TUGAS AKHIR – EE 184801

SISTEM KEAMANAN PINTU PARKIR MENGGUNAKAN TEKNOLOGI *RFID*

Faishal Abdur Rahman
NRP 07111540000106

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.

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

TUGAS AKHIR – EE 184801

SISTEM KEAMANAN PINTU PARKIR MENGGUNAKAN TEKNOLOGI *RFID*

Faishal Abdur Rahman
NRP 07111540000106

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.

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

FINAL PROJECT – EE 184801

PARKING DOOR SECURITY SYSTEM USING RFID TECHNOLOGY

Faishal Abdur Rahman
NRP 07111540000106

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

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

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "SISTEM KEAMANAN PINTU PARKIR MENGGUNAKAN TEKNOLOGI *RFID*" adalah benar-benar hasil karya intelektual sendiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya orang lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun durujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Surabaya, Juni 2019

Faishal Abdur Rahman
NRP. 07111540000106

Halaman ini sengaja dikosongkan.

Sistem Keaman Pintu Parkir Menggunakan Teknologi *RFID*

TUGAS AKHIR

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

Menyetujui:

Dosen Pembimbing,

Dr. Muhammad Rivai, S.T., M.T.
NIP. 196904261994031003

SURABAYA
Juli, 2019

Halaman ini sengaja dikosongkan.

Sistem Keamanan Pintu Parkir Menggunakan Teknologi *RFID*

Nama : Faishal Abdur Rahman
Pembimbing : Dr. Muhammad Rivai, ST., MT.

ABSTRAK

Seiring dengan berkembangnya zaman, keamanan menjadi salah satu fokus utama. Salah satu tempat yang membutuhkan keamanan adalah lahan parkir. Kendalanya adalah semakin hari jumlah kendaraan semakin bertambah sehingga kebutuhan lahan parkir semakin meningkat. Dikarenakan meningkatnya kebutuhan lahan parkir, maka kebutuhan tenaga penjaga lahan parkir juga semakin bertambah. Pada tugas akhir ini telah dirancang dan direalisasikan sebuah sistem yang bisa mengurangi kebutuhan tenaga penjaga sekaligun meningkatkan keamanan lahan parkir. Sistem ini berbasis *Radio Frequency Identification* (*RFID*), yang memungkinkan autentikasi berlapis, sehingga keamanan meningkat tanpa dibutuhkan tenaga penjaga yang lebih. Sistem *database* berbasis MySQL yang dapat memantau waktu masuk dan keluarnya sebuah kendaraan dari lahan parkir. Semua sistem tersebut dikontrol dengan Raspberry Pi dikarenakan kapabilitasnya yang mempunyai koneksi wifi sehingga bisa digunakan sebagai host webserver dengan mudah. *Single Board Computer* ini juga mempunyai 4 USB port yang masing-masing bisa digunakan untuk komunikasi serial, sehingga cocok digunakan sebagai "otak" utama dalam sistem ini. Mikrokontroler Arduino mega digunakan untuk membaca sensor ultrasonik yang mendeteksi keberadaan kendaraan. Mikrokontoler Arduino Nano digunakan untuk membaca MFRC522 *RFID Reader*, dan mengatur gerak palang parkir. Motor DC digunakan untuk menggerakkan palang pintu parkir. Pergerakan palang pintu ini dikontrol menggunakan metode PID dengan *feedback* sudut yang didapat dari pembacaan sensor *IMU* MPU6050. Berdasarkan hasil penelitian ini, tingkat keberhasilan sistem yang dibuat sebesar 80% dari 10 kali percobaan. Untuk palang pintu yang dibuat membutuhkan 0.5 detik untuk membuka dan menutup, lebih baik dari palang pintu konvensional yang rata-rata membutuhkan 1.5 detik.

Kata kunci: Keaman Pintu Parkir, Kontrol PID, MySQL, Teknologi *RFID*

Halaman ini sengaja dikosongkan.

Parking Door Security *System* using *RFID* Technology

Name : Faishal Abdur Rahman
Supervisor : Dr. Muhammad Rivai, ST., MT.

ABSTRACT

Over the years, security has become a main focus for most people. One of the many places that needs security are parking spaces. The obstacle is the amount of vehicles increases day by day, and because of that the needs for parking spaces keeps on increasing, thus the demands for security worker increases as well. In this thesis, a system which can reduce the demands for security workers while increasing security level has been created. This system is based on RFID (Radio Frequency Identification) technology, which enables multiple layers of authentication level thus increasing the security. MySQL based database system used in this thesis makes it possible to keep track on when someone goes in and out of the parking space. All of those systems are controlled using a Raspberry Pi, because of its wifi capabilities making it very suitable to be set up as a webserver host. This single board computer also has 4 USB ports which all of them can be used for serial communication, thus making this device the perfect candidate for the "brain" of this project. Microcontroller Arduino Mega is used to read the value of the Ultrasonic Sensors which are used to detect the presence of vehicles. Mikrokontroler Arduino Nano is used to read from MFRC522 RFID Reader, and control the movement of the barrier gate. DC motor is used as the actuator for the barrier gate. The movement of the barrier gate is controlled using the PID control mechanism, with the feedback being the angle of the barrier gate, acquired using the MPU6050 IMU sensor. Based on the result of this research, the success rate of this system is 80% from the total of 10 experiments. As for the created barrier gate, the time it takes to open and close the gate is 0.5 seconds, better than conventional barrier gate which usually needs 1.5 seconds to open and close.

Keywords: MySQL, Parking Spaces Security, PID control, RFID technology

Halaman ini sengaja dikosongkan.

KATA PENGANTAR

Alhamdulillah, puji syukur saya panjatkan ke hadirat Allah SWT, karena berkat limpahan karunia dan rahmatNya penelitian yang berjudul sistem keamanan pintu parkir menggunakan teknologi *RFID* ini dapat terselesaikan tepat waktu. Dalam menjalankan proses penelitian ini, penulis mendapatkan bantuan dari berbagai pihak. Oleh karena itu penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada berbagai pihak yang mendukung dan membantu dalam penelitian ini, diantaranya :

1. Dr. Muhammad Rivai, S.T., M.T. selaku dosen pembimbing, atas bimbingan, inspirasi, pengarahan, dan motivasi yang diberikan selama pengerjaan tugas akhir ini.
2. Dr.Eng., Ir. Totok Mujiono, M.IKom., Ir. Harris Pirngadi, M.T., Fajar Budiman, S.T., M.Sc., Muhammad Attamimi, B.Eng., M.Eng., Ph.D. selaku dosen penguji yang telah memberikan kritik dan saran.
3. Dr.Eng. Ardyono Priyadi, S.T., M.Eng. selaku kepala departemen Teknik Elektro ITS, tempat penulis menempuh studi Strata 1.
4. Orang tua dan keluarga dari penulis yang senantiasa memberikan dukungan baik moril dan materiil.
5. Seluruh dosen dan karyawan Departemen Teknik Elektro ITS yang telah memberikan banyak ilmu dan menciptakan suasana belajar yang nyaman.
6. Teman-teman laboratorium Elektronika Dasar B202 yang senantiasa membantu dan memberikan dukungan dalam mengerjakan penelitian ini.
7. @vptlk yang senantiasa menemani melalui malam-malam panjang pengerjaan penelitian ini

Penulis sadar bahwa penelitian ini masih belum sempurna dan masih banyak hal yang perlu diperbaiki. Semoga buku ini dapat memberikan manfaat bagi pembaca dan dapat digunakan sebagai referensi untuk percobaan serupa kedepannya.

Surabaya, 25 Juni 2019

Faishal Abdur Rahman

Halaman ini sengaja dikosongkan.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan.

BAB I
PENDAHULUAN

1.1. Latar Belakang

Dalam kehidupan manusia, otomasi adalah sesuatu yang tidak bisa dihindari. Seiring dengan berkembangnya zaman maka akan semakin bertambah hal-hal dalam kehidupan manusia yang akan terotomasi, termasuk keamanan. Kemanan adalah hal yang sudah sangat berubah seiring dengan berkembangnya zaman. Zaman dahulu penjaga keamanan harus berkeliling mengitari area yang dijaganya untuk memastikan area tersebut aman, namun sekarang karena sudah adanya teknologi CCTV maka seorang penjaga keamanan tidak harus sering-sering berkeliling, dia sudah bisa mengawasi area yang sebelumnya harus dia kelilingi melalui siaran langsung sistem CCTV. Begitu juga dengan tempat parkir, tidak bisa dipungkiri kalau semakin hari manusia semakin bergantung pada alat transportasi baik itu mobil, sepeda motor, atau yang lainnya. Semakin lama lahan parkir yang dibutuhkan pun tidak akan semakin berkurang, namun semakin bertambah, dan bertambahnya lahan parkir berarti bertambah pula kebutuhan tenaga untuk menjaga lahan parkir tersebut. Berdasarkan data yang dimiliki oleh BPS (Badan Pusat Statistik), jumlah kendaraan bermotor yang ada di Indonesia semakin bertambah tiap tahunnya. Berjumlah 76.907.127 pada tahun 2010, dan bertambah menjadi 138.556.669 pada tahun 2017 [1]. Dikarenakan banyaknya kebutuhan tersebut, maka sistem keamanan parkir otomatis atau *smart parking system* dirasa dapat menyelesaikan masalah tersebut. Dengan sistem keamanan parkir otomatis maka tenaga yang diperlukan untuk menjaga suatu lahan parkir dapat dikurangi jumlahnya dengan drastis, dan tingkat keamanannya bisa meningkat.

Smart parking system sendiri mempunyai berbagai implementasi yang berbeda. Salah satunya menggunakan *RFID* juga sebagai alat autektikasi untuk memberi akses kepada pengendara [2]. RFID yang digunakan tidak hanya satu macam, ada juga sistem yang menggunakan RFID berfrekuensi tinggi sehingga jarak pembacaannya mencapai 3 meter [3]. Implementasi lain adalah menggunakan sensor ultrasonik untuk membuka palang pintu secara otomatis. Selain itu di sistem yang sama sensor ultrasonik juga digunakan untuk memantau ketersediaan lahan parkir [4].

Pada tugas akhir ini akan dikembangkan sebuah *smart parking system* yang menggunakan 2 buah sensor *RFID* sebagai alat autentikasi. Kemudian data rekaman akses lahan parkir akan disimpan kedalam database sehingga dapat diakses dengan mudah. Palang yang dipakai dalam sistem ini akan menggunakan sebuah motor DC, yang akan dikontrol dengan kontrol PID.

1.2. Perumusan Masalah

Permasalahan yang menjadi dasar dilakukannya penelitian ini dapat dirumuskan dalam poin-poin beriku :

1. Bagaimana cara mengetahui kendaraan yang masuk dan keluar dari lahan parkir secara otomatis.
2. Bagaimana cara agar data kendaraan yang masuk dan keluar dari lahan parkir tersebut dapat diakses dengan mudah.
3. Bagaimana cara agar pergerakan palang pintu bisa dikendalikan.

1.3. Tujuan Penelitian

Penelitian pada tugas akhir ini bertujuan sebagai beriku :

1. Autentikasi setiap orang yang masuk dan keluar lahan parkir menggunakan teknologi *RFID*
2. Sistem *database* MySQL untuk menyimpan data setiap orang yang masuk dan keluar lahan parkir
3. Pengendalian pergerakan palang pintu menggunakan kontrol PID, dengan input *feedback* hasil pembacaan sensor IMU.

1.4. Batasan Masalah

Berdasarkan masalah yang telah dirumuskan maka hasil yang diharapkan dari penelitian ini adalah sebagai berikut :

1. *Database* disimpan dan hanya bisa diakses dalam jaringan lokal, tidak bisa online.
2. Palang pintu yang digunakan berjumlah satu untuk masuk dan keluar.

1.5. Metodologi Penelitian

Langkah-langkah yang dikerjakan pada tugas akhir ini adalah sebagai berikut :

1. **Studi Literatur**

Studi literatur akan berisi pengumpulan dan pengkajian teori, data dan penelitian yang dianggap relevan dan terpercaya untuk mendukung keabsahan tugas akhir ini. Literatur yang digunakan akan memiliki batasan-batasan tertentu. Yaitu, literatur yang digunakan harus bersumber dari paper, jurnal, buku, maupun artikel yang berisi tentang bagaimana cara penggunaan *UHF RFID Reader*, pemrograman menggunakan Python, dan cara membuat *database* menggunakan MySQL

2. **Perancangan Keseluruhan Sistem**

Pada dasarnya sistem ini terdiri dari 2 komponen utama, aktuator palang pintu dan perangkat *web*server. Pada bodi palang pintu terdapat *Reader RFID*, motor DC, dan arduino nano. Sedangkan untuk perangkat *web*server memerlukan Raspberry Pi dan monitor serta perangkat input untuk antarmukanya.

3. **Perancangan dan Pembuatan Perangkat Sistem Autentikasi menggunakan *RFID***

Perangkat yang akan dibuat disini terdiri dari 2 bagian utama, yaitu modul pembaca kartu *RFID* dan juga mikrokontroler yang akan menjembatani antara pembaca *RFID* dengan Raspberry Pi

4. **Perancangan dan Pembuatan *Database* MySQL dengan Raspberry Pi**

MySQL adalah software yang mengelola *database*, biasa disebut *Database Management System*, atau *DBMS*. Selain MySQL masih banyak software *DBMS* yang lainnya seperti PostgreSQL, Orale, Microsoft SQL *Server*, dan lain lain, namun pada tugas akhir ini telah diputuskan untuk menggunakan MySQL sebagai *DBMS*.

5. **Perancangan dan Pembuatan Palang Pintu Parkir**

Pada tahap ini dilakukan perancangan dan pembuatan palang pintu parkir. Palang pintu ini nantinya akan terbuat dari aluminium dengan dimensi 150cm x 4.5cm x 2cm.

6. **Pengujian Kontrol PID**

PID kontrol adalah salah satu metode untuk mengontrol pergerakan dari sebuah motor sehingga motor tersebut bergerak sesuai dengan keinginan kita. Pada tahap ini akan dilakukan pengujian terhadap parameter PID kontrol dan efeknya terhadap pergerakan motor

7. Analisa Data dan Evaluasi

Pada tahap ini, akan dilakukan Analisa terhadap data yang didapatkan. Sehingga didapatkan karakteristik dari hardware yang telah dirancang dan dibuat. Analisa dilakukan khususnya pada palang pintu apakah gerakannya sudah sesuai harapan yang diinginkan, apakah sistem kontrolnya berjalan sesuai harapan dan bisa menghadapi skenario-skenario yang mungkin terjadi di lapanan. Lalu dilakukan Analisa terhadap data yang dihasilkan oleh keseluruhan sistem, apakah sudah dianggap memuaskan atau tidak. Lalu dilakukan evaluasi serta revisi desain apabila diperlukan.

8. Penyusunan Laporan Akhir

Penyusunan laporan akan dilakukan seiring dengan tahap-tahap lainnya. Isinya berkaitan dengan tugas akhir yang dikerjakan, meliputi pendahuluan, studi literatur, perancangan dan pembuatan sistem, pengujian dan analisa serta penutup.

9. Penulisan Makalah Jurnal POMITS

Penulisan jurnal dilakukan sebagai sarana publikasi penelitian. Hal ini juga dilakukan agar penelitian mendapatkan legalitas dan pengakuan resmi dari dunia pendidikan.

1.6. Sistematika Penulisan

Dalam buku tugas akhir ini, pembahasan mengenai sistem yang dibuat terbagi menjadi Lima Bab dengan sistematika penulisan sebagai berikut:

- Bab I : Pendahuluan
  Bab ini meliputi penjelasan latar belakang, rumusan masalah, batasan masalah, tujuan, metodologi, sistematika penulisan, dan relevansi
- Bab II : Tinjauan Pustaka
  Bab ini menjelaskan tentang teori penunjang yang terkait maupun yang dibutuhkan dalam pengerjakan tugas akhir ini. Dasar teori yang menunjang meliputi sensor LIDAR, sensor GPS, dan motor servo.
- Bab III : Perancangan Sistem
  Bab ini menjelaskan tentang bagaimana merangkai tiap rangkaian sehingga menjadi satu kesatuan dari sistem sehingga tercapai tujuan dari penelitian ini.

- Bab IV : Pengujian dan Analisis
  Bab ini menjelaskan tentang hasil uji coba alat beserta analisisnya.
- Bab V : Penutup
  Bab ini berisi tentang kesimpulan yang diperoleh dari pembuatan alat serta saran untuk pengembangan lebih lanjut.

1.7. Relevansi

Sistem keamanan parkiran menggunakan *RFID* ini adalah suatu inovasi untuk membantu institusi atau siapapun yang mempunyai lahan parkiran namun ingin meminimalisir jumlah staff keamanan yang dibutuhkan. Ditambah dengan *RFID Reader* jarak jauh yang menambah kepraktisan pemakaian sistem keamanan ini.

Halaman ini sengaja dikosongkan.

BAB II
TINJAUAN PUSTAKA & DASAR TEORI

2.1. *RFID*

RFID adalah istilah umum untuk teknologi yang menggunakan transponder untuk mengidentifikasi suatu objek. Teknologi ini punya sejarah yang berawal dari abad 20an dan awalnya digunakan untuk mengidentifikasi pesawat militer, apakah pesawat itu bagian dari teman atau musuh. Sekarang, teknologi *RFID* telah berhasil dimanfaatkan di berbagai bidang seperti keamanan, logistik, maintenance, dan yang lainnya [5].

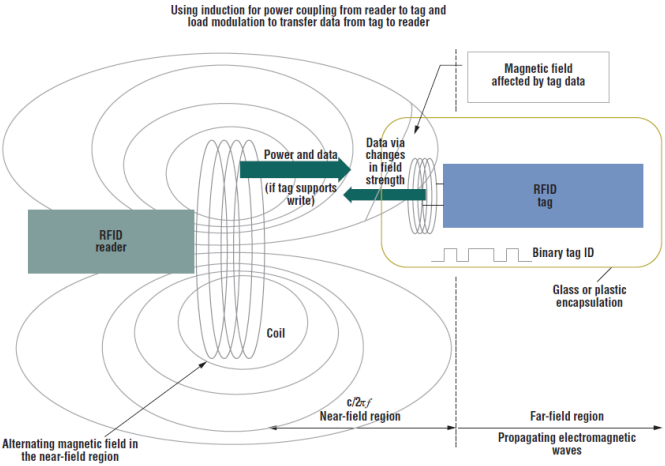
Hardware dari *RFID* sendiri meliputi 2 hal yaitu *RFID tag*s dan *RFID Reade*rs. *RFID Reade*rs adalah perangkat yang berfungsi untuk membaca data yang dibawa oleh *RFID tag*s melalui sinyal radio. Terdapat berbagai versi dan macam *RFID Reader* yang dibedakan berdasarkan aplikasinya, seperti *mobile reader* untuk pembacaan manual, atau gerbang *RFID* yang berfungsi untuk memantau barang-barang yang melalui gerbang itu [6].

Di dalam *RFID tag* terdapat sebuah *chip* sebagai tempat penyimpan data dan komputasi, dan sebuah antena untuk komunikasi. *RFID tag*s bisa dibagi menjadi beberapa kategori berdasarkan sumber energi, kapasitas penyimpanan data, dan frekuensi komunikasi [6].

Pada umumnya *RFID tag* dibagi menjadi 2 macam, yaitu *tag* pasif dan *tag* aktif. *Tag* aktif membutuhkan sumber tenaga, bisa disambungkan ke sumber *list*rik ataupun menggunakan baterai. Pada kasus ini umur sebuah *tag* bergantung pada masa tahan baterainya, ditentukan oleh seberapa sering *tag* tersebut dibaca oleh *Reader*. Namun, harga baterai, ukuran, dan umur aktif *tag* membuatnya tidak praktis digunakan di industri retail. Untuk keperluan ini *tag* pasif jauh lebih disukai karena tidak memerlukan baterai dan umurnya yang sangat panjang. *Tag* pasif juga hampir tidak memerlukan perawatan dan ukurannya sangat kecil sehingga bisa masuk ke kertas label [7].

Prinsip induksi magnetis faraday adalah dasar dari komunikasi jarak pendek antara *RFID Reader* dan *tag*, seperti ditunjukkan pada Gambar 2.1. *Reader* mengalirkan arus bolak-balik yang besar melalui lilitan pembaca, sehingga menimbulkan medan magnet bolak-balik disekitarnya. Jika *tag* dengan lilitan yang lebih kecil diletakkan di medan magnet ini, maka tegangan bolak-balik akan muncul di lilitan *tag* ini. Jika tegangan

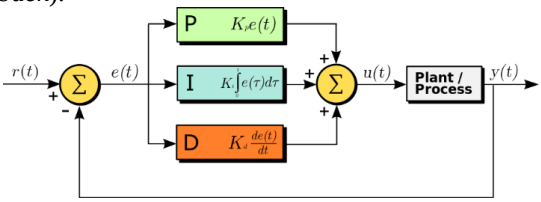
bolak-balik ini disearahkan, maka akan bisa digunakan untuk menyalakan chip yang ada di *tag* tersebut. Lalu chip pada *tag* tersebut melalui coupling jarak dekat mengirimkan data kembali pada *Reader* dengan cara modulasi beban sehingga *Reader* dapat membaca isi data pada chip di *tag*.



Gambar 2.1 Skema Kerja Teknologi *RFID* [7].

