TUGAS AKHIR – TF141581

RANCANG BANGUN PENGENDALIAN KECEPATAN PROTOTIPE MOBIL OTONOM ACEPITS 1.0 DENGAN METODE PID BERBASIS IOT (*INTERNET OF THINGS*)

MUHAMMAD YUSRIL IZZA
NRP. 02311440000101

Dosen Pembimbing
Andi Rahmadiansah

PROGRAM STUDI S1 TEKNIK FISIKA
DEPARTEMEN TEKNIK FISIKA
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT – **TF141581**

DESIGN AND CONSTRUCTION OF A SPEED CONTROL PROTOTYPE AUTONOMOUS CAR ACEPITS 1.0 WITH PID METHOD BASED IOT (INTERNET OF THINGS)

MUHAMMAD YUSRIL IZZA
NRP. 02311440000101

Supervisor
Andi Rahmadiansah

STUDY PROGRAM S1 ENGINEERING PHYSICS
DEPARTMENT OF ENGINEERING PHYSICS
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya **2018**

PERNYATAAN BEBAS PLAGIARISME

Saya yang bertanda tangan di bawah ini:

Nama	: Muhammad Yusril Izza
NRP	: 02311440000101
Departemen/Prodi	: Teknik Fisika/ S-1 Teknik Fisika
Fakultas	: Fakultas Teknologi Industri
Perguruan Tinggi	: Institut Teknologi Sepuluh Nopember

Dengan ini menyatakan bahwa Tugas Akhir saya yang berjudul "Rancang Bangun Pengendalian Kecepatan Prototipe Mobil Otonom ACEPITS 1.0 Dengan Metode PID Berbasis IoT (*Internet of Things*)" adalah benar karya saya sendiri (modifikasi dari pihak lain dengan menyebutkan sumber) dan bukan plagiat dari karya orang lain. Apabila dikemudian hari terbukti terdapat plagiat pada Tugas Akhir ini maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku.

Demikian pernyataan ini saya buat dengan penuh tanggung jawab.

Surabaya, 26 Juli 2018
Yang membuat pernyataan,

Muhammad Yusril Izza
NRP. 02311440000101

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

RANCANG BANGUN PENGENDALIAN KECEPATAN PROTOTIPE MOBIL OTONOM ACEPITS 1.0 DENGAN METODE PID BERBASIS IOT (*INTERNET OF THINGS*)

TUGAS AKHIR

Oleh :
Muhammad Yusril Izza
NRP : 02311440000101

Surabaya, 26 Juli 2018
Mengetahui
Dosen Pembimbing

Andi Rahmadiansah S.T, M.T.
NIPN. 19790517 200312 1 002

Menyetujui,
Kepala Departemen Teknik Fisika FTI-ITS

Agus Muhamad Hatta, ST, Msi, Ph.D
NIPN. 19780902 200312 1 002

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN II

RANCANG BANGUN PENGENDALIAN KECEPATAN PROTOTIPE MOBIL OTONOM ACEPITS 1.0 DENGAN METODE PID BERBASIS IOT (*INTERNET OF THINGS*)

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
pada
Bidang Studi Rekayasa Instrumentasi dan Kontrol
Program Studi S-1 Departemen Teknik Fisika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Oleh :

MUHAMMAD YUSRIL IZZA
NRP. 02311440000101

Disetujui oleh Tim Penguji Tugas Akhir :

1. Andi Rahmadiansah, ST, MT .....(Pembimbing I)
2. Dr. Suyanto, ST, MT .....(Ketua Penguji)
3. Prof. Dr. Ir. Aulia Siti Aisjah, MT .....(Penguji I)

SURABAYA
JULI, 2018

Halaman ini sengaja dikosongkan

RANCANG BANGUN PENGENDALIAN KECEPATAN PROTOTIPE MOBIL OTONOM ACEPITS 1.0 DENGAN METODE PID BERBASIS IOT *(INTERNET OF THINGS)*

Nama Mahasiswa	**: Muhammad Yusril Izza**
NRP	**: 02311440000101**
Departemen	**: Teknik Fisika FTI-ITS**
Dosen Pembimbing	**: Andi Rahmadiansah S.T, M.T.**

Abstrak

Autonomous car merupakan sebuah kendaraan yang dapat bergerak sendiri tanpa adanya seorang sopir atau pengemudi mobil. Berbagai macam variabel yang ada pada *autonomous car* salah satunya adalah kecepatan. Kecepatan merupakan hal yang penting yang ada pada *autonomous car*, mengingat kecepatan adalah kendali utama agar *autonomous car* dapat bergerak sesuai dengan *setpoint* yang telah ditentukan, akan tetapi perlu adanya sistem monitoring untuk kendali kecepatan yang dilakukan oleh manusia agar tidak terjadi hal yang tidak diinginkan seperti kecepatan pada *autonomous car* melebihi *setpoint* atau batas kecepatan yang sudah ditentukan. Penelitian ini dilakukan dengan suatu rancang bangun kendali kecepatan pada mobil otonom ACEPITS 1.0 (*Autonomous Car Engineering Physics* ITS) dengan menggunakan metode PID. Pengujian dilakukan pada sistem kendali kecepatan *autonomous car* yang menghasilkan parameter nilai PID dengan kp, ki dan kd secara berturut-turut adalah sebesar 9,33, 0,26 dan 0,07 untuk pengujian kecepatan rata-rata pada motor BLDC dengan beban mobil otonom ACEPITS sebesar 78,09 rpm. *Autonomous car* dapat dikendalikan dengan menggunakan *remote* atau jarak jauh maka dibutuhkan suatu monitoring yang ditransmisikan dengan menggunakan sistem berbasis IoT (*Internet of Things)*. Fitur pada rancang bangun untuk sistem IoT diperlukan komponen yang dibutuhkan yaitu mikrokontroller arduino mega 2560 dan modul bluetooth hc-05. Sistem monitoring digunakan untuk aplikasi pada *smatphone* android yang dibuat menggunakan *software* android studio didapatkan nilai *delay* normal sebesar 50 ms. Jangkauan jarak paling jauh yang dapat diterima hingga 1300 cm dengan nilai *delay* sebesar 1000 ms.

Kata kunci: *autonomous car*, kecepatan, sistem monitoring

Halaman ini sengaja dikosongkan

DESIGN AND CONSTRUCTION OF A SPEED CONTROL PROTOTYPE AUTONOMOUS CAR ACEPITS 1.0 WITH PID METHOD BASED IOT (INTERNET OF THINGS)

Name	**: Muhammad Yusril Izza**
NRP	**: 02311440000101**
Departement	**: Teknik Fisika FTI-ITS**
Supervisor	**: Andi Rahmadiansah S.T, M.T.**

Abstract

Autonomous car is a vehicle that can move itself without a driver or driver of the car. Various kinds of variables that exist in the autonomous car one of them is the speed. Speed is an important thing in autonomous car, considering the speed is the main control for autonomous car can move in accordance with the setpoint that has been determined, but the need for a monitoring system for speed control done by humans in order to avoid unwanted things such as speed the autonomous car exceeds the specified setpoint or speed limit. This study was conducted with a design of speed control on autonomous car Engineering Physics ITS by ACEPITS 1.0 (Autonomous Car Engineering Physics ITS). The test was performed on the autonomous car speed control system which resulted the parameter of PID value with kp, ki and kd respectively were 9,33, 0,26 and 0,07 for mean speed test on BLDC motor with automotive car load ACEPITS of 78.09 rpm. Autonomous car can be controlled using remote or remote then it needs a monitoring that is transmitted by using IoT based system (Internet of Things). Features on the design for the IoT system required the required components of arduino mega 2560 microcontroller and hc-05 bluetooth module. Monitoring system used for applications on android smatphone made using android studio software obtained a normal delay value of 50 ms. The most acceptable distance reaches up to 1300 cm with a delay value of 1000 ms.

Keywords: autonomous car, speed, monitoring system

This page is left blank

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT, karena atas rahmat dan karunia-Nya sehingga penulis diberikan kesehatan, kemudahan, dan kelancaran dalam menyusun laporan Tugas Akhir yang berjudul:

"RANCANG BANGUN PENGENDALIAN KECEPATAN PROTOTIPE MOBIL OTONOM ACEPITS 1.0 DENGAN METODE PID BERBASIS IOT (*INTERNET OF THINGS*)"

Tugas akhir ini merupakan salah satu persyaratan akademik yang harus dipenuhi dalam Program Studi S-1 Departemen Teknik Fisika FTI-ITS. Penulis menyampaikan terima kasih yang sebesar-besarnya semua pihak yang telah membantu pengerjaan laporan Tugas Akhir ini kepada:

1. Bapak Agus Muhammad Hatta, S.T, M.Si, Ph.D, selaku Ketua Departemen Teknik Fisika FTI - ITS Surabaya.
2. Bapak Andi Rahmadiansah, S.T, M.T, selaku dosen pembimbing tugas akhir ini, yang selalu sabar memberikan dukungan, motivasi dan bimbingan serta arahan pada pengerjaan tugas akhir ini.
3. Bapak Dr. Suyanto, ST, MT., Ibu Prof. Dr. Ir. Aulia Siti Aisjah, MT. dan Bapak Dr. Ir. Syamsul Arifin, MT. selaku penguji yang telah memberikan segala masukan dan saran dalam pengerjaan tugas akhir ini.
4. Ibu Dr.-Ing. Doty Dewi Risanti, ST, MT, selaku dosen wali yang telah memberikan bimbingan dan arahan selama ini.
5. Kedua Orang Tua Penulis, Bapak Moch. Munief dan Ibu Sriyani Purwati, yang telah senantiasa memberikan dukungan, motivasi dan doa yang tiada hentinya.
6. Deni Agprianta, selaku teman satu tim yang telah berkerjasama dan memberikan dukungan penuh dalam pengerjaan tugas akhir ini.

7. Admin Laboratorium Simulasi dan Komputasi Teknik Fisika ITS yang telah membantu dan memberikan dukungan serta fasilitas yang disediakan untuk pengerjaan tugas akhir ini.
8. Teman-teman Angkatan 2014 Tree Fortress F49 atas kebersamaan dan kebahagiaan selama masa perkuliahaan ini.
9. Teman-teman Labkom E205 angkatan 2014 (Fathur, Munir, Deni, Okke, Ami, Niken, Ames dan Juniar) yang telah memberikan dukungan, kebahagiaan dan juga bersama-sama menyelesaikan tugas akhir.
10. Teman-teman Ndas Bocor Familia (Hafidh, Husain, Kayi, Ndut Faiz, Pandu, Iqbul, Ferdi, Mbah Rizky, Senko, Dio dan Haryo) sebagai teman terdekat yang selalu menemani serta memberikan dukungan kepada penulis selama ini.
11. Teman-teman Podo Mampir Squad yang juga memberikan dukungan motivasi kepada penulis.
12. Seluruh dosen, karyawan dan civitas akademik Teknik Fisika FTI-ITS atas segala bantuan dan kerjasamanya.
13. Semua pihak yang tidak dapat disebutkan satu persatu.

Dalam penyusunan laporan tugas akhir ini penulis menyadari bahwa mungkin masih ada kekurangan dalam laporan ini, sehingga kritik dan saran penulis dapat diterima. Semoga laporan ini dapat berguna dan bermanfaat bagi penulis dan semua pihak yang membacanya.

Surabaya, 26 Juli 2018

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Autonomous Car merupakan sebuah kendaraan yang dapat bergerak sendiri tanpa adanya seorang sopir atau pengemudi mobil, biasanya dapat disebut juga sebagai mobil robot otomatis. Dengan semakin berkembang pesatnya teknologi pada zaman sekarang ini, maka dari itu berbagai macam sistem otomatis diciptakan untuk dapat menggantikan beberapa peran manusia salah satunya pada kendaraan. *Autonomous car* menggunakan berbagai macam teknik untuk mendeteksi objek yang ada disekitarnya, seperti radar, sinar laser, GPS, odometri dan penglihatan komputer. Sistem kontrol yang canggih menginterpretasikan informasi sensor untuk mengidentifikasi jalur navigasi yang sesuai. *Autonomous Car* ini harus memiliki sistem kontrol yang mampu menganalisa data sensor untuk membedakan antara mobil yang berbeda dengan objek lain yang ada di jalan. Manfaat dari *autonomous car* meliputi berkurangnya mobilitas dan infrastruktur, peningkatan keamanan, peningkatan mobilitas, peningkatan pelanggan dan berkurangnya kejahatan. Lalu secara khusus penurunan yang paling signifikan yaitu pada kecelakaan lalu lintas. (Thrun, 2010)

Dengan berbagai macam kecanggihan sensor yang dibutuhkan pada *autonomous car* maka dibutuhkannya teknologi yang tinggi pula untuk mendukungnya. Untuk dapat mengendalikan *autonomous car* diperlukan suatu sistem monitoring yang dapat di kendalikan dalam jarak jauh. Sistem monitoring tersebut merupakan gabungan dari berbagai macam sensor yang telah dipasang pada *autonomous car* sehingga kendaraan dapat berjalan di jalan raya sesuai dengan lalu lintas yang ada. Maka dari itu dibutuhkan perangkat IoT (*Internet of Things*) agar *autonomous car* dapat dikendalikan atau di monitoring pada jarak jauh. Untuk sistem IoT sendiri dapat berkerja dengan menggabungkan sebuah perangkat mikrokontroler sebagaimana adalah suatu sistem yang fungsi kerjanya untuk

mengolah data digital menggunakan bahasa pemrograman. Sensor yang telah dipasang akan mengirimkan sinyal data untuk kemudian diproses dan ditransmisikan sebagai *output* data pada *interface* alat monitoring. (Gehrig & Stein, 1999)

Sistem IoT merupakan sebuah jaringan internet yang menyediakan, mengolah dan mentransfer informasi digital yang diperoleh dari peralatan sensor seperti identifikasi radio frekuensi (RFID), sensor infra merah, GPS, scanner dan smart meter. (Momoh, 2009) Sensor yang ada dalam jaringan IoT berfungsi untuk mendeteksi dan mengidentifikasi parameter-parameter sebuah peralatan melalui jaringan komunikasi kabel maupun nirkabel sehingga mampu untuk memperoleh data yang akurat serta proses kontrol secara real time. Penelitian yang berjudul "*Use of IoT Technology to Drive the Automotive Industry from Connected to Full Autonomous Vehicles"* yang dilakukan oleh X. Krasniqi dan E. Hajrizi pada tahun 2016 (Krasniqi & Hajrizi, 2016) telah memberikan revolusi baru terhadap dunia industri otomotif dengan adanya teknologi IoT. Dengan menggunakan *smartphone* yang sudah terhubung pada mobil otonom, kendaraan memiliki fitur yang dapat digunakan untuk mempercepat, mengerem dan membelokan tanpa adanya driver yang mengemudikan. (Ninan, Gangula, Alten, & Sniderman, 2015)

Kecepatan merupakan fitur utama pada mobil otonom sebagai penggerak agar mobil otonom dapat bergerak. Sistem kecepatan agar dapat bergerak sesuai dengan *setpoint* yang ditentukan, dibutuhkan suatu kontrol yang dapat mengendalikan kecepatan pada mobil otonom. Pengendalian PID merupakan sebuah kontroler dengan mekanisme umpan balik yang biasanya dipakai pada sistem kontrol industri. Sebuah kontroler PID secara kontinyu menghitung nilai kesalahan sebagai beda antara *setpoint* yang diinginkan dan variabel proses terukur. Kontroler mencoba untuk meminimalkan nilai kesalahan setiap waktu dengan pengaturan variabel kontrol. (Shamseldin & EL-Samahy, 2014)

Pada penelitian tugas akhir ini penulis merancang bangun kendali kecepatan pada prototipe mobil otonom ACEPITS 1.0 (*Autonomous Car Engineering Physics* ITS) dengan memanfaatkan teknologi sistem IoT. Untuk sistem kendali kecepatan menggunakan motor BLDC (*Brushless Direct Current Motor*) sebagai penggerak utama mobil otonom yang dikendalikan menggunakan metode PID untuk dapat memenuhi kriteria respon agar kecepatan motor dapat mencapai *setpoint* yang telah ditentukan. Sistem monitoring ini menggabungkan mikrokontroller yang digunakan dengan modul *bluetooth* kemudian akan diintegrasikan dengan menggunakan *software* progam pada *smartphone* android. Data yang akan ditransmisikan pada *interface smartphone* android berupa nilai kecepatan pada mobil otonom ACEPITS. Sehingga dengan data monitoring tersebut mobil otonom ACEPITS dapat dikendalikan dari jarak jauh. Penelitian ini diharapkan dapat diaplikasikan pada mobil otonom agar mobil otonom dapat cepat diproduksi terutama di Indonesia.

1.2 Perumusan Masalah

Adapun permasalahan yang didapatkan dari latar belakang diatas pada tugas akhir ini adalah sebagai berikut:

a. Bagaimana cara mengendalikan kecepatan dengan monitoring pada mobil otonom ACEPITS 1.0 menggunakan metode PID?
b. Bagaimana merancang sistem kendali jarak jauh menggunakan IoT *(Internet of Things)* pada mobil otonom ACEPITS 1.0?

1.3 Tujuan

Adapun tujuan yang ingin dicapai dari tugas akhir ini berdasarkan pada perumusan masalah adalah sebagai berikut:

a. Dapat mengendalikan kecepatan dengan monitoring pada mobil otonom ACEPITS 1.0 menggunakan metode PID.
b. Menghasilkan alat yang dapat digunakan untuk kendali jarak jauh menggunakan IoT *(Internet of Things)* pada mobil otonom ACEPITS 1.0.

1.4 Batasan Masalah

Adapun batasan masalah dari dilakukannya penelitian tugas akhir ini adalah mengenai rancang bangun sistem kendali kecepatan motor BLDC (*Brushless Direct Current Motor)* dengan metode PID pada mobil otonom ACEPITS 1.0 yang berbasis IoT (*Internet of Things*) yaitu menggunakan gabungan dari mikrokontroler yang akan diintegrasikan menggunakan modul *Bluetooth HC-05*, data yang sudah didapatkan tersebut berupa nilai kecepatan yang akan ditransmisikan pada aplikasi *smartphone* android untuk dimonitoring secara lokal.

BAB II
TINJAUAN PUSTAKA

2.1 *Autonomous Car*

Mobil otonom (juga dikenal sebagai mobil tanpa sopir, mobil penggerak sendiri, mobil robotik) dan kendaraan yang mampu merasakan lingkungannya dan menavigasi tanpa inputan dari manusia. Mobil otonom menggunakan berbagai teknik untuk mendeteksi sekelilingnya, seperti radar, sinar laser, GPS, odometri dan penglihatan komputer. Sistem kontrol yang canggih menginterpretasikan informasi sensorik untuk mengidentifikasi jalur navigasi yang sesuai, serta rintangan dan papan nama yang relevan. Mobil otonom harus memiliki sistem kontrol yang mampu menganalisa data sensorik untuk membedakan antara mobil yang berbeda di jalan. (Gehrig & Stein, 1999)

Waymo adalah perusahaan pengembangan mobil otonom dan anak perusahaan dari perusahaan induk Google, Alphabet Inc. Google telah mulai menguji proyek mobil *self-driving* pada tahun 2009. Alfabet menggambarkan Waymo sebagai "perusahaan teknologi self-driving dengan misi untuk membuatnya aman dan mudah bagi orang dan hal-hal untuk bergerak". (Hawkins, 2017) Perusahaan baru ini bekerja untuk membuat mobil *self-driving* yang tersedia untuk umum seperti pada Gambar 2.1.

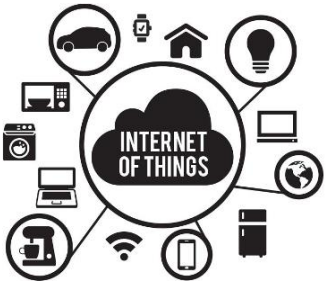
Gambar 2.1 Waymo Google Self-Driving Car Project (Hawkins, 2017)

Pada tahun 2012, pendiri Google Sergey Brin menyatakan bahwa mobil Google Self-Driving pada Gambar 2.1 akan tersedia untuk umum pada tahun 2017, (Tam, 2012) dan pada tahun 2014 jadwal ini diperbarui oleh direktur proyek Chris Urmson untuk menunjukkan kemungkinan rilis dari tahun 2017 sampai 2020. Google telah bermitra dengan pemasok termasuk Bosch, ZF Lenksysteme, LG, Continental, dan Roush, dan telah menghubungi produsen termasuk General Motors, Ford, Toyota (termasuk Lexus), Daimler dan Volkswagen.

2.2 IoT *(Internet of Things*)

Merupakan jaringan perangkat fisik, kendaraan, peralatan rumah tangga dan barang-barang lainnya yang disematkan dengan peralatan elektronik, perangkat lunak, sensor, aktuator, dan konektivitas jaringan yang memungkinkan benda-benda ini terhubung dan bertukar data. (Brown, 2016) Peran penting utama IoT dalam beberapa tahun terakhir adalah pertumbuhan perangkat yang terkendali dan dikendalikan oleh internet. Beragam aplikasi untuk teknologi IoT berarti spesifikasinya bisa sangat berbeda dari satu perangkat ke perangkat berikutnya namun ada karakteristik dasar yang dimiliki oleh kebanyakan pengguna. Sistem monitoring melalui internet menghubungkan, mengkontrol, dan secara langsung melalui Internet. (Zhang, Li, & Bhatt, 2010) Pemantauan harus memberikan informasi yang dibutuhkan oleh pengguna, informasi harus kompak dengan konsep SMART (*Specific, Measurable, Achievable, Realistic, Time-based*) spesifik, terukur, dapat diperoleh, relevan, dalam rentang waktu. (Doran, 1981)

Banyak yang memanfaatkan *realtime* monitoring ini secara wireline seperti LCD dan tidak sedikit pula yang memanfaatkan nya secara wireless seperti bluetooth, text message, dan juga web. IoT dapat digambarkan sebagai koneksi dari perangkat seperti ponsel pintar, komputer pribadi, sensor, dan aktuator melalui jaringan internet, perangkat yang terhubung bisa menghasilkan informasi yang dapat digunakan oleh manusia atau sistem lainnya seperti pada Gambar 2.2.



