IMPLEMENTASI LOGIKA *FUZZY* UNTUK PENENTUAN PARAMETER KONTROLER PID PADA *BALANCING ROBOT* BERODA DUA BERBASIS MIKROKONTROLLER ARDUINO UNO

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK KONTROL

Ditujukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

AGUNG PAMBUDI

NIM. 145060300111049

UNVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2018

LEMBAR PENGESAHAN

IMPLEMENTASI LOGIKA FUZZY UNTUK PENENTUAN PARAMETER KONTROLER PID PADA BALANCING ROBOT BERODA DUA BERBASIS MIKROKONTROLLER ARDUINO UNO

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK KONTROL

Diajukan untuk memenuhi persyaratan

memperoleh gelar Sarjana Teknik

AGUNG PAMBUDI

NIM. 145060300111049

Skripsi ini telah direvisi dan disetujui oleh dosen pembimbing

Pada tanggal 26 Oktober 2018

Dosen Pembimbing

Rahmadwati, S.T., M.T., Ph.D

NIP. 19771102 200604 2 003

Mengetahui,

Ketua Jurusan Teknik Elektro

Ir. Hadi Suyono, S.T., M.T., Ph.D. IPM

NIP. 19730520 200801 1 013

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya dan berdasarkan hasil penelusuran berbagai karya ilmiah, gagasan dan masalah ilmiah yang diteliti dan diulas di dalam Naskah Skripsi ini adalah asli dari pemikiran saya. Tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu Perguruan Tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah Skripsi ini dapat dibuktikan terdapat unsur-unsur jiplakan, saya bersedia Skripsi dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, pasal 25 ayat 2 dan pasal 70).

Malang, 26 Oktober 2018

Mahasiswa,

AGUNG PAMBUDI
NIM. 145060300111049

Teriring Ucapan Terima Kasih kepada :

Ibu dan Bapak Tersayang

Mas dan Adek

RINGKASAN

Agung Pambudi, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, September 2018, *Implementasi Logika Fuzzy Untuk Penentuan Paramater Kontroler PID Pada Balancing Robot Beroda Dua Berbasis Mikrokontroller Arduino Uno*, Dosen Pembimbing: Rahmadwati.

Perkembangan ilmu pengetahuan menuntut adanya berbagai inovasi salah satunya pada bidang robotika. Salah satunya adalah robot beroda dua, konsep robot beroda dua telah digunakan sebagai alat transportasi yang bernama *segway*. Tugas akhir ini juga akan membuat alat yang sama dalam prinsip kerjanya, namun dengan ukuran yang berbeda. Alat (robot beroda dua) ini nantinya dibuat agar dapat menyeimbangkan dirinya sendiri sehingga tidak jatuh. Kedua roda robot dihubungkan dengan motor DC sebagai penggerak secara efisien dan efektif. Tugas akhir ini menerapkan kendali PID sebagai sistem regulator pada plant balancing robot yang memiliki prinsip kerja mirip dengan pendulum terbalik. Dalam pembuatan tugas akhir balancing robot menggunakan logika *Fuzzy* untuk tuning parameter kendali PID. Kecepatan putaran dua motor DC yang digunakan sebagai penggerak dapat diatur dengan mengatur tegangan masukan. Sensor *Inertial Measurement Unit* (IMU) MPU-6050 digunakan sebagai pendeteksi sudut kemiringan *balancing robot*. Tugas akhir ini merancang dan mengimplementasikan metode tuning PID dengan pendekatan *Fuzzy* (*fuzzy-PID)*. Yang pada pernancangan kontroler desain robot dan nilai sensor dijadikan patokan sebagai nilai set untuk dapat menentukan nilai *gain controller* untuk nilai *Proporsional*, *Integral*, dan *difrential*. Dengan menggunakan sensor IMU MPU-6050 robot mampu mendeteksi sudut kemiringan balancing robot dapat diketahui rata-rata kesalahan pengukuran, sehingga dapat mengurangi proses *handtuning (trial dan error)* pada proses kalibrasi nilai PID. *Balancing robot* dapat menyeimbangkan diri ketika tanpa gangguan dan dengan gangguan. Pada pengujian dengan gangguan searah jarum jam (CW) sudut maksimal yang masih dapat diseimbangkan oleh sistem adalah sebesar $18{,}2^0$, sedangkan pada gangguan berlawanan arah jarum jam (CCW) sebesar $-21{,}77^0$. Dengan sistem tersebut, robot mampu menjaga keseimbangan dan tetap stabil tegak lurus dengan permukaan bumi pada bidang datar.

Kata Kunci – *Fuzzy*, sensor IMU MPU-6050, kendali PID, *balancing robot*

SUMMARY

Agung Pambudi, Department of Electrical Engineering, Faculty of Engineering Universitas Brawijaya, September of 2018. *The Implementation of Fuzzy Logic for Deternining Parameters of Differential Integral Proportional Controllers On Two-Wheeled Balancing Robot With Arduino Uno Microcontroller-Based*. Academic Supervisor: Rahmadwati.

The development of science demands the existence of various innovations, one of them is in the field of robotics. One of the innovations is two-wheeled robot, the concept of this two-wheeled robot has been used as a transportation device named segway. This final project would also make the same device in its working principle, yet in the different size. Later on, this device (two-wheeled robot) would be made to be able to balance itself so that it would not fall. The two robot wheels are connected to the DC motor as an efficient and effective driving force. This final project re-applies PID as a regulator system in robotic plant balancing that has a working principle similar to an inverted pendulum. In making the final project, the balancing robot uses fuzzy logic for tuning PID control parameter. The rotation speed of two DC motors used as a booster can be adjusted by adjusting the input voltage. Inertial Measurement Unit (IMU) MPU-6050 sensor is used as a tilt angle detector of the balancing robot. This final project designs and implements the PID tuning method with a fuzzy approach (fuzzy-PID), which is used as a criteria on the design of the robot design controller and the sensor value as a set value to be able to determine the value of the controller gain for the proportional, integral, and differential values. By using the IMU MPU-6050 sensor, the robot is able to detect the angle of the robot balancing angle and the average of measurement error can be known, so that the handtuning process (trial and error) in the calibration process of PID value can be reduced. Balancing robot can balance themselves when without interference and with interference. On testing with a clockwise (CW) interference the maximum angle which is balanced by the system equal to 18.2^0, while the counter clockwise (CCW) interference equal to $-21{,}77^0$. With such system, the robot is able to maintain its balance and remain perpendicularly stable to the surface of the earth on the flat field.

Keywords – *fuzzy*, IMU MPU-6050 sensor, PID controller, *balancing robot*

PENGANTAR

Bismillahirrohmanirrohim. Alhamdulillah, puji dan syukur penulis panjatkan kepada Allah SWT, karena atas segala petunjuk dan nikmat-Nya lah skripsi ini dapat diselesaikan. Skripsi berjudul "*IMPLEMENTASI LOGIKA FUZZY UNTUK PENENTUAN PARAMETER KONTROLER PID PADA BALANCING ROBOT BERODA DUA BERBASIS MIKROKONTROLLER ARDUINO UNO*" ini disusun untuk memenuhi persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

Penulis menyadari bahwa dalam penyusunan skripsi ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, dengan ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada:

1. Kedua orang tua, kakak, dan adik saya yang selalu menjadi motivator dalam pengerjaan skripsi ini.
2. Saudara-saudara saya di Tuban yang selalu memberi motivasi untuk bisa mempercepat pengerjaan skripsi ini.
3. Bapak Ir. Hadi Suyono, S.T., M.T., Ph.D., IPM. selaku Ketua Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.
4. Ibu Ir. Nurussa'adah, M.T. selaku Sekretaris Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.
5. Bapak Ali Mustofa, S.T., M.T. selaku Ketua Program Studi Sarjana Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.
6. Bapak Ir. Purwanto, M.T. selaku Ketua Kelompok Dosen Konsentrasi Teknik Kontrol Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.
7. Ibu Rahmadwati, S.T., M.T., Ph.D. selaku Dosen Pembimbing atas segala bimbingan, pengarahan, saran, dan kritik yang telah diberikan selama proses pengerjaan skripsi.
8. Segenap dosen pengajar dan staff administrasi Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.
9. Sahabat-sahabat *LODAN CREW (*anang, luthfi, dede, yusuf, jihad, haekal, syahrul, dan wahyu) dalam memberikan semangat dan saran dalam pengerjaan skripsi.
10. Teman-teman sebimbingan Rizki Zein, Vilardl, dan Andrian atas motivasi yang telah diberikan.
11. Meilan Sarbaini dalam memberikan semangat, saran, dan bantuannya dalam pengerjaan skripsi.
12. Teman-teman Konsentrasi Teknik Kontrol Universitas Brawijaya.

13. Teman-teman DIODA 2014 atas segala bantuan dan kebersamaan yang telah diberikan selama masa studi.
14. Andika Mada dalam memberikan semangat dan motivasi untuk mempercepat pengerjaan skripsi ini.
15. Semua pihak yang telah memberikan bantuan dan dukungan baik secara langsung maupun tidak langsung atas penyelesaian skripsi ini.

Dalam penyusunan skripsi ini, penulis menyadar bahwa masih terdapat kekurangan karena kendala dan keterbatasan dalam pengerjaan skripsi ini. Oleh karena itu, penulis berharap saran dan kritik yang membangun untuk penyempurnaan tulisan di masa yang akan datang. Semoga tulisan ini dapat bermanfaat dan dapat digunakan untuk pengembangan lebih lanjut.

Malang, Oktober 2018

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi robotika telah membuat kualitas kehidupan manusia semakin tinggi. Saat ini perkembangan teknologi robotika telah mampu meningkatkan kualitas maupun kuantitas berbagai industri. Penelitian dan pengembangan pertama tentang robot dimulai dari tahun 1940-an ketika Argonne National Laboratories di Oak Ridge, Amerika, memperkenalkan sebuah mekanisme robot yang diberi nama *master-slave manipulator*. Robot tersebut dipakai untuk menangani material radioaktif. Pada pertengahan tahun 1960-an produk robot dipakai dalam kegiatan industri. Karena aplikasi robot hampir tak dapat dipisahkan dengan kegiatan industri, maka muncul istilah industrial robot dan robot manipulator. Dewasa ini definisi robot industri sudah tidak sesuai lagi karena teknologi *mobile robot* sudah dipakai meluas sejak awal 1980-an. Seiring itu pula muncul istilah *robot humanoid* (mirip manusia), *animaloid* (mirip binatang), dan sebagainya.

Teknologi robotika juga telah menjangkau sisi hiburan dan pendidikan bagi manusia. Salah satu cara menambah tingkat kecerdasan sebuah robot adalah dengan menambah sensor, metode kontrol bahkan memberikan kecerdasan buatan pada robot tersebut. Salah satunya adalah robot beroda dua. Robot beroda dua merupakan suatu *mobile robot* yang memiliki sebuah roda disisi kanan dan kirinya yang tidak akan seimbang apabila tanpa adanya kontroler. Menyeimbangkan robot beroda dua memerlukan suatu metode kontrol yang baik dan handal untuk mempertahankan posisi robot dalam keadaan tegak lurus terhadap permukaan bumi tanpa memerlukan pengendali lain dari luar. Bahkan sekarang ini konsep robot beroda dua telah digunakan sebagai alat transportasi yang bernama *segway*.

Penelitian ini mengembangkan penelitian sebelumnya yang menggunakan kontroler Proporsional Integral Diferensial (PID), dapat menyempurnakan konstruksi mekanik terutama pada gearbox motor DC agar lebih semitris, dan pada penelitian ini ditambahkan metode pengontrolan logika *Fuzzy*. Kelebihan menggunakan metode pengontrolan logika *Fuzzy* adalah teknik pengontrolan logika *Fuzzy* memiliki cara yang lebih sederhana dalam memberikan keputusan seperti halnya manusia berpikir, dengan menafsirkan data dan

mencari solusi yang lebih tepat, dan logika *Fuzzy* bekerja membantu untuk meminimalkan *overshot/undershot* yang terjadi dan juga meminimalkan *recovery time* dari respon sistem. Sistem kendali logika *Fuzzy* yang didesain mempunyai dua input yaitu *error* dan *delta error* dan *output* kecepatan motor DC. Serta pada penelitian ini mendesain dan membangun balancing robot beroda dua yang mampu menyeimbangkan dirinya sehingga tegak lurus terhadap permukaan bumi pada bidang datar. Pada Penelitian ini menggunakan mikrokontroller arduino, dan sensor IMU MPU-6050. Kontrol *Fuzzy* digunakan untuk tuning parameter kendali PID, sehingga menentukan besarnya kecepatan motor DC sebagai penggerak, berdasarkan sudut kemiringan dari badan robot yang dibaca oleh sensor IMU MPU-6050.

Dalam skripsi ini, dirancang suatu metode kendali kombinasi *Fuzzy* dan PID yang diharapkan mampu mempercepat proses tuning dan meningkatkan kemampuan robot dalam menyeimbangkan badan robot. Sistem tersebut dirancang untuk mendapatkan respon yang cepat dan handal sehingga *balancing robot* otomatis diharapkan dapat dengan cepat menyeimbangkan badan robot.

1.2 Rumusan Masalah

Dari permasalahan yang telah dijelaskan pada latar belakang, terdapat beberapa permasalahan, antara lain :

1. Bagaimana mengimplementasikan sistem pengendalian keseimbangan robot beroda dua dengan metode *Fuzzy*-PID pada *balancing robot* berbasis Arduino Uno.
2. Bagaimana membuat aturan aturan fuzzyfikasi dan defuzzyfikasi yang sesuai untuk kalibrasi nilai PID.

1.3 Batasan Masalah

Dengan mengacu pada permasalahan yang telah dirumuskan, maka hal-hal yang berkaitan dengan alat akan diberi batasan sebagai berikut :

1. *Balancing robot* yang dibuat memiliki 2 roda.
2. Motor DC yang digunakan adalah motor DC 12 Volt.
3. Mikrokontroler yang digunakan Arduino Uno.
4. Sensor yang digunakan sensor IMU MPU-6050.

5. Alat ini hanya dapat menyeimbangkan diri tegak lurus dengan permukaan bumi pada bidang datar.
6. Alat hanya dapat bergerak maju mundur untuk menyeimbangkan diri di bidang datar tanpa mengikuti garis atau jalur tertentu.
7. Sistem aturan logika *Fuzzy* diperuntukan untuk menetukan nilai Kp, Ki dan Kd. Untuk proses selanjutnya diambil dari perhitungan PID untuk pergerakan.
8. Parameter masukan yang digunakan berasal dari sensor IMU MPU-6050.
9. Penelitian hanya difokuskan pada penerapan algoritma *Fuzzy*-PID dan implementasi pada program.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah mengimplementasikan, merancang, dan membuat suatu algoritma tuning nilai Kp, Ki, dan Kd yang akan digunakan untuk perhitungan PID yang digunakan sebagai kendali robot beroda dua. Dan agar dalam perkembangannya memudahkan dalam proses tuning nilai gain kontroler.

1.5 Manfaat Penelitian

Adapun manfaat dari hasil penelitian ini adalah sebagai berikut :

1. Manfaat utama dari penelitian yaitu mampu merancang sebuah *balancing robot* beroda dua.
2. Karena penelitian ini menggunakan robot prototipe, diharapkan untuk penelitian selanjutnya bisa di aplikasikan pada *Segway*.
3. Hasil penelitian dapat dikembangkan kembali untuk aplikasi pada bidang kontrol, robotika, dan transportasi.

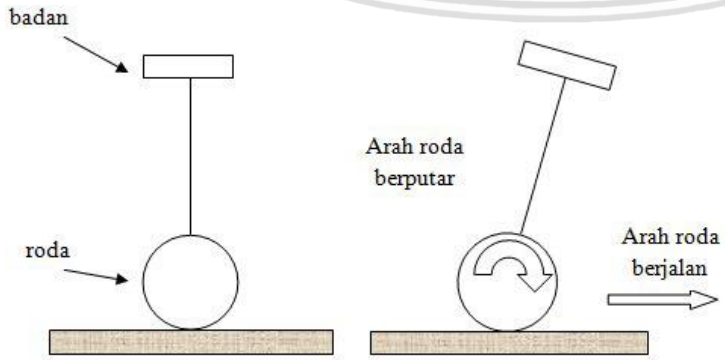
BAB II

TINJAUAN PUSTAKA

2.1 Robot Beroda Dua (*Balancing Robot*)

Robot adalah benda mirip manusia yang memiliki kemampuan untuk mengengerjakan atau membantu suatu pekerjaan dan bertingkah laku seperti manusia. Salah satu cara menambah tingkat kecerdasan sebuah robot adalah dengan menambah sensor, metode kontrol bahkan memberikan kecerdasan buatan pada robot tersebut. Salah satunya adalah robot beroda dua. Robot beroda dua merupakan suatu robot mobile yang memiliki sebuah roda disisi kanan dan kirinya yang tidak akan seimbang apabila tanpa adanya kontroler. Menyeimbangkan robot beroda dua memerlukan suatu metode kontrol yang baik dan handal untuk mempertahankan posisi robot dalam keadaan tegak lurus terhadap permukaan bumi tanpa memerlukan pengendali lain dari luar.

Balancing robot beroda dua merupakan suatu *robot mobile* yang memiliki dua roda di kedua sisinya dan tidak akan seimbang tanpa sebuah metode kontrol yang baik. Cara kerja secara general untuk dapat menyeimbangkan robot ini adalah ketika robot terjatuh kesisi depan, maka untuk menyeimbangkan harus harus mengurimkan sinyal kontrol yang sesuai ke motor DC sehingga motor DC berputar ke arah depan dan mampu kembali ke posisis semula. Atau pada saat *balancing robot* beroda dua condong ke depan atau miring ke kanan pada Gambar 2. 1, maka yang perlu dilakukan adalah motor akan memutar searah jarum jam sehingga *balancing robot* beroda dua akan berputar ke arah depan. Gaya yang digunakan untuk menyeimbangkan dihasilkan dari putaran roda.



Gambar 2. 1 *Balancing robot* Beroda Dua Menyeimbangkan Diri

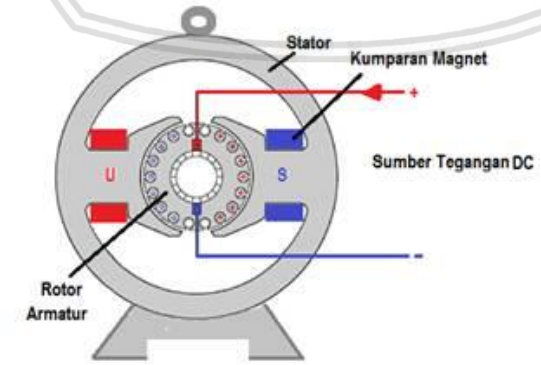
2.2 Motor DC

Sebuah motor listrik mengubah energi listrik menjadi energi mekanik. Kebanyakan motor listrik beroperasi melalui interaksi medan magnet dan konduktor pembawa arus untuk menghasilkan kekuatan, meskipun motor elektrostatik menggunakan gaya elektrostatik. Proses sebaliknya, menghasilkan energi listrik dari energi mekanik, yang dilakukan oleh generator seperti alternator, atau dinamo. Banyak jenis motor listrik dapat dijalankan sebagai generator, dan sebaliknya. Motor listrik dan generator yang sering disebut sebagai mesin-mesin listrik. Motor DC adalah suatu perangkat yang mengubah energi listrik menjadi energi kinetik atau gerakan (*motion*). Motor DC ini juga dapat disebut sebagai Motor Arus Searah. Seperti namanya, DC Motor memiliki dua terminal dan memerlukan tegangan arus searah atau DC (*Direct Current*) untuk dapat menggerakannya. Motor Listrik DC ini biasanya digunakan pada perangkat-perangkat Elektronik dan listrik yang menggunakan sumber listrik DC.

