui respon motor bawah kiri dan bawah kanan terhadap perubahan *roll*. Percobaan dilakukan dengan memberi data secara langsung (*fixed*) dan hasil dioutputkan oleh Arduino dengan Serial.print(). Data dimasukan manual dengan selang 5° dengan awalan -45° dan akhir 45° untuk *roll* dan *error roll (E-roll)*. Didapatkan data berikut:

Tabel 4.5 Hasil Pengujian Respon Motor Bawah.

Input(IMU)		Output(Motor)	
Roll (°)	E-Roll (°)	Kiri (%)	Kanan (%)
45	0	14	45
40	0	14	43
35	0	15	41
30	0	15	39
25	0	15	37
20	0	14	35
15	0	14	31
10	0	13	27
5	0	13	21
0	0	13	13
-5	0	21	13
-10	0	27	13
-15	0	31	14
-20	0	35	14
-25	0	37	15
-30	0	39	15
-35	0	41	15
-40	0	43	14
-45	0	45	14



Gambar 4.9 Grafik Pengujian Respon Motor Bawah.

Seperti yang tampak pada gambar 4.9, didapatkan bahwa motor kiri bergerak ketika terjadi *roll* antara 0° - -50° sedangkan untuk motor kanan bergerak ketika terjadi *roll* antara 0° - 50°. Hal ini sesuai dengan yang dirancang pada tabel 3.1 yang berisi tentang *fuzzy rule* yang dirancang

4.6 Pengujian Respon Motor Samping

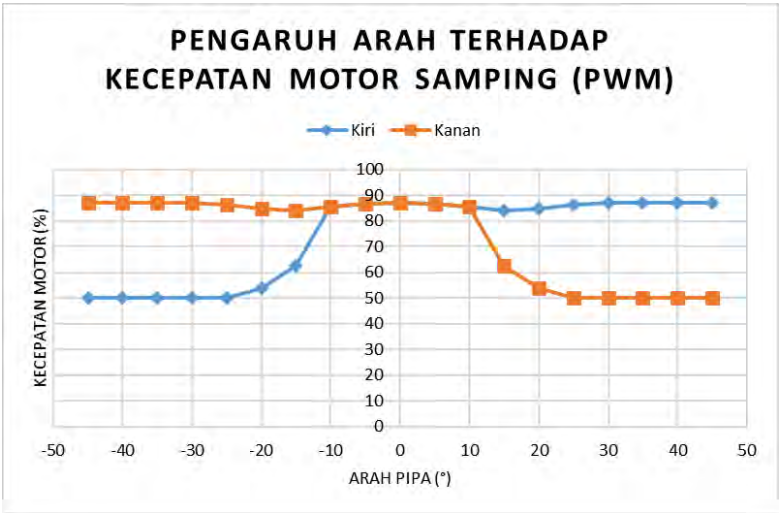
Pengujian ini dilakukan untuk mengetahui respon motor kiri dan kanan dengan input berupa data e-posisi dan arah. Percobaan dilakukan dengan memberi data secara langsung (*fixed*) dan hasil dioutputkan oleh Arduino dengan Serial.print(). Data dimasukan manual dengan selang 10px dengan awalan -75px dan akhir 75px untuk e-posisi, dan selang 5° dengan awalan -45° dan akhir 45° untuk arah.

Pengaruh dari *error* posisi (E-Posisi) untuk motor samping kiri dan kanan dapat dilihat pada tabel 4.6. Didapatkan kesimpulan bahwa motor samping kiri bergerak ketika *error* posisi terjadi diantara 0 - -75 cm dan motor samping kanan bergerak ketika terjadi *error* posisi antara 0 – 75 cm hal ini sudah sesuai dengan *fuzzy rule* yang telah dibuat pada tabel 3.2.

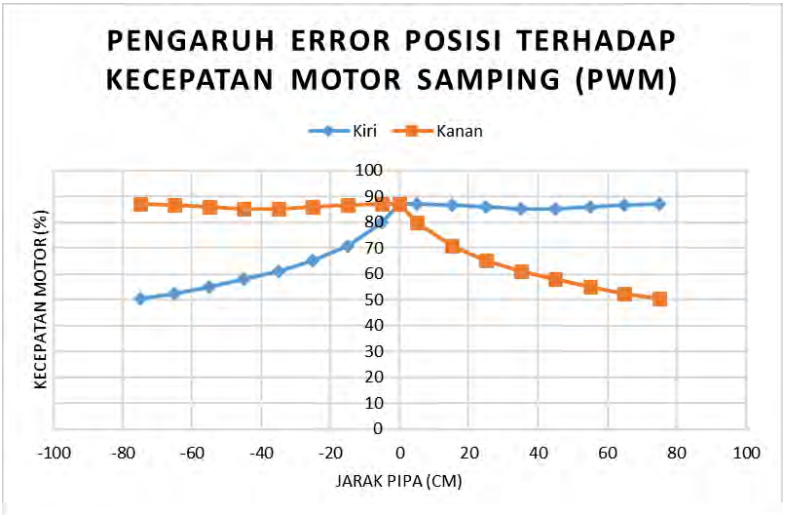
Sedangkan untuk pengaruh arah pipa untuk motor samping kiri dan kanan dapat diliha pada tabel 4.7. didapatkan kesimpulan bahwa motor samping kiri bergerak ketika nilai arah terjadi diantara 0° - 45° dan motor samping kanan bergerak ketika nilai arah terjadi diantara 0° - -45°

Tabel 4.6 Hasil Pengujian Respon Motor Samping degan Arah = 0.