Gambar 2.2 Ilustrasi dari *Internet of Things* (Wetherill, 2015)

2.3 Pengendalian PID

Kontroler PID (*Proportional-Integral-Deriative Controller)* merupakan kontroler mekanisme umpan balik yang biasanya dipakai pada sistem kontrol industri. Sebuah kontroler PID secara kontinyu menghitung nilai kesalahan sebagai beda antara *setpoint* yang diinginkan dan variabel proses terukur. Kontroler mencoba untuk meminimalkan nilai kesalahan setiap waktu dengan penyetelan variabel kontrol, seperti posisi keran kontrol, damper, atau daya. Dengan Kp, Ki dan Kd, semuanya positif, menandakan koefisien untuk *term proporsional*, *integral*, dan *derivatif*, secara berurutan (atau *P*, *I*, dan *D*) pada model ini. (M., 2009)

- **P** bertanggung jawab untuk nilai kesalahan saat ini. Contohnya, jika nilai kesalahan besar dan positif, maka keluaran kontrol juga besar dan positif.
- **I** bertanggung jawab untuk nilai kesalahan sebelumnya. Contoh, jika keluaran saat ini kurang besar, maka kesalahan akan terakumulasi terus menerus, dan kontroler akan merespon dengan keluaran lebih tinggi.
- **D** bertanggung jawab untuk kemungkinan nilai kesalahan mendatang, berdasarkan pada rate perubahan tiap waktu.

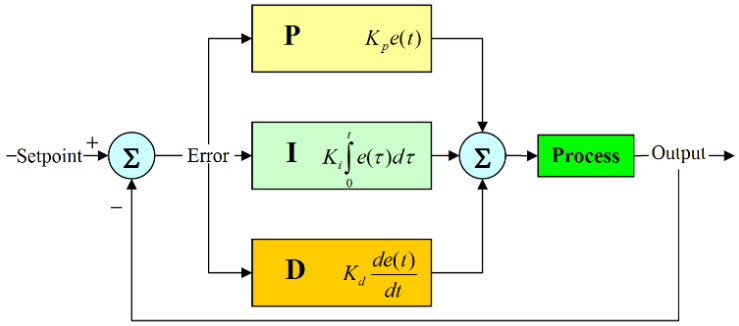
Karena kontroler PID hanya mengandalkan variabel proses terukur, bukan pengetahuan mengenai prosesnya, maka dapat secara luas digunakan. Dengan penyesuaian (*tuning*) ketiga parameter model, kontroler PID dapat memenuhi kebutuhan

proses. Respon kontroler dapat dijelaskan dengan bagaimana responnya terhadap kesalahan, besarnya *overshoot* dari *setpoint*, dan derajat osilasi sistem. Penggunaan algoritm PID tidak menjamin kontrol optimum sistem atau bahkan kestabilannya.

Beberapa aplikasi mungkin hanya menggunakan satu atau dua term untuk memberikan kontrol sistem yang sesuai. Hal ini dapat dicapai dengan mengontrol parameter yang lain menjadi nol. Kontroler PID dapat menjadi kontroler PI, PD, P atau I tergantung aksi apa yang digunakan. Kontroler PI biasanya adalah kontroler paling umum. Untuk sistem waktu diskrit, sering digunakan PSD atau *Proportional-Summation-Difference.* (Veselý & Rosinová, 2011)

Setiap kekurangan dan kelebihan dari masing-masing pengontrol P, I dan D dapat saling menutupi dengan menggabungkan ketiganya secara paralel menjadi pengontrol proporsional tambah integral tambah diferensial (pengontrol PID). Elemen - elemen pengontrol P, I dan D masing-masing secara keseluruhan bertujuan:

- mempercepat reaksi sebuah sistem mencapai set point-nya.
- menghilangkan offset.
- menghasilkan perubahan awal yang besar dan mengurangi overshoot.



Gambar 2.3 Blok Diagram dari Kontroler PID (Achzab, 2014)

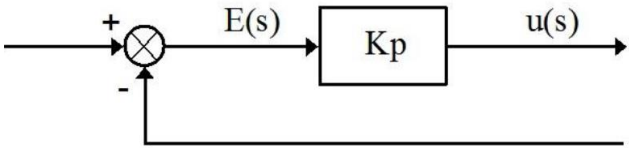
Gambar 2.3 merupakan skema kontrol PID dinamakan dari ketiga term pengendalinya, yang kemudian dijumlahkan menjadi variabel manipulasi. Term proporsional, integral, dan derivatif dijumlahkan untuk menghitung keluaran kontroler PID, bentuk akhir dari algoritme PID adalah pada Persamaan (2.1).

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \qquad (2.1)$$

Dimana, dengan mendefinisikan u(t) sebagai keluaran kontroler, K_p sebagai *gain* Proporsional, K_i sebagai *gain* Integral, K_d sebagai *gain* Derivatif, *e* sebagai *error*, *t* sebagai waktu dan τ sebagai variabel Integrasi.

a. Pengendali Proposional (P)

Kontroler proposional memiliki 2 parameter, yaitu *proportional band* dan konstanta proposional. Daerah kerja kontroler efektif dicerminkan oleh *proportional band* sedangkan konstanta proposional menunjukkan nilai faktor penguatan terhadap sinyal *error*, Kp. Hubungan antara *proportional band* (PB) dengan konstanta proposional (Kp) dapat ditunjukkan pada diagram blok pengendali proposional seperti pada Gambar 2.4.



Gambar 2.4 Diagram Blok Pengendali Proposional (Ogata, 2010)

Dari diagram blok pengendali proposional antara PB dengan Kp dapat ditunjukkan dengan Persamaan (2.2).

$$PB = \frac{100\%}{Kp} \quad (2.2)$$

Dimana, PB merupakan *Proportional Band* dan Kp sebagai *gain Process* dari diagram blok pengendalian.

Kontrol *proportional* sebanding dengan besarnya *input*. Bentuk *transfer function* dari kontrol *proportional* seperti yang ditunjukkan pada Persamaan (2.3).

$$U = K_c \ e \quad (2.3)$$

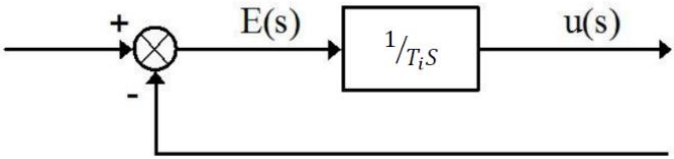
Dimana, U merupakan keluaran dari *transfer function* kontrol P, Kc sebagai *gain* Proporsional dan *e* sebagai *error.*

Penggunaan mode kontrol pengendali proposional harus memperhatikan hal-hal berikut :

- Jika nilai Kp kecil, mode kontrol proposional hanya mampu melakukan koreksi kesalahan yang kecil, sehingga akan menghasilkan respon sistem yang lambat.
- Jika nilai Kp diaikkan, respon sistem menunjukkan semakin cepat mencapai keadaaan stabilnya.
- Namun jika nilai Kp diperbesar sehingg mencapai harga yang berlebihan akan mengakibatkan sistem bekerja tidak stabil, atau respon sistem akan berosilasi.

b. Pengendali Integral (I)

Kontroler integral memiliki karakteristik seperti halnya sebuah integral. Keluaran kontroler dipengaruhi oleh perubahan yang sebanding dengan nilai sinyal *error*. Keluaran kontroler ini merupakan jumlahan yang terus menerus dari perubahan masukannya. Jika sinyal *error* tidak mengalami perubahan, keluaran akan menjaga keadaan seperti sebelum terjadinya perubahan masukan pada Gambar 2.5.



Gambar 2.5 Diagram Blok Pengendali Integral (Ogata, 2010)

Bentuk *transfer function* dari kontrol integral dapat ditunjukkan pada Persamaan (2.4).

$$U = \frac{1}{T_i} K_c \int e \, dt \qquad (2.4)$$

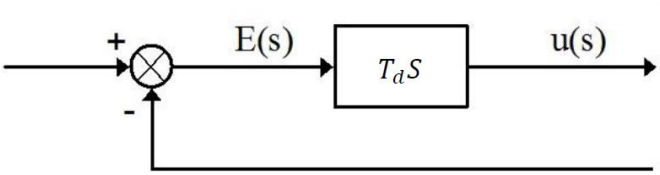
Dimana, U merupakan keluaran dari *transfer function* kontrol Integral, *e* sebagai e*rror* (input dari kontrol) dan Kc sebagai *gain* dari Kontroler.

Kontroler integral mempunyai beberapa karakteristik sebagai berikut:

- Keluaran kontroler membutuhkan selang waktu tertentu, sehingga kontroler integral cenderung memperlambat respon.
- Ketika sinyal *error* berharga nol, keluaran kontroler akan bertahan pada nilai sebelumnya.
- Jika sinyal kesalahan tidak berharga nol, keluaran akan menunjukkan kenaikan atau penurunan yang dipengaruhi oleh besarnya sinyal *error* dan nilai Ki.
- Konstanta integral Ki berharga besar. *offset* akan cepat hilang. Saat nilai Ki besar akan berakibat peningkatan osilasi dari sinyal keluaran *controller*.

c. Pengendali Derivatif (D)

Keluaran *controller differensial* memiliki sifat seperti halnya suatu operasi derivatif. Perubahan yang mendadak pada masukan kontroler akan mengakibatkan perubahan yang sangat besar dan cepat seperti diagram blok pengendalian pada Gambar 2.6.



Gambar 2.6 Diagram Blok Pengendali Derivatif (Ogata, 2010)

Bentuk *transfer function* dari kontrol derivatif dapat ditunjukkan pada persamaan (2.3) berikut ini.

$$U = K_c \ T_d \ \frac{de}{dt} \tag{2.5}$$

Dimana, U merupakan keluaran dari *transfer function* kontrol derivatif, Kc sebagai *gain*, *e* sebagai *error* dan Td sebagai derivatif *time.*

Kontroler derivatif mempunyai beberapa karakteristik adalah sebagai berikut :

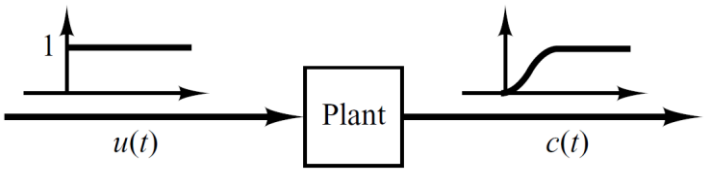
- Kontroler derivatif tidak dapat menghasilkan keluaran bila tidak ada perubahan atau *error* sebagai sinyal kesalahan untuk masukannya.
- Jika sinyal *error* berubah terhadap waktu, maka keluaran yang dihasilkan kontroler tergantung pada nilai Td dan laju perubahan sinyal kesalahan.
- Kontroler derivatif mempunyai karakter untuk mendahului, sehingga kontroler ini dapat menghasilkan koreksi yang signifikan sebelum pembangkit error menjadi sangat besar. Jadi kontroler derivatif dapat mengantisipasi pembangkit *error,* memberikan aksi yang bersifat korektif, dan cenderung meningkatkan stabilitas sistem.

Tabel 2.1 Pengaruh Pengendalian PID (Ogata, 2010)

Mode Kontrol	Rise Time	Overshoot	Settling Time	Error Steady State
Proporsional	Menurunkan	Meningkatkan	Perubahan Kecil	Menurunkan/ Mengurangi
Integral	Menurunkan	Meningkatkan	Meningkatkan	Mengeliminasi
Derivatif	Perubahan Kecil	Menurunkan	Menurunkan	Perubahan Kecil

Karakteristik pengontrol PID sangat dipengaruhi oleh kontribusi besar dari ketiga parameter P, I dan D. Penyetelan konstanta Kp, Ki dan Kd akan mengakibatkan penonjolan sifat dari masing- masing elemen. Satu atau dua dari ketiga konstanta tersebut dapat diatur lebih menonjol dibandingkan yang lain. Konstanta yang menonjol itulah akan memberikan kontribusi pengaruh pada respon sistem secara keseluruhan. Pengaruh dari setiap pengontrol Proporsional, Integral dan Derivatif pada sistem dapat ditunjukkan pada Tabel 2.1.

Mode kontrol D tidak dapat digunakan pada *process variable* yang mengandung *noise* seperti pengendalian *level* dan *flow*, karena *noise* dan gelombang akan dideferensialkan menjadi pulsa-pulsa yang tidak beraturan. Akibatnya *control valve* terbuka dan tertutup secara tidak beraturan dan akan merusak suatu sistem. Selain itu, mode kontrol D tidak dapat mengeluarkan output bila tidak ada perubahan input. Sehingga, kontrol D tidak pernah dipakai sendirian. Unit kontrol D selalu digunakan dalam kombinasi dengan kontrol P dan I. (Ogata, 2010)

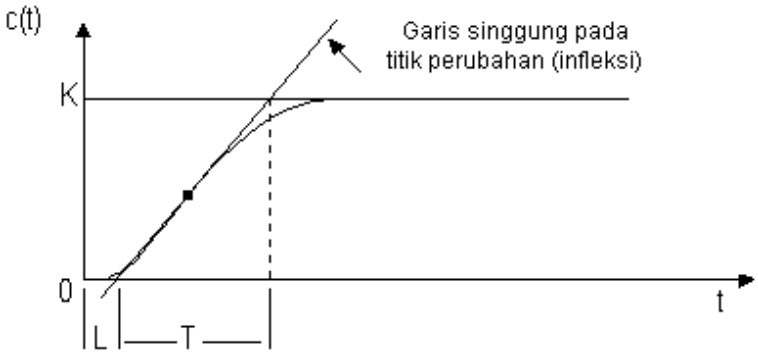


Gambar 2.7 Sistem *Open Loop* (Ogata, 2010)

Suatu sistem pengendalian terdapat proses *tuning* atau penyetelan alat agar didapatkan sistem dengan hasil respon yang stabil. Berbagai macam metode *tuning* telah ditemukan. Salah satunya adalah metode *Ziegler-Nichols* tipe I dengan menentukan parameter - parameter PID digunakan sistem *open loop*. Dengan dilakukannya metode *tuning* parameter PID dapat menganalisa respon yang dihasilkan yaitu nilai Kp, Ki dan Kd. Dari hasil percobaan dengan masukan *unit-step*, hasilnya akan terbentuk kurva berbentuk huruf S seperti pada Gambar 2.7.

Kurva bentuk S pada Gambar 2.7 memiliki karakteristik dengan 2 buah konstanta, yaitu waktu tunda (L) dan *time constant* (T). Kedua parameter tersebut diperoleh dengan menggambar garis tangensial pada titik infleksi kurva S seperi pada Gambar 2.8.

Gambar 2.8 kurva respon S diberikan suatu garis yang bersinggungan dengan garis kurva. Garis singgung itu akan memotong dengan sumbu absis dan garis maksimum. Perpotongan garis singgung dengan sumbu absis merupakan ukuran waktu mati, dan perpotongan dengan garis maksimum merupakan waktu tunda yang diukur dari titik waktu L. Setelah didapatkan nilai parameter L dan T untuk nilai – nilai Kp, Ti dan Td dapat dihitung menggunakan persamaan Ziegler-Nichols seperti yang ditunjukkan pada Tabel 2.2.



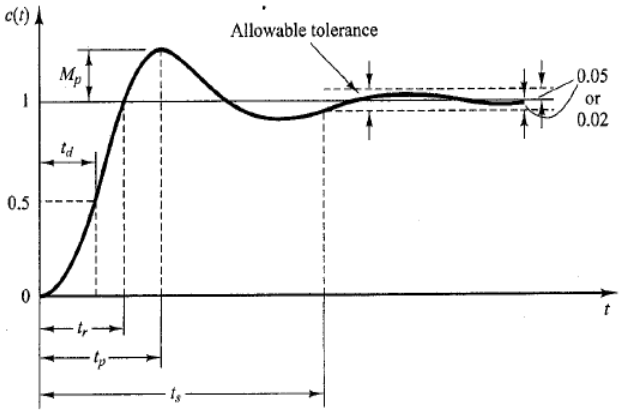
Gambar 2.8 Kurva Respon Berbentuk S (Ogata, 2010)

Tabel 2.2 Aturan PID *Tuning* Ziegler Nichols (Ogata, 2010)

Kontroller	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	2L	0.5L

2.4 Analisa Performansi Pengendalian

Analisa sistem pengendalian digunakan untuk menghasilkan respon system yang telah dirancang. Hasil dari bentuk analisa ini berupa nilai kualitatif. Jenis respon kontrol akan berbeda-beda berdasarkan orde dari system yang dikendalikan. Misalnya saja untuk system orde satu akan berbeda dengan system orde dua. Respon dinamik akan memiliki karakteristik yang berbeda berdasarkan jenis plant yang dikontrol. Sehingga respon dinamik setiap unit tergantung pada nilai masukan pada unit yang telah digunakan. Untuk jenis analisa respon dinamik dapat ditunjukkan pada Gambar 2.9.

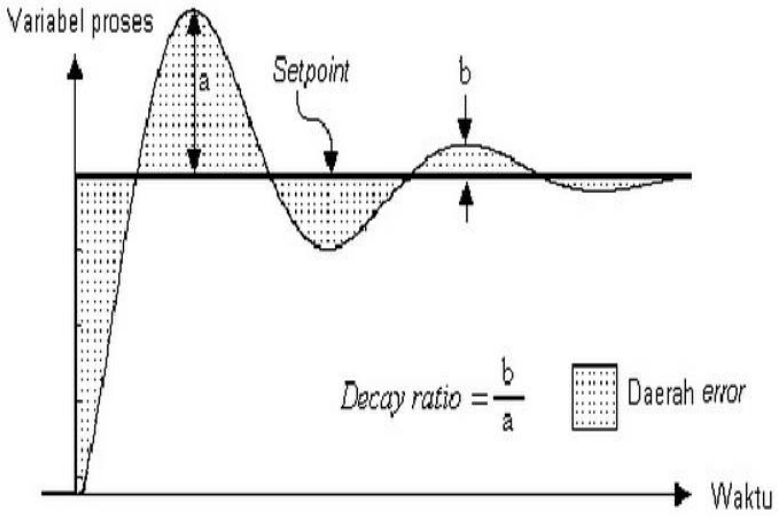


Gambar 2.9 Analisa Karakteristik Performansi Sistem (Ogata, 2010)

Karakteristik yang diperoleh pada Gambar 2.9 dapat dilakukan analisa performansi dari respon system dan untuk mengetahui nilai performansi dari nilai system maka perlu mendefinisikan nilai IAE *(Integral Absolute Error), maximum overshoot* dan *settling time*. Masing –masing dari parameter tersebut dapat dicari dengan cara sebagai berikut menganalisa respon system yang didapatkan dari hasil simulasi dengan cara sebagai berikut.

a. IAE (*Integral Absolute Error*)

IAE (*Integral absolute error*) merupakan kalkulasi dari nilai error dengan m,enjumlahkan setiap nilai error yang terjadi. Nilai IAE dapat direpresentasikan oleh Gambar 2.10.



Gambar 2.10 Penentuan nilai *Integral Absolute Error* (IAE) (Heriyanto, 2010)

Untuk menghitung nilai *Integral Absolute Error* dapat menggunakan Persamaan (2.6).

$$IAE = \int_0^{\infty} e(t)dt \qquad (2.6)$$

b. *Maximum Overshoot*

Nilai *maximum overshoot* adalah nilai puncak dari kurva respon yang diukur. Setiap perusahaan memiliki standarisasi tersendiri untuk nilai *maximum overshoot.* Dan untuk menentukan persamaan *maximum overshoot* dapat dihitung dengan menggunakan Persamaan (2.7).

$$MO = \frac{C(t_p) - C(\infty)}{C(\infty)}\ x\ 100\% \qquad (2.7)$$

c. *Settling Time*

Settling time adalah waktu yang dibutuhkan system untuk mencapai keadaan *set point.* Pada *settling time* terdapat presentase untuk mennentukan nilai *settling time* yaitu sebesar ±2% atau ±5% dari nilai *setpoint.*

d. Waktu Puncak

Waktu puncak (*peak time*) *tp* : adalah waktu yang diperlukan respon untuk mecapai puncak pertama *overshoot*.

e. Waktu Naik

Waktu naik *(rise time) tr* : adalah waktu yang diperlukan oleh respon untuk naik dari 10% menjadi 90%, 5% menjadi 95% atau 0% menjadi 100% dari nilai akhir yang digunakan.

f. Waktu Tunda

Waktu tunda (*delay time) td* : adalah waktu yang diperlukan oleh respon untuk mencapai setengah (50%) nilai akhir untuk waktu yang pertama.

Karakteristik respon suatu sitem pengendalian dicirikan oleh tanggapan transien terhadap masukan sinyal uji tangga satuan (*step*). Jika tanggapan terhadap masukan sinyal uji tangga satuan diketahui, maka secara matematis dapat dihitung tanggapan untuk sembarang masukan. Tanggapan transien suatu sistem pengendalian secara praktis selalu menunjukan osilasi teredam sebelum mencapai keadaan tunaknya. (Ogata, 2010)

2.5 Motor BLDC (*Brushless Direct Current Motor*)

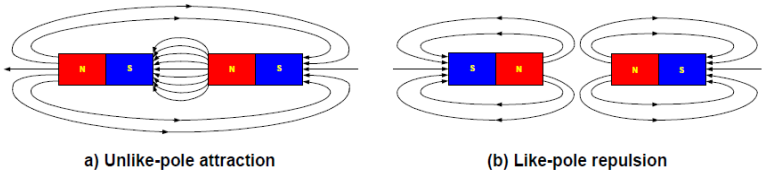
Motor listrik DC (motor BLDC) juga dikenal sebagai motor berganti secara elektronik (ECM, motor EC), atau motor DC sinkron, adalah motor sinkron yang digerakkan oleh listrik DC melalui inverter atau catu daya *switching* yang menghasilkan arus listrik AC ke menggerakkan setiap fase motor melalui pengontrol *loop* tertutup. Pengontrol memberikan pulsa arus ke gulungan motor yang mengontrol kecepatan dan torsi motor. (Zhao & Yu, 2011)

Konstruksi sistem motor *brushless* biasanya mirip dengan motor sinkron magnet permanen (PMSM), tetapi juga bisa menjadi motor keengganan beralih, atau motor induksi (*asynchronous*). Motor BLDC banyak digunakan dalam aplikasi termasuk peralatan, otomotif, *aerospace*, konsumen, peralatan dan instrumentasi industri medis dan otomatis. Motor BLDC secara elektrik dinyalakan oleh saklar daya. Dibandingkan dengan Motor DC atau motor induksi, motor BLDC memiliki banyak kelebihan antara lain adalah sebagai berikut.