2.2. PID

Sistem Kontrol PID (*Proportional–Integral–Derivative* controller) merupakan kontroler untuk menentukan presisi suatu sistem instrumentasi dengan karakteristik adanya umpan balik pada sistem tesebut (*Feedback*).



Gambar 2.2 Diagram Blok PID Control [8].

Sistem kontrol PID terdiri dari tiga buah cara pengaturan yaitu kontrol P (*Proportional*), D (*Derivative*) dan I (*Integral*), seperti ditunjukkan pada Gambar 2.2, dengan masing-masing memiliki kelebihan dan kekurangan. Dalam implementasinya masing-masing cara dapat

bekerja sendiri maupun gabungan diantaranya. Dalam perancangan sistem kontrol PID yang perlu dilakukan adalah mengatur parameter P, I atau D agar tanggapan sinyal keluaran *system* terhadap masukan tertentu sebagaimana yang diinginkan. Berikut adalah persamaan yang digunakan dalam kontroler ini.

$$mv(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \quad (2.1)$$

Dimana mv(t) adalah output dari pengontrol PID, atau Manipulated Variable, K_p adalah konstanta proporsional, K_i adalah konstanta *integral*, K_d adalah konstanta derivatif, dan e(t) adalah *error* (selisih antara set point dengan level aktual).

Kp berlaku sebagai Gain (penguat) saja tanpa memberikan efek dinamik kepada kinerja kontroler. Penggunaan kontrol P memiliki berbagai keterbatasan karena sifat kontrol yang tidak dinamik ini. Walaupun demikian dalam aplikasi-aplikasi dasar yang sederhana kontrol P ini cukup mampu untuk memperbaiki respon transien khususnya *rise time* dan *settling time*. Pengontrol proporsional memiliki keluaran yang sebanding/proporsional dengan besarnya sinyal kesalahan (selisih antara besaran yang diinginkan dengan harga aktualnya).

Keluaran pengontrol *integral* merupakan hasil penjumlahan yang terus menerus dari perubahan masukannya. Jika sinyal kesalahan tidak mengalami perubahan, maka keluaran akan menjaga keadaan seperti sebelum terjadinya perubahan masukan. Sinyal keluaran pengontrol *integral* merupakan luas bidang yang dibentuk oleh kurva kesalahan atau *error*.

Keluaran pengontrol diferensial memiliki sifat seperti halnya suatu operasi derivatif. Perubahan yang mendadak pada masukan pengontrol akan mengakibatkan perubahan yang sangat besar dan cepat. Ketika masukannya tidak mengalami perubahan, keluaran pengontrol juga tidak mengalami perubahan, sedangkan apabila sinyal masukan berubah mendadak dan menaik (berbentuk fungsi *step*), keluaran menghasilkan sinyal berbentuk impuls. Jika sinyal masukan berubah naik secara perlahan (fungsi ramp), keluarannya justru merupakan fungsi step yang besar magnitudenya sangat dipengaruhi oleh kecepatan naik dari fungsi ramp dan factor konstanta Kd [8].

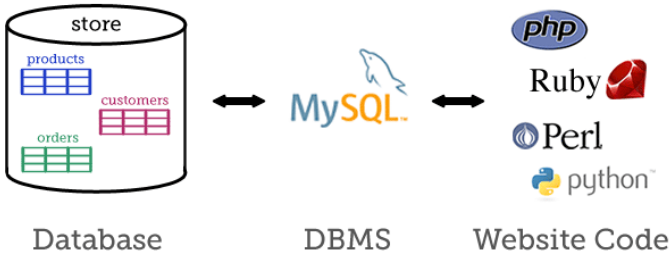
2.3. *Database*

Database adalah koleksi informasi yang dikumpulkan dan diatur sedemikian rupa sehingga mudah diatur dan diperbarui. Data diatur

menggunakan baris, kolom, tabel, dan diberi *index* sehingga lebih mudah untuk menemukan informasi yang relevan. Data diperbarui, ditambah, dan dihapus seiring dengan berjalannya waktu. *Database* memproses pekerjaan dan perintah sehingga ia bisa memperbarui dirinya sendiri.

Software yang mengatur sebuah *database* disebut "*database* management *system*" atau "DBMS". MySQL adalah salah satu contoh dari DBMS. Yang terkadang membingungkan adalah seringkali DBMS juga disebut "*database*", padahal faktanya "*database*" adalah data itu sendiri dan DBMS adalah software yang mengatur dan mengolah data tersebut. Hubungan *database*, DBMS, dan *web*site *code* ditunjukkan di Gambar 2.3.

Ada banyak sekali cara untuk mengatur dan mengolah data dalam suatu *database*, disebut dengan model *database*. Salah satu model *database* yang paling populer adalah model *relational*, yaitu model yang digunakan oleh MySQL (digunakan juga oleh PostgreSQL, Oracle, Microsoft SQL *Server*, dan sistem umum yang lainnya. Karena itu MySQL disebut juga sebagai "*Relational Database Management System*" atau RDBMS.



Gambar 2.3 Hubungan antara *Database*, DBMS, dan Kode *Web*site

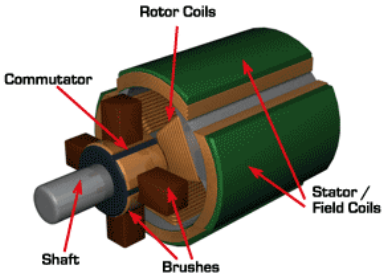
2.4. Motor DC

Motor DC adalah motor listrik yang memerlukan suplai tegangan arus searah pada kumparan medan untuk diubah menjadi energi gerak mekanik. Kumparan medan pada motor dc disebut stator dan kumparan jangkar disebut rotor. Motor arus searah menggunakan arus searah, dan memiliki 3 bagian utama untuk dapat berputar yaitu kutub medan, dinamo, dan komutator, seperti yang diilustrasikan pada Gambar 2.4.

- Kutub Medan : Motor DC sederhana memiliki 2 kutub medan : kutub utara dan kutub selatan. Garis magnetik energi membesar melintasi ruang terbuka diantara kutub-kutub dari utara ke selatan.

Untuk motor yang lebih besar atau lebih komplek terdapat lebih banyak kutub.

- Dinamo : Dinamo yang berbentuk silinder, dihubungkan ke penggerak untuk menggerakkan beban. Untuk kasus motor DC yang kecil, dinamo berputar dalam medan magnet yang dibentuk oleh kutub-kutub, sampai kutub utara dan selatan magnet berganti lokasi.
- Komutator : Komponen ini terutama ditemukan dalam motor DC, berfungsi untuk transmisi arus antara dinamo dan sumber daya.



Gambar 2.4 Motor DC dan Bagian-bagiannya [9].

2.5. Torsi

Torsi, atau momen, adalah gaya rotasi. Seperti konsep gaya linear yang berarti dorong atau tarik, torsi bisa dianggap sebagai gaya yang menyebabkan sebuah objek berputar. Simbol dari torsi adalah τ, huruf kecil yunani yang disebut tau. Rumus dari torsi sendiri adalah

$$\tau = \mathrm{r} \times \mathrm{F} \quad (2.2)$$

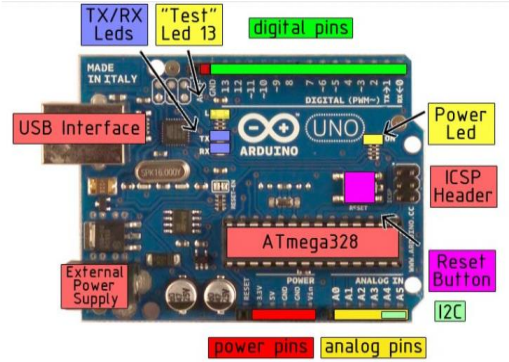
$$\tau = |\mathrm{r}| \times |F| \times sin\theta \quad (2.3)$$

Dimana τ adalah torsi, r adalah jarak dari titik gaya ke sumbu putar, dan F adalah besar gaya. Rumus pertama digunakan jika arah gaya tegak lurus dengan objek yang diputar, sehingga menyebabkan θ bernilai 1. Untuk rumus kedua digunakan jika arah gaya tidak tegak lurus terhadap objek yang diputar sehingga harus dicari nilai sinθnya terlebih dahulu [10].

2.6. Arduino

Arduino adalah sebuah alat untuk membuat komputer yang lebih bisa mengontrol dan merasakan dunia fisik di sekitarnya daripada komputer pada umumnya. Atau sebuah platform komputasi fisik yang open-source, didasarkan pada sebuah board mikrokontroler, dan juga

development environment yang digunakan untuk menulis software ke board tersebut. Arduino dapat digunakan untuk mengembangkan objek interaktif, mengambil input dari berbagai sensor, dan juga mengendalikan berbagai lampu, motor, dan aktuator fisik lainnya. Proyek arduino bisa jadi stand-alone (berdiri sendiri), atau bisa juga berkomunikasi menggunakan software yang berjalan di komputer pengguna (seperti Flash, MaxMSP, software prosesing, dll). Papan arduino bisa dibeli dengan keadaan sudah terpasang, atau bisa dipasang sendiri pada board yang dimiliki oleh pengguna. IDE (*Integrated Development Environment*) dari arduino dapat didownload secara gratis di website resminya. Bahasa pemrograman yang digunakan oleh arduino adalah salah satu implementasi dari "Wiring", platform komputasi fisik yang mirip dengan arduino, dan didasarkan pada "Processing Multimedia Programming Environment". Diagram blok dari arduino uno ditunjukkan pada Gambar 2.5 [11].

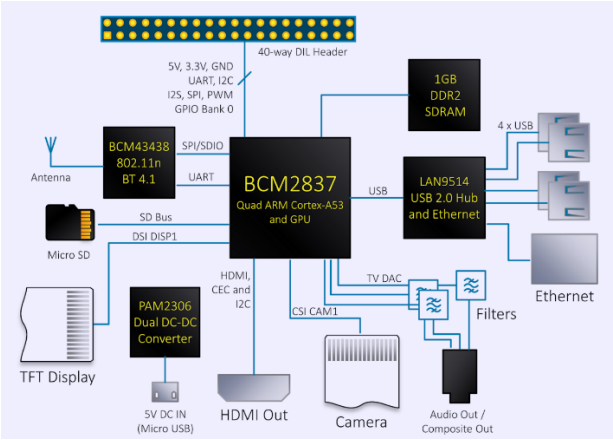


Gambar 2.5 Bagian-bagian Arduino Uno [11].

2.7. Raspberry Pi

Raspberry Pi adalah board komputer berukuran kecil, murah, kuat, bisa digunakan untuk berbagai aplikasi, berorientasi pada pendidikan, yang dikenalkan pada tahun 2012. Cara pengoperasiannya sama dengan komputer pada umumnya, memerlukan sebuah keyboard, mouse, monitor untuk display output, dan juga sumber tenaga atau power supply. Komputer berukuran kartu kredit ini mempunyai performa yang tinggi, dan dihargai antara 25-35USD, sehingga merupakan platform yang cocok untuk komunikasi dengan berbagai divais. Sebagian besar komponen dari sistem ini seperti CPU, GPU, audio, hardware komunikasi, dan juga 1GB

chip memory sudah diintegrasikan menjadi satu komponen tunggal. Prosesor pada Raspberry Pi adalah 64 bit, 1.2GHz SoC, Broadcom BCM2837. Micro SD memori flash digunakan sebagai hard drive pada divais ini. Raspberry Pi ini mendapatkan sumber listrik dari port micro-USB yang bisa menerima sumber dengan arus maksimal 2.5 Ampere. Pada Gambar 2.6 ditunjukkan diagram blok dari Raspberry Pi 3B [12].



Gambar 2.6 Diagram Blok Raspberry Pi 3B [12].

2.8. Flask

Flask adalah sebuah micro *web framework* yang ditulis dalam bahasa Python. Flask disebut sebagai *microframework* karena dia tidak membutuhkan komponen atau library khusus yang lain. Flask tidak mempunyai *database abstraction layer*, *form validation*, atau komponen-komponen lain yang biasa dimiliki oleh library-library yang lain. Namun Flask mendukun ekstensi yang bisa menambahkan fitur aplikasi seakan-akan aplikasi tersebut diimplementasikan dalam Flask itu sendiri. Ekstensi yang tersedia diantaranya adalah yang digunakan untuk object-*relational* mappers, form validation, upload handling, beberapa teknologi open authentication, dan beberapa *framework* yang lain. Ekstensi-ekstensi tersebut lebih sering diperbarui daripada Flask itu sendiri. Contoh website yang menggunakan Flask adalah Pinterest, LinkedIn, dan *web* komunitas untuk Flask itu sendiri.

Flask dibuat oleh Armin Ronacher dari Pocoo, perkumpulan internasional yang terdiri dari pecinta Python yang dibentuk pada 2004.

Menurut Ronacher, ide pembuatan Flask bermula sebagai lelucon april mop, namun lelucon tersebut menjadi cukup populer sehingga direalisasikan menjadi suatu aplikasi yang serius.

Saat Ronarcher dan Georg Brandl membuan sistem papan bulletin yang ditulin di bahasa Python, Project Werkzeug dan Jinja milik Pocoo juga dikembangkan. Meskipun tidak mempunyai rilis yang besar, Flask telah menjadi populer di kalangan pecinta Python. Pada pertengahan tahun 2016, Flask menjadi Python *web* development *framework* yang paling populer di GitHub.

Flask *Microframework* didasarkan pada Werkzeug dan Jinja2 yang merupakan project milik Pocoo juga. Werkzeuh adalah utility library untuk bahasa pemrograman Python. Dalam kata lain, adalah sebuah toolkit untuk aplikasi *Web Server* Gateway Interface (WSGI), dan telah dilisensi dibawah BSD License. Werkzeug bisa merealisasikan objek software untuk request, response, dan utility functions. Werkzeug juga bisa digunakan untuk membangun custom software *framework* sendiri diatasnya dan kompatibel dengan Python 2.6, 2.7, 3.3 [13].

Jinja, dibuat juga oleh Ronacher, adalah sebuah template engine untuk bahasa pemrograman Python dan telah dilisensi dibawah BSD License. Mirip seperti *web framework* Django, template yang disediakan Jinja dievaluasi didalam sandbox. Fitur-fitur dari Flask itu sendiri adalah :

- Mempunyai development *server* dan debugger
- Dukungan untuk unit testing yang telah terintegrasi
- RESTful request dispatching
- Menggunakan template Jinja2
- Mendukung secure cookies
- 100% sesuai dengan WSGI 1.0
- Mempunyai banyak dokumentasi
- Kompatibel dengan Google App Engine
- Bisa menggunakan ekstensi untuk menambah fitur.

2.9. HTML

Hypertext Markup Language atau yang biasa disingkat dengan HTML adalah sebuah bahasa markah (*markup language*) yang digunakan untuk membuat sebuah halaman *web*, menampilkan berbagai informasi di dalam sebuah penjelajah *web* Internet dan pemformatan hiperteks sederhana yang ditulis dalam berkas format ASCII agar dapat menghasilkan tampilan wujud yang terintegerasi. Dengan kata lain,

berkas yang dibuat dalam perangkat lunak pengolah kata dan disimpan dalam format ASCII normal sehingga menjadi halaman *web* dengan perintah-perintah HTML. Bermula dari sebuah bahasa yang sebelumnya banyak digunakan di dunia penerbitan dan percetakan yang disebut dengan SGML (Standard Generalized *Markup Language*), HTML adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman *web*. HTML saat ini merupakan standar Internet yang didefinisikan dan dikendalikan penggunaannya oleh World Wide *Web* Consortium (W3C). HTML dibuat oleh kolaborasi Caillau TIM dengan Berners-lee Robert ketika mereka bekerja di CERN pada tahun 1989 (CERN adalah lembaga penelitian fisika energi tinggi di Jenewa).

Dokumen HTML mirip dengan dokumen tulisan biasa, hanya dalam dokumen ini sebuah tulisan bisa memuat instruksi yang ditandai dengan kode atau lebih dikenal dengan TAG tertentu. Sebagai contoh jika ingin membuat tulisan ditampilkan menjadi tebal seperti: TAMPIL TEBAL, maka penulisannya dilakukan dengan cara: TAMPIL TEBAL. Tanda digunakan untuk mengaktifkan instruksi cetak tebal, diikuti oleh tulisan yang ingin ditebalkan, dan diakhiri dengan tanda untuk menonaktifkan cetak tebal tersebut. HTML lebih menekankan pada penggambaran komponen-komponen struktur dan format di dalam halaman *web* daripada menentukan penampilannya. Sedangkan penjelajah *web* digunakan untuk menginterpretasikan susunan halaman ke gaya built-in penjelajah *web* dengan menggunakan jenis tulisan, tab, warna, garis, dan perataan text yang dikehendaki ke komputer yang menampilkan halaman *web*. Salah satu hal Penting tentang eksistensi HTML adalah tersedianya Lingua franca (bahasa Komunikasi) antar komputer dengan kemampuan berbeda. Pengguna Macintosh tidak dapat melihat tampilan yang sama sebagaimana tampilan yang terlihat dalam pc berbasis Windows. Pengguna Microsoft Windows pun tidak akan dapat melihat tampilan yang sama sebagaimana tampilan yang terlihat pada pengguna yang menggunakan Produk-produk Sun Microsystems [14].

2.10. MySQL

MySQL adalah sebuath *relational database* management *system* (RDBMS) yang open-source. Nama MySQL sendiri berasal dari gabungan antara “My”, nama anak dari pembuat MySQL, dan “SQL”, singkatan dari Structured Query *Language*. MySQL adalah software yang gratis dan open-source dibawah lisensi dari GNU General Public License, dan juga tersedia dibawah lisensi-lisensi yang lain. MySQL dulunya

dimiliki dan didukung oleh perusahaan Swedia yaitu MySQL AB, yang kemudian dibeli oleh Sun Microsystems (sekarang sudah menjadi Oracle Corporation). Di tahun 2010 saad Oracle mengakuisisi Sun, Widenius meng-copy dan mengembangkan sendiri proyek MySQL yang open-source secara terpisah menjadi MariaDB.

MySQL adalah salah satu komponen dari LAMP *web* application and software stack, yangmana adalah singkatan dari Linux, Apache, MySQL, Perl/PHP/Python. MySQL digunakan di berbagai aplikasi *web* yang membutuhkan *database*, seperti Drupal, Joomla, phpBB, dan WordPress. MySQL juga merupakan software *database* open source yang paling populer di dunia, dimana saat ini digunakan lebih dari 100 juta pengguna di seluruh dunia [15], dan juga digunakan oleh berbagai website yang populer seperti Facebook, Twitter, Flickr, dan Youtube. Fitur-fitur dari MySQL tersebut adalah :

- *Relational Database System* : Seperti halnya software *database* lain yang ada di pasaran, MySQL termasuk RDBMS
- Arsitektur *Client-Server* : MySQL memiliki arsitektur *client-server* dimana *server database* MySQL terinstal di *server*. *Client* MySQL dapat berada di komputer yang sama dengan *server*, dan dapat juga di komputer lain yang berkomunikasi dengan *server* melalui jaringan bahkan internet.
- Mengenal perintah SQL Standar : SQL (Structured Query Languate) merupakan suatu bahasa standar yang berlaku di hampir semua software *database*. MySQL mendukung SQL bersi SQL:2003
- Mendukung Sub Select : mulai versi 4.1 MySQL telah mendukung select dalam select (sub select)
- Mendukung Views : MySQL mendukung views sejak versi 5.0
- Mendukung Stored Prosedured : MySQL mendukung SP sejak versi 5.0
- Mendukung Triggers : MySQL mendukung trigger pada versi 5.0 namun masih terbatas. Pengembang MySQL berjanji akan meningkatkan kemampuan trigger pada versi 5.1.
- Mendukung Replication
- Mendukung Transaksi
- Mendukung Foreign Key.

2.11. Ultrasonic

Ultrasonic adalah getaran dengan frekuensi yang lebih tinggi dari batas atas pendengaran manusia, atau lebih tinggi dari 20KHz. Istilah "sonic" diaplikasikan ke gelombang suara ultra dengan aplitudo yang sangat tinggi. Hypersound, atau terkadang disebut praetersound atau microsound, adalah gelombang suara yang frekuensinya lebih tinggi dari 10^13 hertz. Di frekuensi setinggi itu, sangat sulit bagi gelombang suara untuk tersebar secara efisien. Memang gelombang longitudinal dengan frekuensi diatas 1.25*10^13 tidak akan bisa tersebar sama sekali, meskipun melalui bahan liquid ataupun solid, karena molekul yang dilewati oleh gelombang tidak bisa menyalurkan getaran dengan cukup cepat. Banyak hewan mempunyai kemampuan untuk mendengar suara baik di frekuensi dengar manusia maupun di frekuensi ultrasonic.

Sensor ultrasonik adalah sensor yang bekerja berdasarkan prinsip pantulan gelombang suara dan digunakan untuk mendeteksi keberadaan suatu objek atau benda tertentu didepan frekuensi kerja pada daerah diatas gelombang suara dari 20 kHz hingga 2 MHz. Sensor ultrasonik terdiri dari dari dua unit, yaitu unit pemancar dan unit penerima struktur unit pemancar dan penerima. Sangatlah sederhana sebuah kristal piezoelectric dihubungkan dengan mekanik jangkar dan hanya dihubungkan dengan diafragma penggetar tegangan bolak-balik yang memiliki frekuensi kerja 20 kHz hingga 2 MHz. Struktur atom dari Kristal piezoelectric menyebabkan berkontraksi mengembang atau menyusut, sebuah polaritas tegangan yang diberikan dan ini disebut dengan efek piezoelectric pada sensor ultrasonik. Pantulan gelombang ultrasonik terjadi bila ada objek tertentu dan pantulan gelombang ultrasonik akan diterima kembali oleh unit sensor penerima. Selanjutnya unit sensor penerima akan menyebabkan diafragma penggetar akan bergetar dan efek piezoelectric menghasilkan sebuah tegangan bolak-balik dengan frekuensi yang sama [16].

