RANCANG BANGUN SELF BALANCING ROBOT LINE FOLLOWER MENGGUNAKAN MIKROKONTROLER ARDUINO NANO

LAPORAN TUGAS AKHIR

LAPORAN INI DISUSUN UNTUK MEMENUHI SALAH SATU SYARAT MEMPEROLEH GELAR S1 PADA PRODI TEKNIK ELEKTRO FAKULTAS TEKNOLOGI INDUSTRI UNIVERSITAS ISLAM SULTAN AGUNG SEMARANG

DISUSUN OLEH :

ABI TEGUH RAHAYU
NIM 30601601821

PROGRAM STUDI TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM SULTAN AGUNG SEMARANG
2021

FINAL PROJECT

DESIGN SELF BALANCING ROBOT LINE FOLLOWER USE MICROCONTROLLER ARDUINO NANO

Proposed to complete the requirement to obtain a bachelor's degree (S1) at Department of Electrical Engineering, Faculty of Industrial Technology, Universitas Islam Sultan Agung

By :

ABI TEGUH RAHAYU

NIM 30601601821

DEPARTMENT OF ELECTRICAL ENGINEERING

FACULTY OF INDUSTRIAL TECNOLOGY

UNIVERSITAS ISLAM SULTAN AGUNG

SEMARANG

2021

LEMBAR PENGESAHAN PEMBIMBING

Laporan Tugas Akhir dengan judul "**RANCANG BANGUN SELF BALANCING ROBOT LINE FOLLOWER MENGGUNAKAN MIKROKONTROLER ARDUINO NANO**" ini disusun oleh:

Nama	: ABI TEGUH RAHAYU
NIM	: 30601601821
Program Studi	: Teknik Elektro

Telah disahkan dan disetujui oleh dosen pembimbing pada:

Hari	: Jum'at
Tanggal	: 13 Agustus 2021

Pembimbing I	Pembimbing II
Bustanul Arifin S.T.,M.T.	Muhammad Khosyi'in S.T.,M.T
NIDN. 0614117701	NIDN. 0625077901

Mengetahui,
Ka. Program Studi Teknik Elektro

Jenny Putri Hapsari, ST, MT.
NIDN. 0607018501

LEMBAR PENGESAHAN PENGUJI

Laporan Tugas Akhir dengan judul "**RANCANG BANGUN SELF BALANCING ROBOT LINE FOLLOWER MENGGUNAKAN MIKROKONTROLER ARDUINO NANO**" ini telah dipertahankan di depan Penguji sidang Tugas Akhir pada:

Hari : Jum'at

Tanggal : 13 Agustus 2021

Tim Penguji	Tanda Tangan
Ir. Budi Pramono Jati, MM., MT **NIDN. 0623126501** **Ketua**	.....
Agus Suprajitno S.T.,M.T **NIDN. 0602047301** **Penguji I**	.....
Jenny Putri Hapsari S.T.,M.T **NIDN. 0607018501** **Penguji II**	.....

HALAMAN PERSEMBAHAN

Persembahan:

Pertama,

Laporan Tugas Akhir ini saya persembahkan kepada Kedua Orang Tua saya yang sangat saya cintai (Bapak Marlan dan Ibu Siti Utari) yang sudah membesarkan saya dan menjadi motivasi dalam hidup saya dalam menyelesaikan Tugas Akhir ini.

Kedua,

Kepada Kakak saya (Arif Setiawan) dan Adik Saya (Ayu Pratiwi) yang selalu menyemangati saya dalam menyelesaikan Tugas Akhir ini.

Ketiga,

Kepada Pembimbing saya (Bustanul Arifin S.T., M.T. dan Muhammad Khosyi'in S.T., MT.) yang telah membimbing dan memberikan saya arahan dalam pembuatanLaporan Tugas Akhir ini.

Keempat,

Kepada Dosen Fakultas Teknologi Industri Program Studi Teknik Elektro yang senantiasa membimbing saya dan memberikan saya banyak ilmu yang bermanfaat. Tidak lupa juga kepada Teman Seperjuangan Teknik Elektro Angkatan 2016.

SURAT PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan dibawah ini:

Nama	: ABI TEGUH RAHAYU
NIM	: 30601601821
Jurusan	: Teknik Elektro
Fakultas	: Fakultas Teknologi Industri

Dengan ini saya menyatakan bahwa Tugas Akhir yang diajukan dengan judul **"RANCANG BANGUN SELF BALANCING ROBOT LINE FOLLOWER MENGGUNAKAN MIKROKONTROLER ARDUINO NANO"** adalah hasil karya sendiri, tidak pernah diajukan untuk memperoleh gelar kesarjanaan di perguruan tinggi lain maupun ditulis dan diterbitkan orang lain, kecuali secara tertulis diacu dalam daftar pustaka. Tugas Akhir ini adalah milik saya segala bentuk kesalahan dan kekeliruan dalam Tugas Akhir ini adalah tanggung jawab saya.

Demikian surat pernyataan ini saya buat dengan sadar dan penuh tanggung jawab.

Semarang, 21 Agustus 2021

Yang Menyatakan

Abi Teguh Rahayu

PERNYATAAN PERSETUJUAN PUBLIKASI ILMIAH

Saya yang bertanda tangan di bawah ini :

Nama	: Abi Teguh Rahayu
NIM	: 30601601821
Program Studi	: Teknik Elektro
Fakultas	: Teknologi Industri

Dengan ini menyatakan Karya Ilmiah berupa Tugas Akhir dengan Judul : **RANCANG BANGUN SELF BALANCING ROBOT LINE FOLLOWER MENGGUNAKAN MIKROKONTROLER ARDUINO NANO**

Menyetujui menjadi hak milik Universitas Islam Sultan Agung serta memberikan Hak Bebas Royalti Non-Eksklusif untuk disimpan, dialihmediakan, dikelola dalam pangkalan data dan publikasikan di internet dan media lain untuk kepentingan akademis selama tetap mencantumkan nama penulis sebagai pemilik hak cipta. Pernyataan ini saya buat dengan sungguh-sungguh. Apabila dikemudian hari terbukti ada pelanggaran Hak Cipta/ Plagiatisme dalam karya ilmiah ini, maka segala bentuk tuntutan hukum yang timbul akan saya tanggung secara pribadi tanpa melibatkan pihak Universitas Islam Sultan Agung.

Semarang, 21 Aagustus 2021

Yang Menyatakan

Abi Teguh Rahayu

MOTTO

"Tidak perlu bergantung pada orang lain, karena teman terbaik adalah diri sendiri"

KATA PENGANTAR

Bissmillahirrahmanirrahim

Assalamu'alaikum Wr.Wb

Puja dan puji skukur penulis ucapkan kepada Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas akhir dengan judul **"RANCANG BANGUN SELF BALANCING ROBOT LINE FOLLOWER MENGGUNAKAN MIKROKONTROLER ARDUINO NANO"**.

Penyusunan Tugas Akhir adalah salah satu syarat untuk memperoleh gelar sarjana pada Fakultas Teknologi Industri Universitas Islam Sultan Agung. Banyak pihak yang berjasa dalam pengerjaan Tugas Akhir ini, sehingga penulis ingin menyampaikan ucapan terimakasih kepada semua pihak yang membantu dalam pengerjaan Tugas Akhir ini baik berupa dorongan moril dan materil. Oleh karena itu penulis mengucapkan terimakasih kepada :

1. Dr. Novi Marlyana, S.T,. M.T. selaku Dekan Fakultas Teknologi Indusri Universitas Islam Sultan Agung Semarang.
2. Jenny Putri Hapsari S.T,. M.T. selaku Ketua Prodi Teknik Elektro Fakultas Teknologi Industri Universitas Islam Sultan Agung Semarang.
3. Bustanul Arifin, ST, MT selaku Dosen Pembimbing I yang telah memberikan bimbinngan dan dorongan dalam penyusunan Tugas Akhir ini.
4. Muhammad Khosyi'in, ST, MT selaku Dosen Pembimbing II yang telah memberikan bimbingan dan dorongan dalam penyusunan Tugas Akhir ini.
5. Bapak dan Ibu dosen Fakultas Teknologi Industri Universitas Islam Sultan Agung Semarang selaku tenaga pengajar telah bersedia berbagi ilmu yang bermanfaat sehingga penulis memperoleh pengetahuan dan pengalaman selama menempuh studi.
6. Bapak dan Ibu yang selalu mendoakan, menyemangati dan memberikan motivasi serta nasihat yang menenangkan hati serta memberikan dukungan

dari awal baik dukungan moral maupun materi. Terimakasih karena telah menjadi orang tua yang sangat baik.

7. Arif Setiawan kakak saya yang selalu memberikan semangat, doa, serta memberikan dukungan moral maupun materil kepada penulis.
8. Teman-teman seperjuangan saya Teknik Elektro 2016, terimakasih sudah memberikan kritik, saran, kebahagian, dan kekeluargaan selama ini.

Penulis menyadari bahwa ini masih jauh dari kata sempurna, maka dari itu penulis mengharapkan kritik dan saran dari berbagai pihak guna untuk menyempurnakan Tugas Akhir ini. Penulis berharap penelitian ini dapat bermanfaat bagi perkembangan ilmu elektro.

Terimakasih. Wassalamualaikum Wr. Wb

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Self Balancing robot Line follower adalah sebuah robot yang memiliki kemampuan untuk mempertahankan posisi berdiri dengan dua roda dengan mengendalikan sudut kemiringannya serta dapat berjalan mengikuti garis hitam pada background putih. Untuk itu dibutuhkan pengontrolan yang baik untuk menjaga posisi tegak tanpa perlu penyangga.

Salah satu metode kontrol yang dapat digunakan untuk mengendalikan keseimbangan robot adalah PID (Proportional Integral Derifative). Kelebihan dari teknik kendali ini adalah pada fleksibelitasnya untuk dapat diterapkan pada berbagai macam sistem kendali.

Maka dari itu pada Tugas Akhir ini menggunakan metode PID. Proses pengontrolan ini diprogram dengan menggunnakan software arduino IDE dan hasilnya dikirimkan ke motor DC untuk mengatur arah putar motor untuk menyeimbangkan robot. Metode kontrol PID yang digunakan pada penelitian ini yaitu dengan menggunakan metode PID Trial and Error. dan diperoleh nilai Kp=40, Ki=270, Kd=1,9.

Kata kunci: Self Balancing, Robot, Line Follower, PID

ABSTRACT

Self Balancing robot Line follower is a robot that has the ability to maintain a standing position on two wheels by controlling the angle of inclination and can walk along a black line on a white background. For this reason, good control is needed to maintain an upright position without the need for a support.

One of the control methods that can be used to control the balance of the robot is PID (Proportional Integral Derifative). The advantage of this control technique is its flexibility to be applied to a variety of control systems.

Therefore, in this final project using the PID method. This control process is programmed using Arduino IDE software and the results are sent to a DC motor to adjust the direction of rotation of the motor to balance the robot. The PID control method used in this study is the PID Trial and Error method. and obtained the value of Kp = 40, Ki = 270, Kd = 1.9.

Keywords: Self Balancing, Robot, Line Follower, PID

BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Baru-baru ini, ilmu robotika telah menarik perhatian banyak kalangan, contohnya mahasiswa, dosen atau guru, serta peneliti untuk mengembangkan suatu alat yang dapat mempermudah pekerjaan manusia atau hanya untuk sekedar hobi. Untuk menambah tingkat kecerdasan sebuah robot salah satunya adalah dengan cara menambahkan sensor, mikroprosesor, dan suatu kecerdasan buatan pada robot. Salah satu contohnya adalah *self balancing* robot.

Self Balancing Robot adalah sebuah robot yang memiliki kemampuan untuk mempertahankan posisi berdiri dengan dua roda dengan mengendalikan sudut kemiringannya. *Self balancing* robot membutuhkan pengontrolan yang baik untuk menjaga dirinya pada posisi tegak tanpa perlu penyangga. Supaya robot dapat menyeimbangkan diri berdiri tegak lurus pada bidang datar, diperlukan perangkat *hardware* yang baik dan metode control yang baik.

Salah satu metode kontrol yang dapat digunakan untuk mengendalikan keseimbangan robot adalah PID (Proportional Integral Derifative). Kelebihan dari teknik kendali ini adalah pada fleksibelitasnya untuk dapat diterapkan pada berbagai macam sistem kendali.[1]

Pada era modern seperti ini dibutuhkan suatu metode pembelajaran yang menarik, tidak membosankan, dan mudah dimengerti bagi banyak kalangan. Tujuan dibuatnya Tugas Akhir ini adalah untuk mempermudah bagi pelajar dalam memahami apa itu PID, bagaimana penerapannya, dan perhitungan PID.

Maka dari itu dibuatlah Tugas Akhir yang berjudul Rancang Bangun *Self Balancing Robot Line Follower* Menggunakan Mikrokontroler Arduino Nano. dengan menggunakan metode PID dan sebagai pengembangan dari penelitian sebelumnya.

1.2. Perumusan Masalah

1. Bagaimana *self balancing robot line follower* dapat berdiri seimbang dan mengikuti garis?
2. Bagaimana rancangan sistem PID pada *self balancing robot line follower*?

1.3. Pembatasan Masalah

1. Desain robot adalah *prototype* robot dengan dua roda dengan konfigurasi paralel.
2. Sensor yang digunakan adalah MPU-6050 untuk identifikasi sudut kemiringan dan sensor *photodioda* untuk identifikasi garis.
3. Robot mengituti garis hitam dengan lebar 1 cm pada lantai putih.
4. Metode yang digunakan untuk *Self Balancing Robot Line Follower* adalah PID.

1.4. Tujuan

1. Mengetahui cara merancang *self balancing robot line follower* menggunakan mikrokontroler arduino nano.
2. Dapat membuat *Self Balancing Robot Line Follower* dapat berdiri dan mengikuti Garis.
3. Mengetahui nilai PID yang sesuai dan perancangan sistem PID pada *self balancing robot line follower* menggunakan mikrokontroler arduino nano.

1.5. Manfaat

1. Dapat mengetahui penerapan metode PID pada *self balancing robot line follower* menggunakan mikrokontroler arduino nano.
2. Dapat digunakan sebagai referensi pada penelitian selanjutnya yang berkaitan

1.6. Sistematika Penulisan Laporan

Dalam penulisan Tugas Akhir ini, sistematika penulisan yang digunakan adalah sebagai berikut :

BAB I : PENDAHULUAN

Pada bab ini membahas tentang latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, sistematika penulisan, dan manfaat tugas akhir.

BAB II : LANDASAN TEORI

Pada bab ini berisi tentang teori – teori dasar yang berhubungan dengan PID, komponen – komponen yang digunakan dalam pembuatan *Self Balancing Robot Line Follower* secara umum.

BAB III : PERANCANGAN SISTEM

Dalam bab ini berisi gambaran umum tempat penelitian, data penelitian, prosedur/tahapan penelitian serta metode penelitian yang digunakan dalam pembuatan *Self Balancing Robot Line Follower*

BAB IV : HASIL DAN ANALISA

Bab ini berisi tentang semua hasil penelitian yang dilakukan dan pembahasanya yang meliputi perhitungan – perhitungan yang mengatur jalannya robot.

BAB V : PENUTUP

Bab ini membahas kesimpulan hasil penelitian yang telah di lakukan dan saran – saran yang di berikan peneliti berdasarkan kesimpulan.

BAB II
LANDASAN TEORI

2.1. Tinjauan Pustaka

Pada penelitian yang dilakukan oleh Raranda Dan Puput Wanarti Rusimamto dari Program Studi Teknik Elektro Universitas Negeri Surabaya pada tahun 2015 dengan judul Implementasi Kontroler Pid Pada *Two Wheels Self Balancing Robot* Berbasis Arduino Uno didapatkan hasil Sistem ini menggunakan *input* dari sensor IMU (*Inertial Measurement Unit*). *Output* dari sensor tersebut berupa sudut yang dikirim ke Arduino UNO. Sudut yang didapat dibandingkan dengan nilai *setpoint* yang nilainya 0 derajat. Nilai selisih dari setpoint dan sudut keluaran sistem dikontrol menggunakan kendali PID. Proses kendali ini diprogram pada *software* Arduino IDE yang hasilnya dikirim ke motor DC untuk mengatur arah putaran motor DC untuk menyeimbangkan bodi robot. Dari hasil pengujian diperoleh nilai parameter kontroler PID yang akan digunakan dari *tunning* nilai Kcr dengan metode *Ziegler-Nichols* adalah Kp = 70, Ki = 466.67 dan Kd = 2.625 dapat mengatasi keseimbangan pada *self balancing robot* mendekati nilai *setpoint*. Dengan nilai *time response* kurang dari 1.5 *second* saat diberikan gangguan. [1]

Pada penelitian yang dilakukan oleh Adetiya Pratama Dari Program Studi Teknik Elektro Universitas Teknologi Yogyakarta pada tahun 2018 dengan judul Implementasi Pid *Controller* Pada *Self Balancing Robot* di dapatkan kesimpulan Sistem ini mendapatkan masukan dari sensor MPU-6050 yaitu gabungan antara akselerometer dan giroskop. Sistem pengendalian yang digunakan yaitu metode kendali PID (*Proprotional Integral dan Derivative*). Proses kendali PID ini dilakukan dengan membuat suatu kode program menggunakan *software* Arduino IDE yang nantinya dimasukkan ke Arduino Uno yang hasilnya digunakan untuk mengontrol kecepatan dan arah putaran motor DC, serta set point nilai kemiringan sudut dari robot. Motor DC akan berputar ke depan dan belakang apabila sudut yang diperoleh lebih dari nilai *set point*. Nilai konstanta PID yang didapat berdasarkan *tuning trial and error*.[2]

Penelitian yang dilakukan oleh Aqsha Adella, Muammad Kamal, dan Aidi Finawan mahasiswa Teknik Elektro Universitas politeknik Negeri Lhokseumawe pada tahun 2018 dengan judul Rancang Bangun *Robot Mobile Line Follower* Pemindah Minuman Kaleng Berbasis Arduino. Di dapatkan kesimpulan *Robot Line Follower* Pemindah barang ini dapat bergerak mengikuti garis dengan baik, digunakan 6 buah sensor untuk membaca posisi garis apakah robot sudah lurus atau posisi serong, untuk mengatasi eror dalam proses robot mengikuti garis. Setiap sensor garis berbeda-beda nilai teganganya Dengan ini dapat disimpulkan bahwa robot ini dapat berjalan dengan baik pada saat membaca warna minuman kaleng dan menempatkan kaleng tersebut sesuai dengan tempatnya dan manfaat penggunaan robot dalam penyortiran akan lebih efisien dan efektif.[3]

Menurut hasil penelitian oleh Koko Joni, Miftachul Uklum, dan Zainal Abidin dari Program Studi Teknik Elektro Universitas Trunojoyo Madura Yang Berjudul *Robot Line Follower* Berbasis Kendali *Proportional-Integral-Derifative* (PID) Untuk Lintasan Dengan Sudut Ekstrim didapatkan kesimpulan Robot ini menggunakan 8 buah sensor dengan asumsi 1 atau 2 sensor menyentuh garis lintasan dengan tebal 2 cm. Pada percobaan max. speed PWM 100 didapat tunning PID terbaik yaitu Kp = 3, Ki = 3 dan Kd = 0, menghasilkan waktu mencapai finish untuk lapangan A, B dan C masing- masing sebesar 3,59 s, 3,10 s dan 3,69 s. Sedangkan untuk max. speed PWM 255 didapat tunning PID terbaik yaitu Kp = 5, Ki = 5 dan Kd = 5, menghasilkan waktu mencapai finish untuk lapangan A, B dan C masing- masing sebesar 2,67 s, 2,46 s dan 2,78 s. Kecepatan maksimum sangat berpengaruh terhadap hasil tuning PID untuk lintasan extrim, hal ini dikarenakan ketika kecepatan tinggi membutuhkan sistem PID dengan respon time yang relatif cepat dan stabil. Sehingga pengaturan PID untuk kecepatan PWM tertentu tidak dapat diterapkan untuk kecepatan PWM yang lain. [4]

Menurut hasil penelitian oleh Amaludin Jatmiko dari Program Studi Teknik Elektro Universitas Islam Sultan Agung yang berjudul *Prototype* Robot *Balancing*/Keseimbangan Dengan Kendali PID. Pada penelitian yang dilakukan menggunakan metode PID *trial and error*. *Tunning* PID yang diperoleh pada

penelitian ini yaitu Kp = 22 didapatkan respon motor yang sedang, Ki = 66 didapatkan ayunan atau goyangan sangat lambat agar tidak mengalami *shaking* yang berlebihan, dan Kd = 0.6 didapatkan robot mengalami ayunan atau goyangan yang jaraknya sedikit.[5]

Dari penelitian di atas mempunyai prinsip kerja yang berbeda. Pada penelitian [1],[2],[5] merupakan penelitian tentang pembuatan *self balancing robot* dengan dua roda menggunakan metode PID. Sehingga robot dapat menyeimbangkan diri dengaan baik. Kelemahan dari robot pada penelitian ini yaitu hanya dapat menyeimbangkan diri di tempat tanpa berjalan. Sedangkan pada penelitian [3],[4] adalah robot pengikut garis atau biasa disebut *robot line follower*. Pada penelitian [3] kelebihan *line follower* ini yaitu dapat mengantarkan minuman sesuai dengan warna yaitu warna merah, biru, dan hijau. Pada penelitian [4] menggunakan metode PID agar robot dapat melewati garis atau lintasan dengan sudut yang ekstrim.