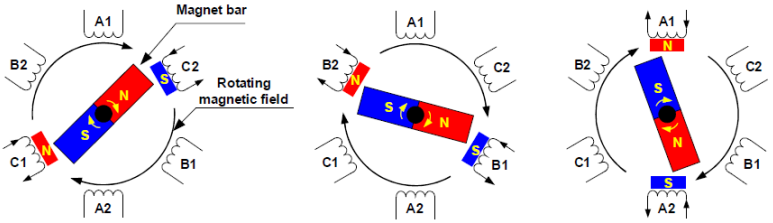
- Efisiensi dan keandalan yang lebih tinggi
- Menurunkan kebisingan
- Lebih kecil dan lebih ringan
- Respon dinamis yang lebih besar
- Kecepatan yang lebih baik dibandingkan karakteristik torsi
- Kecepatan dengan kisaran yang lebih tinggi

Gambar 2.11 *Motor Brushless* DC (Prasetyo B. , 2015)

Konsep dasar motor mengubah energy listrik menjadi mekanik menggunakan prinsip elektromagnetik seperti pada salah satu contoh menggunakan gaya magnetik. Kutub magnet menghasilkan garis-garis tak terlihat dari gaya magnet yang mengalir dari kutub utara ke kutub selatan seperti yang ditunjukkan pada Gambar 2.12. Ketika kutub magnet dari kutub berlawanan saling berhadapan, mereka menghasilkan gaya yang menarik, sementara kutub seperti menghasilkan gaya tolak. Teori operasional motor didasarkan pada daya tarik atau tolakan antara kutub magnet.



Gambar 2.12 Gaya Magnetik Motor BLDC (MPS, 2014)



Gambar 2.13 Rotasi Motor BLDC (MPS, 2014)

Gambar 2.13 menggunakan motor tiga fase, proses dimulai ketika arus mengalir melalui salah satu dari tiga gulungan stator dan menghasilkan kutub magnet yang menarik magnet permanen terdekat dari kutub yang berlawanan. Rotor akan bergerak jika arus bergeser ke belitan yang berdekatan. Mengisi daya setiap belitan secara berurutan akan menyebabkan rotor mengikuti bidang berputar. Torsi dalam contoh ini tergantung pada amplitudo saat ini dan jumlah belokan pada gulungan stator, kekuatan dan ukuran magnet permanen, celah udara antara rotor dan belitan, dan panjang lengan yang berputar.

Setiap motor BLDC memiliki dua bagian utama, rotor (bagian berputar) dan stator (bagian stasioner). Bagian penting lainnya dari motor adalah gulungan stator dan magnet rotor.

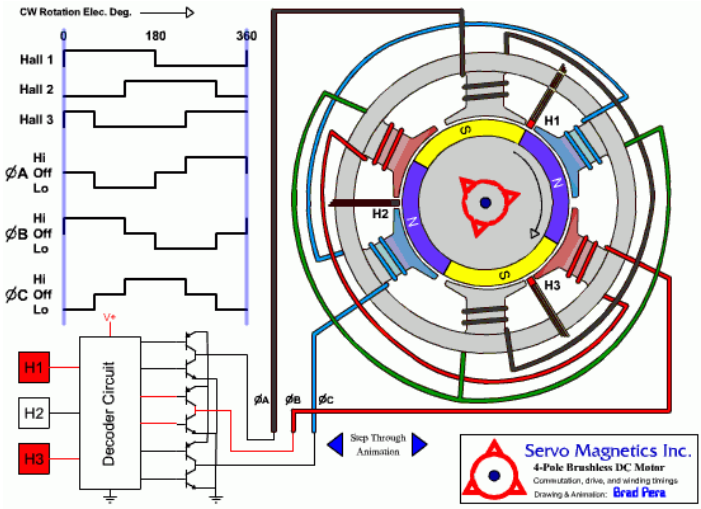
a. Rotor

Rotor adalah bagian pada motor yang berputar karena adanya gaya elektromagnetik dari stator, dimana pada motor DC brushless bagian rotornya berbeda dengan rotor pada motor DC konvensional yang hanya tersusun dari satu buah elektromagnet yang berada diantara brushes (sikat) yang terhubung. Rotor dibuat dari magnet permanen dan dapat desain dari dua sampai delapan kutub Magnet Utara (N) atau Selatan (S). Bahan material magnetis yang baik sangat diperlukan untuk mendapatkan kerapatan medan magnet yang baik pula. Biasanya magnet permanen dibuat menggunakan magnet ferrit. Tetapi saat ini dengan kemajuan

teknologi, campuran logam sudah kurang populer untuk digunakan. Meskipun dinilai lebih murah, magnet ferrit mempunyai kekurangan yaitu kerapatan fluks yang rendah sebagai bahan material yang diperlukan untuk membuat rotor.

b. Stator

Stator adalah bagian pada motor yang diam/statis dan berfungsi sebagai medan putar motor untuk memberikan gaya elektromagnetik pada rotor sehingga motor dapat berputar. Pada motor DC brushless statornya terdiri dari 12 belitan (elektromagnet) yang bekerja secara elektromagnetik dimana stator pada motor DC brushless terhubung dengan tiga buah kabel untuk disambungkan pada rangkaian kontrol sedangkan pada motor DC konvensional stator-nya terdiri dari dua buah kutub magnet permanen seperti pada Gambar 2.14. (Sarjana, 2016)



Gambar 2.14 Skema Kerja Motor BLDC (Husaini, 2015)

Cara kerja pada motor BLDC, yaitu magnet yang berada pada poros motor akan tertarik dan terdorong oleh gaya elektromagnetik yang diatur oleh driver pada motor BLDC. Hal ini membedakakn motor BLDC dengan motor DC yang menggunakan sikat mekanis yang berada pada komutator untuk mengatur waktu dan memberikan medan magnet pada lilitan. Motor BLDC ini juga berbeda dengan motor AC yang pada umumnya menggunakan siklus tenaga sendiri untuk mengatur waktu dan memberi daya pada lilitan. BLDC dapat memberikan rasio daya dan beban yang lebih tinggi secara signifikan dan memberikan efisiensi yang lebih baik dibandingkan motor tanpa sikat tradisional. Pada prinsip dasar medan magnet adalah kutub yang sama akan saling tolak menolak sedangkan apabila berlainan kutub maka akan tarik menarik. Jadi jika kita mempunyai dua buah magnet dan menandai satu sisi magnet tersebut dengan north (utara) dan yang lainnya south (selatan), maka bagian sisi north akan coba menarik south, sebaliknya jika sisi north magnet pertama akan menolak sisi north yang kedua dan seterusnya apabila kedua sisi magnet mempunyai kutub yang sama.

Prinsip mengenai kutub magnet tersebut dapat diterapkan dalam prinsip kerja motor BLDC. Secara umum motor BLDC memiliki medan magnet permanen pada rotor dan magnet yang berasal dari gaya elektromagnet (magnet yang ditimbulkan dari pemberian input arus listrik) pada bagian kumparan stator. Pada motor BLDC, kontroler berfungsi untuk mengatur arus masukan yang harus dialirkan ke kumparan stator untuk dapat menimbulkan medan elektromagnet yang sesuai untuk memutar rotor. Hal inilah yang menjadi pembeda dengan motor DC konvensional, dan menggantikan kerja komutasi mekanisnya. Magnet permanen pada motor BLDC dilengkapi dengan kumparan tiga fase. Kumparan-kumparan tersebut terletak di bagian stator. Magnet bergerak terletak di stator. Fase kumparan diaktifkan dengan penyesuain gerakan rotor. Rotasi berbasis rotasi medan magnet bagian kiri adalah fase pergerakan dan bagian kanan adalah fase eksitasi. Fluks

stator dihasilkan pada saat fase eksitasi, dan fluks rotor dihasilkan oleh magnet permanen. (Husaini, 2015)

Salah satu jenis motor listrik yang paling banyak digunakan akhir-akhir ini adalah Bushless DC (BLDC) Motor dimana motor DC ini tidak menggunakan Brush (sikat) untuk proses komutasi. BLDC motor banyak digunakan pada teknologi otomasi mutakhir, seperti: otomasi industri manufaktur maupun otomasi non-manufaktur (Robot, UAV, ROV, aeromodeling, hingga RC cars). Konstruksi yang sangat simpel menjadi pertimbangan pemakaian BLDC motor pada pengembangan bidang otomasi tersebut. Selain itu, mobil listrik maupun mobil hybrid yang menjadi trend akhir-akhir ini, juga menggunakan BLDC motor sebagai tenaga penggerak karena mempunyai efisiensi yang sangat besar hingga mencapai 95%. BLDC motor juga digunakan pada motor penggerak hardisk yang membutuhkan kecepatan serta ketahanan yang tinggi.

Tabel 2.3 Spesifikasi Motor BLDC (Brilian, 2015)

Tipe Motor	**BLDC**
Tegangan	48 V
Power Watt	350 W
Over Power Watt	> 1000 W
Kerja *Ampere*	16 – 18 A
Over Ampere Maximum	> 35 A
Torsi	18 – 25 Nm
Recommended Controller	48 V, 350 W, 17 A (*Full Features Controller)*
Maximum Controller	48 V, 1000 W, 35 A (*Full Features Controller)*
Model Socket	*Socket O ring*
Socket Hall	6 pin (*Male)*
Bobot	6 kg
Diameter Motor	40 cm
Kecepatan	48 V = 45 km/jam

2.6 Prinsip Kerja Kontrol Motor Listrik

Kata kontrol berarti mengatur atau mengendalikan, jadi yang dimaksud dengan pengontrolan motor adalah pengaturan atau pengendalian motor mulai dari pengasutan, pengoperasian hingga motor itu berhenti. Maka pengontrolan motor dapat dikategorikan menjadi tiga bagian menurut fungsinya yaitu sebagai berikut.

- Pengontrolan pada saat pengasutan (*starting*)
- Pengontrolan pada saat motor dalam keadaan beroperasi (pengaturan kecepatan, pembalikan arah putaran dan lain-lain)
- Pengontrolan pada saat motor berhenti beroperasi (pengereman).

Sesuai dengan perkembangan teknologi yang memicu perkembangan industri, cara untuk mengetahui sistem kontrol itu terus berkembang. Maka dari caranya dapat diklasifikasikan menjadi beberapa diantaranya berikut ini.

- Pengontrolan cara manual (*manual control*)
- Pengontrolan semi-otomatis (*semi-automatic control*)
- Pengontrolan otomatis (*automatic control*)
- Pengontrolan terprogram (*programmable controller*)

Dalam mengoperasikan motor listrik, agar dapat berfungsi andal dan terhindar dariganggu an dan kerusakan, dan terjamin keselamatan terhadap bahaya sengatanlistrik, maka setiap instalasi motor-motor listrik dilengkapi dengan peralatan proteksi."aitu proteksi beban lebih, pentanahan, dan hubung singkat. Motor induksi (*Asynchronous motor*) secara luas banyak digunakan di fasilitas industri dan bangunan besar. Rancangan dan peralatannya sederhana, dapat disesuaikan pada berbagai aplikasi di lapangan dan pengoperasiannya ekonomis. Ini sangat menguntungkan sebagai solusi pengendali motor induksi pada sisi harga dan kualitas. Karakteristik motor induksi tiga-phasa adalah arus bebannya tinggi pada sumber tegangan dengan *direct-on-line starting*. Menghasilkan arus start dan lonjakan yang tinggi jika di aplikasikan pada tegangan penuh, akan mengakibatkan penurunan tegangan sumber dan pengaruh transien torsi pada sistem mekanik.

Brushless DC (BLDC) motor adalah sebuah mesin listrik berputar, dimana stator merupakan belitan stator tiga phasa seperti motor induksi, dan rotor terdapat magnet permanen dipermukaannya. Dalam hal ini, motor BLDC setara dengan motor DC dengan komutator terbalik, di mana magnet berputar sedangkan konduktor tetap diam. Dalam komutator motor DC, polaritas ini diubah oleh komutator dan sikat. Namun, dalam brushless motor DC, pembalikan polaritas dilakukan oleh transistor switching untuk mensinkronkan dengan posisi rotor. Oleh karena itu, BLDC motor sering menggabungkan baik posisi sensor internal atau eksternal untuk merasakan posisi rotor yang sebenarnya, atau posisi dapat dideteksi tanpa sensor. (Elevich, 2005)

Kecepatan motor BLDC paling dipengaruhi oleh jumlah *pole* pair & struktur kumparan, resistansi kumparan, dan *clock voltase* 3 *phase motor*. Sedangkan torsi dipengaruhi oleh lebar magnet, kemampuan kontroler, dan jumlah *pole pair* & struktur kumparan. Jumlah pole pair sama dengan semakin sedikit jumlah *pole pair* maka semakin cepat putaran rpm motor BLDC. Dalam jumlah *pole pair* yang sama maka BLDC yang memiliki resistansi kumparan lebih rendah akan memiliki *top speed* rpm yang lebih tinggi. Dalam resistansi kumparan dan *pole pair* yang sama motor BLDC yang diberikan voltase yang lebih tinggi akan memiliki *top speed* yang lebih tinggi. Hal ini juga sesuai hukum ohm pada Persamaan (2.8).

$$I = \frac{V}{R} \tag{2.8}$$

Dimana, I merupakan keluaran sebagai arus, V sebagai Tegangan dan R sebagai Hambatan.

Apabila R tetap namun V semakin besar, maka I akan semakin besar pula, sehingga bisa dikatakan daya motor BLDC juga akan meningkat. Strutur kumparan sangat berpengaruh terhadap torsi, tipe kumparan segitiga delta dan kumparan tipe star

tentunya memiliki torsi yang berbeda. Jumlah pole pair sama dengan semakin banyak jumlah pole pair akan memili torsi yang semakin besar. Lebar magnet sama dengan semakin lebar dan semakin kuat daya induksi magnet maka semakin besar pula torsi motor BLDC. Untuk yang terakhir adalah kemampuan kontroler, untuk memenuhi kebutuhan torsi yang besar diperlukan *Ampere* yang besar pula, maka kontroler harus *support* dan mampu mengalirkan arus yang besar. (Brilian, 2015)

Kecepatan pada motor BLDC yang didapatkan dalam satuan rpm sehingga diperlukan konversi untuk mengubah ke dalam satuan km/jam menggunakan Persamaan (2.9).

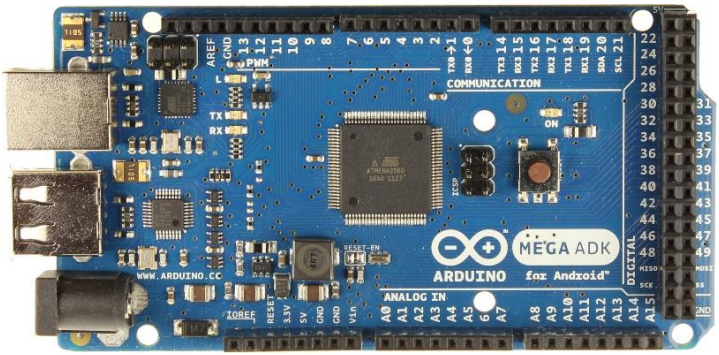
$$K = 2\,\pi\,r \tag{2.9}$$

Dimana, K merupakan keliling lingkaran dan r sebagai jari-jari lingkaran.

Dari Persamaan (2.9) didapatkan nilai keliling lingkaran kemudian akan dikalikan dengan kecepatan rpm yang didapatkan. Akan tetapi, nilai kecepatan masih dalam satuan cm/ menit yang kemudian diubah ke dalam satuan km/jam.

2.7 Arduino Mega 2560

Arduino Mega 2560 adalah *board* pengembangan mikrokontroller yang berbasis Arduino dengan menggunakan chip ATmega2560. Board ini memiliki pin I/O yang cukup banyak, sejumlah 54 buah digital I/O pin (15 pin diantaranya adalah PWM), 16 pin analog input, 4 pin UART (serial port hardware). Arduino Mega 2560 dilengkapi dengan sebuah oscillator 16 Mhz, sebuah *port* USB, *power jack* DC, ICSP *header*, dan tombol *reset*. Board ini sudah sangat lengkap, sudah memiliki segala sesuatu yang dibuthkan untuk sebuah mikrokontroller. Dengan penggunaan yang cukup sederhana, anda tinggal menghubungkan power dari USB ke PC anda atau melalui adaptor AC/DC ke *jack* DC. Gambar *board* dapat ditunjukkan pada Gambar 2.15 (Arduino, 2018)



Gambar 2.15 Mikrokontroler Arduino Mega 2560 (Arduino, 2018)

Pemrograman board Arduino Mega 2560 dilakukan dengan menggunakan Arduino Software (IDE). Chip ATmega2560 yang terdapat pada Arduino Mega 2560 telah diisi program awal yang sering disebut bootloader. Bootloader tersebut yang bertugas untuk melakukan pemrograman lebih sederhana menggunakan Arduino Software. Mega 2560 memiliki fitur-fitur sebagai berikut ini.

- *Pin-Out* : Ditambahkan pin SDA dan pin SCL yang dekat dengan pin AREF dan dua pin baru lainnya ditempatkan dekat dengan pin RESET, IOREF memungkinkan shield untuk beradaptasi dengan tegangan yang tersedia pada papan. Di masa depan, shield akan kompatibel baik dengan papan yang menggunakan AVR yang beroperasi dengan 5 Volt dan dengan Arduino Due yang beroperasi dengan tegangan 3.3 Volt. Dan ada dua pin yang tidak terhubung, yang disediakan untuk tujuan masa depan.
- Sirkuit RESET.
- Chip ATmega16U2 menggantikan chip Atmega 8U2.

Tabel 2.4 Spesifikasi Arduino Mega 2560 (Arduino, 2018)

Chip Mikrokontroller	**ATmega 2560**
Tegangan Operasi	5V
Tegangan Input (yang direkomendasikan, via *jack* DC)	7V - 12V
Tegangan Input (limit, via jack DC)	6V - 20V
Digital I/O pin	54 buah, 6 diantaranya menyediakan PWM *output*
Analog Input pin	16 buah
Arus DC per pin I/O	20 mA
Arus DC pin 3.3V	50 mA
Memori Flash	256 KB, 8 KB telah digunakan untuk *bootloader*
SRAM	8 KB
EEPROM	4 KB
Clock speed	16 Mhz
Dimensi	101.5 mm x 53.4 mm
Berat	37 g

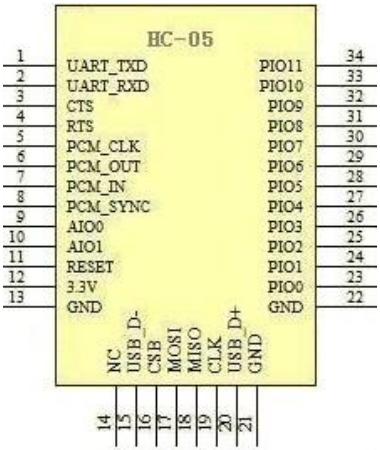
2.8 Modul *Bluetooth* HC-05

HC-05 Adalah sebuah modul *Bluetooth* SPP (*Serial Port Protocol*) yang mudah digunakan untuk komunikasi *serial wireless* (nirkabel) yang mengkonversi port serial *ke Bluetooth*. HC-05 menggunakan modulasi bluetooth V2.0 + EDR (*Enchanced Data Rate*) 3 Mbps dengan memanfaatkan gelombang radio berfrekuensi 2,4 GHz. (eProLabs, 2016)

Gambar 2.16 Modul Bluetooth HC-05 (DSD TECH, 2016)

Gambar 2.16 merupakan modul bluetooth HC-05. Modul ini dapat digunakan sebagai *slave* maupun *master*. HC-05 memiliki dua mode konfigurasi, yaitu *AT mode* dan *Communication mode*. *AT mode* berfungsi untuk melakukan pengaturan konfigurasi dari HC-05. Sedangkan *Communication mode* berfungsi untuk melakukan komunikasi bluetooth dengan piranti lain. Gambar 2.17 merupakan konfigurasi PIN Modul HC-05 dengan Arduino MEGA dengan beberapa PIN berikut.

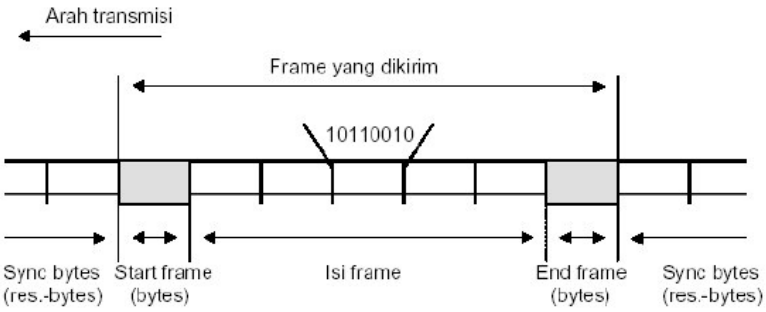
- **Vcc**: Tegangan kerja dari Bluetooth HC-05 adalah 3.6 sampai dengan 6 Volt
- **GND**: hubungkan *ground* dari modul Bluetooth dengan *ground* Arduino MEGA
- **TX**: Pin transmiter *data serial* dari Modul Bluetooth langsung ke pin Rx Arduino MEGA
- **RX**: hubungkan pin *Receive* data serial dari Modul Bluetooth ke Arduino melalui *Voltage devider*



Gambar 2.17 Konfigurasi Pin Modul HC-05 (Prasetyo M. A, 2016)

2.9 Komunikasi Serial

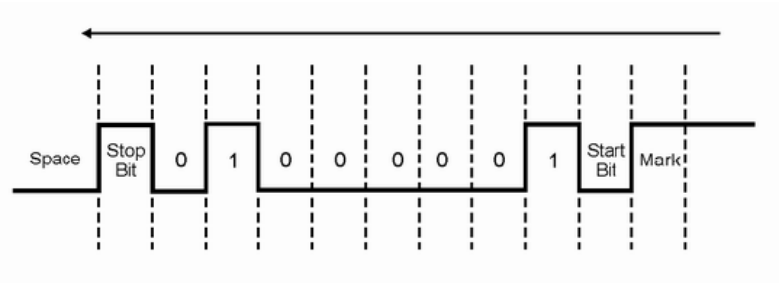
Komunikasi serial adalah pengiriman data secara serial (data dikirim satu persatu secara berurutan), sehingga komunikasi serial lebih lambat daripada komunikasi paralel. Komunikasi Serial dapat digunakan untuk menggantikan Komunikasi Parallel jalur data 8-bit dengan baik. Tidak saja memakan biaya yang lebih murah, namun dapat digunakan untuk menghubungkan dua peralatan yang sangat jauh. Misalnya menumpang pada kabel telpon. Agar komunikasi *serial* dapat bekerja dengan baik, data *byte* harus diubah ke dalam *bit-bit serial* menggunakan peralatan yang disebut *shift register parallel-in serial-out*, kemudian data dikirimkan hanya dengan satu jalur data saja.