Input(Kamera)		Output(Motor)	
E-Posisi (cm)	Arah (°)	Kiri (%)	Kanan (%)
-75	0	50	87
-65	0	52	87
-55	0	55	86
-45	0	58	85
-35	0	61	85
-25	0	65	86
-15	0	71	87
-5	0	80	87
0	0	87	87
5	0	87	80
15	0	87	71
25	0	86	65
35	0	85	61
45	0	85	58
55	0	86	55
65	0	87	52
75	0	87	50



Gambar 4.10 Grafik Pengujian Respon Motor Samping dengan Arah = 0.



Gambar 4.11 Grafik Pengujian Respon Motor Samping dengan E-Posisi = 0.

Tabel 4.7 Hasil Pengujian Respon Motor Samping dengan E-Posisi = 0.

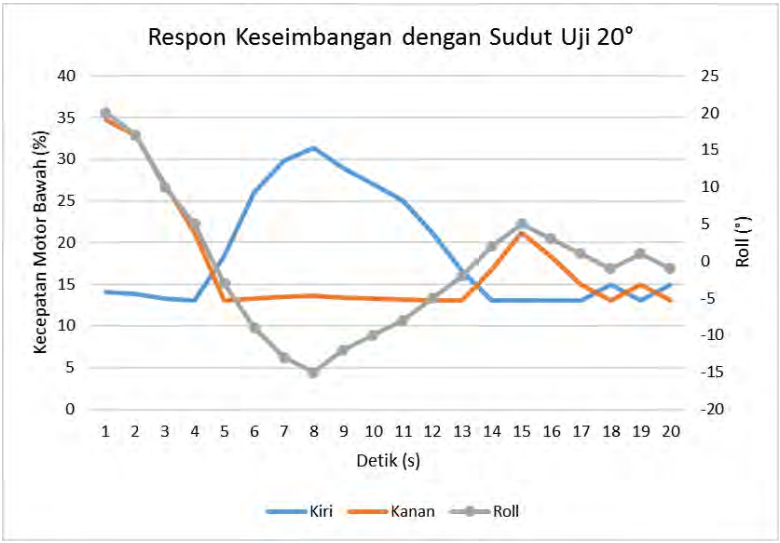
Input(Kamera)		Output(Motor)	
E-Posisi (cm)	Arah (°)	Kiri (%)	Kanan (%)
0	45	87	50
0	40	87	50
0	35	87	50
0	30	87	50
0	25	86	50
0	20	85	54
0	15	84	62
0	10	85	85
0	5	87	87
0	0	87	87
0	-5	87	87
0	-10	85	85
0	-15	62	84
0	-20	54	85
0	-25	50	86
0	-30	50	87
0	-35	50	87
0	-40	50	87
0	-45	50	87

4.7 Pengujian Respon Keseimbangan

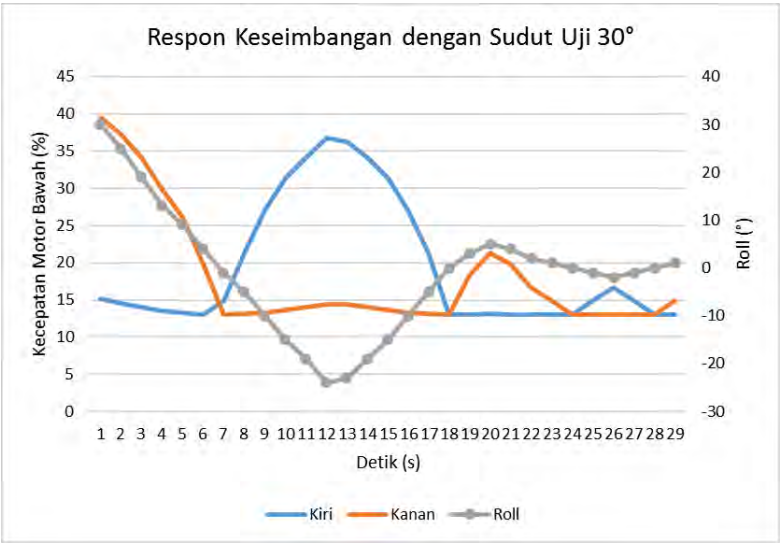
Pengujian ini dilakukan untuk mengetahui respon dari sistem keseimbangan pada balon yang telah dirancang. Pengujian dilakukan dengan memiringkan balon dan dibiarkan terlepas agar sistem dapat menseimbangkan balon yang digunakan. Pengujian dilakukan sebanyak tiga kali dengan sudut sebesar 30° dan 20° miring kanan, serta -20° miring kiri. Didapatkan hasil sesuai dengan tabel 4.8 untuk sudut uji 20°.

