Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – 184801

Rancang Bangun Robot Keseimbangan Dua Roda Untuk Mengikuti Objek Berbasis Visi Komputer

Rizal Aulia Ramadhan
NRP 07111745000071

Dosen Pembimbing
Ronny Mardiyanto, S.T., M.T.,Ph.D.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR - 184801

Rancang Bangun Robot Keseimbangan Dua Roda Untuk Mengikuti Objek Berbasis Visi Komputer

Rizal Aulia Ramadhan
NRP 07111745000071

Dosen Pembimbing
Ronny Mardiyanto, S.T., M.T.,Ph.D.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – 184801

Design Two Wheels Balancing Robot for Following Object Based on Computer Vision

Rizal Aulia Ramadhan
NRP 07111745000071

Advisor
Ronny Mardiyanto, S.T.,M.T.,Ph.d.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Electrical Technology Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Rancang Bangun Robot Keseimbangan Dua Roda Untuk Mengikuti Objek Berbasis Visi Komputer" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

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

Surabaya, Juli 2019

Rizal Aulia Ramadhan
07111745000071

[Halaman ini sengaja dikosongkan]

RANCANG BANGUN ROBOT KESEIMBANGAN DUA RODA UNTUK MENGIKUTI OBJEK BERBASIS VISI KOMPUTER

TUGAS AKHIR

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

Menyetujui :

Dosen Pembimbing I,

Ronny Mardiyanto, S.T.,M.T.,Ph.D.
NIP. 198101182003121003

SURABAYA
JULI, 2019

[Halaman ini sengaja dikosongkan]

RANCANG BANGUN ROBOT KESEIMBANGAN DUA RODA UNTUK MENGIKURI OBJEK BERBASIS VISI KOMPUTER

Nama : Rizal Aulia Ramadhan
Pembimbing I : Ronny Mardiyanto, S.T.,M.T.,Ph.D.

Perkembangan dunia robot berkembang pesat dari setiap tahunnya. Salah satu contohnya adalah perkembangan pada robot keseimbangan. Robot keseimbangan beroda dua merupakan suatu robot mobile yang memiliki dua buah roda pada sisi kiri dan kanannya yang tidak akan seimbang apabila tanpa adanya kontroller. Telah banyak penelitian dari berbagai universitas di dunia yang membuat robot keseimbangan roda dua yang mampu membawa beban di atas badan robot dengan posisi yang seimbang, tentu saja dari hal tersebut robot keseimbangan ini sewajarnya dapat dikembangkan lebih baik lagi dari sisi penggunaannya sebagai contoh dapat dikembangkan untuk menjadi robot pelayan di restoran, robot asisten di bidang industri, dan robot patroli untuk keamanan. Robot keseimbangan beroda dua mempunyai banyak kelebihan jika dibandingkan dengan jenis robot beroda lainnya, seperti mampu mempertahankan posisi seimbang ketika mengangkat barang meskipun medan yang dilalui memiliki medan yang tidak datar. Oleh karena itu, pada tugas akhir ini dilakukan perancangan dari robot keseimbangan ini untuk mengikuti objek secara otomatis dengan menggunakan kamera. Sensor yang digunkan pada pembuatan robot ini yaitu memadukan antara sensor *accelerometer* dan *gyroscope* kemudian data dari kedua sensor tersebut diolah dengan menggunakan filter komplementeri dengan mikrokontroller Arduino Nano sehingga menghasilkan robot yang seimbang. Pada robot ini digunkan Mini PC Raspberry pi beserta kamera untuk mengolah gambar dengan metode pencarian objek berdasarkan warna. Didapatkan data dari pengolahan data sensor MPU6050 menggunakan filter komplementeri dengan koefisien alpha 0,996 dengan rata – rata error 0,145^{o} dan didapatkan pengaturan parameter PID yang terbaik adalah Kp = 28, Ki = 1,5, dan Kd = 25. Sedangkan kecepatan maksimum yang dapat dilakukan oleh robot ketika mengikuti sebuah objek adalah 0,2 m/s.

Kata kunci: Arduino Nano, *Accelerometer*, *Gyroscope*, Filter Kalman, Visi Komputer.

[Halaman ini sengaja dikosongkan]

DESIGN TWO WHEELS SELF BALANCING ROBOT FOR FOLLOWING OBJECT BASED ON COMPUTER VISION

Name ***: Rizal Aulia Ramadhan***
1st Advisor ***: Ronny Mardiyanto, S.T., M.T. Ph.D.***

The development of the robot world is growing rapidly from every year. One example is the development of a balance robot. The two-wheeled balance robot is a mobile robot that has two wheels on the left and right sides that will not be balanced if there is no controller. There have been many studies from various universities in the world that make two-wheeled balance robots capable of carrying loads on a robotic body with a balanced position, of course from this balance robot can naturally be developed better from the side of its use as an example can be developed to be servant robots in restaurants, assistant robots in industry, and patrol robots for security. Of course if this can be realized it will greatly help various jobs carried out by humans. Therefore, in this final project the design of this balance robot is done to automatically follow objects using a camera. The sensor used in making this robot is combining the accelerometer and gyroscope sensors and then the data from the two sensors is processed using complementary filters with an Arduino Nano microcontroller to produce a balanced robot. In this robot, a Mini PC Raspberry pi is used along with a camera to process images using color-based object search methods. Obtained data from MPU6050 sensor data processing using complementary filters with alpha coefficient 0.996 with an average error of 0.145o and obtained the best PID parameter settings are Kp = 28, Ki = 1.5, and Kd = 25. While the maximum speed that can be done by a robot when following an object is 0.2 m/s.

Keywords**:** Arduino Nano, *Accelerometer, Gyroscope, Kalman Filter, Computer Vision.*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala syukur dan puji hanya bagi Tuhan Yesus Kristus, oleh karena anugerahnya yang melimpah,kemurahan dan kasih setianya akhirnya penulis dapat menyusun dan menyelesaikan buku tugas akhir dengan judul: "**Rancang Bangun Robot Keseimbangan Dua Roda Untuk Mengikuti Objek Berbasis Visi Komputer**" maksud dan tujuan penulis menyelesaikan buku tugas akhir ini adalah untuk memenuhi persyaratan kelulusan Strata I bidang studi Elektronika, Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember, Surabaya.

Penulis ingin mengucapkan terima kasih kepada kedua orang tua yang selalu memberikan dukungan dan doa yang tulus tiada henti maka penulis dapat menyelesaikan buku tugas akhir. Terima kasih kepada bapak Ronny Mardiyanto S.T.,M.T.,Ph.D. yang telah bersedia meluangkan waktunya untuk memberikan masukan, arahan dan membimbing penulis dalam pembuatan tugas akhir ini. Penulis juga mengucapkan banyak terima kasih kepada semua pihak yang telah membantu baik secara langsung maupun tidak langsung dalam proses penyelesaian Tugas Akhir ini.

Penulis menyadari dan memohon maaf atas segala kekurangan pada buku tugas akhir ini. buku ini memang masih jauh dari sempurna. Oleh karena itu penulis mengharapkan saran dan kritik yang membangun dari pembaca. Akhir kata, semoga buku tugas akhir ini dapat memberikan manfaat dalam pengembangan keilmuan dikemudian hari.

Surabaya, 3 Juli 2019

Penulis

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi pada bidang robotika telah membuat kualitas hidup manusia semakin tinggi, saat ini perkembangan teknologi robotika telah mampu meningkatkan kualitas maupun kuantitas dari berbagai aspek kehidupan mulai dari bidang industri, pendidikan, dan hiburan. Terlebih pada era industri 4.0 ini semua hal dituntut dengan serba cepat, oleh sebab itu pada sektor industri kebutuhan teknologi robotika akan sangat diperlukan untuk meningkatkan produktivitas. Menurut data dari kementrian tenaga kerja Indonesia pada tahun 2019 akan ada 150 bidang pekerjaan yang akan diambil alih oleh robot[1].

Teknologi robotika memiliki beberapa kelebihan antara lain kemampuan yang sangat cepat dan konsistensi yang baik jika dibandingkan dengan tenaga manusia. Namun, selain dituntut cepat dan konsisten di era saat ini robot dituntut juga memiliki kecerdasan buatan. Salah satu cara meningkatkan tingkat kecerdasan sebuah robot ialah dengan cara menambah sensor, metode control, dan kecerdasan buatan pada robot tersebut. Salah satu robot yang dapat dikembangkan adalah *self balancing* robot.

Robot keseimbangan beroda dua merupakan suatu robot mobile yang memiliki dua roda di sisi kanan dan kirinya yang tidak akan seimbang apabila tanpa adanya controller. Menyeimbangkan robot ini memerlukan suatau perangkat hardware yang baik agar mendapatkan nilai keseimbangan yang baik sebagai masukan untuk metode control agar robot dapat mempertahankan posisi tegak lurus terhadap permukaan bumi[2]. Robot ini mempunyai beberapa kelebihan jika dibandingkan robot dengan jenis beroda yang lain yaitu memiliki dimensi yang lebih *compact*, serta mampu menyeimbangkan beban yang berada diatas robot walaupun dalam berbagai jenis medan kemampuan ini tidak dimiliki oleh robot beroda tiga dan beroda empat.

Selain memperhatikan nilai keseimbangan dari sebuah robot keseimbnagan agar robot ini memiliki kecerdasan dalam hal membedakan suatu gambar, penggunaan robot ini juga dapat dijadikan sebagai alat untuk mendeteksi dan mengikuti suatu objek tertentu. Pengembangan robot ini dalam hal computer visi masih sangat jarang ditemukan terlebih di Indonesia. Padahal dari pengembangan ini, robot keseimbangan dapat

digunakan untuk beberapa hal seperti robot asisten, robot patrol, dan *smart segwey*. Metode pengolahan gambar adalah bagian dari visi komputer, dimana pada metode ini setiap bentuk pengolahan sinyal dimana input adalah berupa gambar seperti foto maupun video. Sedangkan output dari metode pengolahan gambar ini dapat berupa gambar lain atau sejumlah karakteristik dan parameter yang berkaitan dengan gambar[3].

Dari uraian latar belakang tersebut maka akan dibuat suatu rancangan robot keseimbangan dua roda yang dapat digunakan untuk mengikuti sebuah objek dengan menggunakan kamera sebagai visibilitas dari robot dan dengan menggabungkan antara kemampuan robot dan visi komputer agar dapat dihasilkan suatu robot yang mampu menyeimbangkan dirinya sendiri dengan baik serta dapat mengikuti objek dengan tingkat akurasi yang baik.

1.2 Perumusan Masalah

Permasalahan yang akan dibahas dalam tugas akhir ini adalah:

1. Membuat desain robot yang dapat mempertahankan posisi tegak lurus pada permukaan bumi meskipun mendapatkan gangguan dari luar.
2. Bagaimana mengetahui pengaruh control PID pada robot keseimbangan dua roda.
3. Membuat pengolahan gambar dengan metode identifikasi objek melalui perbedaan warna.
4. Membuat komunikasi antara robot dan mini pc sehingga robot dapat menerima data koordinat pada subuah gambar yang telah diolah.

1.3 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah:

1. Menghasilkan suatu rancangan robot dengan hardware yang baik untuk mendukung kinerja robot.
2. Dapat mengetahui parameter Kp, Ki, dan Kd yang tepat agar robot dapat berjalan dengan baik meskipun dapat gangguan dari laur
3. Menghasilkan gambar yang diingikan agar robot dapat membedakan sebuah objek dengan objek yang lain.
4. Mendapatkan data koordinat pixel yang tepat agar robot dapat mengetahui arah dari suatu objek.

1.4 Batasan Masalah

1. Pengujian dan analisa robot keseimbangan dilakukan pada bidang datar.
2. Pengujian pengolahan gambar dilakukan dengan openCV dan Mini Pc Raspberry pi.
3. Pengujian untuk mendeteksi suatu objek dilakukan pada satu objek saja berupa bola.

1.5 Metodologi

Dalam penulisan tugas akhir ini digunakan metodologi sebagai berikut:

1. Studi Literatur Sensor
   Tahap ini meliputi pengumpulan dasar teori tentang robot kesimbangan dua roda yang dikontrol dengan kontrol PID, serta metode pengolahan gambar pengenelan objek dengan warna.

2. Perancangan Sistem *Hardware*
   Tahap ini meliputi perancangan alat yang digunakan yaitu rancangan *body* robot serta rangkain desain rangkain elektronika pada pcb.

3. Perancangan Sistem Sensor MPU 6050
   Sensor yang digunakan untuk menyeimbangkan robot adalah sensor MPU 6050 yang bertujuan untuk mengetahui sudut kemiringan dari badan robot yang nantinya akan di filter dengan metode *kalman filter*.

4. Perangancan Sistem Kontrol PID
   Setelah sistem mampu membaca kemiringan badan robot maka, selanjutnya adalah perancangan system kontrol PID untuk menentukan parameter Kp, Ki dan Kd agar dapat dihasilkan robot yang mampu berdiri dan menyeimbangkan tubuhnya.
5. Perangancan Sistem Pendeteksi Objek
   Pada tahapan ini system pendeteksi objek dilakukan dengan menggunakan *library* OpenCV dengan metode pendeteksi warna pada objek yang sudah ditentukan. Alat yang digunakan untuk memproses gambar tersebut menggunakan *Mini PC Raspberry pi*.

6. Pengujian Sistem Keseluruhan
   Tahapan pengujian sistem akan dilakukan dalam tahap yaitu :
   1. Pengujian pembacaan arah dari suatu objek melalui komunikasi serial antara *Raspberry pi* dan *Arduino Nano* yang merupakan sistem dari robot.
   2. Pengujian gerak robot secara keseluruhan untuk mengikuti gerakan dari suatu objek yang telah ditentukan.
7. Penulisan Laporan Tugas Akhir
   Setelah dilakukan segala percobaan dan pengambilan data, maka akan dilakukan penulisan laporan tugas akhir yang final.

1.6. Sistematika Penulisan

Laporan tugas akhir ini disusun dengan sistematika sebagai berikut:

BAB I Pendahuluan akan membahas latar belakang, tujuan, masalah, metode penelitian, sistematika, pembahasan, dan relevansi.

BAB II Teori penunjang dan literatur yang berkaitan bagi tugas akhir ini.

BAB III Perencanaan alat baik *hardware* maupun *software.*

BAB IV Hasil percobaan dan temuan – temuan alat dibahas pada bab ini.

BAB V Penutup berisi kesimpulan yang diperoleh dari tugas akhir ini, serta saran-saran untuk pengembangan lebih lanjut.

1.7. Relevansi

Hasil dari tugas akhir ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Dapat digunakan sebagai robot keseimbangan yang mampu mengikuti objek dengan akurasi yang baik.
2. Sebagai dasar penelitian lebih lanjut, agar dapat dikembangkan.

BAB 2
TINJAUAN PUSTAKA DAN DASAR TEORI

Suatu penelitian memerlukan teori-teori yang sudah ada sebelumnya untuk dikaji lebih dalam memperkuat argumen penulis. Teori tersebut digunakan untuk membantu penulis dan sebagai dasar dalam membuat suatu penelitian.

Pada bab ini terdapat teori dasar yang menjadi landasan untuk merumuskan dan menyelesaikan masalah yang akan dibahas pada penelitian ini. Pada bagian ini terdapat tinjauan pustaka tentang komponen yang akan digunakan untuk membuat alat pada penelitian ini.

2.1. Tinjauan Pustaka

Pada penelitian tugas akhir ini mengambil beberapa tinjauan pustaka dari penelitian – penelitian yang ada keterkaitan dengan proyek tugas akhir yang dilakukan. Berikut adalah tinjaun pustaka yang digunakan.

2.1.1 *Self Balancing Robot* Menggunakan PID Kontroller

Pada peper ini menjelaskan penggunaan kontroller PID untuk menyeimbangkan *body robot* keseimbangan. Kesamaan dari peper ini dengan tugas akhir yang dilakukan adalah metoda yang digunakan adalah metode kontroller PID tuning *trial and error* [4]. Pada peper ini hasil dari metode tersebut dapat disimpulkan cukup bagus karena ketika robot dapat gangguan dari luar sistem sacara langsung menstabilkan posisi robot dengan time respon kurang dari 1,5 s.

2.1.2 *A Two – Wheeled Self Balancing Robot With The Fuzzy PD Control Method*

Pada peper ini menjelaskan bahwa selain menggunakan *tuning trial and error* untuk mencari parameter PID yang terbaik agar robot keseimbangan dapat menyeimbangkan badan robot saat beroperasi, metode *fuzzy logic* juga sangat efektif untuk digunakan. Dalam proses logika fuzzy, ada beberapa tahapan yang dilakukan yaitu proses fuzifikasi, penentuan basis aturan, dan proses defuzifikasi. Proses pertama yang dilakukan adalah fuzifikasi yaitu pemetaan masukan – masukan kedalam himpunan keanggotaan, kemudian yang kedua adalah penentuan basis aturan,

penentuan basis aturan inilah yang menentukan konstanta parameter PID. Kemudian tahapan terakhir adalah proses defuzifikasi yang merupakan proses proses pengubahan nilai keluaran fuzzy menjadi nilai keluaran tegas (*crisp*) [5]. Hasil yang didapatkan dari metode ini cukup baik karena robot mampu merespon gangguan dengan baik.

2.1.3 *Car Finding Lane Lines Using Hough Lines Transform Method*

Pada peper ini menjelaskan proses pendeteksian garis jalan raya saat pengguna menjalankan mobil. Metode yang digunakan untuk mendeteksi garis pada peper ini menggunakan *library* Open CV yaitu *Hough Lines Transform* [6]. Metode ini hampir sama dengan yang digunakan pada projek tugas akhir ini, hanya saja pada metode ini parameter yang digunakan yaitu (x,y). Sedangkan pada tugas akhir ini menggunakan metode *Hough Circle Transform* sebab yang akan dideteksi adalah garis tepi pada objek yang berbentuk lingkaran sehingga parameter yang dibutuhkan adalah (x,y,R). Dimana nilai R yang dimaksud adalah radius.

2.1.4 *Sensor Fusion Algorithm by Complementary Filter for UAV with Low-cost IMU*

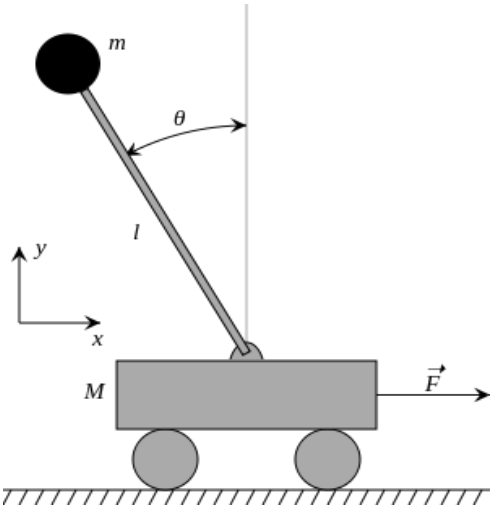
Pada paper ini dikembangkan sebuah algoritma penggabungan menggunakan teknik complementary filter untuk mengetahui perkiraan perilaku Quadrotor UAV menggunakan sensor MEMS IMU yang murah. Sensor yang digunakan adalah MPU6000 dan HMC5883L. Sensor – sensor ini digunakan hanya untuk mengetahui perilaku UAV pada saat terbang sehingga dapat mengoptimalkan kontrol posisi dari Quadrotor UAV. Sedangkan pada tugas akhir ini sensor MEMS IMU yang digunakan ialah MPU6050 [7]. Sensor ini digunakan untuk mengetahui besarnya kemiringan dari badan robot saat berdiri tegak di atas permukaan bumi. Sama halnya dengan peper tersebut nilai kelauaran sensor MPU6050 akan difilter sehingga nilai error dari keluaran sensor tidak jauh dari sudut yang sebenarnya. Tentu saja fungsi untama yang ingin didapat ialah mengoptimalkan kontrol posisi pada self balancing robot.

2.2. Dasar Teori

Pengerjaan tugas akhir ini memerlukan dasar teori sabagai metode – metode untuk pengerjaannya. Berikut adalah beberapa dasar teori dan metode yang digunakan dalam pembuatan robot keseluruhan.

2.2.1 Pendulum Terbalik

Robot keseimbangan merupakan pengembangan dari model pendulum terbalik yang diletakkan di atas kereta beroda. Pendulum terbalik merupakan model sistem yang tidak stabil dimana titik berat sistem pendulum ini berada di atas titik tumpunya. Ide dasar untuk mebuat pengendalian roda searah dengan arah jatuhnya bagian atas sebuah robot. Apabila proses tersebut dapat terlaksana maka robot tersebut dapat setimbang.