Prinsip kerja motor DC yaitu segulung kawat yang dialiri arus listrik dan ditempatkan di dalam suatu medan magnet akan mengalami gaya yang sebanding dengan arus dan kekuatan medan magnetnya. Gaya yang ditimbulkan itu disebut juga dengan Gaya Lorentz, yang dapat dirumuskan sebagai berikut:

$$F = B.I.L \ldots (N) \qquad (2.1)$$

Dalam hal ini B adalah kerapatan fluks magnet (Wb/m^2), I adalah arus yang mengalir (A) dan L adalah panjang kawat (m). Gambar penampang dari motor DC ditunjukan pada Gambar 2.2.

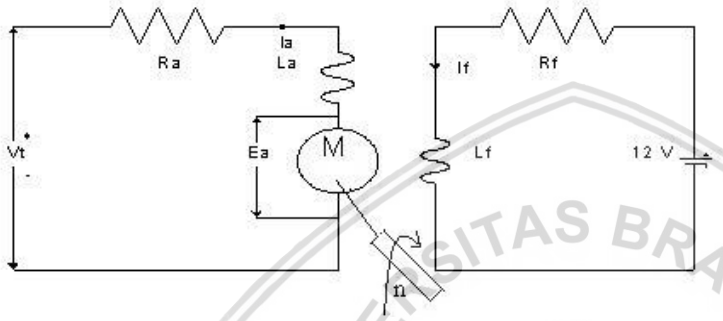


Gambar 2.2 Penampang Melintang Motor DC

Pada persamaan 2.2 merupakan prinsip dasar sebuah motor, dimana terjadinya proses perubahan energi listrik (*I*) menjadi energi mekanik (*F*). Jika motor mempunyai jari-jari sebesar (*r*), maka akan menimbulkan torsi sebesar:

$$\tau = F.r = B.I.L.r \ ...(Nm) \qquad (2.2)$$

Saat dibangkitkan, konduktor akan bergerak di dalam medan magnet dan akan menimbulkan ggl (gaya gerak listrik) yang merupakan reaksi lawan terhadap tegangan penyebabnya. Proses pada Gambar 2.3 menunjukan proses kerja motor DC.



Gambar 2.3 Proses kerja motor DC

Motor dapat berputar jika tegangan masukan motor lebih besar dari ggl yang timbul. Hubungan antara tegangan sumber dan ggl lawan seperti ditunjukan dalam Gambar 2.3 dapat dirumuskan seperti berikut:

$$Ea = Vin - Ia.Ra \ ...(V) \qquad (2.3)$$

Dalam hal ini Ea yaitu tegangan pada jangkar, Vin adalah tegangan masukan, Ia adalah arus jangkar, dan Ra adalah tahanan jangkar, sedangkan induksi yang timbul adalah:

$$Ea = C.n.\Phi \ ...(V) \qquad (2.4)$$

Dengan konstanta yaitu (C), n adalah kecepatan motor, dan Φ adalah fluks magnetik yang besarnya sebanding dengan arus penguatan torsi. Torsi pada motor juga sebanding dengan fluks magnetik dan arus. Hal ini ditunjukan dalam persamaan berikut:

$$\tau = C.\Phi.Ia \ ...(Nm) \qquad (2.5)$$

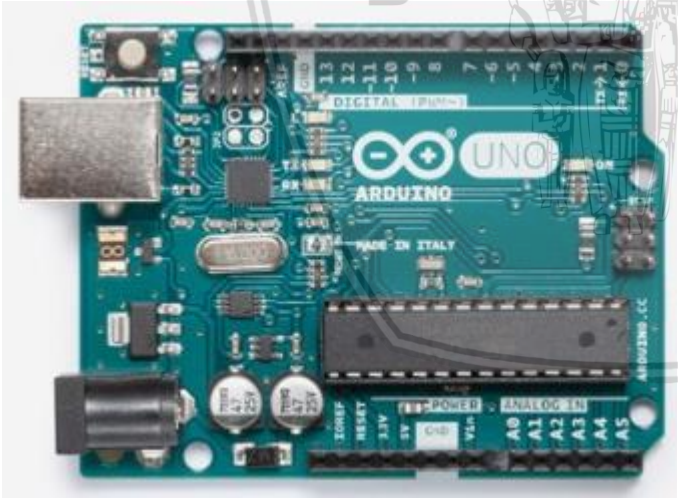
Jika diketahui kecepatan sudut (w) yaitu:

$$w = 2\pi.n/60 \ ...(rad/s) \qquad (2.6)$$

2.3 Arduino Uno

Arduino Uno adalah board mikrokontroler berbasis ATmega328. Memiliki 14 pin input dari output digital dimana 6 pin *input* tersebut dapat digunakan sebagai *output* PWM dan 6 pin input analog, 16 MHz osilator kristal, koneksi USB, *jack power*, ICSP *header*, dan tombol reset. Untuk mendukung mikrokontroler agar dapat digunakan, cukup hanya menghubungkan Board Arduino Uno ke komputer dengan menggunakan kabel USB atau listrik dengan AC yang-ke adaptor-DC atau baterai untuk menjalankannya.

Uno berbeda dengan semua board sebelumnya dalam hal koneksi USB-to-serial yaitu menggunakan fitur Atmega8U2 yang diprogram sebagai konverter USB-to-serial berbeda dengan board sebelumnya yang menggunakan chip FTDI driver USB-to-serial. Nama "Uno" berarti satu dalam bahasa Italia, untuk menandai peluncuran Arduino 1.0. Uno dan versi 1.0 akan menjadi versi referensi dari Arduino. Uno adalah yang terbaru dalam serangkaian board USB Arduino, dan sebagai model referensi untuk platform Arduino, untuk perbandingan dengan versi sebelumnya, lihat indeks board Arduino. Bentuk fisik dari Arduino Uno ditunjukan oleh Gambar 2.4 dan spesifikasi dari Arduino Uno ditunjukan pada Tabel 2.1.



Gambar 2.4 Arduino Uno tampak atas
Sumber : www.arduino.cc (2018).

Tabel 2. 1 Spesifikasi Arduino UNO

Spesifikasi	Nilai	Satuan
Tegangan Operasi	5	V
Tegangan Masukan	7 ~ 12	V
Arus DC setiap I/O Pin	20	mA
Arus DC untuk 3,3V Pin	50	mA
Memori Flash	32 (ATmega328P)	kB
Pin I/O PWM Digital	6	-
Pin masukan Analog	6	-
Pin digital I/O	14 (6 keluaran PWM)	-
Clock Speed	16	MHz
Panjang	68.6	mm
Lebar	53.4	mm
Berat	25	g
LED BUILTIN	13	-
EEPROM	4	kB
SRAM	8	kB

Sumber: ATMEL (2014, p.1).

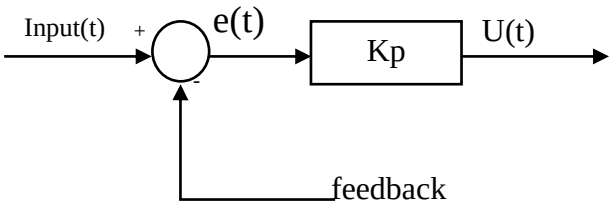
2.4 Kontroler

Kontroler dalam sebuah sistem kontrol mempunyai kontribusi yang besar terhadap perilaku sistem. Hal itu disebabkan oleh tidak dapat dirubahnya komponen penyusun sistem tersebut. Artinya, karakteristik *plant* harus diterima sebagaimana adanya, perubahan perilaku sistem hanya dapat dilakukan melalui penambahan kontroler.

Salah satu fungsi kontroler adalah mengurangi sinyal *error*, sinyal *error* adalah perbedaan nilai *setpoint* dengan niali *output plant*. Adalah nilai aktual yang terukur pada *output plant.* Semakin kecil nilai sinyal error maka kinerja sistem kontrol dinilai semakin baik. Prinsip kerja kontroler adalah membandingkan nilai output dengan nilai *setpoint*, menentukan nilai kesalahan dan akhirnya menghasilkan sinyal kontrol untuk meminimalkan kesalahan (Ogata, 1997).

2.4.1 Kontroler Proporsional

Kontroler proporsional memiliki *output* yang besarnya sebanding dengan besarnya sinyal *error*. *Output* kontroler merupakan perkalian antara penguatan proporsional dengan sinyal *error*. diagram blok kontroler proporsional ditunjukan oleh Gambar 2.5.



Gambar 2. 5 Diagram Blok Kontroler Proporsional

Dimana :

Kp = adalah gain proporsional

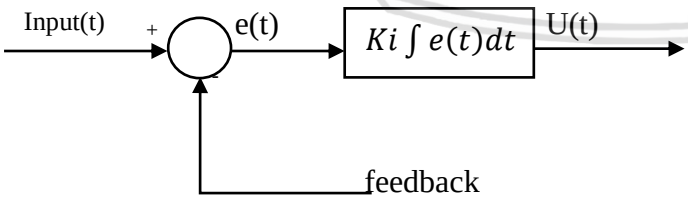
e(t) = sinyal error

u(t) = ouput keluaran

Penambahan Kp akaan mempercepat kecepatan respon *transient* dan mengurangi *error steady state*.

2.4.2 Kontroler Integral

Kontroler Integral memiliki karakteristik seperti sebuah operasi integral, *output* kontroler dipengaruhi oleh perubahan yang sebanding dengan perubahan nilai sinyal *error*. *Output* kontroler merupakan penjumlahan terus menerus dari perubahan sinyal *error*. diagram blok kontroler integral ditunjukan oleh Gambar 2.6.



Gambar 2. 6 Diagram Blok Kontroler Integral

Dimana :

Ki = adalah gain integral

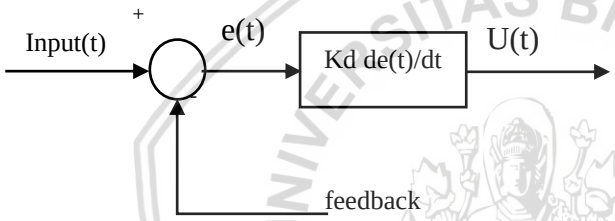
E(t) = sinyal error

U(t) = output kontroler

Aksi kontrol *integral* digunakan untuk menghilangkan sinyal *error* dalam *steady state.* Namun pemilihan Ki yang tidak tepat dapat menyebabkan respon transien yang tinggi, sehingga dapat menyebabkan ketidakstabilan sistem. Pemilihan Ki yang sangat tinggi justru dapat menyebabkan *output* berosilasi karena menambah *orde system*.

2.4.3 Kontroler Derivative

Kontroler *derivative* memiliki sifat seperti suatu operasi turunan. Perubahan yang mendadak pada masukan kontroler mengakibatkan perubahan yang sangat besasr dan cepat, kontroler ini tidak akan menghasilkan *output* saat sinyal *error* konstan sehingga tidak akan mempengaruhi keadaan mantap. diagram blok kontroler *derivative* ditunjukan oleh Gambar 2.7.



Gambar 2. 7 Diagram Blok Kontroler Derivatif

Dimana :

Kd = adalah gain *derivative*

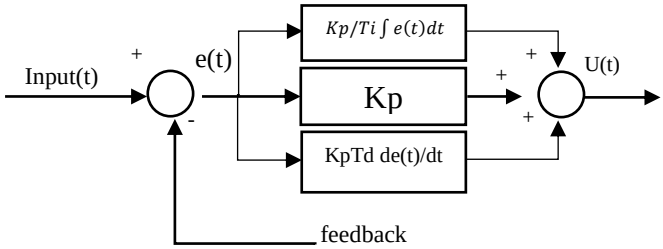
E(t) = sinyal *error*

U(t) = *output* kontroler

Kontroler ini digunakan untuk memperbaiki atau mempercepat respon transient. Kontrol diferensial hanya berubah saat ada perubahan *error* sehingga saat *error* statis kontrol ini tidak bereaksi, hal ini pula yang menyebabkan kontroler diferensial tidak dapat dipakai sendiri.

2.4.4 Kontroler PID

Gabungan aksi kontrol proporsional, integral dan derivative yang terlihat dalam Gambar 2.8 mempunyai keunggulan dapat saling menutupi kekurangan dan kelebihan dari masing-masing kontroler.



Gambar 2. 8 Diagram Blok Kontroler Proporsional integrative dan difrensiative

Dimana :

Kp = adalah penguatan *proportional*

Ti = adalah waktu integral

Td = adalah waktu *derivative*

e(t) = sinyal *error*

u(t) = output kontroler

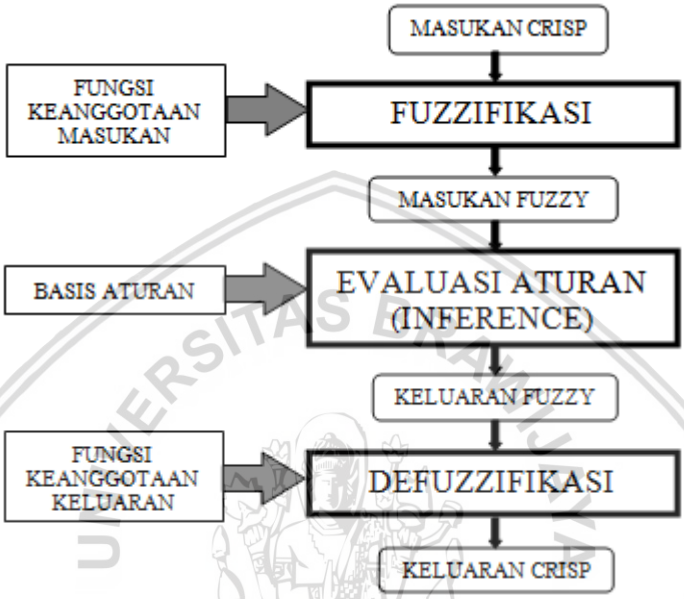
Ti merupakan waktu yang digunakan untuk mengatur aksi kontrol internal sedangkan Td adalah waktu internal dengan laju aksi kontroler proporsional. Sinyal keluaran PID didapat dari persamaan (2.7).

$$u(t) = Kp(e(t) + \frac{1}{Ti}\int_0^t e(t)dt + Td\ \frac{de(t)}{dt} \qquad (2.7)$$

2.5 Logika Fuzzy

Arti dari *Fuzzy* yaitu, samar. Dalam kehidupan sehari-hari kata samar lebih terdengar akrap dari pada nila tegas. Kebalikan dari *Fuzzy* yaitu *crisp* yang berarti tegas. Pada tahun 1965 L.A.Zadeh memodifikasi teori himpunan yang disering disebut himpunan kabur (*Fuzzy Set).* Himpunan *Fuzzy* digunakan untuk memperluas jangkauan fungsi karakteristik sehingga fungsi tersebut mencakup bilangan real pada interval. Nilai keanggotaannya menunjukan bahwa nilai dalam suatu semesta pembicaraan tidak hanya berada pada 0 atau 1, namun juga terdapat nilai yang terletak diantaranya dengan kata lain suatu nilai kebenaran tidak hanya bernilai benar atau salah. Nilai 1 menunjukan benar dan nilai 0 menunjukan salah serta masih terdapat nilai yang terlatak diantaranya. Pada tahun 1982 pengontrolan dengan mengunakan logika *Fuzzy* mengalami perkembangan yang sangat pesat. Terutama dalam hal yang berhubungan dengan penyelesaian masalah kendali yang bersifat linier, sulit dimodelkan, dan karakteristiknya berubah terhadap waktu (*time varying*) serta kompleks. Jadi Logika *Fuzzy* adalah suatu cara yang tepat untuk memetakan

suatu ruang *input* ke dalam suatu ruang output, mempunyai nilai kontinyu. *Fuzzy* dinyatakan dalam derajat dari suatu keanggotaan dan derajat dari kebenaran. Oleh sebab itu sesuatu dapat dikatakan sebagian benar dan sebagian salah pada waktu yang sama . Yang harus ada dalam logika *Fuzzy* adalah fuzzifikasi, evaluasi aturan (*inference*) berdasarkan rule base, dan defuzzifikasi. Jika digambarkan strukturnya dapat dilihat pada Gambar 2.9.



Gambar 2.9 Struktur Dasar Pengendali *Fuzzy*

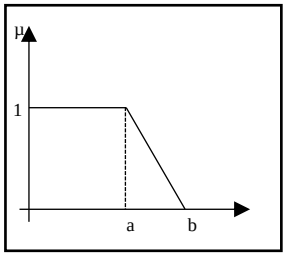
2.5.1 Logika Fuzzy

Suatu himpunan *Fuzzy* A dalam semesta pembicaraan U dinyatakan dengan fungsi keanggotaan, uA yang harganya berada dalam interval [0,1] (Kuswadi,2000:27). Secara matematika hal ini dinyatakan pada persamaan (2.8).

$$\mu A : U \rightarrow [0,1] \qquad (2.8)$$

adapun beberapa pendekatan dalam perancangan aturan *Fuzzy* dirancang untuk kondisi yang terdiri dari *reversegrade, triangle, grade*

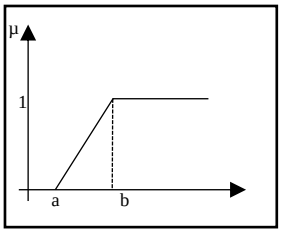
a) Fungsi keanggotaan *reverse grade* ditunjukan dalam Gambar 2.10 dan persamaan matematis ditunjukan pada persamaan 2.9.



$$\mu[x] = \begin{cases} 0, x \geq b \\ \frac{x-a}{b-a}, a < x < b \\ 1, \ x \leq a \end{cases} \qquad (2.9)$$

Gambar 2. 10 Grafik fungsi keanggotaan *reverse grade*

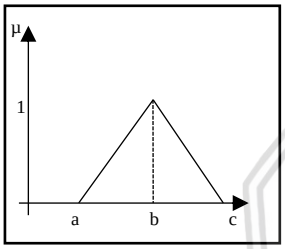
b) Fungsi keanggotaan *grade* ditunjukan dalam Gambar 2.11 dan persamaan matematis ditunjukan pada persamaan 2.10.



$$\mu[x] = \begin{cases} 0, x \leq b \\ \frac{-x+a}{b-a}, a < x < b \\ 1, \ x \geq b \end{cases} \quad (2.10)$$

Gambar 2. 11 Grafik fungsi keanggotaan *grade*

c) Fungsi keanggotaan *triangle* ditunjukan pada Gambar 2.12 dan persamaan matematis ditunjukan pada persamaan 2.11.



$$\mu[x] = \begin{cases} 0, x \geq c \ atau \ x \leq a \\ \frac{x-a}{b-a}, a < x < b \\ \frac{-x+c}{c-b}, b < x < c \\ 1, \ x = b \end{cases} \quad (2.11)$$

Gambar 2. 12 Grafik fungsi keanggotaan *triangle*

2.5.2 Fuzzyfikasi

Fuzzyfikasi merupakan proses dimana mengolah masukan berupa nilai crisp menjadi himpunan *Fuzzy*, diperlukan metode proses ini dikarenakan apa yang kita yakini sebagai bentuk crisp dan deterministik sebenarnya tidak deterministrik sama sekali.

Sebagai contoh, jika ada proses atau pengukuran kebisingan, kita mungkin ingin untuk menjelaskan hal ini dengan menciptakan *fuzzy set* untuk takaran terukur dari pada dengan asumsi mereka akurat yang diukur.