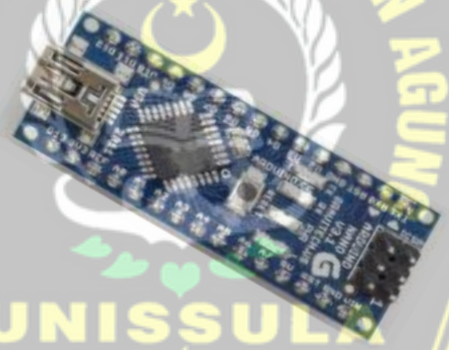
Berdasarkan penelitian di atas maka dibuatlah robot yang menggabungkan prinsip kerja dari keduanya sebagai pengembangan dari penelitian tersebut. Sehingga Tugas Akhir ini berjudul "RANCANG BANGUN *SELF BALANCING ROBOT LINE FOLLOWER* MENGGUNAKAN MIKROKONTROLER ARDUINO NANO".

2.2. Arduino

Arduino adalah papan rangkaian elektronika yang sudah dibentuk sedemikian rupa yang di dalamnya terdapat mikrokontroler AVR yang dapat diprogram dengan mudah melalui komputer dengan menggunakan *sorfware* Arduino IDE. Arduino bersifat *open source wiring*, sehingga setiap orang dapat membuat atau memproduksi sendiri. saat ini arduino sangat populer di dunia robotika. Banyak pemula yang menggunakan arduino untuk belajar elektronika dan robotika karena mudah dipelajari. Bahasa yang digunakan arduino juga mudah untuk dipahami. Arduino menggunakan bahasa C yang disederhanakan dengan bantuan *libraries* arduino. Salah satunya adalah Arduino Nano. Arduino nano adalah *board* mikrokontroler arduino yang berukuran kecil yang menggunakan Atmega 328 untuk Arduino Nano 3.x dan menggunakan Atmega 168 untuk Arduino Nano 2.x.

Arduino Nano memiliki rangkaikan yang sama dengan Arduino Duemilanove, tetapi dengan rangkaian dan desain PCB yang berbeda. Arduino Nano memiliki port yang lebih sedikit dibandingkan dengan arduino lain. Arduino Nano tidak memiliki soket catu daya, tetapi terdapat pin untuk catu daya luar, atau dapat menggunakan mini USB port sebagai catu dayanya.

ATmega168 dan Atmega328 dapat melakukan komunikasi serial UART TTL (5V) yang tersedia pada digital pin 0 (Rx) dan 1 (Tx). FTDI FT232RL pada *board* menjadi saluran komunikasi serial menggunakan USB dan FTDI *driver* (tergabung dengan *software Arduino*) untuk berkomunikasi dengan komputer memanfaatkan *virtual com port.*[6] Spesifikasi Arduino Nano dapat dilihat pada Tabel 2.1.



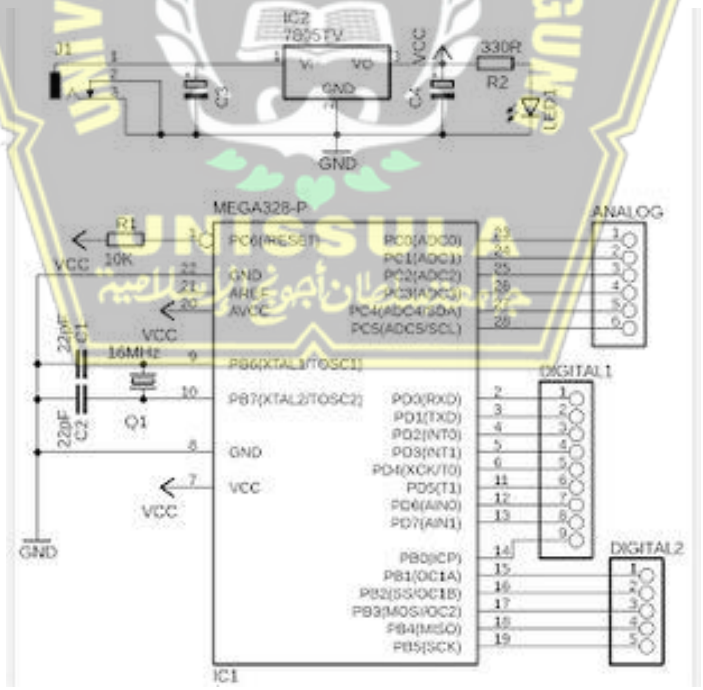
Gambar 2.1 Arduino Nano[7]

Tabel 2. 1 Spesifikasi Arduino Nano[7]

Mikrokontroler	*Atmega 168 atau Atmega 326*
Tegangan kerja	5V
Tegangan *Input*	7V-12V
Pin I/O Digital	14 (dengan 6 PIN PWM)
Pin *Input* Analog	8
Arus DC maksimal	40 mA (per I/O pin)

Tabel 2.1 lanjutan

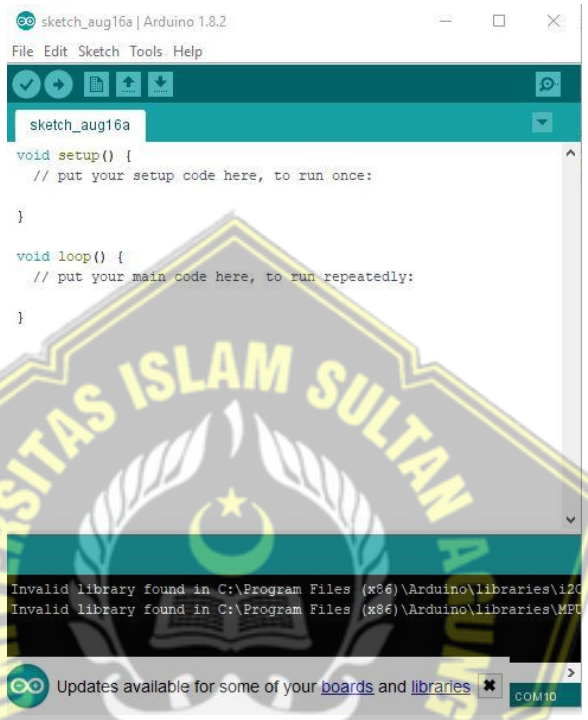
Flash memory	16 kB (ATmega168) atau 32 kB (ATmega328) dengan 2 kB terpakai untuk *bootloader*
SRAM	1 kB (ATmega168) atau 2 kB (ATmega328)
EEPROM	512 *bytes* (ATmega168) atau 1 kB (ATmega328)
Clock Speed	16 MHz
Dimensi	0.73” x1.70’’
Panjang	45 mm
Lebar	18 mm
Berat	5 g



Gambar 2. 2 Skema rangkaian arduino nano

(Sumber : arduinoku.wordpress.com)

Pembuatan program arduino dapat dilakukan dengan menggunakan *software Arduino IDE*. Cara pembuatan program pada *software Aarduino IDE* yaitu dengan cara membuka terlebih dahulu *software Arduino IDE*, kemudian tekan *file*, kemudian tekan *new*. Maka akan muncul lembar kerja baru.



Gambar 2. 3 Arduino IDE

Setelah pembuatan program sudah selesai, kemudian cara meng-*upload* program yang sudah dibuat ke arduino, dengan cara tekan menu *Tools*, kemudian pastikan *Board* dan *Port* sudah sesuai. kemudian tekan *upload*

2.3. MPU-6050

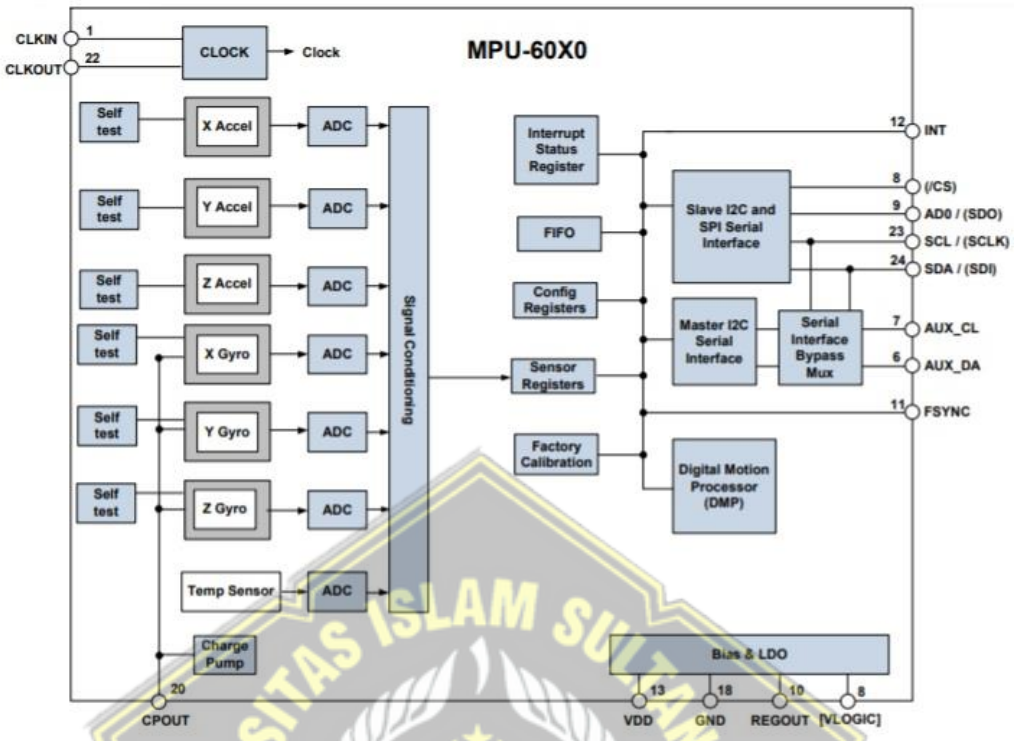
MPU-6050 adalah sebuah modul berinti MPU-6050 yang merupakan 6 *axis Motion Processing Unit* dengan penambahan regulator tegangan dan beberapa komponen pelengkap lainnya yang membuat modul ini siap pakai dengan tegangan *supply* sebesar 3-5 VDC modul MPU-6050 berisi MEMS (*Microelectromechanical Systems*) berisi *gyroscop* dan *accelerometer* pada satu *chip* yang kecil.*Gyroscop* berfungsi untuk menentukan orientasi suatu benda berdasarkan sudut, sedangkan *accelerometer* berfungsi untuk mengukur percepatan akibat gravitasi. Modul ini mampu menangkap data x, y, z pada waktu yang bersamaan. Karena memiliki

hardware khusus untuk konversi analog ke digital selebar 16 bit. Sehingga modul ini dapat terbilang akurat. Spesifikasi MPU-6050 dapat dilihat pada Tabel 2.2.

Gambar 2. 4 MPU-6050[2]

Tabel 2. 2 Spesifikasi MPU 6050

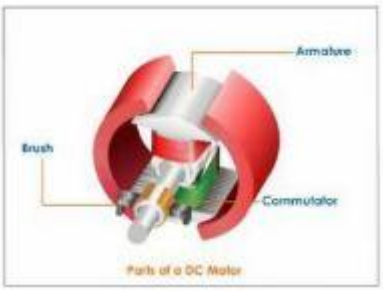
IC	MPU6050
Tegangan operasional	3Vdc – 5Vdc
Komunikasi	I2C (SCL, SDA)
Range dari Gyroscope	250 500 1000 2000 / s
Chip	built-in 16-bit AD converter, 16-bit data *output*
Jumlah pin	8 Pin
Ukuran	2.2cm x 1.7cm



Gambar 2. 5 Diagram Blok MPU-6050

2.4. Motor DC

Motor DC adalah sebuah komponen elektronik yang dapat mengubah arus listrik menjadi mekanik. Pada Tugas Akhir ini penggunaan motor dc sebagai aktuator *self balancing* robot untuk kestabilan keseimbangan robot. Agar robot dapat berdiri tegak, pemilihan motor dc sangatlah penting. Motor dc dengan torsi yang besar dan RPM yang sedang menjadi pilihan yang tepat untuk digunakan pada *self balancing* robot.



Gambar 2. 6 Motor DC[8]

Motor DC juga biasanya disebut sebagai motor arus searah. Terdapat dua terminal pada motor DC dan memerlukan tegangan arus searah untuk menggerakkannya. Motor DC dapat berputar searah jarum jam maupun berlawanan arah jarum jam apabila polaritasnya dibalik.

Motor DC memiliki dua bagian utama, yaitu *stator* dan *rotor*. *Stator* adalah bagian dari motor yang tidak berputar, sedangkan *rotor* adalah, bagian dari motor yang dapat berputar. *Stator* memiliki beberapa bagian, yaitu kerangka magnet, kutub motor, kumparan medan magnet, komutator, dan kuas atau sikat arang.

Jika arus melewati sebuah batang konduktor, maka akan timbul medan magnet di sekitar batang konduktor. Prinsip kerja motor DC dimana medan magnet yang membawa arus mengelilingi konduktor. Aturan genggaman tangan kanan bisa dipakai untuk menentukan arah garis fluks di sekitar konduktor. Genggam konduktor dengan tangan kanan dengan jempol mengarah pada arah aliran arus, maka jari-jari anda akan menunjukkan arah garis fluks. Medan magnet hanya terjadi di sekitar sebuah konduktor jika ada arus mengalir pada konduktor tersebut. Jika konduktor berbentuk U (angker dinamo) diletakkan diantara kutub utara dan selatan yang kuat medan magnet konduktor akan berinteraksi dengan medan magnet kutub.[8]

$$N = \frac{V_{TM} - I_A R_A}{K_\emptyset} \quad (2.\ 1)$$

Keterangan :

N = Kecepatan Putar Motor

V_{TM} = Tegangan Terminal

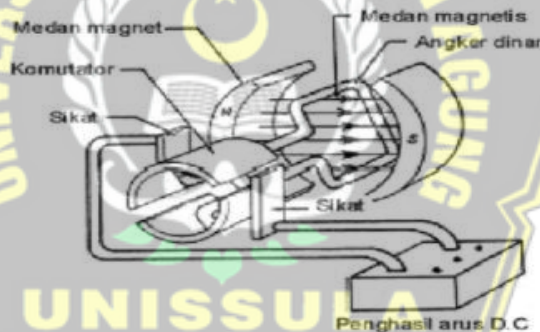
I_A = Arus Jangkar Motor

R_A = Hambatan Jangkar Motor

K = Konstanta Motor

Φ = Fluk magnet yang terbentuk pada motor

Pada motor DC, kumparan medan yang dialiri arus listrik akan menghasilkan medan magnet yang melingkupi kumparan jangkar dengan arah tertentu. Konverter energi baik energi listrik menjadi energi mekanik (motor) maupun sebaliknya dari energi mekanik menjadi energi listrik (generator) berlangsung melalui medium medan magnet. Energi yang akan diubah dari suatu sistem ke sistem yang lain, sementara akan tersimpan pad medium medan magnet untuk kemudian dilepaskan menjadi energi system lainya. Dengan demikian, medan magnet disini selain berfungsi sebagi tempat penyimpanan energi juga sekaligus proses perubahan energi.[8]



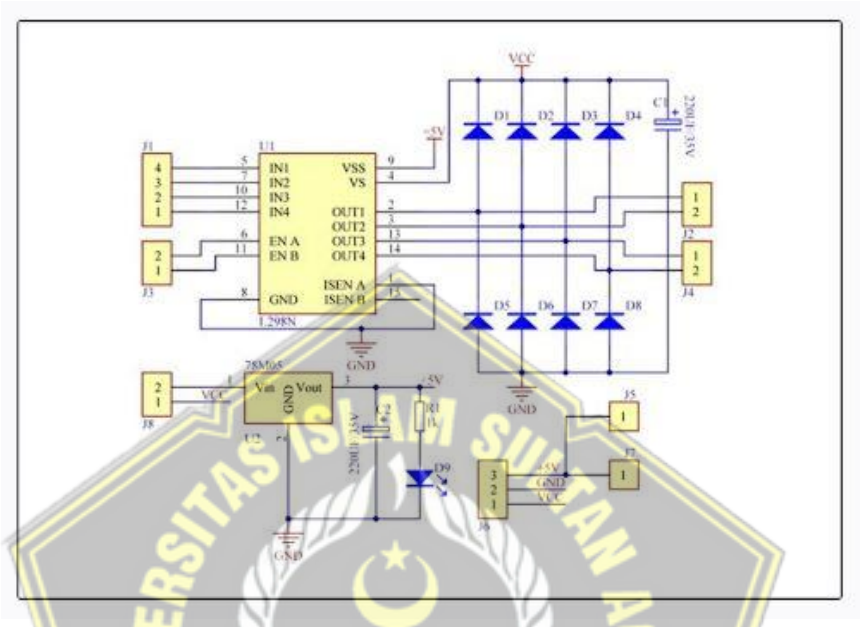
Gambar 2. 7 Bagian Dalam Motor DC

2.5. Driver Motor DC L298N

Motor driver berfungsi sebagai pengatur arah putaran motor maupun kecepatan putaran motor. Driver motor diperlukan untuk board Arduino karena Arduino hanya mampu mengeluarkan arus yang kecil sehingga tidak mampu memenuhi kebutuhan motor DC, sehingga perlu driver motor untuk menyesuaikan tegangan dan arus yang dibutuhkan motor tersebut.[9]