Tabel 4.8 Hasil Pengujian Respon Keseimbangan dengan sudut uji sebesar 20°

Detik (s)	Sudut Uji (°)	Kiri (%)	Kanan (%)	Roll (°)
1	20	14	35	20
2		14	33	17
3		13	27	10
4		13	21	5
5		18	13	-3
6		26	13	-9
7		30	14	-13
8		31	14	-15
9		29	13	-12
10		27	13	-10
11		25	13	-8
12		21	13	-5
13		17	13	-2
14		13	17	2
15		13	21	5
16		13	18	3
17		13	15	1
18		15	13	-1
19		13	15	1
20		15	13	-1



Gambar 4.12 Grafik Pengujian Respon Keseimbangan (20°)



Gambar 4.13 Grafik Pengujian Respon Keseimbangan (30°)

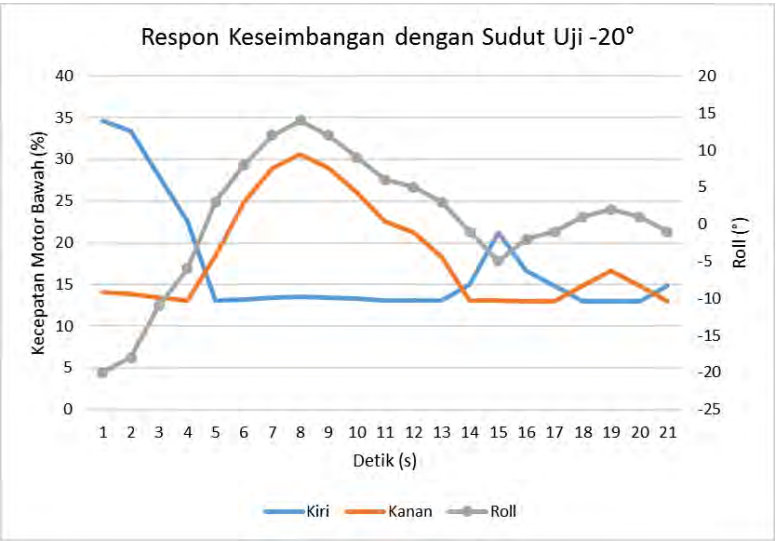
Tabel 4.9 Hasil Pengujian Respon Keseimbangan dengan Sudut Uji sebesar 30°

Detik (s)	Sudut Uji (°)	Kiri (%)	Kanan (%)	Roll (°)
1	30	15	39	30
2		15	37	25
3		14	34	19
4		14	30	13
5		13	26	9
6		13	20	4
7		15	13	-1
8		21	13	-5
9		27	13	-10
10		31	14	-15
11		34	14	-19
12		37	14	-24
13		36	14	-23
14		34	14	-19
15		31	14	-15
16		27	13	-10
17		21	13	-5
18		13	13	0
19		13	18	3
20		13	21	5
21		13	20	4
22		13	17	2
23		13	15	1
24		13	13	0
25		15	13	-1
26		17	13	-2
27		15	13	-1
28		13	13	0
29		13	15	1

Tabel 4.10 Hasil Pengujian Respon Keseimbangan dengan Sudut Uji -20°

Detik (s)	Sudut Uji (°)	Kiri (%)	Kanan (%)	Roll (°)
1	-20	35	14	-20
2		33	14	-18
3		28	13	-11
4		23	13	-6
5		13	18	3
6		13	25	8
7		13	29	12
8		14	31	14
9		13	29	12
10		13	26	9
11		13	23	6
12		13	21	5
13		13	18	3
14		15	13	-1
15		21	13	-5
16		17	13	-2
17		15	13	-1
18		13	15	1
19		13	17	2
20		13	15	1
21		15	13	-1

Dari data yang telah didapatkan, respon keseimbangan pada balon dengan menggunakan logika *fuzzy* dapat mengontrol balon agar dapat kembali ke *set-point* 0°. Pada pengujian dengan sudut uji sebesar 20° baik dari kiri maupun kanan, didapatkan rata-rata respon untuk kembali ke posisi semula sebesar kurang lebih 20 detik, sedangkan untuk pengujian 30° didapatkan sebesar 29 detik. Pada proses pengujian didapatkan bahwa masih terjadi *overshoot* yang cukup besar pada prosesnya.



Gambar 4.14 Grafik Pengujian Respon Keseimbangan (-20°)

