

Seminar II

**SISTEM IDENTIFIKASI KENDARAAN PADA PEMARKIRAN DENGAN
PENGENALAN CITRA PLAT DAN PEMBACAAN RFID**



Oleh
MUH FIKRI SATRIA AMDANI
H131 16 501

Pembimbing Utama : Dr. Eng. Armin Lawi, S.Si., M.Eng.
Pembimbing Pertama : Musfira Putri Lukman, S.T., M.T.
Pengaji :
1. Dr. Hendra, S.Si., M.Kom.
2. Nur Hilal A Syahrir, S.Si., M.Si.

PROGRAM STUDI SISTEM INFORMASI
DEPARTEMEN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS HASANUDDIN
MAKASSAR

2021

DAFTAR ISI

DAFTAR ISI	ii
DAFTAR TABEL	iii
DAFTAR GAMBAR	v
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan Penelitian	3
1.4 Batasan Masalah	3
1.5 Manfaat Penelitian	3
BAB II TINJAUAN PUSTAKA	4
2.1 Internet of Things (IOT)	4
2.2 Automatic License Plate Recognition (ALPR)	5
2.3 Raspberry Pi	6
2.4 Module Sensor	7
2.4.1 RFID MFRC522	7
2.4.2 HC-SR04	9
2.4.3 SG90	10
2.4.4 Kamera Raspberry Pi v2	11
BAB III METODE PENELITIAN	12
3.1 Waktu dan Lokasi Penelitian	12
3.2 Tahapan Penelitian	12
3.3 Sumber Data	13
3.4 Rancangan Sistem	13
3.5 Rancangan <i>Use Case Diagram</i>	14
3.6 Instrumen Penelitian	14

BAB IV HASIL DAN PEMBAHASAN	16
4.1 Hasil Rancangan Sistem Identifikasi Kendaraan Pada Pemarkiran Dengan Pengenalan Citra Dan Pembacaan RFID	16
4.1.1 Hasil Perancangan Perangkat Keras	16
4.1.2 Raspberry Pi dan RFID MFRC522	21
4.1.3 Raspberry Pi dan HC-SR04	23
4.1.4 Raspberry Pi dan SG90	25
4.1.5 Raspberry Pi dan Kamera Pi	26
4.2 Hasil Rancangan Aplikasi Web	27
4.2.1 <i>Entity relationship Diagram</i>	28
4.2.2 Struktur Database	28
4.2.3 Tampilan Website	30
BAB V KESIMPULAN DAN SARAN	35
5.1 Kesimpulan	35
5.2 Saran	35
DAFTAR PUSTAKA	35
LAMPIRAN	36

DAFTAR TABEL

4.1	Rangkaian pin RFID ke Raspberry Pi	21
4.2	Hasil uji jarak baca RFID	22
4.3	Rangkaian pin Ultrasonik ke Raspberry Pi	23
4.4	Hasil uji jarak baca servo	24
4.5	Rangkaian pin Servo ke Raspberry Pi	26
4.6	rfid_tag	28
4.7	tempat_parkir	29
4.8	kendaraan	29
4.9	parkir	30
4.10	reload_id	30

DAFTAR GAMBAR

2.1	Internet of Things	4
2.2	Raspberry Pi model 3B	6
2.3	RFID Tag	8
2.4	Cara Kerja Sensor Ultrasonik	9
2.5	Sensor Ultrasonik	10
2.6	Servo SG90	11
2.7	Kamera Raspberry Pi	11
3.1	Rancangan Sistem	13
3.2	Use Case Diagram	14
4.1	Rangkaian Skematik Alat	16
4.2	Hasil Rancangan dan Pemasangan Alat	17
4.3	Hasil Rancangan Servo	17
4.4	Hasil Rancangan Ultrasonik dan RFID	18
4.5	Hasil Rancangan Kamera	18
4.6	Activity Diagram Masuk	19
4.7	Activity Diagram Keluar	20
4.8	Rangkaian Raspberry Pi dan RFID	21
4.9	Rangkaian Raspberry Pi dan Ultrasonik	23
4.10	Rangkaian Raspberry Pi dan Servo	25
4.11	Rangkaian Raspberry Pi dan Kamera	26
4.12	Hasil Foto Kamera Pi	26
4.13	ERD	28
4.14	<i>Layout</i> Parkir	31
4.15	<i>Layout</i> Parkir Saat Pengendara Baru Masuk	31
4.16	<i>Layout</i> Parkir Saat Ada Kendaraan	32
4.17	<i>Form</i> Isi Data	32
4.18	<i>Form</i> Isi Saldo	33
4.19	Daftar Pengguna	33