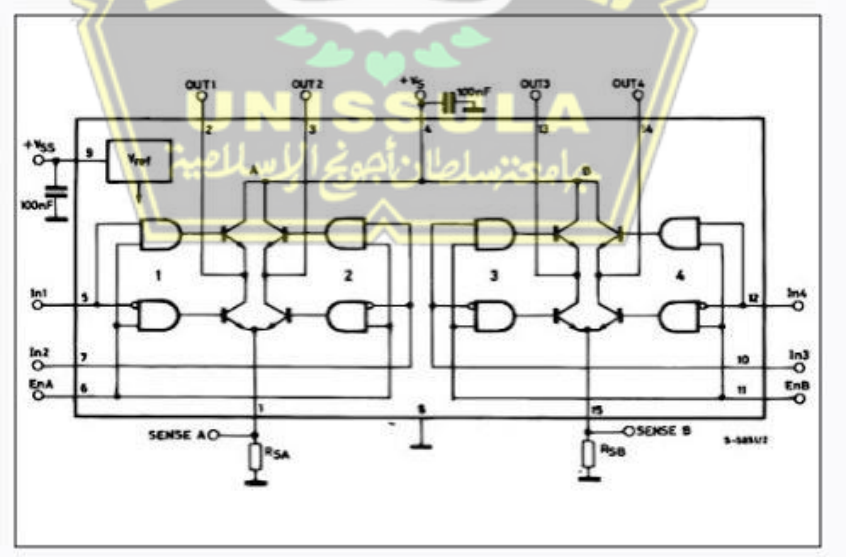
Pada Tugas Akhir ini menggunakan salah satu jenis *driver* motor, yaitu *driver* motor *shield H-bridge L298N module. Driver* motor ini sangat mudah digunakan,

dan sangat banyak dijumai di pasaran. Pada modul ini menggunakan komponen utama yaitu IC L298 dengan tegangan kerja yaitu 12 V. Komponen lengkap dan skema rangkaian dari *Driver* motor L298N dapat dilihat pada gambar 2.8.



Gambar 2. 8 Komponen dan skema rangkaian dari Driver motor L298N

(Sumber : www.mahirelektro.com)



Gambar 2. 9 Diagram Blok L298N

(Sumber : www.mahirelektro.com)

Cara kerja driver motor L298N adalah sebagai berikut :

1. Input1 dan Input 2 digunakan untuk mengontrol Motor 1. Motor hanya akan berputar apabila Enable A diberikan Logika HIGH. Apabila pin enable diberikan logika Low motor tidak akan berputar.
2. Apabila Input 1 High dan Input 2 Low maka motor akan berputar dengan arah tertentu.
3. Apabila Input 1 Low dan Input 2 High maka motor akan berputar kearah sebaliknya.
4. Apabila Input 1 dan 2 logikanya sama High atau Low maka Motor tidak akan berputar.
5. Input3 dan Input 4 digunakan untuk mengontrol Motor 2. Motor hanya akan berputar apabila Enable B diberikan Logika HIGH. Apabila pin enable diberikan logika Low motor tidak akan berputar.
6. Apabila Input 3 High dan Input 4 Low maka motor akan berputar dengan arah tertentu.
7. Apabila Input 3 Low dan Input 4 High maka motor akan berputar kearah sebaliknya.
8. Apabila Input 3 dan 4 logikanya sama High atau Low maka Motor tidak akan berputar.
9. Putaran motor searah jarum jam disebut CW (Clock Wise) sedangkan putaran motor yang berlawanan arah jarum jam disebut CCW (Counter Clock Wise)
10. Capasitor 1 dan 2 berfungsi sebagai Decoupling untung menghilangkan tegangan liar yang berasal dari power supply. Sedangan 8 buah dioda 1N 4007 berfungsi sebagai Proteksi terhadap induksi yang diakibatkan oleh perubahan putaran motor secara tiba-tiba.

2.6. Pulse Width Modulation (PWM)

Pulse Width Modulation (pwm) atau modulasi lebar pulsa, adalah teknik pengubahan sinyal digital berupa gelombang kotak (square wave) dimana duty cycle dari gelombang kotak tersebut dapat diatur sesuai dengan kebutuhan sistem.[10]

Sinyal PWM pada umumnya memiliki amplitudo dan frekuensi dasar yang tetap, namun memiliki lebar pulsa yang bervariasi. Lebar Pulsa PWM berbanding lurus dengan amplitudo sinyal asli yang belum termodulasi. Artinya, Sinyal PWM memiliki frekuensi gelombang yang tetap namun duty cycle bervariasi (antara 0% hingga 100%). Aplikasi PWM berbasis mikrokontroler biasanya berupa pengendalian kecepatan motor DC, Pengendalian Motor Servo, Pengaturan nyala terang LED.[11]

Gambar 2. 10 Pulse Width Modulation

(Sumber : https://teknikelektronika.com)

$$Duty\ cycle = \frac{T_{ON}}{T_{ON} + T_{OFF}} \quad (2.\ 2)$$

Dimana :

T_{ON} = Waktu tegangan keluaran *HIGH* atau 1.

T_{OFF} = Waktu tegangan keluaran *LOW* atau 0.

Untuk mengatur nilai *duty cycle*, kita gunakan fungsi `analogWrite([nomorPin], [nilai])`. Nilai pada parameter kedia berkisar antara 0 hingga 255. Bila kita hendak mengeset duty cycle ke 0%, maka mengatur nilai parameter ke 0, dan untuk *duty cycle* 100%, maka mengatur nilai parameter ke 255.

Jadi apabila ingin mengatur *duty cycle* ke 50%, berarti nilai yang harus diatur adalah 127 (50% x 255).

Sebenarnya berdasarkan konsep PWM di atas, kita dapat mensimulasikan PWM pada semua pin digital. Tapi khusus penggunaan fungsi digitalWrite() kita hanya bisa menggunakannya pada pin-pin PWM. Seperti pada Arduino Uno, pin yang dapat menggunakan fungsi ini hanya pin 3, 5, 6, 9, 10 dan 11. Biasanya pin PWM disimbolkan dengan karakter '~'.

2.7. Sensor Photodioda

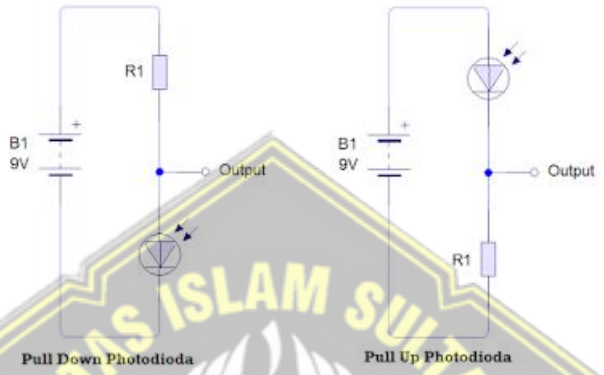
Sensor photodioda adaalah dioda yang peka terhadap cahaya, sensor photodioda akan mengalami perubahan resistansi saat menerima intensitas cahaya dan akan mengalirkan arus listrik secara *forward* seperti prinsip dioda pada umumnya.

Gambar 2. 11 Sensor Photodioda[3]

Prinsip kerja dari photodioda bertolak belakang dengan LED. Tegangan bias mundur dan cahaya yang dipaparkan pada fotodioda akan mengakibatkan timbulnya pasangan elektron-hole di kedua sisi sambungan. Ketika ada cahaya, elektron akan pindah ke jalur konduksi sehingga menyebabkan elektron mengalir ke arah positif sumber tegangan sedangkan hole mengalir ke arah negatif sumber tegangan sehingga arus akan mengalir di dalam rangkaian. Besarnya pasangan elektron-hole yang dihasilkan tergantung dari besarnya intensitas cahaya yang dikenakan pada photodioda.[12]

Sensor photodioda dapat digunakan sebagai sensor cahaya. Rangkaian di bawah ini merupakan rangkaian photodioda yang digunakan sebagai sensor cahaya.

R1 berfungsi sebagai pembatas arus yang masuk pada photodioda dan pembagi tegangan. Sensor phothodioda dapat dibuat dengan konfigurasi *Pull-Up* ataupun *Pull-Down*. Konfigurasi ini mempunyai *output* yang berbeda juka sensor terkena cahaya. Berikut adala gambar rangkaian *pull-down* dan *pull-up* sensor photodioda dapat dilihat pada gambar 2.5.



Gambar 2. 12 Rangkaian *Pull-Up* dan *Pull-Down* Sensor Photodioda

(Sumber : www.andalanelektro.id)

Jika menggunakan *Pull-Up* maka *output* yang dikeluarkan akan berlogika 1 saat mendapatkan cahaya yang cukup. Sebaliknya akan berlogika 0 jika cahaya yangg masuk kurang. Jika menggunakan *Pull-Down* maka *output* akan berlogika 0 jika mendapatkan cahaya yang cukup. Sebaliknya akan berlogika 1 juka cahaya yang masuk kurang.

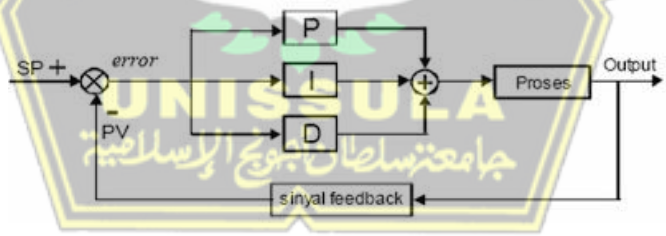
Pada Tugas Akhir ini sensor *photodioda* digunakan sebagai sensor garis. Prinsip kerja *photodioda* sebagai sensor garis adalah dengan memanfaatkan sifat cahaya yang akan dipantulkan jika mengenai benda berwarna terang dan akan diserap jika mengenai benda berwarna gelap. Sebagai sumber cahaya menggunakan led. Prinsip kerja sensor photodioda dapat dilihat pada gambar 2.6.

Gambar 2. 13 Prinsip Kerja Photodioda Sebagai Sensor Garis

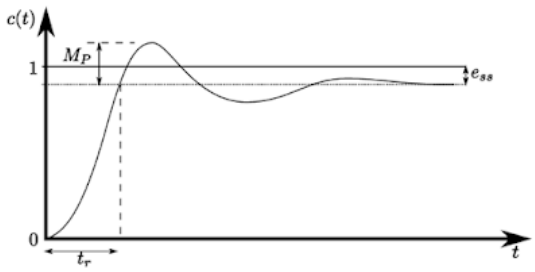
(Sumber : www.andalanelektro.id)

2.8. Kendali PID

Kendali PID (*Proporrtional Integral Derivative*) merupakan suatu sistem kontrol yang menggunakan umpan balik (*feedback*) sebagai penentu presisi suatu sistem instrumensasi. Sistem kontrol PID terdiri dari tiga buah pengaturan yaitu P (*Proportional*), I (*Integral*), dan D (*Derivative*). Pengaturan parameter KP, KI, dan KD akan mempengaruhi kerja sistem, maka dari itu perlu pengaturan parameter yang tepat agar sistem dapat bekerja sesuai dengan yang diinginkan. Berikut adalah gambar diagram blok PID dapat dilihat pada Gambar 2.7.



Gambar 2. 14 Diagram Blok PID[13]



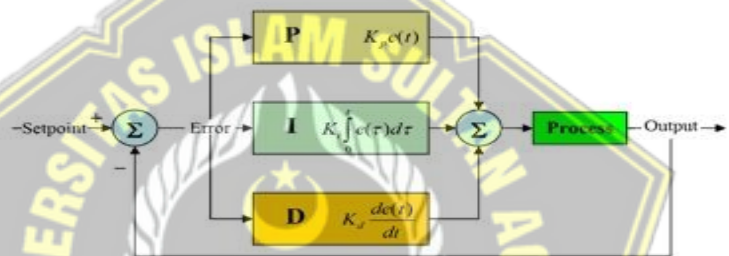
Gambar 2. 15 Respon Sistem PID

(Sumber : www.robotics-university.com)

Dari Gambar 2.15 dapat dijelaskan sebagai berikut:

1. SP = Set Point, secara simple maksudnya ialah suatu prameter nilai acuan atau nilai yang kita inginkan.
2. PV = Present Value, maksudnya ialah nilai bobot pembacaan sensor saat itu atau variabel terukur yang diumpanbalikkan oleh sensor (sinyal feedback dari sensor).
3. Error = nilai kesalahan, yakni deviasi atau simpangan antar variabel terukur atau bobot sensor (PV) dengan nilai acuan (SP).

$$error = SP - PV \tag{2. 3}$$



Gambar 2. 16 Diagram Blok PID

PID dapat dirumuskan sebagai berikut :

$$u(t) = K_P(t) + K_I \int e(t)dt + kd\frac{de(t)}{dt} \tag{2. 4}$$

dengan:

u(t) : sinyal keluaran pengendali PID

Kp : konstanta proporsional

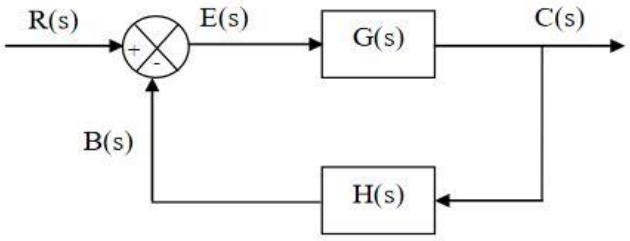
Ki : konstanta integral

Kd : konstanta turunan

e(t) : sinyal kesalahan

Proporsional kontroler (Kp) akan memberikan efek mengurangi waktu naik, tetapi tidak menghapus kesalahan keadaan tunak, Integral kontroler (Ki) akan memberikan efek menghapus keadaan tunak, tetapi berakibat memburuknya respon transien, Diferensial kontroler (Kd) akan memberikan efek meningkatnya stabilitas

sistem, mengurangi *over-shoot*, dan menaikan respon transfer.[13] Efek dari setiap kontroler (Kp, Ki, Kd) dalam sistem *loop* tertutup diperlihatkan pada tabel 2.3.



Gambar 2. 17 Diagram Blok Sistem *Loop* Terturup

R(s) = Referensi sinyal *input*

E(s) = Sinyal *error*

R(s), H(s) = Fungsi Transfer

B(s) = Sinyal *Feedback*

C(s) = Sinyal *output*

Hubungan *input* dan *output* berdasarkan diagram blok diatas yaitu :

$$C(s) = G(s)E(s)$$

$$E(s) = R(s) - B(s)$$

$$= R(s) - H(s)C(s)$$

Eliminasi E(s) sehingga akan didapat persamaan

$$C(s) = G(s)[R(s) - H(s)C(s)]$$

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)}$$

$$C(s) = \frac{G(s)}{1+G(s)H(s)} \quad (2.5)$$

Tabel 2. 3 Efek Dari Setiap Kontroler (Kp,Ki,Kd) Dalam Loop Tertutup

Respon Loop Tertutup	Waktu Naik	Over-Shoot	Waktu Turun	Kesalahan Keadaan
Kp	Menurun	Meninngkat	Perubahan kecil	Menurun
Ki	Menurun	Meningkat	Meningkat	Hilang
Kd	Perubahan kecil	Menurun	Meningkat	Perubahan kecil

2.9. Akurasi dan Presisi

2.9.1 Presisi

Untuk mencari nilai presisi pengukuran yaitu dengan cara mencari deviasi dari setiap pengukuran. Untuk mencari deviasi yaitu pengukuran ke-n dikurangi rata-rata pengukuran.

$$\bar{X} = \frac{x1+x2+x3+xn}{n} \quad (2.\ 6)$$

$$dn = xn - \bar{X} \quad (2.\ 7)$$

$$D = \frac{|d1|+|d2|+|d3|+|d4|+|d5|+|dn|}{n} \quad (2.\ 8)$$

Dengan :

$\bar{X}$ = Rata-rata pengukuran

D = Deviasi

dn = Deviasi ke-n

xn = Pengukuran ke-n

n = Jumlah pengukuran

2.9.2 Akurasi

Untuk mencari nilai akurasi pertama harus mencari nilai selisih absolut antara nilai referensi dan nilai pengukuran terlebih dahulu. Setelah mengetahui nilai selisih absolut kemudian dibagi dengan nilai referensi.

$$Persentase\ \ error = \frac{|\ Selisih\ Nilai\ Pengukuran|}{Nilai\ Acuan}\ x\ 100\% \quad (2.\ 9)$$

$$rata - rata\ Persentase\ error = \frac{\Sigma\ Persentase\ error}{n} \quad (2.\ 10)$$

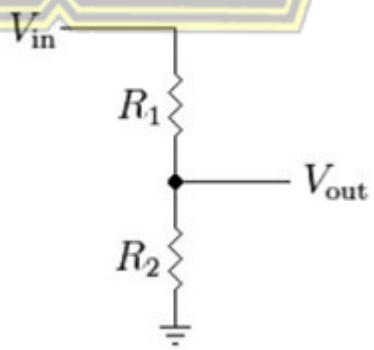
Dengan n adalah banyaknya percobaan.

Maka :

$$Akurasi = 100\% - Persentase\ error \quad (2.\ 11)$$

2.9.3 Pembagi Tegangan

Pembagi Tegangan adalah suatu rangkaian elektronika yang bisa merubah tegangan besar menjadi tegangan yang lebih kecil. Rangkaian pembagi tegangan sederhanya hanya membutuhkan 2 buah resistor yang dirangkai secara seri seperti pada gambar 2.