Gambar 2.7 HC-SR04 Ultrasonic Sensor

2.12. Tinjauan Pustaka

Tinjauan pustaka digunakan untuk menjadi sebuah acuan untuk tugas akhir ini dengan perangkat yang telah ada dan sudah dikembangkan sebelumnya. Berikut merupakan judul-judul *paper* yang akan diajukan sebagai acuan tugas akhir ini.

1. Radio Frequency Identification (RFID) Based Attendance System with Automatic Door unit [17].

   Pada penelitian ini, dibuat sebuah sistem berbasis RFID untuk mencatat waktu kehadiran. Perangkat keras dari sistem ini terdiri dari motor dan RFID reader. Perangkat lunak yang digunakan dalam penelitian ini adalah GUI pencatat kehadiran berdasarkan waktu yang dikembangkan menggunakan visual basic.Net. Riset ini menggunakan RFID reader dengan frekuensi 125kHz, yang memungkinkan pembacaan RFID tag jarak dekat.
2. Utilizing RFID for Smart Parking Applications [18].

   Pada penelitian ini, telah dibuat sebuah sistem yang bisa mengatasi masalah manajemen lahan parkir menggunakan teknologi RFID. Transaksi keluar dan masuk dari lahan parkir akan ditangani oleh RFID readers, label, dan palang pintu. Kendaraan tidak perlu berhenti di palang pintu untuk menunggu proses transaksi. Proses transaksi sudah dilakukan secara otomatis saat kendaraan lewat.
3. Radio Frequency Identification (RFID) Based Car Parking System [3].

   Pada penelitian ini, dibuat sistem manajemen lahan parkir berbasis RFID untuk perusahaan "Fujitsu" dan "Parit Raja". Sistem ini menggunakan RFID berfrekuensi 13.56 MHz, yang memiliki jarak pembacaan sejauh 3 meter. Metodologi yang digunakan pada penelitian ini adalah Object Oriented Software Development (OOSD)
4. RFID Based Automatic Parking System [2].

   Pada penelitian ini dibuat sebuah sistem manajemen lahan parkir berbasis RFID. RFID reader yang digunakan memiliki frekuensi 125kHz, dengan jarak baca sejauh 10cm. Pada penelitian ini juga terdapat sensor infrared yang mendeteksi apakan lahan parkir itu sudah terisi kendaraan atau belum, sehingga pengendara mengetahui lahan mana yang kosong dan tidak perlu bingung mencari.
5. IoT Based Smart Parking System [4].

   Pada penelitian ini dibuat sebuah sistem smart parking management. Pada smart parking ini dapat dipantau apakah sebuah slot parkir tersebut sudah terisi atau belum. Sensor yang dipakai untuk mendeteksi

keberadaan kendaraan adalah sensor ultrasonic, yang dipasang di tiap-tiap slot lahan parkir. Data slot yang kosong ataupun tidak tersebut ditampilkan melalui web interface.

6. IoT Based Smart Parking System [19].

Pada penelitian ini dibuat sebuah sistem smart parking yang dapat memantau ketersediaan lahan parkir. Sensor yang digunakan pada penelitian ini adalah sensor PIR dan picam yang dihubungkan ke Raspberry Pi. Lalu ada sensor tambahan yaitu sensor ultrasonic yang berhubungan secara nirkabel dengan Raspberry Pi melalui modul ESP8266. Data yang ada kemudian disimpan pada database online yang bisa diakses menggunakan aplikasi pada smartphone

7. BS-320 Barrier Gate by Evoxen [20].

Palang pintu ini memiliki spesifikasi lengan sepanjang 3-4,5 m. Motor yang digunakan adalah motor AC 230V dengan kecepatan 680 rpm. Durasi yang dibutuhkan untuk membuka dan menutup lengan adalah masing-masing 2.5 detik.

8. Doorking – Barrier Gate [21].

Palang pintu ini memiliki desain lengan yang bisa melipat saat terangkat. Digerakkan oleh motor AC dengan gear sistem ber-rasio 40:1. Waku yang dibutuhkan untuk menaikkan dan menurunkan lengan dari palang pintu ini adalah 1,5 detik.

9. Gard 4 Barrier Gate by CAME-Americas [22].

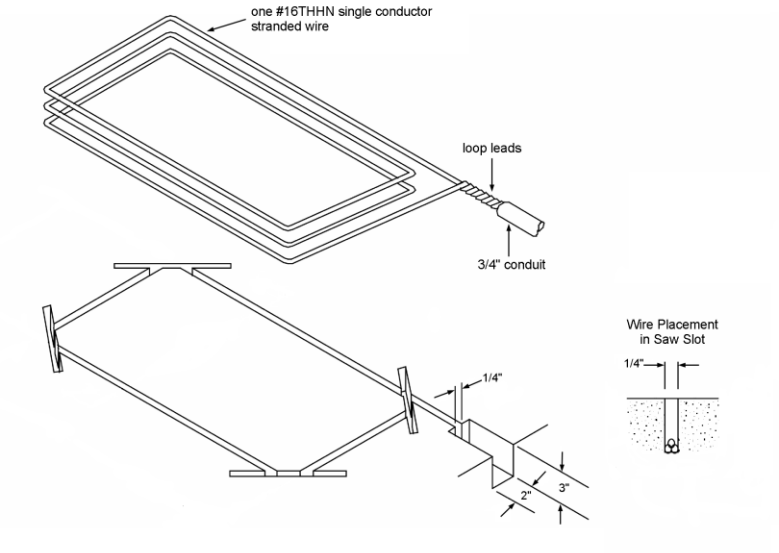
Palang pintu ini memiliki kelebihan desain yang elegan, serta memiliki LED sepanjang lengannya. Palang pintu ini digerakkan dengan motor AC bertegangan 24V. Waktu yang dibutuhkan untuk menaikkan dan menurunkan lengan palang pintu ini bisa diatur mulai 2-6 detik sesuai kebutuhan.

10. X-Bar Barrier Gate by NICE [23].

Palang pintu ini memiliki fitur yang berbeda dari yang lain, yaitu remote control yang bisa digunakan untuk membuka dan menutup palang pintu. Palang pintu ini digerakkan oleh motor AC 230V dengan arus 0.7 Ampere. Waktu yang dibutuhkan untuk membuka dan menutupnya palang pintu adalah maksimum 4 detik.

11. Barrier Gate by Ber-National Controls [24].

Palang pintu ini memiliki fitur tambahan, yaitu pendeteksi kendaraan yang dipakai tidak seperti biasa. Pendeteksi kendaraan pada palang pintu ini menggunakan coil yang ditanam dibawah tanah, yang akan memberikan respon frekuensi yang berbeda ketika ada logam diatasnya.

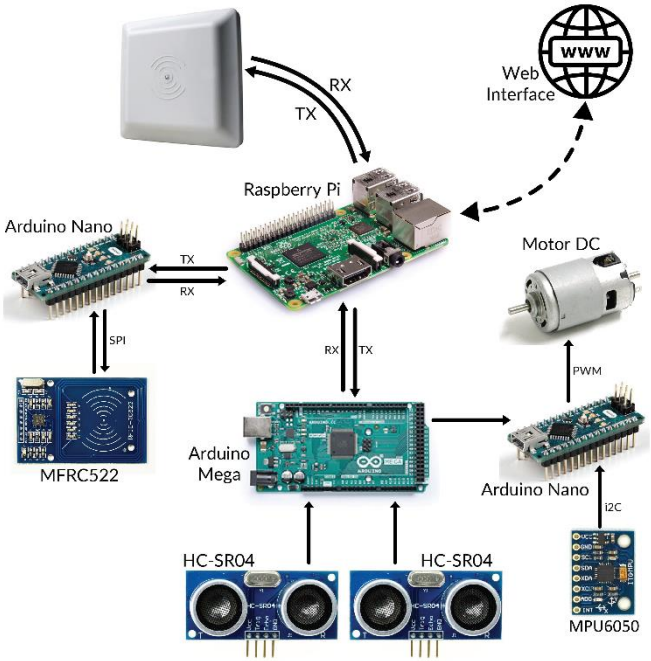


Gambar 2.8 Pemasangan Coil Pendeteksi Kendaraan.

BAB III
PERANCANGAN SISTEM

3.1. Perancangan Keseluruhan Sistem

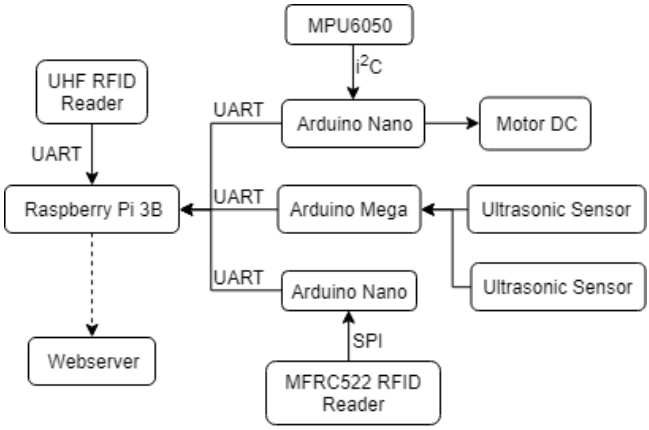
Sistem keamanan pintu parkir menggunakan teknologi *RFID* adalah suatu *system* yang bertujuan untuk mempermudah penjagaan keamanan sehingga tenaga kerja yang dibutuhkan bisa berkurang.



Gambar 3.1 Skema Sistem Secara Keseluruhan

Seperti ditunjukkan pada Gambar 3.1, sistem ini menggunakan 3 buah mikrokontroler dan 1 buah single-board computer. 3 mikrokontroler tersebut terdiri dari satu Arduino Mega untuk menerima data dari sensor ultrasonic, satu Arduino Nano untuk menerima data dari MFRC522 *RFID Reader*, dan satu lagi Arduino Nano sebagai PID Controller untuk aktuator motor DC. Sedangkan single-board Computer yang digunakan disini adalah Raspberry Pi model 3B. Raspberry Pi disini digunakan untuk

mengatur jalannya keseluruhan sistem, menerima data dari *UHF RFID Reader*, dan juga sebagai host dari *web*server yang akan dijalankan. Sedangkan sensor-sensor yang digunakan adalah HC-SR04 Ultrasonic Sensor, MFRC522 *RFID Reader*, *UHF RFID Reader*, dan MPU6050. Sebagai aktuator digunakan sebuah motor DC untuk menggerakkan palang pintu parkiran. Ultrasonic sensor digunakan untuk mendeteksi apakah ada motor yang akan masuk ke lahan parkir, dan juga apakah motor sudah melewati palang pintu. *UHF RFID Reader* digunakan untuk membaca *tag* pada motor saat kendaraan akan masuk ke lahan parkiran. MFRC522 *RFID Reader* digunakan untuk membaca *tag* pengemudi ketika akan mengeluarkan motornya dari lahan parkiran. MPU6050 digunakan sebagai input *feedback* pada PID loop yang mengatur jalannya aktuator palang parkir. Blok diagram sistem dapat dilihat di Gambar 3.2.



Gambar 3.2 Diagram Blok Sistem.

UHF RFID Reader akan membaca dengan interval 1 detik, jadi tiap 1 detik sekali dia akan membaca apakah ada kartu baru atau tidak, jika ada kartu maka identitas dari kartu tersebut akan dibaca dan dikirim ke Raspberry Pi, namun tidak langsung dibaca oleh Raspberry Pi. Raspberry Pi baru akan membaca data dari *UHF RFID Reader* ketika ultrasonic sensor pertama sudah mendeteksi kalau ada objek didepannya. Identitas dari kartu yang dibaca tadi akan dicocokkan dengan data-data kartu yang terdaftar di *database*. Jika kartu yang dibaca sudah terdaftar di *database*, maka palang pintu akan terbuka, dan terus terbuka sampai sensor ultrasonic yang berada di palang parkir mendeteksi kalau sepeda motor

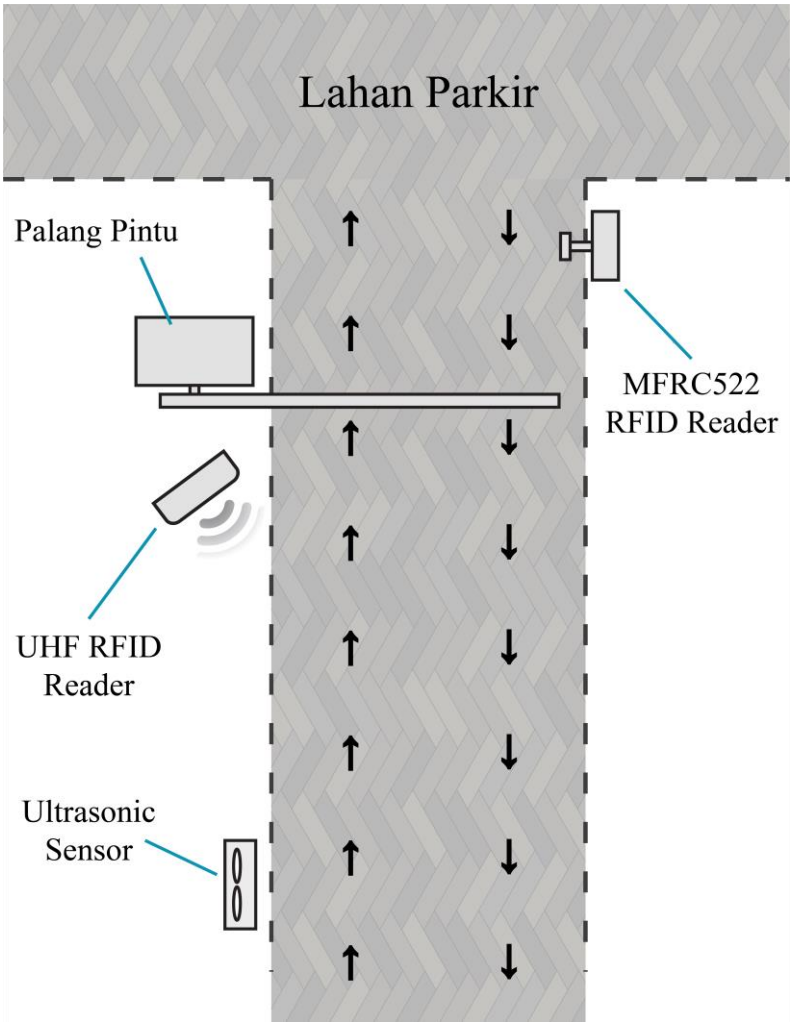
sudah melewati palang, baru palang akan menutup kembali. Untuk prosedur keluar dari lahar parkir kurang lebih sama hanya saja *RFID Reader* yang digunakan adalah MFRC522, *Reader* yang berjarak dekat. Langkah-langkah tersebut diilustrasikan dalam bentuk flowchart yang ditunjukkan pada Gambar 3.4.

Hasil dari data yang dihasilkan oleh *UHF RFID Reader* adalah data string identitas kartu yang dikirim melalui serial usb menuju raspberry pi. Yang kemudian dicocokkan dengan *database* kartu yang ada.

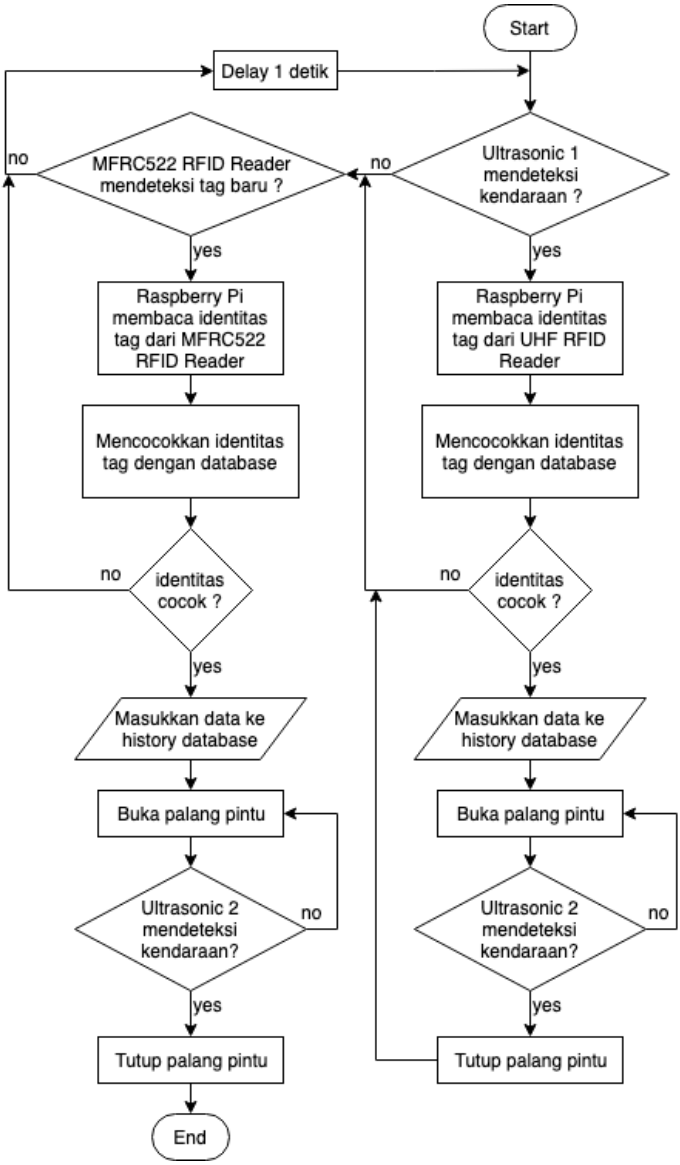
Hasil pembacaan dari HC-SR04 ultrasonic sensor adalah data waktu seberapa lama pantulan gelombang suaranya dibaca, kemudian dikonversi oleh Arduino Mega menjadi data jarak. Yang kemudian digunakan untuk mendeteksi apakah ada kendaraan yang ingin masuk ke lahan parkiran, atau apakah kendaraan sudah melewati palang parkir atau belum.

Hasil dari data yang dihasilkan oleh MPU6050 adalah data akselerometer dan gyroscope, yang kemudiang dikonversi oleh Arduino Nano menjadi data kemiringan sumbu X, Y, dan Z. Namun yang dipakai pada penelitian ini hanyalah sumbu X saja, karena palang pintu hanya bergerak pada satu sumbu yaitu sumbu X. Data sudut tersebut kemudian dimasukkan menjadi *feedback* dari sistem PID untuk mengatur jalannya motor DC palang pintu.

Semua komponen tersebut akan bekerja secara bersamaan, sehingga penempatan tiap-tiap komponen adalah hal yang penting. Sensor ultrasonic pertama diletakkan di paling depan, sehingga bisa mendeteksi ketika sepeda motor ingin masuk. UHF *RFID Reader* diletakkan dibelakangnya, sehingga bisa membaca identitas kartu yang ada di sepeda motor. Kemudian diletakkan palang pintu. Dibelakang palang pintu diletakkan MFRC522 *RFID Reader* yang berfungsi untuk keluar dari lahan parkir. Ilustrasi tata letak alat dan komponen ditunjukkan pada Gambar 3.3. Alasan mengapa sistem ini menggunakan 2 *RFID Reader* yang berbeda adalah untuk meningkatkan keamanan. Jika hanya menggunakan satu *RFID Reader* dengan kartu yang sama, maka pencuri hanya perlu satu kartu untuk melewati gerbang yang dilindungi dengan *RFID Reader*. Jika ada 2 kartu yang berbeda, maka pencuri mmemerlukan 2 kartu yang berbeda, sehingga akan lebih susah untuk melakukan pencurian.



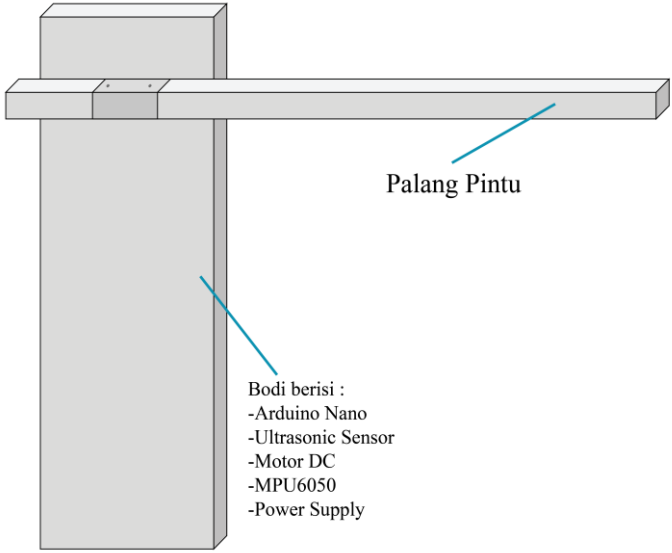
Gambar 3.3 Sketsa Tata Letak Alat.



Gambar 3.4 Flowchart Sistem Secara Keseluruhan.