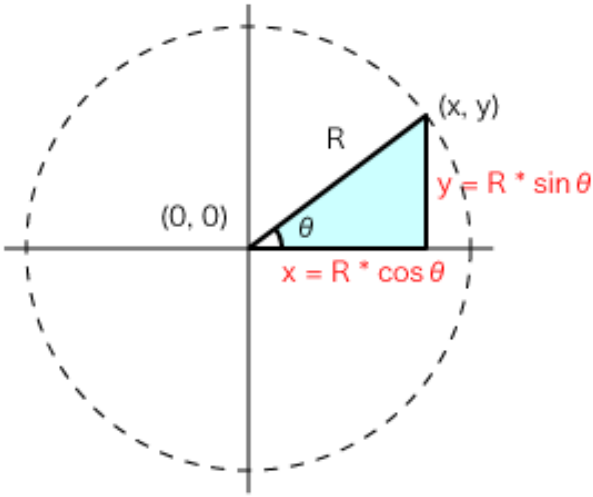


Gambar 2. 1 Pendulum Terbalik[8]

Pada robot keseimbangan ini, gaya yang digunakan untuk menyeimbangkan dihasilkan dari putaran roda. Putaran roda ini berasal dari torsi yang dihasilkan oleh motor. Sedangkan nilai dari kemiringan posisi robot akan ditentukan oleh sensor *accelerometer* yang memiliki 3 sumbu dan sendor *gyroscope* yang memiliki 3 sumbu juga.

2.2.2 Hough Circle Transform

Hough Circle Transform adalah suatu metode pengolahan gambar pada OpenCV. Dimana fungsi dari metode ini adalah untuk mendeteksi lingkaran pada suatu objek. Jika pada hough line transform hanya terdapat dua parameter yaitu (x,y), pada metode ini terdapat tigas parameter yang menentukan dalam mendeteksi garis pada lingkaran yaitu (a,b,R) dimana (a) adalah nilai tengah dari koordinat x dan (b) adalah nilai tengah dari koordinat y, sedangkan R adalah radius dari lingkaran yang dideteksi [9].



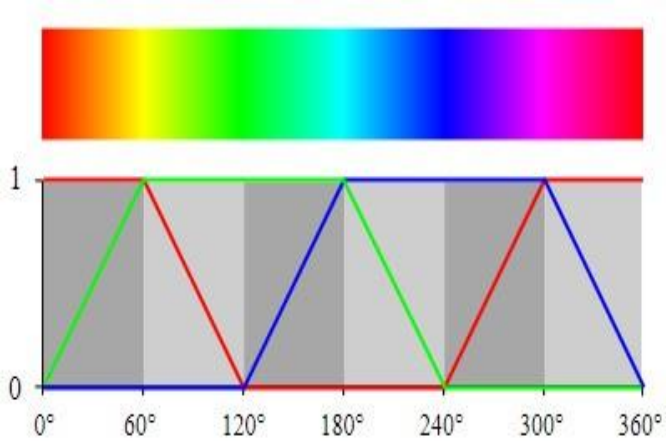
Gambar 2. 2 Metode Hough Circle Transform[9]

2.2.3 Konversi RGB ke HSV

HSV (*hue, saturation, value*) adalah dua representasi alternatif dari model warna RGB yang dirancang pada tahun 1970-an oleh peneliti grafika komputer untuk lebih menyelaraskannya dengan cara membuat atribut penglihatan manusia yang memandang warna. Dalam model ini, warna dari masing-masing *hue* diatur dalam irisan radial, sekitar poros tengah dari warna-warna netral yang berkisar hitam di bagian bawah untuk di bagian putih atas. HSV memodelkan representasi

dengan cara mencampurkan cat dari berbagai warna bersama-sama dengan saturasi dimensi menyerupai berbagai warna cerah cat berwarna dan nilai dimensi yang menyerupai campuran cat-cat dengan jumlah yang bervariasi dari cat hitam atau putih. Adapun pengertian dari *hue, saturation,* dan *value* sebagai berikut :

- *Hue* : Komponen *hue* digambarkan sebagai gradasi warna dari merah, hijau, biru dan kembali ke merah, jika dihubungkan dengan komponen warna RGB maka *hue* bias dibagi menjadi 6.



Gambar 2. 3 Grafik warna RGB ke nilai *hue* [10]

- Saturasi : Saturasi menunjukan seberapa berwarnanya sebuah warna

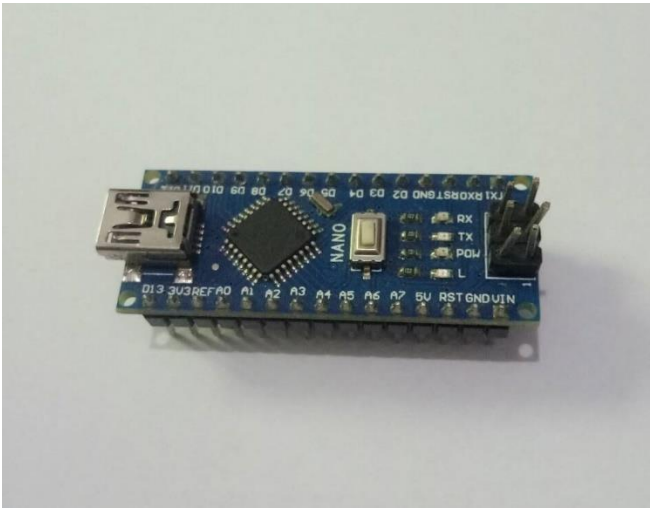
  Saturasi = delta/max(RGB)

- Value : Value berfungsi sebagai nilai kecerahan/*brightness*

  Value = max(RGB)/255

2.2.4 Arduino Nano

Arduino Nano adalah salah satu papan pengembangan mikrokontroler yang berukuran kecil, lengkap dan mendukung penggunaan breadboard. Arduino Nano diciptakan dengan basis mikrokontroler ATmega328. Arduino Nano kurang lebih memiliki fungsi yang sama dengan Arduino Duemilanove, tetapi dalam paket yang berbeda. Arduino Nano tidak menyertakan colokan DC berjenis Barrel Jack, dan dihubungkan ke komputer menggunakan port USB Mini-B. Arduino Nano dirancang dan diproduksi oleh perusahaan Gravitech[11].



Gambar 2. 4 Bentuk fisik arduino nano

Arduino nano memiliki 14 pin *digital output* (6 pin digunakan untuk PWM) dan 8 pin *analog output* tegangan operasional yang digunakan pada board mikrokontroller ini adalah sebesar 5V, sedangkan tegangan input yang dianjurkan adalah 7 – 12 V. selain itu arduino nano memiliki memory flash sebesar 32 KB dan clock speed 16 MHz. dimensi dari arduino nano ini terbilang ukurannya sangat kecil yaitu sebesar 1,85 cm x 4,3 cm.

2.2.5 Raspberry pi 3

Raspberry pi adalah sebuah komputer yang berukuran sangat kecil bahkan dimensi dari komputer kecil ini hanya sekitar 10 cm. Komputer kecil ini dapat dihubungkan dengan beberapa perangkat lain yaitu TV atau layar komputer, keyboard dan mouse. Perangkat ini adalah komputer kecil yang mumpuni, dapat digunakan untuk proyek elektronik dan dapat melakukan banyak hal layaknya PC dekstop. Selain itu alat ini juga mampu memutar video beresolusi tinggi. Raspberry pi *Foundation* merupakan yayasan nirlaba yang pertama kali mengembangkan produk ini. Tujuan awal diproduksinya raspberry pi adalah untuk digunakan oleh orang dewasa dan anak – anak di seluruh dunia untuk belajar pemrograman digital. Raspberry pi diproduksi awal pada tahun 2012, raspberry mempunyai 2 model awal yaitu model A dan model B [12].

Gambar 2. 5 Bentuk fisik Raspberry pi 3

Raspberry pi dapat dijalankan dengan beberapa OS yang mendukung linux contohnya adalah Raspbian. Namun raspberry dapat juga diinstall dengan OS linux lainnya yang mendukung python sebagai bahasa pemrograman utama selain C/C++. Raspberry mempunyai spesifikasi menggunakan SoC Broadcom BCM2837, CPU 4× ARM Cortex-A53, 1.2GHz, dan GPU Broadcom VideoCore IV. Selain itu board ini juga dilengkapi dengan wi-fi dan bluetooth.

2.2.6 Webcam Logitech C170

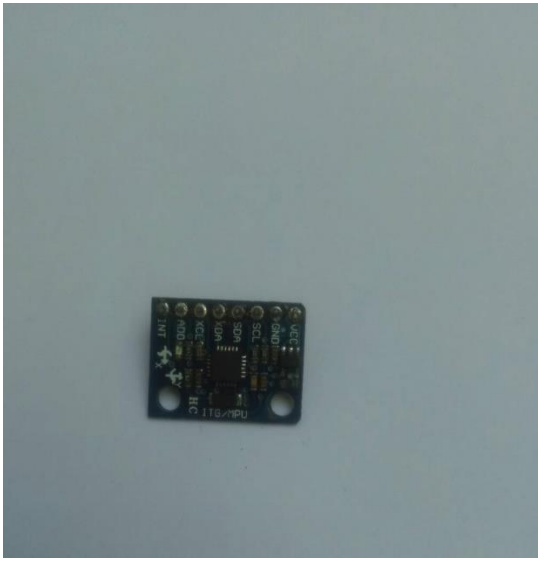
Untuk dapat mengambil gambar pada robot ini diperlukan kelengkapan yang lain yaitu dengan menggunakan kamera. Kamera yang digunakan adalah kamera webcam dengan merk dagang logitech C170. Webcam ini adalah salah satu jenis dari webcam tipe C series yang dikeluarkan oleh logitech. Kamera ini cukup baik digunakan untuk mendeteksi objek karena memiliki kualitas lensa yang cukup baik yaitu 5MP dengan resolusi kamera 640 x 480 dan memiliki fitur autofocus. Namun pada kamera ini kekurangan ada pada video quality yang hanya dibekali dengan kemampuan VGA.

Gambar 2. 6 Bentuk webcam logitech C170

Kamera ini sebenarnya mampu untuk menangkap gambar dengan resolusi yang cukup tinggi yaitu hingga 1024 x 768 pixel, namun resolusi sebesar ini tidak begitu disarankan sebab hasil video yang diambil kualitasnya tidak terlalu baik. Untuk dihubungkan ke perangkat lain webcam ini menggunakan USB 2.0 dengan kecepatan yang cukup tinggi.

2.2.7 MPU6050

MPU6050 merupakan perangkat 6 sumbu *Motion Tracking* pertama di dunia yang mengkombinasikan 3 sumbu *gyroscope* dan 3 sumbu *accelerometer*, dan sebuah *Digital Motion Processor* (DMP) dalam sebuah paket kecil berukuran 4x4x0.9mm [13]. berikut adalah gambar bentuk fisik dari sensor MPU 6050.

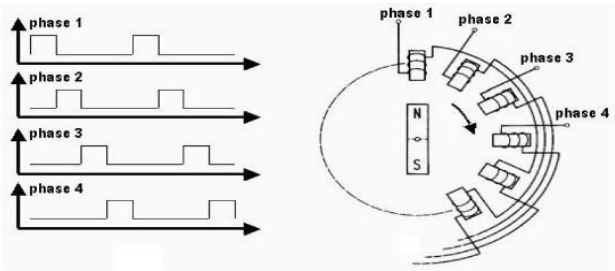


Gambar 2. 7 Sensor MPU 6050

Penggunaan sensor ini dapat kita temukan diberbagai macam alat elektronika, misalnya pada handphone, robot mobile, drone, helikopter dan peralatan canggih lainnya. Untuk penerapannya pada robot keseimbangan yaitu untuk mengetahui titik kemiringan dan percepatan badan robot diatas permukaan bumi. Nilai ini nantinya akan diproses secara komputasi agar mendapatkan nilai keseimbangan. Sensor ini adalah termasuk komponen utama dalam pembuatan robot keseimbangan dua roda ini.

2.2.8 Motor Stepper

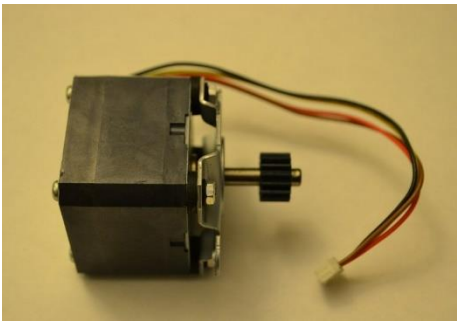
Motor stepper adalah salah satu jenis motor dc yang dikendalikan dengan pulsa-pulsa digital. Prinsip kerja motor stepper adalah bekerja dengan mengubah pulsa elektronis menjadi gerakan mekanis diskrit dimana motor stepper bergerak berdasarkan urutan pulsa yang diberikan kepada motor stepper[14].



Gambar 2. 8 Prinsip kerja motor stepper [14]

Pada robot *self balancing* ini jenis motor stepper yang digunakan adalah ML17A4, dimana jenis motor stepper ini adalah jenis motor stepper *hybrid* yang hanya memiliki 2 fasa. Adapun spesifikasi dari motor stepper ini sebagai berikut :

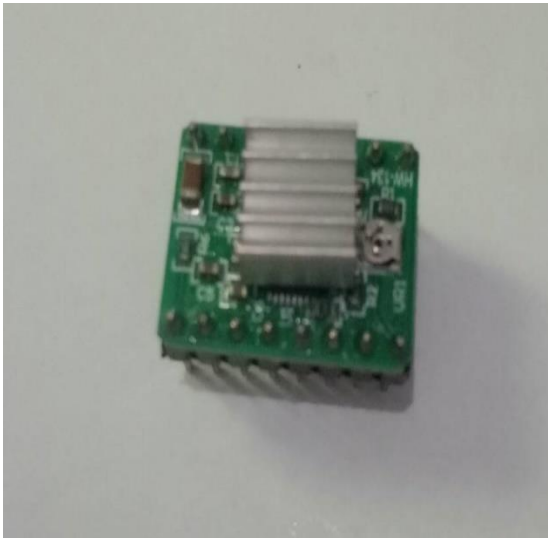
Torsi	: 4kg
StepAngle(degrees)	: 0.9 degree
Current / Phase	: 1.7A
Phase:2Type	: Hybrid



Gambar 2. 9 Motor stepper ML17A4

2.2.9 Driver Motor A4988

A4988 adalah driver motor stepper mikro lengkap dengan penerjemah bawaan untuk pengoperasian yang mudah. Produk ini tersedia dalam mode *full step*, *half step*, 1/4, 1/8 dan 1/16 untuk mengoperasikan motor stepper bipolar dari kemampuan drive output hingga 35 V dan ± 2 A. A4988 mencakup arus off-time tetap regulator, regulator dapat beroperasi dalam mode lambat atau campuran. Driver motor ini mudah untuk dikendalikan dan compatible dengan banyak mikrokontroller termasuk Arduino.



Gambar 2. 10 Driver motor A4988

Dalam operasi mikro stepper, kontrol A4988 secara otomatis memilih dalam mode saat ini dapat berupa mode lambat atau campuran. Dalam mode campuran, perangkat ini awalnya diatur ke bagian tetap dari peluruhan cepat di waktu henti dan lambat untuk sisa waktu henti. Skema kontrol peluruhan arus campuran menghasilkan pengurangan kebisingan motor yang dapat didengar, peningkatan akurasi langkah, dan konsumsi daya yang berkurang. Untuk mengatur step yang digunakan pada stepper motor dapat ditentukan dengan melihat table berikut.

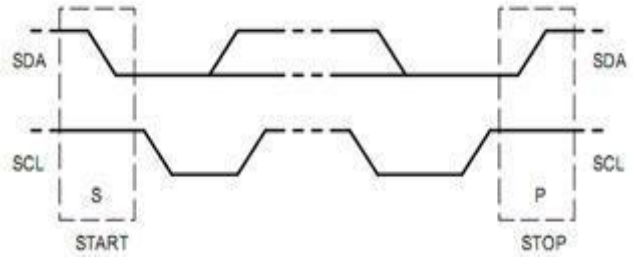
Tabel 2. 1 Pengaturan step pada driver motor A4988

MS1	MS2	MS3	Resolution
LOW	LOW	LOW	Full step
HIGH	LOW	LOW	Half step
LOW	HIGH	LOW	Quarter step
HIGH	HIGH	LOW	Eighth step
HIGH	HIGH	HIGH	Sixteenth step

2.2.10 I2C (Inter Integreted Circuit)

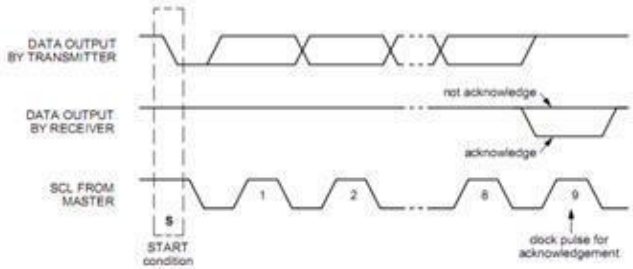
Inter Integrated Circuit atau sering disebut I^2C adalah standar komunikasi serial dua arah menggunakan dua saluran yang didisain khusus untuk meng*IR*im maupun menerima data. Sistem I^2C terdiri dari saluran SCL (*Serial Clock*) dan SDA (*Serial Data*) yang membawa informasi data antara I^2C dengan pengontrolnya. Piranti yang dihubungkan dengan sistem I2C Bus dapat dioperasikan sebagai *Master* dan *Slave*. *Master* adalah piranti yang memulai *transfer* data pada I^2C Bus dengan membentuk sinyal *Start*, mengakhiri *transfer* data dengan membentuk sinyal *Stop*, dan membangkitkan sinyal *clock*. *Slave* adalah piranti yang dialamati *master*.

Sinyal *Start* merupakan sinyal untuk memulai semua perintah, didefinisikan sebagai perubahan tegangan SDA dari "1" menjadi "0" pada saat SCL "1". Sinyal *Stop* merupakan sinyal untuk mengakh*IR*i semua perintah, didefinisikan sebagai perubahan tegangan SDA dari "0" menjadi "1" pada saat SCL "1". Kondisi sinyal *Start* dan sinyal *Stop* seperti tampak pada gambar berikut.



Gambar 2. 11 Kondisi sinyal *start* dan *stop* [15]

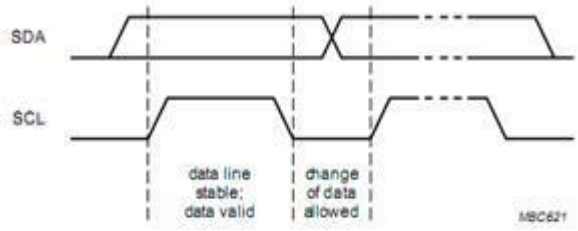
Sinyal dasar yang lain dalam I²C Bus adalah sinyal *acknowledge* yang disimbolkan dengan ACK Setelah transfer data oleh *master* berhasil diterima *slave*, *slave* akan menjawabnya dengan meng*IR*im sinyal *acknowledge*, yaitu dengan membuat SDA menjadi "0" selama siklus *clock* ke 9. Ini menunjukkan bahwa *Slave* telah menerima 8 bit data dari *Master*. Kondisi sinyal *acknowledge* seperti tampak pada Gambar berikut.



Gambar 2. 12 Sinyal ACK dan NACK[15]

Dalam melakukan *transfer* data pada I²C Bus, kita harus mengikuti tata cara yang telah ditetapkan yaitu:

- *Transfer* data hanya dapat dilakukan ketikan Bus tidak dalam keadaan sibuk.
- Selama proses transfer data, keadaan data pada SDA harus stabil selama SCL dalam keadan tinggi. Keadaan perubahan "1" atau "0" pada SDA hanya dapat dilakukan selama SCL dalam keadaan rendah. Jika terjadi perubahan keadaan SDA pada saat SCL dalam keadaan tinggi, maka perubahan itu dianggap sebagai sinyal *Start* atau sinyal *Stop*.



Gambar 2. 13 Transfer Bit pada I²C bus[15]

2.2.11 Kontroller PID

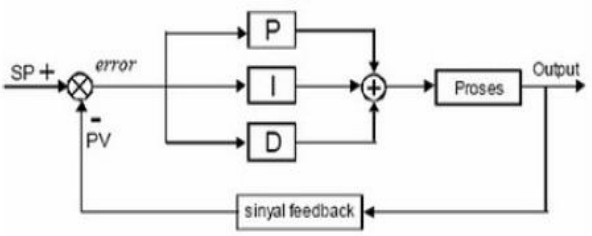
Pengendali PID adalah suatu sistem pengendali yang merupakan gabungan antara pengendali proporsional, integral, dan turunan (*derivative*). Karakteristik PID *controller* sangat dipengaruhi oleh kontribusi besar dari ketiga parameter, yaitu P, I, dan D. Proporsional *controller* (Kp) akan memberikan efek mengurangi waktu naik, tetapi tidak menghapus kesalahan keadaan tunak, Integral *controller* (Ki) akan memberikan efek menghapus keadaan tunak, tetapi berakibat memburuknya respons transien, diferensial *controller* (Kd) akan memberikan efek meningkatnya stabilitas sistem, mengurangi *over-shoot*, dan menaikan respons transfer.