Gambar 2. 18 Rangkaian Pembagi Tegangan

(Sumber : rangkaianelektronika.info)

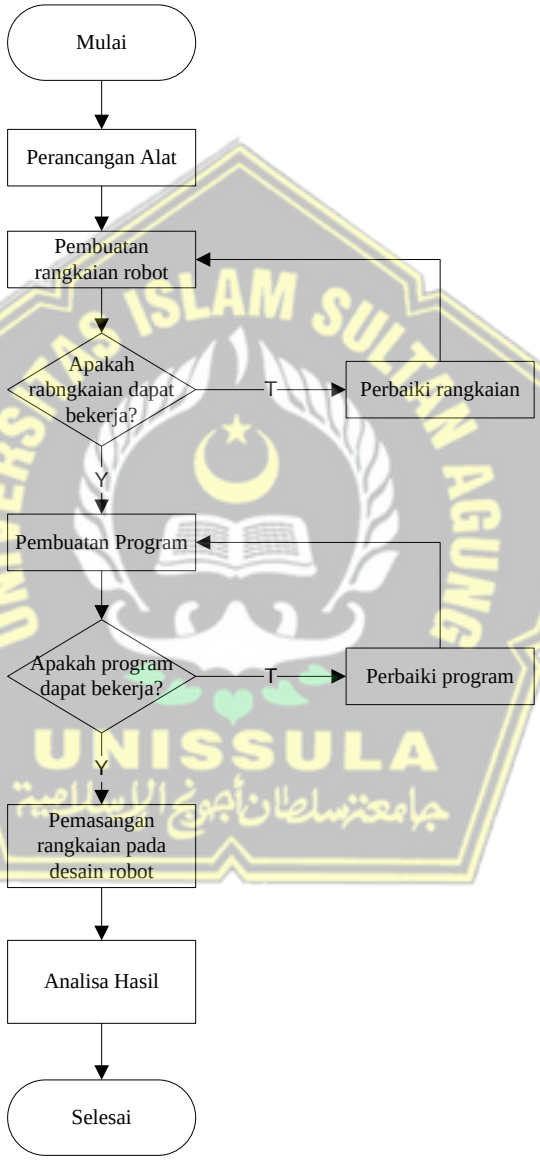
Rumus pembagi tegangan dapat dilihat pada persamaan 2.7

$$Vout = \frac{R2}{R1+R2}\ x\ Vin \qquad (2.\ 12)$$

BAB III

METODE PERANCANGAN

3.1. Flowchart Perancangan



Gambar 3. 1 Flowchart Perancangan

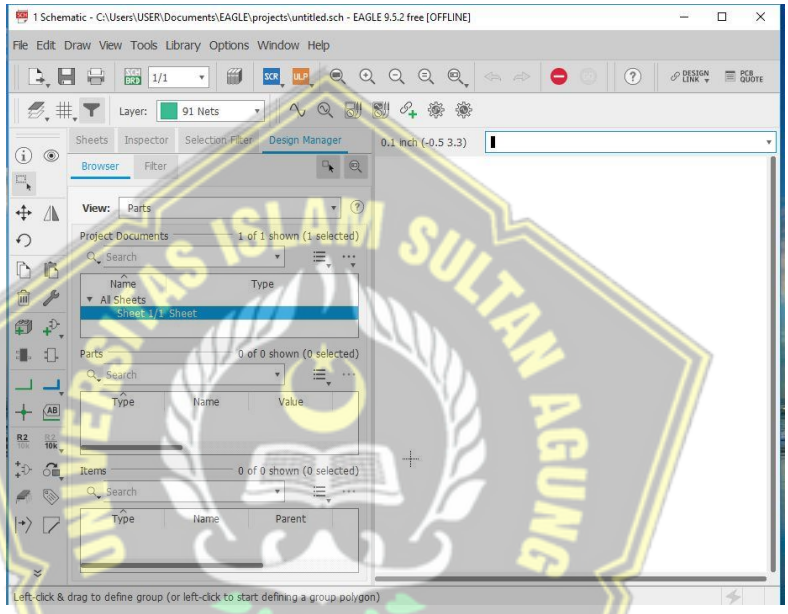
3.1.1 Perancangan Alat

Dalam perancangan alat ada beberapa tahapan, yaitu pembuatan rangkaian, pengujian rangkaian, pembuatan program, pengujian program, memasang

rangkaian yang sudah dibuat, kemudian menganalisa hasil dari alat yang sudah dibuat apakah bisa bekerja dengan baik atau tidak.

3.1.2 Merancang Rangkaian Rangkaian

Langkah selanjutnya yaitu merancang skema rangkaian. Pada tahap ini supaya tidak terjadi kesalahan dalam merangkai semua komponen yang digunakan pada pembuatan robot terlebih dahulu dibuat skema rangkaiannya. Pembuatan skema rangkaian dilakukan dengan menggunakan bantuan software eagle 9.5.2.



Gambar 3. 2 *interface software eagle 2.9.2*

3.1.3 Membuat Program

Setelah merancang skema rangkaian, langkah selanjutnya yaitu membuat program untuk sistem kontrol robot. Pembuatan program dilakukan pada *software Arduino IDE*. Pada pembuatan program diperlukan *library* yang mendukung komponen komponen agar bisa terkoneksi dengan baik. *Library* juga mempermudah dalam pembuatan program, karena dengan menggunakan *library* hanya perlu menggunakan fungsi-fungsi yang sudah tersedia.

3.1.4 Pemasangan Rangkaian Pada Desain Robot

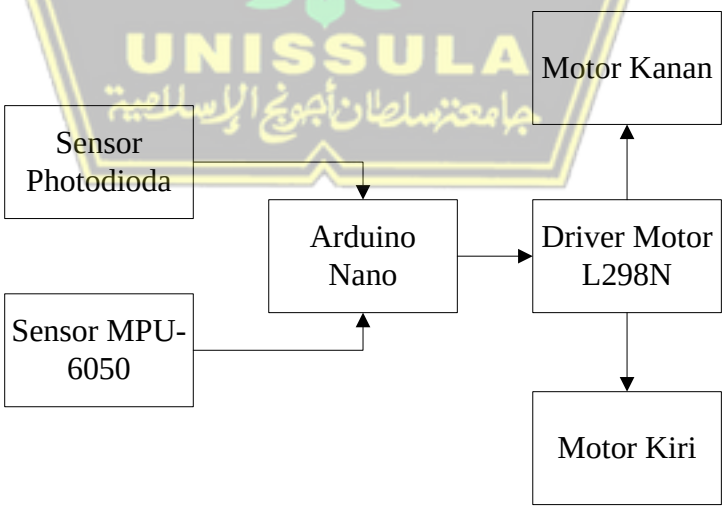
Setelah rangkaian dan program robot sudah selesai dibuat, selanjutnya rangkaian dipasang pada desain robot untuk selanjutnya dilakukan pengujian dan analisa apakah robot sudah berjalan dengan baik atau tidak.

3.1.5 Analisa Hasil

Setelah semua langkah dilakukan, selanjutnya yaitu analisa hasil pengujian. Analisa hasil pengujian alat bertujuan untuk mengetahui apakah alat yang dibuat bekerja dengan semestinya atau tidak. Jika alat tidak bekerja dengan semestinya maka perlu diperbaiki lagi dari langkah-langkah sebelumnya. Dan jika alat sudah bekerja dengan semestinya maka alat sudah berhasil dibuat.

3.2. Deskispsi Umum

Self Balancing Robot Line Follower adalah sebuah robot yang dapat menyeimbangkan diri dan dapat berjalan mengikuti garis hitam pada *background* berwarna putih. Perancangan dan pembuatan robot balancing yang dilakukan pada Tugas Akhir ini terdapat beberapa tahapan penelitian. Pertama, perancangan perangkat keras (hardware) yaitu merangkai semua komponen yang digunakan dalam pembuatan alat meliputi Arduino nano, MPU-6050, sensor *photodioda*, motor DC, *driver* motor DC. Kedua, perancangan perangkat lunak (*software*) yaitu pembuatan program pada arduino IDE dengan menggunakan metode PID. Diagram blok *self balancing robot line follower* dapat dilihat pada Gambar 3.2.

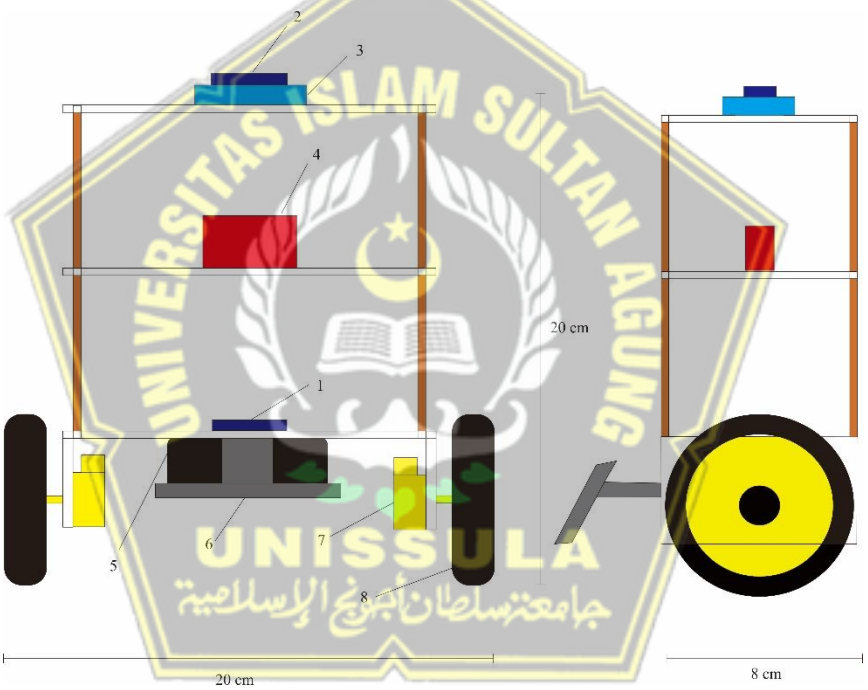


Gambar 3. 3 Diagram Blok Sistem *Self Balancing Robot Line Follower*

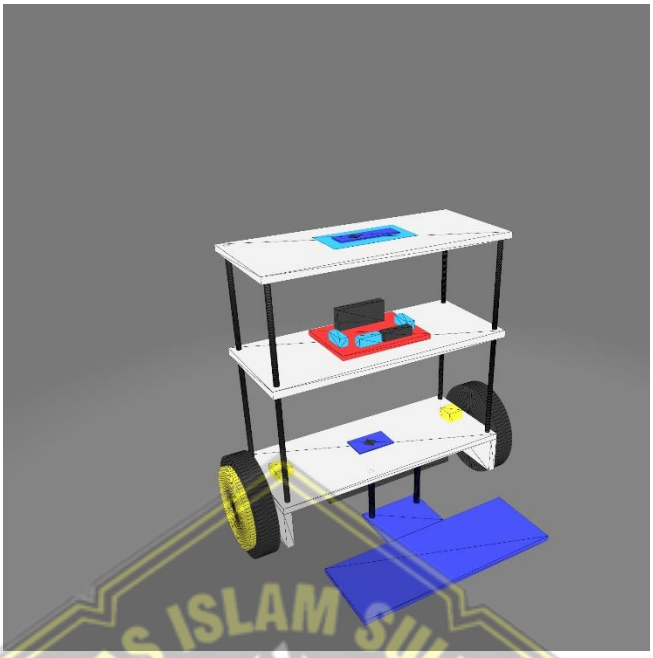
3.3. Perangkat Keras (*Hardware*)

3.2.1 Perancangan Mekanik

Perancangan perangkat keras yaitu membuat desain robot, merangkai semua komponen yang digunakan dalam pembuatan alat, komponen yang digunakan yaitu Arduino Nano, MPU-6050, sensor *photodioda*, motor DC, *driver motor* DC. Penempatan posisi komponen sangatlah penting dalam pembuatan robot ini, karena akan sangat mempengaruhi pergerakan dari robot, maka komponen harus ditempatkan dengan teliti agar robot dapat berdiri seimbang. Gambar 3.3 menunjukkan desain dan dimensi robot.



Gambar 3. 4 Desain dan Dimensi Robot



Gambar 3. 5 Desain 3D robot

Keterangan gambar :

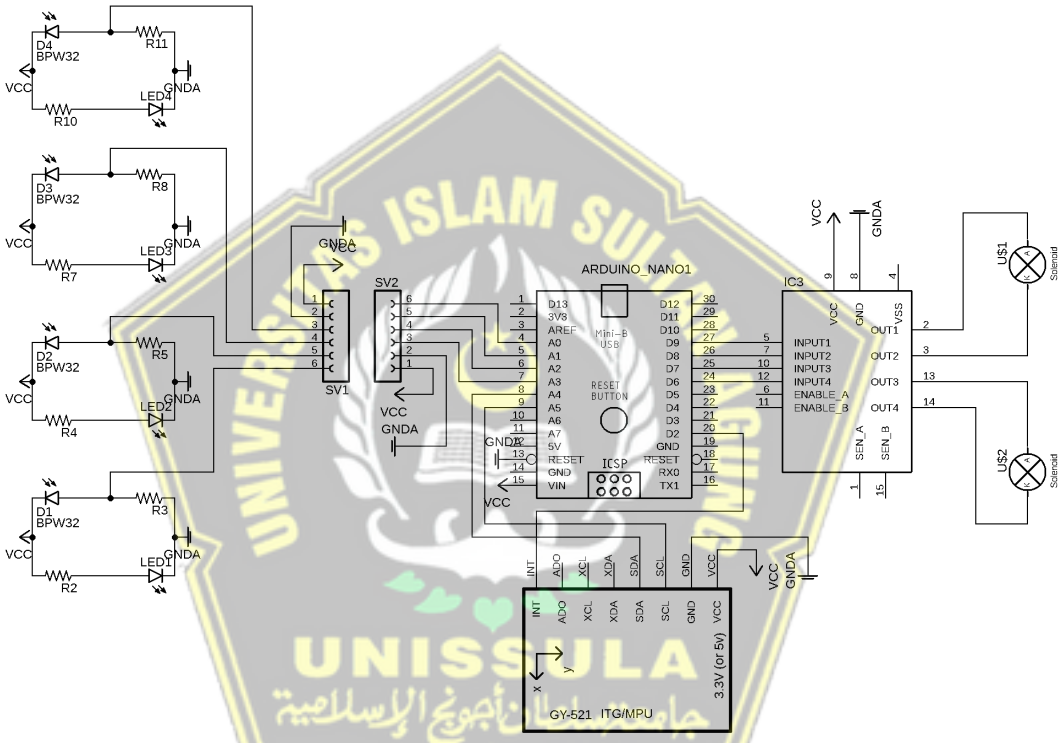
1. Sensor MPU 6050
2. Arduino Nano
3. Project Board
4. Driver motor DC L293N
5. Tempat Battere
6. Sensor *Photodioda*
7. Motor DC
8. Roda

Desain dan dimensi robot juga harus presisi karena mempengaruhi keseimbangan robot pada saat berdiri. Pada Tugas Akhir ini robot menggunakan dimensi dengan panjang 20 cm, tinggi 20 cm, lebar 8 cm. Desain robot menggunakan bahan akrilik dengan ketebalan 3 mm. Penyangga robot menggunakan speser dengan diameter 3 mm dan panjang 3 cm.

3.2.2 Perancangan Elektronik

Self balancing robot line follower agar dapat berdiri seimbang dibutuhkan sebuah sensor *gyro* dan sensor *photodioda*. Sensor *gyro* untuk mendeteksi kemiringannya

sedangkan sensor *phothodioda* untuk mendeteksi garis. Penempatan sensor *gyro* sangat berpengaruh untuk menentukan pergerakan robot. Untuk itu pada penelitian ini sensor diletakkan pada bagian atas robot karena sensor akan lebih mudah mendeteksi kemiringan jika ada pada posisi yang tinggi. Sedangkan sensor *photodioda* akan ditempatkan pada posisi bawah untuk mendeteksi garis hitam pada *background* putih. Gambar 3.5 menunjukkan skema rangkaian *self balancing robot line follower*.



Gambar 3. 6 Skema Rangkaian *self balancing robot line follower*

Robot memiliki perangkat yang terdiri dari perangkat *input* dan perangkat *output*. Perangkat *input* maupun perangkat *output* mempunyai peranan penting dalam mengerjakan fungsinya masing-masing.

3.2.3 Perangkat *Input*

Perangkat *input* yang terdapat pada *Self Balancing Robot line Follower yaitu* :

1. Sensor *Photodioda*

Sensor *photodioda* berfungsi sebagai pendeteksi garis hitam pada *background* putih. Sensor photodioda akan mengalami perubahan resistansi saat menerima

intensitas cahaya dan akan mengalirkan arus listrik secara *forward* seperti prinsip dioda pada umumnya. Pada robot ini *photodioda* dirangkai secara *Pull-Down*, yaitu *output* yang dikeluarkan akan berlogika 1 saat tidak mendapatkan cahaya yang cukup.

2. Sensor MPU6050

Gyroscop dan *accelerometer* pada satu chip yang kecil.*Gyroscop* berfungsi untuk menentukan orientasi suatu benda berdasarkan sudut, sedangkan *accelerometer* berfungsi untuk mengukur percepatan akibat gravitasi. Modul ini mampu menangkap data x, y, z pada waktu yang bersamaan. Karena memiliki hardware khusus untuk konversi analog ke digital selebar 16 bit. Sehingga modul ini dapat terbilang akurat.

3.2.4 Perangkat *Output*

Perangkat *output* yang terdapat pada *Self Balancing Robot Line Follower* yaitu motor DC. Motor DC adalah sebuah komponen yang dapat merubah arus listrik menjadi mekanik atau gerak. Motor DC berfungsi sebagai aktuator pada robot ini. Motor DC akan berputar maju dan mundur untuk menyeimbangkan robot agar tetap berdiri dengan seimbang.

3.2.5 Proses Kontrol

Proses kontrol dari *Self Balancing Robot Line Follower* dengan menggunakan metode PID. Agar robot dapat berdiri dengan seimbang maka nilai PID harus tepat. Jika nilai PID kurang tepat efeknya adalah robot akan mudah jatuh.

Pada *Self Balancing Robot Line Follower* ini menggunakan 2 sensor yaitu sensor *Photodioda* dan Sensor *MPU-6050*. Sensor *Photodioda* digunakan untuk mendeteksi garis dan sensor *MPU-6050* untuk mendeteksi kemiringan robot.