3.2. Perancangan Mekanik

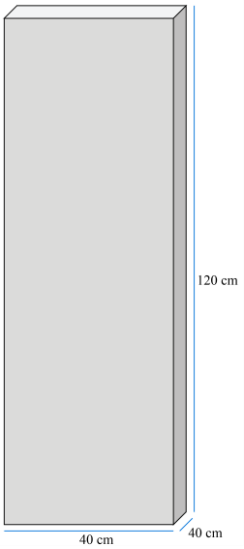
Perancangan mekanik pada penelitian ini berfokus pada desain mekanik palang pintu parkiran. Ilustrasi keseluruhan palang pintu ditunjukkan pada Gambar 3.5.



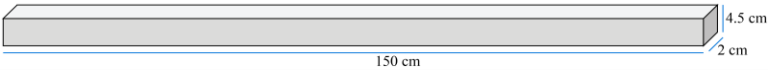
Gambar 3.5 Ilustrasi Palang Pintu.

Palang pintu terdiri dari 2 bagian utama yaitu bodi yang menopang palang pintu, dan juga palang pintu itu sendiri. Bodi berisi Arduino Nano, ultrasonic sensor, motor DC, dan *power supply*. Bodi dari palang pintu ini akan terbuat dari seng dengan ukuran 40cm x 40cm x 120m yang diilustrasikan di Gambar 3.6.

Untuk palang pintunya sendiri, didalamnya terdapat MPU6050 yang digunakan untuk mendeteksi kemiringan palang pintu. Data sudut tersebut digunakan sebagai input *feedback* dalam kontrol PID, yang digunakan untuk menggerakkan palang pintu sesuai yang diinginkan. Ukuran dari palang pintu tersebut adalah 4.5cm x 2cm x 150cm. Bahan yang digunakan untuk membuat palang pintu ini adalah aluminium. Ilustrasi desain palang pintu ditunjukkan di Gambar 3.7.



Gambar 3.6 Ilustrasi Desain Bodi.



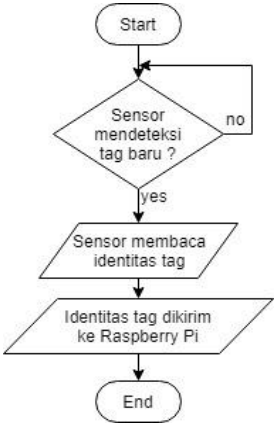
Gambar 3.7 Ilustrasi Desain Palang Pintu.

3.3. Perancangan Perangkat Lunak

Perancangan perangkat lunak ini digunakan untuk mengukur semua sensor yang ada. Sensor MFRC522 *RFID Reader*, HC-SR04 Ultrasonic *Reader*, dan MPU6050 dibaca melalui Arduino, baik Arduino Mega maupun Arduino Nano. Sedangkan *UHF RFID Reader* dibaca melalui Raspberry Pi.

3.3.1. Pembacaan *UHF RFID Reader*

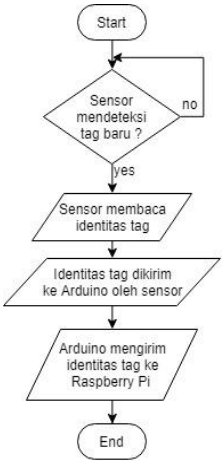
UHF RFID Reader mempunyai berbagai protokol komunikasi, diantaranya USB, Serial RS232, dan Net *Interface*. Pada tugas akhir kali ini digunakan komunikasi Serial RS232 melalui USB. Flowchart pembacaan *UHF RFID Reader* ditunjukkan di Gambar 3.8. RFID Reader ini mempunyai jarak pembacaan maksimal 6 meter dan beroperasi pada frekuensi 902-928 MHz. *Tag* yang digunakan dapat menyimpan data sebanyak maksimal 12 bytes.



Gambar 3.8 Flowchart Pembacaan *UHF RFID Reader*.

3.3.2. Pembacaan MFRC522 *RFID Reader*

MFRC522 *RFID Reader* berkomunikasi menggunakan protokol SPI dengan Arduino nano, kemudian dikirim ke Raspberry pi melalui USB. Berikut flowchart pembacaan data MFRC522 ditampilkan pada Gambar 3.9. Reader ini jarak pembacaannya 4 cm dan beroperasi pada frekuensi 125 kHz. *Tag* yang digunakan dapat menyimpan data sebesar 1KB, ditambah 4 Bytes data identitas.



Gambar 3.9 Flowchart Pembacaan MFRC522 *RFID Reader*.

3.3.3. Pembacaan HC-SR04 Ultrasonic Sensor

HC-SR04 ultrasonic sensor menghasilkan pembacaan jarak dalam satuan centimeter. Sensor ini dibaca dengan Arduino melalui digital pin biasa, dan membutuhkan 2 pin data yaitu trigger dan echo. Berikut flowchart pembacaan ultrasonic sensor ditampilkan dalam Gambar 3.10. Sensor ini mempunyai jarak pembacaan maksimum 4 meter.

Gambar 3.10 Flowchart Pembacaan HC-SR04 Ultrasonic Sensor.

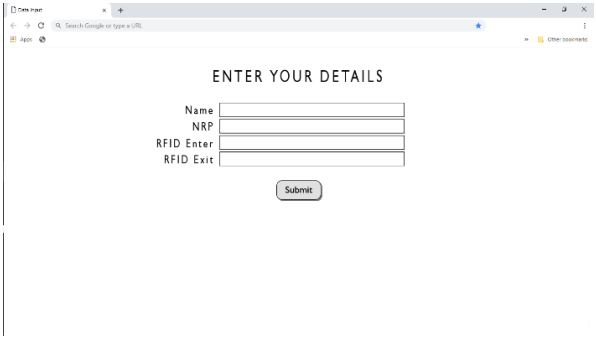
3.3.4. Pembacaan MPU6050

MPU6050 berkomunikasi dengan Arduino menggunakan protokol I^2C. Arduino akan membaca data sudut dari MPU6050 lalu diproses sebagai input PID control, dan mengeluarkan output PWM motor sesuai perhitungan. Flowchart dari pembacaaan MPU6050 ditunjukkan pada Gambar 3.11.

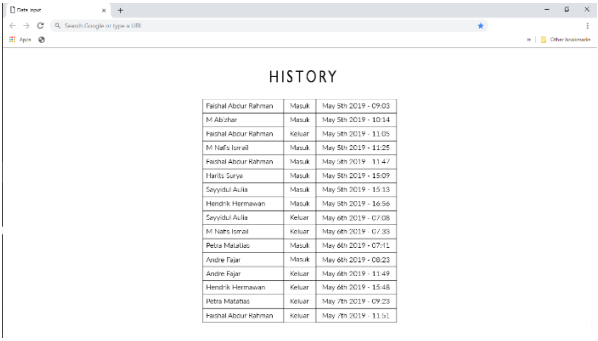
Gambar 3.11 Flowchart Pembacaan MPU6050.

3.3.5. Perancangan *Database* dan *Web Interface* MySQL

Pada tugas akhir ini digunakan *database* MySQL untuk menyimpan data-data *RFID tag* yang terdaftar dan pemilik kartu tersebut serta NRP nya. Digunakan *database* agar mempermudah penyimpanan data sehingga tidak perlu menyimpan satu-persatu data di kode autentikasi. Pada *database* juga disimpan data rekaman aktivitas akses lahan parkir. Penyimpanan *database* disimpan di Raspberry Pi dan diakses secara lokal di jaringan wifi yang sama. Gambar 3.12 menunjukkan desain rancangan yang akan direalisasikan menjadi tampilan *web*site untuk memasukkan data ke *database*. Lalu Gambar 3.13 menunjukkan desain tampilan *web*site yang akan digunakan untuk menampilkan rekaman aktivitas siapa saja yang keluar dan masuk dari pintu parkiran.



Gambar 3.12 Desain Tampilan *Web* untuk Memasukkan Data ke *Database*



Gambar 3.13 Desain Tampilan *Web* untuk Menampilkan Rekaman Akses Lahan Parkir

BAB IV
PENGUJIAN DAN ANALISIS

4.1. Pengujian *UHF RFID Reader*

Pada Gambar 4.1 ditunjukkan proses pengujian *UHF RFID Reader*. Pengujian dilakukan di laboratorium B202 Teknik Elektro ITS, dengan *RFID Reader* dipasang pada tiang penyangga dan subjek yang membawa kartu berdiri dengan jarak yang diubah-ubah. Pada Tabel 4.1 ditunjukkan data yang didapat dari pengujian tersebut. Dari hasil pengujian yang dilakukan terlihat bahwa semakin jauh maka *RFID Reader* semakin sulit untuk membaca identitas *tag*, hal ini bisa dipengaruhi oleh lingkungan sekitar dan interferensi yang ada, serta semakin jauh maka sinyal yang dipancarkan akan semakin lemah sehingga pembacaan jadi lebih susah.

Gambar 4.1 Pengujian *UHF RFID Reader*.

Tabel 4.1 Hasil Pengujian *UHF RFID Reader*.

Jarak (m)	Presentase keberhasilan
> 6	13.33%
5	33.33%
4	60%
3	86.67%
< 2	100%

Tabel 4.2 Tabel Pembacaan UHF *RFID Reader*

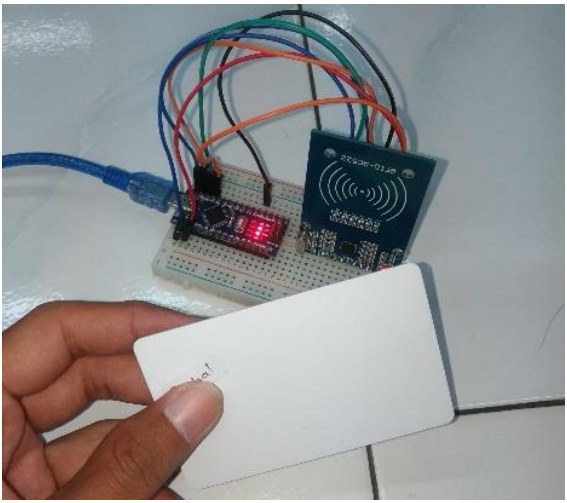
Identitas tag	Identitas yang terbaca
000000032761	000000032761
000000032762	000000032762
000000032763	000000032763
000000032764	000000032764
000000032765	000000032765
000000032766	000000032766
000000032767	000000032767
000000032768	000000032768
000000032769	000000032769
000000032770	000000032770

Pada Tabel 4.2 ditunjukkan data hasil pembacaan UHF *RFID Reader* dengan 10 tag dengan identitas yang berbeda-beda. Tiap tag dibaca kemudian dicocokkan antara identitas sebenarnya dengan identitas yang terbaca oleh *RFID Reader*. Dari data diatas dapat dilihat bahwa semua tag terbaca dengan benar, tidak ada yang salah. Sehingga dapat disimpulkan bahwa UHF *RFID Reader* mempunyai akurasi pembacaan 100%, tidak akan ada tag yang terbaca dengan data yang salah.

4.2. Pengujian MFRC522 *RFID Reader*

Pengujian MFRC522 *RFID Reader* dilakukan dengan menyambungkan *RFID Reader* dengan arduino nano dengan protokol SPI, lalu membaca data *RFID* dengan program yang sudah sebelumnya dibuat. Pada Gambar 4.2 ditunjukkan gambar saat pengujian dilakukan,

sedangkan Tabel 4.3 dan Tabel 4.4 menunjukkan hasil pengujian *RFID Reader* dengan parameter yang berbeda, yaitu jarak dan waktu. Tabel 4.5 menunjukkan pengujian pembacaan MFRC *RFID Reader* dengan tag-tag dengan identitas yang berbeda.



Gambar 4.2 Pengujian MFRC522 *RFID Reader*.

Tabel 4.3 Hasil Pengujian MFRC522 dengan Variabel Jarak

Jarak (cm)	Keberhasilan pembacaan *tag*
6	Tidak terbaca
5	Tidak terbaca
<=4	Terbaca

Tabel 4.4 Hasil Pengujian MFRC dengan Dariabel Durasi Pembacaan

Waktu (detik)	Data yang terbaca
2	39%
4	78%
6	100%

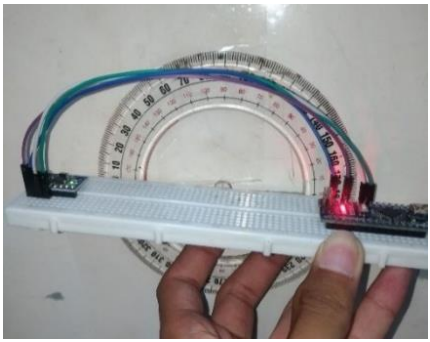
Tabel 4.5 Tabel Pembacaan MFRC522 *RFID Reader*

Identitas tag	Identitas yang terbaca
51F7BE1A	51F7BE1A
51E6341A	51E6341A
51F6311A	51F6311A
51E3FA1A	51E3FA1A
51F6E91A	51F6E91A
61809E1A	61809E1A
51E0221A	51E0221A
61B81E1A	61B81E1A
00B4C6A3	00B4C6A3
D014A47C	D014A47C
51E32F1A	51E32F1A

Pada Tabel 4.5 ditunjukkan pengujian pembacaan 11 tag yang berbeda menggunakan MFRC522 *RFID Reader*. Pada pengujian tersebut dapat dilihat bahwa semua tag terbaca dengan akurat, tidak ada tag yang terbaca dengan data yang salah. Sehingga akurasi pembacaan dari MFRC522 *RFID Reader* ini adalah 100%.

4.3. Pengujian MPU6050

Pengujian MPU6050 dilakukan dengan membandingkan hasil pembacaan sensor dengan hasil sebenarnya yang diukur dengan busur. Pada pengujian ini hanya diambil sampel data dari sudut 0 sampai 90 derajat, dengan interval 10 derajat. Pada Gambar 4.3 ditunjukkan cara pengujian sensor yang dilakukan, dan pada Tabel 4.6 ada hasil yang didapat dari pengujian sensor tersebut. Berdasarkan data yang didapatkan, jika dihitung persentase *error* yang didapat adalah 9.59%. Dikarenakan *error* yang didapat relatif kecil, maka tidak perlu dilakukan kalibrasi, dan data yang didapat dari sensor bisa langsung digunakan.



Gambar 4.3 Pengujian Sensor MPU6050

Tabel 4.6 Hasil pengujian MPU6050

Pengukuran	Pembacaan Sensor
0	0.71
10	11.91
20	23.12
30	34.23
40	45.15
50	55.79
60	66.27
70	76.79
80	87.04
90	96.77
Persentase *error* :	9.59%

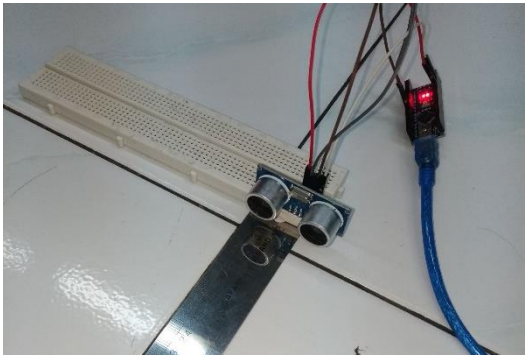
4.4. Pengujian HC-SR04 Ultrasonic Sensor

Pengujian sensor ultrasonic HC-SR04 dilakukan dengan membandingkan hasil pembacaan sensor dengan hasil pengukuran sebenarnya menggunakan penggaris, data sensor dibaca oleh Arduino dan ditampilkan di serial monitor. Gambar 4.4 menunjukkan metode pengujian yang dilakukan dan Tabel 4.7 menunjukkan perbandingan hasil dari pengukuran sebenarnya dan pembacaan sensor. Dari tabel dapat dilihat bahwa persentase error sensor sebesar 2.75%. Karena *error* yang

didapat tidak terlalu besar maka sensor dapat langsung digunakan tanpa perlu dilakukan kalibrasi.

Tabel 4.7 Hasil Pengujian HC-SR04

Pengukuran	Pembacaan sensor
10	10.51
20	20.22
30	29.41
40	38.89
50	48.35
60	58.12
70	67.86
80	77.75
90	87.53
100	98.04
Persentase *error* :	2.75%



Gambar 4.4 Pengujian HC-SR04 Ultrasonic Sensor

4.5. Pengujian Motor DC

Pengujian motor DC dilakukan dengan memberikan suplai gelombang PWM dengan nilai yang berbeda beda, dan menghitung seberapa cepat motor DC berputar. Pengujian juga dilakukan 2 kali untuk mengetahui apakan kecepatan rotasi kekiri dan kekanan motor adalah sama dengan gelombang PWM yang sama. Gambar 4.5 menunjukkan

gambar motor DC saat dilakukan pengujian lalu Tabel 4.8 dan Tabel 4.9 adalah hasil data yang didapat dari pengujian.

Gambar 4.5 Motor DC saat Dilakukan Pengujian

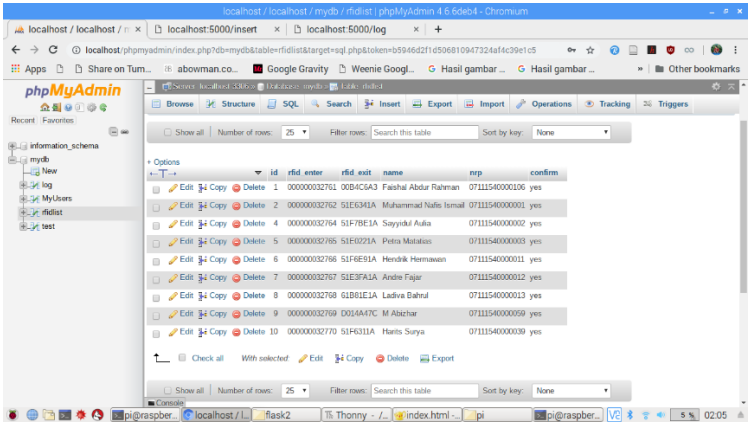
Tabel 4.8 Perbandingan PWM dengan RPM Putaran Kekanan.

PWM (duty cycle)	Kecepatan Putaran (RPM)
0%	0
19.6%	29
39.2%	60
58.8%	89
78.4%	119
100%	153

Tabel 4.9 Perbandingan PWM dengan RPM Putaran Kekiri.

PWM (duty cycle)	Kecepatan Putaran (RPM)
0%	0
19.6%	29
39.2%	59
58.8%	88
78.4%	118
100%	151

4.6. Pengujian *Database* MySQL

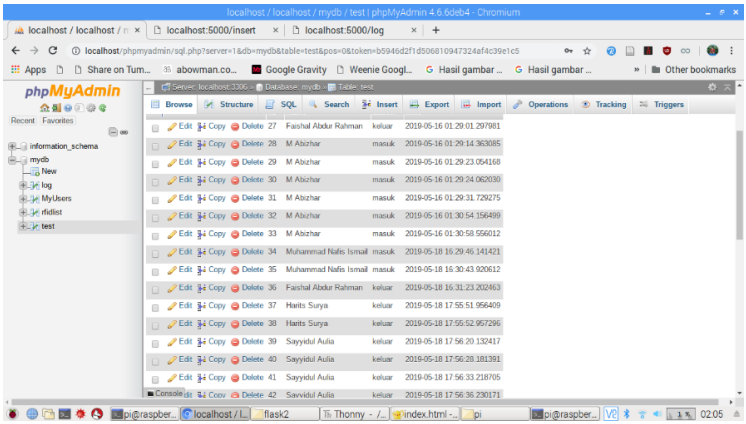


Gambar 4.6 Tabel MySQL yang Menyimpan *RFID Tag* yang Terdaftar

Pada Gambar 4.6 ditunjukkan tabel yang dinamai *RFIDlist*. Tabel tersebut telah dibuat yang berisi data-data *RFID tag* yang telah terdaftar. Di tabel tersebut terdapat 6 kolom yaitu id, *RFID_enter*, *RFID_exit*, *name*, nrp, dan confirm. Id adalah urutan data yang ada di tabel tersebut, dan menjadi sesuatu yang harus ada di suatu tabel MySQL. Value dari kolom id tersebut adalah otomatis, selalu bertambah 1 setiap ada data baru yang ditambahkan ke tabel. Kolom *RFID_enter* berisi identitas *RFID tag* yang digunakan untuk masuk ke lahan parkiran, dengan kata lain adalah *RFID tag* yang bisa dibaca oleh *UHF RFID Reader*, berisi 12 digit huruf atau angka yang bisa diatur sesuai dengan kebutuhan. Kolom *RFID_exit* berisi identitas *RFID tag* yang digunakan untuk keluar dari lahan parkiran, atau *RFID tag* yang bisa dibaca oleh MFRC522 *RFID Reader*, berisi 8 digit huruf atau angka identitas *RFID tag* tersebut yang sudah ditetapkan nilainya sejak *RFID tag* tersebut dibuat, jadi tidak bisa dirubah nilainya. Kolom *name* berisi nama dari tiap pemilik *RFID tag* tersebut. Kolom nrp berisi NRP dari pemilik tiap *RFID tag*. Dan confirm adalah kolom yang berisi string “yes”, digunakan sebagai konfirmasi data saat raspberry pi mengambil data dari kolom tersebut.