Ada beberapa strategi yang dapat dilakukan dalam fuzzyfikasi diantaranya

1. *Fuzzy Singletone*
2. *Fuzzy Number*
3. *Hybrid Fuzzy*/Bilangan Acak

Pedoman memilih fungsi keanggotaan untuk proses fuzzyfikasi dapat menggunakan cara sebagai berikut :

1. Himpunan *Fuzzy* dengan distribusi simetris
2. Gunakan himpunan *Fuzzy* dengan jumlah ganjil, berkaitan erat dengan jumlah kaidah (*rules*).
3. Mengatur himpunan *Fuzzy* agar saling menumpuk.

4. Menggunakan fungsi keanggotaan bentuk segitiga, atau trapesium.

2.5.3 Rule Base Fuzzy

Fuzzy Rule adalah bagian yang menggambarkan dinamika suatu sistem terhadap masukan yang dikarakteristikan oleh sekumpulan variabel – variabel linguistik dan berbasis pengetahuan seorang operator ahli. Penyataan tersebut umumnya dinyatakan oleh suatu pernyataan bersyarat.

Dalam pengendali berbasis *Fuzzy*, aturan pengendalian *Fuzzy* berbentuk aturan "IF_THEN". Untuk sebuah sistem *Multi Input Single Output (MISO)* basis aturan pengendalian *fuzzy* berbentuk seperti berikut ini,

Rule 1 IF X is A1 AND Y is B1 THEN Z is C1

Rule 2 IF X is A2 AND Y is B2 THEN Z is C2

.

Rule IF X is An AND Y is Bn THEN Z is Cn

Dengan X,Y,Z merupakan variabel linguistik, dimana X dan Y merupakan variabel masukan, dan Z merupakan variabel keluaran sistem. An, Bn, dan Cn merupakan nilai linguistik dari X,Y,Z (Lee, 1990).

2.5.4 Defuzzyfikasi

Defuzzifikasi adalah proses kebalikan fuzzyfikasi. Hal ini diperlukan untuk mengolah hasil berupa nilai himpunan *Fuzzy* untuk diubah kembali kedalam bentuk *crisp.* Defuzzyfikasi adalah proses untuk mendapatkan nilai numerik dari data *Fuzzy* yang dihasilkan dari proses inferensi. Proses defuzzifikasi dinyatakan pada persamaan 2.12.

$$yo = defuzzifier\ (y) \qquad (2.12)$$

Dengan :

Y : aksi kontrol

Y0 : aksi kontrol crisp

Defuzzyfier : operator defuzzifikasi

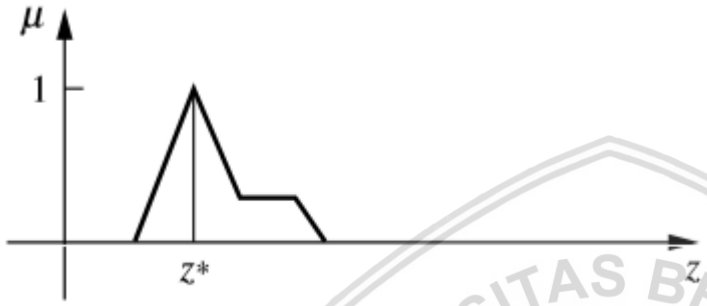
Ada beberapa cara yang bisa dilakukan dalam proses defuzzyfikasi antara lain

2.5.5 Prinsip Min-Max Membership

Max membership bisa dikatan sebagai metode ketinggian. Dalam metode ini terbatas pada keluaran yang memuncak. Max membership dapat dinyatakan dalam persamaan 2.13.

$$U_c{\sim}(z*) \geq \mu C \sim (z)\ for\ all\ z\ \in Z \tag{2.13}$$

Dengan z* adalah nilai defuzzyfikasi yang dapat dilihat pada Gambar 2.13 dibawah ini.



Gambar 2. 13 Max Membership defuzzyfikasi

2.5.6 Metode Centroid

Metode ini juga disebut *centre of gravity*. Metode ini yang paling umum dan menarik untuk digunakan. (Sugeno,1985 : Lee.1990);
Diberikan ekspresi aljabar pada persamaan 2.14.

$$z = \frac{\int \mu c\ (z).\ z\ dz}{\int \mu c\ (z)\ dz} \tag{2.14}$$

Metode ini sangat sering digunakan karena cara pengunaanya sangat efisien. Metode ini digunakan untuk output membership yang simetris. Persamaan matematisnya ditunjukan pada persamaan 2.15.

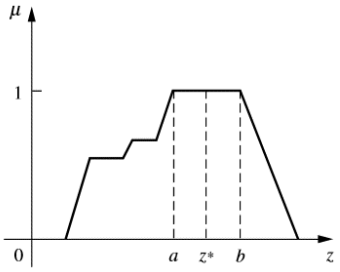
$$z = \frac{\sum \mu c\ (z).\ z}{\sum \mu c\ (z)} \tag{2.15}$$

2.5.7 Mean Max Membership

Dikenal juga sebagai middle of maxima, metode hampir serupa dengan Max Membership, kecuali dari metode Max membership bisa juga no-unik. Diberikan persamaan 2.16.

$$z = \frac{a+b}{2} \tag{2.16}$$

Dimana nilai a dan b diberikan pada Gambar 2.14.



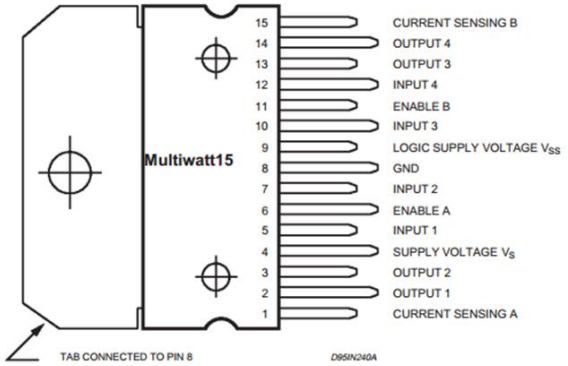
Gambar 2.14 Membership metode defuzzyfikasi

2.6 Driver L298N

L298N adalah IC *driver* yang mengatur arah dan kecepatan motor DC. L298 dapat digunakan pada tegangan tinggi dan memiliki keluaran arus sampai dengan 2 A setiap *channel*-nya dengan ketahanan terhadap suhu yg cukup tinggi. Tabel 2.2 menunjukkan *absolute maximum rating* dari L298N. Gambar 2.15 menunjukkan *pin* konfigurasi dari L298.

Tabel 2.2 Absolute Maximum Rating Pin IC L298N

Symbol	**Parameter**	**Value**	**Unit**
V_S	Power supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I, V_{en}	Input and Enable Voltage	-0.3 to 7	V
I_O	Peak Output Current (Each Channel)		
	-Non Repetitive	3	A
	-Repetitive (80% on -20% off; t_{on} = 10ms)	2.5	A
	-DC Operation	2	A
V_{sens}	Sensing Voltage	-1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75˚C)	25	W
T_{op}	Junction Operation Temperature	-25 to 130	C
T_{stg}, T_j	Storage and Junction Temperature	-40 to 150	C



Gambar 2.15 Konfigurasi Pin IC L298N

Sumber: ST, 2000: 1

2.7 Sensor IMU MPU6050

IMU (*Inertial Measurement Unit*) adalah perangkat elektronika yang mampu mengukur dan melaporkan kecepatan, orientasi, dan gaya gravitasi menggunakan kombinasi dari *accelerometer* dan *gyroscope*. Adapun sensor-sensor yang akan digunakan untuk IMU *Digital Combo Board* ini adalah MPU6050 untuk 3-axis *accelerometer* dan GY521 untuk 3-axis gyroscope. Sensor IMU *digital combo board* ini berkomunikasi melalui I2C (*Inter Integrated Circuit*). I2C adalah standar komunikasi serial dua arah menggunakan dua saluran yang didisain khusus untuk mengirim maupun menerima data. Sistem I2C terdiri dari saluran SCL (*Serial Clock*) dan SDA (*Serial Data*) yang membawa informasi data antara I2C dengan pengontrolnya. Sensor ini dapat bekerja pada tegangan 2,1 Volt sampai 3,6 Volt, tetapi dianjurkan untuk dicatu dengan tegangan 3,3Volt. Bentuk fisik Sensor IMU *Digital Combo Board* dapat dilihat pada Gambar 2.16.

Gambar 2.16 Bentuk fisik Sensor IMU MPU6050
Sumber: Datasheet Sensor IMU MPU6050

Tabel 2.3 Spesifikasi IMU MPU6050

VDD	2.375V-3.46V
VLOGIC	1.8V±5%
Ta	25°C
Full-Scale Range FS_SEL=0	±250º/s
Full-Scale Range FS_SEL=1	±500º/s
Full-Scale Range FS_SEL=2	±1000º/s
Full-Scale Range FS_SEL=3	±2000º/s
Acceleration range	± 2 ± 4 ± 8 ± 16 g
Gyroscope mechanical frequencies X-Axis	30-36 kHz
Gyroscope mechanical frequencies Y-Axis	27-33 kHz
Gyroscope mechanical frequencies Z-Axis	24-30 kHz
Dimensi modul	20.3mm x 15.6mm
Jarak antar pin header	2.54 mm
Chip built-in 16 bit AD converter	16 bits data output
Communication standard	I2C

Sumber: Datasheet Sensor IMU MPU6050

Accelerometer adalah alat yang digunakan untuk mengukur percepatan, mendeteksi dan mengukur getaran (vibrasi), dan mengukur percepatan akibat gravitasi. Sensor accelerometer mengukur percepatan akibat gerakan benda yang melekat padanya. Pendeteksian gerakan berdasarkan pada 3 sumbu yaitu kanan-kiri, atas-bawah dan depan-belakang. Sensor *Gyroscope* adalah sensor yang dapat membaca kecepatan sudut yang dinamis. Prinsip kerja dari *gyroscope* ini adalah pada saat *gyroscope* berotasi maka *gyroscope* akan memiliki nilai keluaran.

Dengan menggunakan kombinasi *accelerometer* dan *gyroscope* pada suatu sistem maka *accelerometer* dapat memberikan pengukuran sudut saat sistem berada pada kondisi diam. Sedangkan pada saat sistem berotasi *accelerometer* tidak bisa bekerja secara maksimal karena memiliki respon yang lambat. Kelemahan inilah yang dapat diatasi oleh *gyroscope* karena *gyroscope* dapat membaca kecepatan sudut yang dinamis. Namun *gyroscope* juga memiliki kelemahan yaitu proses perpindahan kecepatan sudut dalam

jangka waktu yang panjang menjadi tidak akurat karena ada efek bias yang dihasilkan oleh *gyroscope*.

BAB III

METODE PENELITIAN

Penyusunan skripsi ini dilakukan berdasarkan masalah yang bersifat aplikatif, yaitu perencanaan dan perealisasian alat agar dapat bekerja sesuai dengan yang direncanakan yang mengacu pada rumusan masalah. Langkah-langkah yang perlu dilakukan untuk merealisasikan alat yang dirancang adalah penentuan spesifikasi alat, studi literatur, perancangan dan pembuatan alat, pengujian alat, dan pengambilan kesimpulan.

3.1 Spesifikasi Alat

Sebelum Sebelum melakukan perencanaan dan pembuatan alat, ditentukan spesifikasi alat yang akan dibangun terlebih dahulu. Hal ini bertujuan sebagai acuan untuk mempermudah dalam membangun alat agar alat yang dibangun sesuai dengan yang diinginkan. Adapun spesifikasi alat yang akan dibangun dan direalisasikan adalah sebagai berikut:

1. Robot berbahan dasar mika (*acrylic*).
2. Balancing robot beroda dua menggunakan sistem penggerak roda yang terletak pada sisi kanan dan kiri badan robot, serta digerakan mengunakan dua buah motor DC.
3. Tempat yang digunakan dalam melakukan pengujian memiliki permukaan datar (kemiringan 0º terhadap bidang datar).
4. Sistem keseimbangan robot beroda dua menggunakan modul Sensor IMU MPU6050.
5. Sistem keseimbangan *balancing* robot beroda dua menggunakan modul Arduino Uno sebagai board utama yang berfungsi untuk memproses input dari sensor menuju akatuator.
6. Menggunakan catu daya rangkaian elektronik sebesar 5 V DC dan catu daya motor sebesar 12 V DC.

3.2 Perancangan Dan Perealisasian Alat

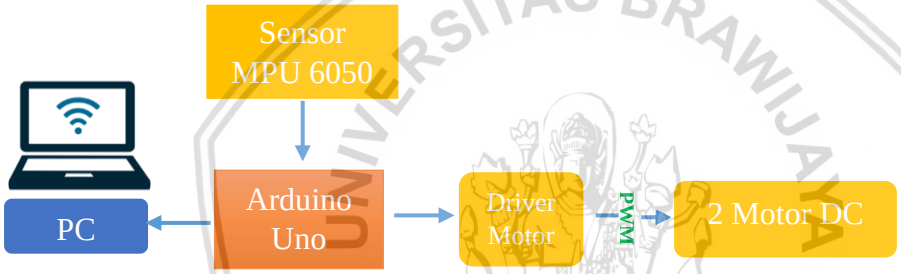
Perancangan dan pembuatan alat dalam skripsi ini dibagai menjadi dua bagian, yaitu pembuatan *hardware* dan *software*.

3.2.1 Perancangan dan Pembuatan Perangkat Keras (*Hardware*)

Secara garis besar perancangan dan pembuatan perangkat keras dapat dibagai menjadi beberapa bagian yaitu perancangan diagram blok sistem keseluruhan dan perancangan mekanika robot.

3.2.1.1 Perancangan Diagram Blok Sistem Keseluruhan

Diagram blok keseluruhan sistem yang dirancang ditunjukkan dalam Gambar 3.1.

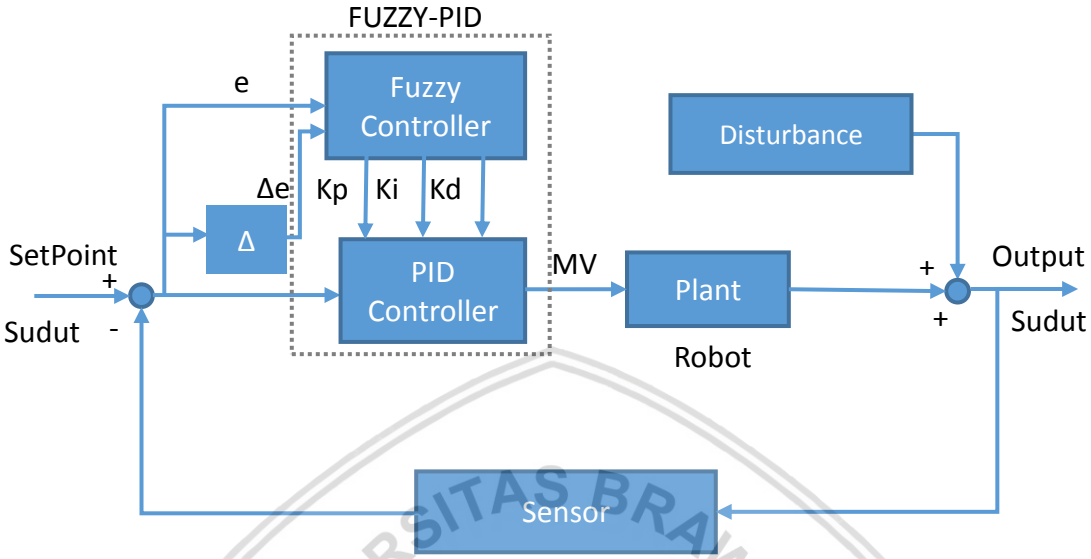


Gambar 3. 1 Diagram blok sistem keseluruhan

Fungsi masing-masing bagian dalam diagram blok ini adalah sebagai berikut :

1. Arduino Uno digunakan sebagai mikrokontroler utama dari sistem untuk mengolah hasil pengolahan keseluruhan sistem
2. Sensor MPU 6050 berfungsi untuk mendeteksi kemiringan sudut pada robot.
3. Modul pengendali atau *driver* motor DC digunakan sebagai antarmuka mikrokontroller dengan aktuator.
4. 2 Motor DC sebagai aktuator penggerak sistem.
5. Komputer digunakan untuk pemantauan proses pembacaan yang dilakukan oleh robot.

Diagram blok kendali sistem yang dirancang ditunjukkan dalam Gambar 3.2.



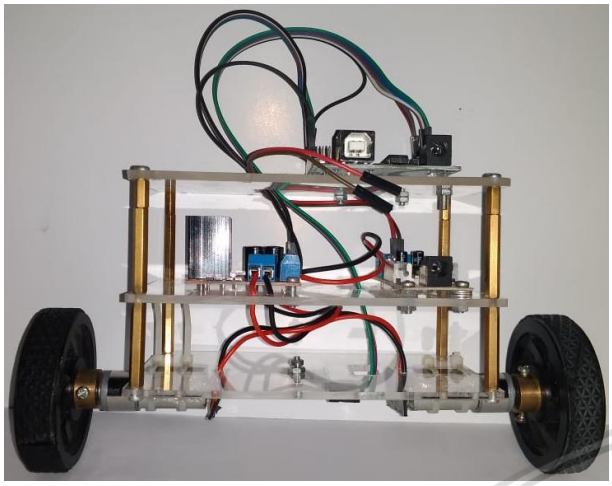
Gambar 3. 2 Diagram blok kendali sistem

Fungsi masing-masing bagian dalam diagram blok ini adalah sebagai berikut :

1. *Setpoint* berupa nilai sudut.
2. Masukan Kontroler *Fuzzy* berupa nilai *error* dan *derror*, dan masukan PID berupa nilai *error*.
3. Keluaran *Fuzzy* berupa gain kontrol Kp, Ki, dan Kd, Keluaran PID adalah variabel pemanipulasi sudut.
4. Gangguan berupa perubahan sudut saat robot bergerak menyeimbangkan diri.
5. Plant berupa robot atau kombinasi aktuator robot yaitu dua Motor DC yang pada pergerakannya untuk menyeimbangkan badan *balancing robot*.
6. Sensor berupa sensor sudut MPU 6050.

3.2.1.2 Pembuatan Mekanika Robot

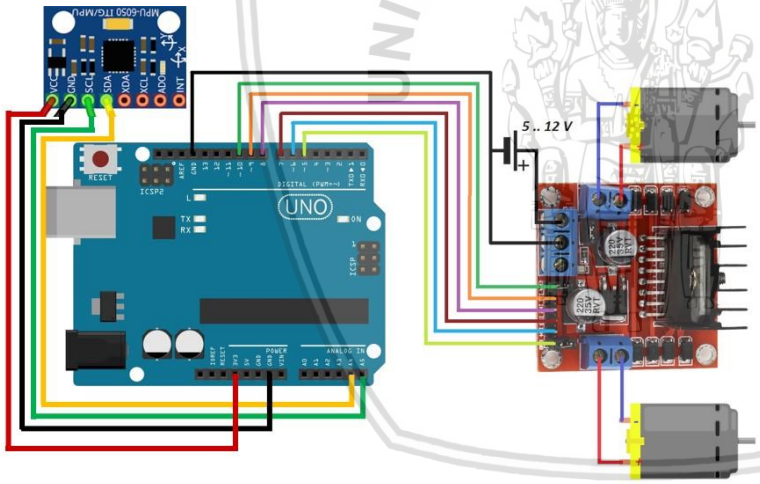
Setelah dirancang baik hardware maupun software, maka didapat hasil robot dengan panjang = 22 cm, lebar = 8 cm, dan tinggi = 13 cm seperti Gambar mekanik robot beroda dua ditunjukkan dalam Gambar 3.3.