Gambar 2.18 Sinyal Transmisi Sinkron (Suyadi, 2012)

Gambar 2.18 merupakan sinyal transmisi yang dikirimkan secara sinkron. Komunikasi *data serial* mengenal dua buah metode, yaitu *synchronous* dan *asynchronous*. Metode *sychronous* mengirimkan datanya beberapa *byte* atau karakter sebelum meminta konfirmasi apakah data sudah diterima dengan baik atau tidak. Sementara metode *asynchronous* data dikirim satu *byte* setiap pengiriman. Namun dewasa ini proses pengiriman data serial tersebut sudah dilakukan oleh sebuah chip tersendiri (*Hardware*). *Chip* tersebut disebut UART (*Universal Asynchronous Reciever Transmiter*) dan USART (*Universal Synchronous Asynchronous Reciever Transmiter*). Dalam protokol berbeda, *sychronous* memerlukan sinyal tambahan yang digunakan untuk men-*sychron*-isasi setiap denyut dari proses transfer. (Suyadi, 2012)

Komunikasi *data serial Asynchronous* sekarang sudah digunakan demikian luas untuk transmisi yang berorientasi karakter, sementara metode *Synchronous* digunakan untuk transmisi yang berorientasi blok. Pada mode *Asynchronous*, setiap karakter ditempatkan berada diantara *bit start* dan *bit stop*. *Bit start* selalu satu bit, tapi stop bit bisa satu bit atau dua bit. Start bit selalu 0 (low) dan stop bit selalu 1 (high). Contohnya, pada gambar 2.12 di mana karakter A (01000001 biner) dibingkai (dikurung) oleh start bit dan satu stop bit.



Gambar 2.19 Pembingkaian Karakter ASCII (Suyadi, 2012)

Gambar 2.19 pada komunikasi *serial Asynchronous*, peralatan dan modem dapat diprogram untuk menggunakan lebar data 7 atau 8-bit. Tentu saja ditambah dengan Stop bit. Dahulu, system karakter ASCII masih terbatas pada data 7-bit, namun sekarang ASCII extended sudah lazim menggunakan lebar data 8-bit. Untuk setiap karakter 8-bit kita masih menambahkan bit paritas disamping bit start dan bit stop. Sehingga total adalah 11-bit. Adapun bit paritas adalah bit yang menunjukkan bahwa data yang dimaksud adalah memiliki jumlah bit 1s (high) ganjil atau genap. Bit paritas adalah bit di luar data yang bersangkutan atau merupakan tambahan.

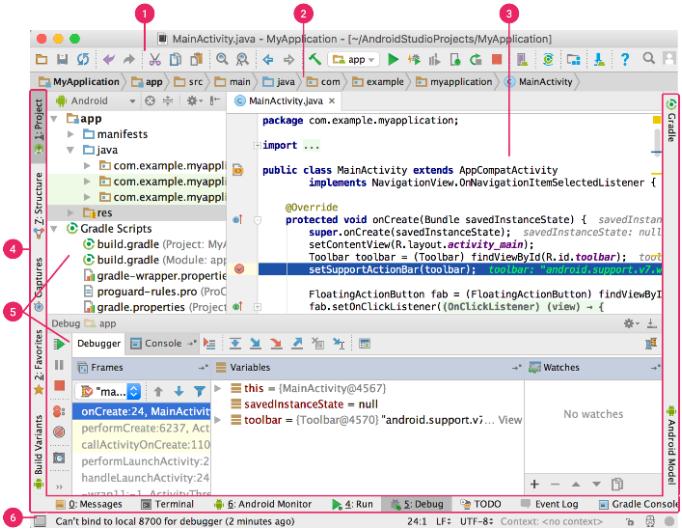
2.10 Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu - *Integrated Development Environment* (IDE) untuk pengembangan aplikasi Android, berdasarkan *IntelliJ* IDEA. Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, *Android Studio* menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi Android, misalnya seperti berikut ini.

- Sistem versi berbasis *Gradle* yang fleksibel.
- Emulator yang cepat dan kaya fitur.

- Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android.
- Instant Run untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru.
- Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode.
- Alat pengujian dan kerangka kerja yang ekstensif.
- Alat *Lint* untuk meningkatkan kinerja, kegunaan, kompabilitas versi dan masalah lain.
- Dukungan C++ dan NDK.
- Dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine.*

Android adalah *software* untuk perangkat *mobile* yang mencakup sistem operasi, *middleware* dan aplikasi kunci. Pengembangan aplikasi pada *platform Android* menggunakan bahasa pemrograman Java. Serangkaian aplikasi inti *Android* antara lain klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain. Dengan menyediakan sebuah platform pengembangan yang terbuka, pengembang *Android* menawarkan kemampuan untuk membangun aplikasi yang sangat kaya dan inovatif. Pengembang bebas untuk mengambil keuntungan dari perangkat keras, akses informasi lokasi, menjalankan *background services*, mengatur alarm, tambahkan pemberitahuan ke status bar, dan banyak lagi. *Android* bergantung pada versi *Linux* 2.6 untuk layanan sistem inti seperti keamanan, manajemen memori, manajemen proses, *network stack*, dan model driver. Kernel juga bertindak sebagai lapisan abstraksi antara hardware dan seluruh *software stack*. (Android Studio, 2018)



Gambar 2.20 Jendela Utama Android Studio (Android Studio, 2018)

Keterangan bagian yang ada pada jendela utama pada *Android Studio* adalah sebagai berikut ini.

a. Bilah Alat

Memungkinkan Anda untuk melakukan berbagai jenis tindakan, termasuk menjalankan aplikasi dan meluncurkan alat *Android.*

b. Bilah Navigasi

Membantu Anda bernavigasi di antara proyek dan membuka file untuk diedit. Bilah ini memberikan tampilan struktur yang terlihat lebih ringkas dalam jendela *Project.*

c. Jendela Editor

Tempat Anda membuat dan memodifikasi kode. Bergantung pada jenis file saat ini, editor dapat berubah. Misalnya, ketika melihat file tata letak, editor menampilkan Layout Editor.

d. Bilah Jendela Alat
Muncul di luar jendela IDE dan berisi tombol yang memungkinkan anda meluaskan atau menciutkan jendela alat individual.

e. Jendela Alat
Memberi Anda akses ke tugas tertentu seperti pengelolaan proyek, penelusuran, kontrol versi, dan banyak lagi. Anda bisa meluaskan dan juga menciutkannya.

f. Bilah Status
Menampilkan status proyek Anda dan IDE itu sendiri, serta setiap peringatan atau pesan.

2.11 Database MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*database management system*) atau DBMS yang multialur, multipengguna, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual di bawah lisensi komersial untuk kasus-kasus di mana penggunaannya tidak cocok dengan penggunaan GPL. Tidak sama dengan proyek-proyek seperti Apache, di mana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, di mana memegang hak cipta hampir atas semua kode sumbernya. (MySQL, 2018)

MySQL adalah sebuah implementasi dari sistem manajemen basis data relasional (RDBMS) yang didistribusikan secara gratis di bawah lisensi GPL (*General Public License*). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada sebelumnya SQL (*Structured Query Language*). SQL adalah

sebuah konsep pengoperasian basisdata, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

MySQL dibuat menggunakan bahasa pemrograman C dan C++ serta dapat berjalan di semua platform seperti Windows, Linux dan Unix. MySQL merupakan database pilihan untuk beberapa bahasa pemrograman web seperti PHP, Ruby on Rails dan Python. MySQL juga menjadi pilihan untuk beberapa aplikasi web sistem pengelolaan konten (content management system | CMS) sumber terbuka seperti Joomla, Wordpress dan Drupal. Selain itu, MySQL juga menjadi salah satu komponen penting dari web service solution stack LAMP (Linux, Apache, MySQL and PHP) yaitu platform pengembangan web sumber terbuka dimana Linux sebagai sistem operasi, Apache sebagai *Web Server*, MySQL sebagai RDBMS dan PHP sebagai bahasa skrip berorientasi obyek (*object-oriented scripting language)*. Berikut ini adalah beberapa keunggulan MySQL jika dibandingkan dengan RDMS lain. (MySQL, 2018)

- **Performa**. Tidak dipungkiri lagi bahwa MySQL lebih cepat dari RDBMS pesaingnya. Hasil perbandingan lima database sebagai database untuk aplikasi web, MySQL berada di urutan pertama kemudian diikuti oleh Oracle DB.
- **Biaya rendah**. MySQL tersedia gratis dibawah lisensi sumber terbuka (GPL) atau opsi lainnya dengan biaya rendah untuk lisensi komersialnya.
- **Kemudahan penggunaan**. Kebanyakan database modern saat ini menggunakan SQL sebagai bahasa standar. Namun apabila dibandingkan dengan RDBMS lainnya, MySQL masih lebih mudah dalam pengaturannya.
- **Portabilitas**. MySQL dapat dijalankan di semua platform yang ada seperti Windows, Linux dan Unix.
- **Kode sumber terbuka**. Kode sumber MySQL dapat diperoleh maupun dimodifikasi sekalipun untuk kebanyakan penggunan tidak perlu melakukan ini.

- **Ketersediaan dukungan**. Tidak semua produk sumber terbuka memiliki perusahaan induk yang dapat menawarkan dukungan, pelatihan dan sertifikasi, namun Anda bisa mendapatkan semua itu dari MySQL.

2.12 Standar ITU-T

ITU-T (*International Telecommunication Union of Telecommunication*) adalah standar internasional yang digunakan sebagai acuan dalam bidang Telekomunikasi baik itu telepon maupun data. Termasuk standar yang digunakan dalam jaringan yang tertera pada standar ITU-T G.114 yaitu berdasarkan nilai *delay*. (ITU, 2003)

Tabel 2.5 Standar ITU-T G.114 untuk *Delay* (ITU, 2003)

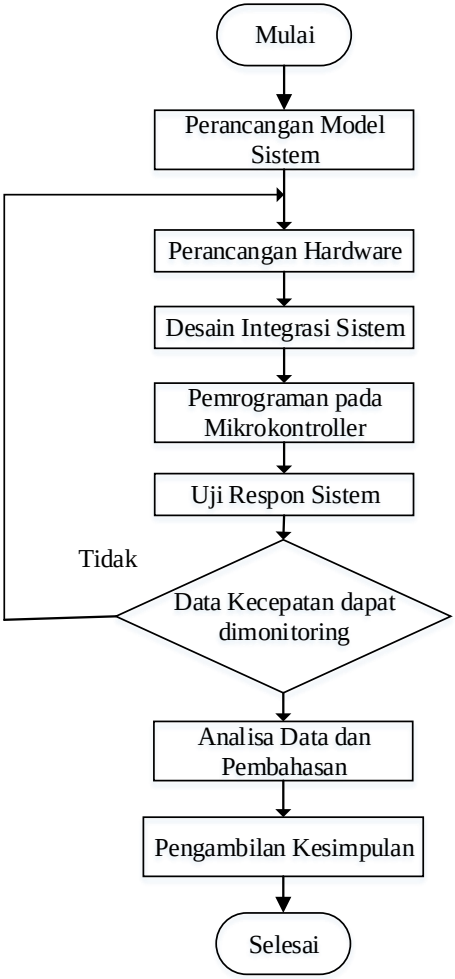
Kategori *Delay*	**Besar *Delay***
Sangat Bagus	< 150 ms
Bagus	150 ms s/d 300 ms
Jelek	300 ms s/d 450 ms
Sangat Jelek	> 450 ms

Tabel 2.4 menunjukkan standar yang berlaku pada *delay* dimana terbagi empat kategori yaitu apabila nilai dari *delay* kurang dari 150 ms maka *delay* dapat dikategorikan sangat bagus. Apabila nilai *delay* pada sistem transmisi terpaut 150ms sampi dengan 300ms maka *delay* termasuk kategori bagus, jika *delay* terpaut 300ms sampai 450ms dapat dikatakan *delay* dengan kategori jelek dan jika lebih besar dari 450 ms maka sistem transmisi dapat dikatakan sangat jelek.

Halaman ini sengaja dikosongkan

BAB III
METODOLOGI PENELITIAN

Pada penelitian tugas akhir ini dijelaskan mengenai beberapa tahap seperti pada Gambar 3.1.

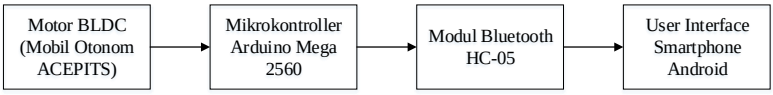


Gambar 3.1 Diagram Alir Tugas Akhir

3.1 Perancangan Model Sistem

Pada pemodelan sistem ini dilakukan pemilihan langkah-langkah perancangan yang harus dilakukan untuk dapat terciptanya sistem monitoring jarak jauh pada mobil otonom ACEPITS (*Autonomous Car Engineering Physics* ITS) dengan kendali kecepatan menggunakan metode PID seperti pada gambar 3.2 diagram blok pemodelan sistem. Dari motor BLDC (*Brushless Direct Current Motor)* yang terpasang pada mobil otonom kemudian akan dikontrol menggunakan mikrokontroller arduino mega 2560. Dilakukan pemrograman pada mikrokontroller arduino mega 2560 ini dengan memberikan nilai *input* berupa *setpoint* kecepatan motor BLDC dalam satuan rpm. Dengan menambahkan nilai parameter PID yaitu kp, ki, kd menggunakan metode Ziegler-Nichols yang dapat berpengaruh terhadap *gain* PID. Dengan adanya parameter PID tersebut juga berpengaruh terhadap performa dari kecepatan motor BLDC yang akan dikendalikan pada mobil otonom.

Pemrograman juga dilakukan pada mikrokontroller sebagai alat integrasi dengan menggunakan modul bluetooth hc-05 agar data kecepatan pada motor BLDC tersebut dapat ditransmisikan untuk ditampilkan pada *smartphone* android. Kemudian data yang sudah dapat dimonitoring akan ditampilkan dan disimpan ke dalam *database.* Gambar 2.3 merupakan gambar diagram blok dari keseluruhan sistem yang akan dibuat.



Gambar 3.2 Diagram Blok Model Sistem

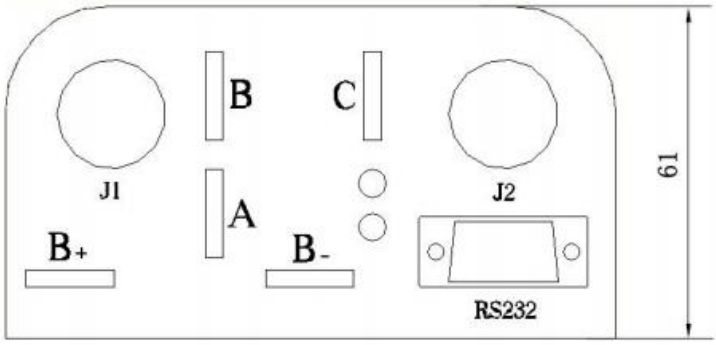
3.2 Perancangan *Hardware*

Dari hasil pemodelan sistem yang sudah dilakukan kemudian dimulai dari perancangan alat ini dengan melakukan perakitan bagian – bagian untuk mobil otonom ACEPITS seperti pemasangan motor BLDC yang sekaligus sebagai roda belakang dengan rangka pada mobil. Kemudian dilanjutkan pemasangan pada bagian-bagian lainnya yang dibutuhkan untuk mobil otonom ACEPITS seperti roda depan, sistem rem, dan juga kontroller untuk laju dari mobil otonom sendiri seperti pada Gambar 3.3.

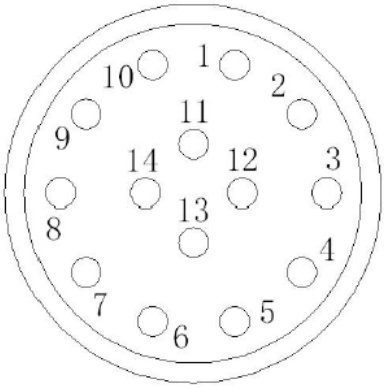
Gambar 3.3 Perancangan Prototipe Mobil Otonom ACEPITS

Dilakukan perancangan *hardware* alat untuk sistem kendali kecepatan pada mobil otonom ACEPITS berupa pemasangan dari beberapa pin yang ada pada motor BLDC yang dihubungkan dengan port yang ada pada kelly kontroller. Kemudian dari port tersebut juga dihubungkan mikrokontroller arduino mega 2560. Untuk pemasangan kabel pin sudah terdapat pada *wiring diagram* yang didapatkan pada *website* kelly kontroller. Untuk informasi port pada kelly kontroller juga sudah terdapat keterangan pada *user manual.*

Gambar 3.4 merupakan panel untuk tampak depan dari *Kelly Controllers.* Untuk mendefinisikan *port* B+ sebagai *battery positive, port* B- sebagai *battery negative,* port A sebagai *Output* U/1/A *phase, port* B sebagai *Output* V/2/B *phase, port* C sebagai *Output* W/3/C *phase.*



Gambar 3.4 Panel Depan *Kelly Controller* (Kelly Controller, 2018)

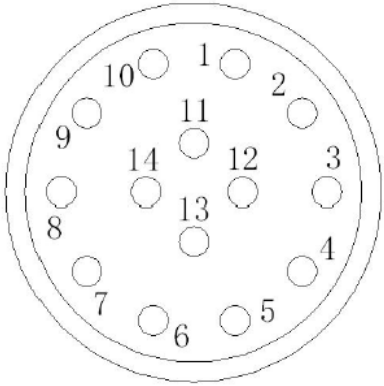


Gambar 3.5 *Port* Penghubung pada J1 (Kelly Controller, 2018)

Gambar 3.5 merupakan *port* penghubung *Kelly Controllers* pada *port* yang ada di motor BLDC. Dengan keterangan beberapa fungsi pada pin sebagai berikut.

a. Definisi Keterangan Pin pada J1:

1 - 12V 30mA (Hanya dapat digunakan untuk sinyal *switch)*
2 - *Current meter*. <200mA
3 - *Main contactor driver* (<2A)
4 - *Alarm*: *To drive reverse beeper* (<200Ma)
5 - RTN: *Signal return*
6 - *Green* LED: *Running indication*
7 - RTN: *Signal return*
8 - RS232 *receiver*
9 - RS232 *transmitter*
10 - CAN *bus high*
11 - CAN *bus low*
12 - *Reserved*
13 - RTN: *Signal return, or power supply return*
14 - *Red* LED: *Fault code*



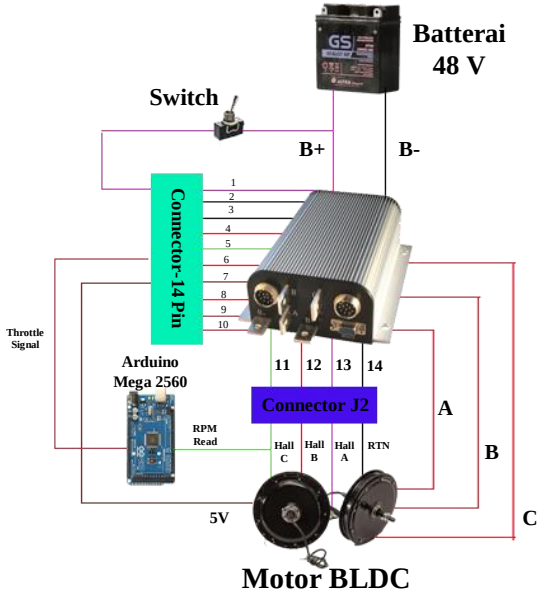
Gambar 3.6 *Port* Penghubung pada J2 (Kelly Controller, 2018)

Gambar 3.6 merupakan *port* penghubung *Kelly Controllers* pada *port* yang ada di motor BLDC. Dengan keterangan beberapa fungsi pada pin sebagai berikut.

b. Definisi Keterangan Pin pada J2:

1. PWR: *Controller power supply (input)*
2. RTN: *Signal return, or power supply return*
3. RTN: *Signal return*
4. 12V *high-level brake and motor temperature input*
5. *Throttle analog input*, 0-5V
6. *Brake analog input*, 0-5V
7. 5V: 5V *supply output*. <40mA
8. Micro_SW: *Throttle switch input*
9. Reversing switch input
10. *Brake switch input*
11. *Hall phase* C
12. *Hall phase* B
13. *Hall phase* A
14. RTN: *Signal return*

Dengan keterangan port yang sudah terdefinisi tersebut sehingga dapat dilakukan pemasangan kabel pin pada beberapa port sesuai dengan yang dibutuhkan. Dari motor BLDC sampai ke port Kelly Kontroller dengan menggunakan beberapa pin pada port konektor. Motor BLDC terdapat 5 pin konektor yaitu *Hall A, Hall B, Hall C, RTN* dan VCC (5V *Output*) yang masuk ke pin J2 *Connector on Controllers*. Untuk mensuplai *Kelly Controllers* dibutuhkan baterai *accu* sebanyak 4 buah (48 V) dengan masing masing daya baterai *accu* 12 V. Untuk mikrokontroller arduino mega 2560 digunakan sebagai pembacaan RPM dengan menghubungkan *Hall C* pada Motor BLDC dan juga sebagai fungsi *throtle*. Sedangkan untuk input sinyal pada mikrokontroller akan masuk pada *connector*-14 pin atau J1 pin *connector*. Untuk *wiring diagram* alat untuk Kelly Kontroller dapat ditunjukkan pada Gambar 3.7.