Dalam pengaplikasian, masing-masing pengendali dapat berdiri sendiri atau dapat melakukan pengombinasian. Dalam perancangan sistem kontrol PID yang perlu dilakukan adalah mengatur parameter Kp, Ki atau Kd agar tanggapan sinyal keluaran sistem sesuai dengan yang diinginkan. Dalam waktu kontinyu, sinyal keluaran pengendali PID dirumuskan pada persamaan (1).

$$u(t) = Kp \times e(t) + Ki \times \int e(t)dt + Kd + \frac{de(t)}{dt} \qquad (1)$$

dengan:

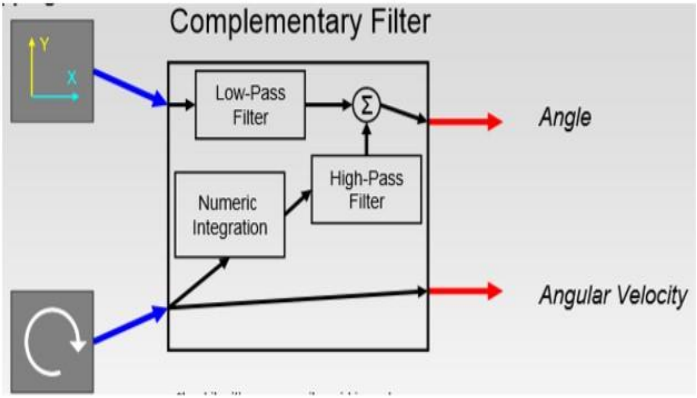
u(t) : sinyal keluaran pengendali PID
Kp : konstanta proporsional
Ki : konstanta integral
Kd : konstanta turunan
e(t) : sinyal kesalahan = eferensi – *output*



Gambar 2. 14 Diagram blok kontroller PID[16]

2.2.12 *Complementary Filter*

Complementary Filter adalah metode *filtering*, yang dapat berfungsi sebagai filter nilai sensor agar nilai tersebut memiliki noise yang kecil atau bahkan tidak ada, sehingga datanya akurat. Metode ini merupakan gabungan dari high-pass filter yang berasal dari output *gyroscope* yang terintegrasi dan *low-pass filter* yang berasal dari output *accelerometer* yang telah diolah. Metode ini juga membutuhkan nilai waktu konstan dan waktu sampling untuk perhitungan nilai alpha.



Gambar 2. 15 Diagram blok *complementary filter*[17]

Berikut adalah persamaan untuk mencari nilai dari suatu *complementary filter* :

x_angleC = a*(x_angleC + ang_gyro *dT) + (1-a) * (ang_accl) (2)

dimana:

x_angleC	= nilai complementary
a	= alpha
ang_gyro	= keluaran gyroscope
dT	= delta time
ang_accl	= keluaran accelerometer

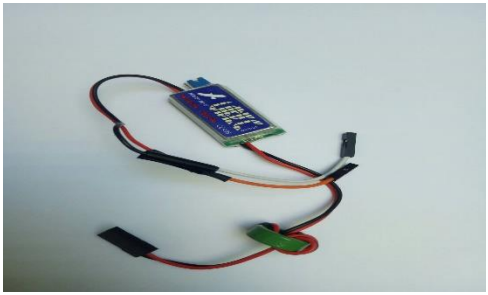
Sedangkan untuk mencari nilai alpha terdapat rumus sebagai berikut:

t =(a*dT)/(1-a)
a =t/(t+dT)
dimana:
a = koefisien filter atau alpha
dT = waktu sampling

Time constant (t) adalah lamanya waktu update sinyal keluaran dari complementary filter. Dengan menggunkan perhitungan *complementary filter*, hasil keluaran dari MPU 6050 akan semakin baik dan noise yang didapat dari sensor tersebut akan semakin kecil.

2.2.13 Ubec

UBEC (*Universal Battery Elimination Circuit*) adalah rangkaian elektronik yang mengambil daya dari *battery pack* atau sumber DC lainnya, dan menurunkannya ke level tegangan 5V atau 6V. Tegangan input maksimum tergantung pada spesifikasi UBEC. UBEC biasanya digunakan pada aplikasi yang memerlukan arus lebih tinggi, dan divais mampu mengirim daya dengan efisiensi hingga 92%.



Gambar 2. 16 Bentuk fisik UBEC

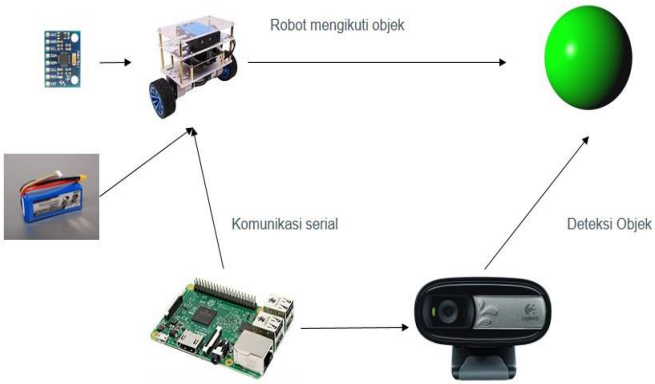
UBEC dipilih pada penggunaan robot ini adalah sebagai catu daya untuk mengaktifkan raspberry pi 3. Sebab kebutuhan tegangan adalah sebesar 5V. Sedangkan catu daya yang digunakan pada robot adalah 11V. UBEC dinilai lebiih efektif dan aman dari pada menggunakan DC *to* DC *converter* jenis lain.

BAB 3
PERANCANGAN SISTEM

Pada bab ini akan dibahas beberapa tahap perancangan sistem robot keseimbangan, yaitu diagram blok sistem, *hardware* robot, perancangan sirkuit elektronika, dan perancangan *software* sistem keseluruhan.

3.1 Diagram Sistem Keseluruhan

Pada diagram blok keseluruhan ini akan menjelaskan semua alur dari sistem robot keseimbangan untuk mengikuti mengikuti objek, pada kasus ini objek yang akan digunakan adalah bola berwarna hijau. Alur kerja dari sistem ini dapat dilihat pada gambar berikut.



Gambar 3. 1 Diagram sistem keseluruhan

Pada gambar 3.1 di atas dapat dilihat bahwa robot mendapat masukan dari MPU6050 berupa sudut kemiringan badan robot, tujuannya adalah untuk dapat mengontrol gerak robot agak tetap menjaga keseimbangan. Sedangkan baterai memperi supplai untuk kinerja dari robot tersebut. Setelah robot tersebut seimbang maka raspberry pi 3 akan mendeteksi objek berupa bola warna hijau dan mengirimkan koordinat, sehingga robot dapat mengikuti bola.

3.2 Diagram Blok Sistem Keseluruhan

Pada diagram blok sistem ini akan menjelaskan tentang sistem keseluruhan dari robot keseimbangan untuk mengikuti sebuah objek berbasis visi komputer.

Seperti yang digambarkan pada Gambar 3.1, perancangan sistem pada robot keseimbangan akan berjalan seperti pada diagram blok tersebut. Diagram blok secara umum terdiri pengujian sensor MPU6050 untuk mengetahui suatu kemiringan robot, kemudian dikontrol dengan PID kontroller, dan hasil keluaran dari PID tersebut digunakan untuk mengontrol PWM motor agar tetap menjaga titik kesetimbnagan pada robot. Raspberry pi 3 digunakan untuk memproses gambar yang ditangkap oleh kamera yang nantinya data dari pemrosesan tersebut akan dikirim ke Arduino Nano secara serial untuk mengontrol arah gerak robot sesuai pergerakan objek.

Perangkat keras pada sistem robot keseimbnagan terdiri dari :

Sudut Kemiringan Robot

Robot keseimbnagan beroda dua tentu saja tidak dapat menyeimbangkan dirinya sendiri tanpa dikontrol oleh perangkat tertentu sebab tubuh robot tidak dapat dikontrol untuk melawan gravitasi bumi. Namun untuk mencari nilai keseimbnagan pada sebuah robot tentu memiliki sudut kemiringan badan tersendiri.

Modul Sensor MPU 6050

Sensor yang digunankan MPU6050 dimana fungsinya untuk mendapatkan nilai *gyroscope* dan *accelerometer* pada posisi badan robot tegak lurus terhadap permukaaan bumi.

Arduino Nano

Arduino Nano digunakan untuk membaca nilai sensor MPU6050, memproses perhitungan pada PID kontroller dan hasil dari perhitungan secara matematis pada PID kontroller digunkan untuk mengontrol kecepatan putarandan arah pada motor stepper.

Motor Stepper

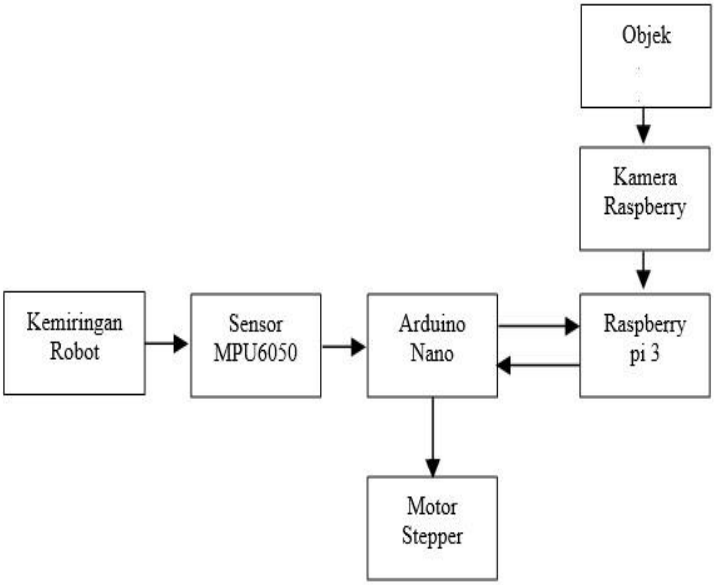
Motor stepper digunakan sebagai penggerak roda pada badan robot keseimbangan beroda dua yang bertujuan untuk menjaga badan robot agar tetap berdiri diatas permukaan bumi.

Raspberry Kamera

Raspberry kamera digunakan seolah –olah sebagai pengganti indra visual pada robot keseimbangan untuk menangkap gambar yang melintas didepan robot.

Raspberry pi 3

Raspberry pi 3 yang berfungsi sebagai komputer kecil ini digunakan untuk memproses gambar yang ditangkap oleh kamera kemudian diolah menjadi gambar lain dan didapatkan titik koordinat pixel pada gambar yang nantinya akan dikirim ke Arduino Nano secara serial.



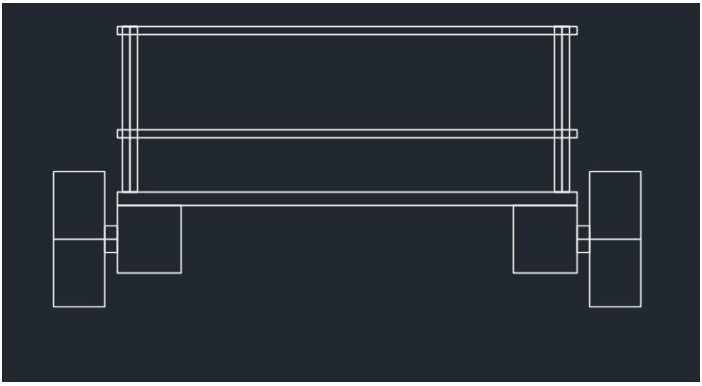
Gambar 3. 2 Diagram Blok Sistem

3.3 Perancangan *Hardware*

Pada perancangan *hardware* ini terdiri dari dua perancangan utama yaitu perancangan mekanik dari bentuk badan robot keseimbangan dan perancangan *cover* dan peletakan kamera raspberry pada bagian atas dari badan robot.

3.3.1 Perancangan Mekanik Robot

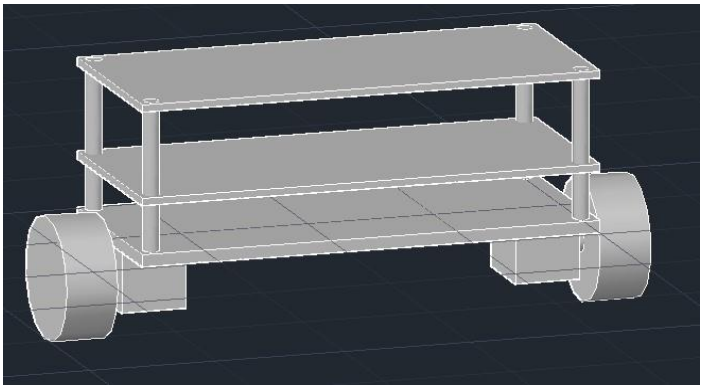
Pada perancangan mekanik badan robot ini menggunakan software design 3D Autocad 2019. Badan robot yang digunakan pada robot keseimbangan ini menggunkan bahan akrilik dengan tebal akrilik untuk bagian *body* bawah 5mm dan pada bagian *body* atas tebal yang digunakan adalah 3mm. *Body* robot bagian bawah dibuat sedikit lebih tebal dengan tujuan agar mendapatkan struktur badan robot yang kokoh untuk menopang bagian atas robot. Desain robot keseimbangan dapat dilihat pada gambar 3.2 dan gambar 3.3 berikut.



Gambar 3. 3 Sketsa badan robot tampak depan

Tabel 3. 1 Dimensi badan robot

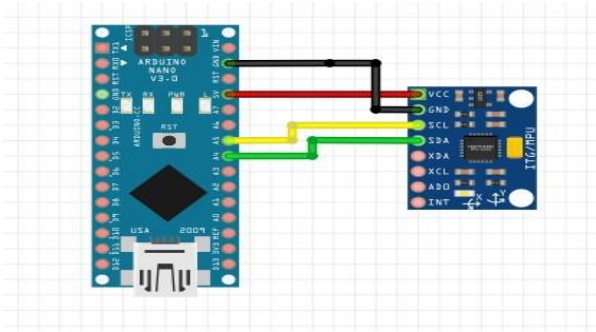
Panjang	180mm
Lebar	80mm
Tinggi	200mm
Diameter roda	50mm
Berat Asumsi	1kg



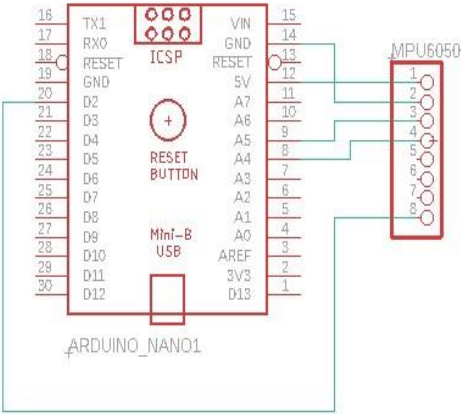
Gambar 3. 4 Desain utama dari badan robot keseimbangan

3.3.2 Perancangan Elektronika MPU6050

Perancanagn rangkaian elektronika pada MPU6050 ini dibuat dengan menggunakan desain *board* fritzing 0.9. Arduino nano sebagai mikrokontroller membaca data sensor yang ada pada MPU6050 melalui SCL yang terhubung dengan pin A5 dan SDA terhubung pada pin A4 sedangkan pin interrupt terhubung dengan pin timer pada Arduino Nano yaitu pada pin D2. Desain board dan skematik rangkaian dapat dilihat pada gambar 3.5 dan 3.6 berikut.



Gambar 3. 5 *wiring* Arduino Nano dengan MPU6050



Gambar 3. 6 Skematik Arduino Nano dengan MPU 6050

Pin-pin yang digunakan pada Arduino Nano dan MPU6050 pada sistem ini, yaitu:

- Pin A4 => SDA
- Pin A5 => SCL
- Pin D2 => INT
- 5V => Sumber tegangan untuk sensor MPU6050
- GND

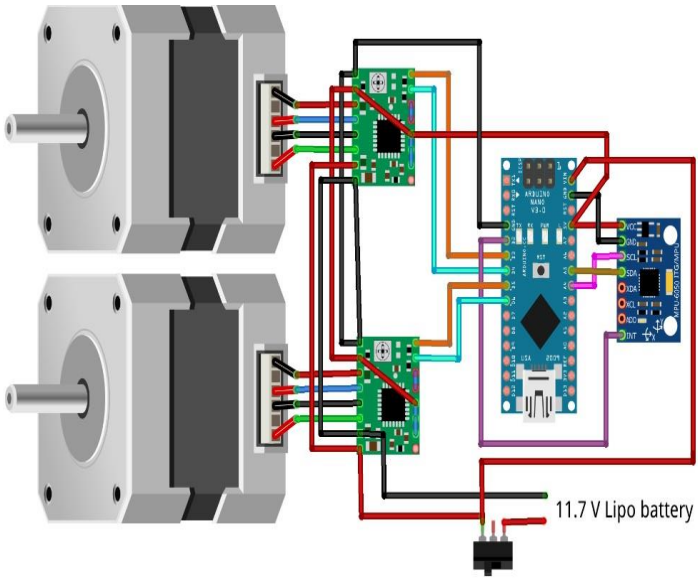
3.3.3 Perancangan Elektronika Keseluruhan

Untuk perancangan board keseluruhan menggunkan PCB dengan ukuran 12 x 8 cm. Perancangan ini dilakukan menggunkan software Eagle 9 dan fritzing. Pada rangkaian ini terdapat beberapa komponen yang digunkan untuk mengontrol kinerja dari *self balancing* robot. Berikut adalah beberapa komponen yang digunkan :

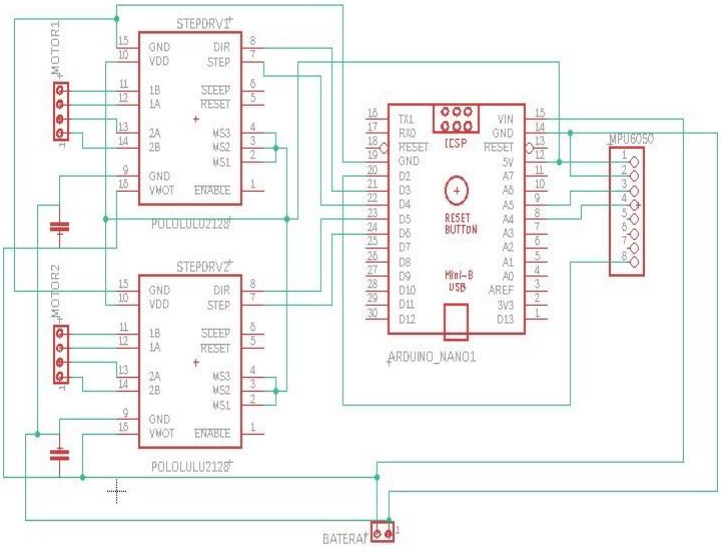
Tabel 3. 2 List komponen pada rangkaian kontrol robot

Jenis Komponen	***Quantity***
Arduino Nano	1
Driver motor a4988	2
MPU6050	1
Kapasitor 100uF	2
Terminal Screw	8

Pada rangkaian ini penggunaan kapasitor bertujuan untuk mengurangi noise pada motor stepper. *Wiring* rangkain dapat dilihat pada gambar 3.7 berikut.



Gambar 3. 7 *Wiring* rangkaian keseluruhan

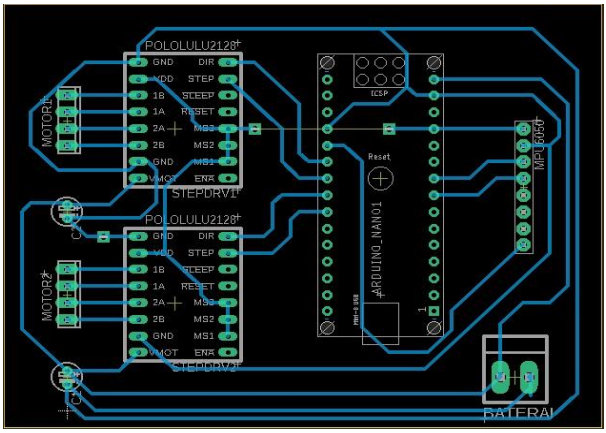


Gambar 3. 8 Skematik keseluruhan *self balancing robot*

Pin-pin yang digunakan pada Arduino Nano pada sistem ini, yaitu:

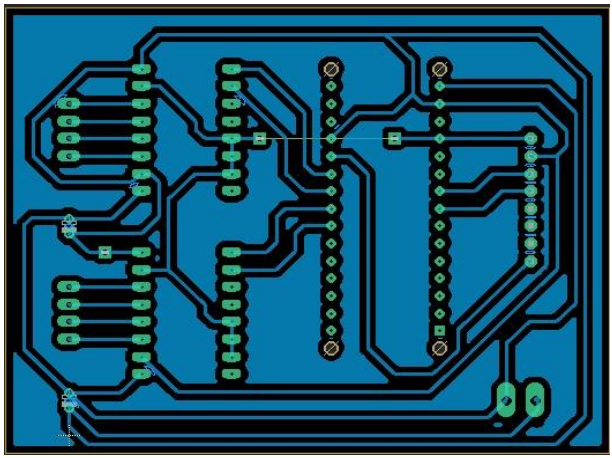
- Pin A4 => SDA
- Pin A5 => SCL
- Pin D2 => INT
- Pin D3 => ENA1
- Pin D4 => DIR 1
- Pin D5 => ENA2
- Pin D6 => DIR 2
- 5V => MS1, MS2, MS3
- 5V => Sumber tegangan untuk sensor dan *relay*
- GND

Hasil desain skematik pada gambar 3.8 tersebut kemudian diproses lagi menjadi bentuk board agar dapat dicetak menjadi board rangkaian *self balancing robot* yang sesungguhnya. Hasil dari desain dalam bentuk board dapat dilihat pada gambar 3.9 berikut.