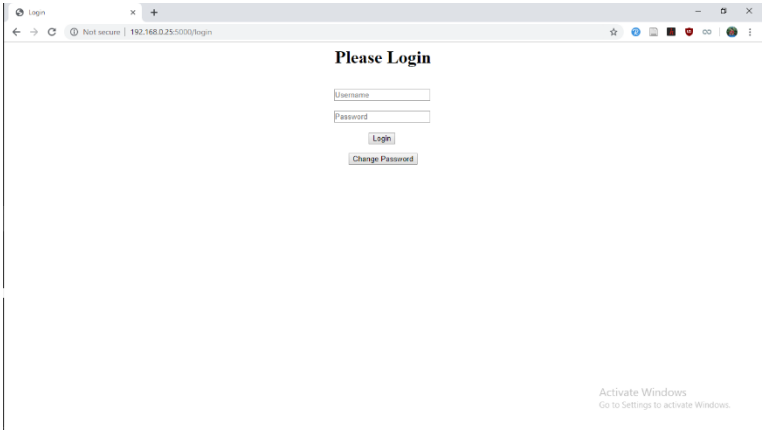
Gambar 4.7 menunjukkan tabel *database* yang digunakan untuk menyimpan rekaman aktifitas keluar masuknya orang dari lahan parkiran. Tabel ini mempunyai 4 kolom yaitu Id, *Name*, NRP, dan *timestamp*. Id adalah kolom yang harus ada dalam setiap tabel MySQL, nilai nya secara

otomatis bertambah 1 setiap ada data baru yang ditambahkan. *Name* berisi nama dari pemilik *RFID tag*. NRP berisi NRP dari pemilik *RFID tag*. *Action* berisi aksi yang dilakukan oleh orang tersebut, apakah dia keluar atau masuk dari lahan parkiran. *Timestamp* berisi tanggal dan waktu data tersebut dimasukkan ke *database*. Dengan kata lain kapan orang itu keluar atau masuk dari lahan parkiran.



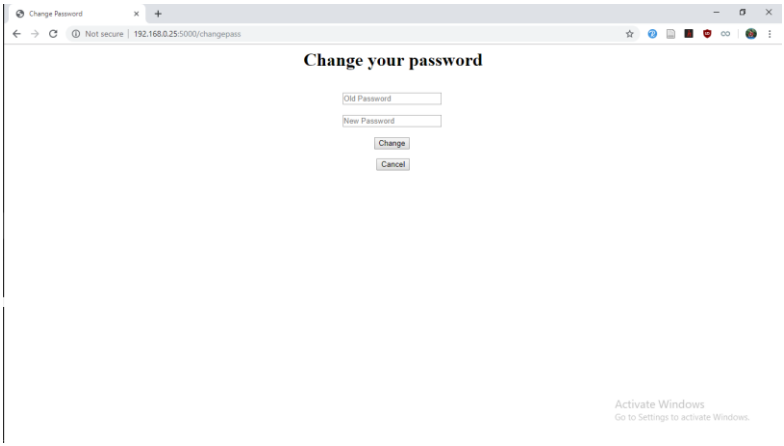
Gambar 4.7 Tabel MySQL yang Menyimpan Rekaman Akses Parkir.

4.7. Pengujian *Web* Interface



Gambar 4.8 Tampilan Halaman "/login"

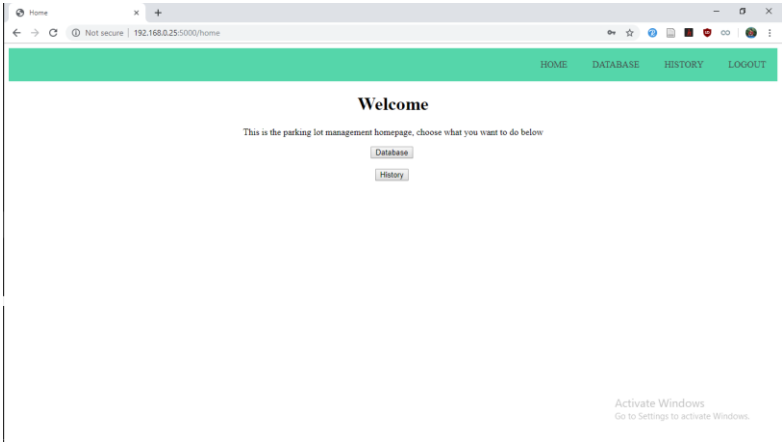
Web interface pada penelitian ini digunakan untuk menampilkan dan mengolah data seperti rekaman kendaraan yang masuk dan keluar dari lahan parkir, dan juga *database* kendaraan yang terdaftar mendapat akses ke lahan parkir. Pada Gambar 4.8 ditunjukkan tampilan *web* saat memasukkan alamat "localhost:5000/login" pada browser. Ini adalah alamat pertama yang muncul saat pengguna ingin mengakses *database*, jadi pengguna harus login dahulu sebelum bisa mengakses lebih lanjut. Pada halaman ini terdapat 2 textfield dan 2 button. 2 textfield tersebut digunakan untuk memasukkan username dan password. Tombol "Login" digunakan untuk login setelah memasukkan username dan password. Tombol "Change Password" digunakan untuk menuju ke halaman "/changepass". Pada saat pengguna menekan tombol login, maka username dan password yang dimasukkan akan dicocokkan dengan yang ada pada *database*, kemudian jika sudah benar maka pengguna akan diarahkan ke halaman "/home".



Gambar 4.9 Tampilan Halaman "/changepass"

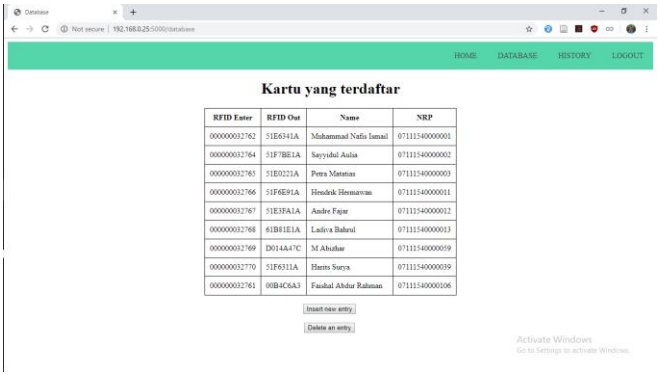
Pada Gambar 4.9 ditunjukkan tampilan *web* saat memasukkan alamat "localhost:5000/changepass", atau bisa diakses dengan menekan tombol "Change Password" pada halaman login. Pada halaman ini pengguna bisa mengganti password yang digunakan untuk login ke *database*. Pada textfield "Old Password" pengguna harus memasukkan password yang lama, sedangkan pada textfield "New Password" pengguna harus memasukkan password yang baru, kemudian baru

pengguna menekan tombol “Change”. Tombol “Cancel” berfungsi untuk mengembalikan pengguna ke halaman login tanpa mengganti password. Saat pengguna mengganti password, yang terjadi adalah entry username dan password yang ada di *database* akan dihapus, dan akan dibuat entry baru dengan username “admin” dan password sesuai dengan yang dimasukkan di textfield “New Password”.



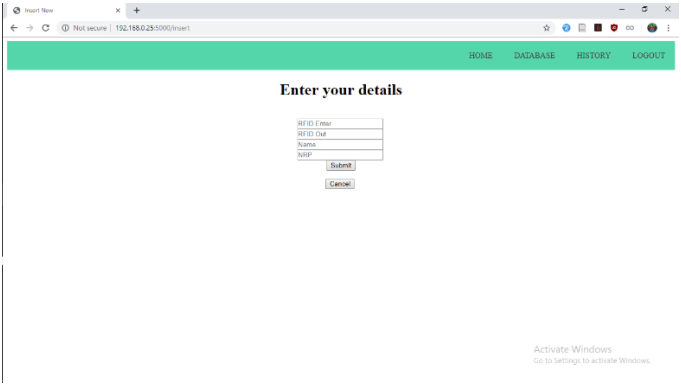
Gambar 4.10 Tampilan Halaman “/home”

Setelah pengguna berhasil login, maka akan diarahkan ke alamat “localhost:5000/home”, seperti yang ditunjukkan pada Gambar 4.10. Pada halaman ini terdapat 1 navigation bar yang ada di semua halaman kecuali login dan change password. Pada navigation bar tersebut terdapat 4 link yang bisa ditekan oleh pengguna, yaitu “Home”, “*Database*”, “History”, dan “Logoout”. Link “Home” mengarahkan pengguna ke halaman “/home”. Link “*Database*” mengarahkan pengguna ke halaman “/*database*”. Link “History” mengarahkan pengguna ke halaman “/log”. Dan link “Logout” akan mengarahkan pengguna ke halaman “/login” sembari menghapus login session yang ada pada pengguna, sehingga jika pengguna ingin melanjutkan mengakses *database* lewat *web*, harus login terlebih dahulu.Di halaman ini juga terdapat 2 tombol yaitu “*Database*” dan “History”. Tombol “*Database*” mengarahkan pengguna pada halaman “/*database*”, dan tombol “History” mengarahkan pengguna pada halaman “/log”.



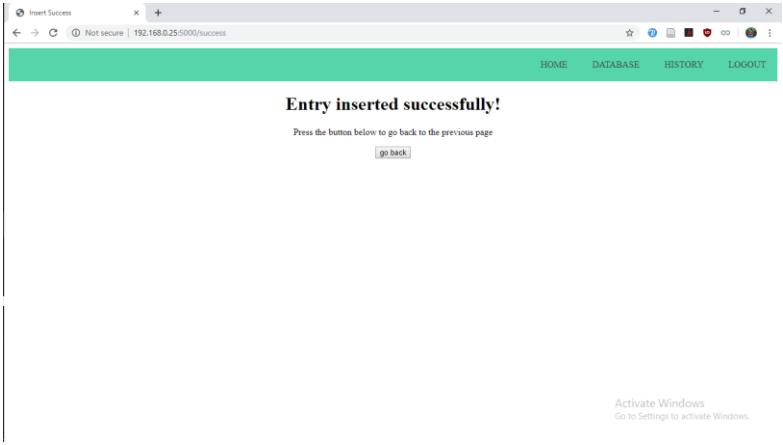
Gambar 4.11 Tampilan Halaman "/*database*"

Pada Gambar 4.11, ditunjukkan tampilan tampilan halaman "localhost:5000/*database*". Pada halaman ini, ditampilkan data siapa saja yang terdaftar mempunyai akses ke lahan parkir. Terdapat tabel untuk menampilkan datanya, lalu ada 2 tombol. Yang pertama adalah tombol "Insert an entry", jika ditekan mengarahkan pengguna ke alamat "localhost:5000/insert". Dimana pada halaman itu pengguna bisa mendaftarkan orang baru untuk diberi akses ke lahan parkir. Lalu tombol kedua adalah "Delete an entry", yang mengarahkan pengguna ke alamat "localhost:5000/delete". Dimana pada halaman itu pengguna bisa menghapus akses ke lahan parkir milih salah satu orang yang terdaftar.



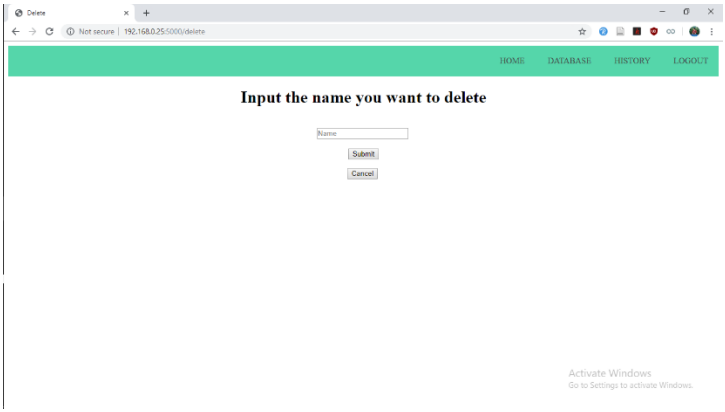
Gambar 4.12 Tampilan Halaman "/insert"

Pada Gambar 4.12, ditunjukkan tampilan halaman "localhost:5000/insert". Pada halaman insert ini, pengguna bisa mendaftarkan orang baru untuk diberi akses ke lahan parkir. Terdapat 4 textfield yaitu "*RFID* Enter", "*RFID* Out", "Name", dan "NRP". "*RFID* Enter" adalah identitas *RFID* tag yang digunakan untuk masuk ke lahan parkiran, atau yang bisa dibaca oleh UHF *RFID Reader*. "*RFID* Out" adalah identitas *RFID* tag yang digunakan untuk keluar dari lahan parkiran, atau yang bisa dibaca oleh MFRC522 *RFID Reader*. "Name" adalah nama dari pemilik kartu dan "NRP" adalah NRP dari pemilik kartu. Pada halaman ini juga terdapat 2 tombol yaitu "Submit" dan "Cancel". Tombol "Submit" ditekan ketika semua data textfield sudah diisi, maka kemudian pengguna akan diarahkan ke halaman "localhost:5000/success". Sedangkan tombol "Cancel" ditekan ketika pengguna tidak jadi mendaftarkan orang baru, dan akan dikembalikan ke "/*database*".



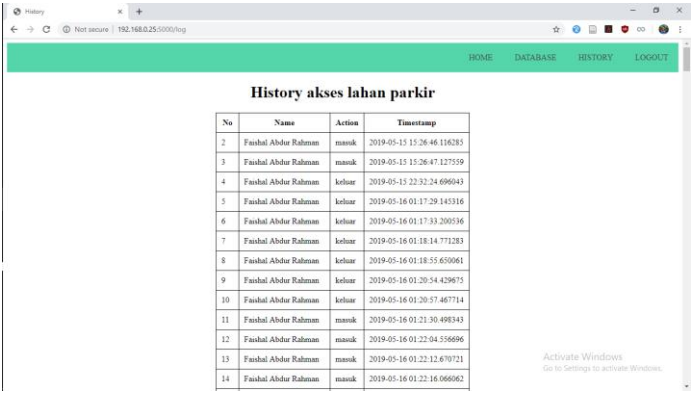
Gambar 4.13 Tampilan Halaman "/success"

Pada Gambar 4.13, ditunjukkan tampilan halaman "localhost:5000/success". Halaman ini hanya ditunjukkan untuk menandakan bahwa pengguna telah berhasil mendaftarkan orang baru kedalam lahan parkir.



Gambar 4.14 Tampilan Halaman "/delete"

Pada Gambar 4.14, ditunjukkan tampilan halaman "localhost:5000/delete". Fungsi dari halaman ini adalah sehingga pengguna bisa menghapus salah satu akses lahan parkir yang terdaftar. Pada halaman ini terdapat satu textfield dengan judul "Name", yang akan diisi nama pengguna yang akan dihapus aksesnya. Lalu ada 2 tombol yaitu "Submit" dan "Cancel". Tombol "Submit" berguna untuk menghapus akses lahan parkir milik orang yang namanya telah diisikan pada textfield "Name". Tombol "Cancel" digunakan ketika pengguna tidak jadi menghapus, dan akan diarahkan kembali ke halaman "/delete".



No	Name	Action	Timestamp
2	Faishal Abdur Rahman	masuk	2019-05-15 15:26:46.116285
3	Faishal Abdur Rahman	masuk	2019-05-15 15:26:47.127559
4	Faishal Abdur Rahman	keluar	2019-05-15 22:32:24.696043
5	Faishal Abdur Rahman	keluar	2019-05-16 01:17:29.145316
6	Faishal Abdur Rahman	keluar	2019-05-16 01:17:33.200536
7	Faishal Abdur Rahman	keluar	2019-05-16 01:18:14.771283
8	Faishal Abdur Rahman	keluar	2019-05-16 01:18:55.650061
9	Faishal Abdur Rahman	keluar	2019-05-16 01:20:54.429675
10	Faishal Abdur Rahman	keluar	2019-05-16 01:20:57.467714
11	Faishal Abdur Rahman	masuk	2019-05-16 01:21:30.498343
12	Faishal Abdur Rahman	masuk	2019-05-16 01:22:04.556696
13	Faishal Abdur Rahman	masuk	2019-05-16 01:22:12.670721
14	Faishal Abdur Rahman	masuk	2019-05-16 01:22:16.066062

Gambar 4.15 Tampilan Halaman "/log"

Pada Gambar 4.15, ditunjukkan tampilan halaman "localhost:5000/log". Halaman ini berfungsi untuk menunjukkan rekaman akses lahan parkir. Terdapat tabel dengan 4 kolom yang berisi data, kolom pertama adalah "No" atau nomor. Kolom kedua adalah "Name", menunjukkan nama dari orang yang mengakses lahan parkir. Kolom ketiga adalah "Action", yaitu menunjukkan apakah orang tersebut masuk atau keluar dari lahan parkir. Kolom keempat adalah "Timestamp", yaitu menunjukkan waktu orang tersebut mengakses lahan parkir.

4.8. Pengujian Keseluruhan Sistem

Pengujian keseluruhan sistem dilakukan dengan tujuan memastikan bahwa alur sistem bekerja seperti semestinya. Pengujian ini dilakukan di samping lapangan futsal Jurusan Teknik Elektro ITS. Langkah pertama yang dilakukan adalah memasang UHF *RFID* tag pada sepeda motor, seperti ditunjukkan pada Gambar 4.16

Gambar 4.16 Pemasangan *RFID* Tag pada Sepeda Motor

Seperti ditunjukkan pada Gambar 4.16, UHF *RFID* Tag dipasang pada sebelah kiri setir sepeda motor. Hal ini dikarenakan setelah percobaan yang berulang kali, disimpulkan bahwa posisi ini memiliki tingkat pembacaan yang paling maksimal. Setelah itu dilakukan pengujian dengan menjalankan sepeda motor melalui palang pintu. Gambar 4.17 menunjukkan kendaraan yang akan masuk ke lahan parkir. Kendaraan tersebut berhenti didepan sensor ultrasonik 1, dan dalam jarak

pembacaan UHF *RFID Reader*, sehingga kartu yang ada pada kendaraan tersebut bisa terbaca.

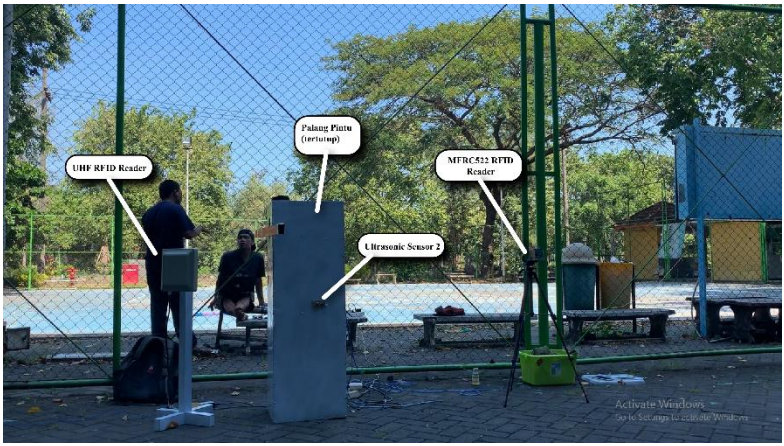
Gambar 4.17 Pengendara Ingin Masuk ke Lahan Parkir.

Setelah identitas kartu pada kendaraan terbaca, kemudian Raspberry Pi melakukan autentikasi terhadap identitas kartu tersebut. Jika autentikasi berhasil, maka palang pintu terbuka seperti ditunjukkan pada Gambar 4.18.

Gambar 4.18 Autentikasi Berhasil dan Pintu Terbuka.

Setelah autentikasi berhasil dan palang pintu terbuka, pintu tidak akan tertutup sampai ada objek yang melewati sensor ultrasonik 2. Pada kasus ini objek tersebut adalah kendaraan yang ingin masuk ke lahan parkir, seperti ditunjukkan pada Gambar 4.19.

Gambar 4.19 Kendaraan Memasuki Lahan Parkir, Melewati Sensor Ultrasonic 2.



Gambar 4.20 Kendaraan Telah Lewat dan Palang Pintu Kembali Tertutup.

2 detik setelah kendaraan melewati sensor ultrasonik 2, maka pintu akan secara otomatis kembali tertutup, seperti ditunjukkan pada Gambar 4.20. Gambar 4.21 dan 4.22 menunjukkan proses keluar dari lahan parkir menggunakan MFRC522 *RFID Reader*. Prosesnya sama dengan masuk ke lahan parkir hanya saja sensor *RFID Reader* yang digunakan berbeda.

Gambar 4.21 Pengendara Menempelkan *RFID* tag ke MFRC522 *RFID Reader*.

Gambar 4.22 Autentikasi Berhasil dan Pintu Terbuka.

Gambar 4.23 Kendaraan Keluar dari Lahan Parkir dan Melewati Sensor Ultrasonic 2.

Gambar 4.24 Kendaraan Telah Lewat dan Palang Pintu Kembali Tertutup.

Gambar 4.23 dan Gambar 4.24 menunjukkan proses keluarnya sebuah kendaraan dari lahan parkir. Seperti saat masuk, 2 detik setelah kendaraan melewati sensor ultrasonik 2 maka pintu akan otomatis tertutup. Gambar 4.25 dan 4.26 menunjukkan data yang direkam pada *database* MySQL. Gambar 4.25 adalah data yang direkam saat kendaraan masuk,

dan Gambar 4.26 adalah data yang direkam saat kendaraan keluar dari lahan parkir.

317	Andre Fajar	masuk	2019-06-23 12:27:19.480106

Gambar 4.25 Data yang Direkam pada *Database* saat Masuk.

319	Andre Fajar	keluar	2019-06-23 12:28:13.243838

Gambar 4.26 Data yang Direkam pada *Database* saat Keluar.

Pada tahap ini, dilakukan pengujian sebanyak 10 kali, dengan hasil pengujian seperti ditunjukkan pada tabel 4.10. Yang dimaksud dengan "Delay 1s" pada tabel tersebut adalah UHF RFID Reader atau sensor ultrasonic terlambat 1 detik saat mendeteksi kendaraan maupun RFID tag. Tetapi sistem tetap dapat berlanjut dengan baik, meskipun terdapat delay tersebut.