Gambar 3. 3 Perspekstif robot secara keseluruhan

3.3 Perancangan Rangkaian Antarmuka Mikrokontroler Utama

Rangkaian utama yang digunakan dalam perancangan menggunakan Arduino Uno sebagai pusat pengolah utama dalam proses *balancing robot* ditunjukan dalam Gambar 3.4.



Gambar 3.4 Rangkaian Mikrokontroler Utama Arduino Uno

Pada perancangan perangkat keras robot ini menggunakan modul mikrokontroller Arduino Uno sebagai pengolah utama untuk pemrosesan algoritma dan data sensor. Pada perancangan ini pin-pin yang digunakan adalah :

PIN A.5 = dihubungkan dengan keluaran SCL rangkaian mikrokontroler *slave* pengolah rangkaian sensor sudut

PIN A.4 = dihubungkan dengan keluaran SDA rangkaian mikrokontroler *slave* pengolah rangkaian sensor sudut

PIN D.9 = dihubungkan dengan masukan arah motor kiri/ pin INPUT1 rangkaian *driver* L298N

PIN D.8 = dihubungkan dengan masukan arah motor kiri/ pin INPUT2 rangkaian *driver* L298N

PIN D.7 = dihubungkan dengan masukan arah motor kanan/ pin INPUT3 rangkaian *driver* L298N

PIN D.6 = dihubungkan dengan masukan arah motor kanan/ pin INPUT4 rangkaian *driver* L298N

PIN D.5 = dihubungkan dengan masukan PWM motor kanan/ pin ENABLE_B rangkaian *driver* L298N

PIN D.10 = dihubungkan dengan masukan PWM motor kiri/ pin ENABLE_A rangkaian *driver* L298N

3.4 Perancangan Kontroler PID

Kontroler PID dibentuk dari gabungan kontroler *proporsional integral dan difrensial*, yang mana persamaan kontroler ini secara umum dapat ditunjukan pada persamaan 3.1

$$u(t) = Kp(e(t) + \frac{1}{Ti}\int_0^t e(t)dt + Td\ \frac{de(t)}{dt}\ ) \quad (3.1)$$

Dimana u(t) adalah signal kontrol dari PID, Kp adalah gain proporsional, Ti adalah waktu integral, dan Td waktu difrensial, dimana untuk e(t) didefinisikan sebagai selisih nilai *setpoint* dan pembacaan sensor sudut yang digunakan. Integral error (sigma error) didefinisikan sebagai jumlahan *error* sekarang dan sebelumnya sedangkan difrensial *error* (delta *error*) didefinisikan sebagai selisih *error* sekarang dan sebelumnya. Jika bentuk persamaan umum tersebut diubah dalam bentuk *laplace* akan terbentuk persamaan 3.2

$$u(s) = Kp(1 + \frac{1}{Tis} + Tds)E(s) \quad (3.2)$$

$Ki = Kp/Ti$; $Kd = Kp * Td$; α = Ti/Td;

$Kp = Kp'(Kpmax - Kpmin) + Kpmin$; $Kd = Kd'(Kdmax - Kdmin) + Kdmin$;

Persamaan tersebut belum dapat dimasukan ke mikrokontroler maka dari itu persamaan kontinyu 3.2 diubah ke bentuk diskrit dengan Transformasi Z. Yang mana untuk transformasi ini dibutuhkan waktu sapmling (Ts) dengan *Bilinear Transform* nilai notasi s setara dengan persamaan 3.3

$$s = \frac{2}{Ts}(\frac{1-z^{-1}}{1+z^{-1}}) \quad (3.3)$$

Maka persamaan 3.2 setara dengan persamaan 3.4

$$u(z) = (Kp + \frac{KpTs(1+z^{-1})}{2Ti(1-z^{-1})} + KpTd\frac{2}{Ts}(\frac{1-z^{-1}}{1+z^{-1}})\ )E(z) \quad (3.4)$$

Berikutnya dibutuhkan modifikasi persamaan agar didapat bentuk yang lebih sederhana kedua ruas dikali dengan pembagi masing masing parameter. Seperti terlihat pada persamaan 3.4

$$u(z)(1-z^{-2}) = (Kp(1-z^{-2}) + \frac{KpTs(1+2z^{-1}+z^{-2})}{2Ti} + KpTd\frac{2}{Ts}(1-2z^{-1}+z^{-2}))E(z) \quad (3.5)$$

Kemudian disusun kembali untuk mendapatkan persamaan kontroler seperti terlihat pada persamaan 3.6.

$$u(z) = u(z)(z^{-2}) + \left(Kp(E(z) - E(z)(z^{-2})\right) + \frac{KpTs(E(z)+2E(z)(z^{-1})+E(z)(z^{-2}))}{2Ti} + \\ \ldots\ldots\ldots\ldots\ldots\ldots KpTd\frac{2}{Ts}(E(z) - 2E(z)(z^{-1}) + E(z)(z^{-2})) \quad (3.6)$$

Kemudian persamaan tersebut diubah ke persamaan beda sehingga di dapat persamaan 3.7.

$$u(z) = u(k-2) + (Kp(e(k) - e(k-2)) + \frac{KpTs(E(k)+2E(k-1)+E(k-2))}{2Ti} + \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots KpTd\frac{2}{Ts}(E(k) - 2E(k-1) + E(k-2)) \quad (3.7)$$

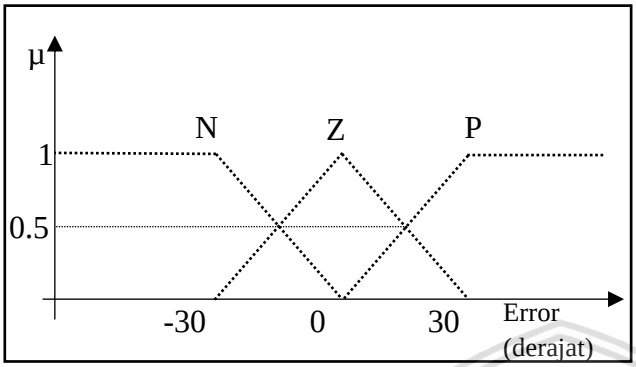
$$u(z) = Kp\Delta e + KieTs + Kd\Delta\left(\frac{\Delta e}{Ts}\right)$$

Dimana (k-n) adalah kondisi sebelumnya, yang kemudian bentuk ini dapat diterapkan pada pemerograman mikrokontroler.

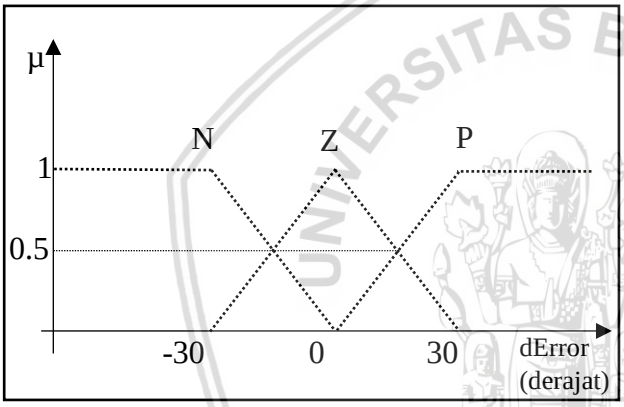
3.5 Perancangan Aturan Fuzzy

Perancangan aturan fuzzyfikasi dengan metode *Fuzzy Mamdani* merupakan pendekatan untuk membangkitkan sinyal manipulasi yang akan digunakan untuk mendapatkan parameter kontrol, dalam perancangan algoritma *Fuzzy* dirancang dengan masukan berupa *error* dan turunan *error*, dimana error pada perancangan ini merupakan selisih antara nilai *setpoint* dan pembacaan sensor sudut, *error* adalah parameter untuk menentukan kemiringan robot terhadap bidang datar, kemudian turunan *error* digunakan untuk mengetahui seberapa besar perubahan yang terjadi antara *error* sekarang dan *error* sebelumnya, yang mana untuk menentukan nilai parameter Kp, Ki, dan Kd, digunakan pendekatan Mean-Max atau nilai tertinggi antara keduanya, sedangkan defuzzyfikasi menggunakan *weighted average.* Dimana untuk fungsi keanggotaan dari error ditunjukan pada Gambar 3.5 dan turunan error ditunjukan pada Gambar 3.6.

Error adalah selisih antara set point dikurangi hasil keluaran sistem pengendalian dan delta error adalah selisih antara error dikurangi error terakhir.

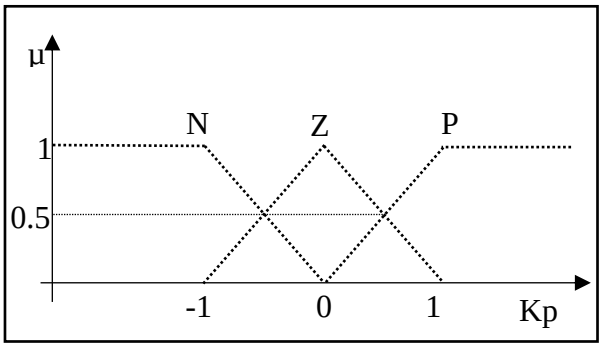


Gambar 3. 5 Fungsi keanggotaan Error



Gambar 3. 6 Fungsi keanggotaan dError

Untuk fungsi keanggotaan dari parameter Kp, Ki dan Kd, dapat diperoleh dengan persamaan PID, dimana pada penentuannya dibutuhkan nilai Ti dan Td, dan Kp, yang fungsi keanggotaannya ditunjukan oleh Gambar 3.7 untuk fungsi keanggotaan Kp, fungsi keanggotaan Ti ditunjukan oleh Gambar 3.8 dan fungsi keanggotaan Td ditunjukan pada Gambar 3.9.

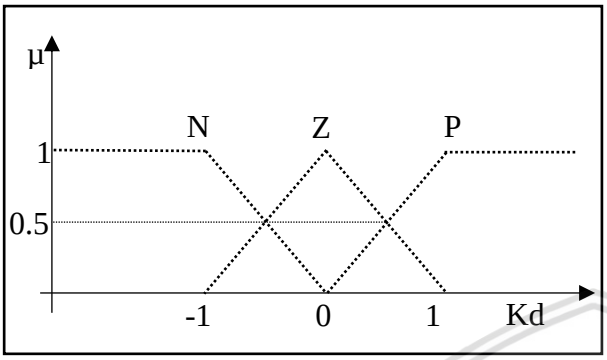


Gambar 3. 7 Fungsi keanggotaan Kp

$$\mu[x] = \begin{cases} 0, x \geq 1 \\ \frac{x-(-1)}{0-(-1)}, -1 < x < 0 \\ 1, \ x \leq -1 \end{cases}$$

$$\mu[x] = \begin{cases} 0, x \geq 1 \ atau \ x \leq -1 \\ \frac{x-(-1)}{0-(-1)}, -1 < x < 0 \\ \frac{-x+1}{1-0}, 0 < x < 1 \\ 1, \ x = 0 \end{cases}$$

$$\mu[x] = \begin{cases} 0, x \leq 0 \\ \frac{-x+0}{1-0}, 0 < x < 1 \\ 1, \ x \geq 1 \end{cases}$$

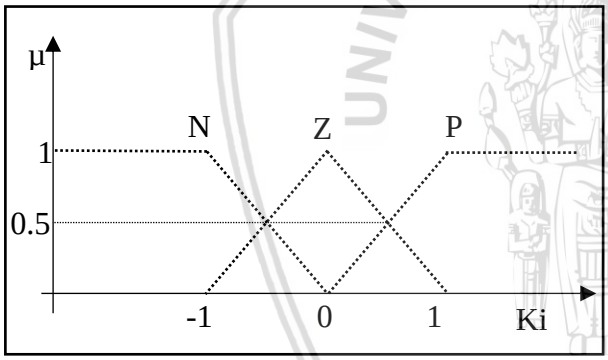


$$\mu[x] = \begin{cases} 0, x \geq 1 \\ \frac{x-(-1)}{0-(-1)}, -1 < x < 0 \\ 1,\ x \leq -1 \end{cases}$$

$$\mu[x] = \begin{cases} 0, x \geq 1\ atau\ x \leq -1 \\ \frac{x-(-1)}{0-(-1)}, -1 < x < 0 \\ \frac{-x+1}{1-0}, 0 < x < 1 \\ 1,\ x = 0 \end{cases}$$

$$\mu[x] = \begin{cases} 0, x \leq 0 \\ \frac{-x+0}{1-0}, 0 < x < 1 \\ 1,\ x \geq 1 \end{cases}$$

Gambar 3. 8 Fungsi keanggotaan Kd



$$\mu[x] = \begin{cases} 0, x \geq 1 \\ \frac{x-(-1)}{0-(-1)}, -1 < x < 0 \\ 1,\ x \leq -1 \end{cases}$$

$$\mu[x] = \begin{cases} 0, x \geq 1\ atau\ x \leq -1 \\ \frac{x-(-1)}{0-(-1)}, -1 < x < 0 \\ \frac{-x+1}{1-0}, 0 < x < 1 \\ 1,\ x = 0 \end{cases}$$

$$\mu[x] = \begin{cases} 0, x \leq 0 \\ \frac{-x+0}{1-0}, 0 < x < 1 \\ 1,\ x \geq 1 \end{cases}$$

Gambar 3. 9 Fungsi keanggotaan Ki

Setelah diketahui fungsi keanggotaan *Fuzzy* dari parameter kontroler PID, agar didapatkan keluaran dari nilai *error* dan turunan *error*, dibutuhkan hubungan yang ditentukan dalam fuzzy rule.

Hubungan *error* dan turunan *error* dijabarkan sebagai berikut :

- Saat e relatif besar memperbesar nilai Kp dan memperkecil nilai Kd, dimana Ki juga relatif kecil
- Saat *e* dan d*e* berada disekitar nilai set maka parameter Kp diperkecil agar mengurangi *overshoot*. Nilai Ki dan Kd tidak berubah.
- Saat e nilainya sangan kecil maka Kp dan Ki, dinaikan. Saat *delta error* juga kecil maka Kd diperbesar.

Yang mana aturan *fuzzy rule* tersebut tergambar pada Tabel 3.1 menunjukan *fuzzy rule Kp,* Tabel 3.2 menunjukan *fuzzy rule Kd,* Tabel 3.3 menunjukan *fuzzy rule α,*

Tabel 3. 1 *fuzzy Rule Kp*

dError / Error	N	Z	P
N	P	P	P
Z	N	P	N
P	P	P	P

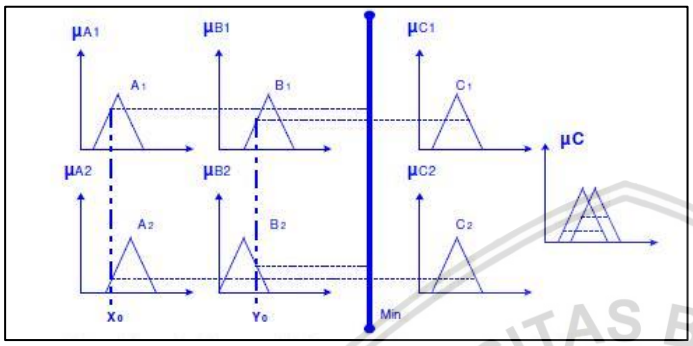
Tabel 3. 2 *fuzzy Rule Kd*

dError / Error	N	Z	P
N	N	N	N
Z	P	P	P
P	N	N	N

Tabel 3. 3 fuzzy Rule Ki

dError / Error	N	Z	P
N	N	N	Z
Z	N	Z	P
P	Z	P	P

Tahap defuzzifikasi merupakan tahap terakhir dalam perancangan kontroler Fuzzy. Metode proses defuzzifikasi yang diinginkan adalah metode Max-Min. Hasil proses tersebut dikomposisikan yang didefuzzifikasikan dengan menggunakan metode *Center of Area*.



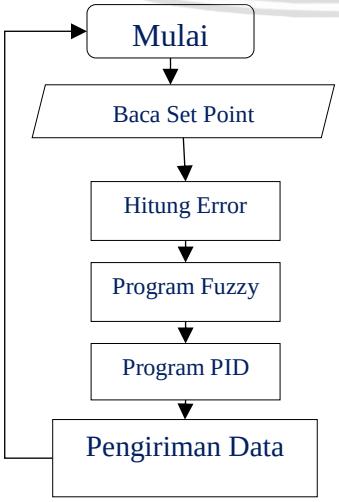
Gambar 3.10 Proses Inferensi dengan metode Max-Min

Pada Gambar 3.10 merupakan kompisisi keluaran dan batas-batas areayang telah ditentukan dengan persamana matematis, sehingga hasil defuzzifikasi memiliki persamaan sebagai berikut:

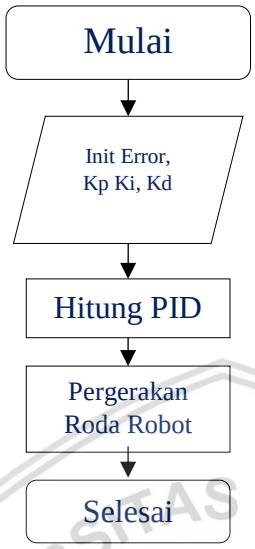
$$z = \frac{\sum \mu c\,(z).\ z}{\sum \mu c\,(z)} \qquad (3.1)$$

3.6 Perancangan Algoritma

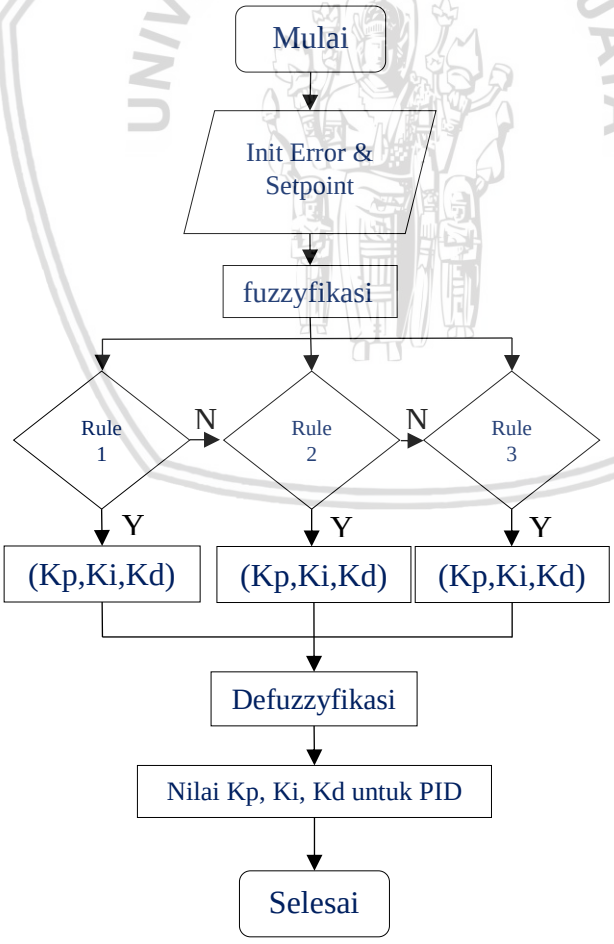
Perancangan Algortima adalah perancangan untuk algoritma untuk mendapatkan nilai *tuning* PID dengan seleksi sensor jarak dari nilai pembacaan, dimana secara garis besar algoritma untuk penelitian ini yaitu dibagi menjadi perancangan algoritma fuzzyfikasi, PID dan algoritma gabungan yang ditunjukan pada gambar berikut:



Gambar 3. 11 Digram alir keseluruhan sistem



Gambar 3. 12 Digram alir perhitungan PID



Gambar 3. 13 Digram alir proses *Fuzzy*

Pembacaan *setpoint* dan *feedback* kecepatan akan digunakan sebagai masukan untuk program *Fuzzy* dan PID. Hasil program fuzzy akan digunakan untuk mendapatkan parameter kontrol PID yaitu Kp, Ki, dan Kd. Nilai keluaran PID akan digunakan sebagai *manipulated variabel* untuk memanipulasi sudut pada roda-roda robot.