Halaman ini sengaja dikosongkan

LAMPIRAN

Kode utama pada Raspberry Pi:

```
#tentukan penggunaan kamera dan serial
usePiCamera = 0  #0 = webcam;   1 = piCamera;  2 = gambar
useLowRes   = 1  #0 = 640,480;  1 = 320,240; hanya untuk
piCamera
useSerial   = 0  #0 = noSerial; 1 = Serial
serialPort  = 'COM6'
baud = 115200
#tentukan jumlah ROI
column = 9
row    = 15
border = 3

class ExampleApp(QtGui.QMainWindow, coba1.Ui_MainWindow):
 def __init__(self):
  super(self.__class__, self).__init__()
  self.setupUi(self)
  #
  self.serialView_B.clicked.connect(self.saveSerialView)
  self.debugView_B.clicked.connect(self.saveDebug)
  self.Start_B.clicked.connect(self.startTracking)
  self.startTrack = False
  #
  if (useSerial==1):
   self.ser = serial.Serial(
    port = serialPort,
    baudrate = baud,
    parity = serial.PARITY_NONE,
    stopbits = serial.STOPBITS_ONE,
    bytesize = serial.EIGHTBITS,
    timeout = 1
   )
   sleep(1)
   self.thread =
threading.Thread(target=self.getSerialData(14),args=(self.se
r,))
   self.thread.start()
  self.roiMean = np.zeros((row,column))
  self.setFPS(30)
  if(useLowRes==1):
   self.resolution = (320,240)
  else:
   self.resolution = (640,480)
  if(usePiCamera==1):
   self.stream = PiVideoStream(resolution =
self.resolution,framerate = self.fps)
```

```
  elif(usePiCamera==0):
   self.cap = cv2.VideoCapture(0)
  self.initData(500)
   self.serialView.append("hari;bulan;tahun;jam;menit;detik;
ax;ay;az;gx;gy;gz;mx;my;mz;pitch;yaw;roll;heading;height;")
  self.start()
 def setFPS(self, fps):
  self.fps = fps
 def startTracking(self):
  self.startTrack = not self.startTrack
  if (self.startTrack):
   self.Start_B.setText("STOP")
  else:
   self.Start_B.setText("START")
 def nextFrame(self):
  #ambil index tab saat ini
  self.indexTab = self.tabWidget.currentIndex()
  #load template landing
  self.imagetemplate = cv2.imread('test.png',1)
  #penggunaan kamera
  if(usePiCamera==1):
   orimg = self.stream.read()
  elif(usePiCamera==0):
   ret, orimg = self.cap.read()
   if useLowRes==1:
    orimg =
cv2.resize(orimg,(self.resolution[0],self.resolution[1]),int
erpolation = cv2.INTER_CUBIC)
  else:
   orimg = cv2.imread('4.jpg',1)
   if useLowRes==1:
    orimg =
cv2.resize(orimg,(self.resolution[0],self.resolution[1]),int
erpolation = cv2.INTER_CUBIC)
  #mulai pendeteksian
  self.imgthres,self.hsv = self.preprocessingImage(orimg)
  #fungsi tab landing
  if(self.indexTab==2 or self.startTrack):
   self.imgthres,deg = self.roiCalculate(self.imgthres,
orimg)
   print(deg)
   self.imgthres = cv2.cvtColor(self.imgthres,
cv2.COLOR_BGR2RGB)
   if (useSerial==1):
    Raw = self.serData
   else:
    Raw = np.zeros(14)
    for i in range(0,14,1):
        Raw[i] = int(random(0,300))
   self.updateRawData(Raw)
```

```
    self.appendData(Raw)
   #fungsi tab plot IMU
  def roiCalculate(self, imageSource, oriImage):
   oriImage = cv2.bitwise_and(oriImage,oriImage,mask =
imageSource)
   roiHeight = imageSource.shape[0]/row
   roiWidth  = imageSource.shape[1]
   deg = 0
   xy = np.zeros((row,2))
   imageOut = imageSource
   font = cv2.FONT_HERSHEY_COMPLEX
   for i in range(border,row-border,1):
    pt1x = 0
    pt1y = int(roiHeight*i)