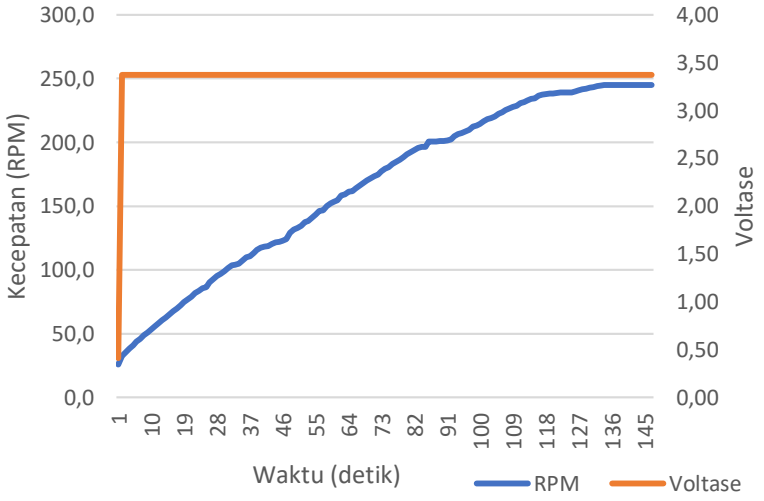


Gambar 3.7 Wiring Diagram Pembuatan Alat

3.3 Penentuan Nilai *Gain* PID *Tuning* Ziegler-Nichols

Sistem kontrol pada motor BLDC untuk penentuan nilai kp, kid dan kd didapatkan dengan menggunkan *tuning* metode Zigler-Nichols dari pemodelan secara dinamik dengan menggunakan uji *open loop*. Respon uji ini dilakukan dengan pengambilan data pada motor BLDC dengan *input* tegangan 0-5 volt sehingga didapatkan respon uji *open loop* pada grafik respon yang ditunjukkan pada Gambar 3.8.

Dibutuhkan nilai waktu tunda (L) dan *time constant* (T) dari respon sistem untuk dari Gambar 3.8 respon pada grafik tersebut didapatkan nilai L dan T yang kemudian dilakukan perhitungan menggunakan metode Ziegler-Nichols untuk mendapatkan parameter kontrol. Berikut nilai L, T dan nilai parameter Kp, Td dan Kd pada Tabel 3.1.



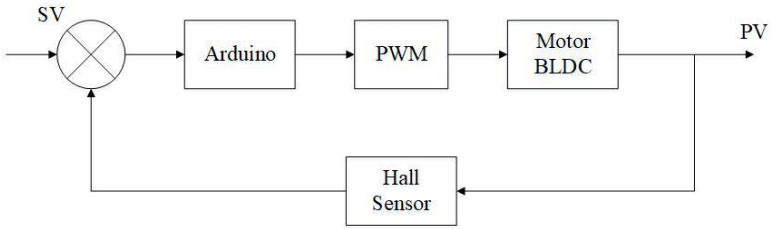
Gambar 3.8 Respon Uji *Open Loop*

Tabel 3.1 Nilai Parameter PID *Tuning* Zigler-Nichols

L	T	Kp	Ki	Kd
1,02	0,13	9,33	0,26	0,07

Nilai *gain* parameter kontrol yang didapatkan pada Tabel 3.1 selanjutnya akan di aplikasikan ke dalam *gain* kontrol pada pengendalian kecepatan secara *close loop* pada *plant* motor BLDC dapat ditunjukkan pada diagram blok pengendalian sistem seperti pada Gambar 3.9.

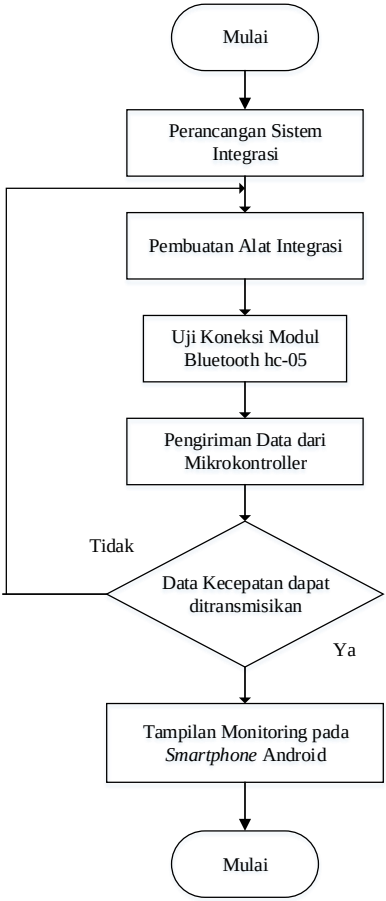
Kontrol pengendalian PID diberikan pada mikrokontroller Arduino mega 2560 yang digunakan sebagai pengendalian utama yang mengatur kecepatan pada motor BLDC dengan *input setpoint* berupa PWM *(Pulse Witdh Modulation).* Dari kecepatan motor yang bergerak kemudian dapat dihitung dengan *hall sensor* sehingga keluaran dari kecepatan motor BLDC yang bergerak berupa satuan rpm. Gambar 3.9 merupakan diagram blok dari pengendalian untuk sistem kontrol yang diberikan pada motor BLDC.



Gambar 3.9 Diagram Blok Pengendalian Sistem

3.4 Perancangan Sistem Integrasi

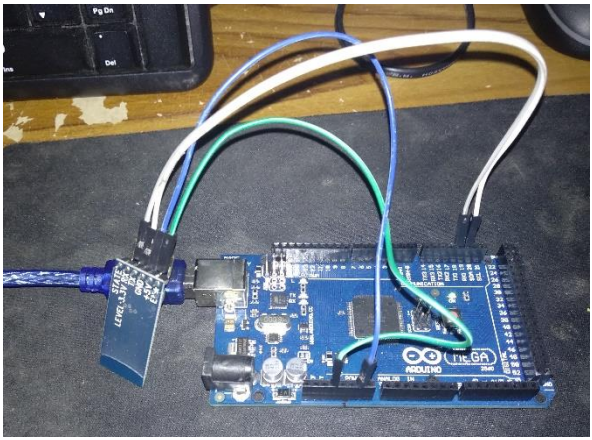
Pada tahap ini dilakukan rancang bangun sistem dengan menggabungkan seluruh sistem yang telah dibuat pada *hardware* sampai dengan *user interface*. Data kecepatan yang telah didapatkan pada mobil otonom dapat dimonitoring menggunakan aplikasi *smartphone* android seperti pada Gambar 3.10.



Gambar 3.10 Diagram Alir Perancangan Sistem Integrasi

3.5.1 Pembuatan Alat Integrasi

Dilakukan pembuatan alat untuk sistem integrasi yang menghubungkan mikrokontroller arduino mega 2560 dengan menggunakan modul bluetooth hc-05. Kemudian pin pada modul bluetooth hc-05 disambungkan menggunakan kabel *jumper* ke mikrokontroller sesuai dengan port yang dibutuhkan. Dikarenakan modul bluetooth hc-05 menerima tegangan dari 3,6V hingga 6V maka pin VCC pada modul tersebut akan dihubungkan dengan pin 5V pada mikrokontroller. Untuk sinyal pada pin TX pada modul akan dihubungkan dengan pin RX/Pin 19 pada mikrokontroller. Kemudian untuk sinyal pada pin RX pada modul juga akan dihubungkan dengan pin TX/Pin 18 pada mikrokontroller. Lalu untuk pin GND pada modul dihubungkan dengan pin GND pada mikrokontroller seperti pada Gambar 3.11.



Gambar 3.11 Sistem Integrasi pada Modul hc-05

3.5.2 Aplikasi Android Studio

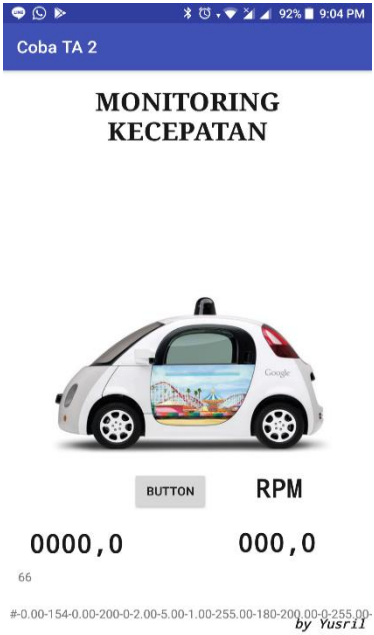
Pada tahap ini dilakukan pembuatan *software* aplikasi android pada *mobile* menggunakan progam Android Studio. Pada Android Studio digunakan bahasa pemrograman *java*. Untuk aplikasi android yang dihasilkan ini digunakan sebagai alat monitoring data kecepata RPM *read* pada motor BLDC mobil otonom yang masuk pada mikrokontroller arduino Mega 2560 yang kemudian akan diintegrasikan menggunakan modul *bluetooth* yang tersambung pada aplikasi *mobile* yang sudah dibuat pada Android Studio sebagai alat untuk monitoring.

3.5.3 Database MySQL

Pada pembuatan *database* MySQL dilakukan untuk menyimpan data kecepatan yang telah berhasil ditransmisikan pada *user interface* aplikasi android yang telah dibuat. Untuk pembuatan *database* MySQL dilakukan menggunakan aplikasi XAMPP pada server lokal yang ada pada *web server*. Data yang dikirimkan dari mikrokontroller Arduino mega 2560 menggunakan modul *bluetooth* hc-05 kemudian data tersebut dimonitoring pada aplikasi *smartphone* android dan akan bersamaan ditampilkan pada *database* MySQL. Digunakan *database* MySQL ini bertujuan untuk menyimpan dan menghindari dari kehilangan data yang kemungkinan akan mengalami data *lost*.

3.5.4 Integrasi Alat dengan Kontroller

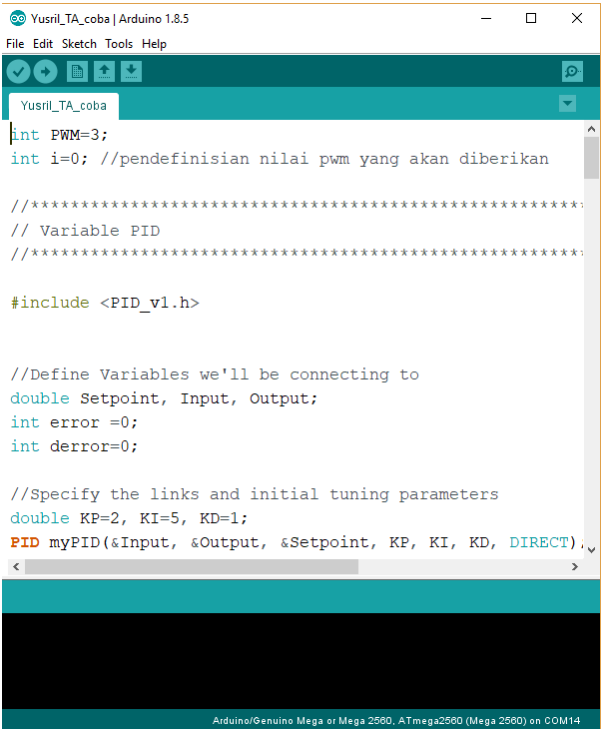
Pada bagian ini dilakulan pengintegrasian untuk dapat membuat sistem ini dapat melakukan monitoring kecepatan pada mobil otonom ACEPITS. Integrasi hardware dengan controller sebagai tujuan utama dalam sistem ini. Integrasi ini dilakukan dengan menggunakan progam aplikasi pada *user interface mobile* Android yang sudah dibuat pada *software* Android Studio. Dengan menggunakan Mikrokontroller Arduino Mega 2560 yang sudah terkoneksikan dengan modul *bluetooth* hc-05. Sehingga dengan menggunakan progam pada aplikasi *mobile* Android data kecepatan yang ditransmisikan seperti pada Gambar 3.12.



Gambar 3.12 Integrasi Alat dengan Aplikasi Android

3.5 Pemrograman pada Mikrokontroller

Pada tahap ini dilakukan pemrograman pada mikrokontroller Arduino Mega 2560 menggunakan *software* Arduino. Untuk kodingan progam ini diinputkan berupa tegangan 0-5V sebagai *throttle* sehingga mengeluarkan output berupa RPM pada Motor BLDC. Untuk penambahan nilai parameter PID yaitu nilai kp, ki, kd juga dilakukan kodingan pada mikrokontroller ini. Kemudian ditambahkan kodingan sebagai alat komunikasi serial untuk integrasi alat mikrokontroller Arduino mega 2560 pada modul Bluetooth hc-05 agar data kecepatan dapat dimonitoring pada aplikasi *smatphone* android dapat ditunjukkan pada Gambar 3.13.



Gambar 3.13 Pemrograman pada Mikrokontroller

3.6 Pengujian Respon pada Sistem

Untuk pengujian respon sistem integrasi pada mikrokontroller hingga data kecepatan dapat dimonitoring aplikasi android pertama dilakukan dengan menghubungkan koneksi bluetooth agar dapat tersambung dengan aplikasi pada android. Kemudian pada progam aplikasi android dilakukan pengujian apabila sudah terhubung dengan mikrokontroller maka setelah dilakukan *trigger* pada aplikasi android sehingga motor BLDC pada mobil otonom sudah dapat berjalan. Pada motor BLDC tersebut akan memberikan data kecepatan RPM yang akan ditransmisikan oleh bluetooth pada aplikasi android dan data akan dapat dimonitoring seperti yang ditunjukkan pada Gambar 3.14.

55.68	431	8.00	-55	-7	15.00	12.50	1.50	0.00	0	0.00	255
55.56	54	1.00	-55	0	15.00	12.50	1.50	0.00	0	0.00	255
55.56	54	1.00	-55	0	15.00	12.50	1.50	0.00	0	0.00	255
54.55	55	1.00	-54	1	15.00	12.50	1.50	0.00	0	0.00	255
55.56	54	1.00	-55	-1	15.00	12.50	1.50	0.00	0	0.00	255
55.56	54	1.00	-55	0	15.00	12.50	1.50	0.00	0	0.00	255
55.56	54	1.00	-55	0	15.00	12.50	1.50	0.00	0	0.00	255
54.55	55	1.00	-54	1	15.00	12.50	1.50	0.00	0	0.00	255
109.09	55	2.00	-109	-55	15.00	12.50	1.50	0.00	0	0.00	255
50.85	59	1.00	-50	59	15.00	12.50	1.50	0.00	0	0.00	255
52.63	57	1.00	-52	-2	15.00	12.50	1.50	0.00	0	0.00	255
105.26	57	2.00	-105	-53	15.00	12.50	1.50	0.00	0	0.00	255
49.18	61	1.00	-49	56	5.81	4.84	2.42	0.00	0	0.00	255
76.57	431	11.00	-76	-27	15.00	12.50	1.50	0.00	0	0.00	255
55.56	54	1.00	-55	21	15.00	12.50	1.50	0.00	0	0.00	255
111.11	54	2.00	-111	-56	15.00	12.50	1.50	0.00	0	0.00	255
55.56	54	1.00	-55	56	15.00	12.50	1.50	0.00	0	0.00	255
54.55	55	1.00	-54	1	15.00	12.50	1.50	0.00	0	0.00	255
107.14	56	2.00	-107	-53	15.00	12.50	1.50	0.00	0	0.00	255
98.36	61	2.00	-98	9	15.00	12.50	1.50	0.00	0	0.00	255
53.57	56	1.00	-53	45	15.00	12.50	1.50	0.00	0	0.00	255
107.14	56	2.00	-107	-54	15.00	12.50	1.50	0.00	0	0.00	255
49.18	61	1.00	-49	58	5.81	4.84	2.42	0.00	0	0.00	255
97.45	431	14.00	-97	-48	15.00	12.50	1.50	0.00	0	0.00	255
54.55	55	1.00	-54	43	15.00	12.50	1.50	0.00	0	0.00	255

Gambar 3.14 Pengujian pada Respon Sistem

3.7 Analisa Data dan Pembahasan

Pada tahap ini dilakukan analisa data pada hasil pengujian sistem monitoring yang telah didapatkan. Dengan hasil data yang diperoleh tersebut dari motor BLDC sudah dapat bergerak sesuai dengan *setpoint* yang telah ditentukan. Untuk hasil data monitoring dapat ditransmisikan pada aplikasi android menggunakan mikrokontroller dengan integrasi dari modul bluetooth.

Halaman ini sengaja dikosongkan

BAB IV
ANALISA DATA DAN PEMBAHASAN

4.1 Pengujian untuk Respon pada Motor BLDC

Dilakukan pengujian pada motor BLDC untuk respon kecepatan agar motor dapat bergerak. Untuk pengujian ini dilakukan pemrograman pada mikrokontroller menggunakan mode *full throttle* atau *input* pada gas *throttle* diberikan nilai 255. Dalam hal ini bertujuan untuk memberikan *power* yang besar agar motor BLDC mampu untuk mulai bergerak. Data pengujian dapat ditunjukkan pada Tabel 4.1.

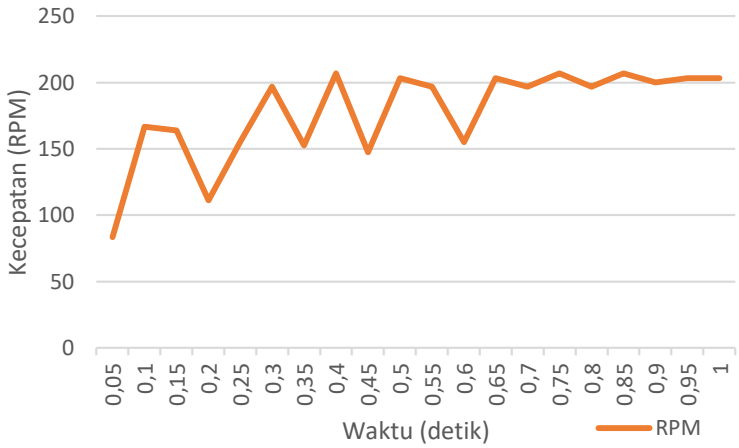
Tabel 4.1 Data Pengujian pada Motor BLDC

No	Waktu	Kecepatan	Full Throttle
	(detik)	(RPM)	
1	0,05	83,33	255
2	0,1	166,67	255
3	0,15	163,64	255
4	0,2	111,11	255
5	0,25	155,17	255
6	0,3	196,72	255
7	0,35	152,54	255
8	0,4	206,9	255
9	0,45	147,54	255
10	0,5	203,39	255
11	0,55	196,72	255
12	0,6	155,17	255

Tabel 4.1 Data Pengujian pada Motor BLDC (Lanjutan)

No	Waktu	Kecepatan	Full Throttle
	(detik)	(RPM)	
13	0,65	203,39	255
14	0,7	196,72	255
15	0,75	206,9	255
16	0,8	196,72	255
17	0,85	206,9	255
18	0,9	200	255
19	0,95	203,39	255
20	1	203,39	255

Tabel 4.1 dari data pengujian yang sudah didapatkan, bisa dilihat bahwa motor BLDC sudah dapat bergerak. Untuk kecepatan motor sendiri pada awalnya berada di rpm 83,33 kemudian pada data-data selanjutnya akan terus mengalami kenaikan untuk nantinya akan mencapai *setpoint* yang telah ditentukan pada mode *full throttle* ini. Pada pengujian ini diambil sampel 18 data untuk mengetahui respon pada motor BLDC apakah motor sudah dapat bergerak untuk kemudian akan dilakukan pengujian pada tahap selanjutnya. Agar mengetahui pergerakan kecepatan pada motor BLDC dari data yang sudah didapatkan kemudian diplotkan kedalam grafik seperti pada Gambar 4.1



Gambar 4.1 Respon Pengujian Respon pada Motor BLDC

4.2 Analisa Uji Sistem Kendali Kecepatan pada Motor BLDC

Pengujian pada sistem kendali kecepatan ini dilakukan untuk mengetahui karakteristik respon dari sistem kendali kecepatan agar motor BLDC dapat bergerak sesuai dengan *setpoint* yang telah ditentukan untuk kecepatan putaran rpm pada motor. Dengan menambahkan parameter PID untuk masing-masing kontrol *Proportional, Integratif, Derivatif*. Untuk proses penambahan formula pada kontrol PID yaitu dengan nilai kp, ki, kd dilakukan dengan menggunakan metode *trial and error* hingga diperoleh hasil nilai yang diinginkan. Untuk respon sistem parameter-parameter seperti nilai puncak kurva maksimum pada respon (*maximum overshoot)*, waktu yang dibutuhkan respon untuk masuk daerah *steady (settling time)* dan sinyal respon mencapai keadaan stabil *(error steady state)*, untuk kriteria *ess* berada didaerah *error* 2% atau 5%. Pada penelitian ini digunakan parameter PID untuk nilai kp, ki, dan kd masing-masing secara berurutan adalah sebesar 9,33, 0,26 dan 0,07. Pada tahap ini akan dilakukan dengan melakukan beberapa pengujian pada sistem kendali kecepatan.

4.2.1. Uji Kecepatan pada Setpoint 200

Pada pengujian ini dilakukan dengan memberikan *input setpoint* yang telah ditentukan sebesar 200. Pengujian ini dilakukan tanpa menambahkan beban dari kerangka mobil otonom. Untuk hasil data yang diperoleh dapat ditunjukkan pada Tabel 4.2.