BAB IV

HASIL DAN PEMBAHASAN

Hasil dan pembahasan dalam penelitian ini dikerjakan untuk mengetahui kinerja dari keseluruhan sistem apakah sudah sesuai dengan perancangan. Pengujian dilakukan secara bertahap pada masing-masing perangkat keras kemudian dilakukan pengujian secara kesuluruhan. Pengujian yang dilakukan adalah sebagai berikut:

1. Pengujian Aktuator.
2. Pengujian Sensor IMU MPU6050.
3. Pengujian Keseluruhan Sistem

4.1 Pengujian Aktuator

4.1.1 Tujuan Pengujian

Tujuan pengujian aktuator adalah untuk mengetahui *output* dari aktuator yaitu motor DC apabila diberi *input* yang berbeda-beda.

4.1.2 Peralatan Yang Digunakan

Berikut adalah alat yang digunakan dalam pengujian aktuator, antara lain:

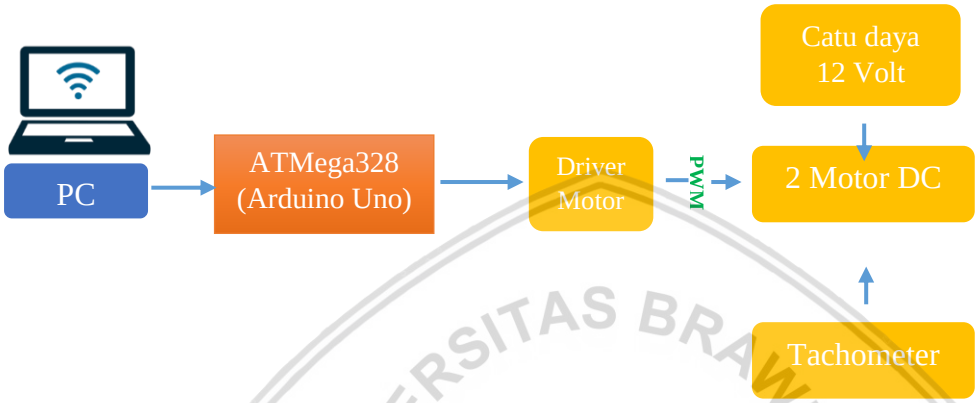
1. Mikrokontroler ATMega 328
2. Motor DC
3. Komputer
4. *Software Arduino*
5. Tachometer Digital
6. Catu daya 12 Volt.

4.1.3 Langkah pengujian

Langkah pengujian dilakukan dengan tahapan sebagai berikut:

1. Merangkai alat seperti pada Gambar 4.1
2. Membuat program untuk menetukan arah dan kecepatan motor pada *Software Arduino*.
3. Load Flash program yang telah dibuat, hapus program yang ada dalam mikrokontroler ATMega 328, tulis program yang telah dibuat ke dalam mikrokontroler ATMega 328 menggunakan *Software Arduino*.

4. Mengukur tegangan pada pin PWM dari mikrokontroler ATMega 328 yang digunakan sebagai pinenable driver motor.
5. Mengukur kecepatan pada motor mengguakan tachometer.
6. Menggambar data yang didapatkan pada grafik.



Gambar 4.1 Diagram Blok Pengujian *Driver* Motor

4.1.4 Hasil Pengujian dan Analisis

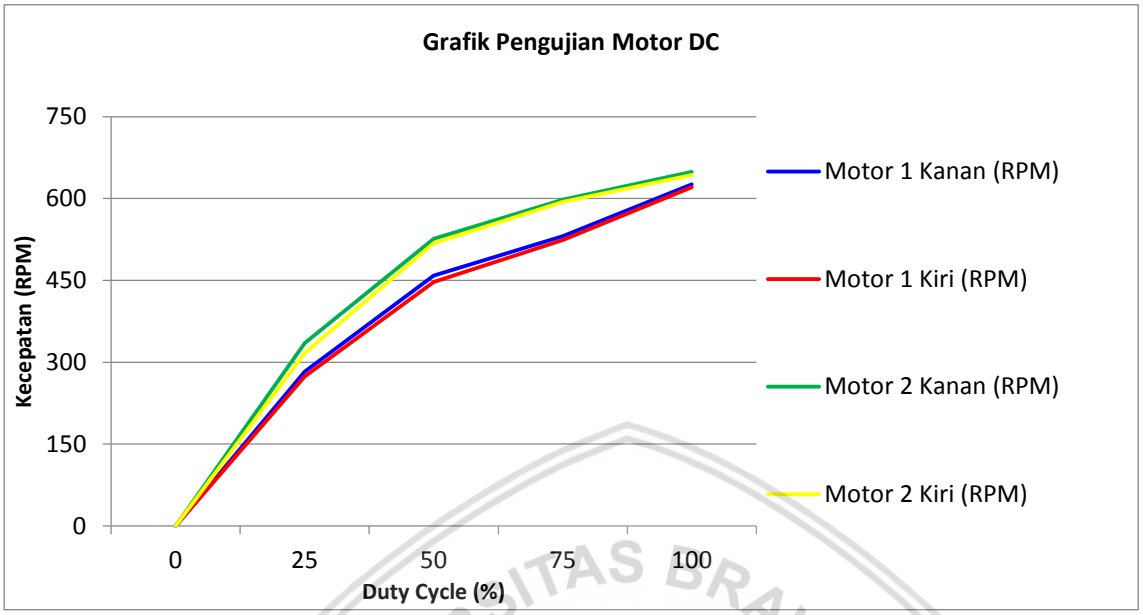
Pada Gambar 4.1 menunjukan diagram *blok* pengujian motor DC. Ada enam buah *output* dari mikrokontroler ATMega328 (Arduino Uno) yang digunakan sebagai input driver motor. Enam buah output itu adalah *pin* D9, *pin* D8, *pin* D7, dan *pin* D6 yang menjadi input IN1, IN2, IN3, dan IN4 dari driver motor dan berfungsi sebagai penentu arah berputarnya motor, serta *pin* D5 dan *pin* D10 yang menjadi input 2 PWM dari *driver* motor yang berfungsi sebagai penentu kecepatan motor.

Pada program dilakukan *setting output* PWM. Pin D7 di *setting* aktif pada logika tinggi dan *pin* D6 di *setting* aktif pada logika rendah. Dengan *setting* tersebut maka apabila *input pin* D5 pada block diberi masukan bernilai 255 maka pin D5 akan aktif, dan pada Tabel 4.1 menunjukan data pengukuran motor DC.

Tabel 4.1 Pengukuran Arah dan Kecepatan Motor DC

No.	Data Motor 1			
	Duty Cycle (%)	PWM	RPM	Arah Putaran Motor DC
1.	0	0	0	Diam
2.	25	64	279	Kanan
3.	50	127	457	Kanan
4.	75	191	531	Kanan
5.	100	255	626	Kanan
6.	0	0	0	Diam
7.	25	-64	274	Kiri
8.	50	-127	451	Kiri
9.	75	-191	524	Kiri
10.	100	-255	621	Kiri

No.	Data Motor 2			
	Duty Cycle (%)	PWM	RPM	Arah Putaran Motor DC
1.	0	0	0	Diam
2.	25	64	335	Kanan
3.	50	127	526	Kanan
4.	75	191	598	Kanan
5.	100	255	649	Kanan
6.	0	0	0	Diam
7.	25	-64	317	Kiri
8.	50	-127	518	Kiri
9.	75	-191	594	Kiri
10.	100	-255	643	Kiri



Gambar 4.2 Grafik Pengujian Motor DC

Dari hasil pengujian aktuator yaitu motor DC berdasarkan Tabel 4.1 dan Gambar 4.2 didapatkan karakteristik dari motor DC semakin besar *duty cycle* (%) yang digunakan maka akan semakin besar pula kecepatan putaran yang akan dihasilkan oleh motor DC. Namun, besarnya kecepatan putaran tiap motor berbeda meskipun dengan jenis dan tipe motor yang sama. Hal ini dikarenakan untuk motor 1 terjadi kelonggaran dalam gearbox motor tersebut. Untuk cara melakukan pengukuran terhadap motor DC ditunjukan pada Gambar 4.3.

Gambar 4.3 Cara Pengujian Motor DC

4.2 Pengujian Sensor IMU MPU6050

4.2.1 Tujuan Pengujian

Tujuan pengujian sensor IMU MPU6050 adalah untuk mengetahui tingkat keakuratan dari sensor IMU MPU6050 dalam membaca perubahan sudut dari robot beroda dua.

4.2.2 Peralatan Yang Digunakan

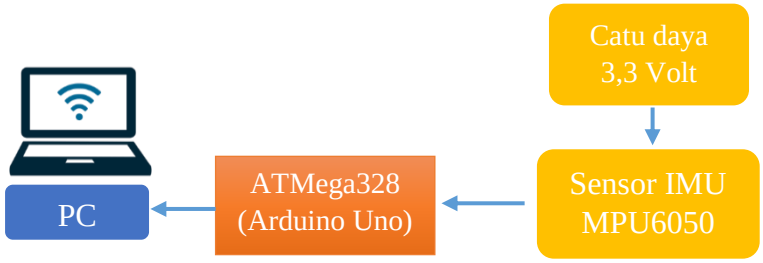
Berikut adalah alat yang digunakan dalam pengujian aktuator, antara lain:

1. Sistem mekanik robot beroda dua yang terhubung dengan sensor IMU MPU6050 dan mikrokontroler ATMega 328.
2. Komputer
3. *Software Arduino*
4. Papan sudut atau busur sudut
5. Catu daya 3,3 Volt

4.2.3 Langkah pengujian

Langkah pengujian dilakukan dengan tahapan sebagai berikut:

1. Merangkai alat seperti pada Gambar 4.4
2. Membuat program untuk sensor IMU MPU6050 pada *Software Arduino*
3. Mengubah posisi kemiringan dari robot beroda dua dan mengukur sudut dengan menggunakan papan sudut
4. Melihat besaran sudut melalui computer
5. Menggambarkan data keluaran sensor IMU MPU6050 pada grafik untuk melihat kelinieran dari sensor IMU MPU6050.



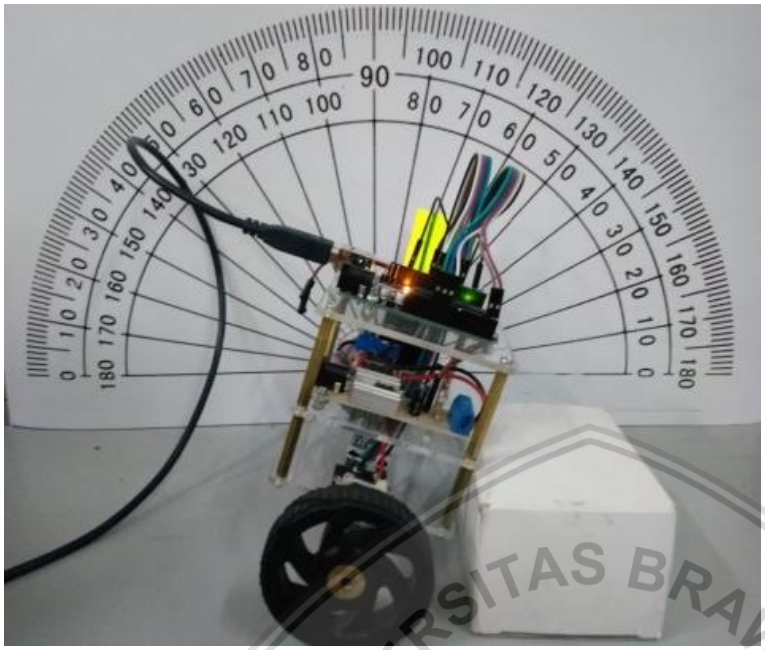
Gambar 4.4 Diagram Blok Pengujian Sensor IMU MPU6050

4.2.4 Hasil Pengujian dan Analisis

Pada gambar 4.4 menunjukan diagram blok pengujian sensor IMU MPU6050. Sensor IMU MPU6050 terhubung dengan system mekanik robot beroda dua. Kemudian data dari sensor IMU MPU6050 akan dibuat input dan keluaranya adalah kecepatan putar motor DC. Kemiringan dari robot beroda dua diubah-ubah sesuai dengan papan sudut yang telah disediakan. Perubahan sudut yang terjadi akan ditampilkan lewat komputer. Tabel 4.2 menunjukkan pengamatan pada sensor dan Gambar 4.5 menunjaukkan cara pengambilan data pada sensor.

Tabel 4.2 Data Pengamatan Pada Sensor

No	Data			
	Sudut Sensor (º)	Sudut Aktual (º)	Error	Arah Mekanik Robot
1.	41,07	40	1,07	Kanan
2.	30,69	30	0.69	Kanan
3.	20,84	20	0.84	Kanan
4.	10,91	10	0.91	Kanan
5.	0,27	0	0.27	Diam
6.	-9,67	-10	-0,33	Kiri
7.	-19,39	-20	-0,61	Kiri
8.	-29,76	-30	-0.24	Kiri
9.	-39,43	-40	-0,57	Kiri
Rata-rata error			0,61	



Gambar 4.5 Cara Mendapatkan Data Sensor

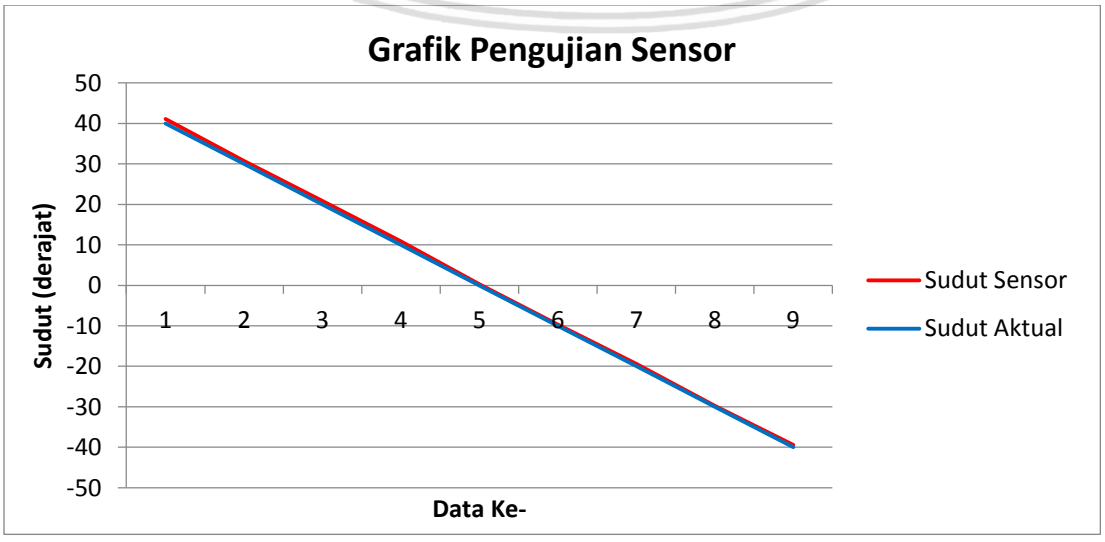
Terlihat dari Tabel 2 variasi pengukuran derajat yang memilik nilai *error* paling besar yaitu pada pengukuran 40^0 sebesar $1{,}07^0$ dan yang memiliki nilai *error* paling kecil yaitu pada pengukuran -30^0 sebesar $0{,}24^0$. Rata-rata kesalahan sensor dapat dihitung dengan menggunakan rumus 4.1.

$$\text{rata-rata kesalahan (\%)} = \frac{\sum error}{jumlah\ data} \quad (4.1)$$

sehingga didapat:

$$\text{rata-rata kesalahan (\%)} = \frac{5{,}53}{9} \quad (4.2)$$

$$\text{rata-rata kesalahan (\%)} = 0{,}61 \quad (4.3)$$



Gambar 4.6 Grafik Pengujian Sensor

Dari table 4.2 dan gambar 4.6 terlihat bahwa pembacaan sudut sensor sudah akurat, terlihat dari selisih *error* dengan sudut aktual yang nilainya kecil. Maka dapat disimpulkan bahwa sensor IMU MPU 6050 sudah bekerja dengan baik.

4.3. Pengujian Keseluruhan

4.3.1 Tujuan Pengujian

Tujuan pengujian keseluruhan adalah untuk mengetahui kerja dari perangkatkeras dan perangkat lunak setelah diintegrasikan dalam sebuah sistem terpadu.

4.3.2 Peralatan Yang Digunakan

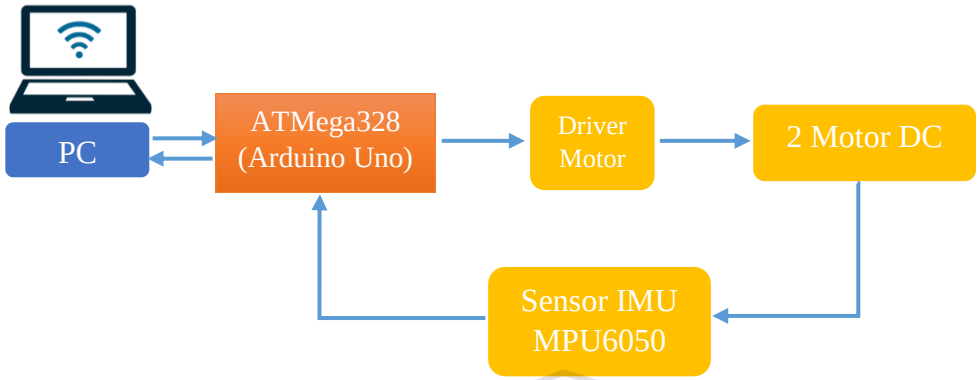
Berikut adalah alat yang digunakan dalam pengujian aktuator, antara lain:

1. Plant robot beroda dua yang terhubung dengan sensor IMU MPU6050 dan motor DC.
2. Driver motor L298N
3. Mikrokontroler ATMega 328.
4. Komputer
5. Kabel serial
6. *Software Arduino*
7. Program *Microsoft Exel*

4.2.3 Langkah pengujian

Langkah pengujian dilakukan dengan tahapan sebagai berikut:

1. Merangkai alat seperti pada Gambar 4.7
2. Mengaktifkan semua catu daya
3. Membuat program untuk pengendalian robot beroda dua pada *Software Arduino*
4. Melakukan *make a project* pada *Software Arduino*
5. Melakukan *verify* pada *project* yang tela dibuat, apabila tidak ada *error* langsung *upload* ke dalam mikrokontroler.
6. Mengamati kinerja dari robot beroda dua dalam menjaga keseimbangan.