Gambar 3. 9 Skematik dalam bentuk board *self balancing robot*

Setelah konversi dari skematik dalam board maka board akan dijadikan dalam bentuk siap untuk dicetak dengan cara mengatur layer pada board kemudian disesuaikan pengaturan DRC agar saat dicetak sesuai yang diinginkan. Tampilan board dapat dilihat pada gambar 3.10 berikut.



Gambar 3. 10 Desain *board* robot yang siap dicetak

3.4 Perancangan *Software*

Dalam pembuatan perangkat lunak ada beberapa program yang harus dibuat agar sistem pada robot dan pemrosesan gambar dapat berjalan dengan baik. Tahapan pembuatan tersebut adalah sebagai berikut:

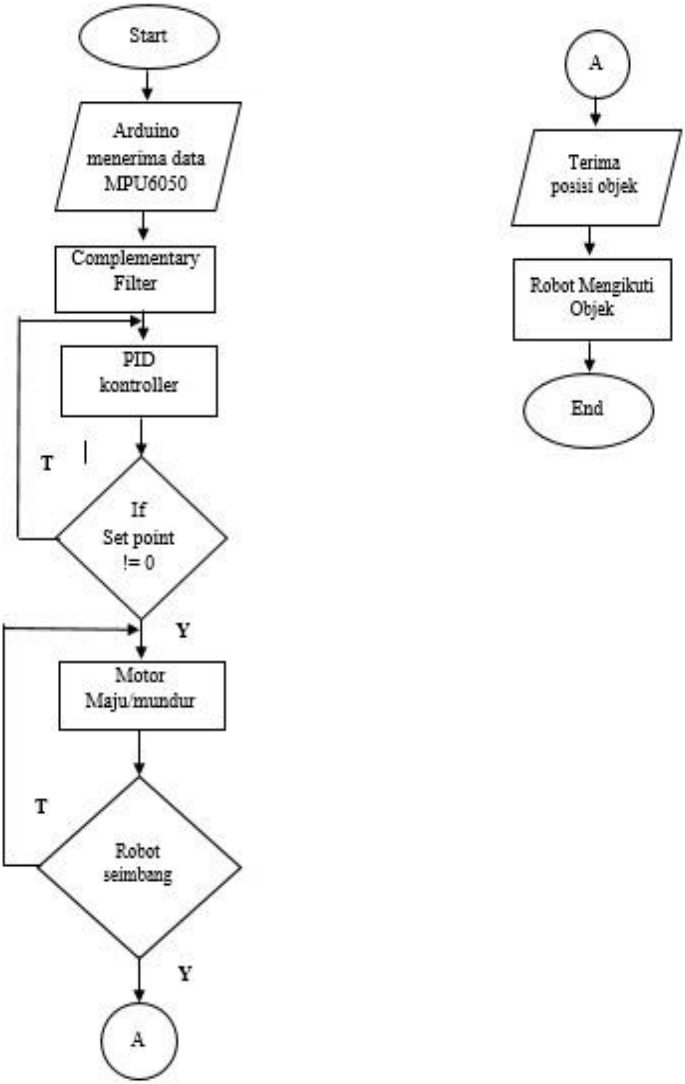
3.4.1 Pembuatan *Flowchart* Program

Flowchart merupakan bagan yang memperlihatkan urutan dan hubungan antar proses beserta instruksinya. Bagan ini dinyatakan dengan simbol. Dengan demikian setiap simbol menggambarkan proses tertentu. Sedangkan hubungan antar proses digambarkan dengan garis penghubung. *Flowchart*merupakan langkah awal pembuatan program. Dengan adanya *flowchart* urutan poses kegiatan menjadi lebih jelas.

Untuk pengolahan data dengan komputer, dapat dirangkum urutan dasar untuk pemecahan suatu masalah, yaitu:

- *START*: Berisi instruksi untuk persiapan perlatan yang diperlukan sebelum menangani pemecahan masalah.
- *READ*: Berisi instruksi untuk membaca data dari suatu peralatan.
- *PROCESS*: Berisi kegiatan yang berkaitan dengan pemecahan persoalan sesuai dengan data yang dibaca.
- *WRITE*: Berisi instruksi untuk merekam hasil kegiatan ke perlatan *output*.
- *END*: Mengakhiri kegiatan pengolahan.

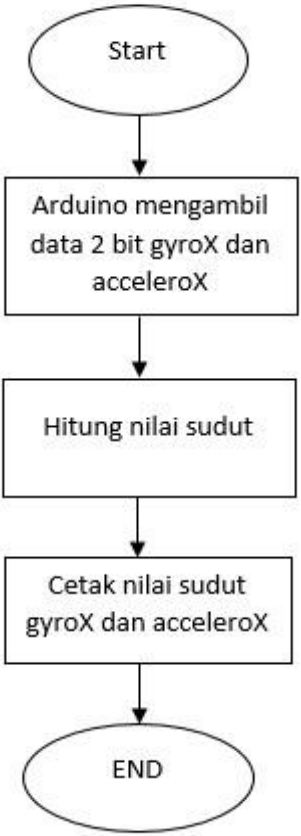
Flowchart program dari Tugas Akhir ini meliputi seluruh sistem jalannya alat ini. Sistem yang dimaksud adalah sistem umum secara keseluruhan. Berikut merupakan *flowchart* robot kesimbangan dua roda untuk mengikuti objek berbasis visi komputer dapat dilihat pada gambar 3.11.



Gambar 3. 11 *Flowchart* perancangan software keseluruhan

3.4.2 Perancangan Program Membaca Sudut MPU6050

Segmen program untuk mencari *set value* merupak bagian program yang terpisah dari program utama pada robot keseimbangan. Program ini dikhususkan hanya untuk menguji *hardware*. Robot diposisikan tegak lurus terhadap permukaan bumi. Kemudain program dijalankan untuk *scanning* nilai *accelerometer* ketika berdiri tegak lurus.



Gambar 3. 12 Flowchat program pembacaan sensor sudut

3.4.3 Perancangan Program *Pulse Generator*

Pada segmen program ini bertujuan untuk menentukan timer pada Arduino Nano untuk mengontrol pulsa dari motor stepper melalui interrupt timer. Ada dua jenis timer yang terdapat pada ATmega 328P. yaitu timer 8 bit dan timer 16 bit. Pada program ini akan menggunkan timer 8 bit. Timer 8 bit akan menghitung dari 0 – 255 dan jika telah overflow maka timer akan kembali ke nilai 0.

Perintah akses *subroutine timer* :

```
TCCR2A = 0;
TCCR2B = 0;
TIMSK2 |= (1 << OCIE2A);
TCCR2B |= (1 << CS21);
OCR2A = 39;
TCCR2A |= (1 << WGM21);
```

Pada program diatas perintah subroutine timer akan dijalankan setiap 20us. Dengan cara mengihitung prescaler clock dari ATmega328P. berikut perhitungan program diatas.

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk_{T2S}/(No prescaling)
0	1	0	clk_{T2S}/8 (From prescaler)
0	1	1	clk_{T2S}/32 (From prescaler)
1	0	0	clk_{T2S}/64 (From prescaler)
1	0	1	clk_{T2S}/128 (From prescaler)
1	1	0	clk_{T2S}/256 (From prescaler)
1	1	1	clk_{T2S}/1024 (From prescaler)

Gambar 3. 13 Pilihan *clock* pada ATmega328

Karena clock pada ATmega328P adalah 16MHz dan timer yang digunakan adalah 8 bit maka :

$$\frac{16MHz}{8} = 2Mhz$$

Kemudian jika nilai dari register CS21 bernilai 1 maka nilai dari pembagian diatas akan dibagi dengan nilai register CS21 :

$$\frac{1}{2MHz} = 0{,}5\ us$$

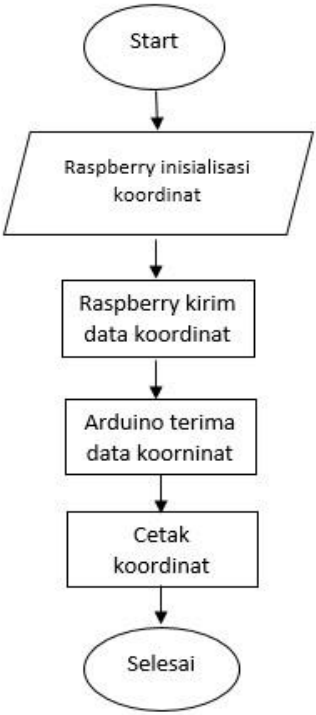
Untuk mendapatkan waktu perulangan pada interrupt selama 20us maka dicari nilai register dari OCR2A yang tepat yaitu dengan cara perhitungan sebagai berikut :

$$\frac{20\ us}{0{,}5\ us} = 40 =\gg -1 = 39$$

Karena waktu awal timer adalah dimulai dari 0 maka nilai hasil pembagi dikurang satu sehingga didapatkan nilai dari register OCR2A adalah 39.

3.4.4 Perancangan Program Komunikasi Serial

Pada sistem ini program komunikasi serial digunakan untuk mengirim data koordinat pixel pada program OpenCV dari raspberry pi3 ke arduino nano. Komunikasi serial menggunakan baudrate 115200 karena pada baudrate ini transfer data yang dilakukan lebih cepat jika dibandingkan dengan baudrate 9600. Berikut adalah *flowchart* program komunikasi serial.



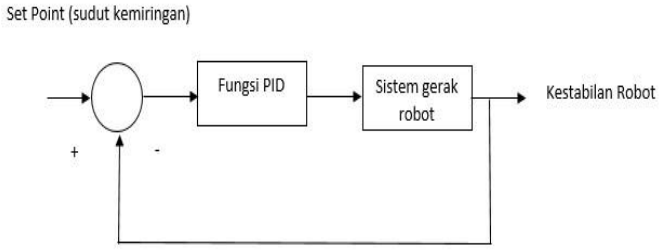
Gambar 3. 14 *Flowchart* program komunikasi serial

Perintah akses program :

```
If (Serial.available()){
    a = Serial.read();
    Serial.println(a);
}

If (receive_counter <= 25)receive_counter ++;
```

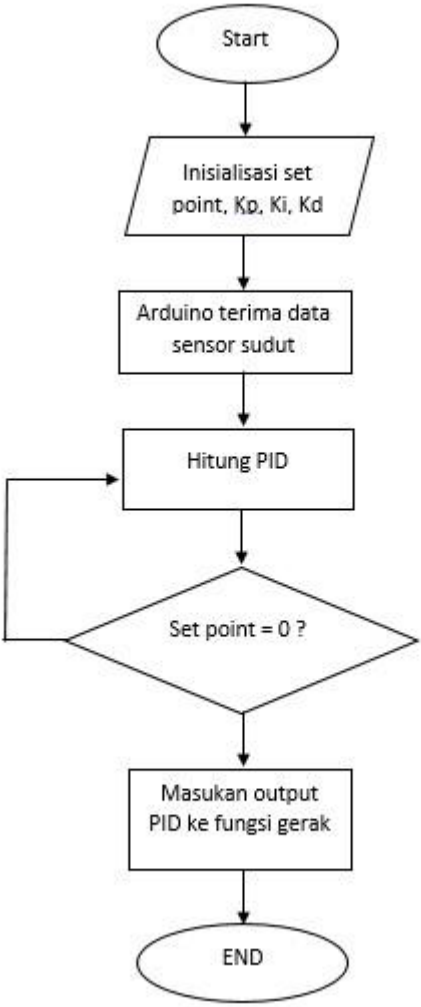
3.4.5 Perancangan Program Kontroller PID

Robot keseimbangan akan selalu seimbang apabila nilai set point dari sudut kemiringan badan robot selalu dalam kondisi 0. Untuk dapat mempertahankan posisi set point pada nilai 0 maka cara terbaik adalah dengan menggunakan kontroller PID. Metode PID yang digunakan pada robot ini adalah metode tuning PID. Kontroller PID membutuhkan masukan data untuk memproses sebuah sistem, data yang dimaksud adalah nilai sudut dari badan robot tersebut.



Gambar 3. 15 Diagram blok kontroller PID

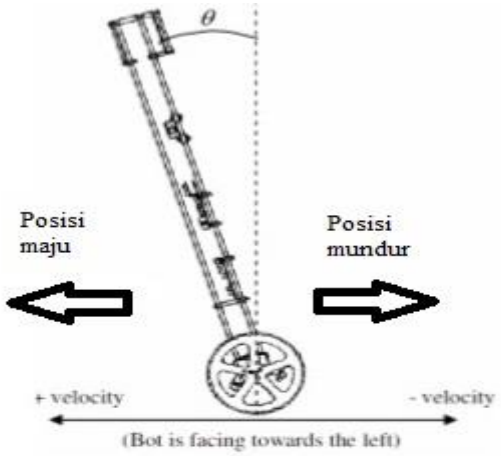
Kontrol PID di atas adalah jenis kontrol PID *close loop*, dimana jika nilai *output* dari kontrol PID tersebut terdpat error maka program akan kembali ke awal untuk mencocokan nilai *setpoint* dan nilai *error* sampai didapat nilai output yang diinginkan. Nilai keluaran tesebut nantinya akan dijadikan kontrol kecepatan dari motor stepper.



Gambar 3. 16 Flowchart kontroller PID

3.4.6 Perancangan Program *Control Calculation*

Perhitungan kontrol gerak robot didapat dari hasil output kontrol PID sebelumnya, dimana hasil output tersebut digunakan untuk menggerakan kedua sisi roda robot. Arah pergerakan robot disesuaikan dari data yang masuk melalui raspberry pi, data tersebut berisi koordinat dari objek. Untuk dapat menggerakan robot keseimbnagan maju, mundur, dan berbelok maka dibuatlah pengaturan sebagai berikut.



Gambar 3. 17 Arah gerak robot berdasarkan nilai *velocity*

Tabel 3. 3 Kalkulasi pergerakan robot keseimbangan

Arah Gerak	Motor Kanan	Motor Kiri
Maju	PID_output ditambah	Nilai PID ditambah
Mundur	PID_output dikurang	Nilai PID dikurang
Belok Kanan	PID_output dikurang	PID_output ditambah
Belok Kiri	PID_output ditambah	PID_output dikurang

Perintah akses program kalkulasi kontrol :

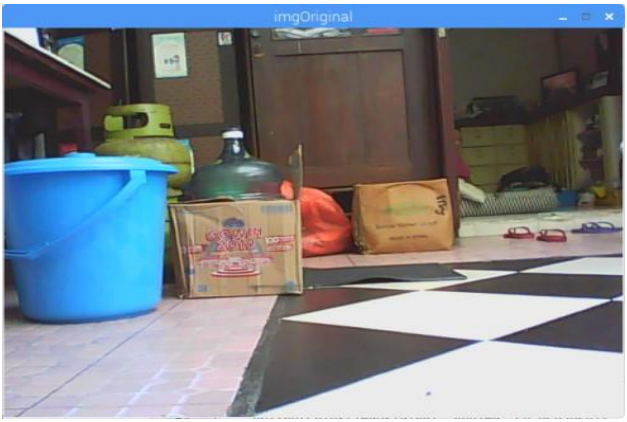
```
//nilai PID dijadikan nilai output motor

  pid_output_left = pid_output;
  pid_output_right = pid_output;

//kontrol gerak belok kanan
  if(a == 2){
    pid_output_left += turning_speed;
    pid_output_right -= turning_speed;
  }

//kontrol gerak belok kiri
  if(a == 3){
    pid_output_left -= turning_speed;
    pid_output_right += turning_speed;
  }

//kontrol gerak mundur
  if(a == 4){
    if(pid_setpoint > -2.5)pid_setpoint -= 0.05;
    if(pid_output > max_target_speed * -1)pid_setpoint -= 0.005;
  }

//kontrol gerak maju
  if(a == 1){
    if(pid_setpoint < 2.5)pid_setpoint += 0.05;
    if(pid_output < max_target_speed)pid_setpoint += 0.005;
  }

//kontrol robot diam
if(!(received_byte & B00001100)){
    if(pid_setpoint > 0.5)pid_setpoint -=0.05;
    else if(pid_setpoint < -0.5)pid_setpoint +=0.05;
    else pid_setpoint = 0;
  }
```

3.4.7 Perancangan Program Membuka Kamera

Dasar dari imaga processing pada komputer adalah menggunakan library OpenCV. Pada program yang dibuat library yang digunakan adalah OpenCV 3.2. Program ini adalah dasar untuk mendapatkan gambar dari suatu kamera, dimana dari hasil gambar yang ditangkap tersebut nantinya dapat diproses menjadi gambar dengan jenis yang lain sesuai metode yang digunakan. Pada robot kesimbnagan dua roda ini kamera yang digunakan beresolusi 640 x 480 *pixel*.

Perintah akses membuka kamera :

```
cv::VideoCapture capWebcam(0);
if (capWebcam.isOpened() == false)
{
std::cout << "error: Webcam connect unsuccessful\n";
return(0);
}
cv::Mat imgOriginal;                    // Input image
while (charCheckForEscKey != 27 && capWebcam.isOpened()) {
bool blnFrameReadSuccessfully = capWebcam.read(imgOriginal);
if (!blnFrameReadSuccessfully || imgOriginal.empty()) {
std::cout << "error: frame can't read \n";
break;    }
```



Gambar 3. 18 Hasil tangkapan gambar dari kamera

3.4.8 Perancangan Program Trackbar Pengaturan HSV

Nilai HSV (*Hue, Saturation, Value*) pada setiap warna memiliki nilai yang berbeda – beda. Pada tugas akhir ini beberapa warna dicoba untuk mengambil data dari setiap nilai warna. Maka dari itu dibuat trackbar pengaturan nilai set *hue, saturation,* dan *value.*

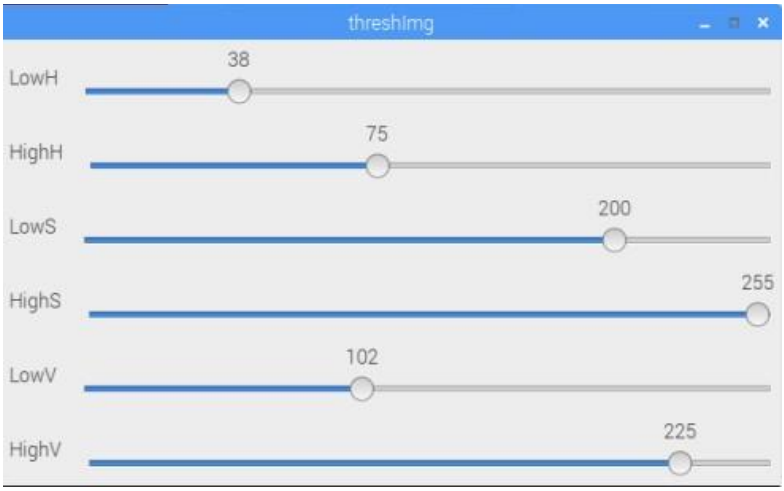
Perintah pembuatan trackbar :

Create trackbars in "threshImg" window to adjust according to object and environment.*/

cv::createTrackbar("LowH", "threshImg", &lowH, 179); //Hue (0 - 179)
cv::createTrackbar("HighH", "threshImg", &highH, 179);

cv::createTrackbar("LowS", "threshImg", &lowS, 255); //Saturation (0 - 255)
cv::createTrackbar("HighS", "threshImg", &highS, 255);

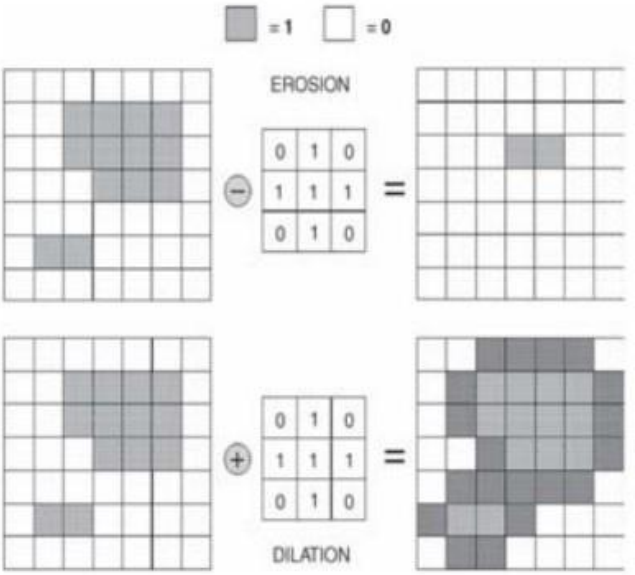
cv::createTrackbar("LowV", "threshImg", &lowV, 255); //Value (0 - 255)
cv::createTrackbar("HighV", "threshImg", &highV, 255);



Gambar 3. 19 *Trackbar* pengaturan nilai HSV

3.4.9 Perancangan Program Filter Erode dan Dilate

Pada segmen program ini hasil gambar masukan yang telah ditangkap oleh kamera akan difilter menggunakan filter erode dan dilate. Erode dan dilate adalah operasi pemrosesan gambar morfologis. Pemrosesan gambar morfologis pada dasarnya berkaitan dengan memodifikasi struktur geometris dalam gambar. Operasi ini terutama ditentukan untuk gambar biner, tetapi kita juga dapat menggunakannya pada gambar skala abu-abu. Erosi pada dasarnya menghapus lapisan piksel terluar dalam suatu struktur, di mana pelebaran menambahkan lapisan piksel tambahan pada suatu struktur. Gambar contoh dari filter dapat dilihat pada gambar 3.20 berikut.