    pt2x = int(pt1x+roiWidth)
    pt2y = int(pt1y+roiHeight)
    imgROI = imageSource[pt1y:pt2y,pt1x:pt2x]
    cnts= cv2.findContours(imgROI, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    cnts = cnts[0] if imutils.is_cv2() else cnts[1]
    largArea = 0
    largAreaIndex = 0
    for cnt in cnts:
     a = cv2.contourArea(cnt)
     if(a > largArea):
      largArea = a
      largAreaIndex = cnt
    if(cnts!=[]):
     M = cv2.moments(largAreaIndex)
     cx = int(M['m10']/M['m00'])
     cy = int(M['m01']/M['m00'])
     cy += int(pt1y + (roiHeight/2))
     xy[i][0] = cx
     xy[i][1] = cy
cv2.putText(oriImage,str(cx)+','+str(cy)+','+str(i),(cx,cy),
font, 0.25, (255,255,255),1,8)
     if(i<=int(row/2)):
      if(i!=border):
       cv2.line(oriImage,(int(xy[i-1][0]),int(xy[i-
1][1])),(int(cx),int(cy)),(255,255,0),1)
      cv2.circle(oriImage,(int(cx),int(cy)),5,(255,255,0),1)
  for i in xy:
   if (i[0] != 0):
    if (i[1] != 0):
     deg = math.atan2( (imageSource.shape[0]-i[1]) -
(imageSource.shape[0]-xy[int(row/2)][1]) ,
          (i[0]-xy[int(row/2)][0])) * (180/math.pi)
     break
cv2.line(oriImage,(int(imageSource.shape[1]/2),0),(int(image
Source.shape[1]/2),imageSource.shape[0]),(255,0,0),1)
```

```
  line2mid = 1 + (int(xy[int(row/2)][0]) -
int(imageSource.shape[1]/2))
  direction = 90 - deg
  print(line2mid)
  # self.ser.write('ok'.encode())
  # print(xy[0][0],xy[0][1])
  # return imageOut, 90-deg
  return oriImage, direction
 def preprocessingImage(self, source):
  #dapatkan ukuran gambar
  self.width  = source.shape[1]
  self.height = source.shape[0]
  #konversi gambar BGR ke HSV
  img = cv2.cvtColor(source, cv2.COLOR_BGR2HSV)
  #ambil nilai dari slider bar
  Low = np.array([self.LowH_T.value(), self.LowS_T.value(),
self.LowV_T.value()])
  High = np.array([self.HighH_T.value(),
self.HighS_T.value(), self.HighV_T.value()])
  Low2 = np.array([self.LowH_T_2.value(),
self.LowS_T_2.value(), self.LowV_T_2.value()])
  High2 = np.array([self.HighH_T_2.value(),
self.HighS_T_2.value(), self.HighV_T_2.value()])
  #filter untuk landing
  destination2 = cv2.inRange(img, Low2, High2)
  ret,destination2 = cv2.threshold(destination2,127,255,1)
  #filter untuk pipe tracking
  destination = cv2.inRange(img, Low, High)
  kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,
(5,5))
  destination = cv2.morphologyEx(destination,
cv2.MORPH_OPEN, kernel)
  destination = cv2.morphologyEx(destination,
cv2.MORPH_OPEN, kernel)
  destination = cv2.morphologyEx(destination,
cv2.MORPH_OPEN, kernel)
  destination = cv2.morphologyEx(destination,
cv2.MORPH_CLOSE, kernel)
  destination = cv2.morphologyEx(destination,
cv2.MORPH_OPEN, kernel)
  #kembalikan frame
  return destination , destination2
 def getSerialData(self, maxData):
  self.serData = str(self.ser.readline())
  self.serData = self.serData[2:-
6].split(";",maxsplit=maxData)
  for i in range(0,maxData,1):
   self.serData[i] = int(self.serData[i])
 def start(self):
  self.timer = QtCore.QTimer()
```

```
  self.timer.timeout.connect(self.nextFrame)
  self.timer.start(1000./self.fps)
def main():
    app = QtGui.QApplication(sys.argv)
    form = ExampleApp()
    form.show()
    app.exec_()
if __name__ == '__main__':
    main()
```