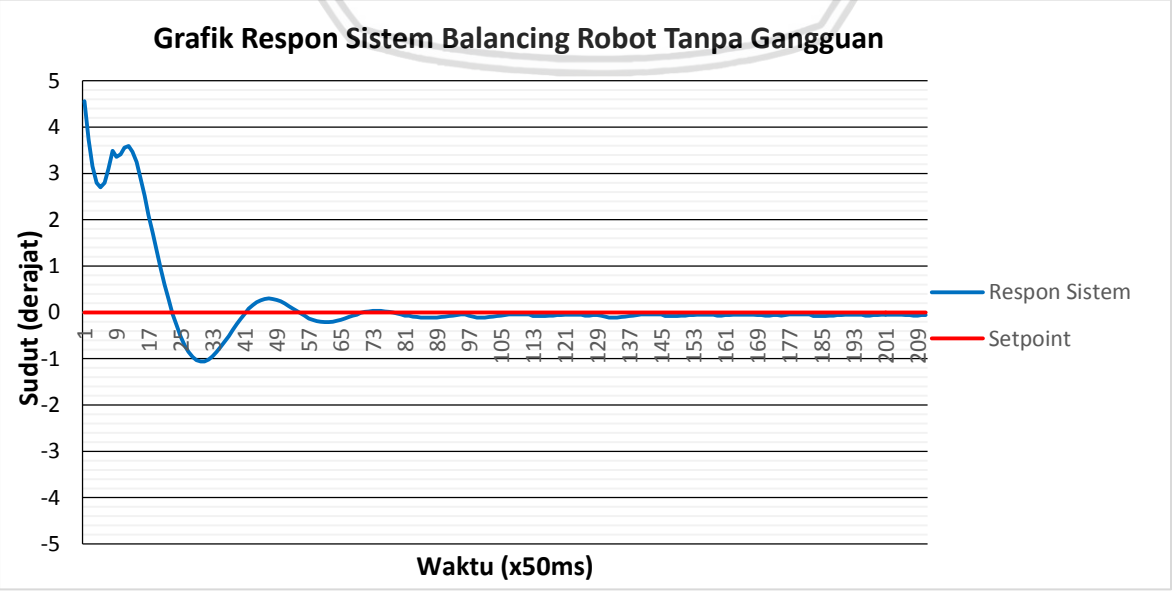
Gambar 4.7 Diagram Blok Pengujian Keseluruhan

4.3.4 Hasil Pengujian dan Analisis

Pada gambar 4.7 menunjukan blok pengujian keseluruhan sistem keseimbangan robot beroda dua. Mula-mula robot beroda dua diberdirikan posisi tegak lurus terhadap permukaan bidang datar (0^0) kemudian dilepas untuk melihat kemampuan sistem penyeimbang dalam menjaga keseimbangan robot beroda dua. Pada pengujian keseluruan ini terbagi menjadi dua yaitu Pengujian *balancing robot* tanpa gangguan dan Pengujian *balancing robot* dengan gangguan.

4.3.4.1 Pengujian *Balancing Robot* Tanpa Gangguan

Pada Pengujian *balancing robot* tanpa gangguan dilakukan setelah kendali PID berfungsi mengendalikan robot agar dapat menjaga keseimbangan robot beroda dua.



Gambar 4.8 Pengujian *Balancing Robot* Tanpa Gangguan

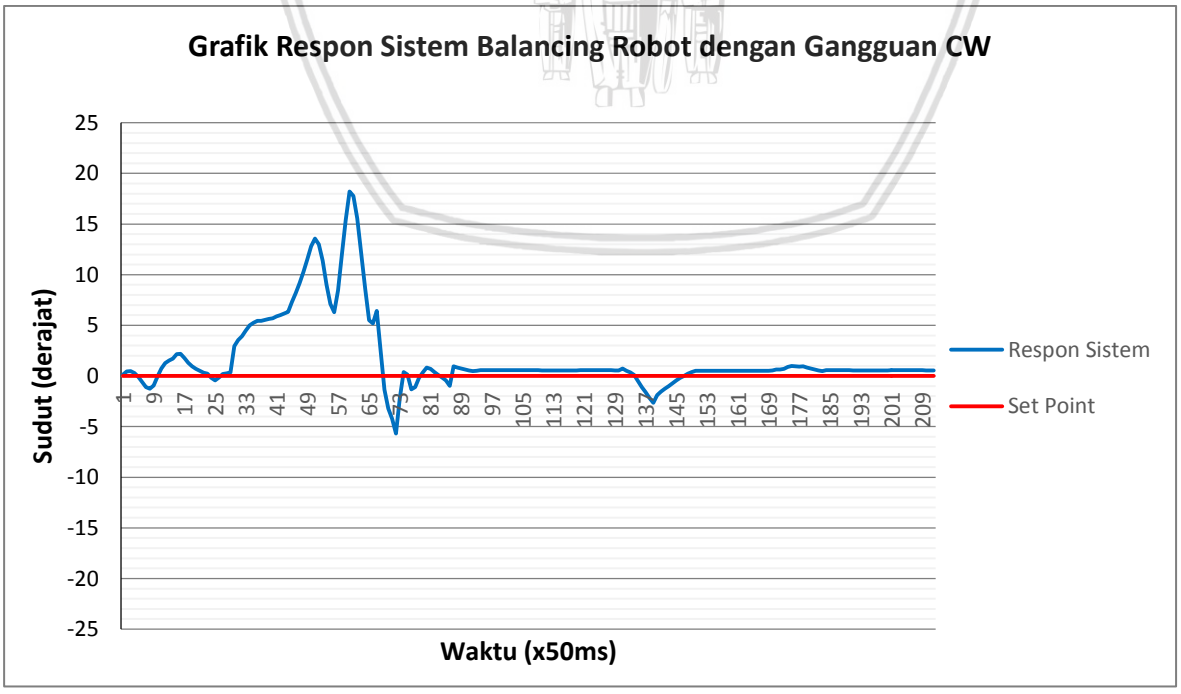
Berdasarkan data yang didapatkan pada pengujian robot mampu memberikan aksi kontrol. Terlihat pada gambar 4.8 respon sistem *balancing robot* dapat menyeimbangkan diri tanpa diberi gangguan dari luar. Respon sistem *balancing robot* dapat seimbang mendekati nilai *set point* 0^0 yaitu antara 0^0 hingga 1^0, dengan persentase *error steady state* sebesar kurang dari 2% dimana terlihat ketika respon nilai error mendekati nilai *set point*.

4.3.4.2 Pengujian *Balancing Robot* dengan Gangguan

Pada Pengujian *balancing robot* dengan gangguan dari luar. Gangguan saat kondisi sedang menyeimbangkan diri adalah memberikan dorongan searah jarum jam atau *clockwise* (CW) dan berlawanan arah jarum jam atau *counter clockwise* (CCW) secara perlahan sehingga mengakibatkan adanya perubahan sudut.

A. Pengujian dengan Gangguan Searah Jarum Jam (CW)

Dalam pengujian ini *balancing robot* diberi sebuah gangguan secara perlahan searah jarum jam atau *clockwise* (CW).

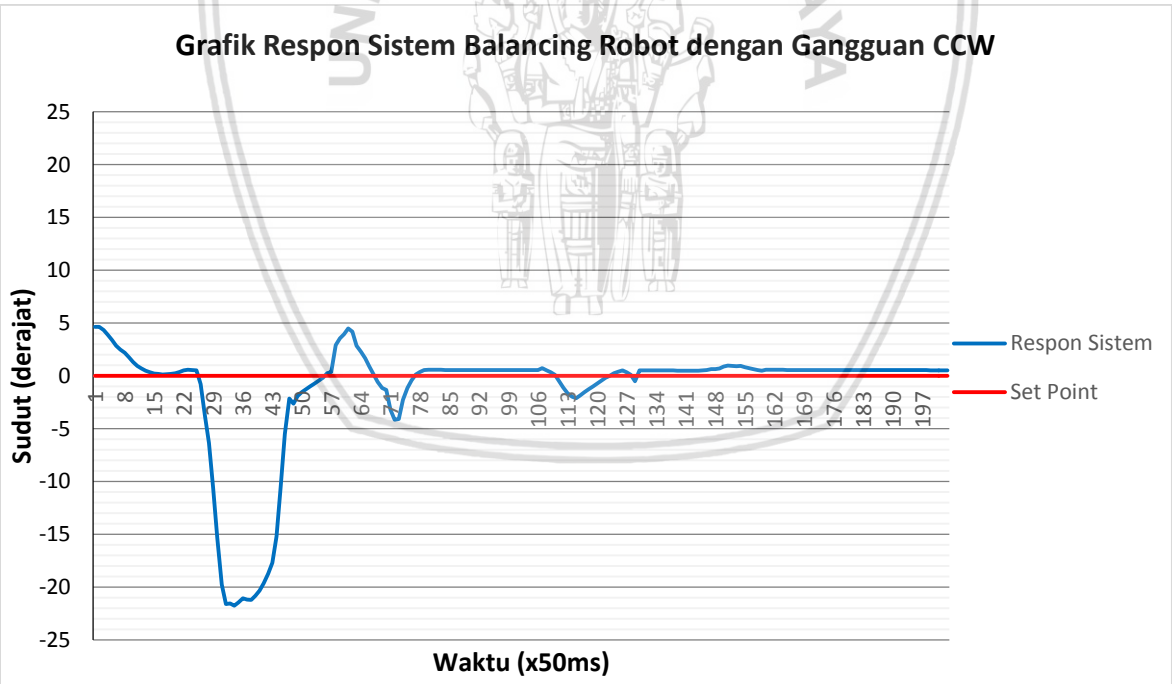


Grafik 4.9 Respon *Balancing Robot* Penguijian CW

Dalam pengujian ini diberi sebuah gangguan secara perlahan searah jarum jam atau *clockwise* (CW), dengan cara didorong perlahan searah jarum jam. Terjadi ganguan pada *balancing robot* saat waktu sampling ke 60 gangguan yang didapat *balancing robot* yaitu sebesar $18{,}2^0$. Setelah diberikan sebuah gangguan, *balancing robot* langsung berusaha menyeimbangkan diri. *Balancing robot* ternyata dapat menyeimbangkan diri dan tidak terjatuh. Respon sistem *balancing robot* dapat seimbang mendekati nilai *set point* 0^0 yaitu antara 0^0 hingga 3^0, dengan persentase *error steady state* sebesar kurang dari 5% dimana terlihat ketika respon nilai error mendekati nilai *set point*.

B. Pengujian dengan Gangguan Berlawanan Arah Jarum Jam (CCW)

Dalam pengujian ini *balancing robot* diberi sebuah gangguan secara perlahan berlawanan arah jarum jam atau *counter clockwise* (CCW).



Grafik 4.10 Respon *Balancing Robot* Penguijian CCW

Dalam pengujian ini *robot* diberi sebuah gangguan secara perlahan berlawanan arah jarum jam atau *counter clockwise* (CCW), dengan cara didorong perlahan berlawanan arah jarum jam. Terjadi ganguan pada *balancing robot* saat waktu sampling ke 36 gangguan yang didapat *balancing robot* yaitu sebesar $-21{,}77^0$. Setelah diberikan sebuah gangguan,

balancing robot langsung berusaha menyeimbangkan diri. *Balancing robot* ternyata dapat menyeimbangkan diri dan tidak terjatuh. Respon sistem *balancing robot* dapat seimbang mendekati nilai *set point* 0^0 yaitu antara 0^0 hingga 4^0, dengan persentase *error steady state* sebesar kurang dari 5% dimana terlihat ketika respon nilai error mendekati nilai *set point*.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan perencanaan, pembuatan, dan pengujian yang telah dilakukan terhadap alat baik pengujian pada sub-sistem maupun pengujian seluruh sistem, maka dapat disusun kesimpulan sebagai berikut:

1. Penerapan logika *Fuzzy* ditunjukan untuk mendapatkan paramater kontrol PID, sesuai dengan perancangan, serta sistem elektronik, mekanik maupun pola keseimbangan robot agar dapat diimplementasikan pada *balancing robot.* Dimana saat pembacaan sensor sudut semakain mendekati nilai *set point* maka aksi kontrol akan berkurang dan error semakin kecil yaitu terlihat pada respon sistem *balancing robot* dapat seimbang mendekati nilai *set point* 0^0 diperoleh antara 0^0 hingga 4^0, dengan persentase *error steady state* sebesar kurang dari 5% dimana terlihat ketika respon nilai error mendekati nilai *set point*.
2. Pembacaan sensor IMU MPU6050 dapat digunakan sebagai nilai batas yang menetukan aturan logika *Fuzzy*. Dimana nilai kondisi yang diharapkan yaitu robot bisa menyeimbangkan diri di sudut 0^0 pada permukaan yang datar. Kemudian nilai pembacaan sensor akan digolongkan sebagai kondisi miring searah jarum jam (CW), miring berlawanan arah jarum jam (CCW), dan tegak oleh defuzzyfikasi, dengan mencari rata-rata dari keluaran seleksi kondisi *fuzzyfier*. Kemudian nilai tersebut dijadikan parameter masukan nilai *gain* kontroler PID. Yang kemudian dijadikan sinyal manipulasi gerak untuk pergerakan roda robot. *Balancing robot* dapat menyeimbangkan diri pada permukaan yang datar ketika tanpa gangguan. Pada pengujian dengan gangguan searah jarum jam (CW) sudut maksimal yang masih dapat diseimbangkan oleh sistem adalah sebesar $18{,}2^0$, sedangkan pada gangguan berlawanan arah jarum jam (CCW) sebesar $-21{,}77^0$.

5.2 Saran

Beberapa saran yang diberikan untuk perbaikan skripsi ini antara lain:

1. Penyempurnaan konstruksi mekanik terutama pada sistem mekanik dari gearbox motor DC agar lebih simetris antara satu dengan yang lainya dan pemilian roda motor yang sesuai dengan motor DC.
2. Untuk penelitian selanjutnya, dapat digunakan metode kontrol lain
3. Dilakukan metode pengambilan data menggunakan wireless. Karena apabila menggunkan kabel maka respon data dari robot beroda dua terjadi lebih besar error yang disebabkan oleh beratnya kabel. Hal ini membebani pada salah satu sisi sehingga pada saat pengambilan data digunakan perangkat tanmbahan yaitu pembeban yang ditempatkan pada sisi berlawananya

DAFTAR PUSTAKA

ATMEL. (2017, Oktober 17). *Datasheet.* Diambil kembali dari atmel.com: http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf.

Bachri, Samsul. 2004. Sistem Kendali Hybrid PID - Logika Fuzzy Pada Pengaturan Kecepatan Motor DC. Jember. Teknik Elektro, Program Studi Teknik, Universitas Jember. Skripsi.

Dwi Madyanto, Tunjung. 2006. Pengontrolan Suhu Menggunakan Metode *FUZZY*-PID Pada Model Sistem Hipertermia. Semarang. Fakultas Teknik Universitas Diponegoro. Skripsi.

InvenSense Inc. 2013. MPU 6050 GY-521 3 Axis Gyro Accelerometer Datasheet

Lee, C. (1990). *Fuzzy logic in control systems: fuzzy logic controller, Parts I and II. IEEE Trans Syst Man Cybern.*

Ogata, K. (1997). *Modern Control Engineering Fifth Edition.* New Jersey: Prentince Hall.

Rokhmat, M. Miftahur. 2013. Implementasi Sistem Keseimbangan Robot Beroda Dua Dengan Menggunakan Kontroler Proporsional Integral Diferensial. Malang. Fakultas Teknik Universitas Brawijaya. Skripsi.

Rusli, Mochammad. 2017. *Dasar Perancangan Kendali Logika Fuzzy*. Malang. UB Media.

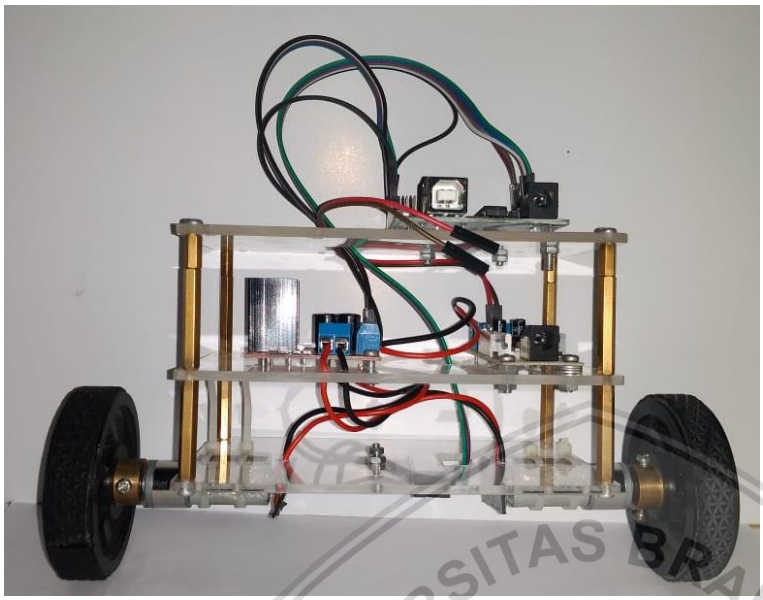
Putra, D. A. 2015. *Penerapan Kontroler Self Tuning Parameter PI dengan metode Logika Fuzzy pada Mobile Robot*. Malang. Universitas Brawijaya.