Gambar 3. 20 Perbedaan gambar ketika difilter erode dan dilate

Pada tugas akhir ini filter erode dan dilate digunakan untuk mengurangi noise pada gambar yang telah dithreshold. Gambar yang difilter pada kasus ini yaitu berupa hasil threshold dari bola berwarna.

Perintah akses erode dan dilate :

```
//Blur Effect
cv::GaussianBlur(threshImg, threshImg, cv::Size(3, 3), 0);
// Dilate Filter Effect
cv::dilate(threshImg, threshImg, 0);
// Erode Filter Effect
cv::erode(threshImg, threshImg, 0);
```

3.4.10 Perancangan Program *Hough Circle Transform*

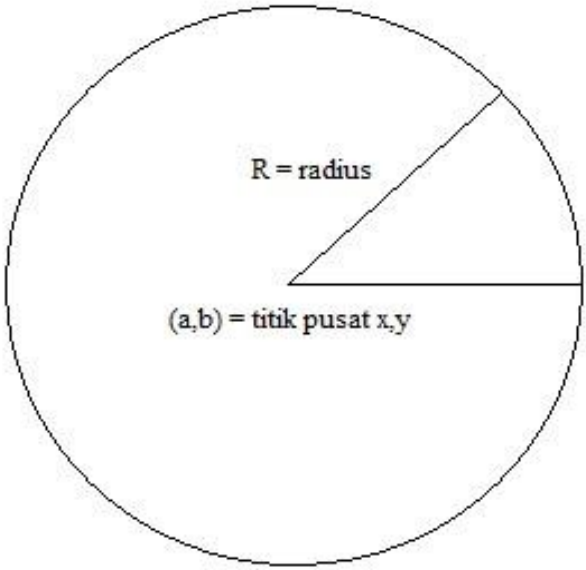
Metode *Hough Circle Transform* digunkan untuk mendeteksi garis pada bagian tepi lingkaran. Pada metode ini terdapat beberapa parameter yang digunakan. (a,b) adalah sebagai titik pusat (x,y) dan R adalah jari – jari lingkaran. Jika titik 2D (x, y) ditetapkan, maka parameter dapat ditemukan sesuai dengan rumus dibawah. Ruang parameter akan menjadi tiga dimensi, (a, b, r). Dan semua parameter yang memenuhi (x, y) akan terletak pada permukaan kerucut sudut kanan terbalik yang puncaknya berada pada (x, y, 0).

$$(x - a)^2 + (y - b)^2 = R^2 \qquad (3)$$

Perintah akses *Hough Circle Transform* :

```
// algorithm for detecting circles
cv::HoughCircles(threshImg,v3fCircles,CV_HOUGH_GRADIENT,2,thr
eshImg.rows / 4,100,50,10,800);
for (int i = 0; i < v3fCircles.size(); i++) {                // for each circle
 // x position of center point of circle
std::cout << "Ball position X = "<< v3fCircles[i][0]
// y position of center point of circle
<<",\tY = "<< v3fCircles[i][1]
// radius of circle
<<",\tRadius = "<< v3fCircles[i][2]<< "\n";
// draw small green circle at center of object detected
```

```
cv::circle(imgOriginal, cv::Point((int)v3fCircles[i][0],
(int)v3fCircles[i][1]),3, cv::Scalar(0, 255, 0),          CV_FILLED);

// draw red circle around object detected
cv::circle(imgOriginal,
cv::Point((int)v3fCircles[i][0], (int)v3fCircles[i][1]),
(int)v3fCircles[i][2],
cv::Scalar(0, 0, 255),
3);
```



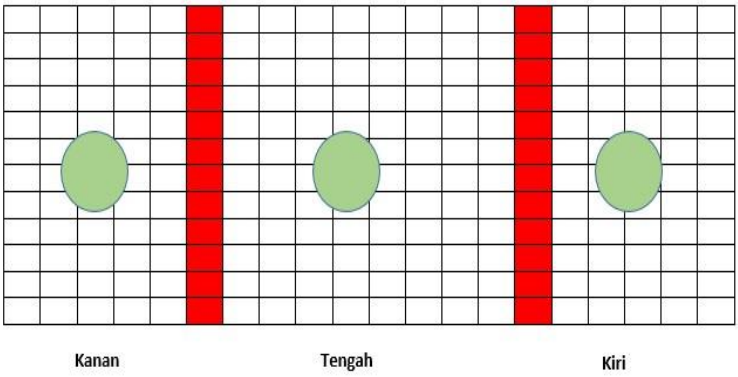
Gambar 3. 21 Tiga parameter utama pada *Hough Circle Transform*

3.4.11 Perancangan Program Pengkondisian Posisi Objek

Pada segmen ini pengkondisian posisi dari objek akan ditentukan dari nilai pembacaan titik pusat X. Data posisi yang telah didapat menggunakan *Hough Circle Transform* berupa nilai koordinat pixel. Nilai pixel tersebut diambil kemudian dibagi menjadi tiga bagian. Tiga bagian ini dibuat untuk membedakan apakah objek berada pada posisi tengah, kanan atau kiri terhadap posisi badan robot. Karena menggunakan kamera dengan resolusi 640 x 480 *pixel* maka pembagian dapat dilihat pada tabel berikut.

Tabel 3. 4 Kalkulasi pergerakan robot keseimbangan

Posisi Objek	***Range Pixel***
Kanan	0 – 240
Tengah	241 – 430
Kiri	430 – 680



Gambar 3. 22 Ilustrasi pengkondisian posisi objek

Perintah akses program :

```
  if (v3fCircles[i][0] > 241 && v3fCircles[i][0] < 430 &&
v3fCircles[i][2] < 40) {
        int i = 1;
        std::cout << "1"<<"\n";
        serialPutchar (fd, i) ;
        }
  if (v3fCircles[i][0] < 241) {
        int i = 2;
        std::cout << "2"<<"\n";
        serialPutchar (fd, i) ;
        }
   if (v3fCircles[i][0] > 430) {
        int i = 3;
        std::cout << "3"<<"\n";
        serialPutchar (fd, i) ;
        }
   if (v3fCircles[i][0] > 241 && v3fCircles[i][0] < 430 &&
v3fCircles[i][2] > 60) {
        int i = 4;
        std::cout << "4"<<"\n";
        serialPutchar (fd, i) ;
        }
    if (v3fCircles[i][0] > 190 && v3fCircles[i][0] < 450 &&
v3fCircles[i][2] > 40 && v3fCircles[i][2] < 60) {
        int i = 5;
        std::cout << "5"<<"\n";
        serialPutchar (fd, i) ;
        }
```

Pada program diatas posisi benda dibagi menjadi 3 bagian dan ditambah dua bagian program untuk maju, mundur, dan diam. Data pengkondisian ini akan dikirim ke Arduino Nano secara serial. Data yang masuk pada Arduino diproses untuk dijadikan perintah gerak dari robot keseimbangan dua roda.

BAB 4
HASIL SIMULASI DAN ANALISA

Pada Bab ini dibahas tentang pengujian dan analisa sistem yang telah dibuat. Pengujian dan analisa meliputi pengujian perangkat keras sistem dan pengujian perangkat lunak.

4.1 Gambaran Umum Pengujian Sistem

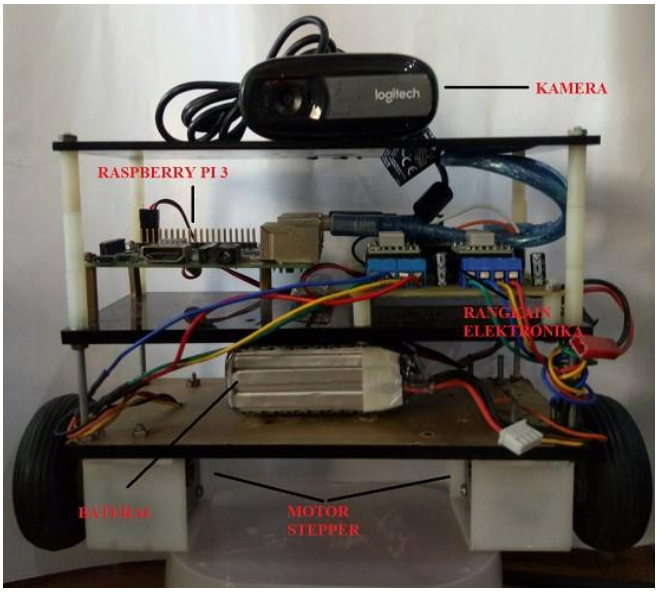
Pada pengerjaan Tugas Akhir ini, akan dilakukan beberapa pengujian pada sistem keseluruhan robot keseimbangan. Pengujian dilakukan meliputi pengambilan data sudut pada sesnor MPU6050 kemudian dibandingkan dengan sudut sebenarnya, pengujian keseimbangan pada hardware dengan metode tuning PID secara manual, pengujian pengolahan gambar menggunakan HSV dan terakhir pengujian robot untuk mengikuti objek.

4.2 Pengujian Perangkat Keras

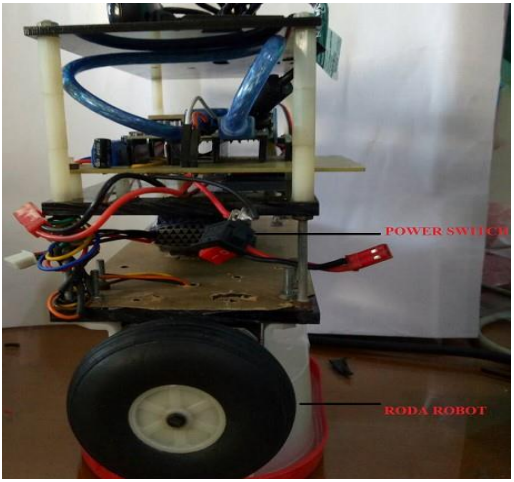
Pada pengujian perangkat keras terdapat beberapa tahap pengujian yang dilakukan natar lain pengujian *body* robot, pengujian rangkaian elektronika, pengujian sensor MPU 6050, pengujian *complementery filter*, pengujian PID controller pada motor stepper dan pengujian keseimbanagan pada robot. Pengujian ini dilakukan dengan menggunakan alat ukur dan dilakukan di laboratorium A206.

4.2.1 Pengujian *Body* Robot

Body robot yang dibuat sesuai dengan perancangan pada bab 3 sebelumnya. Karena robot keseimbangan adalah jenis inverted pendulum maka beban pada robot dimaksimalkan pada bagian tengah dan atas robot. Bahan yang digunakan pada robot ini menggunakan bahan jenis akrilik yang dipotong secara persegi panjang dengan ukuran 18 x 8 cm. Pada bagian bawah untuk menopang berat beban diatas digunakan mur dengan panjang 5cm dan tebal 3mm. Sedangkan pada bagian tengah menggunakan spicer dengan panjang 6 cm dengan tebal 5mm. Berikut gambar dari *body* robot kesimbnagan beroda dua.



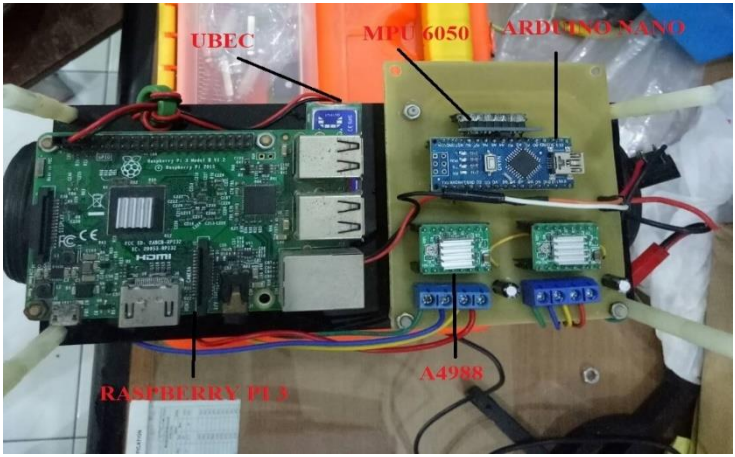
Gambar 4. 1 *Body* robot keseimbangan tampak depan



Gambar 4. 2 *Body* robot keseimbangan tampak samping

4.2.2 Pengujian Rangkaian Elektronika

Pembuatan *board* pada desain rangkaian elektronika untuk kontrol robot keseimbangan ini dibuat dengan mencetak desainnya pada papan PCB ukuran 12 x 8 cm. Pengujian pada rangkaian elektronika ini meliputi semua aspek yang digunakan untuk mengontrol robot keseimbangan tersebut. Adapaun yang diuji adalah nilai sensor MPU6050, kontrol motor arah putar motor stepper.



Gambar 4. 3 Ranngkain elektronika pada robot keseimbangan

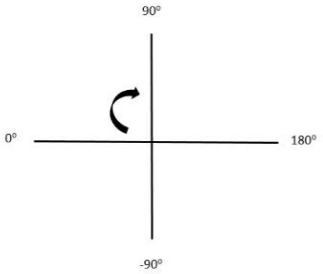
Tabel 4. 1 Penujian rangkaian elektronika pada robot

Pengujian	Hasil
Pembacaan MPU 6050	Berhasil
Motor 1 maju	Berhasil
Motor 1 mundur	Berhasil
Motor 2 maju	Berhasil
Motor 2 mundur	Berhasil

Berdasarkan pengujian setiap komponen pada rangkaian elektronika diatas semua komponen berfungsi dengan baik, maka dapat dikatakan bahwa rangkaian elektronika sudah baik.

4.2.3 Pengujian Sensor MPU6050

Pengujian MPU6050 untuk tingkat akurasi pembacaan kemiringan oleh sensor dan juga untuk mengetahui pembacaan sensor telah sesuai pada gambar 4.4. Percobaan ini menggunakan busur, sebuah Laptop, sensor MPU6050, board mikrokontroler, dan sebuah catu daya. Pengujian ini dilakukan di laboratorium A206 dan dilakukan dengan cara memutar board mikrokontroler searah jarum jam dan membandingkan data sensor dengan busur. Pemutaran sudut pada sensor dapat dilihat pada gambar 4.3 berikut.



Gambar 4. 4 Pengujian sudut pada MPU6050

Tabel 4. 2 Data pembacaan sensor pada mikrokontroller

Sudut Sebenarnya	Sudut Pembacaan	Error
0°	1,58°	1,58°
20°	18,78°	1,22°
40°	38,62°	1.38°
60°	62,01°	2,01°
80°	78,83°	1,83°
90°	88,07°	1,93°
120°	117,73°	2,27°
-120°	-118,66°	1,34°
-90°	-91,18°	1,12°
-80°	-78,92°	1,08°
-60°	-58,21°	1,79°
-40°	-38,82°	1,18°
-20°	-22,80°	2,80°

Dari tabel 4.2 hasil pengujian sensor diatas didapatkan bahwa nilai error dari sudut sebenarnya dengan sudut pembacaan sensor MPU6050 dengan rata – rata error ± 1,65º. Dengan demikian dapat disimpulkan bahwa sensor MPU6050 yang digunakan pada robot dalam kondisi pembacaan data yang cukup baik. Untuk mendapatkan hasil pembacaan keluaran sensor yang baik maka output dari sensor tersebut haruslah difilter terlebih dahulu menggunakan *complementary filter*.

4.2.4 Pengujian *Complementary Filter* Sensor MPU6050

Karena hasil dari pengujian sensor MPU6050 menunjukan hasil output dari *gyroscope* maupun *accelerometer* masih terdapat noise yang cukup tinggi yaitu dengan rata – rata error mencapai ± 1,65º . Maka untuk mengurangi atau meniadakan noise digunakan metode *complementary filter* pada keluaran nilai MPU 6050. Pada pengujian ini terdapat dua tahap pengujian yaitu pengujian nilai sensor tanpa menggunakan sensor dan pengujian nilai keluaran sensor dengan menggunakan *complementary filter*. Pada pengujian ini nilai koefisien alpha yang diuji ada beberapa nilai yaitu 0,95, 0,96, 0,9996. Nilai koefisien dari alpha dapat ditentukan dan disesuaikan sampai nilai output dari filter mendapatkan nilai output yang terbaik.

Tabel 4. 3 Hasil Pengujian filter dengan koefisien alpha 0,95

Koefisien Filter (0,95), waktu sampling (0,04) Tilt Angel X				
Sudut Pengujian	**Output Tanpa Filter**	**Hasil Selisih**	**Output Dengan Filter**	**Hasil Selisih**
0º	1,18º	1,18	0,21º	0,21
15º	16,55º	1,55	15,40º	0,40
30º	33,20º	3,2	30,5º	0,5
45º	44,33º	0,63	45,22º	0,22
60º	62,08º	2,08	60,07º	0,07
90º	88,71º	1,29	90,83º	0,83
Rata – rata		1,655	Rata – rata	0,371

Pada hasil pengujian diatas didapatkan nilai rata – rata error pada pembacaan sudut tanpa filter adalah ± 1,655º dan pada pembacaan sudut menggunakan filter dengan koefisien alpha 0,95 adalah 0,371º. Hal ini

menunjukan bahwa pengaruh dari filter complementary terhadap nilai keluaran sensor MPU6050 sangat berpengaruh dalam mengurangi noise.

Tabel 4. 4 Hasil Pengujian filter dengan koefisien alpha 0,96

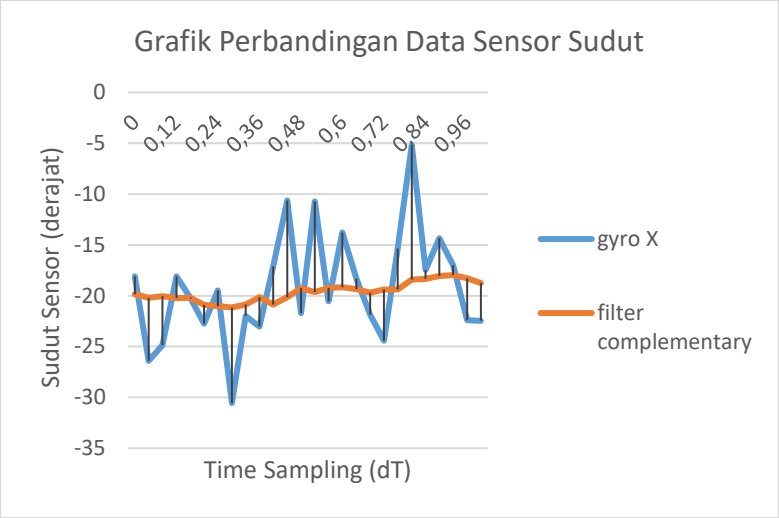
Koefisien Filter (0,96), waktu sampling (0,04) Tilt Angel X				
Sudut Pengujian	**Output Tanpa Filter**	**Hasil Selisih**	**Output Dengan Filter**	**Hasil Selisih**
0º	-1,55º	1,55	-0,95º	0,05
15º	16,75º	1,75	15,21º	0,21
30º	32,80º	2,80	29,91º	0,09
45º	44,33º	0,63	45,02º	0,02
60º	61,58º	1,58	60,15º	0,15
90º	88,70º	1,30	89,07º	0,93
Rata - rata		1,602	Rata - rata	0,228

Pada pengujian kedua yaitu dengan menggunakan nilai koefisien alpha sebesar 0,96 didapatkan hasil error yang lebih kecil jika dibandingkan dengan *complementary filter* dengan menggunakan nilai koefisien 0,95. Dimana nilai error yang didapat pada saat menggunakan koefisien 0,96 adalah sebesar ± 0,228º

Tabel 4. 5 Hasil Pengujian filter dengan koefisien alpha 0,996

Koefisien Filter (0,996), waktu sampling (0,04) Tilt Angel X				
Sudut Pengujian	**Output Tanpa Filter**	**Hasil Selisih**	**Output Dengan Filter**	**Hasil Selisih**
0º	-1,55º	1,55	-0,98º	0,02
15º	16,75º	1,75	15,12º	0,12
30º	32,80º	2,80	29,95º	0,05
45º	44,33º	0,63	44,71º	0,29
60º	61,58º	1,58	60,11º	0,11
90º	88,70º	1,30	90,28º	0,28
Rata - rata		1,602	Rata - rata	0,145

Pada pengujian filter yang ketiga didapatkan hasil output dengan filter nilai keluaran dari sensor sangat baik yaitu dengan nilai error yang sangat kecil, adapun nilai error yang didapat sebesar ± 0,145°. Gambar grafik pengujian *complementary filter* pada *tilt angle* X dengan koefisien 0,966 terhadap gyro X dapat dilihat pada gambar 4.5 berikut.