Kode utama pada Arduino Mega:

```
#include <Wire.h>
#include "IMU.h"
#include "Fuzzy.h"
#include "Motor.h"
void setup(){
 HWire.begin();
 pinMode(intPin, INPUT);
 digitalWrite(intPin, LOW);
 pinMode(myLed, OUTPUT);
 digitalWrite(myLed, HIGH);
 byte c = readByte(MPU9250_ADDRESS, WHO_AM_I_MPU9250);
 delay(1000);
 if (c == 0x73){
  MPU9250SelfTest(SelfTest);
  calibrateMPU9250(gyroBias, accelBias);
  initAK8963(magCalibration);  if (MagCal){
   magcalMPU9250(magBias, magScale);
   delay(2000);
  }
  delay(1000);
 }
 else{
  while (1) ;
 }
 counter = 0;
}
void loop(){
 if(Serial.available()){
  inbyte = Serial.readString();
 }
 else{
  int dir_B = getValue(inbyte,';',0).toInt();
  int speed_B = getValue(inbyte,';',1).toInt();
  int dir_Ki = getValue(inbyte,';',2).toInt();
  int speed_Ki = getValue(inbyte,';',3).toInt();
  int dir_Ka = getValue(inbyte,';',4).toInt();
  int speed_Ka = getValue(inbyte,';',5).toInt();
  int flag = getValue(inbyte,';',6).toInt();
```

```
  int controlMode = getValue(inbyte,';',7).toInt();
  if(controlMode==1){
    counter = 0;
    long duration, distance;
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance = (duration/2) / 29.1;
    if (flag==0){
    if(sinyalkontrol<0){sinyalkontrol=0;}
    if((distance)<450 && sinyalkontrol>-1 &&
sinyalkontrol<(256)){
    sinyalkontrol = sinyalkontrol+ kp * e0 + ki *(e1+e0) +
kd * (e1-e0);
    if(sinyalkontrol>255){sinyalkontrol=255;}
    e1 = e0;
    e0 = setpoin - distance;
    {motor_bawah(1,sinyalkontrol);}
    if(sinyalkontrol<0)
    {motor_bawah(1,0);}}}
  }
  if(controlMode==1){
   if (readByte(MPU9250_ADDRESS, INT_STATUS) & 0x01) {
    readIMU();
   }
   Now = micros();
   deltat = ((Now - lastUpdate) / 1000000.0f);
   lastUpdate = Now;
   sum += deltat;   sumCount++;
   MadgwickQuaternionUpdate(ax, ay, az, gx*PI/180.0f,
gy*PI/180.0f, gz*PI/180.0f,  my,  mx, mz);
   yaw   = atan2(2.0f * (q[1] * q[2] + q[0] * q[3]), q[0] *
q[0] + q[1] * q[1] - q[2] * q[2] - q[3] * q[3]);
   pitch = -asin(2.0f * (q[1] * q[3] - q[0] * q[2]));
   roll  = atan2(2.0f * (q[0] * q[1] + q[2] * q[3]), q[0] *
q[0] - q[1] * q[1] - q[2] * q[2] + q[3] * q[3]);
   pitch *= 180.0f / PI;
   yaw   *= 180.0f / PI;
   yaw   -= 1.03;
   roll  *= 180.0f / PI;
   eroll = roll-oldroll;
   giro_fisInput[0] = roll;
   giro_fisInput[1] = eroll;
   giro_fisOutput[0] = 0;
   giro_fisOutput[1] = 0;
   fis_evaluate_giro();
   motor_Bkiri(0,giro_fisOutput[0]);
```

```
    motor_Bkanan(0,giro_fisOutput[1]);
    mot_fisInput[0] = line2mid;
    mot_fisInput[1] = dir;
    mot_fisOutput[0] = 0;
    mot_fisOutput[1] = 0;
    fis_evaluate_mot();
    motor_Bkiri(0,mot_fisOutput[0]);
    motor_Bkanan(0,mot_fisOutput[1]);
    oldroll = roll;
    }
   }
  }
 }
 void readIMU(){
  readAccelData(accelCount);
  getAres();
  ax = (float)accelCount[0] * aRes;
  ay = (float)accelCount[1] * aRes;
  az = (float)accelCount[2] * aRes;
  readGyroData(gyroCount);
  getGres();
  gx = (float)gyroCount[0] * gRes;
  gy = (float)gyroCount[1] * gRes;
  gz = (float)gyroCount[2] * gRes;
  readMagData(magCount);
  getMres();
  if(!MagCal){
   magBias[0] = +470.;
   magBias[1] = +120.;
   magBias[2] = +125.;
  }
  mx = (float)magCount[0]*mRes*magCalibration[0] -
 magBias[0];
  my = (float)magCount[1]*mRes*magCalibration[1] -
 magBias[1];
  mz = (float)magCount[2]*mRes*magCalibration[2] -
 magBias[2];
  if(MagCal){
   mx *= magScale[0];
   my *= magScale[1];
   mz *= magScale[2];
  }
 }
 String getValue(String data, char separator, int index){
  int found = 0;
  int strIndex[] = { 0, -1  };
  int maxIndex = data.length()-1;
  for(int i=0; i<=maxIndex && found<=index; i++){
   if(data.charAt(i)==separator || i==maxIndex){
    found++;
```

```
   strIndex[0] = strIndex[1]+1;
   strIndex[1] = (i == maxIndex) ? i+1 : i;
  }
 }
 return found>index ? data.substring(strIndex[0],
strIndex[1]) : "";
}
```