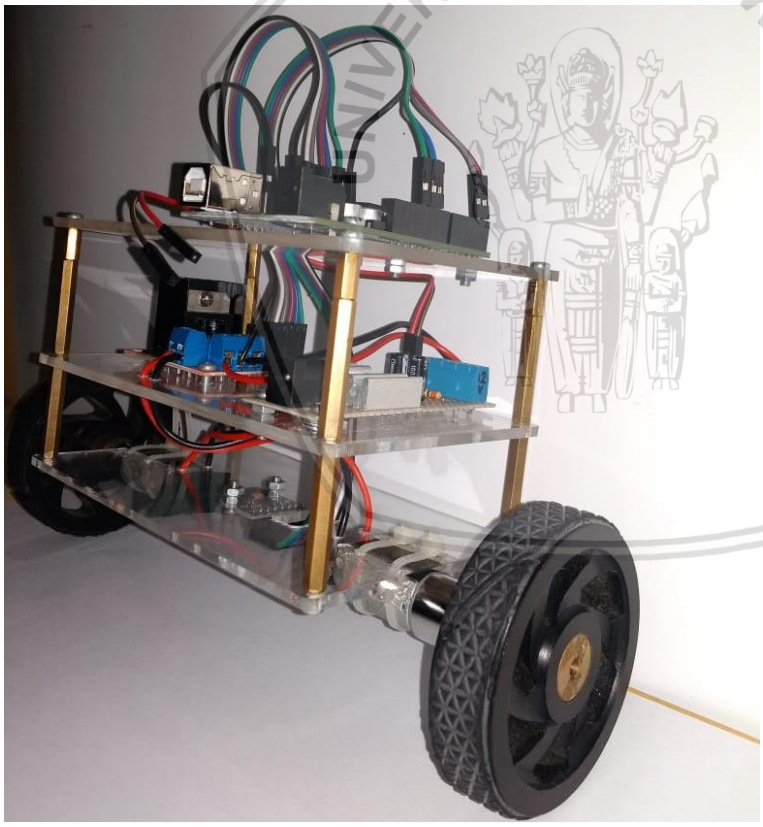
Sugeno, M., & Takagi, T. 1985. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. Systems Man Cybern*, 116–132

STMicroelectronics. 2010. L298, *DUAL FULL-BRIDGE DRIVER*. https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf, diakses tanggal 30 Mei 2018

LAMPIRAN

Lampiran 1. Dokumentasi Alat





Lampiran 2. *Listing* Program

```
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif
MPU6050 mpu;
#define OUTPUT_READABLE_YAWPITCHROLL
#define LED_PIN 13

#define NB  0   // Kondisi Miring Kanan
#define ZB   1   // Kondisi Tegak
#define PB  2   // Kondisi Miring Kiri

bool blinkState = false;
// 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
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector
float sudut;
uint8_t teapotPacket[14] = { '$', 0x02, 0,0, 0,0, 0,0, 0,0, 0x00, 0x00, '\r', '\n' };
//                                  Variables
float OldP = 0;         // Previous value used to calculate change in P //  DELTA P
float P = 0;            //  Proportional component
float I = 0;            //  Integral        just the sum of P over time
float OldI = 0;         //  previous value of I for calculation of Delta I
float D = 0;            //  Differential       D = P - OldP
float bp = -60;         // balance point
float pwm = 0;          // value of Pulse Width Modulation  to ENA ENB
long a = 0;             // L298N to IN 1 to 4
long b = 0;             //
const int PinR1 = 6;    //  arduino  pin 5 to l298  pin IN4
const int PinR2 = 7;    //  arduino  pin 6 to l298  pin IN3
const int PinL1 = 9;    //  arduino  pin 7 to l298  pin IN1
const int PinL2 = 8;    //  arduino  pin 8 to l298  pin IN2
const int PwmR  = 5;    //  arduino  pin 9 to l298  pin ENB
const int PwmL  = 10;   //  arduino  pin 10 to l298  pin ENA
// ===                Fuzzy PID DECLARATION                ===
```

```
  float KpF=0;
  float KiF=0;
  float KdF=0;
float Ts = 0.05;  // settling time (50ms)
float MAX = 250;
float MIN = -250;
float error_v[2],PID,u[2],
      last1_error,last2_error,
      error,derror,serror,
      last1error,last2error,
      error_fuzzy;
float derror_fuzzy=0;
float  serror_fuzzy=0;
int check_rule[9];
float Error[3];
float dError[3];
float umin;
float Gain[3];
float Gain_out = 0;
float MIN_Value(float data1, float data2);
float MAX_Value(float data1, float data2);
float rule_base_kp();
float rule_base_ki();
float rule_base_kd();
float fuzzy_kp();
float fuzzy_ki();
float fuzzy_kd();
// =====              INTERRUPT DETECTION ROUTINE              =====
volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone high
void dmpDataReady() {
   mpuInterrupt = true;
}
// =====                  INITIAL SETUP                  =====
void setup() {
// arduino to l298 pins hopefully self explanatory
      pinMode(PinR1,OUTPUT);
      pinMode(PinR2,OUTPUT);
      pinMode(PinL1,OUTPUT);
      pinMode(PinL2,OUTPUT);
   // 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
   Serial.begin(57600);
   while (!Serial);
   // 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
 //  use calibration program to get your own values
    mpu.setXGyroOffset(60);//(220);
    mpu.setYGyroOffset(-2);//(76);
    mpu.setZGyroOffset(25);//(-85);
    mpu.setXAccelOffset(-380);//(1788); // 1688 factory default for my test chip
    mpu.setYAccelOffset(500);
    mpu.setZAccelOffset(2519);
    // 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();
    } else {
        Serial.print(F("DMP Initialization failed (code "));
        Serial.print(devStatus);
        Serial.println(F(")"));
    }
    // configure LED for output
    pinMode(LED_PIN, OUTPUT);
}
// ===                MAIN PROGRAM LOOP                ===
void loop() {
sudut=((ypr[2])*90)/1.1;
    // 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) {
        // other program behavior stuff here
     }
    // 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;
   #ifdef OUTPUT_READABLE_YAWPITCHROLL
    // display Euler angles in degrees
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

    OldP =P;                    // save value of P
    KpF = fuzzy_Kp();
    error = (ypr[2] * 1000) + bp;
    P = KpF * error;
    OldI = I;                  // save old I
    I = I + (P * 0.05) ;
    I = I + ((I - OldI)*2  );     // calulate new I
    if( I >  250 ) I =  250;          // LIMIT  Stop I building up too high
    if( I < -250 ) I = -250;          // or too low value
    D = P - OldP;                    //  D differential   change in P
    pwm = ( P * 1 ) + ( I  ) + ( D * 10 ) ; // P I D
    a = 0;
    b = 0;
    if(pwm < 0){
      a = 0;
      b = 1;
      bp = bp - 0.01;
      digitalWrite(13, 0);
    }
    if(pwm > 0){
      a = 1;
      b = 0;
      bp = bp + 0.01;
      digitalWrite(13, 1);
    }
    // remove sign from PWM as - value has no meaning
    pwm  = abs(pwm);
```

```
        if ( pwm < 0) pwm = 0;
        if ( pwm > 255) pwm = 255;
          if(abs(ypr[2]) < abs(0.35)){
            analogWrite(PwmR, pwm);
            digitalWrite(PinR1, a);
            digitalWrite(PinR2 ,b);
            analogWrite(PwmL ,pwm);
            digitalWrite(PinL1 ,a);
            digitalWrite(PinL2 ,b);
            }
          else{
            analogWrite(PwmR , 0);
            analogWrite(PwmL , 0);
            I = 0;
            bp = -98;
            delay(1000); }
       Serial.print("sudut :");
       Serial.println(sudut);
       Serial.print("pwm :");
       Serial.println(pwm);
       Serial.print("error :");
       Serial.println(error);

       #endif
   }
}
float MIN_Value(float data1, float data2)
{
  float output = 0 ;
  if (data1 < data2) output=data1 ;
  else output=data2 ;
  return output;  }
float MAX_Value(float data1, float data2)
{
  float output = 0 ;
  if (data1 < data2) output=data2 ;
  else output=data1 ;
  return output;  }
////// -----------FUZZY- MAMDANI RULE's-------------------------------------/////
float triangle(float value,float x0, float x1, float x2)
{
  float result = 0;
  float x;
  x = value;  // x = adalah variabel yang mewakili alamat dari PING yang diakses
    if ((x <=x0)||(x>=x2))
    {
      result=0;
    }
    else if(x==x1)
    {
```

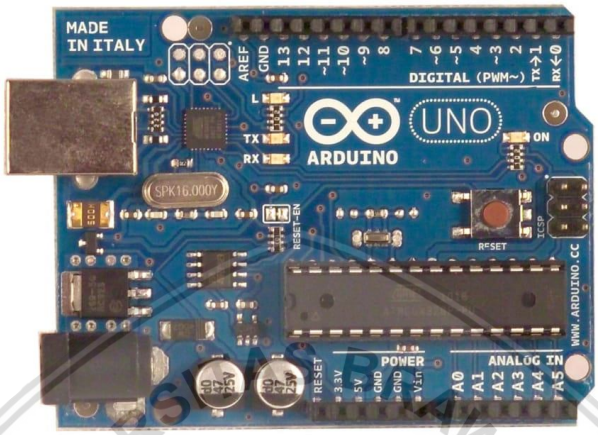
```
      result=1;
    }
    else if ((x>=x0)&&(x<x1))
    {
      result=((x-x0)/(x1-x0));
    }
    else
    {
      result = (((-x)+x2)/(x2-x1));
    }
    return result;  }
float grade(float value, float x0, float x1)
{
  float result = 0;
  float x;
  x = value;
  if (x<=x0)
  {
    result=0;
  }
  else if(x>=x1)
  {
    result=1;
  }
  else
  {
    result = (x-x0)/(x1-x0);
  }
  return result;
}
float reverse_grade(float value,float x0, float x1)
{
  float result = 0;
  float x;
  x = value;
  if (x<=x0)
  {
    result=1;
  }
  else if(x>=x1)
  {
    result=0;
  }
  else
  {
    result =(-x+x1)/(x1-x0);
  }
  return result;
}
void fuzzyfikasi()
```

```
{
 Error[NB] = reverse_grade(abs(ypr[2]),0.35,0);
 Error[ZB] = triangle(abs(ypr[2]),0.35,0,0.35);
 Error[PB] = grade(abs(ypr[2]),0,0.35);
}
float rule_base_Kp()
{
 char x,y,z;
 float Gain[3];
 float Gain_out = 0;
 for (x=NB;x<=PB;x++)
 {
  if     (x==NB) Gain[0]   = -1*Error[x];
  else if (x==ZB) Gain[0]   = 0;
  else           Gain[0]   = 1*Error[x];
 }
 Gain_out = Gain[0];
 return Gain_out;
}
float rule_base_Ki()
{
 char x,y,z;
 float Gain[3];
 float Gain_out = 0;
 for (x=NB;x<=PB;x++)
 {
      if     (x==NB) Gain[0]  = -2*Error[x];
      else if (x==ZB) Gain[0]  = 0;
      else           Gain[0]  = 2*Error[x];
 }
 Gain_out = Gain[0];
 return Gain_out;
}
float rule_base_Kd()
{
 char x,y,z;
 float Gain[3];
 float Gain_out = 0;
 for (x=NB;x<=PB;x++)
 {
  if     (x==NB) Gain[0]   = -0.5*Error[x];
  else if (x==ZB) Gain[0]   = 0;
  else           Gain[0]   = 0.5*Error[x];
 }
Gain_out = Gain[0];
return Gain_out;
}
float fuzzy_Kp(){fuzzyfikasi();return rule_base_Kp();}
float fuzzy_Ki(){fuzzyfikasi();return rule_base_Ki();}
float fuzzy_Kd(){fuzzyfikasi();return rule_base_Kd();}
```

Lampiran 3. ***Datasheet***

1. Datasheet Mikrokontroler Atmega32

Arduino UNO



CE

Product Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduno, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Index

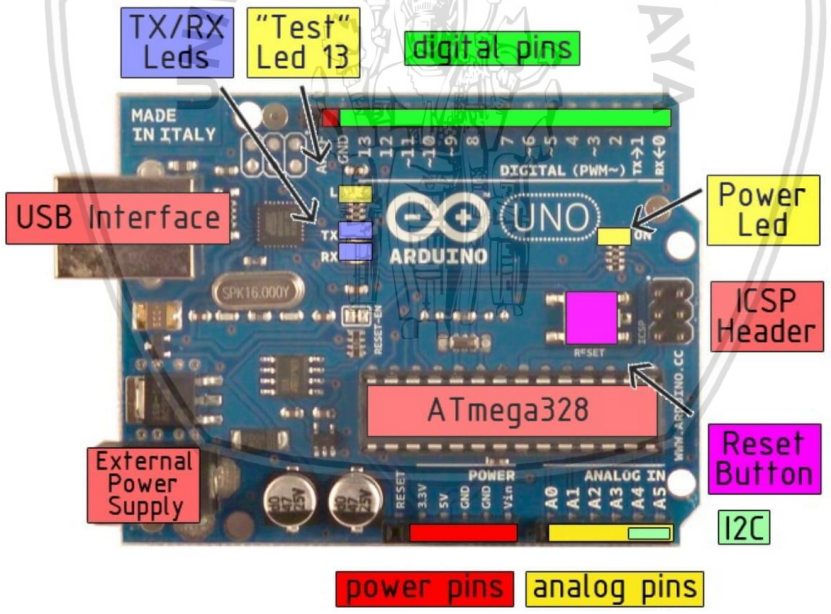
Technical Specification

EAGLE files: arduino-duemilanove-uno-design.zip Schematic: arduino-uno-schematic.pdf

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

the board



Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. TThese pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **I²C: 4 (SDA) and 5 (SCL).** Support I²C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and Atmega328 ports.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required..

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega328 datasheet.

Programming

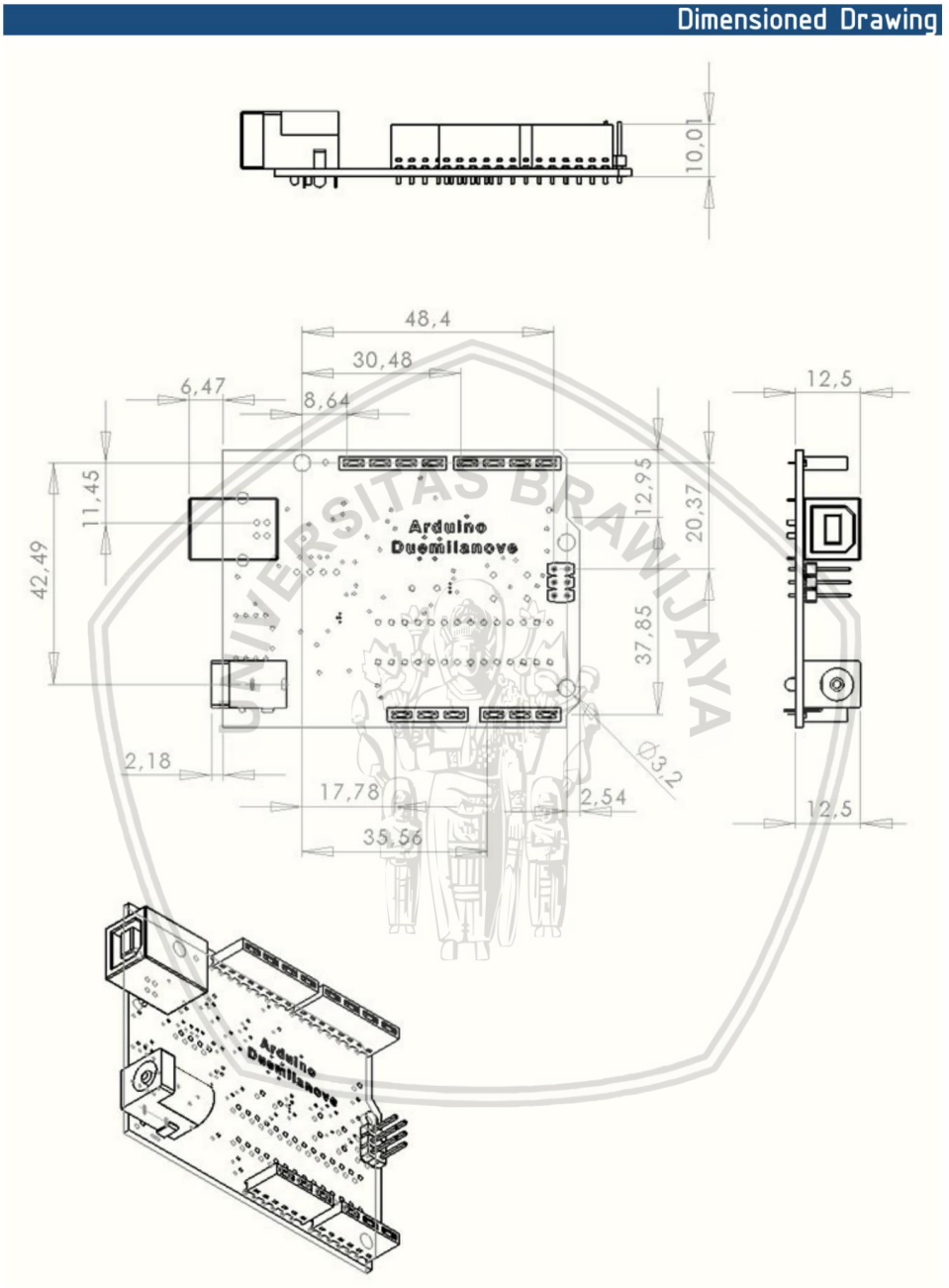
The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno w/ ATmega328" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader).

Dimensioned Drawing



2. Datasheet MPU6050

3 Product Overview

3.1 MPU-60X0 Overview

MotionInterface™ is becoming a "must-have" function being adopted by smartphone and tablet manufacturers due to the enormous value it adds to the end user experience. In smartphones, it finds use in applications such as gesture commands for applications and phone control, enhanced gaming, augmented reality, panoramic photo capture and viewing, and pedestrian and vehicle navigation. With its ability to precisely and accurately track user motions, MotionTracking technology can convert handsets and tablets into powerful 3D intelligent devices that can be used in applications ranging from health and fitness monitoring to location-based services. Key requirements for MotionInterface enabled devices are small package size, low power consumption, high accuracy and repeatability, high shock tolerance, and application specific performance programmability – all at a low consumer price point.

The MPU-60X0 is the world's first integrated 6-axis MotionTracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP) all in a small 4x4x0.9mm package. With its dedicated I^2C sensor bus, it directly accepts inputs from an external 3-axis compass to provide a complete 9-axis MotionFusion™ output. The MPU-60X0 MotionTracking device, with its 6-axis integration, on-board MotionFusion™, and run-time calibration firmware, enables manufacturers to eliminate the costly and complex selection, qualification, and system level integration of discrete devices, guaranteeing optimal motion performance for consumers. The MPU-60X0 is also designed to interface with multiple non-inertial digital sensors, such as pressure sensors, on its auxiliary I^2C port. The MPU-60X0 is footprint compatible with the MPU-30X0 family.

The MPU-60X0 features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs and three 16-bit ADCs for digitizing the accelerometer outputs. For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps) and a user-programmable accelerometer full-scale range of ±2*g*, ±4*g*, ±8*g*, and ±16*g*.

An on-chip 1024 Byte FIFO buffer helps lower system power consumption by allowing the system processor to read the sensor data in bursts and then enter a low-power mode as the MPU collects more data. With all the necessary on-chip processing and sensor components required to support many motion-based use cases, the MPU-60X0 uniquely enables low-power MotionInterface applications in portable applications with reduced processing requirements for the system processor. By providing an integrated MotionFusion output, the DMP in the MPU-60X0 offloads the intensive MotionProcessing computation requirements from the system processor, minimizing the need for frequent polling of the motion sensor output.

Communication with all registers of the device is performed using either I^2C at 400kHz or SPI at 1MHz (MPU-6000 only). For applications requiring faster communications, the sensor and interrupt registers may be read using SPI at 20MHz (MPU-6000 only). Additional features include an embedded temperature sensor and an on-chip oscillator with ±1% variation over the operating temperature range.

By leveraging its patented and volume-proven Nasiri-Fabrication platform, which integrates MEMS wafers with companion CMOS electronics through wafer-level bonding, InvenSense has driven the MPU-60X0 package size down to a revolutionary footprint of 4x4x0.9mm (QFN), while providing the highest performance, lowest noise, and the lowest cost semiconductor packaging required for handheld consumer electronic devices. The part features a robust 10,000*g* shock tolerance, and has programmable low-pass filters for the gyroscopes, accelerometers, and the on-chip temperature sensor.

For power supply flexibility, the MPU-60X0 operates from VDD power supply voltage range of 2.375V-3.46V. Additionally, the MPU-6050 provides a VLOGIC reference pin (in addition to its analog supply pin: VDD), which sets the logic levels of its I^2C interface. The VLOGIC voltage may be 1.8V±5% or VDD.