Gambar 4. 5 Grafik perbandingan nilai keluaran sensor dengan filter

Dapat dilihat dari hasil pengujian diatas grafik tersebut menunjukan perubahan nilai keluaran dari sensor MPU6050 pada sudut gyro X nilainya tidak konstan ketika sensor mendapat getaran sedikit saja, sedangkan pada hasil nilai keluaran yang menggunakan *complementary filter* lebih stabil. Pengujian diatas menggunakan koefisien alpha sebesar 0,996 dan waktu sampling sepanjang 1 detik. Sedangkan sudut yang diukur adalah sudut acak. Dari pengujian ini dapat disimpulkan bahwa perubahan nilai alpha sangat berpengaruh terhadap nilai hasil metode *complementary filter* dikarenakan nilai alpha jika dijabarkan terdapat nilai time constant dan waktu sampling yang akan mempengaruhi kestabilan nilai keluaran sensor. Untuk menghasilkan keluaran dengan error yang lebih baik dapat dicoba dengan menggunakan *kalman filter*.

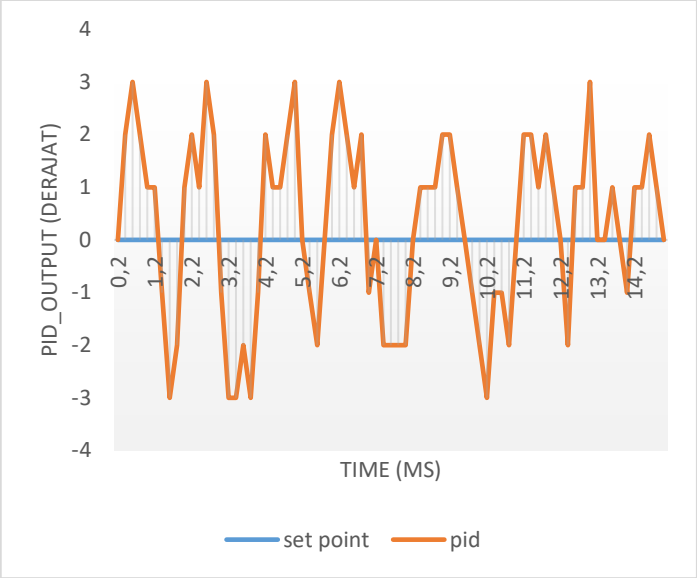
4.2.5 Pengujian PID Kontroller

Pada pengujian kontrol PID ini dilakukan dengan tujuan untuk mencari nilai keseimbangan yang baik pada robot. Dimana titik keseimbangan yang diinginkan adalah 0^o. Dimana nilai masukan adalah nilai output dari sensor sudut yang telah difilter. Metode PID yang digunakan adalah metode *manual tuning* dan *trial error*. Berikut adalah tabel pengujian untuk mencari parameter Kp, Ki, dan Kd yang terbaik yang digunakan pada robot ini.

Tabel 4. 6 Hasil pengujian *manual tuning* kontrol PID

NO	Kp	Ki	Kd	Keterangan
1	0	0	0	Kedua motor tidak bergerak
2	25	1	30	Motor bergerak tapi putarannya lemah dan bergoncang
3	28	1	25	Motor bergerak dan robot dapat berdiri tegak namun bergoncang
4	28	1,5	20	Motor bergerak dan robot dapat menstabilkan diri dengan baik namun masih ada getaran kecil
5	28	1,5	25	Motor bergerak dengan baik sehingga robot mampu menjaga keseimbangan walaupun ada gangguan dari luar
6	30	1,5	25	Motor bergerak dengan baik dan mampu membuat robot berdiri tegak namun getaran semakin besar
7	32	1,5	28	Motor begerak sangat kuat dan robot terjatuh
8	32	1,7	30	Motor bergerak sangat kuat dan robot terjatuh

Pada tabel diatas pengujian untuk mencari nilai Kp, Ki, dan Kd yang terbaik agar robot dapat menyeimbangkan diri dengan baik walaupun dapat gangguan dilakukan sebanyak delapan kali percobaan, dan dapat dilihat pada tabel hasil dari percobaan yang terbaik adalah pada percobaan 5 dengan parameter Kp = 28, Ki = 1,5, dan Kd = 25.



Gambar 4. 6 Grafik nilai keluaran PID pada percobaan 5

Pada gambar diatas dapat dilihat hasil pengujian keluaran nilai PID pada percobaan 5 nilai keluaran terlihat nilai error tidak terlalu besar berkisar ± 3. Karena nilai error tidak terlalu besar maka bisa dikatakan pada percobaan ini sistem keseimbangan dari robot bekerja dengan baik. Pengujian tersebut dilakukan tanpa diberikan gangguan dari luar. Metode yang digunakan saat ini sudah cukup baik namun sebagai alternatif dapat juga digunakan metode fuzzy logic untuk mencari parameter PID yang lebih sesuai.

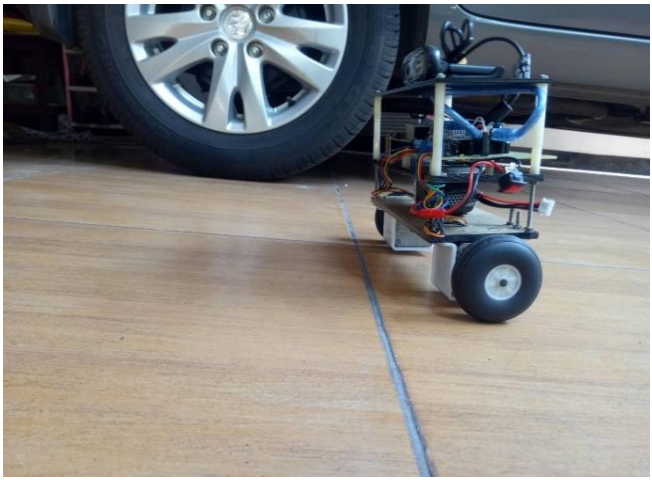
4.2.6 Pengujian Keseimbangan Robot

Paengujian kestabilan robot sangat diperlukan untuk menguji apakah robot mampu mempertahankan posisi tegak lurus diatas permukaan bumi. Kestabilan robot sangat dipengaruhi oleh kontrol PID, nilai set point dari PID kontroller mencadi acuan pada keseimbangan *body* robot ketika sedang bergerak. Pada pengujian ini dilakukan dengan beberapa tahapan pengujian yaitu pengujian pada bidang datar dan pengujian pada bidang miring. Pengujian pada bidang miring dilakukan dengan menentukan sudut yang diujikan, dimana sudut yang diujikan antara lain yaitu 0^o, 10^o, 15^o ,20^o ,25^o ,30^o ,40^o, 45^o. Berikut hasil pengujian kestabilan robot keseimbangan beroda dua.

Tabel 4. 7 Hasil pengujian kestabilan robot pada sudut tertentu

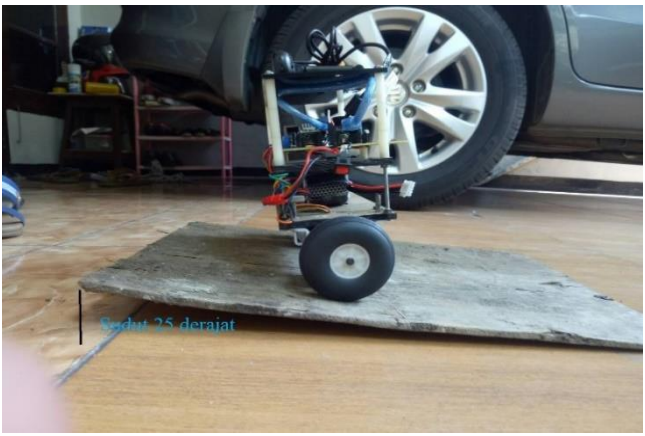
Sudut Pengujian	**Hasil Pengujian Kestabilan**
0^o	Berhasil
10^o	Berhasil
15^o	Berhasil
20^o	Berhasil
25^o	Berhasil
30^o	Berhasil
40^o	Gagal
45^o	Gagal

Pada pengujian kestabilan robot diatas robot mampu mempertahankan kestabilan dengan baik pada saat diuji dengan sudut kemiringan mulai dari 0^o sampai dengan kemiringan 30^o. Pada saat robot diuji dengan kemiringan lebih dari 30^o robot tidak mampu mempertahankan posisi stabilnya dan terjatuh. Berikut gambar pengujian kestabilan pada permukaan datar dan bidang miring pada robot.



Gambar 4. 7 Pengujian kestablian robot pada bidang datar

Terlihat pada bidang datar kestabilan robot sama sekali tidak mendapatkan masalah dan tidak jatuh. Untuk pengujian pada bidang miring dapat dilihat pada gambar berikut.



Gambar 4. 8 Pengujian kestablian robot pada bidang miring

Pengujian pada bidang miring ini dilakukan dengan sebuah papan yang posisinya diatur agar mendapat kemiringan tertentu. Untuk pengukuran kemiringan diukur dengan menggunkan busur. Terlihat pada gambar 4.8 tersebut bahwa tobot mampu menyeimbangkan badannya walaupun dalam kondisi miring. Robot keseimbnagan ini hanya mampu menjaga kestabilan posisi tegak lurus pada kemiringan maksimal 30º. Jadi pada sistem keseluruhan robot ini dibuat agar ketika robot memeiliki kemiringan lebih dari 30º maka sensor sudut akan mereset nilai awal sensor sehingga robot dalam kondisi mati. Untuk menghasilkan robot yang lebih stabil maka haruslah dibuat desain robot yang lebih seimbang.

4.2.7 Pengujian Rintangan Pada Robot

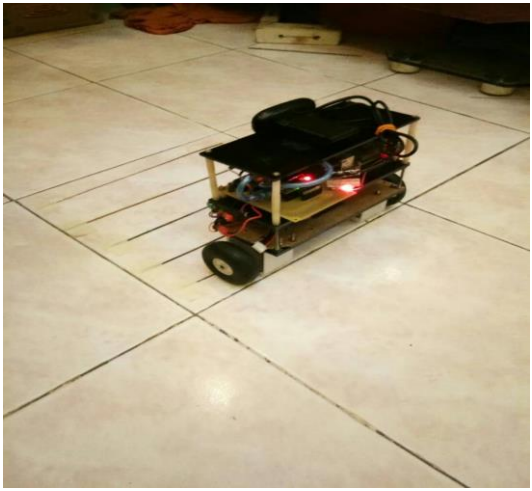
Pengujian ini dilakukan untuk mengetahui apakah robot ini mampu berjalan maju dan mundur dengan baik apabila permukaan jalan yang digunakan sebagai media uji dibuat dengan tidak rata. Sebagai alat bantu pada pengujian ini, batang lidi digunakan untuk membuat permukaan dari lantai tidak merata. Batang lidi yang digunakan guna mendukung pengujian bervariasi mulai dari satu batang sampai 5 batang. Berikut adalah tabel pengujian rintangan pada robot keseimbangan.

Tabel 4. 8 Hasil pengujian laju robot terhadap adanya rintangan

Jumlah Lidi	Maju	Mundur
1	Berhasil	Berhasil
2	Berhasil	Berhasil
3	Berhasil	Gagal
4	Gagal	Gagal
5	Gagal	Gagal

Pada pengujian ini hasil yang didapatkan ketika posisi robot maju mampu melintasi lidi dengan tebal 3 buah dengan baik namun ketika jumlah ketebalan lidi lebih dari 3 robot tidak mampu untuk melintasi

susunan batang llidi tersebut. Sedangkan untuk laju saat posisi mundur robot hanya mampu melintasi 2 buah sususan batang lidi.



Gambar 4. 9 Pengujian laju robot pada permukaan tidak rata

4.3 Pengujian Perangkat Lunak

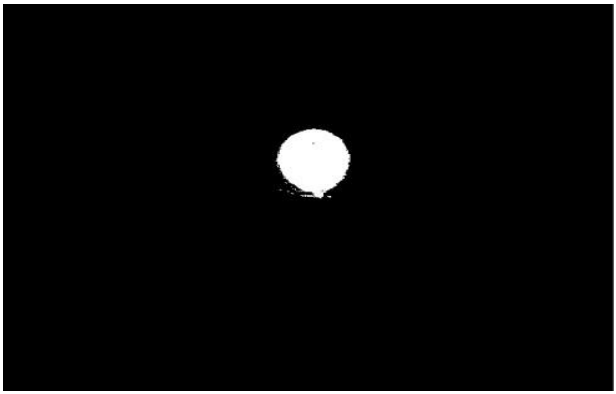
Tahap ini dilakukan untuk mengetahui dan menguji proses kerja dari program yang telah dibuat. Adapun program yang akan diuji adalah program visi komputer untuk mendeteksi pergerakan bola menggunakan kamera, dan pengujian untuk robot keseimbangan mengikuti objek.

4.3.1 Pengujian Deteksi Objek Menggunakan HSV

Pada pengujian ini dilakukan dengan tujuan agar raspberry pi dapat mengidentifikasi parameter – parameter yang ada pada setiap bola berdasarkan warnanya. Tahapan pengujian ini dilakukan dengan menguji beberapa warna yang akan ditangkap oleh kamera. Untuk dapat mengetahui kualitas dari program yang dibuat maka pengujian dilakukan dengan membedakan lima warna bola. Warna yang dijadikan bahan uji adalah warna merah, kuning, hijau, biru, dan ungu. Pada pengujian ini cara yang digunakan untuk pengujian warna dengan cara *manual tuning* nilai dari HSV.

Gambar 4. 10 Pengujian deteksi warna menggunkan lima bola

Pada gambar 4.10 dapat dilihat pengujian deteksi warna pada bola dengan cara menempatkan kelima bola dalam satu frame kamera. Pada percobaan pertama warna akan dideteksi adalah warna hijau. Dimana pada warna hijau ini nilai hue yang digunakan adalah 38 – 75 sedangkan nilai saturasi adalah 165-255. Berikut adalah gambar hasil dari deteksi bola dengan warna hijau.



Gambar 4. 11 Deteksi objek pada warna hijau

Dapat dilihat pada gambar 4.11 tersebut posisi objek yang terdeteksi sama dengan posisi bola warna hijau pada gambar 4.10. Hal ini menunjukan bahwa pengujian deteksi warna pada bola dengan warna hijau berhasil. Untuk percobaan kedua pengujian akan dilakukan pada bola dengan warna kuning, dimana warna kuning sendiri memiliki nilai *hue* antara 15 – 38 sedangkan untuk nilai saturasi antara 169 – 255. Berikut adalah gambar pengujian untuk mendeteksi warna pada bola kuning.

Gambar 4. 12 Deteksi objek pada warna kuning

Dapat dilihat pada gambar diatas bahwa hasil dari pengujian deteksi bola dengan warna kuning berhasil, posisi bola pada frame asli menunjukan kesamaan dengan posisi bola pada frame pengujian tersebut. Adapun hasil pengujian dari beberapa contoh objek yang digunakan dapat dilihat pada tabel berikut ini.

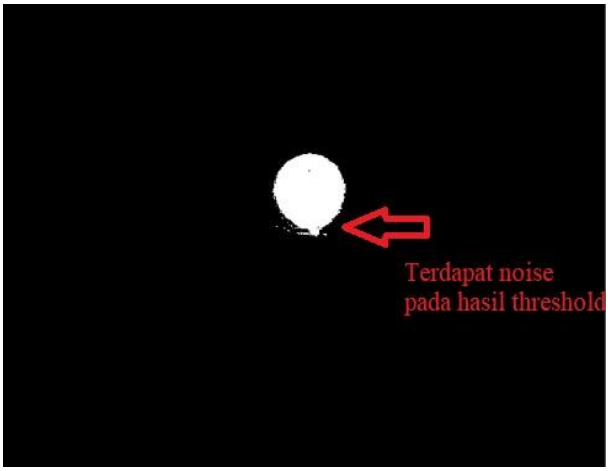
Tabel 4. 9 Hasil pengujian nilai *hue* dan *saturation* pada objek

Warna	***Hue***	***Saturation***
Merah	0 – 30	120 – 255
Kuning	15 – 38	169 – 255
Hijau	38 – 75	165 – 255
Biru	75 – 125	150 – 255
Ungu	120 - 169	110 - 255

Nilai pada tabel tersebut adalah hasil tuning dari HSV pada setiap warna objek yang digunakan. Untuk nilai *value* sendiri yaitu intensitas cahaya tidak dapat dijadikan tolak ukur yang pasti untuk setiap warna yang diuji sebab nilai *value* sangat bergantung pada pencahayaan ruangan pada saat pengambilan gambar objek. Semakin terang maka *range* dari *value* akan semakin pendek jika cahaya dari ruangan semakin kurang mendapat cahaya maka tuning nilai value range semakin besar. Pengujian menggunakan warna ini cukup baik namun ada beberapa kendali karena faktor pencahayaan, untuk mendapatkan suatu pendeteksian yang bagus dapat menggunakan metode pengenalan objek yang lain.

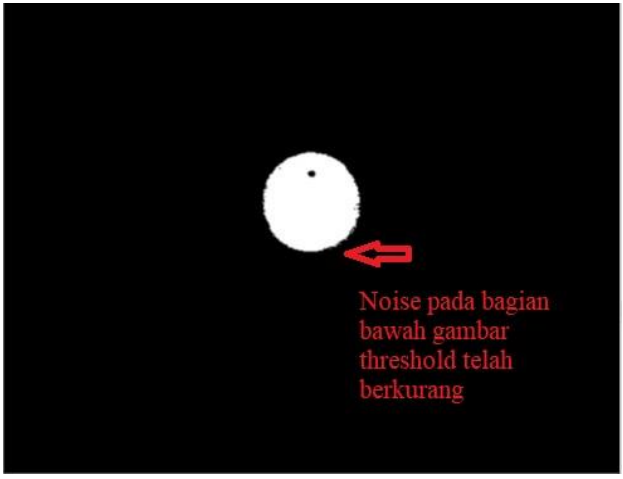
4.3.2 Pengujian Filter Erode dan Dilate

Proses pengujian filter ini dilakukan setelah raspberry mampu mendeteksi sebuah objek dengan menggunakan warna. Hasil threshold dari objek berwarna tersebut tidak begitu baik dalam membentuk pola lingkaran, hal ini disebabkan karena masih adanya noise pada gambar tersebut. Untuk mengurangi atau meniadakan noise pada hasil threshold pada objek berwarna tersebut maka digunakan filter morfologi erode dan dilate. Dimana fungsi dari erode adalah memperkecil blob pada gambar dan dilate digunakan untuk memperbesar blob pada gambar. Dengan menggunakan filter ini akan didapatkan hasil threshold gambar yang lebih baik. Gambar 4.9 berikut menunjukan hasil threshold objek berwarna yang tidak menggunakan filter.



Gambar 4. 13 Hasil threshold tanpa menggunakan filter

Sebagai pembanding data yang menunjukan bahwa pada saat program OpenCV untuk mendeteksi bola ini ditambahkan filter morfologi maka hasilnya dpat dilihat pada gambar 4.13 berikut.

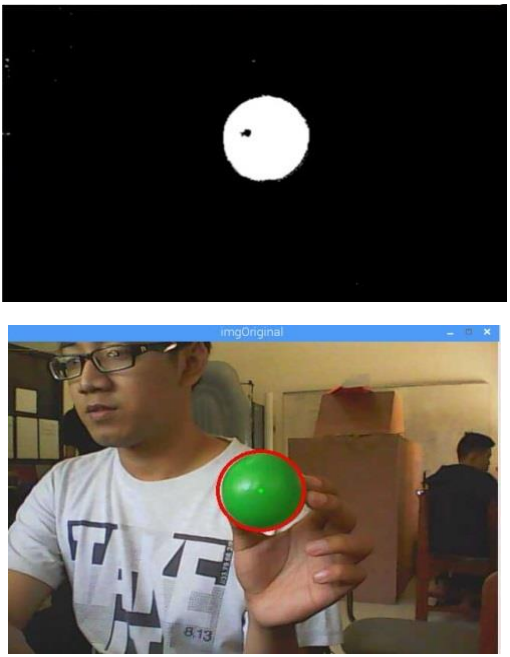


Gambar 4. 14 Hasil threshold dengan menggunakan filter

Terlihat pada gambar 4.14 tersebut hasil threshold gambar yang menggunakan filter morfolgi erode dan dilate memiliki hasil gambar yang lebih baik. *Noise* yang mulanya terdapat pada bagian bawah objek telah hilang, meski begitu masih terdapat beberapa *noise* kecil disekeliling objek yang berbentuk lingkaran tersebut.