Tabel 4.10 Hasil Pengujian Sistem Keseluruhan

No	Ultrasonic 1	UHF RFID	Ultrasonic 2	MFRC522 RFID	MySQL Database
1	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi
2	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi
3	Delay 1s	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi
4	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi
5	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi
6	Mendeteksi	Delay 1s	Mendeteksi	Mendeteksi	Mendeteksi
7	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi
8	Mendeteksi	Delay 1s	Mendeteksi	Mendeteksi	Mendeteksi
9	Delay 1s	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi
10	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi	Mendeteksi

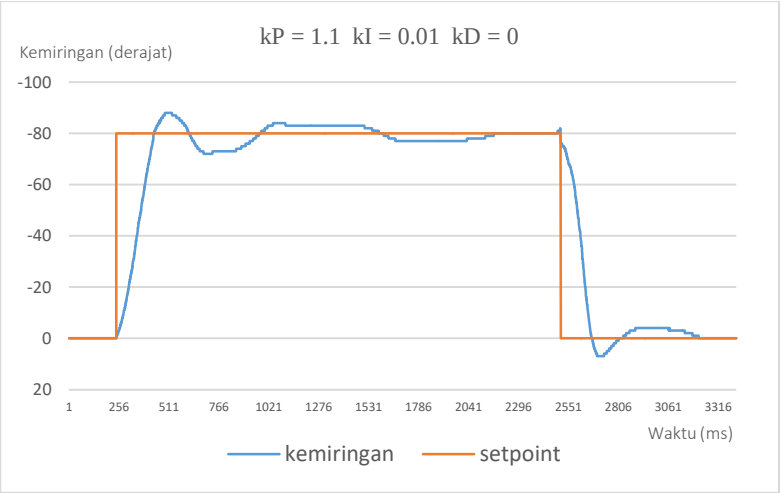
Setelah dilakukan pengujian keseluruhan, dan database terisi, terdapat 2 tabel yang digunakan, yaitu "rfidlist" dan "log". "rfidlist" digunakan untuk menyimpan data kartu yang terdaftar dan "log" digunakan untuk menyimpan rekaman akses lahan parkir. "rfidlist" mempunyai 10 entry, dan jika dilihat pada information_schema, mempunyai besar file sebesar 15.625kb. Begitu juga dengan "log", dengan isi 364 entry, information_schema menunjukkan besar filenya

15.625kb. Hal ini dikarenakan information_schema tidak bisa menunjukkan besar file secara akurat tiap waktunya. Fungsi information_schema akan menunjukkan perubahan besar file ketika terjadi perubahan jumlah entry yang besar. Setelah dilakukan pengujian, entry dibawah 600 hanya memakan disk size sebesar 15.625kb. Entry diantara 600 sampai 1000 memakan disk size sebesar 46.875kb. Entry diantara 1000 sampai 1500 memakan disk size sebesar 62.5kb. Kemudian entry diatas 1500 akan memakan disk size sebesar 78.125kb dan terus bertambah seiring dengan bertambahnya entry.

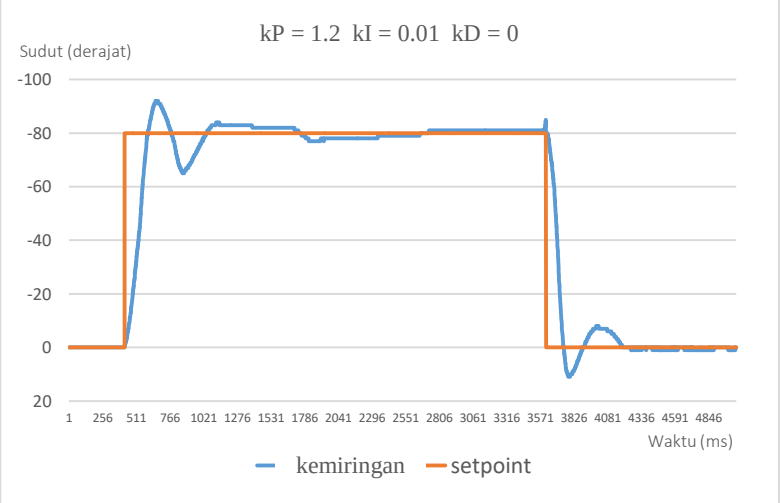
Selanjutnya dilakukan pengujian kontrol PID. Konstanta yang dipakai dalam pengujian pertama ini adalah : kP = 1.1, kI = 0.01, kD = 0. Output *integral* hanya digunakan ketika error kurang 20' dan lebih 20' dari *setpoint*. Output *integral* juga dibatasi, dengan batas maksimum sebesar 30. Sehingga meskipun sistem telah berjalan sangat lama, output tidak meningkat jumlahnya secara tak terkendali. Dengan konfigurasi seperti itu, respon sudut terhadap waktu yang didapatkan ditunjukkan pada Gambar 4.27. Dari grafik respon sudut dapat dilihat bahwa saat transisi pertama dari *setpoint* 0 ke *setpoint* -80, *rise time* yang didapat adalah 561 ms, *overshoot* maksimum sebesar 8 derajat, dan *settling time* sebesar 2309 ms. Saat transisi dari *setpoint* -80 ke *setpoint* 0, *rise time* yang didapat sebesar 464 ms, *overshoot* maksimum sebesar 7 derajat, dan *settling time* sebesar 875 ms.

Pada pengujian kedua, dilakukan dengan nilai konstanta kP = 1.2, kI = 0.01, dan kD = 0. Respon sudut terhadap waktu dari sistem dapat dilihat pada Gambar 4.28. Dari grafik respon sudut dapat dilihat bahwa saat transisi pertama dari setpoint 0 ke setpoint -80, rise time yang didapat adalah 520 ms, overshoot maksimum sebesar 13 derajat, dan settling time sebesar 1812 ms. Saat transisi dari setpoint -80 ke 0, rise time yang didapat sebesar 386 ms, overshoot maksimum sebesar 11 derajat, dan settling time yang didapat sebesar 1519 ms.

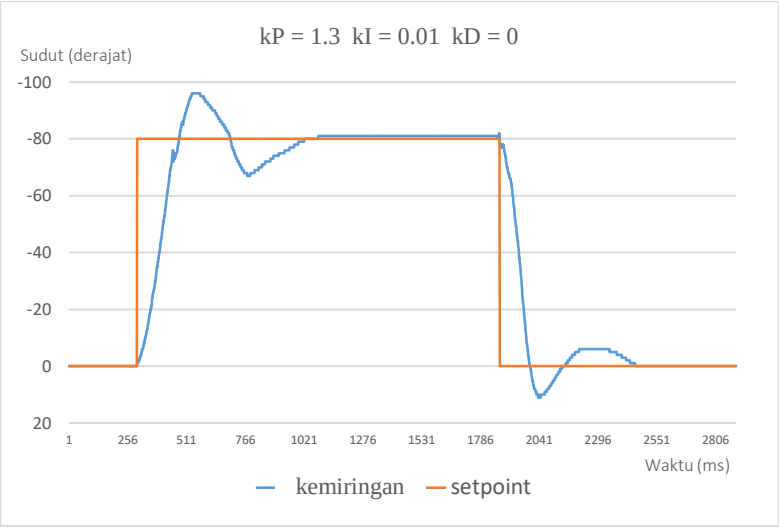
Pada pengujian ketiga, dilakukan dengan nilai konstanta kP = 1.3, kI = 0.01, dan kD = 0. Respon sudut terhadap waktu dari sistem dapat dilihat pada Gambar 4.29. Dari grafik respon sudut dapat dilihat bahwa saat transisi pertama dari setpoint 0 ke setpoint -80, risetime yang didapat adalah 535 ms, overshoot maksimumnya sebesar 14 derajat, dan settling time yang didapat sebesar 1941 ms. Saat transisi dari setpoint -80 ke 0, rise time yang didapat sebesar 381 ms, overshoot maksimum yang didapat adalah 10 derajat, dan settling time yang didapat adalah 1492 ms.



Gambar 4.27 Grafik Pengujian Pertama



Gambar 4.28 Grafik Pengujian Kedua



Gambar 4.29 Grafik Pengujian Ketiga.

Halaman ini sengaja dikosongkan.

BAB V
PENUTUP

5.1. Kesimpulan

Berdasarkan percobaan yang telah dilakukan pada penelitian ini, didapat beberapa kesimpulan yang didasarkan pada pengujian sistem. Secara keseluruhan sistem dapat berjalan dengan cukup baik, meskipun terdapat beberapa kendala. Dari 10 kali pengujian, sensor ultrasonic 1 terlambat membaca kendaraan sebanyak 2 kali. *UHF RFID Reader* juga terlambat membaca kendaraan sebanyak 2 kali dari total 10 kali pengujian.

Berdasarkan pengujian pembacaan *UHF RFID Reader*, semua data yang dibaca sesuai dengan data yang disimpan pada tag, sehingga *UHF RFID Reader* memiliki akurasi pembacaan 100%.

Berdasarkan pengujian MFRC522 *RFID Reader*, semua tag yang dibaca datanya sesuai dengan yang tersimpan pada tag, sehingga *RFID Reader* ini memiliki tingkat akurasi 100%.

Pada pengujian *database MySQL*, setiap ada kendaraan yang masuk dan keluar selalu berhasil disimpan rekaman aksesnya kedalam *database*. Dengan ukuran data sebesar 15.625 KB untuk tabel dengan jumlah data kurang dari 500, dan akan terus bertambah seiring dengan bertambahnya jumlah data data.

Berdasarkan pengujian kontrol PID, setelah diuji dengan berbagai konfigurasi, dapat disimpulkan bahwa angka kP = 1.1, kI = 0.01, dan kD = 0 adalah konfigurasi yang terbaik. Dikarenakan saat kP ditambah, *rise time* cenderung sama, namun *overshoot* maksimum yang dialami sistem bertambah 37.5%. Selain itu, rise time yang didapat pada sistem yaitu 561 ms saat naik dan 464 ms saat turun, sudah jauh lebih baik dari palang pintu komersial yang pada umumnya membutuhkan minimal 1.5 detik untuk membuka.

5.2. Saran

Saran untuk pengembangan tugas akhir ini adalah bisa digunakan motor DC yang lebih kuat torsinya, serta bisa dibuat aplikasi mobile (baik android atau iOS) untuk mengakses *database-database* yang ada. Bisa juga digunakan kamera untuk mendeteksi keberadaan kendaraan, atau untuk mengklasifikasikan kendaraan apa yang ingin masuk kedalam lahan parkir.

Halaman ini sengaja dikosongkan.

DAFTAR PUSTAKA

[1] "Badan Pusat Statistik." [Online]. Available: https://www.bps.go.id/linkTableDinamis/view/id/1133.

[2] R. Kannadasan, A. Krishnamoorthy, N. Prabakaran, K. Naresh, V. Vijayarajan, and G. Sivashanmugam, "Rfid Based Automatic Parking System," Australian Journal of Basic and Applied Sciences, p. 7, 2016.

[3] M. S. Mazlan, I. R. A Hamid, and H. Kamaludin, "Radio Frequency Identification (RFID) Based Car Parking System," JOIV : International Journal on Informatics Visualization, vol. 2, no. 4–2, p. 318, Sep. 2018.

[4] S. Rahman, "IoT Based Smart Parking System," vol. 4, no. 1, p. 7, 2019.

[5] C. Yudianto dan M. Rivai, "Sistem Pengamanan Gudang Senjata Menggunakan RFID dan Sidik Jari," Jurnal Teknik ITS, vol. 7, no. 1, Mar. 2018.

[6] S. Evdokimov, "RFID and the Internet of Things: Technology, Applications, and Security Challenges," Foundations and Trends® in Technology, Information and Operations Management, vol. 4, no. 2, pp. 105–185, 2010.

[7] R. Want, "An Introduction to RFID Technology," p. 9, 2006.

[8] N. Hasanah, "Optimasi Kontrol PID Secara Konvensional," Program Pascasarjana Magister Terapan Teknik Elektro Politeknik Elektronika Negeri Surabaya 2018.

[9] Cahtayu, "DC Motor Theory," p. 8.

[10] C. Ferrero, "Mechanical Metrology : Force and Torque.," p. 13.

[11] A. Aniedu, S. Uchenna, and G. N. Okechukwu, "Real Time Data Acquisition and Logging Using Gsm Technology," Advances in Multidisciplinary & Scientific Research, vol. 2, no. 2, p. 17, 2016.

[12] M. Maksimović, V. Vujović, N. Davidović, V. Milošević, and B. Perišić, "Raspberry Pi as Internet of Things hardware: Performances and Constraints," p. 7.

[13] M. Grinberg, Flask web development, First edition. Sebastopol, CA: O'Reilly, 2014.

[14] A. O. Eltahawey, "Hyper Text Markup Language HTML: A Tutorial," p. 26.

[15] A. Solichin Achmatim, MySQL Dari pemula Hingga Mahir. .

[16] B. Suprianto, “Aplikasi Sensor Ultrasonik Untuk Deteksi Posisi Jarak Pada Ruang Menggunakan Arduino Uno Bakhtiyar Arasada,” vol. 06, p. 8, 2017.
[17] O. G. Chiagozie and O. G. Nwaji, “Radio Frequency Identification (RFID) Based Attendance System with Automatic Door Unit,” vol. 2, no. 2, p. 17, 2012.
[18] Z. Pala and N. Inanc, “Utilizing RFID for Smart Parking Applications,” p. 19.
[19] A. Khanna and R. Anand, “IoT based smart parking system,” in 2016 International Conference on Internet of Things and Applications (IOTA), Pune, India, 2016, pp. 266–270.
[20] “EVOXEN - VAA.” <URL : http://www.vaa.co.id/ evoxen.html>
[21] “Doorking 1601 Parking Barrier Arm | Elite Gates.” [Online]. <URL : https://elitegates.net/proddetail.asp?prod=Doorking1601>
[22] “CAME,” Smart contactless, 08-Aug-2015.
[23] “Road barriers,” Nice, 07-Jan-2016. [Online]. <URL : https://www.niceforyou.com/en/road-barriers>
[24] “3M/ Federal APD.” [Online]. <URL : http://www.bernational controls.com/support-gates-federal.html>

LAMPIRAN

1. Program utama Python Raspberry Pi

```
import time
import MySQL.connector
import serial

mega = serial.Serial('/dev/ttyACM0',9600)
nano = serial.Serial('/dev/ttyUSB1',9600)
uhf = serial.Serial(
   port='/dev/ttyUSB0',
   baudrate = 9600,
   parity=serial.PARITY_NONE,
   stopbits=serial.STOPBITS_ONE,
   bytesize=serial.EIGHTBITS,
   timeout=0.5
   )
counter=0

mydb = MySQL.connector.connect(
   host="localhost",
   user="pi",
   passwd="raspberry",
   database="mydb")

def readmfrc():
   raw_uid = nano.readline()
   flushnano()
   str_uid = str(raw_uid)
   uid = str(str_uid[2:10])
   return uid

def readuhf():
   x=uhf.readline()
   y=str(x)
   uid=y[25:37]
   return uid

def post(name):
```

```
    mycursor = mydb.cursor()
    sql = ("INSERT INTO test (name, action ) values (%s, %s)")
    adr = (name, "masuk")
    mycursor.execute(sql,adr)
    mydb.commit()
    print(mycursor.rowcount, "record inserted")
    return

def postmfrc(name):
    mycursor = mydb.cursor()
    sql = ("INSERT INTO test (name, action ) values (%s, %s)")
    adr = (name, "keluar")
    mycursor.execute(sql,adr)
    mydb.commit()
    print(mycursor.rowcount, "record inserted")
    return

def getconfirm(uid):
    mycursor = mydb.cursor()
    sql = ("SELECT confirm FROM rfidlist WHERE RFID_enter
= %s")
    adr = (uid, )
    mycursor.execute(sql,adr)
    myresult = mycursor.fetchall()
    x = str(myresult)
    x = x[3:6]
    return x

def getconfirmmfrc(uid):
    mycursor = mydb.cursor()
    sql = ("SELECT confirm FROM rfidlist WHERE RFID_exit
= %s")
    adr = (uid, )
    mycursor.execute(sql,adr)
    myresult = mycursor.fetchall()
    x = str(myresult)
    x = x[3:6]
    return x
```

```
def getname(uid):
    mycursor = mydb.cursor()
    sql = ("SELECT name FROM rfidlist WHERE RFID_enter
= %s")
    adr = (uid, )
    mycursor.execute(sql,adr)
    myresult = mycursor.fetchall()
    x = str(myresult)
    x = x.replace("["," ")
    x = x.replace("("," ")
    x = x.replace("'"," ")
    x = x.replace("'"," ")
    x = x.replace(","," ")
    x = x.replace(")"," ")
    x = x.replace("]"," ")
    x = x.strip()
    return x

def getnamemfrc(uid):
    mycursor = mydb.cursor()
    sql = ("SELECT name FROM rfidlist WHERE RFID_exit
= %s")
    adr = (uid, )
    mycursor.execute(sql,adr)
    myresult = mycursor.fetchall()
    x = str(myresult)
    x = x.replace("["," ")
    x = x.replace("("," ")
    x = x.replace("'"," ")
    x = x.replace("'"," ")
    x = x.replace(","," ")
    x = x.replace(")"," ")
    x = x.replace("]"," ")
    x = x.strip()
    return x

def getnrp(uid):
    mycursor = mydb.cursor()
    sql = ("SELECT nrp FROM rfidlist WHERE RFID_enter = %s")
```

```
    adr = (uid, )
    mycursor.execute(sql,adr)
    myresult = mycursor.fetchall()
    x = str(myresult)
    x = x.replace("["," ")
    x = x.replace("("," ")
    x = x.replace("'"," ")
    x = x.replace("'"," ")
    x = x.replace(","," ")
    x = x.replace(")"," ")
    x = x.replace("]"," ")
    x = x.strip()
    return x

def getnrpmfrc(uid):
    mycursor = mydb.cursor()
    sql = ("SELECT nrp FROM rfidlist WHERE RFID_exit = %s")
    adr = (uid, )
    mycursor.execute(sql,adr)
    myresult = mycursor.fetchall()
    x = str(myresult)
    x = x.replace("["," ")
    x = x.replace("("," ")
    x = x.replace("'"," ")
    x = x.replace("'"," ")
    x = x.replace(","," ")
    x = x.replace(")"," ")
    x = x.replace("]"," ")
    x = x.strip()
    return x

def authenticate(uid):
    x = getconfirm(uid)
    if x == "yes":
        post(getname(uid))
        print(getname(uid) + " " + getnrp(uid))
        y = mega.readline()
        y = str(y)
        y = y[2]
```

```
        while y!="y":
            mega.write(b'1')
            y = mega.readline()
            y = str(y)
            y = y[2]
        return "go in"
    else:
        y = mega.readline()
        y = str(y)
        y = y[2]
        while y!="y":
            mega.write(b'2')
            y = mega.readline()
            y = str(y)
            y = y[2]
        return "kartu tidak terdaftar"

def authenticatemfrc(uid):
    x = getconfirmmfrc(uid)
    if x == "yes":
        postmfrc(getnamemfrc(uid))
        print(getnamemfrc(uid) + " " + getnrpmfrc(uid))
        flushnano()
        y = mega.readline()
        y = str(y)
        y = y[2]
        while y!="y":
            mega.write(b'1')
            y = mega.readline()
            y = str(y)
            y = y[2]
        return "selamat jalan"
    else:
        y = mega.readline()
        y = str(y)
        y = y[2]
        while y!="y":
            mega.write(b'2')
            y = mega.readline()
```

```
            y = str(y)
            y = y[2]
        return "kartu tidak terdaftar"

def flushuhf():
    while uhf.in_waiting > 0:
        sampah = uhf.readline()
    return

def flushnano():
    while nano.in_waiting > 0:
        sampah = nano.readline()
    return

while 1:
    if(nano.in_waiting >0):
        uid2 = readmfrc()
        flushnano()
        print(authenticatemfrc(uid2))
    if(mega.in_waiting >0):
        dataraw=mega.readline()
        data = str(dataraw)
        vehicle = data[2]
        print("\n")
        print (vehicle)
        if vehicle == "1":
            uid = str(readuhf())
            print(authenticate(uid))
            flushuhf()
        else :
            mega.write(b'3')
            nano.write(b'1')
            print("nope")
            flushuhf()
```