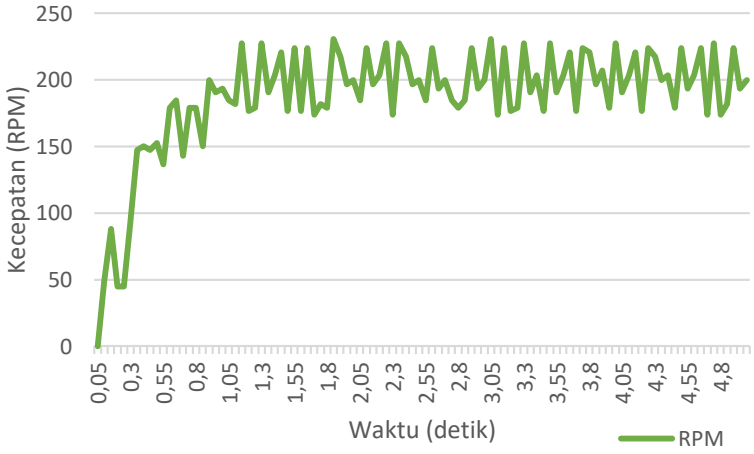
Tabel 4.2 Data Uji Kecepatan pada Setpoint 200

No	Waktu	Kecepatan		Setpoint
	(Detik)	(RPM)	(Km/Jam)	
1	0,05	0	0,00	200
2	0,1	48,39	3,65	200
3	0,15	88,24	6,66	200
4	0,2	44,78	3,38	200
5	0,25	44,78	3,38	200
6	0,3	92,31	6,96	200
7	0,35	147,54	11,13	200
8	0,4	150	11,31	200
9	0,45	147,54	11,13	200
10	0,5	152,54	11,51	200
11	0,55	136,36	10,29	200
12	0,6	179,1	13,51	200
13	0,65	184,62	13,93	200
14	0,7	142,86	10,78	200
15	0,75	179,1	13,51	200
16	0,8	179,1	13,51	200
17	0,85	150	11,31	200

*untuk tabel lengkapnya dapat dilihat pada lampiran

Pengambilan data pada pengujian sistem kendali kecepatan dengan memberikan nilai *setpoint* sebesar 200 didapatkan data sebanyak 100 data. Untuk hasil data tersebut didapatkan karena *delay time* pada mikrokontroller di *set* sebesar 100 milisekon atau setara dengan 0,1 detik. Hal tersebut dilakukan untuk proses perhitungan pada mikrokontroller sebagai *output* nilai rpm dari putaran roda motor BLDC. Nilai Kecepatan untuk satuan km/jam merupakan hasil dari perhitungan konversi dari kecepatan satuan rpm menggunakan Persamaan (2.9).

Nilai kecepatan dengan satuan km/jam pada data yang lainnya dapat dilakukan dengan cara yang sama yaitu dengan menggunakan Persamaan (2.9). Dari hasil data yang sudah didapatkan tersebut pada Tabel 4.2 diperoleh nilai awal pada saat motor BLDC mulai bergerak sebesar 48,39 rpm atau setara dengan 3,64 km/jam. Untuk mengetahui pergerakan dari nilai hasil data tersebut maka dapat diplotkan kedalam grafik seperti yang ditunjukkan pada Gambar 4.2.



Gambar 4.2 Respon Uji Kecepatan pada *Setpoint* 200

Gambar 4.2 dari grafik uji kecepatan terlihat bahwa kecepatan rpm pada motor BLDC terus meningkat dari 0 hingga akhirnya kecepatan rpm pada motor BLDC berosilasi pada *setpoint* yang telah ditentukan yaitu berada pada angka 200. Didapatkan nilai puncak maksimal pada respon grafik pengujian kecepatan dengan *setpoint* 200 sebesar 230,77 rpm. Dan waktu yang dibutuhkan untuk menuju *steady* memiliki *settling time* sebesar 4,96 detik. Terdapat beberapa data *error* yang besar pada saat awal motor BLDC mulai bergerak, kemungkinan hal ini dapat terjadi dikarenakan kontroller yang digunakan untuk menggerakkan kedua motor BLDC ini hanya satu. Sehingga diperlukannya *power* gas pada *throttle* yang besar diawal untuk menarik kedua motor BLDC tersebut agar mampu untuk bergerak sesuai dengan *setpoint* yang tentukan.

4.2.2. Uji Kecepatan pada Setpoint 150

Pada pengujian ini dilakukan dengan memberikan *input setpoint* yang telah ditentukan sebesar 150. Pengujian ini dilakukan tanpa menambahkan beban dari kerangka mobil otonom. Untuk hasil data yang diperoleh dapat ditunjukkan pada Tabel 4.3.

Tabel 4.3 Data Uji Kecepatan pada *Setpoint* 150

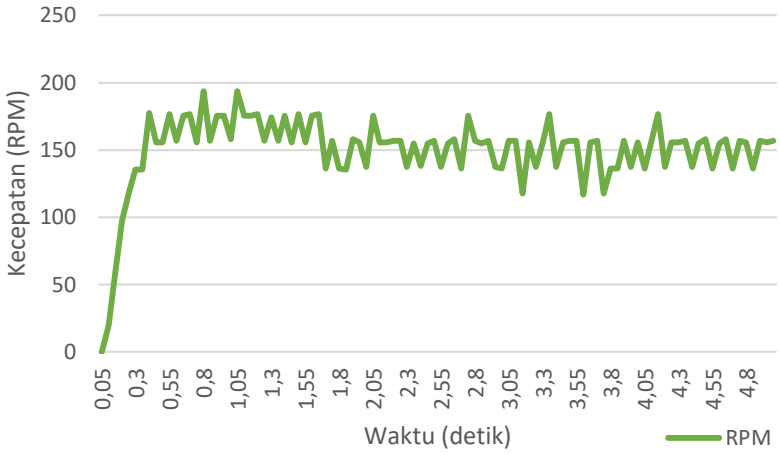
No	Waktu	Kecepatan		Setpoint
	(detik)	RPM	Km/Jam	
1	0,05	0	0,00	150
2	0,1	19,61	1,48	150
3	0,15	57,69	4,35	150
4	0,2	97,4	7,35	150
5	0,25	118,42	8,93	150
6	0,3	135,48	10,22	150
7	0,35	135,48	10,22	150
8	0,4	177,63	13,40	150
9	0,45	155,84	11,75	150

Tabel 4.3 Data Uji Kecepatan pada *Setpoint* 150 (Lanjutan)

No	Waktu	Kecepatan		Setpoint
	(detik)	RPM	Km/Jam	
10	0,5	155,84	11,75	150
11	0,55	176,47	13,31	150
12	0,6	156,86	11,83	150
13	0,65	175,32	13,22	150
14	0,7	176,47	13,31	150
15	0,75	155,84	11,75	150
16	0,8	193,55	14,60	150
17	0,85	156,86	11,83	150

*untuk tabel lengkapnya dapat dilihat pada lampiran

Pengambilan data pada pengujian sistem kendali kecepatan dengan memberikan nilai *setpoint* sebesar 150. didapatkan data sebanyak 100 data. Kemudian dihitung nilai kecepatan pada satuan km/jam sebagai hasil konversi dari data rpm yang didapatkan menggunakan persamaan (4.1). Dari hasil data yang sudah didapatkan tersebut pada tabel 4.2 diperoleh nilai awal pada saat motor BLDC mulai bergerak sebesar 19,61 rpm atau setara dengan 1,48 km/jam. Untuk mengetahui pergerakan dari nilai hasil data tersebut maka dapat diplotkan kedalam grafik seperti yang ditunjukkan pada Gambar 4.3.



Gambar 4.3 Respon Uji Kecepatan pada *Setpoint* 150

Gambar 4.4 dari grafik uji kecepatan terlihat bahwa kecepatan rpm pada motor BLDC terus meningkat dari 0 hingga akhirnya kecepatan rpm mencapai *setpoint* 150. Kemudian kecepatan pada motor BLDC mulai berosilasi pada data ke-11 dengan kecepatan sebesar 176,47 rpm atau setara dengan 13,31 km/jam. Untuk nilai puncak maksimal yang didapatkan pada grafik pengujian kecepatan dengan *setpoint* 150 sebesar 193,55 rpm. Dan yang untuk yang dibutuhkan untuk mencapai keadaan *steady* yaitu memiliki *settling time* sebesar 4,89 detik dengan memiliki *error steady state* sebesar 0,0475%.

4.3 Analisa Uji Sistem Kecepatan dengan Mobil Otonom

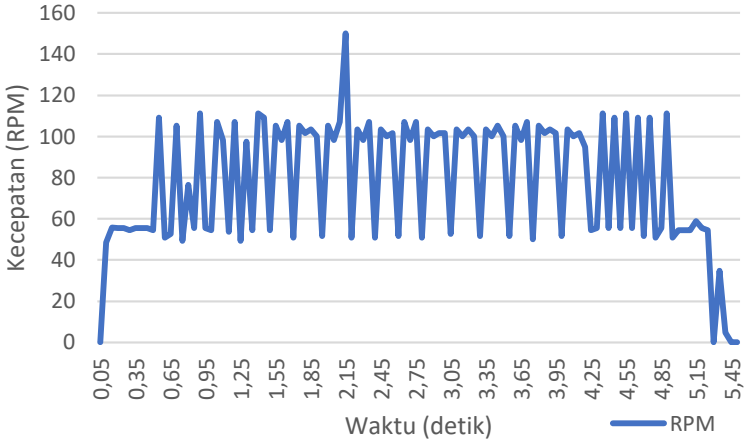
Pengujian sistem kecepatan pada motor BLDC kali ini dilakukan dengan menggunakan mobil otonom. Untuk pengujian pada sistem kendali kecepatan ini dilakukan guna mengetahui karakteristik dari respon pada sistem kecepatan secara keseluruhan dari motor BLDC maupun dengan beban pada mobil otonom. Sistem kendali kecepatan untuk pengujian digunakan *input* nilai *setpoint* 255 atau dapat disebut dengan mode *full throttle*. Hal ini

digunakan untuk memancing *power* yang besar pada motor BLDC agar motor mampu untuk dapat bergerak mengingat bahwa motor BLDC yang bergerak juga menarik beban yang ada pada mobil otonom. Pengambildan data pada pengujian ini dilakukan dengan mengambil hasil data kecepatan rpm motor BLDC dari awal motor mulai bergerak sampai motor kembali berhenti. Hasil data pada pengujian sistem kecepatan dengan mobil otonom dapat ditunjukkan pada Tabel 4.5.

Tabel 4.4 Data Uji Kecepatan dengan Mobil Otonom

No	Waktu	Kecepatan		Mode Full Throtlle
	(detik)	RPM	Km/Jam	
1	0,05	0	0,00	255
2	0,1	48,61	3,67	255
3	0,15	55,68	4,20	255
4	0,2	55,56	4,19	255
5	0,25	55,56	4,19	255
6	0,3	54,55	4,11	255
7	0,35	55,56	4,19	255
8	0,4	55,56	4,19	255
9	0,45	55,56	4,19	255
10	0,5	54,55	4,11	255
11	0,55	109,09	8,23	255
12	0,6	50,85	3,84	255
13	0,65	52,63	3,97	255
14	0,7	105,26	7,94	255
15	0,75	49,18	3,71	255
16	0,8	76,57	5,78	255
17	0,85	55,56	4,19	255

*untuk tabel lengkapnya dapat dilihat pada lampiran



Gambar 4.4 Respon Uji Sistem Kecepatan dengan Beban

Didapatkan hasil data dari pengujian kecepatan motor BLDC dengan mobil otonom sebanyak 110 data. Kemudian dari hasil data kecepatan yang diperoleh tersebut dapat diplotkan kedalam grafik seperti yang ditunjukkan pada Gambar 4.4.

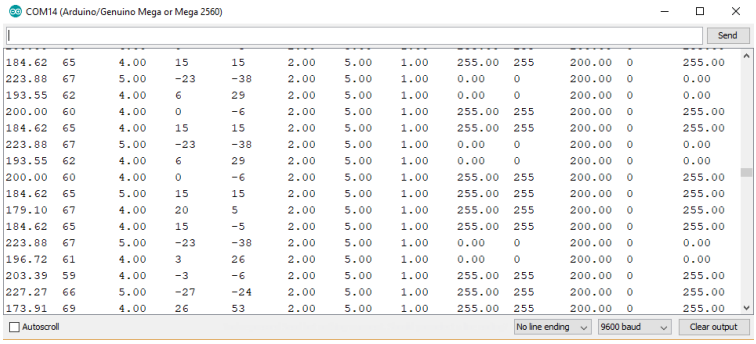
Pada gambar grafik yang sudah didapatkan tersebut dapat dianalisa bahwa pengujian untuk respon kecepatan pada motor BLDC dengan mobil otonom mulai bergerak pada kecepatan 48,61 rpm atau sekitar 3,66 km/jam. Kemudian berosilasi pada kecepatan sekitar 100-110 rpm atau 7-8 km/jam. Pada saat mobil otonom akan berhenti diberikan *setpoint* 0 pada data ke-103. Kecepatan mobil otonom mulai melambat pada kecepatan 58,82 rpm atau 4,43 km/jam. Pada data yang telah didapatkan dengan dilakukannya pengujian kecepatan motor BLDC dengan menggabungkan beban dari mobil otonom didapatkan kecepatan rata-rata pada mobil otonom sebesar 78,09 rpm atau 5,89 km/jam. Untuk nilai puncak maksimal yang dapat didapatkan seperti pada grafik pengujian dengan nilai sebesar 150 rpm. Pada Gambar 4.4 dapat dilihat bahwa terjadinya osilasi yang besar. Kemungkinan hal ini dapat terjadi dikarenakan beberapa faktor yang

mempengaruhi diantaranya adalah seperti beban yang ada pada mobil otonom sendiri. Lalu untuk kontroller yang digunakan hanya satu sebagai penggerak utama dari dua roda motor BLDC. Dan kurangnya pengambilan data yang dilakukan untuk mengetahui karakteristik dari respon kecepatan yang lebih lama karena terbatasnya panjang lintasan yang digunakan untuk menguji kecepatan pada mobil otonom ini.

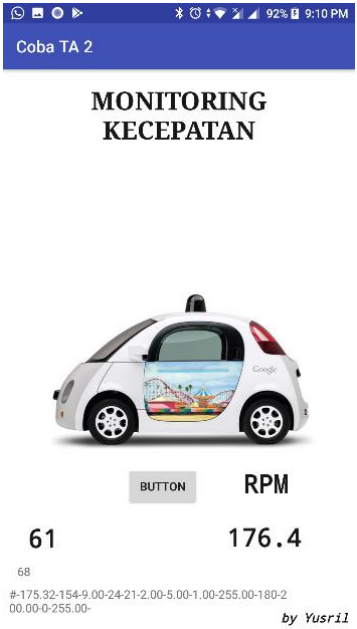
Apabila dilakukan analisa perbandingan dari data yang sudah didapatkan pada kecepatan motor BLDC yang tidak menggunakan beban mobil otonom dengan kecepatan motor BLDC yang menggunakan beban dari mobil otonom, maka nilai kecepatan yang didapatkan sangat jauh berbeda. Untuk kecepatan motor BLDC tanpa menggunakan beban dari mobil otonom bisa melaju hingga kecepatan 227,27 rpm atau 17,14 km/jam. Akan tetapi, untuk kecepatan motor BLDC yang menggunakan beban dari mobil otonom hanya dapat melaju hingga 111,11 rpm atau 8,38 km/jam. Hal ini dapat diambil kesimpulan bahwa kecepatan keseluruhan sebenarnya untuk mobil otonom hanya dapat melaju dengan kecepatan rendah.

4.4 Pengujian Sistem Monitoring pada Aplikasi Android

Hasil data pengujian kecepatan mobil otonom yang telah didapatkan, kemudian dapat langsung diintegrasikan untuk dimonitoring kecepatannya seperti pada Gambar 4.5.



Gambar 4.5 Data Pengujian dari *Serial Monitor*



Gambar 4.6 Monitoring Kecepatan pada Aplikasi Android

Data yang telah diterima dengan menggunakan Mikrokontroller Arduino Mega 2560, kemudian diintegrasikan oleh mdoul Bluetooth hc-05 yang kemudian diintegrasikan untuk dapat ditampilkan melalui aplikasi pada *mobile* Android. Dari hasil integrasi yang sudah dilakukan telah dapat berjalan dan data yang didapatkan tersebut setelah ditampilkan pada aplikasi monitoring kecepatan pada Android akan disimpan pada penyimpanan *database. Database* berfungsi sebagai tempat penyimpanan hasil dari pembacaan serial dari mikrokontroller Arduino Mega 2560. Untuk monitoring data kecepatan seperti pada Gambar 4.6.

Tabel 4.5 Data Uji Pengaruh Jarak pada Sistem Monitoring

No	Jarak (cm)	Data diterima	Delay (ms)
1	100	Ya	50
2	200	Ya	50
3	300	Ya	50
4	400	Ya	50
5	500	Ya	50
6	600	Ya	50
7	700	Ya	50
8	800	Ya	70
9	900	Ya	90
10	1000	Ya	100
11	1100	Ya	200
12	1200	Ya	1000
13	1300	Tidak	-

Kemudian dilakukan analisa uji pada sistem monitoring untuk transmisi data yang dikirimkan dari mikrokontroller arduino mega 2560 pada aplikasi android dengan menggunakan modul bluetooth hc-05. Pada pengujian ini akan dilihat apakah data yang dikirimkan akan mengalami *delay* pada saat proses monitoring. Untuk hasil data pada pengujian sistem monitoring kecepatan dengan mobil otonom dapat ditunjukkan pada Tabel 4.5.

Tabel 4.5 dari hasil data yang telah didapatkan berikut dapat dilihat bahwa *delay* pada pengiriman data masih berada pada angka normal yaitu sebesar 50 ms sesuai dengan standar ITU-T (*International Telecommunication Union of Telecommunication*) G.114 untuk *Delay*. (ITU, 2003) Pada pengujian ini dilakukan pengaruh jarak pada modul bluetooth hc-05 ke aplikasi android yang dapat mempengaruhi besar *delay* pada saat proses pengiriman

data. Untuk jarak maksimal pada spesifikasi *datasheet* modul bluetooth hc-05 dengan gelombang radio frekuensi 2,4 Ghz adalah sebesar 10 meter.

Tabel 4.6 dari hasil pengujian yang telah didapatkan untuk jangkauan jarak pada bluetooth hc-05 yaitu pada jarak 100-700 cm, *delay* yang didapatkan adalah sebesar 50 ms dan data kecepatan dapat ditampilkan dengan baik sesuai dengan pengiriman dari mikrokontroller kepada aplikasi android. Kemudian *delay* mulai naik pada jarak 800-1200 cm dengan delay pada jangkauan jarak 1200 cm sebesar 1000 ms. Pada jarak ini data masih bisa ditampilkan pada aplikasi android akan tetapi data kecepatan yang dimonitoring mengalami *delay* yang besar. Kemudian untuk jarak diatas 1300 cm data kecepatan yang dimonitoring pada aplikasi android sudah tidak dapat ditampilkan sehingga koneksi bluetooth terputus.

Untuk pengiriman data dimulai dari sensor pembacaan motor BLDC pada mobil otonom hingga masuk ke progam aplikasi dan kemudian akan diupload data ke dalam *database* untuk nantinya dapat dimonitoring dibutuhkan waktu pengiriman data dengan rata-rata *delay* selama 0,25 detik atau sebesar 250 milidetik. Dari waktu awal pengiriman data sebesar 0,05 detik hingga pada waktu penerimaan data adalah sebesar 0,3 detik. Dari nilai *delay* yang telah didapatkan tersebut sesuai dengan ITU-TG. 114 untuk *delay* sebesar 250 ms masih dapat dikategorikan *delay* yang bagus yaitu pada rentang 150-300 ms.

BAB V
KESIMPULAN

5.1 Kesimpulan

Adapun kesimpulan dari penelitian ini yang telah dilakukan adalah sebagai berikut.

a. Pengendalian kecepatan pada mobil otonom ACEPITS dapat dirancang menggunakan metode PID dengan *tuning* Ziegler-Nichols dan didapatkan nilai parameter *gain* kp, ki, dan kd secara berturut-turut adalah sebesar 9,33, 0,26 dan 0,07 untuk pengujian kecepatan rata-rata pada motor BLDC dengan beban mobil otonom ACEPITS sebesar 78,09 rpm.
b. Sistem kendali kecepatan pada mobil otonom ACEPITS dapat dirancang menggunakan mikrokontroller Arduino Mega 2560 yang diintegrasikan dengan aplikasi *mobile* pada Android. Pada hasil pengambilan uji data dapat berjalan dengan baik dan dapat ditampilkan pada *user interface* aplikasi Android dengan *delay* normal sebesar 50 ms. Kemudian untuk jangkauan jarak paling jauh yang dapat diterima hingga 1300 cm dengan nilai *delay* sebesar 1000 ms.

5.2 Saran

Adapun saran pada penelitian ini yang telah dilakukan karena penelitian ini masih dibutuhkan pengembangan kedepan agar mendapatkan hasil yang lebih baik antara lain adalah sebagai berikut.

a. Diperlukannya pencatatan yang terintegrasi dengan menggunakan teknologi modern seperti *Blockchain* karena sistem ini juga sebagai alat pengamanan yang kuat.
b. Diperlukannya *troubleshooting* pada sistem yang berbasis IoT ini dikarenakan apabila tidak adanya *troubleshooting* maka proses pada kontroller akan akan lebih mempersulit pengguna mobil otonom ACEPITS.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

Achzab, A. (2014, Februari 7). *Kontrol PID*. Retrieved from Blog Buaya Instrument: http://blog.buaya-instrument.com/kontrol-pid.html

Android Studio. (2018, Januari 27). *Mengenal Android Studio*. Retrieved from Developers Android: https://developer.android.com/studio/intro/?hl=id

Arduino. (2018, Januari 25). *Arduino Mega ADK*. Diambil kembali dari Arduino: https://www.arduino.cc/en/Main/ArduinoBoardMegaADK?from=Main.ArduinoBoardADK

Brilian, I. (2015, Juli 18). *TENTANG SEPUTAR MOTOR BLDC KENDARAAN LISTRIK*. Retrieved from ELectric Arts: https://www.electricisart-bogipower.com/2014/11/tentang-seputar-motor-bldc-kendaraan.html

Brown, E. (2016, September 13). *Who Needs the Internet of Things?* Retrieved from Linux.com: https://www.linux.com/news/who-needs-internet-things

Crothers, B. (2015, November 12). *Google Is Leader In 'Revolutionary' Self-Driving Cars, Says IHS*. Retrieved from Forbes: https://www.forbes.com/sites/brookecrothers/2015/11/12/google-is-leader-in-revolutionary-self-driving-cars-says-ihs/#7902cd039b76

Doran, G. T. (1981). *There's a S.M.A.R.T. way to write management's goals and objectives.* Washington: AMA FORUM.