4.3.3 Pengujian *HoughCircle Transform*

Untuk mendeteksi tepi lingkaran pada sebuah gambar yang telah dithreshold digunakan metode *hough circle transform.* Pada program ini fungsi dari metode ini adalah menemukan titik tengah koordinat x,y pada frame kamera. Selain itu metode ini juga dapat mengukur besar jari – jari dari sebuah lingkaran. Langkah awal dari program ini adalah membentuk sisi tepi lingkaran dengan membuat garis. Berikut gambar hasil pengujian membuat deteksi garis tepi lingkaran.

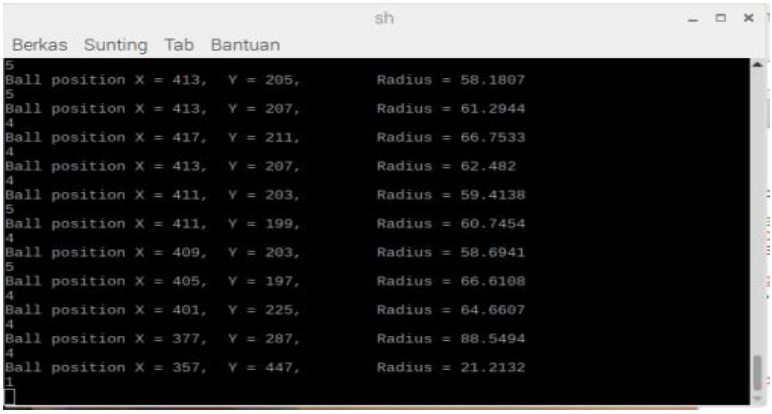


Gambar 4. 15 Pembuatan garis tepi pada lingkaran

Dapat dilihat pada gambar 4.15 gambar hasil threshold diatas dideteksi garis tepi lingkarannya kemudian pada gambar asli tepi lingkaran pada bola yang berwarna hijau digambarkan garis warna merah dengan titik pusat x, y adalah titik warna hijau. Untuk lebih jelasnya dapat dilihat pada gambar 4.16 berikut.

Gambar 4. 16 Titik hijau menunjukan pusat dari lingkaran

Nilai besarnya pixel dalam variabel radius dapat ditentukan dari garis tepi lingkaran dan titik pusat lingkaran. Hasil dari pengukuran posisi titik tengah (x,y) dan radius dapat dilihat pada gambar 4.17 berikut.



Gambar 4. 17 Pembacaan titik koordinat bola pada gambar

Pada pengujian posisi dan jarak pada bola ini bertujuan untuk mengetahui arah bola pada saat pengujian pergerakan pada robot keseimbnagan. Untuk arah gerak dari robot keseimbangan ini dibagi menjadi tiga bagian yaitu kanan, kiri, dan tengah. Sedangkan untuk maju dan mundur dari robot keseimbangan ini ditentukan dengan besar kecilnya radius. Pada pengujian posisi dibuat lima jenis keluaran yaitu sebagai berikut :

- Tengah dan maju = 1
- Tengah dan diam = 5
- Tengah dan mundur = 4
- Kiri = 3
- Kanan = 2

Inisial angka tersebut sebagai nilai keluaran pada serial monitor, inisial angka tersebut yang akan dikirim dari *raspberry pi* ke *arduino*. Pada gambar 4.16 dan gambar 4.17 dapat dilihat pada saat titik tengah lingkaran berada pada tengah frame kamera maka output yang keluar menunjukan koordinat x adalah 357 maka nilai output adalah 1. Untuk menguji jarak objek dari robot keseimbnagan dilakukan pengukuran radius dengan menggunakan alat ukur penggaris. Berikut adalah gambar pengujian jarak.

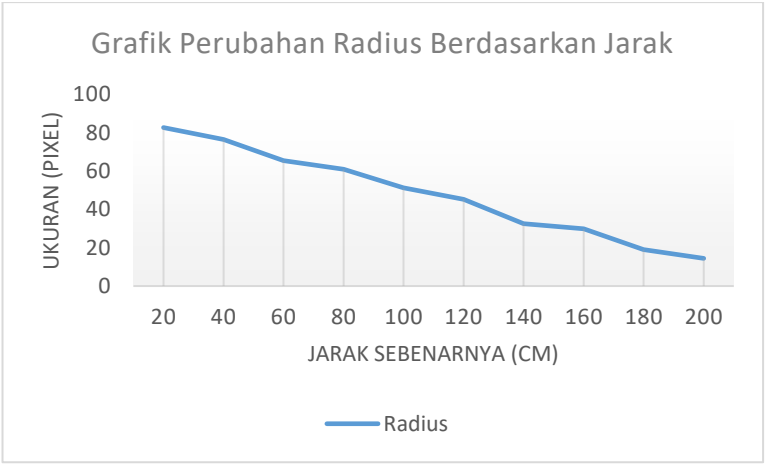
Gambar 4. 18 Pengujian radius dengan jarak tertentu

Pada pengukuran jarak berdasarkan radius bola, batas – batas jarak yang digunakan dengan rentan jarak mulai dari 20 cm sampai 220 cm. Berikut adalah data pengujian jarak berdasarkan radius dapat dilihat pada tabel 4.10.

Tabel 4. 10 Hasil pengujian jarak berdasarkan radius bola

Jarak Sebenarnya	**Radius (pixel)**
20	82,753
40	76,565
60	65,432
80	60,910
100	51,287
120	45,210
140	32,557
160	29,881
180	19,091
200	14,441
220	Tidak tedeteksi
240	Tidak terdeteksi

Data dari tabel pengujian diatas menunjukan bahwa nilai *pixel* dari radius akan semakin besar jika bola yang terdeteksi pada jarak yang dekat dan akan semakin kecil jika bola menjauh dari jangkaun kamera. Jarak minimal yang dapat dideteksi oleh kamera adalah 20 cm sedangkan jarak terjauh adalah 200 cm, diatas jarak tersebut program tidak dapat mendeteksi bentuk objek. Grafik dari tabel pengujian diatas dapat dilihat pada gambar 4.19 berikut



Gambar 4. 19 Grafik perubahan radius berdasarkan jarak objek

Pada tabel diatas dapat disimpulkan bahwa laju pergerakan dari perubahan radius berdasarkan jarak objek besifat linier. Semakin dekat objek semakin besar pula nilai *pixel* pada radius lingkaran begitu pula sebaliknya.

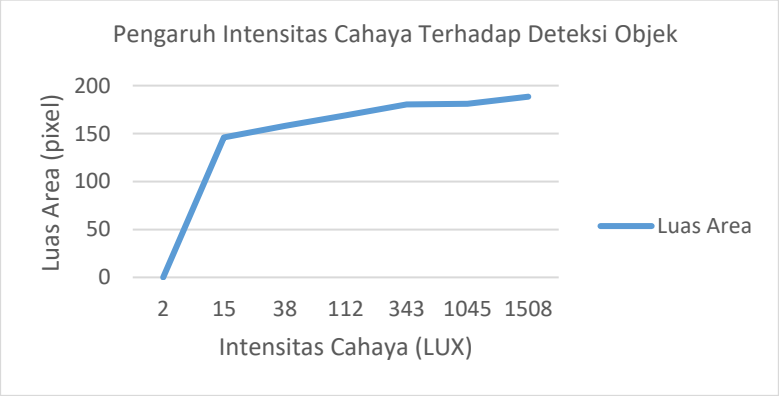
4.3.4 Pengujian Intensitas Cahaya Tehadap Deteksi Objek

Pengujian ini dilakukan untuk dapat mengetahui kemampuan program dalam mendeteksi bola dengan intensitas cahaya yang berbeda – beda. Nilai intensitas cahaya dapat dilihat dari besar kecilnya lux yang terdeteksi. Untuk dapat mengetahui nilai lux pada saat pengujian digunakan aplikasi *lux meter* pada *smartphone*. Pengujian ini dilakukan di beberapa tempat dengan tingkat pencahayaan yang berbeda – beda. Data yang diambil pada pengujian ini adalah besarnya diameter lingkaran dan apakah objek terdeteksi dengan baik. Pengujian dilakukan sebanyak 7 kali dengan jarak antara objek dan robot sejauh 30 cm. Berikut adalah tabel hasil pengujian pengaruh intensitas cahaya terhadap pendeteksian objek.

Tabel 4. 11 Hasil pengujian jarak berdasarkan radius bola

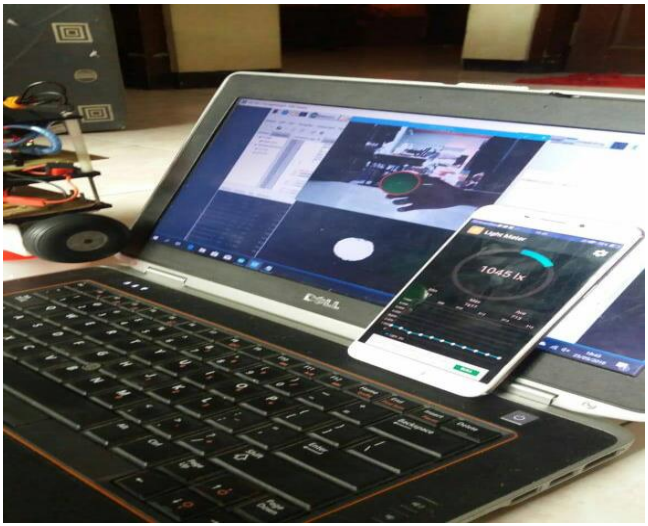
No	Lux	Jarak	Diameter	Keterangan
1	2	30 cm	0	Tidak terdeteksi
2	15	30 cm	145,88	Terdeteksi
3	38	30 cm	158,12	Terdeteksi
4	112	30 cm	169,09	Terdeteksi
5	343	30 cm	180,31	Terdeteksi
6	1045	30 cm	180,92	Terdeteksi
7	1508	30 cm	188,45	Terdeteksi

Pada tabel tersebut nilai lux yang sangat rendah atau dapat dikatakan tidak ada pencahayaan sama sekali robot tidak dapat mendeteksi objek yang ada di depannya. Sedangkan pada pengujian kedua sampai tujuh, robot mampu mendeteksi objek bergerak dengan baik. Grafik pengaruh intensitas cahaya pada pendeteksian objek dapat dilihat pada gambar berikut.



Gambar 4. 20 Grafik perubahan radius berdsarkan jarak objek

Pada grafik di atas dapat disimpulkan bahwa pengaruh intensitas cahaya pada pendeteksian objek menggunakan metode HSV sangat berpengaruh. Intesitas cahaya yang sangat kecil tidak dapat dideteksi oleh sistem sebab tidak ada cahaya yang masuk pada lensa kamera. Sedangkan jika cahaya yang masuk mencukupi maka pendeteksian objek dapat dikatakan cukup stabil. Pengujian pengaruh intensitas cahaya dapat dilihat pada gambar 4.21 berikut.



Gambar 4. 21 Pengujian pengaruh intensitas cahaya

4.3.5 Pengujian Kecepatan Robot

Pada pengujian ini dilakukan untuk mengetahui kecepatan robot saat melakukan pergerakan maju maupun mundur. Ketika robot pada posisi maju maka nilai *set point* pada program akan ditambahkan sebesar 0,5^{o}, dan nilai PID *output* yang menjadi nilai kecepatan motor ditambah kecepatannya sesuai dengan kondisi yang diinginkan begitu pula dengan saat robot mundur *set point* akan dikurangi 0,5^{o}, dan PID *output* dikurangi sesuai yang diinginkan. Pengujian ini dilakukan dengan variabel speed yang berbeda dengan jarak tempuh maju maupun mundur sejauh 2 m.

Tabel 4. 12 Hasil pengujian jarak berdasarkan radius bola

Max Speed	**Jarak Uji**	**Waktu**	**Kecepatan Robot**	**Keterangan**
+70	2 m	14,01 s	0,142 m/s	Maju (stabil)
+90	2 m	10 s	0,2 m/s	Maju (stabil)
+120	2 m	7,31 s	0,297 m/s	Maju (terjatuh)
-70	2 m	13 s	0,153 m/s	Mundur (stabil)
-90	2 m	9,5 s	0,21 m/s	Mundur (stabil)
-120	2 m	6,45 s	0,31 m/s	Mundur (terjatuh)

Hasil pengujian kecepatan motor menunjukan kecepatan maksimum yang dapat ditempuh adalah 0,2 m/s, jika *max speed* pada program *balancing robot* ditambah lebih dari 90 maka yang terjadi adalah kecepatan motor lebih cepat namun robot tidak dapat menyeimbangkan badan dengan baik sehingga terjatuh. Sedangkan untuk mundur kecepatan maksimum yang dapat diperoleh adalah 0,21 m/s.

Gambar 4. 22 Pengujian kecepatan motor

4.3.6 Pengujian Sistem Keseluruhan

Pada pengujian ini adalah pengujian dari keseluruhan sistem yang telah dibuat, pengujian ini meliputi robot mengikuti objek dengan arah yang berbeda – beda dan dengan kecepatan yang berbeda – beda pula. Pengujian kecepatan pendeteksian objek oleh raspberry pi 3 ini sangat penting dilakukan sebab agar dapat mengetahui apakah robot keseimbagan dua roda ini dapat merespon dengan baik objek yang bergerak dengan cukup cepat. Berikut adalah data yang telah didapatkan saat pengujian.

Tabel 4. 13 Hasil pengujian gerak robot ketika mengikuti objek

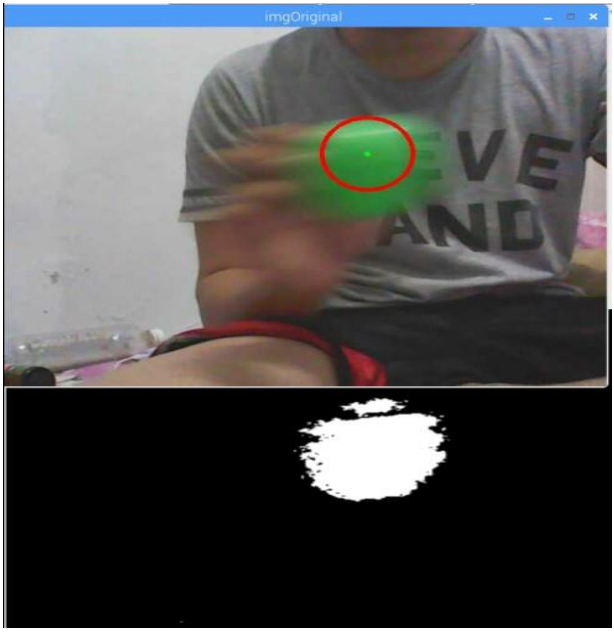
Arah Gerak	Status Objek	Keterangan
Maju	Terdeteksi	Berhasil
Mundur	Terdeteksi	Berhasil
Kanan	Terdeteksi	Sedikit gangguan
Kiri	Terdeteksi	Berhasil

Pada pengujian robot saat mengikuti objek ini, robot keseimbangan dua roda mampu mengikuti objek dengan baik namun ada sedikit kendala saat arah belok pada sisi kanan, hasil yang didapat saat pengujian menunjukan gerak putar robot pada arah kanan tidak berputar dengan sempurna karena masih terdapat beberapa gerakan yang tidak diinginkan, hal tersebut dapat dikarenakan mekanik pada *body* robot yang kurang baik. Sedangkan untuk pembacaan objek yang dilakukan saat robot melakukan gerakan hasil yang didapatkan cukup baik karena objek terdeteksi ketika robot bergerak ke segala arah. Dengan demikian dapat dikatakan bahwa sistem gerak dari robot keseimbangan dua roda ini sudah berjalan sesuai yang diharapkan. Namun ada beberapa kendala dari sisi pendeteksian pada objek yang bergerak dengan cepat. Pada tabel berikut dapat dilihat hasil pengujian pendeteksian objek dengan berbagai kecepatan gerak yang berbeda.

Tabel 4. 14 Hasil pengujian pendeteksian objek pada benda bergerak

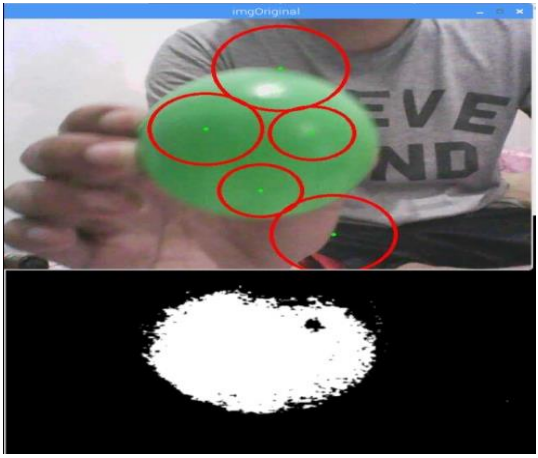
Pengujian	Kecepatan	Waktu Terbaca	Keterangan
1	Lambat	32 ms	Terdeteksi
2	Sedang	55 ms	Terdeteksi
3	Cepat	-	Tidak terdeteksi
4	Sangat Cepat	-	Tidak terdeteksi

Pada pengujian tersebut objek hanya mampu terdeteksi ketika kecepatan objek yang melintas tidak terlalu cepat. Sebab delay deteksi objek yang terjadi apabila objek bergerak cukup cepat maka akan semakin tinggi. Pengujian ini dilakukan dengan menggunakan *stopwatch*.



Gambar 4. 23 Pengujian dengan kecepatan gerak objek sedang

Permasalahan dalam pembacaan objek yang digunakan pada sistem ini ditemukan pada saat pengujian dilakukan yaitu ketika jarak antara objek dan robot sangat dekat, maka akan muncul beberapa *circle* lain selain *circle* sesungguhnya. Gambar pembacaan objek dengan jarak dekat dapat dilihat pada gambar berikut.



Gambar 4. 24 Pembacaan objek dengan jarak dekat

Gambar 4. 25 Pengujian sistem keseluruhan

4.3.7 Pengujian Sistem Keseluruhan Menggunakan Beban

Pada pengujian ini dilakukan untuk mengetahui seberapa besar robot ini mampu menopang beban maksimum yang ada diatas badan robot dan apakah robot ini mampu melaju dengan membawa beban baik itu pada bidang datar maupun bidang miring. Pengujian berat dillakukan dengan cara memberi beban pada robot dengan minuman kemasan dengan berat yang tertera yaitu 200 mL setiap kemasan. Pengujian dilakukan sebanyak 4 kali dengan setiap masing – masing pengujian ditambahkan 1 gelas. Berikut adalah table hasil pengujian beban maksimum.

Tabel 4. 15 Pengujian beban maksimum pada robot keseimbangan

Jumlah Beban	Berat Beban	Keterangan
1 gelas	200 mL	Berhasil
2 gelas	400 mL	Berhasil
3 gelas	600 mL	Berhasil
4 gelas	800 mL	Terjatuh

Hasil dari penguian beban maksimum menunjukan bahwa beban maksimum yang mampu ditopang oleh robot baik ketika diam maupun bergerak adalah sebesar 600 mL. Namun ketika beban mencapai lebih dari 600 mL maka yang terjadi adalah robot tidak mampu menopang dirinya sendiri sehingga menganggu kestabilan robot. Untuk menghasilkan robot yang mampu menopang beban yang besar dapat digunakan dengan jenis motor stepper dengan spesifikasi *torsi* dan RPM yang lebih tinggi. Sedangkan untuk pengujian robot berjalan dan membawa beban pada kondisi permukaan yang tidak rata, yaitu dengan sudut kemiringan 25^{o}, robot ini mampu membawa beban pada keadaan diam dan pada keadaan ketika mengikuti pergerakan bola baik itu melaju kedepan maupun kebelakang. Berikut adalah gambar pengujian robot mengikuti objek dengan membawa beban dapat dilihat pada gambar 4.26.

Gambar 4. 26 Pengujian beban maksimum pada robot keseimbanga

Sedangkan gambar pengujian robot membawa beban pada area kemiringan sudut 25^{o} dapat dilihat pada gambar 4.27 berikut ini.