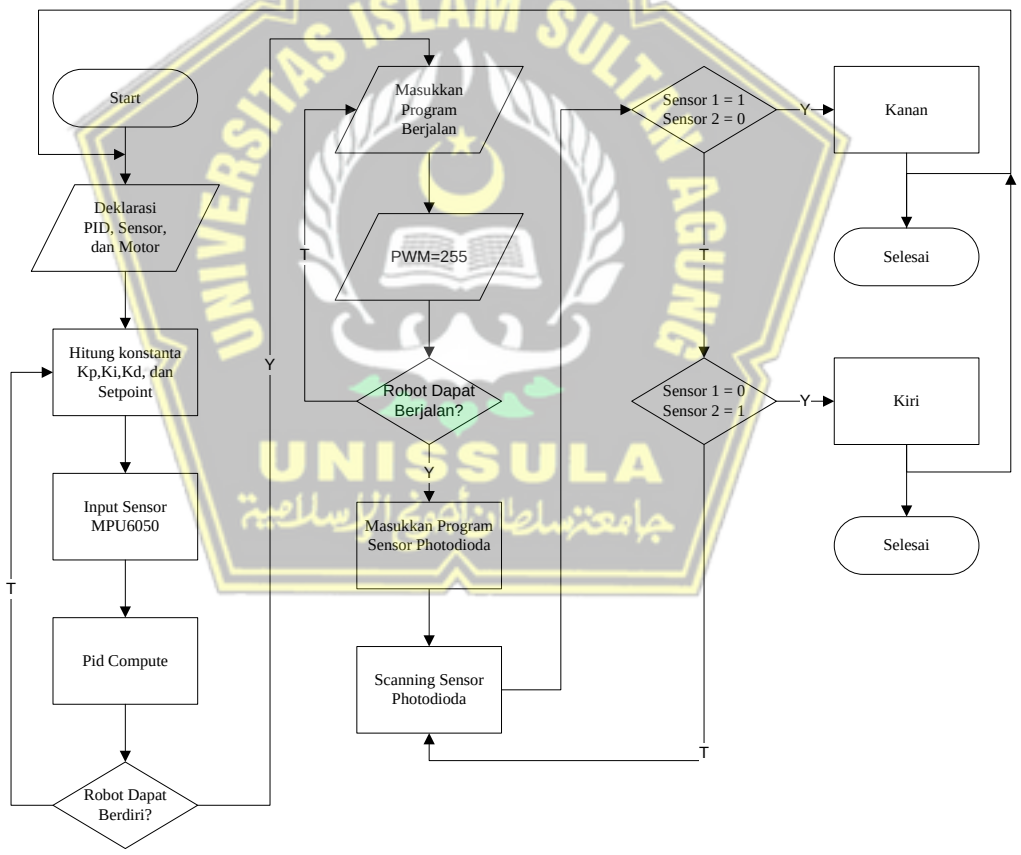
Robot Menggunakan 4 buah sensor *Photodioda* yang dihubungkan ke pin analog A0, A1, A2, A3 untuk mendeksi garis hitam supaya lebih akurat. Sensor dirangkai secara *Pull-Up* maka *output* yang dihasilkan akan berlogika 1 saat mendapatkan

ahaya yang cukup. Sinyal dari sensor *Photodioda* akan dimasukkan ke arduino nano.

Sensor *MPU-6050* digunakan sebagai pendeteksi kemiringan dari robot. Dari nilai sudut kemiringan yang dihasilkan *MPU-6050* dimasukkan ke Arduino Nano untuk diproses dengan menggunakan metode PID

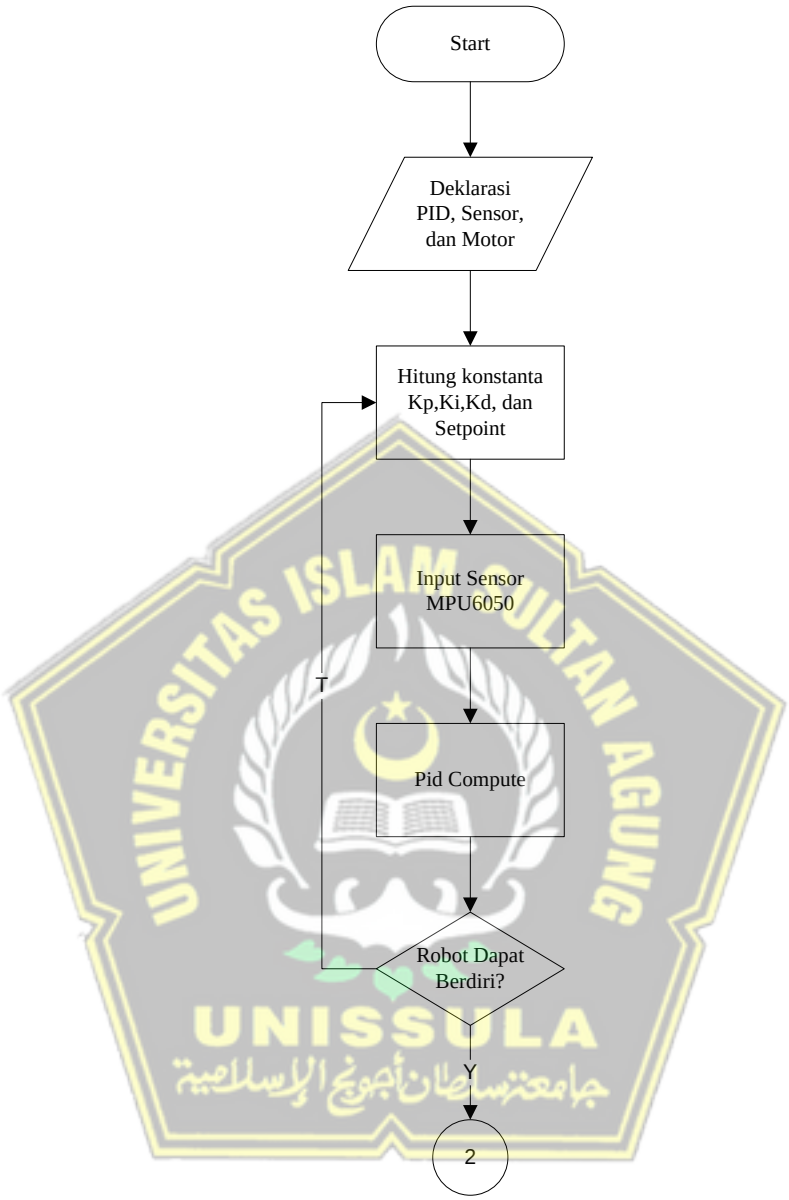
3.4. Flowchart Software Sistem

Pada suatu algoritma pemrograman dibutuhkan susunan atau serangkaian langkah dan prosedur untuk mencapai tujuan sistem yang dibuat, yaitu dengan membuat diagram alir (*flowchart*). Gambar 3.8 menunjukkan diagram alir sistem *self balancing robot line follower* yang akan dibuat pada Tugas Akhir ini.



Gambar 3. 7 Diagram Alir Sistem

Tahapan Pembuatan Program



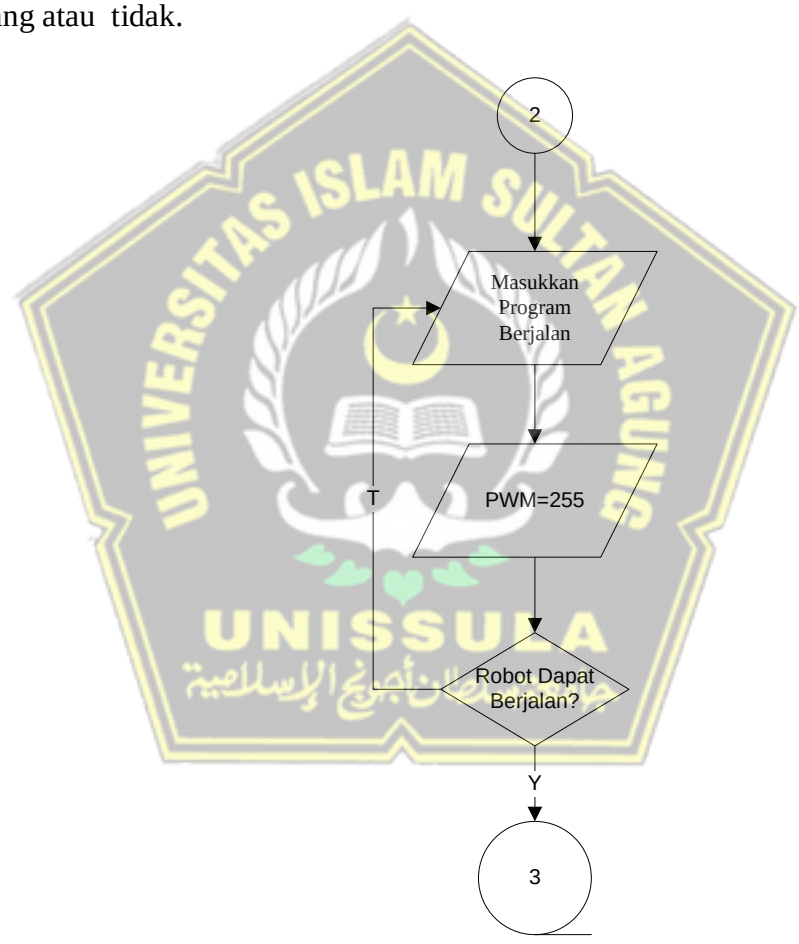
Gambar 3. 8 *Flowchart* algoritma deklarasi variabel

```
Deklarasi variabel

//PID

#if MANUAL_TUNING

  double kp , ki, kd;

  double prevKp, prevKi, prevKd;
```

```
#endif
double originalSetpoint = 176.29;//-1.10; //176.29;
double setpoint = originalSetpoint;
double movingAngleOffset = 0.3;
double input, output;
int moveState=0; //0 = balance; 1 = back; 2 = forth
#if MANUAL_TUNING
  PID pid(&input, &output, &setpoint, 0, 0, 0, DIRECT);
#else
  PID pid(&input, &output, &setpoint, 40, 270, 1.9,
DIRECT);
#endif
//photodioda
int IR1=10;       //Right sensor
int IR2=11;     //left Sensor
int a;
int b;
//MOTOR CONTROLLER
int ENA = 3;
int IN1 = 8;
int IN2 = 4;
int IN3 = 7;
int IN4 = 5;
```

```
int ENB = 6;

LMotorController motorController(ENA, IN1, IN2, ENB,
IN3, IN4, 0.6, 1);
```

Pada tahap ini mendeklarasikan variabel-variabel yang akan digunakan seperti PID, sensor, motor, Kp, Ki, Kd, dan Setpoint. Input pada tahap ini adalah sensor MPU6050 yang digunakan untuk mendeteksi kemiringan robot. Data dari sensor MPU6050 akan dihitung dengan perintah *PID Compute* agar robot mampu berdiri seimbang atau tidak.

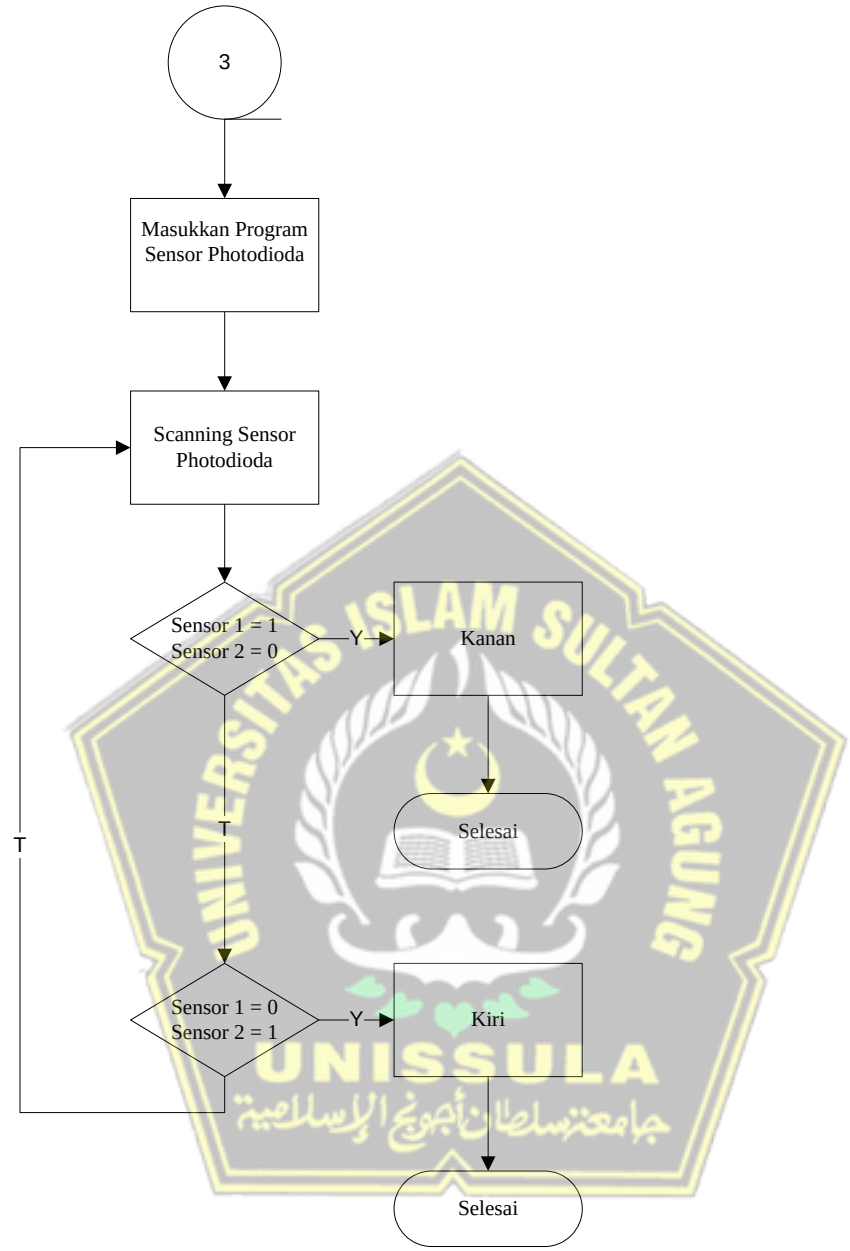


Gambar 3. 9 *Flowchart* algoritma robot berdiri dan berjalan maju

```
  if (!dmpReady) return;

    // wait for MPU interrupt or extra packet(s)
available

    while (!mpuInterrupt && fifoCount < packetSize)
```

```
    {
        //no mpu data - performing PID calculations and
output to motors
         Serial.println(setpoint);
         pid.Compute();
         Serial.print (" y : "); Serial.println(input);
//Serial.print(" =>"); Serial.println(output);
         motorController.move(output, MIN_ABS_SPEED);
```

Pada tahap ini yaitu pembuatan program keseimbangan robot dan robot berjalan ke depan. Jika perhitungan PID kurang tepat kemudian robot tidak dapat berdiri seimbang maka menghitung kembali nilai Kp, Ki, Kd. Dan jika perhitungan PID sudah tepat dan robot dapat berdiri seimbang, maka memasukkan program robot berjalan maju dengan PWM 255.



Gambar 3. 10 *Flowchart* algoritma *scanning* sensor *photodioda*

```
//memasukkan program sensor photodioda

//sensor melakukan scanning

if(input>150 && input<200)

        {
```

```
      a = digitalRead(IR1);
      b = digitalRead(IR2);
      if(output>0)
      {
        if(a == HIGH && b == HIGH)
        {
             pid.Compute();
motorController.move(output, MIN_ABS_SPEED);
        }
        else if(a == LOW && b == LOW)
        {
       pid.Compute();
      motorController.move(output, MIN_ABS_SPEED);
        }
        else if(a == LOW && b == HIGH)
        {
      digitalWrite(IN1,HIGH);
      digitalWrite(IN2,LOW);
      digitalWrite(IN3,LOW);
      digitalWrite(IN4,HIGH);
      analogWrite (ENA, 200);
      analogWrite (ENB, 100);
        }
```

```
            else if(a == HIGH && b == LOW)
            {
          digitalWrite(IN1,LOW);
          digitalWrite(IN2,HIGH);
          digitalWrite(IN3,HIGH);
          digitalWrite(IN4,LOW);
          analogWrite (ENA, 100);
          analogWrite (ENB, 200);
            }
          }
          else if(output == -255)
          {
             pid.Compute();
motorController.move(output, MIN_ABS_SPEED);
          }
        }
        else
        {
           pid.Compute();
motorController.move(output, MIN_ABS_SPEED);
        }
```

Pada tahap ini *scanning* sensor *photodioda* untuk robot dapat mengikuti garis hitam pada *background* putih. Jika sensor 1 dan 3 bernilai 1 atau *HIGH*, maka robot

berjalan maju. Jika sensor 1 dan 2 bernilai 1 atau *HIGH*, maka robot belok kiri. Jika sensor 3 dan 4 bernilai 1 atau *HIGH* maka robot belok kanan. Selain itu robot berhenti.

3.5. Perangkat Lunak (*software*)

Sedangkan perancangan *software* yaitu membuat program untuk *Self Balancing Robot Line Follower* dengan menggunakan Arduino IDE. Yang perlu diprogram meliputi pembacaan sudut kemiringan dari sensor MPU6050, pembacaan nilai sensor phothodioda, dan pergerakan motor. Yang akan dibuat pertama adalah program unuk sensor *mpu-6050* untuk mengetahui nilai sudut dari sensor. Selanjutnya memprogram sensor *Photodioda* agar robot dapat berjalan mendekti garis hitam. Setelah itu pergerakan motor harus selaras dengan pembacaan sensor *Photodioda* dan *MPU-6050* agar menyeimbangkan robot dan bergerak maju mengikuti garis hitam pada *background* putih. Untuk menentukan keseimbangan robot dengan menggunakan metode PID pada program arduino.