Kode akuisisi IMU pada Arduino Mega:

```
void readAccelData(int16_t * destination){
 uint8_t rawData[6];
 readBytes(MPU9250_ADDRESS, ACCEL_XOUT_H, 6, &rawData[0]);
 destination[0] = ((int16_t)rawData[0] << 8) | rawData[1] ;
 destination[1] = ((int16_t)rawData[2] << 8) | rawData[3] ;
 destination[2] = ((int16_t)rawData[4] << 8) | rawData[5] ;
}
void readGyroData(int16_t * destination){
 uint8_t rawData[6];
 readBytes(MPU9250_ADDRESS, GYRO_XOUT_H, 6, &rawData[0]);
 destination[0] = ((int16_t)rawData[0] << 8) | rawData[1] ;
 destination[1] = ((int16_t)rawData[2] << 8) | rawData[3] ;
 destination[2] = ((int16_t)rawData[4] << 8) | rawData[5] ;
}
void readMagData(int16_t * destination){
 uint8_t rawData[7];
 if (readByte(AK8963_ADDRESS, AK8963_ST1) & 0x01) {
  readBytes(AK8963_ADDRESS, AK8963_XOUT_L, 7, &rawData[0]);
  uint8_t c = rawData[6];
  if (!(c & 0x08)) {
   destination[0] = ((int16_t)rawData[1] << 8) | rawData[0];
   destination[1] = ((int16_t)rawData[3] << 8) | rawData[2];
   destination[2] = ((int16_t)rawData[5] << 8) | rawData[4];
  }
 }
}
int16_t readTempData(){
 uint8_t rawData[2];
 readBytes(MPU9250_ADDRESS, TEMP_OUT_H, 2, &rawData[0]);
 return ((int16_t)rawData[0] << 8) | rawData[1] ;
}

void MadgwickQuaternionUpdate(float ax, float ay, float az,
float gx, float gy, float gz, float mx, float my, float mz){
 float q1 = q[0], q2 = q[1], q3 = q[2], q4 = q[3];
 float norm;
 float hx, hy, _2bx, _2bz;
 float s1, s2, s3, s4;
 float qDot1, qDot2, qDot3, qDot4;
 float _2q1mx;
```

```
 float _2q1my;
 float _2q1mz;
 float _2q2mx;
 float _4bx;
 float _4bz;
 float _2q1 = 2.0f * q1;
 float _2q2 = 2.0f * q2;
 float _2q3 = 2.0f * q3;
 float _2q4 = 2.0f * q4;
 float _2q1q3 = 2.0f * q1 * q3;
 float _2q3q4 = 2.0f * q3 * q4;
 float q1q1 = q1 * q1;
 float q1q2 = q1 * q2;
 float q1q3 = q1 * q3;
 float q1q4 = q1 * q4;
 float q2q2 = q2 * q2;
 float q2q3 = q2 * q3;
 float q2q4 = q2 * q4;
 float q3q3 = q3 * q3;
 float q3q4 = q3 * q4;
 float q4q4 = q4 * q4;
 norm = sqrt(ax * ax + ay * ay + az * az);
 if (norm == 0.0f) return;
 norm = 1.0f/norm;
 ax *= norm;
 ay *= norm;
 az *= norm;
 norm = sqrt(mx * mx + my * my + mz * mz);
 if (norm == 0.0f) return;
 norm = 1.0f/norm;
 mx *= norm;
 my *= norm;
 mz *= norm;
 _2q1mx = 2.0f * q1 * mx;
 _2q1my = 2.0f * q1 * my;
 _2q1mz = 2.0f * q1 * mz;
 _2q2mx = 2.0f * q2 * mx;
 hx = mx * q1q1 - _2q1my * q4 + _2q1mz * q3 + mx * q2q2 +
_2q2 * my * q3 + _2q2 * mz * q4 - mx * q3q3 - mx * q4q4;
 hy = _2q1mx * q4 + my * q1q1 - _2q1mz * q2 + _2q2mx * q3 -
my * q2q2 + my * q3q3 + _2q3 * mz * q4 - my * q4q4;
 _2bx = sqrt(hx * hx + hy * hy);
 _2bz = -_2q1mx * q3 + _2q1my * q2 + mz * q1q1 + _2q2mx * q4
- mz * q2q2 + _2q3 * my * q4 - mz * q3q3 + mz * q4q4;
 _4bx = 2.0f * _2bx;
 _4bz = 2.0f * _2bz;
 s1 = -_2q3 * (2.0f * q2q4 - _2q1q3 - ax) + _2q2 * (2.0f *
q1q2 + _2q3q4 - ay) - _2bz * q3 * (_2bx * (0.5f - q3q3 -
q4q4) + _2bz * (q2q4 - q1q3) - mx) + (-_2bx * q4 + _2bz *
q2) * (_2bx * (q2q3 - q1q4) + _2bz * (q1q2 + q3q4) - my) +
```

```
_2bx * q3 * (_2bx * (q1q3 + q2q4) + _2bz * (0.5f - q2q2 -
q3q3) - mz);
 s2 = _2q4 * (2.0f * q2q4 - _2q1q3 - ax) + _2q1 * (2.0f *
q1q2 + _2q3q4 - ay) - 4.0f * q2 * (1.0f - 2.0f * q2q2 - 2.0f
* q3q3 - az) + _2bz * q4 * (_2bx * (0.5f - q3q3 - q4q4) +
_2bz * (q2q4 - q1q3) - mx) + (_2bx * q3 + _2bz * q1) * (_2bx
* (q2q3 - q1q4) + _2bz * (q1q2 + q3q4) - my) + (_2bx * q4 -
_4bz * q2) * (_2bx * (q1q3 + q2q4) + _2bz * (0.5f - q2q2 -
q3q3) - mz);
 s3 = -_2q1 * (2.0f * q2q4 - _2q1q3 - ax) + _2q4 * (2.0f *
q1q2 + _2q3q4 - ay) - 4.0f * q3 * (1.0f - 2.0f * q2q2 - 2.0f
* q3q3 - az) + (-_4bx * q3 - _2bz * q1) * (_2bx * (0.5f -
q3q3 - q4q4) + _2bz * (q2q4 - q1q3) - mx) + (_2bx * q2 +
_2bz * q4) * (_2bx * (q2q3 - q1q4) + _2bz * (q1q2 + q3q4) -
my) + (_2bx * q1 - _4bz * q3) * (_2bx * (q1q3 + q2q4) + _2bz
* (0.5f - q2q2 - q3q3) - mz);
 s4 = _2q2 * (2.0f * q2q4 - _2q1q3 - ax) + _2q3 * (2.0f *
q1q2 + _2q3q4 - ay) + (-_4bx * q4 + _2bz * q2) * (_2bx *
(0.5f - q3q3 - q4q4) + _2bz * (q2q4 - q1q3) - mx) + (-_2bx *
q1 + _2bz * q3) * (_2bx * (q2q3 - q1q4) + _2bz * (q1q2 +
q3q4) - my) + _2bx * q2 * (_2bx * (q1q3 + q2q4) + _2bz *
(0.5f - q2q2 - q3q3) - mz);
 norm = sqrt(s1 * s1 + s2 * s2 + s3 * s3 + s4 * s4);
 norm = 1.0f/norm;
 s1 *= norm;
 s2 *= norm;
 s3 *= norm;
 s4 *= norm;
 qDot1 = 0.5f * (-q2 * gx - q3 * gy - q4 * gz) - beta * s1;
 qDot2 = 0.5f * (q1 * gx + q3 * gz - q4 * gy) - beta * s2;
 qDot3 = 0.5f * (q1 * gy - q2 * gz + q4 * gx) - beta * s3;
 qDot4 = 0.5f * (q1 * gz + q2 * gy - q3 * gx) - beta * s4;
 q1 += qDot1 * deltat;
 q2 += qDot2 * deltat;
 q3 += qDot3 * deltat;
 q4 += qDot4 * deltat;
 norm = sqrt(q1 * q1 + q2 * q2 + q3 * q3 + q4 * q4);
 norm = 1.0f/norm;
 q[0] = q1 * norm;
 q[1] = q2 * norm;
 q[2] = q3 * norm;
 q[3] = q4 * norm;
}
```