DSD TECH. (2016, Mei 28). *DSD TECH HC-05 Bluetooth Serial Pass-through Module Wireless Serial Communication with Button for Arduino*. Retrieved from Amazon: https://www.amazon.co.uk/DSD-TECH-HC-05-Pass-through-Communication/dp/B01G9KSAF6

Elevich, L. N. (2005). 3-Phase BLDC Motor Control with Hall Sensors Using 56800/E Digital Signal Controllers. *Freescale Semiconductor*, Rev. 2.0.

eProLabs. (2016, Mei 26). *Bluetooth Module HC-05*. Retrieved from ePro Labs WiKi: https://wiki.eprolabs.com/index.php?title=Bluetooth_Module_HC-05

Gehrig, S. K., & Stein, F. J. (1999). Dead Reckoning and Cartography Using Stereo Vision for an Autonomous Car. *Proceedings of the 1999 IEEGRSJ International Conference on Intelligent Robots and Systems*, 0-7803-5184-3/99/$10.00.

Hawkins, A. J. (2017, 7 November). *Waymo is first to put fully self-driving cars on US roads without a safety driver*. Retrieved Januari 24, 2018, from The Verge: https://www.theverge.com/2017/11/7/16615290/waymo-self-driving-safety-driver-chandler-autonomous

Husaini, A. N. (2015, September 17). *insiyoer.com*. Retrieved from Prinsip Kerja Motor Brushless DC (BLDC Motor): http://www.insinyoer.com/prinsip-kerja-motor-brushless-dc-bldc-motor/

ITU. (2003). TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU. *INTERNATIONAL TELECOMMUNICATION UNION*, G. 114.

Kelly Controller. (2016, December). Kelly KBL Brushless Motor. *Kelly Controller User's Manual*, Rev.3.4.

Krasniqi, X., & Hajrizi, E. (2016). Use of IoT Technology to Drive the Automotive Industry from Connected to Full Autonomous Vehicles. *ScienceDirect*, 269-274.

M., A. (2009). *CONTROL SYSTEM, ROBOTICS AND AUTOMATION*. Oxford, United Kingdom: EOLSS Publishers/UNESCO.

Momoh, J. (2009). Smart grid design for efficient and flexible power networks operation and control. *Power Systems Conference and Exposition*, 1-8.

MPS. (2014). Brushless DC Motor Fundamentals. *MPS The Future of Analog IC Technology*, Rev 1.0.

MySQL. (2018, Februari 4). *MySQL for the Internet of Things*. Retrieved from MySQL: https://www.mysql.com/why-mysql/white-papers/mysql-for-internet-of-things/

Ninan, S., Gangula, B., Alten, M. v., & Sniderman, B. (2015, August 18). *Who owns the road? The IoT-connected car of today—and tomorrow*. Retrieved from The Internet of Things in automotive: https://www2.deloitte.com/insights/us/en/focus/internet-of-things/iot-in-automotive-industry.html

Ogata, K. (2010). *Modern Control Engineering.* New Jersey: Pearson Education.

Prasetyo, B. (2015, Oktober 8). *BERAPA SIH BIAYA MERAKIT SEPEDA LISTRIK?* Retrieved from Electric Art-Bogipower.com: https://www.electricisart-bogipower.com/2015/05/berapa-sih-biaya-merakit-sepeda-listrik.html

Prasetyo, M. A. (2016, Mei 13). *Masuk ke Mode AT Command HC-05 dengan Arduino*. Retrieved from Boarduino: http://www.boarduino.web.id/2016/01/cara-masuk-ke-mode-at-command-hc-05.html

Sarjana. (2016, September 4). *Kontruksi Motor BLDC*. Retrieved from OnExperience: https://onexpirience.wordpress.com/2016/09/04/blog-post-title-2/

Shamseldin, M. A., & EL-Samahy, A. A. (2014, September 9-11). Speed Control of BLDC Motor By Using PID Control and Self-tuning Fuzzy PID Controller. *15th International Workshop on Research and Education in Mechatronics (REM)*, 978-1-4799-3029-6/14.

Suyadi. (2012). Komunikasi Serial dan Port Serial (COM). *Teknik Informatika UMS*.

Tam, D. (2012, September 25). *Google's Sergey Brin: You'll ride in robot cars within 5 years*. Retrieved from Cnet:

https://www.cnet.com/news/googles-sergey-brin-youll-ride-in-robot-cars-within-5-years/

Thrun, S. (2010). Toward Robotic Cars. *Communications of the ACM, 53*(4), 99-106.

Veselý, V., & Rosinová, D. (2011, June 14-17). Robust PSD controller design. *18th International Conference on Process Control*, 565-570.

Wetherill, J. (2015, Maret 17). *What Does it Mean to be "On" The Internet of Things?* Retrieved from The New Stack: https://thenewstack.io/what-does-it-mean-to-be-on-the-internet-of-things/

Zhang, P., Li, F., & Bhatt, N. (2010, July 26). Next-Generation Monitoring, Analysis, and Control for the Future Smart Control Center. *IEEE Transactions on Smart Grid*, *1*(2), 186-192.

Zhao, J., & Yu, Y. (2011). Brushless DC Motor Fundamentals Application Note. *MPS The Future of Analog IC Technology*, AN047.

LAMPIRAN A

PENGAMBILAN DATA KECEPATAN

A. PENGUJIAN KECEPATAN PADA SETPOINT 200

No	Waktu	Kecepatan		Setpoint
	(Detik)	(RPM)	(Km/Jam)	
1	0,05	0	0,00	200
2	0,1	48,39	3,65	200
3	0,15	88,24	6,66	200
4	0,2	44,78	3,38	200
5	0,25	44,78	3,38	200
6	0,3	92,31	6,96	200
7	0,35	147,54	11,13	200
8	0,4	150	11,31	200
9	0,45	147,54	11,13	200
10	0,5	152,54	11,51	200
11	0,55	136,36	10,29	200
12	0,6	179,1	13,51	200
13	0,65	184,62	13,93	200
14	0,7	142,86	10,78	200
15	0,75	179,1	13,51	200
16	0,8	179,1	13,51	200
17	0,85	150	11,31	200
18	0,9	200	15,09	200
19	0,95	190,48	14,37	200
20	1	193,55	14,60	200
21	1,05	184,62	13,93	200
22	1,1	181,82	13,71	200

No	Waktu	Kecepatan		Setpoint
	(Detik)	(RPM)	(Km/Jam)	
23	1,15	227,27	17,14	200
24	1,2	176,47	13,31	200
25	1,25	179,1	13,51	200
26	1,3	227,27	17,14	200
27	1,35	190,48	14,37	200
28	1,4	203,39	15,34	200
29	1,45	220,59	16,64	200
30	1,5	176,47	13,31	200
31	1,55	223,88	16,89	200
32	1,6	176,47	13,31	200
33	1,65	223,88	16,89	200
34	1,7	173,91	13,12	200
35	1,75	181,82	13,71	200
36	1,8	179,1	13,51	200
37	1,85	230,77	17,41	200
38	1,9	217,39	16,40	200
39	1,95	196,72	14,84	200
40	2	200	15,09	200
41	2,05	184,62	13,93	200
42	2,1	223,88	16,89	200
43	2,15	196,72	14,84	200
44	2,2	203,39	15,34	200
45	2,25	227,27	17,14	200
46	2,3	173,91	13,12	200
47	2,35	227,27	17,14	200
48	2,4	217,39	16,40	200
49	2,45	196,72	14,84	200

No	Waktu	Kecepatan		Setpoint
	(Detik)	(RPM)	(Km/Jam)	
50	2,5	200	15,09	200
51	2,55	184,62	13,93	200
52	2,6	223,88	16,89	200
53	2,65	193,55	14,60	200
54	2,7	200	15,09	200
55	2,75	184,62	13,93	200
56	2,8	179,1	13,51	200
57	2,85	184,62	13,93	200
58	2,9	223,88	16,89	200
59	2,95	193,55	14,60	200
60	3	200	15,09	200
61	3,05	230,77	17,41	200
62	3,1	173,91	13,12	200
63	3,15	223,88	16,89	200
64	3,2	176,47	13,31	200
65	3,25	179,1	13,51	200
66	3,3	227,27	17,14	200
67	3,35	190,48	14,37	200
68	3,4	203,39	15,34	200
69	3,45	176,47	13,31	200
70	3,5	227,27	17,14	200
71	3,55	190,48	14,37	200
72	3,6	203,39	15,34	200
73	3,65	220,59	16,64	200
74	3,7	176,47	13,31	200
75	3,75	223,88	16,89	200
76	3,8	220,59	16,64	200

No	Waktu	Kecepatan		Setpoint
	(Detik)	(RPM)	(Km/Jam)	
77	3,85	196,72	14,84	200
78	3,9	206,9	15,61	200
79	3,95	179,1	13,51	200
80	4	227,27	17,14	200
81	4,05	190,48	14,37	200
82	4,1	203,39	15,34	200
83	4,15	220,59	16,64	200
84	4,2	176,47	13,31	200
85	4,25	223,88	16,89	200
86	4,3	217,39	16,40	200
87	4,35	200	15,09	200
88	4,4	203,39	15,34	200
89	4,45	179,1	13,51	200
90	4,5	223,88	16,89	200
91	4,55	193,55	14,60	200
92	4,6	203,39	15,34	200
93	4,65	223,88	16,89	200
94	4,7	173,91	13,12	200
95	4,75	227,27	17,14	200
96	4,8	173,91	13,12	200
97	4,85	181,82	13,71	200
98	4,9	223,88	16,89	200
99	4,95	193,55	14,60	200
100	5	200	15,09	200

B. PENGUJIAN KECEPATAN PADA SETPOINT 150

No	Waktu	Kecepatan		Setpoint
	(detik)	(RPM)	(Km/Jam)	
1	0,05	0	0,00	150
2	0,1	19,61	1,48	150
3	0,15	57,69	4,35	150
4	0,2	97,4	7,35	150
5	0,25	118,42	8,93	150
6	0,3	135,48	10,22	150
7	0,35	135,48	10,22	150
8	0,4	177,63	13,40	150
9	0,45	155,84	11,75	150
10	0,5	155,84	11,75	150
11	0,55	176,47	13,31	150
12	0,6	156,86	11,83	150
13	0,65	175,32	13,22	150
14	0,7	176,47	13,31	150
15	0,75	155,84	11,75	150
16	0,8	193,55	14,60	150
17	0,85	156,86	11,83	150
18	0,9	175,32	13,22	150
19	0,95	175,32	13,22	150
20	1	157,89	11,91	150
21	1,05	193,55	14,60	150
22	1,1	175,32	13,22	150
23	1,15	175,32	13,22	150
24	1,2	176,47	13,31	150
25	1,25	156,86	11,83	150
26	1,3	174,19	13,14	150

No	Waktu	Kecepatan		Setpoint
	(detik)	(RPM)	(Km/Jam)	
27	1,35	156,86	11,83	150
28	1,4	175,32	13,22	150
29	1,45	155,84	11,75	150
30	1,5	176,47	13,31	150
31	1,55	155,84	11,75	150
32	1,6	175,32	13,22	150
33	1,65	176,47	13,31	150
34	1,7	136,36	10,29	150
35	1,75	156,86	11,83	150
36	1,8	136,36	10,29	150
37	1,85	135,48	10,22	150
38	1,9	157,89	11,91	150
39	1,95	155,84	11,75	150
40	2	137,25	10,35	150
41	2,05	175,32	13,22	150
42	2,1	155,84	11,75	150
43	2,15	155,84	11,75	150
44	2,2	156,86	11,83	150
45	2,25	156,86	11,83	150
46	2,3	137,25	10,35	150
47	2,35	154,84	11,68	150
48	2,4	138,16	10,42	150
49	2,45	154,84	11,68	150
50	2,5	156,86	11,83	150
51	2,55	137,25	10,35	150
52	2,6	154,84	11,68	150
53	2,65	157,89	11,91	150

No	Waktu	Kecepatan		Setpoint
	(detik)	(RPM)	(Km/Jam)	
54	2,7	136,36	10,29	150
55	2,75	175,32	13,22	150
56	2,8	156,86	11,83	150
57	2,85	154,84	11,68	150
58	2,9	156,86	11,83	150
59	2,95	137,25	10,35	150
60	3	136,36	10,29	150
61	3,05	156,86	11,83	150
62	3,1	156,86	11,83	150
63	3,15	117,65	8,87	150
64	3,2	155,84	11,75	150
65	3,25	137,25	10,35	150
66	3,3	155,84	11,75	150
67	3,35	176,47	13,31	150
68	3,4	137,25	10,35	150
69	3,45	155,84	11,75	150
70	3,5	156,86	11,83	150
71	3,55	156,86	11,83	150
72	3,6	116,88	8,82	150
73	3,65	155,84	11,75	150
74	3,7	156,86	11,83	150
75	3,75	117,65	8,87	150
76	3,8	136,36	10,29	150
77	3,85	136,36	10,29	150
78	3,9	156,86	11,83	150
79	3,95	137,25	10,35	150
80	4	155,84	11,75	150

No	Waktu	Kecepatan		Setpoint
	(detik)	(RPM)	(Km/Jam)	
81	4,05	136,36	10,29	150
82	4,1	155,84	11,75	150
83	4,15	176,47	13,31	150
84	4,2	137,25	10,35	150
85	4,25	155,84	11,75	150
86	4,3	155,84	11,75	150
87	4,35	156,86	11,83	150
88	4,4	137,25	10,35	150
89	4,45	154,84	11,68	150
90	4,5	157,89	11,91	150
91	4,55	136,36	10,29	150
92	4,6	154,84	11,68	150
93	4,65	157,89	11,91	150
94	4,7	136,36	10,29	150
95	4,75	156,86	11,83	150
96	4,8	155,84	11,75	150
97	4,85	136,36	10,29	150
98	4,9	156,86	11,83	150
99	4,95	155,84	11,75	150
100	5	156,86	11,83	150

C. ANALISA UJI SISTEM KECEPATAN DENGAN MOBIL OTONOM

No	Waktu	Kecepatan		Mode Full Throtlle
	(detik)	RPM	Km/Jam	
1	0,05	0	0,00	255
2	0,1	48,61	3,67	255
3	0,15	55,68	4,20	255
4	0,2	55,56	4,19	255
5	0,25	55,56	4,19	255
6	0,3	54,55	4,11	255
7	0,35	55,56	4,19	255
8	0,4	55,56	4,19	255
9	0,45	55,56	4,19	255
10	0,5	54,55	4,11	255
11	0,55	109,09	8,23	255
12	0,6	50,85	3,84	255
13	0,65	52,63	3,97	255
14	0,7	105,26	7,94	255
15	0,75	49,18	3,71	255
16	0,8	76,57	5,78	255
17	0,85	55,56	4,19	255
18	0,9	111,11	8,38	255
19	0,95	55,56	4,19	255
20	1	54,55	4,11	255
21	1,05	107,14	8,08	255
22	1,1	98,36	7,42	255
23	1,15	53,57	4,04	255
24	1,2	107,14	8,08	255
25	1,25	49,18	3,71	255

No	Waktu	Kecepatan		Mode Full Throtlle
	(detik)	RPM	Km/Jam	
26	1,3	97,45	7,35	255
27	1,35	54,55	4,11	255
28	1,4	111,11	8,38	255
29	1,45	109,09	8,23	255
30	1,5	54,55	4,11	255
31	1,55	105,26	7,94	255
32	1,6	98,36	7,42	255
33	1,65	107,14	8,08	255
34	1,7	50,85	3,84	255
35	1,75	105,26	7,94	255
36	1,8	101,69	7,67	255
37	1,85	103,45	7,80	255
38	1,9	100	7,54	255
39	1,95	51,72	3,90	255
40	2	105,26	7,94	255
41	2,05	98,36	7,42	255
42	2,1	107,14	8,08	255
43	2,15	150	11,31	255
44	2,2	50,85	3,84	255
45	2,25	103,45	7,80	255
46	2,3	98,36	7,42	255
47	2,35	107,14	8,08	255
48	2,4	50,85	3,84	255
49	2,45	103,45	7,80	255
50	2,5	100	7,54	255
51	2,55	101,69	7,67	255
52	2,6	51,72	3,90	255

No	Waktu	Kecepatan		Mode Full Throtlle
	(detik)	RPM	Km/Jam	
53	2,65	107,14	8,08	255
54	2,7	98,36	7,42	255
55	2,75	107,14	8,08	255
56	2,8	50,85	3,84	255
57	2,85	103,45	7,80	255
58	2,9	100	7,54	255
59	2,95	101,69	7,67	255
60	3	101,69	7,67	255
61	3,05	52,63	3,97	255
62	3,1	103,45	7,80	255
63	3,15	100	7,54	255
64	3,2	103,45	7,80	255
65	3,25	100	7,54	255
66	3,3	51,72	3,90	255
67	3,35	103,45	7,80	255
68	3,4	100	7,54	255
69	3,45	105,26	7,94	255
70	3,5	100	7,54	255
71	3,55	51,72	3,90	255
72	3,6	105,26	7,94	255
73	3,65	98,36	7,42	255
74	3,7	107,14	8,08	255
75	3,75	50	3,77	255
76	3,8	105,26	7,94	255
77	3,85	101,69	7,67	255
78	3,9	103,45	7,80	255
79	3,95	101,69	7,67	255

No	Waktu	Kecepatan		Mode Full Throtlle
	(detik)	RPM	Km/Jam	
80	4	51,72	3,90	255
81	4,05	103,45	7,80	255
82	4,1	100	7,54	255
83	4,15	101,69	7,67	255
84	4,2	94,94	7,16	255
85	4,25	54,55	4,11	255
86	4,3	55,56	4,19	255
87	4,35	111,11	8,38	255
88	4,4	55,56	4,19	255
89	4,45	109,09	8,23	255
90	4,5	55,56	4,19	255
91	4,55	111,11	8,38	255
92	4,6	55,56	4,19	255
93	4,65	109,09	8,23	255
94	4,7	51,72	3,90	255
95	4,75	109,09	8,23	255
96	4,8	50,85	3,84	255
97	4,85	55,56	4,19	255
98	4,9	111,11	8,38	255
99	4,95	50,85	3,84	255
100	5	54,55	4,11	255
101	5,05	54,55	4,11	255
102	5,1	54,55	4,11	255
103	5,15	58,82	4,44	0
104	5,2	55,56	4,19	0
105	5,25	54,55	4,11	0
106	5,3	0	0,00	0

No	Waktu	Kecepatan		Mode Full Throtlle
	(detik)	RPM	Km/Jam	
107	5,35	34,72	2,62	0
108	5,4	4,75	0,36	0
109	5,45	0	0,00	0
110	5,5	0	0,00	0

Halaman ini sengaja dikosongkan

LAMPIRAN B

KODINGAN UNTUK PEMROGRAMAN

A. Kodingan Untuk Pemrograman Pada Mikrokontroller Arduino Mega 2560

```
int PWM=3;
int i=0; //pendefinisian nilai pwm yang akan diberikan

//**************************************************
**********************
// Variable PID
//**************************************************
**********************

#include <PID_v1.h>

//Define Variables we'll be connecting to
double Setpoint, Input, Output;
int error =0;
int derror=0;

//Specify the links and initial tuning parameters
double KP=2, KI=5, KD=1;
PID myPID(&Input, &Output, &Setpoint, KP, KI, KD,
DIRECT);

//**************************************************
**********************
// Variable tachometer
//**************************************************
**********************

float rev=0;
float rpm;
```

```
int oldtime=0;
int time;
int val;
void isr() //interrupt service routine
{
rev++;
}

// Setup routine runs once when you press reset:
void setup()
{
   pinMode(PWM,OUTPUT);
   Serial.begin(9600);

attachInterrupt(digitalPinToInterrupt(2),isr,RISING);//attaching
the interrupt

   //Input = analogRead(PIN_INPUT);
   Setpoint = 200;

   //turn the PID on
   myPID.SetMode(AUTOMATIC);

}

// Loop routine runs over and over again forever:
void loop()
{
   bluetut();
   analogWrite(PWM,Setpoint);
   tacho();
   int error0 = error;
   error = Setpoint-rpm;
   derror= error-error0;

   Serial.print(error);
```

```
    Serial.print("\t");
    Serial.print(derror);
    Serial.print("\t");

    Serial.print( KP);
    Serial.print("\t");
    Serial.print( KI);
    Serial.print("\t");
    Serial.print( KD);
    Serial.print("\t");

    Input=rpm;
    myPID.Compute();
    Serial.print(Output);
    Serial.print("\t");
   val= Output;

    Serial.print(val);
    Serial.print("\t");
    Serial.print(Setpoint);
    Serial.print("\t");
    Serial.print(i);
    Serial.print("\t");
    Serial.println(Output);

    analogWrite(PWM,Output);

    //delay(15);
}

void bluetut(){

   while(Serial.available())
  {
    char blue=Serial.read();
```

```
if(blue=='A')
{

}
else if(blue=='B')
{
   Setpoint=0;
}
else if(blue=='C')
{
   Setpoint=0;
}
else if(blue=='D')
{
   Setpoint=255;
}
else if(blue=='E')
{
  KP=KP+2;
}
else if(blue=='F')
{
  KI=KI+2;
  }
else if(blue=='G')
{
  Setpoint= Setpoint+10;
}
else if(blue=='H')
{
  Setpoint= Setpoint-10;
}
else if(blue=='I')
{
  KD=KD+0.2;
}
```

```
    else if(blue=='J')
    {
      KP=KP-2;
    }
    else if(blue=='K')
    {
      KI=KI-2;
    }
    else if(blue=='L')
    {
      KD=KD-0.2;
    }
    delay(100);
  }
}

//*****************************************************
*********************
// Function tachometer
//*****************************************************
*********************

void tacho(){

    delay(50);
  //detachInterrupt(0);          //detaches the interrupt          //saves
the current time
    time=millis()-oldtime;       //finds the time
    rpm=(rev/time)*3000;         //calculates rpm
    oldtime=millis();
    Serial.print(rpm);
    Serial.print("\t");
    Serial.print(time);
    Serial.print("\t");
    Serial.print(rev);
    Serial.print("\t");
```

```
    rev=0;
}
```