3.5.1 Deklarasi

Supaya komponen-komponen dan program dapat berjalan dengan baik maka harus dideklarasikan terlebih dahulu. Contoh pendeklarasian yaitu int, float, double, char, booleab, dan lain-lain. Contoh pendeklarasian pada program dapat dilihat di bawah ini.

```
double originalSetpoint = 174.3; //176.5
double setpoint = originalSetpoint;
double movingAngleOffset = 0.1;
double input, output;
int ENA = 5;
int IN1 = 6;
int IN2 = 7;
int IN3 = 8;
int IN4 = 9;
int ENB = 10;
```

3.5.2 *Library*

Librari program adalah kumpulan kode program yang dibuat dan memiliki fungsi-fungsi tertentu dan dapat di panggil ke dalam program lain. *Library* dibuat untuk memudahkan *programer* dalam membuat suatu program. Dengan menggunakan *library programer* tidak harus membuat program dari awal untuk suatu fungsi tertentu. Di bawah ini merupakan penggunaan *library* pada program.

```
//LIB PID
#include <PID_v1.h>
//LIB MOTOR
#include <LMotorController.h>
//MPU
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include "Wire.h"
```

3.5.3 Program Pembacaan Sensor *MPU-6050*

Supaya robot dapat berdiri seimbang, maka dibutuhkan nilai dari sensor *MPU-6050* sebagai data kemiringan dari robot. Pada penelitian ini menggunakan data dari *axis* x sebagai acuan sudut robot. Di bawah ini merupakan program pembacaan sensor *MPU-6050*.

```
void setup() {
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
        TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
    #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif
    Serial.begin(115200);
    while (!Serial);
    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();
```

```
        Serial.println(F("Testing device connections..."));
        Serial.println(mpu.testConnection() ? F("MPU6050
connection successful") : F("MPU6050 connection failed"));
        Serial.println(F("Initializing DMP..."));
        devStatus = mpu.dmpInitialize();
        mpu.setXGyroOffset(220);
        mpu.setYGyroOffset(76);
        mpu.setZGyroOffset(-85);
        mpu.setZAccelOffset(1788);
        if (devStatus == 0) {
            Serial.println(F("Enabling DMP..."));
            mpu.setDMPEnabled(true);
            Serial.println(F("Enabling interrupt detection
(Arduino external interrupt 0)..."));
            attachInterrupt(0, dmpDataReady, RISING);
            mpuIntStatus = mpu.getIntStatus();
            Serial.println(F("DMP ready! Waiting for first
interrupt..."));
            dmpReady = true;
    pid.SetMode(AUTOMATIC);
    pid.SetSampleTime(10);
    pid.SetOutputLimits(-255, 255);
    }
    else {
            Serial.print(F("DMP Initialization failed (code "));
            Serial.print(devStatus);
            Serial.println(F(")"));
        }
    }
    void loop() {
        if (!dmpReady) return;
        while (!mpuInterrupt && fifoCount < packetSize)
        {
            pid.Compute();
```

```
            Serial.print(input);        Serial.print("        =>");
Serial.println(output);
         }
```

3.5.4 Program Pergerakan Motor

Motor berperan penting pada robot ini, tanpa motor maka robot ini tidak dapat bekerja. Supaya motor dapat bergerak menyesuaikan dengan sudut kemiringan dari robot, maka diperlukan program untuk menggerakkan motor. Di bawah ini merupakan program pergerakan motot.

```
void Forward()
{
    analogWrite(5,output);
    analogWrite(10,output);
    analogWrite(6,output);
    analogWrite(7,0);
    analogWrite(8,output);
    analogWrite(9,0);
    Serial.print("F");
}

void Reverse()
{
 analogWrite(5,output*-1);
    analogWrite(10,output*-1);
    analogWrite(6,0);
    analogWrite(7,output*-1);
    analogWrite(8,0);
    analogWrite(9,output*-1);
    Serial.print("R");
}

void Stop()
{
```

```
  analogWrite(5,output);
    analogWrite(10,output);
    analogWrite(6,0);
    analogWrite(7,0);
    analogWrite(8,0);
    analogWrite(9,0);
    Serial.print("S");
}
```

3.5.5 Program Pembacaan Sensor *Photodioda*

Pada penelitian ini menggunakann sensor *photodioda* sebagai sensor garis. Supaya robot dapat mengikuti garis, maka diperlukan data dari sensor *photodioda*. Di bawah ini meerupakan program sensor *photodioda.*

```
const int pd1 = A0;
const int pd2 = A1;
const int pd3 = A2;
const int pd4 = A3;
int baca1;
int baca2;
int baca3;
int baca4;
void setup()
{
  Serial.begin(115200)
  pinMode(pd1, INPUT);
  pinMode(pd2, INPUT);
  pinMode(pd3, INPUT);
  pinMode(pd4, INPUT);
}
void loop()
{
  baca1 = analogRead(pd1);
  Serial.println(baca1);
```

```
  baca2 = analogRead(pd2);
  Serial.println(baca2);
  baca3 = analogRead(pd3);
  Serial.println(baca3);
  baca4 = analogRead(pd4);
  Serial.println(baca4);
  if (pd2 > 50 && pd3 > 50)
  {
    forward();
  }
  else if (pd1 > 50 && pd2 > 50)
  {
    kiri();
  }
  else if (pd3 > 50 && pd4 > 50)
  {
    kanan();
  }
}
```

Listing Program

```
//LIB PID
#include <PID_v1.h>
//LIB MOTOR
#include <LMotorController.h>
//MPU
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif
#define MIN_ABS_SPEED 20
```

```
MPU6050 mpu;
#define OUTPUT_READABLE_YAWPITCHROLL
//PID
double originalSetpoint = 174.3; //176.5
double setpoint = originalSetpoint;
double movingAngleOffset = 0.1;
double input, output;
//adjust these values to fit your own design
double Kp = 43;//32;//32;//29//27
double Kd = 1.9;//1.7;//1.3//0.8
double Ki = 250;//250;//230;//220//178
PID pid(&input, &output, &setpoint, Kp, Ki, Kd, DIRECT);
double motorSpeedFactorLeft = 0.6;
double motorSpeedFactorRight = 0.6;
//MOTOR CONTROLLER
int ENA = 5;
int IN1 = 6;
int IN2 = 7;
int IN3 = 8;
int IN4 = 9;
int ENB = 10;
LMotorController motorController(ENA, IN1, IN2, ENB, IN3, IN4,
motorSpeedFactorLeft, motorSpeedFactorRight);
void loop(){
while(!mpuInterrup && fifoCCount<packetSize)
{
pid.Compute();
Serial.print(input); Serial.print(" =>"); Serial.println(output);
if (input>150 && input<200){//If the Bot is falling
        if (output>0) //Falling towards front
        Forward(); //Rotate the wheels forward
        else if (output<0) //Falling towards back
        Reverse(); //Rotate the wheels backward
        }
```

```
        else //If Bot not falling
        Stop(); //Hold the wheels still
    }
}
```

3.6. Flowchart Pengujian

Gambar 3. 11 Flowchart Pengujian

Flowchart pengujian yaitu langkah-langkah dalam melakukan pengujian pada suatu alat yang sudah dibuat. Pertama, menyalakan robot, yaitu memberikan tegangan pada robot. Setelah memberikan tegangan pada robot, langkah kedua yaitu melakukan pengecekkan komponen, apakah komponen bekerja dengan semestinya atau tidak. Jika komponen sudah bekerja dengan baik, ketiga yaitu apakah robot dapat berdiri atau bekerja dengan semestinya, jika tidak maka melakukan pengecekan ulang pada semua komponen, jika robot dapan berdiri atau bekerja dengan semestinya, maka sudah berhasil.

BAB IV

HASIL DAN ANALISA

4.1 Pengujian Sensor MPU-6050

Pengujian sensor MPU-6050 dilakukan dengan dengan cara membandingkan hasil pengukuran sudut kemiringan manual dengan menggunakan busur sebagai alat ukur dan hasil pengukuran dari sensor MPU-6050. Pada sensor MPU-6050 terdapat 3 sumbu axis, yaitu X, Y, dan Z. Sumbu yang digunakan dalam penelitian menggunakan sumbu Y dengan satuan derajat.

Cara mengukur sudut kemiringan robot dengan mennggunakan busur dengan meletakkan busur dan robot pada bidang datar setelah itu mengukur sesuai dengan sudut yang diinginkan. Pada penelitian ini menggunakan sudut -90°, -60º, -45°,-30º, 20º, 30º, 45°, 60º, 90° sebagai sudut perbandingannya.

Gambar 4. 1 Pengujian Sudut Robot

Dari pengujian sudut sensor MPU-6050 didapatkan hasil pembacaan sudut yang digunakan dalam penelitian sudah mendekati sudut aslinya. Sudut

yang diukur pada rentang -90° sampai 90°. Pembacaan sudut MPU-6050 diambil dari serial monitor arduino. berikut ini merupakan fungsi untuk melihat data sudut sensor MPU-6050.

```
mpu.dmpGetQuaternion(&q, fifoBuffer);
     mpu.dmpGetGravity(&gravity, &q);
     mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
     #if LOG_INPUT
          Serial.print("ypr\t");
          Serial.print(ypr[0] * 180/M_PI);
          Serial.print("\t");
          Serial.print(ypr[1] * 180/M_PI);
          Serial.print("\t");
          Serial.println(ypr[2] * 180/M_PI);
     #endif
     input = ypr[1] * 180/M_PI ;
     Serial.print(input);
```

Gambar 4. 2 Pembacaan Sudut Pada Sumbu Y

Tabel 4. 1 Data Sudut Pembacaan Sensor.

Sudut Terbaca	Sudut Terukut										
	-90	-60	-45	-30	-20	0,5	20	30	45	60	90
1	-65,86	-46,56	-29,69	-23,67	-13,7	0,55	13,67	23,35	28,6	47,7	70,56
2	-65,76	-46,58	-29,75	-23,6	-13,73	0,6	13,63	23,36	28,58	47,68	70,56
3	-65,73	-45,54	-29,67	-23,53	-13,67	0,57	13,65	23,38	28,66	47,69	70,56
4	-65,66	-45,2	-29,67	-23,46	-13,67	0,6	13,63	23,39	28,67	47,68	70,56
5	-65,66	-44,69	-29,65	-23,39	-13,65	0,6	13,62	23,4	28,71	47,675	70,56
6	-65,71	-44,18	-29,63	-23,32	-13,64	0,52	13,61	23,42	28,62	47,67	70,57
7	-65,65	-43,67	-29,69	-23,25	-13,65	0,51	13,6	23,43	28,58	47,665	70,57
8	-65,69	-43,16	-29,7	-23,18	-13,61	0,52	13,59	23,45	28,65	47,66	70,57
9	-65,78	-42,65	-29,68	-23,11	-13,59	0,52	13,58	23,46	28,63	47,655	70,57
10	-65,67	-42,14	-29,68	-23,04	-13,58	0,5	13,57	23,48	28,63	47,65	70,57

Data yang didapat dari Tabel 4.1 digunakan untuk menentukan *setpoint* robot. Pada tabel 4.1 pembacaan nilai sensor dilakukan sebanyak sepuluh kali percobaan dan terdapat perubahan nilai sensor yang tidak terlalu signifikan.

4.2 Menentukan Titik *Set Point*

Berdasarkan dari data pada Tabel 4.1 untuk menentukan *setpoint* yang harus dicari nilai presisi dan akurasi pengukuran terlebih dahulu, supaya kinerja robot lebih maksimal.

Tabel 4. 2 Nilai Presisi Sensor MPU-6050

no	DEVISASI |d|										
	-90	-60	-45	-30	-20	0,5	20	30	45	60	90
1	0,143	2,123	0,009	0,315	0,051	0,001	0,055	0,055	0,033	0,0275	0,005
2	0,043	2,123	0,069	0,245	0,081	0,051	0,015	0,015	0,053	0,0075	0,005
3	0,013	2,143	0,011	0,175	0,021	0,021	0,035	0,035	0,027	0,0175	0,005
4	0,057	1,103	0,011	0,105	0,021	0,051	0,015	0,015	0,037	0,0075	0,005
5	0,057	0,763	0,031	0,035	0,001	0,051	0,005	0,005	0,077	0,0025	0,005
6	0,007	0,253	0,051	0,035	0,009	0,029	0,005	0,005	0,013	0,0025	0,005
7	0,067	0,257	0,009	0,105	0,001	0,039	0,015	0,015	0,053	0,0075	0,005
8	0,027	0,767	0,019	0,175	0,039	0,029	0,025	0,025	0,017	0,0125	0,005
9	0,063	1,277	0,001	0,245	0,059	0,029	0,035	0,035	0,003	0,0175	0,005
10	0,047	1,787	0,001	0,315	0,069	0,049	0,045	0,045	0,003	0,0225	0,005
rata-rata	0,0524	1,2596	0,0212	0,175	0,0352	0,035	0,025	0,025	0,0316	0,0125	0,005
presisi	99,9476	98,7404	99,9788	99,825	99,9648	99,965	99,975	99,975	99,9684	99,9875	99,995

Contoh perhitungan pada sudut -90°

Mencari rata-rata pengukuran sudut

Berdasarkan Persamaan 2.1 didapatkan nilai rata-rata sebagai berikut

$$\bar{X} = \frac{x1 + x2 + x3 + xn}{n}$$

$$\bar{X} = -65{,}717$$

Menghitung deviasi

Berdasarkan persamaan 2.2 didapatkan nilai deviasi sebagai berikut :

$$dn = xn - \bar{X}$$

$$d1 = x1 - \bar{X}$$

$$d1 = |(-65.86) - (-65{,}717)|$$

$$d1 = 0{,}143$$

Menghitung rata-rata deviasi

Berdasarkan persamaan 2.3 didapatkan rata-rata deviasi sebagai berikut :

$$D = \frac{|d1| + |d2| + |d3| + |d4| + |d5| + |dn|}{n}$$

$$D = 0{,}0525$$

Menghitung presisi

$$presisi = 100 - D$$

$$presisi = 100 - 0{,}0524$$

$$presisi = 99{,}9476$$

Tabel 4. 3 Nilai Akurasi Sensor MPU-6050

Sudut Terbaca	Sudut Terukut										
	-90	-60	-45	-30	-20	0,5	20	30	45	60	90
1	-65,86	-46,56	-29,69	-23,67	-13,7	0,55	13,67	23,35	28,6	47,7	70,56
2	-65,76	-46,58	-29,75	-23,6	-13,73	0,6	13,63	23,36	28,58	47,68	70,56
3	-65,73	-45,54	-29,67	-23,53	-13,67	0,57	13,65	23,38	28,66	47,69	70,56
4	-65,66	-45,2	-29,67	-23,46	-13,67	0,6	13,63	23,39	28,67	47,68	70,56
5	-65,66	-44,69	-29,65	-23,39	-13,65	0,6	13,62	23,4	28,71	47,675	70,56
6	-65,71	-44,18	-29,63	-23,32	-13,64	0,52	13,61	23,42	28,62	47,67	70,57
7	-65,65	-43,67	-29,69	-23,25	-13,65	0,51	13,6	23,43	28,58	47,665	70,57
8	-65,69	-43,16	-29,7	-23,18	-13,61	0,52	13,59	23,45	28,65	47,66	70,57
9	-65,78	-42,65	-29,68	-23,11	-13,59	0,52	13,58	23,46	28,63	47,655	70,57
10	-65,67	-42,14	-29,68	-23,04	-13,58	0,5	13,57	23,48	28,63	47,65	70,57
rata-rata	-65,717	-44,437	-29,681	-23,355	-13,649	0,549	13,615	23,412	28,633	47,6725	70,565
Persentase Error	26,98111	25,93833	34,04222	22,15	31,755	9,8	31,925	21,96	36,37111	20,54583	21,59444
Akurasi	73,01889	74,06167	65,95778	77,85	68,245	90,2	68,075	78,04	63,62889	79,45417	78,40556

Menghitung akurasi

Berdasarkan persamaan 2.4 didapatkan nilai persentase error sebagai berikut :

$$Persentase\ error = \left(\frac{|Selisih\ Nilai\ Pengukuran|}{Nilai\ acuan}\right) x\ 100\%$$

$$Persentase\ error = \left(\frac{24{,}283}{-90}\right) x\ 100\%$$

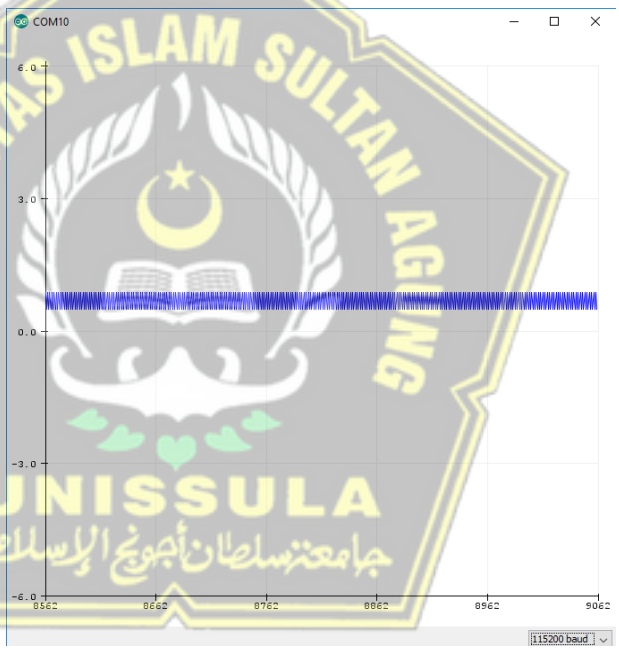
$$Persentase\ error = 26{,}98$$

$$Akurasi\ = 100\% - Persentase\ error$$

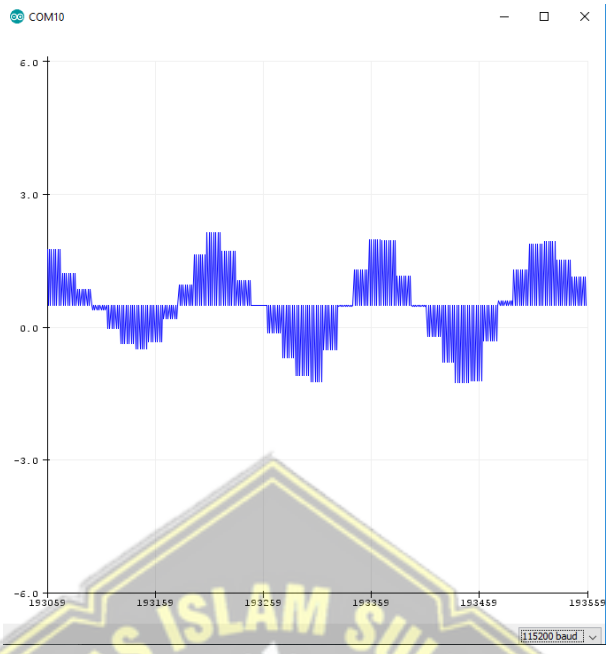
$$Akurasi\ = 73{,}01\%$$

Dari hasil pengukuran tersebut menunjukkan bahwa hasil pengukuran pada sudut -90°, -60º, -45°,-30º, 20º, 30º, 45°, 60º, 90° memiliki tingkat presisi dan akurasi yang kurang baik. Sedangkan pada sudut 0 memiliki akurasi dan presisi yang sangat baik yaitu 99,6%. Sehingga sudut 0° aman digunakan untuk *setpoint* robot.

Setelah mengetahui presisi dan akurasi pengukuran sudut dari sensor MPU-6050 maka dapat ditentukan titik acuan atau *setpoint* yang akan digunakan. *Setpoint* yang akan digunakan yaitu pada sudut 0° tepatnya pada sudut terbaca 0.50. karena pada sudut 0° lebih presisi dan lebih akurat pembacaannya dibandingkan dengan sudut yang lain.



Gambar 4. 3 Respon Sensor Pada Sudut 0° Saat Diam



Gambar 4. 4 Respon Sensor Saat Digerakkan

Dari Gambar 4.3 dan Gambar 4.4 garis ke atas menunjukkan sudut terbaca pada sumbu Y dan garis ke kanan menunjukkan waktu. Dari pengujian tersebut yang menunjukkan respon sensor saat diam dan saat digerakkan maka dapat disimpulkan bahwa sensor dapat bekerja dengan baik.

4.3 Pengujian PWM motor DC

Pulse Width Modulation (PWM) atau modulasi lebar pulsa, adalah teknik pengubahan sinyal digital berupa gelombang kotak (*Square Wave*) dimana *duty cycle* dari gelombang kotak tersebut dapat diatur sesuai dengan kebutuhan sistem.