BAB V
PENUTUP

Dari penelitian yang dilakukan, didapatkan kesimpulan dan saran sebagai berikut :

5.1 Kesimpulan

Telah dirancang balon udara dengan kendali yang terdiri dari *image processing* untuk menjejak pipa dan sensor IMU dengan kontrol fuzzy untuk mempertahankan keseimbangan. Berdasarkan hasil pengujian dan analisis, dapat diperoleh kesimpulan sebagai berikut:

1. Berdasarkan data yang diperoleh, metode penjejakan pipa dengan menggunakan ROI dan contour finding mampu menjejak pipa dengan kesalahan sebesar 4,7% atau 0.19 cm.
2. Penggunaan logika *fuzzy* dapat diterapkan pada sistem kontrol pergerakan balon dan keseimbangan balon. Dengan *variable* yang digunakan berupa jarak *error* posisi dan derajat arah pipa untuk sistem motor pergerakan balon, serta data giroskop *roll* dan *error roll* untuk sistem keseimbangan balon udara.
3. Dengan menggunakan data yang didapat dari pengujian sensor giroskop, disimpulkan bahwa metode filter magdwick mampu memfilter data giroskop dengan kesalahan rata-rata sebesar 0,76% atau 0.3°.

5.2 Saran

Dari hasil penelitian yang dilakukan, untuk pengembangan berikutnya, disarankan beberapa hal berikut ini:

1. Penggunana Raspberry Pi 3 dirasa kurang cepat untuk metode yang dilakukan. Diharapkan pada pengembangan lebih lanjut dapat digunakan sistem minimum yang cepat.
2. Perlu digunakan sistem mikrokontroler tersendiri untuk memproses algoritma Madgwick sehingga tidak memberatkan sistem mikrokontroler utama.

DAFTAR PUSTAKA

[1] Anonim. "E-Ballon" <URL: http://www.eballoon.org/balloon/balloon-usages.html>. Mei. 2016

[2] Anonim. "Intro to Weather Balloons" <URL: http://www.highaltitudescience.com/pages/intro-to-weather-balloons>. Mei. 2016

[3] Skuza, J. R., Park, Y., & Kim, H. J. et al. "Feasibility study of cargo airship transportation systems powered by new green energy technologies". NASA Center for AeroSpace Information. 2014

[4] Elfes, A., Bueno, S. S., Bergerman, M., & Ramos, J. G., Jr. "A semi-autonomous robotic airship for environmental monitoring missions". In Proceedings of the IEEE international conference on robotics and automation. Leuven. Belgium. Mei. 1998

[5] Kanistras, K., Martins, G., & Rutherford, M. J., et al. "Survey of unmanned aerial vehicles (UAVs) for traffic monitoring". In Handbook of Unmanned Aerial Vehicles. Netherlands. 2014

[6] Yang, Y., Wu, J., & Zheng, W. "Design, modeling and control for a stratospheric telecommunication platform". Acta Astronautica. 80(November–December). 181–189. 2012

[7] Ramos, J. G., Jr., Carneiro de Paiva, E., & Azinheira, J. R., et al. "Autonomous flight experiment with a robotic unmanned airship". In IEEE international conference on robotics and automation. Seoul. Korea. 2001

[8] de Paiva, E. C., Bueno, S. S., & Gomes, S .B. V., et al. "A control system development environment for AURORA's semi-autonomous robotic airship". In 1999 IEEE international conference on robotics and automation. Detroit. USA. 1999

[9] Anonim, "Explanation of GPS/GNSS Drift" <URL: https://sites.aces.edu/group/crops/precisionag/Publications/Timely%20Information/GPS-GNSS_Drift.pdf>. Mei. 2016

[10] D. Jeronimo, R. Alcacer dan F. C. Alegria. "Line Following and Ground Vehicle Tracking by an Autonomous Aerial Blimp". Institute for Systems and Robotics Lisboa. 2016

[11] G. Pang dan H. Liu. "Evaluation of a low-cost MEMS accelerometer for distance measurement". *Journal of Intelligent and Robotic System* 30. 2001

[12] K. Seifert dan O. Camacho. "Implementing Positioning Algortihms Using Accelerometers". Freescale Semiconductor. 2007

[13] A. Noth, S. Fux dan S. Bouabdallah. "Inertial Measurement Unit" <URL: http://sky-sailor.ethz.ch/docs/Development_of_a_Plannar_Low_Cost_Inertial_Measurement_Unit_for_UAVs and MAVs_(Fux2008).pdf>. Januari. 2016

[14] Anonim. "hyperphysic.phy-astr.gsu.edu". <URL: http://hyperphysic.phy-astr.gsu.edu/hbase/gyr.html. Mei. 2016

[15] S. O. Madgwick. "An Efficient Orientation Filter for Inertial and Inertial/Magnetic Sensor Arrays". *Report x-io and University of Bristol (UK).* 2010

[16] W. R. Hamilton. "Elements of Quaternions". Longmans: Green & Company. 1866

[17] R. Munir. "Pengolahan Citra Digital dengan Pendekatan Algoritmik Informatika". Bandung. 2004

[18] S. I. Syafi'i. "Open Computer Vision (OpenCV)". 2011

[19] L. Shapiro dan G. C. Stockman. "Computer Vision". Prentice Hall. 2001

[20] T. Pavlidis. "Algorithms for Graphic and Image Processing". 1882.

[21] D. Davies, D. Bathurst dan R. Bathurst. "The Telling Image: The Changing Balance Between Pictures and Words in Technological Age". Clarendon Pr. 1990

[22] T. Sutojo, E. Mulyanto dan V. Suhartono. "Kecerdasan Buatan". Yogyakarta: Andi Offset. 2011

[23] S. Kusumadewi. "Artificial Intelligence (Teknik dan Aplikasinya)". Yogyakarta: Graha Ilmu. 2003

[24] R. AS. "Modul Pembelajaran Rekayasa Perangkat Lunak". Bandung. 2011

BIODATA PENULIS

Dion Hayu Fandiantoro dilahirkan di Surabaya, pada tanggal 26 Juli 1994 dari pasangan Bapak Budiono dan Ibu Sri Rahayu. Penulis adalah anak kedua dari dua bersaudara. Penulis menyelesaikan pendidikan dasar di SDN Sidotopo Wetan II Surabaya dilanjutkan dengan pendidikan menengah di SMPN 2 Surabaya dan SMAN 5 Surabaya. Pada tahun 2012 Penulis memulai pendidikan di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Selama kuliah Penulis aktif membantu penyelenggaraan kegiatan dan aktif sebagai asisten laboratorium Elektronika Dasar dan praktikum Elektronika maupun sebagai koordinator praktikum rangkaian elektronika pada semester ganjil 2015-2016.

Email: dionhayu@gmail.com