B. Kodingan Untuk Pemrograman Pada Aplikasi *Smartphone* Di Android Studio

```
package com.example.yusril.cobata2;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;

public class Main2Activity extends AppCompatActivity {

   Button btnOn;
    TextView txtArduino, txtString, txtStringLength,
sensorView0, sensorView1;
    Handler bluetoothIn;

    final int handlerState = 0;
//used to identify handler message
    private BluetoothAdapter btAdapter = null;
    private BluetoothSocket btSocket = null;
    private StringBuilder recDataString = new
StringBuilder();

    private ConnectedThread mConnectedThread;

    // SPP UUID service - this should work for most
devices
    private static final UUID BTMODULEUUID =
UUID.fromString("00001101-0000-1000-8000-
```

```
00805F9B34FB");

    // String for MAC address
    private static String address;


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main2);

        //Link the buttons and textViews to respective
views
       btnOn = (Button) findViewById(R.id.button);
       // btnOff = (Button)
findViewById(R.id.buttonOff);
        txtString = (TextView)
findViewById(R.id.stringdata);
        txtStringLength = (TextView)
findViewById(R.id.leg);
        sensorView0 = (TextView)
findViewById(R.id.textView3);
        sensorView1 = (TextView)
findViewById(R.id.textView4);


        bluetoothIn = new Handler() {
            public void
handleMessage(android.os.Message msg) {
                if (msg.what == handlerState)
{                                     //if message is
what we want
                    String readMessage = (String)
msg.obj;
// msg.arg1 = bytes from connect thread
                    recDataString.append(readMessage);
//keep appending to string until ~
                    int endOfLineIndex =
recDataString.indexOf("~");                    //
determine the end-of-line
                    if (endOfLineIndex > 0)
{                                           // make
sure there data before ~
                        String dataInPrint =
```

```
recDataString.substring(0, endOfLineIndex);     //
extract string

txtString.setText(dataInPrint);
                        int dataLength =
dataInPrint.length();                           //get
length of data received

txtStringLength.setText(String.valueOf(dataLength));

                        if (recDataString.charAt(0) ==
'#')                                //if it starts with #
we know it is what we are looking for
                        {
//                              String sensor0 =
recDataString.substring(1, 5);              //get
sensor value from string between indices 1-5
//                              String sensor1 =
recDataString.substring(6, 10);             //same
again...
//                              String sensor2 =
recDataString.substring(11, 15);
//                              String sensor3 =
recDataString.substring(16, 20);
                                String[] dataku =
recDataString.toString().split("-");


sensorView0.setText(dataku[1]);     //update the
textviews with sensor values

sensorView1.setText(String.valueOf(dataku.length));
                                ///sensorView2.setText("
Sensor 2 Voltage = " + sensor2 + "V");
                                //sensorView3.setText("
Sensor 3 Voltage = " + sensor3 + "V");
                        }
                        recDataString.delete(0,
recDataString.length());                    //clear
all string data
                        // strIncom =" ";
                        dataInPrint = " ";
                    }
                }
            }
```

```
        };

        btAdapter =
BluetoothAdapter.getDefaultAdapter();       // get
Bluetooth adapter
        checkBTState();

        // Set up onClick listeners for buttons to
send 1 or 0 to turn on/off LED
//        btnOff.setOnClickListener(new
View.OnClickListener() {
//            public void onClick(View v) {
//                mConnectedThread.write("0");    //
Send "0" via Bluetooth
//                txtString.setText("00");
//                Toast.makeText(getBaseContext(),
"Turn off LED", Toast.LENGTH_SHORT).show();
//            }
//        });
//
        btnOn.setOnClickListener(new
View.OnClickListener() {
            public void onClick(View v) {
                mConnectedThread.write("A");    //
Send "1" via Bluetooth
                Toast.makeText(getBaseContext(), "Turn
on LED", Toast.LENGTH_SHORT).show();
            }
        });
    }

    private BluetoothSocket
createBluetoothSocket(BluetoothDevice device) throws
IOException {

        return
device.createRfcommSocketToServiceRecord(BTMODULEUUID)
;
        //creates secure outgoing connecetion with BT
device using UUID
    }
```

```
    @Override
    public void onResume() {
        super.onResume();

        //Get MAC address from DeviceListActivity via
intent
        Intent intent = getIntent();

        //Get the MAC address from the
DeviceListActivty via EXTRA
        address =
intent.getStringExtra(MainActivity.EXTRA_DEVICE_ADDRES
S);

        //create device and set the MAC address
        BluetoothDevice device =
btAdapter.getRemoteDevice(address);

        try {
            btSocket = createBluetoothSocket(device);
        } catch (IOException e) {
            Toast.makeText(getBaseContext(), "Socket
creation failed", Toast.LENGTH_LONG).show();
        }
        // Establish the Bluetooth socket connection.
        try
        {
            btSocket.connect();
        } catch (IOException e) {
            try
            {
                btSocket.close();
            } catch (IOException e2)
            {
                //insert code to deal with this
            }
        }
        mConnectedThread = new
ConnectedThread(btSocket);
        mConnectedThread.start();

        //I send a character when resuming.beginning
transmission to check device is connected
        //If it is not an exception will be thrown in
```

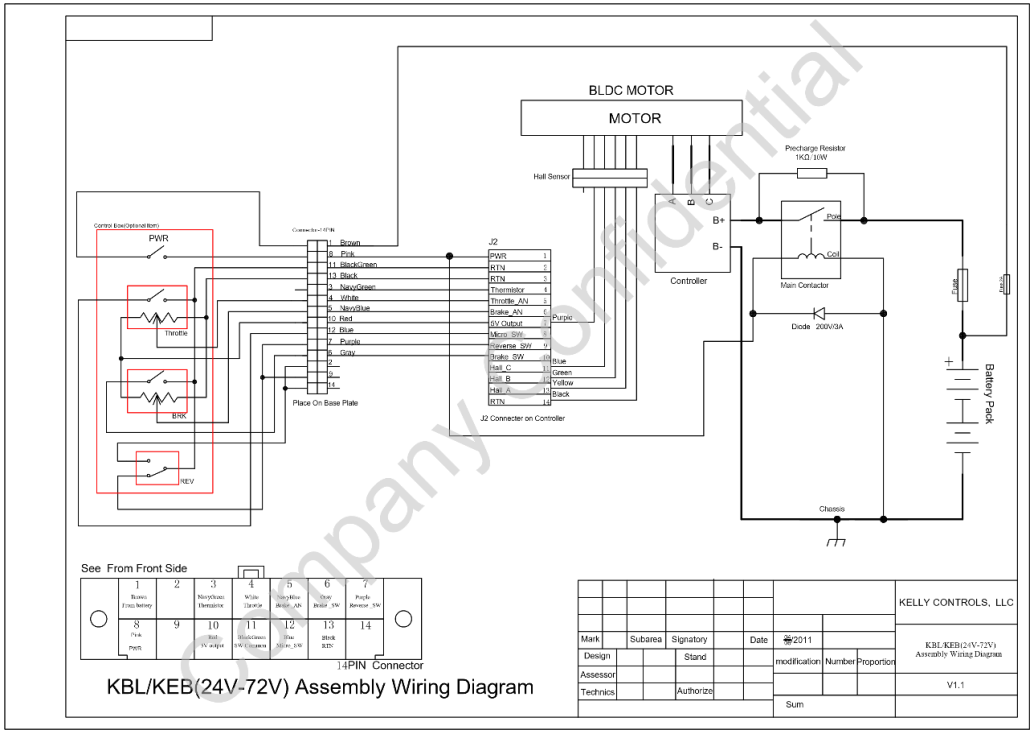
```
the write method and finish() will be called
        //mConnectedThread.write("x");
    }



    @Override
    public void onPause()
    {
        super.onPause();
        try
        {
            //Don't leave Bluetooth sockets open when
leaving activity
            btSocket.close();
        } catch (IOException e2) {
            //insert code to deal with this
        }
    }

    //Checks that the Android device Bluetooth is
available and prompts to be turned on if off
    private void checkBTState() {

        if(btAdapter==null) {
            Toast.makeText(getBaseContext(), "Device
does not support bluetooth",
Toast.LENGTH_LONG).show();
        } else {
            if (btAdapter.isEnabled()) {
            } else {
                Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent,
1);
            }
        }
    }

    //create new class for connect thread
    private class ConnectedThread extends Thread {
        private final InputStream mmInStream;
        private final OutputStream mmOutStream;

        //creation of the connect thread
```

```
        public ConnectedThread(BluetoothSocket socket)
{
            InputStream tmpIn = null;
            OutputStream tmpOut = null;

            try {
                //Create I/O streams for connection
                tmpIn = socket.getInputStream();
                tmpOut = socket.getOutputStream();
            } catch (IOException e) { }

            mmInStream = tmpIn;
            mmOutStream = tmpOut;
        }

        public void run() {
            byte[] buffer = new byte[256];
            int bytes;

            // Keep looping to listen for received
messages
            while (true) {
                try {
                    bytes = mmInStream.read(buffer);
//read bytes from input buffer
                    String readMessage = new
String(buffer, 0, bytes);
                    // Send the obtained bytes to the
UI Activity via handler

bluetoothIn.obtainMessage(handlerState, bytes, -1,
readMessage).sendToTarget();
                } catch (IOException e) {
                    break;
                }
            }
        }
        //write method
        public void write(String input) {
            byte[] msgBuffer = input.getBytes();
//converts entered String into bytes
            try {
                mmOutStream.write(msgBuffer);
//write bytes over BT connection via outstream
            } catch (IOException e) {
```

```
                //if you cannot write, close the
application
                Toast.makeText(getBaseContext(),
"Connection Failure", Toast.LENGTH_LONG).show();
                finish();

            }
        }
    }
}
```

LAMPIRAN C

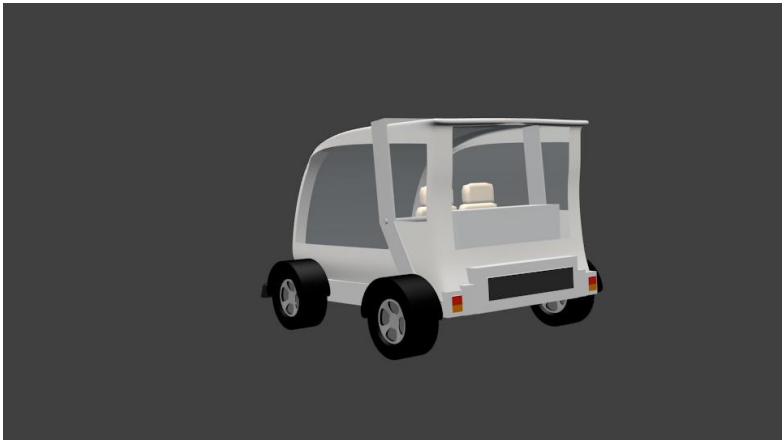
WIRING DIAGRAM KELLY CONTROLLERS



Halaman ini sengaja dikosongkan

LAMPIRAN D

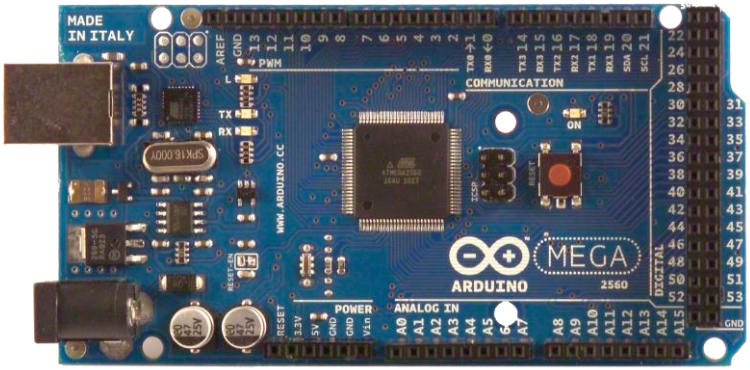
RANCANGAN DESAIN MOBIL OTONOM ACEPITS 1.0



LAMPIRAN E

DATA SHEET ARDUINO MEGA 2560

Arduino MEGA 2560



Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), 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 Mega is

compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index



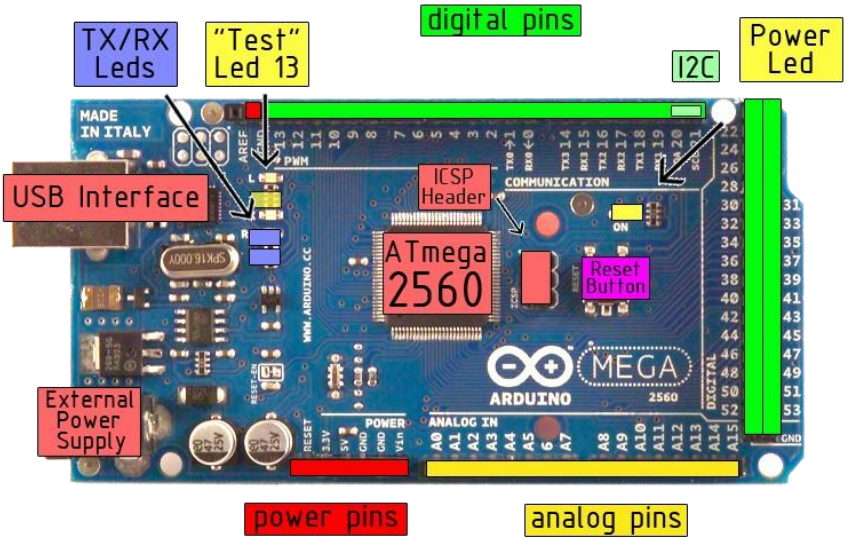
EAGLE files: arduino-mega2560-reference-design.zip
Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader

SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



The Arduino Mega2560 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.

Power

The Mega2560 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.
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 ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega 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); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** 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: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the

underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.

- **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.
- **I²C: 20 (SDA) and 21 (SCL).** Support I²C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I²C pins on the Duemilanove.

The Mega2560 has 16 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 analogReference() function.
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.

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 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 Mega's digital pins.

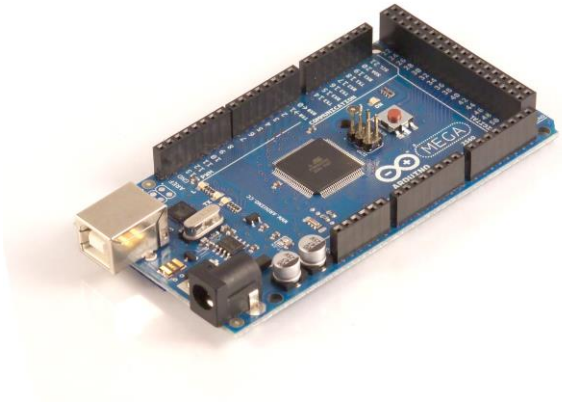
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega 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.



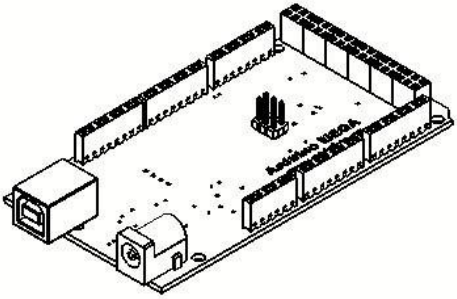
How to use Arduino

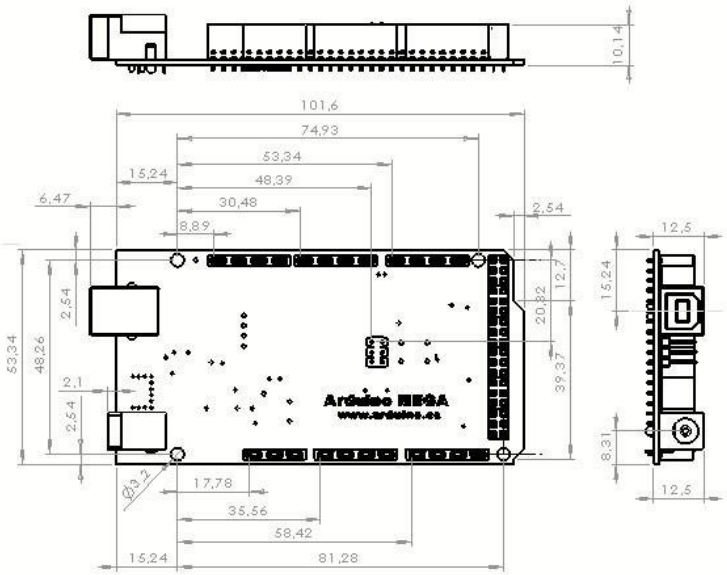
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

Once you have your skecth you'll see something very close to the screenshot on the right.
In **Tools>Board** select MEGA
Now you have to go to **Tools>SerialPort**
and select the right serial port, the one arduino is attached to.

Dimensioned Drawing





LAMPIRAN F

DATA SHEET MODUL BLUETOOTH HC-05

DATASHEET BLUETOOTH TO SERIAL PORT MODULE HC05

Overview

HC-05 module is an easy to use Bluetooth SPP (Serial Port Protocol) module, designed for transparent wireless serial connection setup.
Serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Bluecore 04-External single chip Bluetooth system with CMOS technology and with AFH (Adaptive Frequency Hopping Feature). It has the footprint as small as 12.7mmx27mm. Hope it will simplify your overall design/development cycle.

Specifications

Hardware features

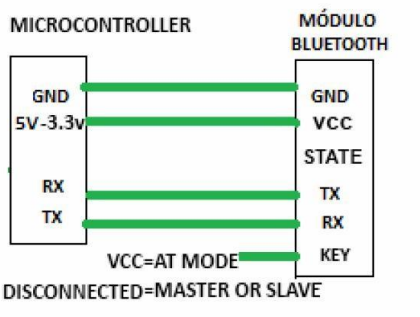
- Typical -80dBm sensitivity.
- Up to +4dBm RF transmit power.
- Low Power 1.8V Operation, 3.3 to 5 V I/O.
- PIO control.
- UART interface with programmable baud rate.
- With integrated antenna.
- With edge connector.

Software features

- Slave default Baud rate: 9600, Data bits:8, Stop bit:1,Parity:No parity.
- PIO9 and PIO8 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing **PINCODE:"1234"** as default.
- Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

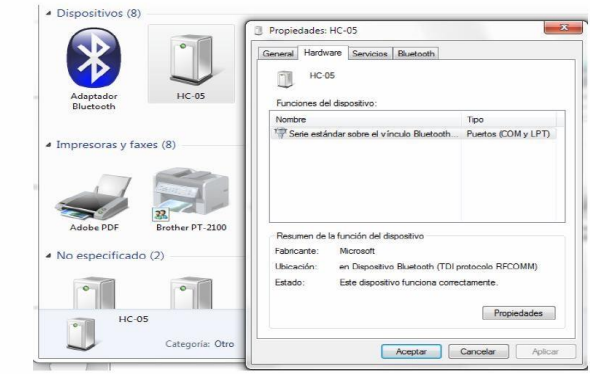
Pin out configuration

Typical Application Circuit



After connect the Bluetooth module, scan for new devices from the PC and you will find the module with the device name “HC-05”, after that, click to connect, if some message appears asking about “Pairing code” just put **“1234”** as default code.

BLUE LED = ACTIVE (Blinking 500ms period inactive connection, change 1seg with active connection)



Open a serial terminal and select the serial COM x port number that assigned Windows to Bluetooth Module.

Configure the serial terminal with these parameters:

- Baud rate: 9600.
- Data bits:8.
- Stop bit:1.
- Parity: No parity.

www.electronica60norte.com
electronica60norte@hotmail.com

BIODATA PENULIS

Nama lengkap penulis adalah Muhammad Yusril Izza, lahir di Kota Surabaya pada tanggal 15 September 1996. Penulis merupakan anak kedua dari tiga bersaudara dari ayah bernama Moch. Munief dan ibu bernama Sriyani Purwati dan memiliki kakak bernama Silvia Rachmawati dan adik bernama Fidyanita Safitri S. Pada tahun 2008 penulis menyelesaikan pendidikan Sekolah Dasar di SD Negeri Jemurwonosari I Surabaya, pada tahun 2011 menyelesaikan pendidikan Sekolah Menengah Pertama di SMP Negeri 13 Surabaya, pada tahun 2014 menyelesaikan pendidikan Sekolah Menengah Atas di SMA Negeri 10 Surabaya. Pada tahun 2014, penulis terdaftar sebagai mahasiswa di Departemen Teknik Fisika Institut Teknologi Sepuluh Nopember.

Penulis telah aktif dalam beberapa organisasi kemahasiswaan dan kepanitiaan diantaranya menjadi pengurus Admin Laboratorium Simulasi dan Komputasi E205, panitia Gerigi ITS 2015 dan panitia *Engineering Physics Week* 2016 sekaligus menjadi tim IT pada acara tersebut .

Konsentrasi tugas akhir yang didalami adalah bidang rekayasa intrumentasi dan kontrol. Pada bulan Juni 2018 penulis telah menyelesaikan Tugas Akhir dengan judul **"RANCANG BANGUN PENGENDALIAN KECEPATAN PROTOTIPE MOBIL OTONOM ACEPITS 1.0 DENGAN METODE PID BERBASIS IOT (*INTERNET OF THINGS*)"**.

Apabila pembaca ingin berdiskusi lebih lanjut mengenai tugas akhir, serta memberikan kritik dan saran maka dapat menghubungi penulis melalui *email* : myusrilizza@gmail.com