Pengujian PWM motor DC bertujuan untuk mengetahui arah putar dari motor dc. Hasil pengujian ditunjukkan pada Tabel 4.2.

Tabel 4. 4 Pengujian PWM Dan Arah Putar Motor

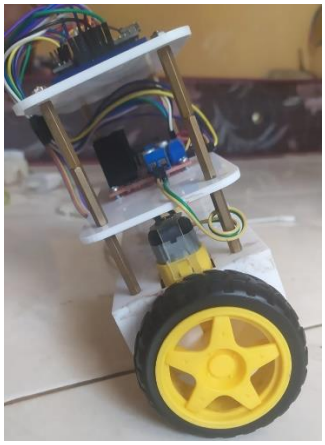
PWM	Kanan	Kiri
0	Berhenti	Berhenti
100	Maju	Maju
255	Maju Cepat	Maju Cepat

Dari hasil pengujian kontrol motor tersebut, didapatkan hasil bahawa kontrol PWM motor sudah sesuai dengan arah putar motor dan PWM semakin besar kecepatan motor juga semakin bertambah.

4.4 Mengkonfigurasikan Nilai Sepoint Terhadap Motor Dc

Setelah mendapatkan nilai *setpoint* dan konfigurasi arah putar motor dc yang sesuai pada pengujian sebelumnya, selanjutnya yaitu mengkonfigurasikan *setpoint* terhadap motor dc. *Setpoint* yang digunakan yaitu 0.50°, maka robot harus mempertahankan pada posisi tersebut. Jika sudut < 0.50° maka motor bergerak maju dengan PWM +255. Jika sudut > 0.50° maka motor bergerak mundur dengan PWM -255.

Gambar 4. 5 Kondisi Motor Diam Pada Titik *Setpoint* = 0.50



Gambar 4. 6 Motor Bergerak Maju Dengan PWM +255 pada posisi < 0,50

Gambar 4. 7 Motor Bergerak Mundur Dengan PWM -255 pada posisi > 0,50

4.5 Pengujian PID

Pengujian nilai PID dilakukan dengan cara *Trial and error* sampai menemukan nilai PID yang sesuai sehingga sistem dapat bekerja dengan sebagaimana mestinya. Untuk menentukan nilai PID yang pertama dilakukan yaitu dengan memasukkan nilai Kp, Ki, dan Kd sama dengan 0. Selanjutnya memasukkan nilai Kp hingga respon sistem sesuai. terlalu sedikit nilai Kp maka menyebabkan respon sistem lemah dan robot mudah terjatuh. Terlalu banyak nilai Kp maka akan membuat robot bergerak secara liar (*shacking*) sehingga robot mudah jatuh. Nilai Kp yang cukup akan membuat robot bergerak dengan baik. Selanjutnya mengatur nilai Kd yaitu untuk mengatur respon ayuna robot dalam mencapai nilai *setpoint*. Nilai Kd mempengaruhi

kecepatan motor terhadap nilai *error*. Nilai Kd yang terlalu kecil tidak dapat menahan kecepatan motor pada saat menuju nilai *setpoint*. Nilai Kd yang terlalu besar dapat menyebabkan osilasi. Nilai Kd yang tepat dapat menyebabkan robot dapat berdiri seimbang sementara waktu. Selanjutnya mengatur nilai Ki. Nilai Ki yang tepat dapat menyebabkan respon robot cepat dalam menyeimbangkan diri. Data hasil pengujian nilai Kp dapat dilihat pada Tabel 4.5.

Tabel 4. 5 Data Pengujian Nilai Kp.

Set point (°)	PID			Keterangan
	Kp	Ki	Kd	
0,50	0	0	0	Robot tidak bergerak
0,50	10	0	0	Robot dapat bergerak, respon motor untuk menuju nilai *setpoint* masih lemah
0,50	25	0	0	Robot dapat bergerak, respon motor masih kurang dalam menuju *setpoint*,robot dapat berdiri selama 3 detik.
0,50	40	0	0	Robot dapat bergerak bolak-balik menyeimbangkan diri, respon motor dalam menuju titik *setpoint* bagus, robot dapat menyeimbangkan diri selama 8 detik tanpa gangguan.
0,50	55	0	0	Robot bergerak secara liar, terlalu banyak getaran (*shacking*) sehingga robot mudah jatuh

Pada Tabel 4.5 nilai Kp yang digunakan adalah 40. Karena respon motor saat menuju titik *setpoint* sudah bagus dan tidak terlalu banyak getaran saat menuju titik *setpoint*. Robot juga dapat menyeimbangkan diri selama 8 detik tanpa gangguan. Jika menggunakan nilai Kp < 40 respon motor dalam menuju set point masih lemah sehingga robot mudah jatuh. Jika menggunakan nilai Kp > 40 robot akan bergerak secara liar dan terlalu banyak getaran (*shacking*) sehingga robot mudah jatuh. Maka nilai Kp yang tepat adalah 40.

Selanjutnya menentukan nilai Kd untuk mengatur ayunan robot pada saat mencapai titik *setpoint*. Data pengujian nilai Kd dapat dilihat pada Tabel 4.6.

Tabel 4. 6 Data Pengujian Nilai Kd

Set point (°)	PID			Keterangan
	Kp	Ki	Kd	
0,50	40	0	0.5	Robot dapat bertahan pada titik *setpoint* selama 4 detik, robot berjalan maju dengan PWM +255 dan tidak dapat kembali ke titik *setpoint*.
0,50	40	0	0.9	Robot dapat bertahan pada titik *setpoint* selama detik. Robot berjalan dengan PWM +255 tetapi tidak dapat kembali ke titik *setpoint*
0,50	40	0	1.5	Robot dapat bertahan pada titik *setpoint* selama 4 detik, setelahnya robot berjalan dengan PWM +255 lebih lama untuk mempertahankan titik *setpoint*.
0,50	40	0	1.9	Robot dapat bertahan pada titik *setpoint* selama 5 detik, robot berjalan bolak-balik untuk mempertahankan titik *setpoint* selama 2 detik. Setelah itu robot jatuh.

Tabel 4.6 Lanjutan

Set point (°)	PID			Keterangan
	Kp	Ki	Kd	
0,50	40	0	2	Robot dapat bertahan pada titik *setpoint* selama 2 detik. Pada titik *setpoint* terdapat getaran (*shacking*) sehingga robot mudah jatuh.

Pada Tabel 4.6 nilai Kd yang digunakan adala 1.9. karena ayunan robot tidak terlalu lemah dan tidak terlalu banyak getaran (*shacking*) sehingga robot dapat bertahan pada titik *setpoint*. Jika nilai Kd < 1.9 ayunan robot lemah sehingga robot tidak dapat bertahan pada titik *setpoint*. Jika Kd > 1.9 ayunan robot terlalu kuat sehingga terlalu banyak getaran pada saat mempertahankan titik *setpoint* sehingga robot mudah jatuh. Maka nilai Kd yang tepat adalah 1.9.

Tabel 4. 7 Data Pengujian Nilai Ki

Set point (°)	PID			Keterangan
	Kp	Ki	Kd	
0,50	40	100	1.9	Robot dapat mempertahankan titik *setpoint* tetapi masih bergerak maju mundur. Pada saat diberi gangguan dorongan robot belum bisa mempertahankan keseimbangan.
0,50	40	140	1.9	Robot dapat mempertahankan titik *setpoint*, bergerak maju mundur untuk memperthankan titik *setpoint* berkurang. Pada saat diberi gangguan dorongan robot belum bisa mempertahankan keseimbangan.

Tabel 4.7 Lanjutan

Set point (°)	PID			Keterangan
	Kp	Ki	Kd	
0,50	40	200	1.9	Robot dapat mempertahankan titik *setpoint*. Bergerak maju mundur untuk mempertahankan keseimbangan. Pada saat diberi gangguan robot mulai bisa mempertahankan keseimbangan.
0,50	40	140	1.9	Robot dapat mempertahankan titik *setpoint*, bergerak maju mundur untuk memperthankan titik *setpoint* berkurang. Pada saat diberi gangguan dorongan robot belum bisa mempertahankan keseimbangan.
0,50	40	200	1.9	Robot dapat mempertahankan titik *setpoint*. Bergerak maju mundur untuk mempertahankan keseimbangan. Pada saat diberi gangguan robot mulai bisa mempertahankan keseimbangan.
0,50	40	270	1.9	Robot dapat mempertahankan titik *setpoint*. Bergerak maju mundur berkurang.dapat mempertahankan keseimbangan saat diberi gangguan
0,50	40	>270	1.9	Robot banyak getaran (*shacking*)

Pada Tabel 4.7 nilai Ki yang tepat adalah 270. Karena pada saat mempertahankan titik *setpoint* tidak terlalu banyak ayunan dan pada saat diberi

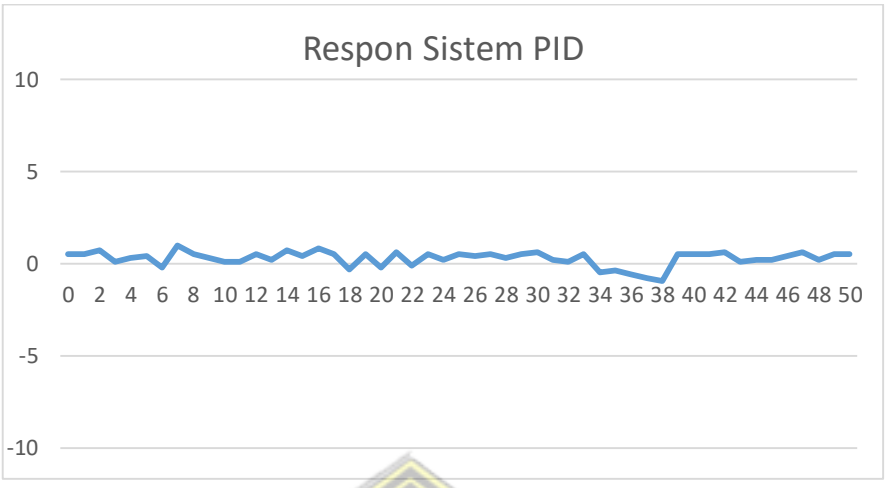
gangguan berupa dorongan robot masih tetap bisa bertahan pada titik *setpoint*. Jika nilai Ki < 270 pada saat mempertahankan nilai *setpoint*, robot masih bergerak maju mundur dan saat diberi gangguan berupa dorongan, robot masih belum bisa mempertahankan keseimbangan. Jika nilai *setpoint* > 270 robot mengalami getaran (*shacking*) sehingga robot mudah terjatuh.

Untuk mengetahui pengaruh nilai Kp terhadap respon sistem, dilakukan dengan cara melakukan penambahan dan pengurangan pada nilai Kp, nilai Ki dan Kd dibuat tetap. Dan didapatkan bahwa nilai Kp yang terlalu cepat membuat respon robot cepat kembali pada titik *setpoint*, akan tetapi menimbulkan osilasi dan respon tidak stabil. Nilai Kp yang kecil respon robot stabil akan tetapi lambat untuk mencapai titik *setpoint*.

Untuk mengetahui pengaruh nilai Ki terhadap respon sistem, dilakukan dengan menambah dan mengurangi nilai Ki, nilai Kp dan Kd dibuat tetap. Nilai Ki yang terlalu besar menyebabkan sulitnya membalikkan arah putar motor saat mencapai titik *setpoint*. Sedangkan nilai Ki yang terlalu kecil menyebabkan respon motor yang lambat untuk mencapai *setpoint*.

Untuk mengetahui pengaruh nilai Kd terhadap respon sistem, dilakukan dengan menambah dan mengurahi nilai Kd, nilai Kp dan Ki dibuat tetap.nilai Kd yang besar akan mengurangi kecepatan motor saat mendekadi *setpoint* dan akan mengurangi *overshoot*. Tetapi saat ada penambahan *error* kecepatan motor akan bertambah dan menyebabkan osilasi. Nilai Kd yang terlalu kecil tidak dapat menahan kecepatan motor saat mengembalikan nilai sudut pada *set point* sehingga menimbulkan *over shoot.*

Setelah mendapatkan nilai Kp, Ki, dan Kd selanjutnya adalah melakukan pengujian keseimbangan pada robot. Melihat apakah robot dapat berdiri seimbang mempertahankan pada titik *setpoint* atau tidak.



Gambar 2. 19 Respon Sistem Kendali PID

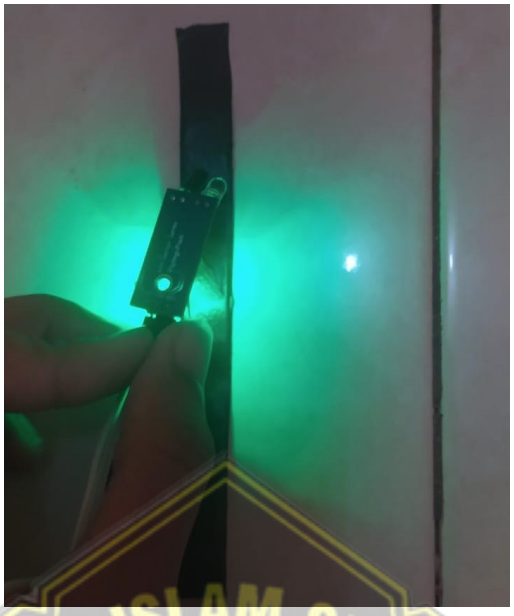
Pada Gambar 2.19 respon sistem kendali PID dengan nilai parameter Kp = 40,Ki = 270, Kd = 1,9, robot dapat berdiri dengan stabil, robot mampu mempertahankan keseimbangan pad atitik *setpoint* dan perubahan nilai sensor tidak terlalu signifikan, sehinga dapat disimpulkan bahwa nilai parameter Kp, Ki, dan Kd tepat.

4.6 Pengujian Sensor Garis

Pengujian selanjutnya yaitu pengujian sensor garsi. Robot ini menggunakan 2 buah sensor IR sebagai pendeteksi garis hitam pada *background* warna putih. Sensor dilektakkan pada bagian depan bawah robot sehingga mendeteksi garis bisa maksimal. Sensor garis terdiri dari transiver dan *receiver*. Data yang diterima oleh *receiver* akan diproses oleh mikrokontroler arduino nano agar robot dapat mengikuti garis hitam. Hasil pengujian sensor garis ditunjukkan pada tabel 4.8.

Tabel 4. 8 Hasil Pengujian Sensor Garis

Sensor Input		Output
Sensor 1	Sensor 2	
1	0	Kiri
0	1	Kanan
1	1	Balancing
0	0	Balancing



Gambar 4. 8 Pengujian Sensor Garis

Dapat diperoleh data seperti pada Tabel 4.8 sensor menggunakan prinsip kerja dari pembagi tegangan. Selanjutnya *output* yang dikeluarkan dari sensor akan di komparasikan oleh komparator sehingga menghasilkan nilai 1 berarti mendekati VCC dan 0 berarti mendekati – VCC. Prinsip kerja komparator yaitu Jika V+ > V- maka tegangan *Output* adalah VCC, jika V+ < V- maka tegangan *Output* adalah –VCC.

Menghitung *Output* sensor dengan pembagi tegangan :

Diketahui jika D1 = 150 kΩ

R3 = 10 kΩ

Vin = 5v

Maka :

Berdasarkan pada persamaan 2.7 didapatkan nilai Vout sebagai berikut :

$$Vout = \frac{D1}{D1 + R3}\ x\ Vin$$

$$Vout = \frac{150}{150 + 10}\ x\ 5$$

$$Vout = 4{,}68\ V$$

Jika diketahui D1 = 10 kΩ

R3 = 10 kΩ

Vin = 5v

Maka :

$$Vout = \frac{D1}{D1 + R3}\ x\ Vin$$

$$Vout = \frac{10}{10 + 10}\ x\ 5$$

$$Vout = 2{,}5\ V$$

Setelah mengetahui nilai Vout dari sensor selanjutnya nilai *Output* akan dibandingkan dengan komparator. Jika V+ > V- maka tegangan *Output* adalah VCC, jika V+ < V- maka tegangan *Output* adalah –VCC. Jika V- mendekati 5 v maka sensor *HIGH* atau 1. Jika V- Mendekati 0 v maka sensor *LOW* atau 0.

Gambar 4. 9 Robot Pada Saat Melintasi Garis

4.7 Pengujian Beban Robot

Setelah *self balancing* robot sudah dapat menyeimbangkan diri dengan nilai Kp, Ki, dan Kd yang tepat, selanjutnya perlu mengetahui spesifikasi dari robot. Pengujian selanjutnya yaitu pengujian beban pada robot. Pengujian beban robot bertujuan untuk mengetahui spesifikasi dari robot dan seberapa kuat robot mengangkat beban maksimum. Berat robot tanpa beban yaitu 483 gram. Pengujian dilakukan dengan beban 8 gram sampai 186 gram.

Gambar 4. 10 Berat Robot Tanpa Beban

Pengujian beban robot dapat dilihat pada Tabel 4.10.

Tabel 4. 10 Pengujian Beban Robot

Beban (gram)	Keterangan
8	Robot masih dapat menyeimbangkan diri dengan baik.
25	Robot masih bisa mempertahankan keseimbangan dengan baik.
45	Robot masih bisa mempertahankan keseimbangan dengan berjalan maju dan mundur.
55	Robot masih bisa mempertahankan keseimbangan dengan berjalan maju dan mundur.
70	Robot masih bisa mempertahankan keseimbangan dengan berjalan maju dan mundur.

Tabel 4.9 Lanjutan

85	Robot masih bisa mempertahankan keseimbangan dengan berjalan maju dan mundur.
95	Robot masih bisa mempertahankan keseimbangan dengan berjalan maju dan mundur lebih jauh.
100	Robot masih bisa mempertahankan keseimbangan dengan berjalan maju dan mundur lebih jauh.
150	Robot masih bisa mempertahankan keseimbangan dengan berjalan maju dan mundur selama 15 detik
170	Robot masih bisa mempertahankan keseimbangan dengan berjalan maju dan mundur selama 10 detik.

Gambar 4. 11 Robot Saat Membawa Beban

Pada Tabel 4.10 robot dapat mengangkat beban dengan baik dengan beban kurang dari 8 gram dan maksimal 100 gram. Pada beban 150 gram dan 170 gram robot hanya mampu mempertahankan keseimbangan selama 10-15

detik dikarenakan beban terlalu berat sehingga robot tidak dapat mempertahankan keseimbangan pada titik *setpoint*.

Dengan menggnakan sensor MPU-6050 dan Sensor Photodioda robot serta menggunakan metode PID dengan nilai Kp = 40, Kd = 1,9, Kd = 270 dapat mempertahankan keseimbangan dengan cukup baik serta dapat mengikuti garis hitam dengan lebar garis 1 cm pada *Background* berwarna putih. Dengan desain robot yang sedemikian rupa, robot dapat membawa beban maksimal 170 gram dan mampu mempertahankan keseimbangan selama 10-15 detik.