The MPU-6000 and MPU-6050 are identical, except that the MPU-6050 supports the I^2C serial interface only, and has a separate VLOGIC reference pin. The MPU-6000 supports both I^2C and SPI interfaces and has a single supply pin, VDD, which is both the device's logic reference supply and the analog supply for the part. The table below outlines these differences:

5 Features

5.1 Gyroscope Features

The triple-axis MEMS gyroscope in the MPU-60X0 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec
- External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.6mA
- Standby current: 5µA
- Factory calibrated sensitivity scale factor
- User self-test

5.2 Accelerometer Features

The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2*g*, ±4*g*, ±8*g* and ±16*g*
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500µA
- Low power accelerometer mode current: 10µA at 1.25Hz, 20µA at 5Hz, 60µA at 20Hz, 110µA at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- High-G interrupt
- User self-test

5.3 Additional Features

The MPU-60X0 includes the following additional features:

- 9-Axis MotionFusion by the on-chip Digital Motion Processor (DMP)
- Auxiliary master I^2C bus for reading data from external sensors (e.g., magnetometer)
- 3.9mA operating current when all 6 motion sensing axes and the DMP are enabled
- VDD supply voltage range of 2.375V-3.46V
- Flexible VLOGIC reference voltage supports multiple I^2C interface voltages (MPU-6050 only)
- Smallest and thinnest QFN package for portable devices: 4x4x0.9mm
- Minimal cross-axis sensitivity between the accelerometer and gyroscope axes
- 1024 byte FIFO buffer reduces power consumption by allowing host processor to read the data in bursts and then go into a low-power mode as the MPU collects more data
- Digital-output temperature sensor
- User-programmable digital filters for gyroscope, accelerometer, and temp sensor
- 10,000 *g* shock tolerant
- 400kHz Fast Mode I^2C for communicating with all registers
- 1MHz SPI serial interface for communicating with all registers (MPU-6000 only)
- 20MHz SPI serial interface for reading sensor and interrupt registers (MPU-6000 only)

6 Electrical Characteristics

6.1 Gyroscope Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		°/s	
	FS_SEL=1		±500		°/s	
	FS_SEL=2		±1000		°/s	
	FS_SEL=3		±2000		°/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)	
	FS_SEL=1		65.5		LSB/(°/s)	
	FS_SEL=2		32.8		LSB/(°/s)	
	FS_SEL=3		16.4		LSB/(°/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	
GYROSCOPE ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	25°C		±20		°/s	
ZRO Variation Over Temperature	-40°C to +85°C		±20		°/s	
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		°/s	
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.5V		4		°/s	
Linear Acceleration Sensitivity	Static		0.1		°/s/*g*	
SELF-TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	1
GYROSCOPE NOISE PERFORMANCE	**FS_SEL=0**					
Total RMS Noise	DLPFCFG=2 (100Hz)		0.05		°/s-rms	
Low-frequency RMS noise	Bandwidth 1Hz to10Hz		0.033		°/s-rms	
Rate Noise Spectral Density	At 10Hz		0.005		°/s/ √ Hz	
GYROSCOPE MECHANICAL FREQUENCIES						
X-Axis		30	33	36	kHz	
Y-Axis		27	30	33	kHz	
Z-Axis		24	27	30	kHz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		256	Hz	
OUTPUT DATA RATE						
	Programmable	4		8,000	Hz	
GYROSCOPE START-UP TIME	**DLPFCFG=0**					
ZRO Settling (from power-on)	to ±1°/s of Final		30		ms	

1. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

6.2 Accelerometer Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	
ZERO-G OUTPUT						
Initial Calibration Tolerance	X and Y axes		±50		mg	1
	Z axis		±80		mg	
Zero-G Level Change vs. Temperature	X and Y axes, 0°C to +70°C		±35			
	Z axis, 0°C to +70°C		±60		mg	
SELF TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	2
NOISE PERFORMANCE						
Power Spectral Density	@10Hz, AFS_SEL=0 & ODR=1kHz		400		µg/ √Hz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		260	Hz	
OUTPUT DATA RATE						
	Programmable Range	4		1,000	Hz	
INTELLIGENCE FUNCTION INCREMENT			32		mg/LSB	

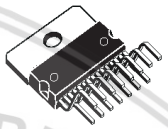
1. Typical zero-g initial calibration tolerance value after MSL3 preconditioning
2. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

3. Datasheet *Driver* L298N

L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15

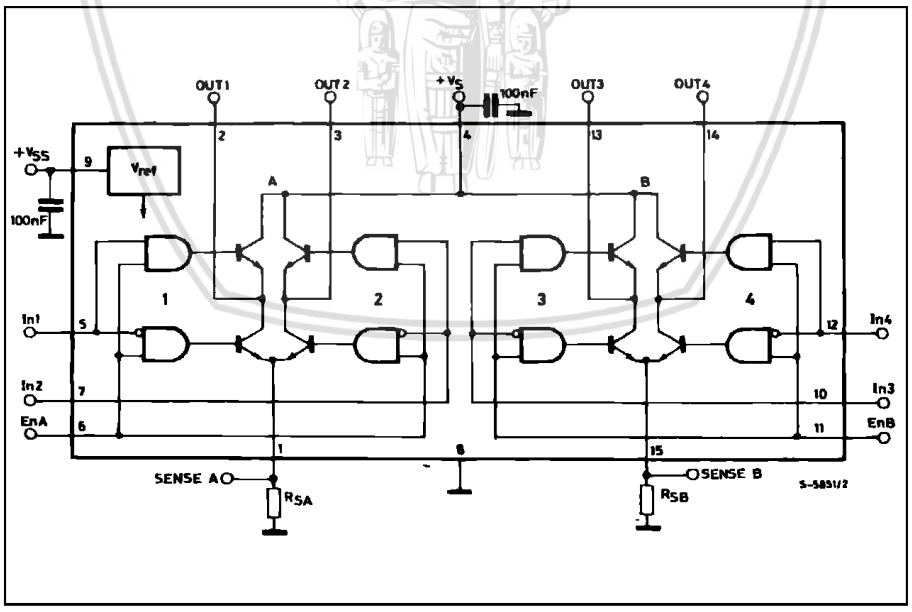
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

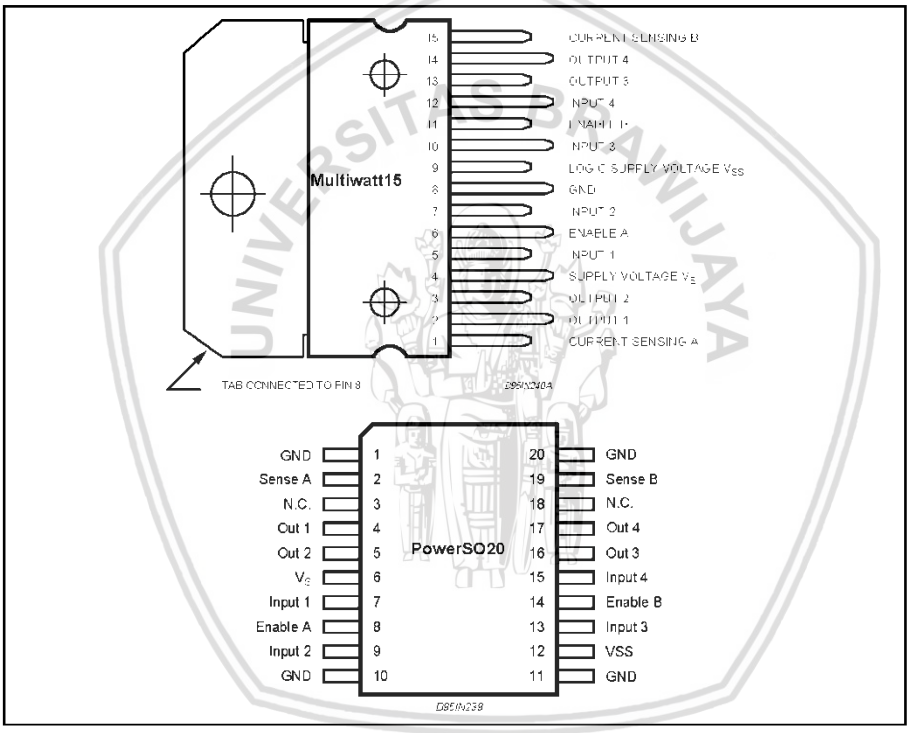
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_i,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

PIN CONNECTIONS (top view)



THERMAL DATA

Symbol	Parameter		PowerSO20	Multiwatt15	Unit
$R_{th\ j\text{-}case}$	Thermal Resistance Junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	13 (*)	35	°C/W

(*) Mounted on aluminum substrate

PIN FUNCTIONS (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3;18	N.C.	Not Connected

ELECTRICAL CHARACTERISTICS (V_S = 42V; V_{SS} = 5V, T_j = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 4)	Operative Condition	V_{IH} +2.5		46	V
V_{SS}	Logic Supply Voltage (pin 9)		4.5	5	7	V
I_S	Quiescent Supply Current (pin 4)	V_{en} = H; I_L = 0 V_i = L V_i = H		13 50	22 70	mA mA
		V_{en} = L V_i = X			4	mA
I_{SS}	Quiescent Current from V_{SS} (pin 9)	V_{en} = H; I_L = 0 V_i = L V_i = H		24 7	36 12	mA mA
		V_{en} = L V_i = X			6	mA
V_{il}	Input Low Voltage (pins 5, 7, 10, 12)		–0.3		1.5	V
V_{ih}	Input High Voltage (pins 5, 7, 10, 12)		2.3		VSS	V
I_{il}	Low Voltage Input Current (pins 5, 7, 10, 12)	V_i = L			–10	µA
I_{iH}	High Voltage Input Current (pins 5, 7, 10, 12)	Vi = H ≤ V_{SS} –0.6V		30	100	µA
V_{en} = L	Enable Low Voltage (pins 6, 11)		–0.3		1.5	V
V_{en} = H	Enable High Voltage (pins 6, 11)		2.3		V_{SS}	V
I_{en} = L	Low Voltage Enable Current (pins 6, 11)	V_{en} = L			–10	µA
I_{en} = H	High Voltage Enable Current (pins 6, 11)	V_{en} = H ≤ V_{SS} –0.6V		30	100	µA
$V_{CEsat(H)}$	Source Saturation Voltage	I_L = 1A I_L = 2A	0.95	1.35 2	1.7 2.7	V V
$V_{CEsat(L)}$	Sink Saturation Voltage	I_L = 1A (5) I_L = 2A (5)	0.85	1.2 1.7	1.6 2.3	V V
V_{CEsat}	Total Drop	I_L = 1A (5) I_L = 2A (5)	1.80		3.2 4.9	V V
V_{sens}	Sensing Voltage (pins 1, 15)		–1 (1)		2	V

ELECTRICAL CHARACTERISTICS (continued)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T_1 (V_i)	Source Current Turn-off Delay	0.5 V_i to 0.9 I_L (2); (4)		1.5		µs
T_2 (V_i)	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		0.2		µs
T_3 (V_i)	Source Current Turn-on Delay	0.5 V_i to 0.1 I_L (2); (4)		2		µs
T_4 (V_i)	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.7		µs
T_5 (V_i)	Sink Current Turn-off Delay	0.5 V_i to 0.9 I_L (3); (4)		0.7		µs
T_6 (V_i)	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.25		µs
T_7 (V_i)	Sink Current Turn-on Delay	0.5 V_i to 0.9 I_L (3); (4)		1.6		µs
T_8 (V_i)	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.2		µs
fc (V_i)	Commutation Frequency	I_L = 2A		25	40	KHz
T_1 (V_{en})	Source Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (2); (4)		3		µs
T_2 (V_{en})	Source Current Fall Time	0.9 I_L to 0.1 I_L (2); (4)		1		µs
T_3 (V_{en})	Source Current Turn-on Delay	0.5 V_{en} to 0.1 I_L (2); (4)		0.3		µs
T_4 (V_{en})	Source Current Rise Time	0.1 I_L to 0.9 I_L (2); (4)		0.4		µs
T_5 (V_{en})	Sink Current Turn-off Delay	0.5 V_{en} to 0.9 I_L (3); (4)		2.2		µs
T_6 (V_{en})	Sink Current Fall Time	0.9 I_L to 0.1 I_L (3); (4)		0.35		µs
T_7 (V_{en})	Sink Current Turn-on Delay	0.5 V_{en} to 0.9 I_L (3); (4)		0.25		µs
T_8 (V_{en})	Sink Current Rise Time	0.1 I_L to 0.9 I_L (3); (4)		0.1		µs

1) 1)Sensing voltage can be –1 V for t ≤ 50 µsec; in steady state V_{sens} min ≥ –0.5 V.
2) See fig. 2.
3) See fig. 4.
4) The load must be a pure resistor.

Figure 1 : Typical Saturation Voltage vs. Output Current.

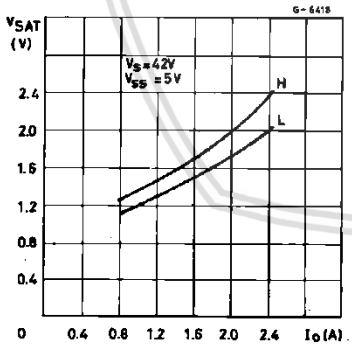
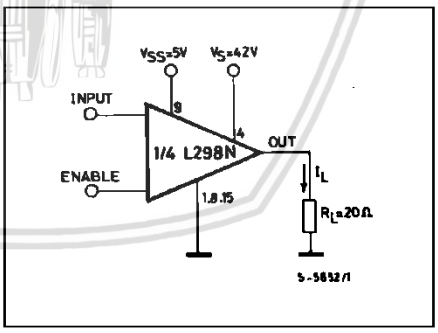


Figure 2 : Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H
For ENABLE Switching, set IN = H

Figure 5 : Sink Current Delay Times vs. Input 0 V Enable Switching.

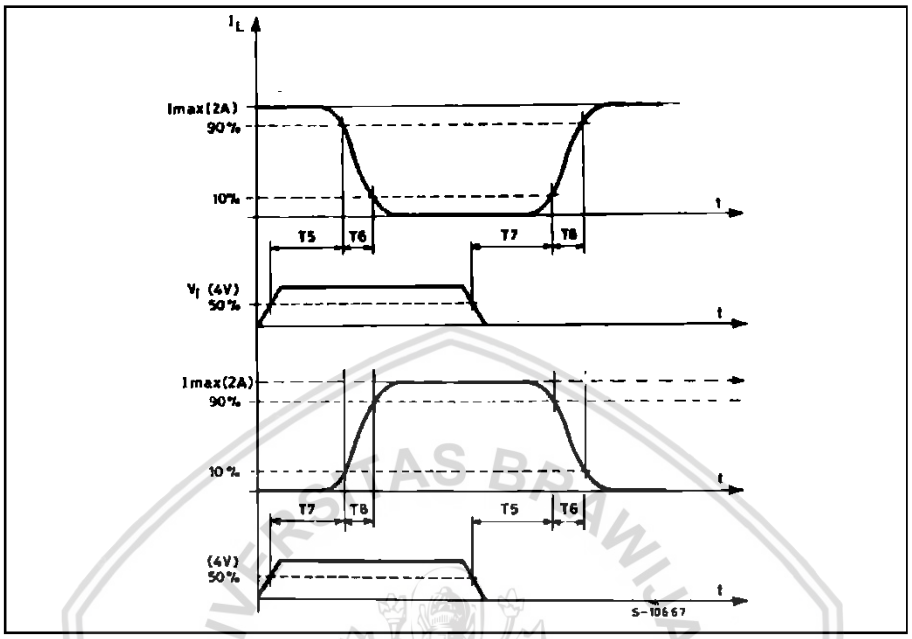
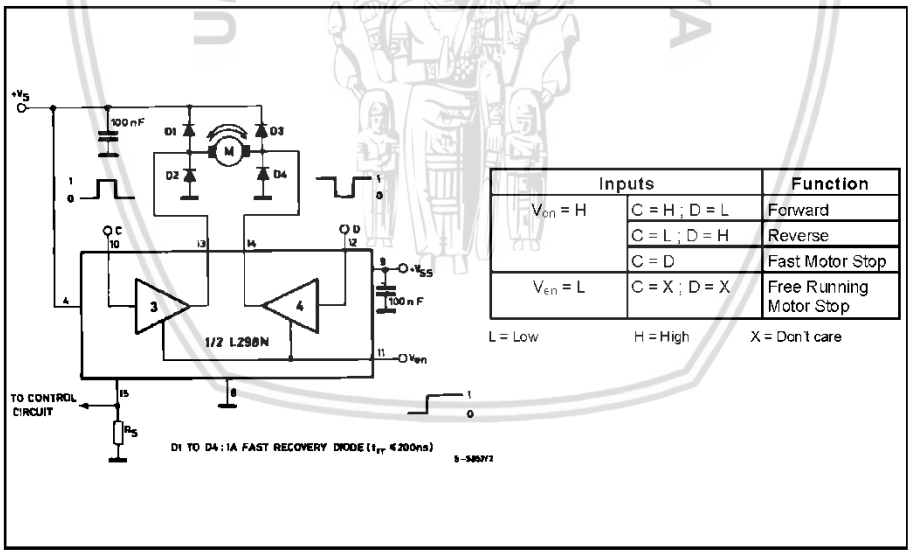


Figure 6 : Bidirectional DC Motor Control.



Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

This solution can drive until 3 Amps In DC operation and until 3.5 Amps of a repetitive peak current.

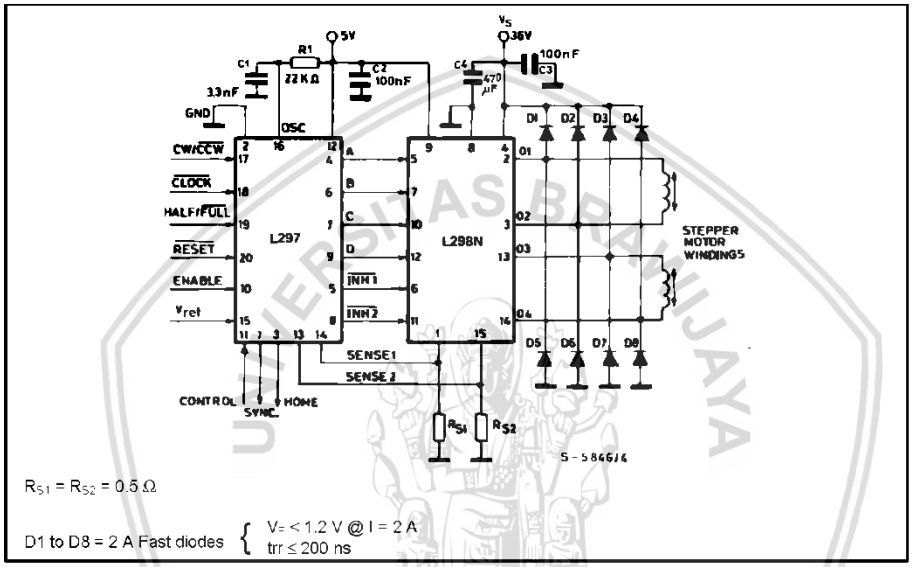
On Fig 8 it is shown the driving of a two phase bipolar stepper motor ; the needed signals to drive the inputs of the L298 are generated, in this example, from the IC L297.

Fig 9 shows an example of P.C.B. designed for the application of Fig 8.

Fig 10 shows a second two phase bipolar stepper motor control circuit where the current is controlled by the I.C. L6506.

Figure 8 : Two Phase Bipolar Stepper Motor Circuit.

This circuit drives bipolar stepper motors with winding currents up to 2 A. The diodes are fast 2 A types.



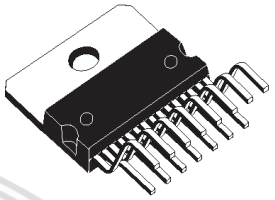
$R_{S1} = R_{S2} = 0.5\ \Omega$

D1 to D8 = 2 A Fast diodes { $V_F < 1.2$ V @ I = 2 A
$t_{rr} \leq 200$ ns

L298

DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			5			0.197
B			2.65			0.104
C			1.6			0.063
D		1			0.039	
E	0.49		0.55	0.019		0.022
F	0.66		0.75	0.026		0.030
G	1.02	1.27	1.52	0.040	0.050	0.060
G1	17.53	17.78	18.03	0.690	0.700	0.710
H1	19.6			0.772		
H2			20.2			0.795
L	21.9	22.2	22.5	0.862	0.874	0.886
L1	21.7	22.1	22.5	0.854	0.870	0.886
L2	17.65		18.1	0.695		0.713
L3	17.25	17.5	17.75	0.679	0.689	0.699
L4	10.3	10.7	10.9	0.406	0.421	0.429
L7	2.65		2.9	0.104		0.114
M	4.25	4.55	4.85	0.167	0.179	0.191
M1	4.63	5.08	5.53	0.182	0.200	0.218
S	1.9		2.6	0.075		0.102
S1	1.9		2.6	0.075		0.102
Dia1	3.65		3.85	0.144		0.152

OUTLINE AND MECHANICAL DATA



Multiwatt15 V