2. Program Flask Raspberry Pi

```
import time
from flask import Flask, render_template, request, flash, url_for,
redirect, session, g
```

```
from flask_mysqldb import MySQL
from flask_nav import Nav
from flask_nav.elements import Navbar, Subgroup, View, Link,
Text, Separator
import os

app = Flask(__name__)
app.secret_key = os.urandom(24)

app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'pi'
app.config['MYSQL_PASSWORD'] = 'raspberry'
app.config['MYSQL_DB'] = 'mydb'

MySQL = MySQL(app)

@app.before_request
def before_request():
    g.user = None
    if 'user' in session:
        g.user = session['user']

@app.route('/')
def default():
    return redirect(url_for('login'))

@app.route('/home')
def home():
    if g.user:
        return render_template("home.html", title='Home')
    return redirect(url_for('login'))

@app.route('/login', methods=['GET', 'POST'])
def login():
    session.pop('user',None)
    error = None
    cur = MySQL.connection.cursor()
    cur.execute("SELECT uname FROM auth WHERE confirm
= 'yes'")
```

```
    uname = cur.fetchall()
    uname = str(uname)
    uname = uname.replace("("," ")
    uname = uname.replace("'"," ")
    uname = uname.replace(","," ")
    uname = uname.replace(")"," ")
    uname = uname.strip()
    cur.execute("SELECT password FROM auth WHERE
confirm = 'yes'")
    password = cur.fetchall()
    password = str(password)
    password = password.replace("("," ")
    password = password.replace("'"," ")
    password = password.replace(","," ")
    password = password.replace(")"," ")
    password = password.strip()
    if request.method == 'POST':
        if request.form['username'] != uname or
request.form['password'] != password:
            session.pop('user',None)
            error = "Error! credentials wrong, please try again."
        else:
            session['user']=request.form['username']
            return redirect(url_for('home'))
    return render_template('login.html', error=error, title='Login')

@app.route('/changepass', methods=['GET','POST'])
def changepass():
    error = None
    cur = MySQL.connection.cursor()
    cur.execute("SELECT password FROM auth WHERE
confirm = 'yes'")
    password = cur.fetchall()
    password = str(password)
    password = password.replace("("," ")
    password = password.replace("'"," ")
    password = password.replace(","," ")
    password = password.replace(")"," ")
    password = password.strip()
```

```
    if request.method == 'POST':
        newpass = request.form['newpass']
        newpass = str(newpass)
        if request.form['oldpass'] != password:
            error = "Error! Old password doesn't match, please try
again."
        else:
            cur.execute("DELETE FROM auth WHERE confirm =
'yes'")
            MySQL.connection.commit()
            cur.execute("INSERT INTO auth(uname, password)
VALUES (%s, %s)", ("admin", newpass))
            MySQL.connection.commit()
            return redirect(url_for('login'))
    return render_template('changepass.html', error = error, title =
'Change Password')

@app.route('/database')
def database():
    if g.user:
        cur = MySQL.connection.cursor()
        cur.execute("SELECT * FROM rfidlist")
        data = cur.fetchall()
        cur.close()
        return render_template('database.html', result = data,
title='Database')
    return redirect(url_for('login'))

@app.route('/log',)
def log():
    if g.user:
        cur = MySQL.connection.cursor()
        cur.execute("SELECT * FROM test")
        data = cur.fetchall()
        cur.close()
        return render_template('log.html', result = data,
title='History')
    return redirect(url_for('login'))
```

```
@app.route('/success')
def success():
    if g.user:
        return render_template('success.html', title='Insert Success')
    return redirect(url_for('login'))

@app.route('/insert', methods=['GET', 'POST'])
def insert():
    if g.user:
        if request.method == "POST":
            details = request.form
            rfidin = details['rfidin']
            rfidout = details['rfidout']
            name = details['name']
            nrp = details['nrp']
            cur = MySQL.connection.cursor()
            cur.execute("INSERT INTO rfidlist(RFID_enter,
RFID_exit, name, nrp) VALUES (%s, %s, %s, %s)", (rfidin,
rfidout, name, nrp))
            MySQL.connection.commit()
            cur.close()
            return redirect(url_for('success'))
        return render_template('insert.html', title='Insert New')
    return redirect(url_for('login'))

@app.route('/delete', methods=['GET', 'POST'])
def delete():
    if g.user:
        if request.method == "POST":
            cur = MySQL.connection.cursor()
            cur.execute("delete from rfidlist where name = (%s)",
[request.form['name']])
            MySQL.connection.commit()
            cur.close()
            return redirect(url_for('delsuccess'))
        return render_template('delete.html', title='Delete')
    return redirect(url_for('login'))

@app.route('/delsuccess')
```

```
def delsuccess():
    if g.user:
        return render_template('delsuccess.html', title='Delete
Success')
    return redirect(url_for('login'))
```

3. **Program changepass.html**

```
<html>
<head>
<title>{{title}}</title>
</head>
<body>
<div class="containter">
<h1 align="center">Change your password</h1>
<br>
<form align="center" action="" method="post">
<input type="text" placeholder="Old Password"
name="oldpass" value="{{request.form.oldpass}}"> <br><br>
<input type="text" placeholder="New Password"
name="newpass" value="{{request.form.newpass}}">
<br><br>
<input class="btn btn-default" type="submit"
value="Change"> <br>
</form>
<center> <a href="/login"><input type="button"
value="Cancel"></a> </center> <br>
{% if error %}
<p align="center"
class="error"><strong>Error:</strong>{{error}}</p>
{% endif %}
</div>
</body>
</html>
```

4. **Program *database*.html**

```
{% extends "layout.html" %}
{% block content %}
    <h1 align="center">Kartu yang terdaftar</h1>
    <table align="center">
```

```
        <tr>
          <th>RFID Enter</th>
          <th>RFID Out</th>
          <th>Name</th>
          <th>NRP</th>
        </tr>
        {% for row in result %}
          <tr>
            <td> {{ row.1 }} </td>
            <td> {{ row.2 }} </td>
            <td> {{ row.3 }} </td>
            <td> {{ row.4 }} </td>
          </tr>
        {% endfor %}
      </table><br>
      <center>
        <a href="/insert"><input type="button" value="Insert
new entry"></a> <br><br>
        <a href="/delete"><input type="button" value="Delete
an entry"></a> <br>
      </center>
{% endblock content %}
```

5. Program delete.html

```
{% extends "layout.html" %}
{% block content %}
<h1 align="center" style="font-size:200%">Input the name you
want to delete</h1> <br>
<form align="center" action="" method="post">
<input type="text" placeholder="Name" name="name"
value="{{request.form.name}}"> <br><br>
<input class="btn btn-default" type="submit">
</form>
<center><a href="/database"><input type="button"
value="Cancel"></a> <br></center>
{% endblock content %}
```

6. **Program delsuccess.html**

```
{% extends "layout.html" %}
{% block content %}
<h1 align="center">Entry deleted successfully!</h1>
<p align="center">Press the button below to go back to the
previous page</p>
<center>
<a href="/database"><input type="button" value="go
back"></a>
</center>
{% endblock content %}
```

7. **Program home.html**

```
{% extends "layout.html" %}
{% block content %}
<h1 align="center">Welcome</h1>
<p align="center">This is the parking lot management homepage,
choose what you want to do below</p>
<center>
<a href="/database"><input type="button"
value="Database"></a> <br><br>
<a href="/log"><input type="button" value="History"></a>
<br><br>
</center>
{% endblock content %}
```

8. **Program insert.html**

```
{% extends "layout.html" %}
{% block content %}
<h1 align="center" style="font-size:200%">Enter your details
</h1> <br>
<form align="center" action="" method="post">
<input type="text" placeholder="RFID Enter" name="rfidin"
value="{{request.form.rfidin}}"> <br>
<input type="text" placeholder="RFID Out" name="rfidout"
value="{{request.form.rfidout}}"> <br>
<input type="text" placeholder="Name" name="name"
value="{{request.form.name}}"> <br>
<input type="text" placeholder="NRP" name="nrp"
```

```
value="{{request.form.nrp}}"> <br>
<input class="btn btn-default" type="submit">
</form>
<center><a href="/database"><input type="button"
value="Cancel"></a> <br></center>
{% endblock content %}
```

9. Program layout.html

```
<html>
<head>
<title>{{title}}</title>
<style>
table, th, td {
border: 1px solid black;
border-collapse:collapse;
padding:10px;
}

.container{
  width:80%;
  margin:0 auto;
  }

header{
  background:#55d6aa;
  }

header::after{
  content: '';
  display:table;
  clear:both;
  }

nav{
  float:right;
  }

nav ul{
  margin:0;
```

```
    padding:0;
    list-style:none;
    }

  nav li{
    display: inline-block;
    margin: 20px;
    }

  nav a{
    color:#444;
    text-decoration:none;
    text-transform:uppercase;
    }

  </style>
  </head>
  <body>
    <header>
    <div class="containter">
    <nav>
    <ul>
    <li><a href="/home">Home</a></li>
    <li><a href="/database">Database</a></li>
    <li><a href="/log">History</a></li>
    <li><a href="/login">Logout</a></li>
    </ul>
    </nav>
    </div>
    </header>

    {% block content %}{% endblock %}
  </body>
  </html>
```

10. Program log.html

```
{% extends "layout.html" %}
{% block content %}
    <h1 align="center">History akses lahan parkir</h1>
```

```
    <table align="center">
      <tr>
        <th>No</th>
        <th>Name</th>
        <th>Action</th>
        <th>Timestamp</th>
      </tr>
      {% for row in result %}
        <tr>
          <td> {{ row.0 }} </td>
          <td> {{ row.1 }} </td>
          <td> {{ row.2 }} </td>
          <td> {{ row.3 }} </td>
        </tr>
      {% endfor %}
    </table>
{% endblock content %}
```

11. Program styles.css

```
body {
  background: #fafafa;
  color: #333333;
  margin-top: 5rem;
}

.container{
   width: 80%;
   margin: 0 auto;
}

header{
   background: #55d6aa;
}

nav{
   float:right;
}
```

12. Program login.html

```
<html>
<head>
<title>{{title}}</title>
</head>
<body>
<div class="containter">
<h1 align="center">Please Login</h1>
<br>
<form align="center" action="" method="post">
<input type="text" placeholder="Username"
name="username" value="{{request.form.username}}">
<br><br>
<input type="password" placeholder="Password"
name="password" value="{{request.form.password}}">
<br><br>
<input class="btn btn-default" type="submit" value="Login">
<br>
</form>
<center> <a href="/changepass"><input type="button"
value="Change Password"></a> </center> <br>
{% if error %}
<p align="center"
class="error"><strong>Error:</strong>{{error}}</p>
{% endif %}
</div>
</body>
</html>
```

13. Program success.html

```
{% extends "layout.html" %}
{% block content %}
<h1 align="center">Entry inserted successfully!</h1>
<p align="center">Press the button below to go back to the
previous page</p>
<center>
<a href="/database"><input type="button" value="go
back"></a>
```

```
</center>
{% endblock content %}
```

14. Program Arduino Mega

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
const int trig = 53;          //trig buat ultrasonic
const int trig2 = 25;
const int echo = 51;          //echo buat ultrasonic
const int echo2 = 23;
const int ledopen = 45;       //led kalo auth berhasil
const int ledclose = 49;      //led kalo auth gagal
const int ledstandby = 47;    //led kalo gaada orang lewat
char r[3];                    //array to story serial value from raspi
int distance;                 //distance buat ultrasonic
int distance2;
long duration;
long duration2;
unsigned long waktu;

void setup() {
  // put your setup code here, to run once:
lcd.begin(16,2);

pinMode(trig, OUTPUT);        //tris buat ultrasonic
pinMode(echo, INPUT);         //echo buat ultrasonic
pinMode(trig2, OUTPUT);
pinMode(echo2, INPUT);
pinMode(22, OUTPUT);
Serial.begin(9600);
pinMode(ledopen, OUTPUT);     //led kalo auth berhasil
pinMode(ledclose, OUTPUT);    //led kalo auth gagal
pinMode(ledstandby, OUTPUT);  //led kalo standby
digitalWrite(ledopen, LOW);   //inisialisasi biar awalnya mati
digitalWrite(22,LOW);
digitalWrite(ledclose, LOW);  //inisialisasi biar awalnya mati

}
```

```
void printlcdint(int x, int y, int objek){
  lcd.setCursor(x,y);
  lcd.print("               ");
  lcd.setCursor(x,y);
  lcd.print(objek);
  delay(50);
  }

void printlcdstring(int x, int y, char objek[]){
  lcd.setCursor(x,y);
  lcd.print("               ");
  lcd.setCursor(x,y);
  lcd.print(objek);
  delay(50);
  }

int ultrasonicread(int echo, int trig){

  digitalWrite(trig,LOW);
  delayMicroseconds(2);

  digitalWrite(trig,HIGH);
  delayMicroseconds(10);
  digitalWrite(trig,LOW);

  duration = pulseIn(echo,HIGH);
  distance = duration*0.034/2;
  return distance;
  }

void loop() {
  // put your main code here, to run repeatedly:

distance = ultrasonicread(echo,trig);

if(distance<60){            //if ultrasonic 1 deteksi, send "1" to raspi
over serial
  Serial.println("1");       //else, send "0" to raspi over serial
```

```
  }
  else{
  Serial.println("0");
  }

delay(500);                //delay 1 detik biar loopnya ga kecepetan

if(Serial.available()){       //if di serial ada data, then
  Serial.readBytes(r, 1);     //datanya dibaca, di store ke char r[0]
  distance2 = ultrasonicread(echo2,trig2);

  while(Serial.available()>0){          //flush serial
    Serial.read();
    }

  if(r[0] == '1'){                      //if data serial = '1' (uid dikenali)
    Serial.println("y");
    distance2 = ultrasonicread(echo2,trig2);
    digitalWrite(ledopen,HIGH);
    digitalWrite(22,HIGH);
    while(distance2>60){                //selama ultrasonic2 belom
mendeteksi
      distance2 = ultrasonicread(echo2,trig2);
      digitalWrite(ledopen,HIGH);           //ledopen menyala
      digitalWrite(22,HIGH);
      digitalWrite(ledclose,LOW);           //ledclose mati
      digitalWrite(ledstandby,LOW);         //ledstandby mati
      }
    delay(500);
    while(distance2<60){
      distance2 = ultrasonicread(echo2,trig2);
      digitalWrite(ledopen,HIGH);
      digitalWrite(22,HIGH);
      digitalWrite(ledclose,LOW);
      digitalWrite(ledstandby,LOW);
      }
    delay(2000);
    digitalWrite(ledopen,LOW);          //setelah ultrasonic 2
mendeteksi maka ledopen mati
```

```
      digitalWrite(22,LOW);
      }
    else if(r[0]=='2'){
      Serial.println("y");
      digitalWrite(ledopen,LOW);        //if data serial = '2' (uid tidak
    dikenali)
      digitalWrite(22,LOW);
      digitalWrite(ledstandby,LOW);
      digitalWrite(ledclose,HIGH);      //maka ledclose menyala selama
    1 detik lalu mati
      }
    else if(r[0]=='3'){
      digitalWrite(ledstandby,HIGH);    //kalo tidak ada orang lewat
    ledstandby nyala
      digitalWrite(ledopen,LOW);
      digitalWrite(ledopen,LOW);
      digitalWrite(ledclose,LOW);
      }
    }
  }
```

15. Program Arduino Nano MFRC522

```
#include <SPI.h>
#include <MFRC522.h>
#define RST_PIN 9
#define SS_PIN 10

MFRC522 MFRC522(SS_PIN, RST_PIN);

char r[3];

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  SPI.begin();
  MFRC522.PCD_Init();
}

void loop() {
```

```
  // put your main code here, to run repeatedly:

  if ( ! MFRC522.PICC_IsNewCardPresent()) {
    return;
  }
  if ( ! MFRC522.PICC_ReadCardSerial()) {
    return;
  }

String UID = "";
  for (byte m = 0; m < MFRC522.uid.size; m++){
    UID = UID + String(MFRC522.uid.uidByte[m], HEX);
}
UID.toUpperCase();
UID = "0" + UID;
int k = UID.length();
if (k >= 9){
  UID = UID.substring(1);
  }

if(Serial.available()){
  Serial.readBytes(r,1);
  while(Serial.available()>0){
    int sampah = Serial.read();
    }
  if(r[0]=='1'){
    Serial.println(UID);
}
}
else{
  int sampah = 1;
  }

MFRC522.PICC_HaltA();
}
```

16. Program Arduino Nano PID

```
#include <MPU6050_tockn.h>
#include <Wire.h>
```

```
MPU6050 mpu6050(Wire);
int x;
int tombol = A0;
int val;
int pwmoffset;
#define PIN_INPUT 0
#define PIN_OUTPUT 3

//Define Variables we'll be connecting to
float Input, Output, Error, Currenttime, Elapsedtime, PIDp, PIDd;
float Lasterror = 0;
float PIDi = 0;
float Prevtime = 0;
float Setpoint;
float prevSet;

//Specify the links and initial tuning parameters
double Kp=1.3, Ki=0.01, Kd=0;

int RPWM=5;
int LPWM=6;
int L_EN=7;
int R_EN=8;


int pid(float sudut){
  Input = sudut;
  Error = Setpoint-Input;
  Currenttime = millis();
  Elapsedtime = (Currenttime - Prevtime)/1000;
  PIDp = Kp*Error;
  if((Error<15) && (Error>-15)){
  PIDi = PIDi+(Ki*Error);
  if(PIDi>20){
    PIDi = 20;
    }
  else if(PIDi<-20){
    PIDi = -20;
```

```
    }
   }
   else{PIDi = 0;}
   PIDd = Kd*(Error-Lasterror)/Elapsedtime;
   Output = PIDp+PIDi+PIDd;
   if(Output>255){
     Output=255;
    }
   else if(Output<-255){
     Output=-255;
    }
   Prevtime = Currenttime;
   Lasterror = Error;
   return Output;
   }

 void setup() {
   // put your setup code here, to run once:
   Serial.begin(115200);
   Wire.begin();
   mpu6050.begin();
   mpu6050.calcGyroOffsets(false);

   pinMode(tombol,INPUT);
   for(int i=5;i<9;i++){
    pinMode(i,OUTPUT);
   }
    for(int i=5;i<9;i++){
    digitalWrite(i,LOW);
   }
   Serial.println("tunggu 5 detik");
   delay(5000);
 }

 void loop() {
   // put your main code here, to run repeatedly:
   digitalWrite(L_EN,HIGH);
   digitalWrite(R_EN,HIGH);
```