BAB V

PENUTUP

5.1 Kesimpulan

Kesimpulan dari hasil penelitian tersebut adalah sebagai berikut

1. Sensor MPU-6050 memiliki pembacaan yang kurang akurat pada sudut -90, -45, 45, dan 90.
2. Robot dapat mempertahankan diri pada titik setpoint dengan menggunakan metode PID trial and error dan didapatkan nilai Kp = 40, Ki = 270, dan Kd = 1,9.
3. Robot dapat mempertahan keseimbangan dengan mengikuti garis hitam dengan lebar garis 1 cm pada background putih dengan 2 buah sensor Photodioda.

5.2 Saran

1. Menggunakan metode lain dalam penerapan self balancing robot.
2. Memilih sensor MPU-6050 dengan pembacaan nilai yang akurat dan presisi sehingga kinnerja robot lebih maksimal.
3. Membuat self balancing robot line follower yang dapat melintasi garis yang memiliki lebih banyak tikungan.
4. Semakin tinggi robot maka beban robot akan semakin berat, sehingga motor harus memiliki torsi yang kuat dan speed yang tinggi sehingga dapat menahan beban robot dan dapat merespon dengan cepat saat setpoint berubah.

DAFTAR PUSTAKA

[1] Raranda and P. W. Rusimamto, "Implementasi Kontroler Pid Pada Two Wheels Self Balancing Robot Berbasis Arduino Uno," *J. Tek. Elektro*, vol. 6, no. 2, pp. 89–96, 2017.

[2] A. Pratama, "Implementasi Pid Controller Pada Self Balancing Robot," *J. FTIE UTY*, 2018.

[3] A. Adella, M. Kamal, and A. Finawan, "Rancang Bangun Robot Mobile Line Follower Pemindah Minuman Kaleng Berbasis Arduino," *J. TEKTRO*, vol. 2, no. 2, 2018.

[4] K. Joni, M. Ulum, and Z. Abidin, "Robot Line Follower Berbasis Kendali Proportional- Integral-Derivative (PID) Untuk Lintasan Dengan Sudut Ekstrim," *J. Infotel*, vol. 8, no. 2, pp. 138–142, 2016.

[5] A. Jatmiko, "Prototype Robot Balancing/Keseimbangan Dengan Kendali PID," *Repos. Unissula*, 2018.

[6] M. Arifin and B. Harsono, "Two Wheels Self-Balancing Robot Berbasis Arduino Nano Menggunakan Metode PID," *J. Elektro*, vol. 9, pp. 69–80, 2016.

[7] J. Desember, E. Setyaningsih, and D. Prastiyanto, "Penggunaan Sensor Photodioda sebagai Sistem Deteksi Api pada Wahana Terbang Vertical Take-Off Landing (VTOL)," *J. Tek. Elektro*, vol. 9, no. 2, pp. 53–59, 2017.

[8] D. Muhammad Hasan Basri, "Penggunaan Motor Sinkron Tiga Phasa Tipe Salient Pole Sebagai Generator Sinkron Denny," *J. Simetrik*, vol. 9, no. 2, pp. 208–214, 2019.

[9] R. Rio Bagus, Teknik, D. T. Elektro, and U. N. S. Teknik, "Pengembangan Two Wheels Self Balancing Robot Dengan Pi Controller Berbasis Labview 2014 Bagus Rio Rynaldo Endryansyah pada Two Wheels Self Balancing Robot Berbasis Arduino," vol. 7, pp. 127–136, 2018.

[10] R. Birdayansyah, N. Soedjarwanto, and O. Zebua, "Pengendalian Kecepatan Motor DC Menggunakan Perintah Suara Berbasis Mikrokontroler Arduino," *Rekayasa dan Teknol. Elektro Pengendali.*, vol. 9, no. 2, pp. 96–107, 2015.

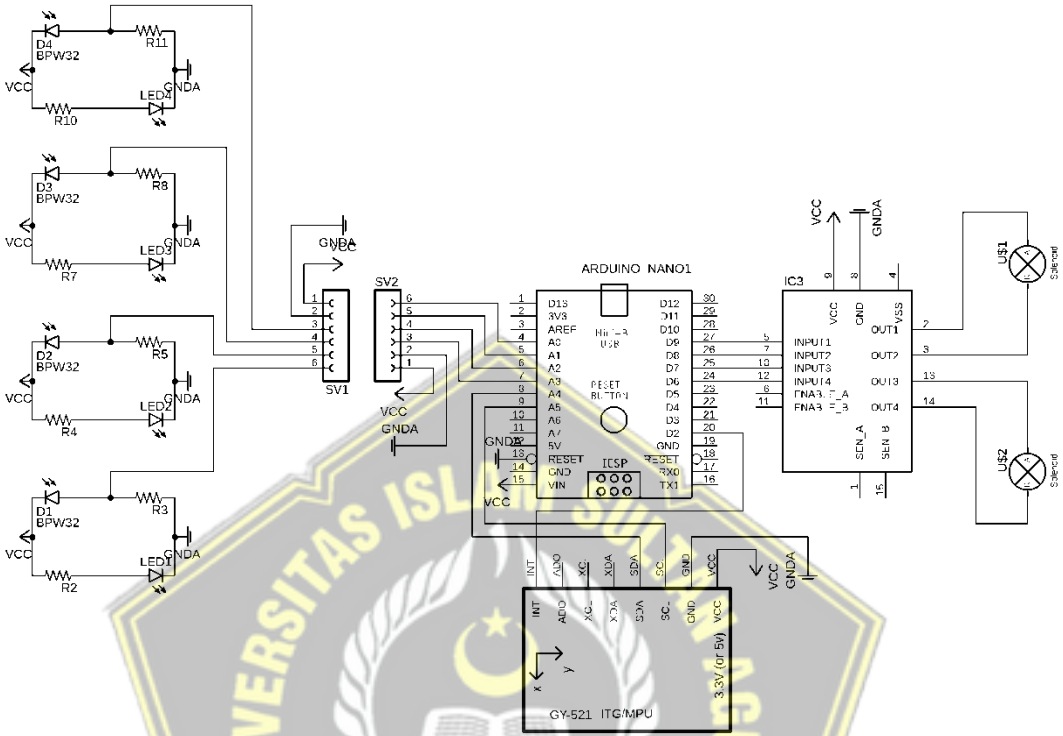
[11] D. Setiawan, J. Yos Sudarso Km, K. Kunci, and A. Uno, "Sistem Kontrol Motor Dc Menggunakan Pwm Arduino Berbasis Android System," *J. Sains, Teknol. dan Ind.*, vol. 15, no. 1, pp. 7–14, 2017.

[12] L. Katriani, D. Darmawan, and A. Noer, "Rancang Bangun Sistem Kontrol Box Uv Sebagai Media Sterilisasi Menggunakan Sensor Fotodioda," *J. Sains Dasar*, vol. 4, no. 1, pp. 71–76, 2015.

[13] S. Ruswanto, E. S. Ningrum, and I. Ramli, "Pengaturan Gerak Dan Keseimbangan Robot Line Tracer Dua Roda Menggunakan PID Controller," *13th Ind. Electron. Semin. 2011*, vol. 2011, no. Ies, pp. 978–979, 2011.

Lampiran

Skema Rangkaian



Desain Robot

Program Robot

```
#include <PID_v1.h>


//#include <PID_v1.h>
#include <LMotorController.h>
#include "I2Cdev.h"


#include "MPU6050_6Axis_MotionApps20.h"


#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    #include "Wire.h"
#endif
#define LOG_INPUT 0
#define MANUAL_TUNING 0
#define LOG_PID_CONSTANTS 0 //MANUAL_TUNING must be 1
#define MOVE_BACK_FORTH 0
#define MIN_ABS_SPEED 30
//MPU
MPU6050 mpu;
// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was
successful
uint8_t mpuIntStatus;   // holds actual interrupt
status byte from MPU
uint8_t devStatus;      // return status after each
device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size
(default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently
in FIFO
```

```
uint8_t fifoBuffer[64]; // FIFO storage buffer


// orientation/motion vars
Quaternion q;           // [w, x, y, z]
quaternion container
VectorFloat gravity;    // [x, y, z]            gravity
vector
float ypr[3];           // [yaw, pitch, roll]
yaw/pitch/roll container and gravity vector
//PID
#if MANUAL_TUNING
  double kp , ki, kd;
  double prevKp, prevKi, prevKd;
#endif
double originalSetpoint = 176.29;//-1.10; //176.29;
double setpoint = originalSetpoint;
double movingAngleOffset = 0.3;
double input, output;
int moveState=0; //0 = balance; 1 = back; 2 = forth


#if MANUAL_TUNING
  PID pid(&input, &output, &setpoint, 0, 0, 0, DIRECT);
#else
  PID pid(&input, &output, &setpoint, 40, 270, 1.9,
DIRECT);
#endif


//photodioda
int IR1=10;      //Right sensor
int IR2=11;    //left Sensor
```

```
int a;
int b;
//MOTOR CONTROLLER
int ENA = 3;
int IN1 = 8;
int IN2 = 4;
int IN3 = 7;
int IN4 = 5;
int ENB = 6;
LMotorController motorController(ENA, IN1, IN2, ENB,
IN3, IN4, 0.6, 1);
//timers
long time1Hz = 0;
long time5Hz = 0;
volatile bool mpuInterrupt = false;     // indicates
whether MPU interrupt pin has gone high
void dmpDataReady()
{
    mpuInterrupt = true;
}
void setup()
{
  pinMode(IR1, INPUT);
  pinMode(IR2, INPUT);
    // join I2C bus (I2Cdev library doesn't do this
automatically)
    #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
        Wire.begin();
```

```
        TWBR = 24; // 400kHz I2C clock (200kHz if CPU
is 8MHz)
    #elif I2CDEV_IMPLEMENTATION ==
I2CDEV_BUILTIN_FASTWIRE
        Fastwire::setup(400, true);
    #endif

    // initialize serial communication
    // (115200 chosen because it is required for Teapot
Demo output, but it's
    // really up to you depending on your project)
    Serial.begin(115200);
    while (!Serial); // wait for Leonardo enumeration,
others continue immediately

    // initialize device
    Serial.println(F("Initializing I2C devices..."));
    mpu.initialize();

    // verify connection
    Serial.println(F("Testing device connections..."));
    Serial.println(mpu.testConnection() ? F("MPU6050
connection successful") : F("MPU6050 connection
failed"));

    // load and configure the DMP
    Serial.println(F("Initializing DMP..."));
    devStatus = mpu.dmpInitialize();

    // supply your own gyro offsets here, scaled for
min sensitivity
```

```
    mpu.setXGyroOffset(220);
    mpu.setYGyroOffset(76);
    mpu.setZGyroOffset(-85);
    mpu.setZAccelOffset(1788); // 1688 factory default
for my test chip

    // make sure it worked (returns 0 if so)
    if (devStatus == 0)
    {
        // turn on the DMP, now that it's ready
        Serial.println(F("Enabling DMP..."));
        mpu.setDMPEnabled(true);

        // enable Arduino interrupt detection
        Serial.println(F("Enabling interrupt detection
(Arduino external interrupt 0)..."));
        attachInterrupt(0, dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();

        // set our DMP Ready flag so the main loop()
function knows it's okay to use it
        Serial.println(F("DMP ready! Waiting for first
interrupt..."));
        dmpReady = true;

        // get expected DMP packet size for later
comparison
        packetSize = mpu.dmpGetFIFOPacketSize();

        //setup PID
```

```
        pid.SetMode(AUTOMATIC);
        pid.SetSampleTime(10);
        pid.SetOutputLimits(-255, 255);
    }
    else
    {
        // ERROR!
        // 1 = initial memory load failed
        // 2 = DMP configuration updates failed
        // (if it's going to break, usually the code
will be 1)
        Serial.print(F("DMP Initialization failed (code
"));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }
 // pinMode(IR1,INPUT);
 // pinMode(IR2,INPUT);
}
void loop()
{
    // if programming failed, don't try to do anything
    if (!dmpReady) return;

    // wait for MPU interrupt or extra packet(s)
available
    while (!mpuInterrupt && fifoCount < packetSize)
    {
```

```
        //no mpu data - performing PID calculations and
output to motors
         Serial.println(setpoint);
         pid.Compute();
         Serial.print (" y : "); Serial.println(input);
//Serial.print(" =>"); Serial.println(output);
         motorController.move(output, MIN_ABS_SPEED);


        if(input>150 && input<200)
        {
          a = digitalRead(IR1);
          b = digitalRead(IR2);

          if(output>0)
          {
            if(a == HIGH && b == HIGH)
            {
              /*digitalWrite(IN1,LOW);
                digitalWrite(IN2,LOW);
                digitalWrite(IN3,LOW);
                digitalWrite(IN4,LOW);
                analogWrite (ENA, 0);
                analogWrite (ENB, 0);*/
                pid.Compute();
                motorController.move(output,
MIN_ABS_SPEED);
            }
            else if(a == LOW && b == LOW)
            {
```

```
            /*digitalWrite(IN1,HIGH);
          digitalWrite(IN2,LOW);
          digitalWrite(IN3,LOW);
          digitalWrite(IN4,HIGH);
          analogWrite (ENA, 200);
          analogWrite (ENB, 200);*/
           pid.Compute();
                motorController.move(output,
MIN_ABS_SPEED);
            }
            else if(a == LOW && b == HIGH)
            {
              digitalWrite(IN1,HIGH);
          digitalWrite(IN2,LOW);
          digitalWrite(IN3,LOW);
          digitalWrite(IN4,HIGH);
          analogWrite (ENA, 200);
          analogWrite (ENB, 100);
            }
            else if(a == HIGH && b == LOW)
            {
              digitalWrite(IN1,LOW);
          digitalWrite(IN2,HIGH);
          digitalWrite(IN3,HIGH);
          digitalWrite(IN4,LOW);
          analogWrite (ENA, 100);
          analogWrite (ENB, 200);
            }
          }
```

```
          else if(output == -255)
          {
             pid.Compute();
                 motorController.move(output,
MIN_ABS_SPEED);
          }
        }
        else
        {
           pid.Compute();
                 motorController.move(output,
MIN_ABS_SPEED);
        }

       /* unsigned long currentMillis = millis();

        if (currentMillis - time1Hz >= 1000)
        {
            loopAt1Hz();
            time1Hz = currentMillis;
        }


        if (currentMillis - time5Hz >= 5000)
        {
            loopAt5Hz();
            time5Hz = currentMillis;
        }*/
    }
```

```
    // reset interrupt flag and get INT_STATUS byte
    mpuInterrupt = false;
    mpuIntStatus = mpu.getIntStatus();

    // get current FIFO count
    fifoCount = mpu.getFIFOCount();

    // check for overflow (this should never happen
unless our code is too inefficient)
    if ((mpuIntStatus & 0x10) || fifoCount == 1024)
    {
        // reset so we can continue cleanly
        mpu.resetFIFO();
        Serial.println(F("FIFO overflow!"));

    // otherwise, check for DMP data ready interrupt
(this should happen frequently)
    }
    else if (mpuIntStatus & 0x02)
    {
        // wait for correct available data length,
should be a VERY short wait
        while (fifoCount < packetSize) fifoCount =
mpu.getFIFOCount();

        // read a packet from FIFO
        mpu.getFIFOBytes(fifoBuffer, packetSize);

        // track FIFO count here in case there is > 1
packet available
```

```
        // (this lets us immediately read more without
waiting for an interrupt)
        fifoCount -= packetSize;

        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
        #if LOG_INPUT
            Serial.print("ypr\t");
            Serial.print(ypr[0] * 180/M_PI);
            Serial.print("\t");
            Serial.print(ypr[1] * 180/M_PI);
            Serial.print("\t");
            Serial.println(ypr[2] * 180/M_PI);
        #endif
        input = ypr[1] * 180/M_PI + 180;
   }
   //Serial.print("y:");
Serial.println(input);if(Serial.available()){setpoint =
input;}
}

/*void sensor()
{
 if(digitalRead(IR1)==HIGH && digitalRead(IR2)==HIGH)
        {
          digitalWrite(IN1,LOW);
          digitalWrite(IN2,LOW);
          digitalWrite(IN3,LOW);
```

```
          digitalWrite(IN4,LOW);
          analogWrite (ENA, 0);
          analogWrite (ENB, 0);
        }
        if(digitalRead(IR1)==LOW &&
digitalRead(IR2)==LOW)
        {
          digitalWrite(IN1,HIGH);
          digitalWrite(IN2,LOW);
          digitalWrite(IN3,LOW);
          digitalWrite(IN4,HIGH);
          analogWrite (ENA, 200);
          analogWrite (ENB, 200);
        }
        if(digitalRead(IR1)==LOW &&
digitalRead(IR2)==HIGH)
        {
          digitalWrite(IN1,HIGH);
          digitalWrite(IN2,LOW);
          digitalWrite(IN3,LOW);
          digitalWrite(IN4,HIGH);
          analogWrite (ENA, 200);
          analogWrite (ENB, 100);
        }
        if(digitalRead(IR1)==HIGH &&
digitalRead(IR2)==LOW)
        {
          digitalWrite(IN1,LOW);
          digitalWrite(IN2,HIGH);
```

```
            digitalWrite(IN3,HIGH);
            digitalWrite(IN4,LOW);
            analogWrite (ENA, 100);
            analogWrite (ENB, 200);
          }
}*/
/*void loopAt1Hz()
{
#if MANUAL_TUNING
    setPIDTuningValues();
#endif
}


void loopAt5Hz()
{
    #if MOVE_BACK_FORTH
        moveBackForth();
    #endif
}


//move back and forth


void moveBackForth()
{
    moveState++;
```

```
    if (moveState > 2) moveState = 0;

    if (moveState == 0)
      setpoint = originalSetpoint;
    else if (moveState == 1)
      setpoint = originalSetpoint - movingAngleOffset;
    else
      setpoint = originalSetpoint + movingAngleOffset;
}


//PID Tuning (3 potentiometers)

#if MANUAL_TUNING
void setPIDTuningValues()
{
    readPIDTuningValues();

    if (kp != prevKp || ki != prevKi || kd != prevKd)
    {
#if LOG_PID_CONSTANTS
        Serial.print(kp);Serial.print(",
");Serial.print(ki);Serial.print(",
");Serial.println(kd);
#endif

        pid.SetTunings(kp, ki, kd);
        prevKp = kp; prevKi = ki; prevKd = kd;
    }
```

```
}



void readPIDTuningValues()
{
    int potKp = analogRead(A0);
    int potKi = analogRead(A1);
    int potKd = analogRead(A2);


    kp = map(potKp, 0, 1023, 0, 25000) / 100.0; //0 -
250
    ki = map(potKi, 0, 1023, 0, 100000) / 100.0; //0 -
1000
    kd = map(potKd, 0, 1023, 0, 500) / 100.0; //0 - 5
}
#endif*/
```