Gambar 4. 27 Robot melaju pada bidang miring dengan beban

BAB 5
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari hasil pengujian dan analisa pada bab 4, penulis mengambil beberapa kesimpulan sebagai berikut:

1. Pembuatan mekanik pada robot keseimbangan sangat berpengaruh dalam pengujian untuk menstabilkan gerakan robot. Desain yang baik akan membuat laju pergerakan robot semakin baik.
2. Melihat hasil data dari penggunaan *filter complementary* nilai keluaran sensor MPU6050 dengan pengujian nilai koefisien alpha 0,996 adalah yang terbaik dengan rata – rata nilai *error* sebesar ± 0,145º. Sedangkan hasil dari pengujian kontroller PID didapatkan hasil pengujian yang terbaik adalah dengan parameter Kp = 28, Ki = 1,5 dan Kd = 25.
3. Hasil pengujian keseluruhan sistem memperlihatkan bahwa robot dapat mengikuti objek dengan baik dengan kecepatan pergerakan objek yang tidak terlalu cepat dikarenakan kemampuan raspberry pi 3 yang memiliki RAM dan GPU yang tidak terlalu besar dan kecepatan maksimum yang dapat dilakukan oleh robot keseimbangan ketika melaju adalah 0,2 m/s.

5.2. Saran

Saran yang penulis berikan untuk pengembangan berikutnya dari tugas akhir ini adalah bahwa model desain dari *body* robot dapat dibuat lebih besar agar dapat dikembangkan menjadi robot dengan fungsi yang lebih banyak lagi. Serta untuk pengembangan pengolahan gambar dapat menggunakan metode lain yang lebih baik dari metode HSV dan *Hough Circle Transform*.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] "Menghadapi Kemunculan Teknologi Robotika di Era Industri 4.0." [Online].Available :https://news.okezone.com/read/2019/03/16/65/2030940/menghadapi-kemunculan-teknologi-robotik-di-era-industri-4-0. [Accessed : 25 April 2019].

[2] Kurniawan, Dayat, Aplikasi Elektronika dengan Visual C# 2008 Express Edition, Rlrx Media Komputindo, Jakarta, 2010.

[3] T. Sutoyo, E. Mulyanto, V. Suhartono, O.D. Nurhayati, Wijanarto, Teori Pengolahan Citra Digital, Penerbit Andi, Yogyakarta, 2009, p.256.

[4] Elvin, Toh Boon Heng, *Development of a Self Balancing Scooter*, Nanyang Technology University, Mei 2017.

[5] Su, X., Wang, C., Su, W., & Ding, Y. (2016). *Control of balancing mobile robot on a ball with fuzzy self-adjusting PID. 2016 Chinese Control and Decision Conference (CCDC).*

[6] Sun, T., Tang, S., Wang, J., & Zhang, W. (2013). *A robust lane detection method for autonomous car-like robot. 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*

[7] Ngo, H.-Q.-T., Nguyen, T.-P., Huynh, V.-N.-S., Le, T.-S., & Nguyen, C.-T. (2017). *Experimental comparison of Complementary filter and Kalman filter design for low-cost sensor in quadcopter. 2017 International Conference on System Science and Engineering (ICSSE).*

[8] Hutama, I., 2011. Kendali Pendulum Terbalik Dinamis, Skripsi, Teknik, Universitas Gadjah Mada, Yogyakarta.

[9] J. Bernd, H. Horst, *Computer Vision and Applications*, Academic Press, San Diego, California, 2000, p.679.

[10] Cameron, K., & Islam, M. S. (2018). *Multiple Objects Detection using HSV. NAECON 2018 - IEEE National Aerospace and Electronics Conference.*

[11] Setyawan, L. B., Susilo, D., & Irawan, D. (2014). Sistem Kendali Gerak Segway Berbasis Mikrokontroller. Slatiga: Jurnal IlmiahElektronika Vol. 13 No. 1 April Hal 125 – 134.

[12] "*What Is The Raspberry Pi ?"*. [Online]. Available : https://www.zdnet.com/article/what-is-the-raspberry-pi-3-everything-you-need-to-know-about-the-tiny-low-cost-computer/. [Accessed : 25 April 2019].

[13] Wetzstein, Gordon. "Inertial Measurement Unit I" Class Lecture, *Virtual Reality from Stanford University, California*, February 10, 2019.

[14] Kim, W., Yang, C., & Chung, C. C. (2011). *Design and Implementation of Simple Field-Oriented Control for Permanent Magnet Stepper Motors Without DQ Transformation. IEEE Transactions on Magnetics, 47(10), 4231–4234.*

[15] "*Basic Of The I2C Communication Protocol"*. [Online]. Available : http://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/. [Accessed : 25 April 2019].

[16] Ogata, K. (1997). Teknik Kontrol Automatik – terjemahan: Ir. Edi Laksono. Erlangga: Jakarta.

[17] A. P. Gunawan, H. Subagiyo, and R. T. Wahyuni, " Kontrol Kesetimbangan pada Robot Beroda Dua Menggunakan Pengendali PID dan Filter Komplementari," p.11.

LAMPIRAN A

Program *Self Balancing Robot* :

```
#include <Wire.h>
int gyro_address = 0x68;
int acc_calibration_value = 950;

//Various settings
float pid_p_gain = 28 ;
float pid_i_gain = 1.5;
float pid_d_gain = 25;
float turning_speed = 25;
float max_target_speed = 70;

byte start, low_bat;
int a;

int left_motor, throttle_left_motor, throttle_counter_left_motor, throttle_left_motor_memory;
int right_motor, throttle_right_motor, throttle_counter_right_motor, throttle_right_motor_memory;
int receive_counter;
int gyro_pitch_data_raw, gyro_yaw_data_raw, accelerometer_data_raw;

long gyro_yaw_calibration_value, gyro_pitch_calibration_value;

unsigned long loop_timer;

float angle_gyro, angle_acc, angle, self_balance_pid_setpoint;
float pid_error_temp, pid_i_mem, pid_setpoint, gyro_input, pid_output, pid_last_d_error;
float pid_output_left, pid_output_right;

void setup(){
  Serial.begin(115200);                                    //Start the serial port at 9600 kbps
```

```
  Wire.begin();                                        //Start the I2C bus as
master
  TWBR = 12;                                           //Set the I2C clock
speed to 400kHz

  TCCR2A = 0;
  TCCR2B = 0;
  TIMSK2 |= (1 << OCIE2A);
  TCCR2B |= (1 << CS21);
  OCR2A = 39;
  TCCR2A |= (1 << WGM21);

  //By default the MPU-6050 sleeps. So we have to wake it up.
  Wire.beginTransmission(gyro_address);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission();
  //Set the full scale of the gyro to +/- 250 degrees per second
  Wire.beginTransmission(gyro_address);
  Wire.write(0x1B);
  Wire.write(0x00);
  Wire.endTransmission();
  //Set the full scale of the accelerometer to +/- 4g.
  Wire.beginTransmission(gyro_address);
  Wire.write(0x1C);
  Wire.write(0x08);
  Wire.endTransmission();
  //Set some filtering to improve the raw data.
  Wire.beginTransmission(gyro_address);
  Wire.write(0x1A);
  Wire.write(0x03);
  Wire.endTransmission();
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
```

```
  pinMode(13, OUTPUT);

  for(receive_counter = 0; receive_counter < 500; receive_counter++){
    if(receive_counter % 10 == 0)digitalWrite(13, !digitalRead(13));
    Wire.beginTransmission(gyro_address);
    Wire.write(0x43);
    Wire.endTransmission();
    Wire.requestFrom(gyro_address, 4);
    gyro_yaw_calibration_value += Wire.read()<<8|Wire.read();
    gyro_pitch_calibration_value += Wire.read()<<8|Wire.read();
    delayMicroseconds(3700);
  }
  gyro_pitch_calibration_value /= 500;
  gyro_yaw_calibration_value /= 500;

  loop_timer = micros() + 4000;
}

////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////
//Main program loop
////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////
void loop(){
  if(Serial.available()){
    a = Serial.read();
    Serial.println(a);
  }


  Wire.beginTransmission(gyro_address);
  Wire.write(0x3F);
  Wire.endTransmission();
  Wire.requestFrom(gyro_address, 2);
  accelerometer_data_raw = Wire.read()<<8|Wire.read();
  accelerometer_data_raw += acc_calibration_value;
```

```
  if(accelerometer_data_raw > 8200)accelerometer_data_raw = 8200;
  if(accelerometer_data_raw < -8200)accelerometer_data_raw = -8200;

  angle_acc = asin((float)accelerometer_data_raw/8200.0)* 57.296;

  if(start == 0 && angle_acc > -0.5&& angle_acc < 0.5){
    angle_gyro = angle_acc;
  }

  Wire.beginTransmission(gyro_address);
  Wire.write(0x43);
  Wire.endTransmission();
  Wire.requestFrom(gyro_address, 4);
  gyro_yaw_data_raw = Wire.read()<<8|Wire.read();
  gyro_pitch_data_raw = Wire.read()<<8|Wire.read();

  gyro_pitch_data_raw -= gyro_pitch_calibration_value;
  angle_gyro += gyro_pitch_data_raw * 0.000031;

  gyro_yaw_data_raw -= gyro_yaw_calibration_value;
  //Uncomment the following line to make the compensation active
  angle_gyro -= gyro_yaw_data_raw * 0.0000003;

  angle_gyro = angle_gyro * 0.9996 + angle_acc * 0.0004;

  pid_error_temp = angle_gyro - self_balance_pid_setpoint -
pid_setpoint;
  if(pid_output > 10 || pid_output < -10)pid_error_temp += pid_output *
0.015 ;

  pid_i_mem += pid_i_gain * pid_error_temp;
  if(pid_i_mem > 400)pid_i_mem = 400;
  else if(pid_i_mem < -400)pid_i_mem = -400;
  //Calculate the PID output value
  pid_output = pid_p_gain * pid_error_temp + pid_i_mem + pid_d_gain
* (pid_error_temp - pid_last_d_error);
```

```
  if(pid_output > 400)pid_output = 400;
  else if(pid_output < -400)pid_output = -400;

  pid_last_d_error = pid_error_temp;

  if(pid_output < 5 && pid_output > -5)pid_output = 0;

  if(angle_gyro > 30 || angle_gyro < -30 || start == 0 || low_bat == 1){
    pid_output = 0;
    pid_i_mem = 0;
    start = 0;
    self_balance_pid_setpoint = 0;
  }


////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////
  //Control calculations

////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////
  pid_output_left = pid_output;
  pid_output_right = pid_output;

  if(a == 3){
    pid_output_left += turning_speed+100;
    pid_output_right -= turning_speed;
  }
  if(a == 2){
    pid_output_left -= turning_speed;
    pid_output_right += turning_speed;
  }

  if(a == 1){
    if(pid_setpoint > -2.5)pid_setpoint -= 0.05;
    if(pid_output > max_target_speed * -1)pid_setpoint -= 0.005;
```

```
  }
  if(a == 4){
    if(pid_setpoint < 2.5)pid_setpoint += 0.05;
    if(pid_output < max_target_speed)pid_setpoint += 0.005;
  }

  if(a == 5){
    if(pid_setpoint > 0.5)pid_setpoint -=0.05;
    else if(pid_setpoint < -0.5)pid_setpoint +=0.05;
    else pid_setpoint = 0;
  }

  //The self balancing point is adjusted when there is not forward or backwards movement from the transmitter. This way the robot will always find it's balancing point
  if(pid_setpoint == 0){
    if(pid_output < 0)self_balance_pid_setpoint += 0.0015;
    if(pid_output > 0)self_balance_pid_setpoint -= 0.0015;
  }

  //To compensate for the non-linear behaviour of the stepper motors the folowing calculations are needed to get a linear speed behaviour.
  if(pid_output_left > 0)pid_output_left = 405 - (1/(pid_output_left + 9)) * 5500;
  else if(pid_output_left < 0)pid_output_left = -405 - (1/(pid_output_left - 9)) * 5500;

  if(pid_output_right > 0)pid_output_right = 405 - (1/(pid_output_right + 9)) * 5500;
  else if(pid_output_right < 0)pid_output_right = -405 - (1/(pid_output_right - 9)) * 5500;

  //Calculate the needed pulse time for the left and right stepper motor controllers
  if(pid_output_left > 0)left_motor = 400 - pid_output_left;
  else if(pid_output_left < 0)left_motor = -400 - pid_output_left;
```

```
  else left_motor = 0;

  if(pid_output_right > 0)right_motor = 400 - pid_output_right;
  else if(pid_output_right < 0)right_motor = -400 - pid_output_right;
  else right_motor = 0;

  //Copy the pulse time to the throttle variables so the interrupt subroutine
can use them
  throttle_left_motor = left_motor;
  throttle_right_motor = right_motor;

  //The angle calculations are tuned for a loop time of 4 milliseconds. To
make sure every loop is exactly 4 milliseconds a wait loop
  //is created by setting the loop_timer variable to +4000 microseconds
every loop.
  while(loop_timer > micros());
  loop_timer += 4000;
}

ISR(TIMER2_COMPA_vect){
  //Left motor pulse calculations
  throttle_counter_left_motor ++;
  if(throttle_counter_left_motor > throttle_left_motor_memory){
    throttle_counter_left_motor = 0;
    throttle_left_motor_memory = throttle_left_motor;
    if(throttle_left_motor_memory < 0){
      PORTD &= 0b00100000;
      throttle_left_motor_memory *= -1;
    }
    else PORTD |= 0b11011111;
  }
  else if(throttle_counter_left_motor == 1)PORTD |= 0b01000000;
  else if(throttle_counter_left_motor == 2)PORTD &= 0b10111111;

  throttle_counter_right_motor ++;
  if(throttle_counter_right_motor > throttle_right_motor_memory){
```

```
    throttle_counter_right_motor = 0;
    throttle_right_motor_memory = throttle_right_motor;
    if(throttle_right_motor_memory < 0){
      PORTD |= 0b11110111;
      throttle_right_motor_memory *= -1;
    }
    else PORTD &= 0b00001000;
  }
  else if(throttle_counter_right_motor == 1)PORTD |= 0b00010000;
  else if(throttle_counter_right_motor == 2)PORTD &= 0b11101111;
}
```

LAMPIRAN B

Program raspberry pi :

```
//g++ -ggdb %f -o %e $(pkg-config --libs --cflags opencv ) -lwiringPi
#include <iostream>
#include <vector>
#include "opencv2/opencv.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/core/core.hpp"
#include <wiringPi.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <wiringSerial.h>
using namespace cv;
using namespace std;
int main ()
{
  int fd ;
  int i;
  if ((fd = serialOpen ("/dev/ttyUSB0", 115200)) < 0)
  {
    fprintf (stderr, "Unable to open serial device: %s\n", strerror (errno)) ;
    return 1 ;
  }

  if (wiringPiSetup () == -1)
  {
    fprintf (stdout, "Unable to start wiringPi: %s\n", strerror (errno)) ;
    return 1 ;
  }
cv::VideoCapture capWebcam(0);                  // declare a VideoCapture
object to associate webcam, 0 means use 1st (default) webcam

if (capWebcam.isOpened() == false)              //  To check if object was
associated to webcam successfully
            {
```

```
std::cout << "error: Webcam connect unsuccessful\n";          // if not
then print error message
return(0);
                                                    // and exit program
}

cv::Mat imgOriginal;                // Input image
cv::Mat hsvImg;                             // HSV Image
cv::Mat threshImg;                          // Thresh Image

std::vector<cv::Vec3f> v3fCircles;          // 3 element vector of
floats, this will be the pass by reference output of HoughCircles()

char charCheckForEscKey = 0;

int lowH = 38;                      // Set Hue
int highH = 75;

int lowS = 200;                             // Set Saturation
int highS = 255;

int lowV = 102;                             // Set Value
int highV = 225;

while (charCheckForEscKey != 27 && capWebcam.isOpened()) {

bool blnFrameReadSuccessfully = capWebcam.read(imgOriginal);

if (!blnFrameReadSuccessfully || imgOriginal.empty()) {
                    // if frame read unsuccessfully
std::cout << "error: frame can't read \n";
                    // print error message
break;
jump out of loop
                    }

cv::cvtColor(imgOriginal, hsvImg, CV_BGR2HSV); // Convert
Original Image to HSV Thresh Image
```

```
cv::inRange(hsvImg, cv::Scalar(lowH, lowS, lowV), cv::Scalar(highH,
highS, highV), threshImg);

cv::GaussianBlur(threshImg, threshImg, cv::Size(3, 3), 0);
        //Blur Effect
cv::dilate(threshImg, threshImg, 0);            // Dilate Filter Effect
cv::erode(threshImg, threshImg, 0);             // Erode Filter Effect
// fill circles vector with all circles in processed image
cv::HoughCircles(threshImg,v3fCircles,CV_HOUGH_GRADIENT,2,th
reshImg.rows / 4,100,50,10,800);  // algorithm for detecting circlesfor
(int i = 0; i < v3fCircles.size(); i++) {
                // for each circle
 std::cout << "Ball position X = "<< v3fCircles[i][0]
        // x position of center point of circle
<<",\tY = "<< v3fCircles[i][1]
        // y position of center point of circle
<<",\tRadius = "<< v3fCircles[i][2]<< "\n";
        // radius of circle


// draw small green circle at center of object detected
cv::circle(imgOriginal,             // draw on original image
cv::Point((int)v3fCircles[i][0], (int)v3fCircles[i][1]),        // center
point of circle
3,      // radius of circle in pixels
cv::Scalar(0, 255, 0),
                                    // draw green
CV_FILLED);
// draw red circle around object detected
cv::circle(imgOriginal,
                                        // draw on original
image
cv::Point((int)v3fCircles[i][0], (int)v3fCircles[i][1]),        // center
point of circle
(int)v3fCircles[i][2],
                                    // radius of circle in pixels
cv::Scalar(0, 0, 255),3);

                // thickness
```

```
        if (v3fCircles[i][0] > 190 && v3fCircles[i][0] < 450 &&
v3fCircles[i][2] < 40) {
        int i = 1;
        std::cout << "1"<<"\n";
        serialPutchar (fd, i) ;
        }
        if (v3fCircles[i][0] < 190) {
        int i = 2;
        std::cout << "2"<<"\n";
        serialPutchar (fd, i) ;
        }
        if (v3fCircles[i][0] > 450) {
        int i = 3;
        std::cout << "3"<<"\n";
        serialPutchar (fd, i) ;
        }
        if (v3fCircles[i][0] > 190 && v3fCircles[i][0] < 450 &&
v3fCircles[i][2] > 60) {
        int i = 4;
        std::cout << "4"<<"\n";
        serialPutchar (fd, i) ;
        }
        if (v3fCircles[i][0] > 190 && v3fCircles[i][0] < 450 &&
v3fCircles[i][2] > 40 && v3fCircles[i][2] < 60) {
        int i = 5;
        std::cout << "5"<<"\n";
        serialPutchar (fd, i) ;
        }
                        }


// declare windows
cv::namedWindow("imgOriginal", CV_WINDOW_AUTOSIZE);
cv::namedWindow("threshImg", CV_WINDOW_AUTOSIZE);
 /* Create trackbars in "threshImg" window to adjust according to object
and environment.*/
cv::createTrackbar("LowH", "threshImg", &lowH, 179);          //Hue (0 -
179)
cv::createTrackbar("HighH", "threshImg", &highH, 179);
```

```
cv::createTrackbar("LowS", "threshImg", &lowS, 255);
        //Saturation (0 - 255)
cv::createTrackbar("HighS", "threshImg", &highS, 255);

cv::createTrackbar("LowV", "threshImg", &lowV, 255);        //Value
(0 - 255)
cv::createTrackbar("HighV", "threshImg", &highV, 255);


cv::imshow("imgOriginal", imgOriginal);
        // show windows
cv::imshow("threshImg", threshImg);

charCheckForEscKey = cv::waitKey(1);
// delay and get key press
        }

        return(0);

}
```

[Halaman ini sengaja dikosongkan]

RIWAYAT HIDUP PENULIS

Rizal Aulia Ramadhan lahir pada 15 Februari 1995 di Kota Surabaya, Jawa Timur, merupakan anak kedua dari tiga bersaudara dari pasangan Bapak Lasiman dan Ibu Endah Poliyandari. Setalah menyelesaikan pendidikan di sekolah dasar pada tahun 2007 di SD Al-Irsyad Surabaya, sekolah menengah pertama pada tahun 2010 di SMPN 2 Surabaya, dan sekolah menegah atas pada tahun 2013 di SMAN 9 Surabaya. Kemudian pada tahun 2014 melanjutkan pendidikan tinggi di Diploma III Teknik Elektro ITS dan berkonsentrasi pada Bidang Studi Komputer Kontrol. Penulis meneruskan pendidikan Lintas Jalur S1 di Institut Teknologi Sepuluh Nopember, Surabaya di departemen Teknik Elektro, Fakultas Teknnologi Elektro. Penulis pada bulan Juni 2019 mengikuti ujian Tugas Akhir di Teknik Elektro ITS Surabaya sebagai salah satu persyaratan untuk memperoleh gelar Sarjana Teknik Elektro.

Email : rizalbbone15@gmail.com
Hp / WA : 089622266874
Facebook : Rizal Aulia Ramadhan
Line : rizalbone95

[Halaman ini Sengaja dikosongkan]