```
  val = analogRead(tombol);
  if(val<500){
    Setpoint = 0;
  }
  else{
    Setpoint = -90;
    }
  mpu6050.update();
  x = mpu6050.getAngleX();
  x = -x;
  if(x>360){
    x=x-360;
    }
  pwmoffset = ((90+x)/3);
  Output = pid(x);
  if(Output<=0){
  Output = -Output;
  Output = Output + pwmoffset;
  if(Output>255){
    Output = 255;
    }
  analogWrite(LPWM,Output);
//  Serial.print("LPWM : ");
//  Serial.print(Output);
  }
  else if(Output>0){
  analogWrite(RPWM,Output);
//  Serial.print("RPWM : ");
//  Serial.print(Output);
  }
//  Serial.print(" Error : ");
//  Serial.print(Error);
//  Serial.print(" Sudut : ");
  Serial.println(x);
//  Serial.print(" Setpoint : ");
//  Serial.print(Setpoint);
//  Serial.print(" PIDi : ");
//  Serial.print(PIDi);
//  Serial.print(" PWM offset : ");}
```

17. Datasheet MPU6050

3 Register Map

The register map for the MPU-60X0 is listed below.

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0D	13	SELF_TEST_X	R/W	XA_TEST[4-2]			XG_TEST[4-0]				
0E	14	SELF_TEST_Y	R/W	YA_TEST[4-2]			YG_TEST[4-0]				
0F	15	SELF_TEST_Z	R/W	ZA_TEST[4-2]			ZG_TEST[4-0]				
10	16	SELF_TEST_A	R/W	RESERVED		XA_TEST[1-0]		YA_TEST[1-0]		ZA_TEST[1-0]	
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	-	-	-	FS_SEL [1:0]		-	-	-
1C	28	ACCEL_CONFIG	R/W	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]				
23	35	FIFO_EN	R/W	TEMP _FIFO_EN	XG _FIFO_EN	YG _FIFO_EN	ZG _FIFO_EN	ACCEL _FIFO_EN	SLV2 _FIFO_EN	SLV1 _FIFO_EN	SLV0 _FIFO_EN
24	36	I2C_MST_CTRL	R/W	MULT _MST_EN	WAIT _FOR_ES	SLV_3 _FIFO_EN	I2C_MST _P_NSR	I2C_MST_CLK[3:0]			
25	37	I2C_SLV0_ADDR	R/W	I2C_SLV0 _RW	I2C_SLV0_ADDR[6:0]						
26	38	I2C_SLV0_REG	R/W	I2C_SLV0_REG[7:0]							
27	39	I2C_SLV0_CTRL	R/W	I2C_SLV0 _EN	I2C_SLV0 _BYTE_SW	I2C_SLV0 _REG_DIS	I2C_SLV0 _GRP	I2C_SLV0_LEN[3:0]			
28	40	I2C_SLV1_ADDR	R/W	I2C_SLV1 _RW	I2C_SLV1_ADDR[6:0]						
29	41	I2C_SLV1_REG	R/W	I2C_SLV1_REG[7:0]							
2A	42	I2C_SLV1_CTRL	R/W	I2C_SLV1 _EN	I2C_SLV1 _BYTE_SW	I2C_SLV1 _REG_DIS	I2C_SLV1 _GRP	I2C_SLV1_LEN[3:0]			
2B	43	I2C_SLV2_ADDR	R/W	I2C_SLV2 _RW	I2C_SLV2_ADDR[6:0]						
2C	44	I2C_SLV2_REG	R/W	I2C_SLV2_REG[7:0]							
2D	45	I2C_SLV2_CTRL	R/W	I2C_SLV2 _EN	I2C_SLV2 _BYTE_SW	I2C_SLV2 _REG_DIS	I2C_SLV2 _GRP	I2C_SLV2_LEN[3:0]			
2E	46	I2C_SLV3_ADDR	R/W	I2C_SLV3 _RW	I2C_SLV3_ADDR[6:0]						
2F	47	I2C_SLV3_REG	R/W	I2C_SLV3_REG[7:0]							
30	48	I2C_SLV3_CTRL	R/W	I2C_SLV3 _EN	I2C_SLV3 _BYTE_SW	I2C_SLV3 _REG_DIS	I2C_SLV3 _GRP	I2C_SLV3_LEN[3:0]			
31	49	I2C_SLV4_ADDR	R/W	I2C_SLV4 _RW	I2C_SLV4_ADDR[6:0]						
32	50	I2C_SLV4_REG	R/W	I2C_SLV4_REG[7:0]							
33	51	I2C_SLV4_DO	R/W	I2C_SLV4_DO[7:0]							
34	52	I2C_SLV4_CTRL	R/W	I2C_SLV4 _EN	I2C_SLV4 _INT_EN	I2C_SLV4 _REG_DIS	I2C_MST_DLY[4:0]				
35	53	I2C_SLV4_DI	R	I2C_SLV4_DI[7:0]							
36	54	I2C_MST_STATUS	R	PASS_ THROUGH	I2C_SLV4 _DONE	I2C_LOST _ARB	I2C_SLV4 _NACK	I2C_SLV3 _NACK	I2C_SLV2 _NACK	I2C_SLV1 _NACK	I2C_SLV0 _NACK
37	55	INT_PIN_CFG	R/W	INT_LEVEL	INT_OPEN	LATCH _INT_EN	INT_RD _CLEAR	FSYNC_ INT_LEVEL	FSYNC _INT_EN	I2C _BYPASS _EN	-
38	56	INT_ENABLE	R/W	-	-	-	FIFO _OFLOW _EN	I2C_MST _INT_EN	-	-	DATA _RDY_EN
3A	58	INT_STATUS	R	-	-	-	FIFO _OFLOW _INT	I2C_MST _INT	-	-	DATA _RDY_INT

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							
49	73	EXT_SENS_DATA_00	R	EXT_SENS_DATA_00[7:0]							
4A	74	EXT_SENS_DATA_01	R	EXT_SENS_DATA_01[7:0]							
4B	75	EXT_SENS_DATA_02	R	EXT_SENS_DATA_02[7:0]							
4C	76	EXT_SENS_DATA_03	R	EXT_SENS_DATA_03[7:0]							
4D	77	EXT_SENS_DATA_04	R	EXT_SENS_DATA_04[7:0]							
4E	78	EXT_SENS_DATA_05	R	EXT_SENS_DATA_05[7:0]							
4F	79	EXT_SENS_DATA_06	R	EXT_SENS_DATA_06[7:0]							
50	80	EXT_SENS_DATA_07	R	EXT_SENS_DATA_07[7:0]							
51	81	EXT_SENS_DATA_08	R	EXT_SENS_DATA_08[7:0]							
52	82	EXT_SENS_DATA_09	R	EXT_SENS_DATA_09[7:0]							
53	83	EXT_SENS_DATA_10	R	EXT_SENS_DATA_10[7:0]							
54	84	EXT_SENS_DATA_11	R	EXT_SENS_DATA_11[7:0]							
55	85	EXT_SENS_DATA_12	R	EXT_SENS_DATA_12[7:0]							
56	86	EXT_SENS_DATA_13	R	EXT_SENS_DATA_13[7:0]							
57	87	EXT_SENS_DATA_14	R	EXT_SENS_DATA_14[7:0]							
58	88	EXT_SENS_DATA_15	R	EXT_SENS_DATA_15[7:0]							
59	89	EXT_SENS_DATA_16	R	EXT_SENS_DATA_16[7:0]							
5A	90	EXT_SENS_DATA_17	R	EXT_SENS_DATA_17[7:0]							
5B	91	EXT_SENS_DATA_18	R	EXT_SENS_DATA_18[7:0]							
5C	92	EXT_SENS_DATA_19	R	EXT_SENS_DATA_19[7:0]							
5D	93	EXT_SENS_DATA_20	R	EXT_SENS_DATA_20[7:0]							
5E	94	EXT_SENS_DATA_21	R	EXT_SENS_DATA_21[7:0]							
5F	95	EXT_SENS_DATA_22	R	EXT_SENS_DATA_22[7:0]							
60	96	EXT_SENS_DATA_23	R	EXT_SENS_DATA_23[7:0]							
63	99	I2C_SLV0_DO	R/W	I2C_SLV0_DO[7:0]							
64	100	I2C_SLV1_DO	R/W	I2C_SLV1_DO[7:0]							
65	101	I2C_SLV2_DO	R/W	I2C_SLV2_DO[7:0]							
66	102	I2C_SLV3_DO	R/W	I2C_SLV3_DO[7:0]							

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	FIFO_COUNT[15:8]							
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-

Note: Register Names ending in _H and _L contain the high and low bytes, respectively, of an internal register value.

In the detailed register tables that follow, register names are in capital letters, while register values are in capital letters and italicized. For example, the ACCEL_XOUT_H register (Register 59) contains the 8 most significant bits, *ACCEL_XOUT*[15:8], of the 16-bit X-Axis accelerometer measurement, *ACCEL_XOUT*.

The reset value is 0x00 for all registers other than the registers below.

- Register 107: 0x40.
- Register 117: 0x68.

18. Datasheet ATmega328P

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 20 MIPS Throughput at 20 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 4/8/16/32K Bytes of In-System Self-Programmable Flash progam memory (ATmega48P/88P/168P/328P)
  - 256/512/512/1K Bytes EEPROM (ATmega48P/88P/168P/328P)
  - 512/1K/1K/2K Bytes Internal SRAM (ATmega48P/88P/168P/328P)
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C(1)
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and QFN/MLF package
    - Temperature Measurement
  - 6-channel 10-bit ADC in PDIP Package
    - Temperature Measurement
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface (Philips I²C compatible)
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48P/88P/168PV
  - 2.7 - 5.5V for ATmega48P/88P/168P
  - 1.8 - 5.5V for ATmega328P
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - ATmega48P/88P/168PV: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 10 MHz @ 2.7 - 5.5V
  - ATmega48P/88P/168P: 0 - 10 MHz @ 2.7 - 5.5V, 0 - 20 MHz @ 4.5 - 5.5V
  - ATmega328P: 0 - 4 MHz @ 1.8 - 5.5V, 0 - 10 MHz @ 2.7 - 5.5V, 0 - 20 MHz @ 4.5 - 5.5V
- Low Power Consumption at 1 MHz, 1.8V, 25°C for ATmega48P/88P/168P:
  - Active Mode: 0.3 mA
  - Power-down Mode: 0.1 µA
  - Power-save Mode: 0.8 µA (Including 32 kHz RTC)

Note: 1. See "Data Retention" on page 7 for details.

8-bit AVR® Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash

ATmega48P/V*
ATmega88P/V*
ATmega168P/V
ATmega328P**

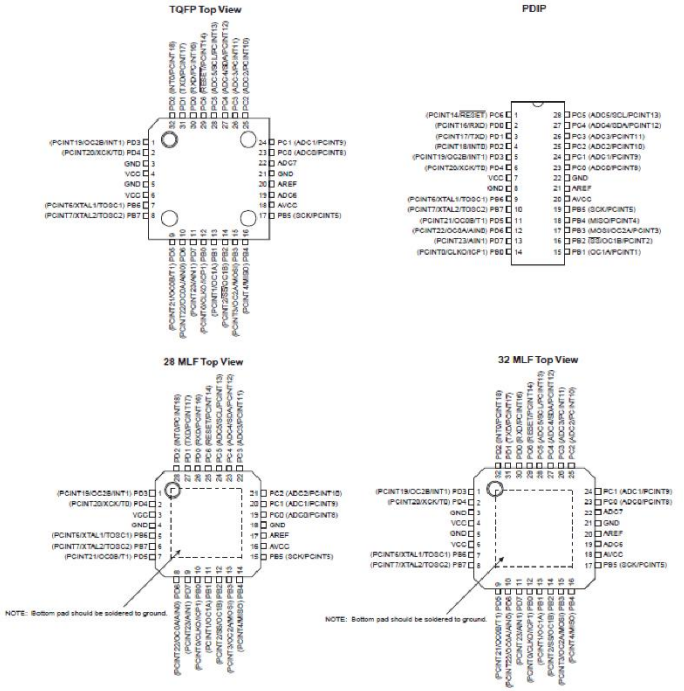
**Preliminary

* Not recommended for new designs.

Rev. 8025I–AVR–02/09

1. Pin Configurations

Figure 1-1. Pinout ATmega48P/88P/168P/328P



19. Datasheet Atmega2560

Atmel

Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V

8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash

DATASHEET

Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 135 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 × 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16MHz
  - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
  - 64K/128K/256KBytes of In-System Self-Programmable Flash
  - 4Kbytes EEPROM
  - 8Kbytes Internal SRAM
  - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/ 100 years at 25°C
  - Optional Boot Code Section with Independent Lock Bits
    - In-System Programming by On-chip Boot Program
    - True Read-While-Write Operation
  - Programming Lock for Software Security
    - Endurance: Up to 64Kbytes Optional External Memory Space
- Atmel® QTouch® library support
  - Capacitive touch buttons, sliders and wheels
  - QTouch and QMatrix acquisition
  - Up to 64 sense channels
- JTAG (IEEE® std. 1149.1 compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four 8-bit PWM Channels
  - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
  - Output Compare Modulator
  - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
  - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
  - Master/Slave SPI Serial Interface
  - Byte Oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
  - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
  - 64-pad QFN/MLF, 64-lead TQFP (ATmega1281/2561)
  - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
  - RoHS/Fully Green
- Temperature Range:
  - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
  - Active Mode: 1MHz, 1.8V: 500µA
  - Power-down Mode: 0.1µA at 1.8V
- Speed Grade:
  - ATmega640V/ATmega1280V/ATmega1281V:
    - 0 - 4MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega2560V/ATmega2561V:
    - 0 - 2MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
  - ATmega640/ATmega1280/ATmega1281:
    - 0 - 8MHz @ 2.7V - 5.5V, 0 - 16MHz @ 4.5V - 5.5V
  - ATmega2560/ATmega2561:
    - 0 - 16MHz @ 4.5V - 5.5V

2549Q-AVR-02/2014

Figure 1-2. CBGA-pinout ATmega640/1280/2560

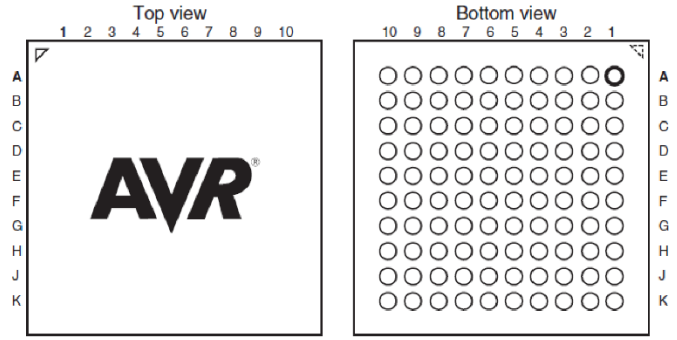


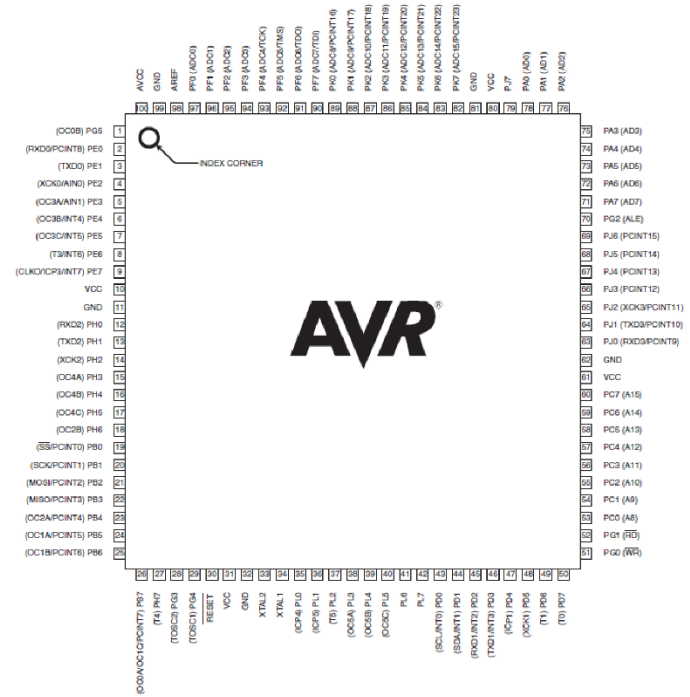
Table 1-1. CBGA-pinout ATmega640/1280/2560

	1	2	3	4	5	6	7	8	9	10
A	GND	AREF	PF0	PF2	PF5	PK0	PK3	PK6	GND	VCC
B	AVCC	PG5	PF1	PF3	PF6	PK1	PK4	PK7	PA0	PA2
C	PE2	PE0	PE1	PF4	PF7	PK2	PK5	PJ7	PA1	PA3
D	PE3	PE4	PE5	PE6	PH2	PA4	PA5	PA6	PA7	PG2
E	PE7	PH0	PH1	PH3	PH5	PJ6	PJ5	PJ4	PJ3	PJ2
F	VCC	PH4	PH6	PB0	PL4	PD1	PJ1	PJ0	PC7	GND
G	GND	PB1	PB2	PB5	PL2	PD0	PD5	PC5	PC6	VCC
H	PB3	PB4	RESET	PL1	PL3	PL7	PD4	PC4	PC3	PC2
J	PH7	PG3	PB6	PL0	XTAL2	PL6	PD3	PC1	PC0	PG1
K	PB7	PG4	VCC	GND	XTAL1	PL5	PD2	PD6	PD7	PG0

Note: The functions for each pin is the same as for the 100 pin packages shown in Figure 1-1 on page 2.

1. Pin Configurations

Figure 1-1. TQFP-pinout ATmega640/1280/2560



20. Datasheet HC-SR04

Tech Support: services@elecfreaks.com

Ultrasonic Ranging Module HC - SR04

⌧ Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

⌧ Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

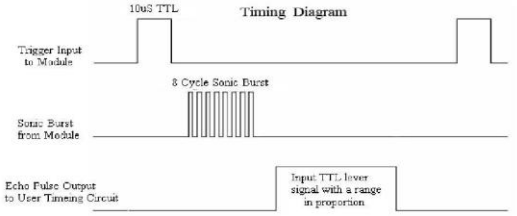
Electric Parameter

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

Vcc Trig Echo GND

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



21. Datasheet MFRC522 RFID Reader

MFRC522

Contactless Reader IC

Rev. 3.2 — 22 May 2007
112132

Product data sheet
PUBLIC INFORMATION

1. Introduction

This document describes the functionality of the contactless reader/writer MFRC522. It includes the functional and electrical specifications.

2. General description

The MFRC522 is a highly integrated reader/writer for contactless communication at 13.56 MHz. The MFRC522 reader supports ISO 14443A / MIFARE® mode.

The MFRC522's internal transmitter part is able to drive a reader/writer antenna designed to communicate with ISO/IEC 14443A/MIFARE® cards and transponders without additional active circuitry. The receiver part provides a robust and efficient implementation of a demodulation and decoding circuitry for signals from ISO/IEC 14443A/MIFARE® compatible cards and transponders. The digital part handles the complete ISO/IEC 14443A framing and error detection (Parity & CRC).The MFRC522 supports MIFARE®Classic (e.g. MIFARE® Standard) products. The MFRC522 supports contactless communication using MIFARE® higher transfer speeds up to 848 kbit/s in both directions.

Various host interfaces are implemented:

- SPI interface
- serial UART (similar to RS232 with voltage levels according pad voltage supply)
- I^2C interface.

3. Features

- Highly integrated analog circuitry to demodulate and decode responses
- Buffered output drivers to connect an antenna with minimum number of external components
- Supports ISO/IEC 14443A / MIFARE®
- Typical operating distance in Reader/Writer mode for communication to a ISO/IEC 14443A / MIFARE® up to 50 mm depending on the antenna size and tuning
- Supports MIFARE® Classic encryption in Reader/Writer mode
- Supports ISO/IEC 14443A higher transfer speed communication up to 848 kbit/s
- Support of the MFIN / MFOUT
- Additional power supply to directly supply the smart card IC connected via MFIN / MFOUT
- Supported host interfaces

- SPI interface up to 10 Mbit/s
- I^2C interface up to 400 kbit/s in Fast mode, up to 3400 kbit/s in High-speed mode
- serial UART in different transfer speeds up to 1228.8 kbit/s, framing according to the RS232 interface with voltage levels according pad voltage supply

- Comfortable 64 byte send and receive FIFO-buffer
- Flexible interrupt modes
- Hard reset with low power function
- Power-down mode per software
- Programmable timer
- Internal oscillator to connect 27.12 MHz quartz
- 2.5 - 3.3 V power supply
- CRC Co-processor
- Free programmable I/O pins
- Internal self test

4. Quick reference data

Table 1. Quick reference data

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
AV_{DD}	Supply Voltage	$AV_{SS} = DV_{SS} = PV_{SS} = TV_{SS} = 0$ V, $PV_{DD} \leq AV_{DD} = DV_{DD} = TV_{DD}$	[1][2]	2.5	-	3.6	V
DV_{DD}			[1][2]				
TV_{DD}			[1][2]				
PV_{DD}	Pad power supply	$AV_{SS} = DV_{SS} = PV_{SS} = TV_{SS} = 0$ V, $PV_{DD} \leq AV_{DD} = DV_{DD} = TV_{DD}$	[3]	1.6	-	3.6	V
SV_{DD}	MFIN/MFOUT Pad Power Supply	$AV_{SS} = DV_{SS} = PV_{SS} = TV_{SS} = 0$ V,		1.6	-	3.6	V
I_{HPD}	Hard Power-down Current	$AV_{DD} = DV_{DD} = TV_{DD} = PV_{DD} = 3$ V, N_{RESET} = LOW	[4]	-	-	5	µA
I_{SPD}	Soft Power-down Current	$AV_{DD} = DV_{DD} = TV_{DD} = PV_{DD} = 3$ V, RF level detector on	[4]	-	-	10	µA
I_{DVDD}	Digital Supply Current	$DV_{DD} = 3$ V		-	6.5	9	mA
I_{AVDD}	Analog Supply Current	$AV_{DD} = 3$ V, bit RCVOff = 0		-	7	10	mA
$I_{AVDD,RCVOFF}$	Analog Supply Current, receiver switched off	$AV_{DD} = 3$ V, bit RCVOff = 1		-	3	5	mA
I_{PVDD}	Pad Supply Current		[2]	-	-	40	mA
I_{TVDD}	Transmitter Supply Current	Continuous Wave	[1][3][8]	-	60	100	mA
T_{amb}	operating ambient temperature			-25		+85	°C

[1] Supply voltage below 3 V reduces the performance (e.g. the achievable operating distance).

[2] AV_{DD}, DV_{DD} and TV_{DD} shall always be on the same voltage level.

[3] PV_{DD} shall always be on the same or lower voltage level than DV_{DD}.

[4] I_{TVDD} depends on TV_{DD} and the external circuitry connected to Tx1 and Tx2

[5] I_{PVDD} depends on the overall load at the digital pins.

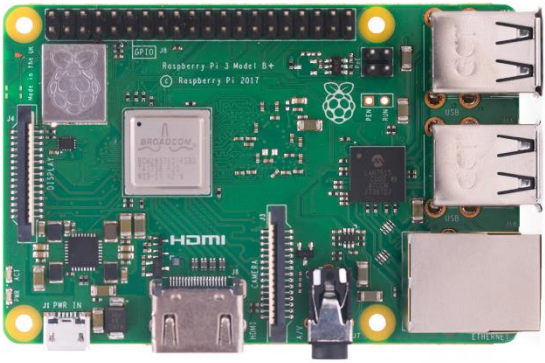
[6] During operation with a typical circuitry the overall current is below 100 mA.

[7] I_{SPD} and I_{HPD} are the total currents over all supplies.

[8] Typical value using a complementary driver configuration and an antenna matched to 40 Ω between TX1 and TX2 at 13.56 MHz

22. Datasheet Raspberry Pi 3B+

Overview



The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4 GHz, dual-band 2.4 GHz and 5 GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT

The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

raspberrypi.org

Halaman ini sengaja dikosongkan.

BIODATA PENULIS

Penulis lahir di Sidoarjo pada tanggal 15 Maret 1998. Penulis mulai menjalani studi Strata-1 di Institut Teknologi Sepuluh Nopember pada tahun 2015. Selama menjadi mahasiswa penulis aktif mengikuti kegiatan baik di dalam kampus maupun di luar kampus. Di tahun pertama kuliah penulis sering berpartisipasi di organisasi kampus baik di tingkat himpunan, fakultas, maupun institut. Di tahun kedua penulis bergabung dengan organisasi luar kampus yang bernama AIESEC, yaitu sebuah organisasi terbesar di dunia yang dijalankan oleh anak muda. AIESEC bertujuan untuk mencapai "*Peace and Fulfilment of Humankind's Potential*" atau perdamaian dunia dengan cara mengembangkan potensi pemuda di dunia melalui *cross-cultural exchanges* di lingkungan yang menantang sehingga pemuda di dunia bisa menjadi versi yang terbaik dari diri mereka masing-masing. Penulis berada di AIESEC selama 3 tahun, dan disana belajar banyak hal tentang manajemen diri, manajemen orang lain, manajemen *event*, desain grafis, dan masih banyak lagi yang lain.

E-mail : faisal@tantowi.com
Whatsapp : +6287865145405
Instagram : @fffffaaaaaiiiii
LinkedIn : /in/faishal-abdur-rahman/