4.20 Informasi	34
--------------------------	----

BAB I

PENDAHULUAN

1.1 Latar Belakang

Revolusi Industri merupakan periode di mana terjadinya perubahan secara besar-besaran di bidang pertanian, manufaktur, tekstil dan logam, pertambangan, transportasi, teknologi, dan sosial ekonomi (Azli Yahya 2017). Pada abad ke-18, mesin uap pertama ditemukan di Inggris. Mesin uap tersebut digunakan sebagai alat tenun mekanis pertama yang dapat meningkatkan produktivitas industri tekstil. Saat itu mesin uap mulai menggantikan peralatan kerja yang awalnya bergantung pada tenaga manusia dan hewan sekaligus memulai era revolusi industri pertama yang dikenal dengan Revolusi Industri 1.0. Revolusi industri kedua ditandai dengan penemuan tenaga listrik pada awal abad ke-20. Revolusi industri ketiga ditandai oleh mesin yang dapat bergerak dan berpikir secara otomatis, yaitu komputer dan robot (Rahayu 2019).

Di abad ke-21 revolusi industry telah masuk ke era baru. Yakni telah berada pada revolusi industri keempat atau lebih dikenal dengan Revolusi Industri 4.0. Era ini telah mengubah banyak bidang kehidupan manusia, termasuk ekonomi, dunia kerja, bahkan gaya hidup. Revolusi industri 4.0 menawarkan teknologi cerdas yang dapat terhubung dengan berbagai bidang kehidupan manusia. Revolusi industri 4.0 menerapkan *Internet of Things (IoT)* dan teknologi pada kegiatan analisis, manufaktur, robotik, komputasi canggih, *artificial intelligence*, teknologi, kognitif, *advance materials* dan *augmented reality* dalam melaksanakan siklus operasi bisnis (Suharman dkk. 2019). Saat ini negara-negara di dunia mulai berkופetisi dalam pemanfaatan teknologi pada setiap sektor industrinya. Lalu bagaimana dengan indonesia ? Mempelajari konsep industri 4.0 untuk penerapannya di Indonesia menjadi suatu keharusan, sebab jika tidak maka industry dan manufaktur di Indonesia tidak akan dapat bersaing dengan industry dan manufaktur di negara-negara lain di dunia.

Revolusi industri 4.0 mencakup beragam teknologi canggih, seperti kecerdasan buatan (AI), *wearables*, robotika canggih, *3D printing*, dan *Internet of Things (IoT)*. *Internet of Things (IoT)* adalah sekenario dari suatu objek yang dapat melakukan

pengiriman data/informasi melalui jaringan tanpa campur tangan manusia (Limantara dkk. 2017). *Konsep Internet of Things* sudah banyak digunakan dalam kehidupan sehari-hari di berbagai bidang, seperti bidang pertanian, bidang kesehatan, bidang industri, bidang keamanan, serta bidang transportasi.

Salah satu permasalahan yang banyak dijumpai diera sekarang adalah sistem parkir yang masih menggunakan metode konvensional untuk mencatat nomor pelat kendaraan yang akan parkir dan pembayaran yang masih menggunakan uang cash. Hal ini dapat memicu kemacetan, polusi udara dan suara, dan menambah tingkat stress pengendara. Untuk mengatasi masalah tersebut, dibuatlah suatu sistem dengan memanfaatkana konsep *Internet of Things* di bidang transportasi dan keamanan. Sistem yang dimaksud adalah sistem parkir otomatis yang diletakkan di tempat khusus seperti di apartemen atau di perkantoran.

Salah satu aspek dalam sistem parkir otomatis adalah identifikasi citra pelat kendaraan untuk mendapatkan data nomor pelat tanpa campur tangan manusia. Identifikasi pelat kendaraan pada sistem parkir otomatis dapat dilakukan dengan kartu RFID dan pengolahan citra pelat kendaraan. Kemampuan RFID sebagai media pengenal secara nirkabel membuat RFID sering digunakan sebagai otorisasi untuk akses ruangan dan tempat, akan tetapi penggunaan RFID masih rentan terhadap keamanan akses. Penelitian ini menggabungkan identifikasi kartu RFID dan pembacaan citra pelat kendaraan sehingga dapat meningkatkan keamanan parkir dan dapat mempersingkat waktu pencatatan nomor pelat kendaraan yang masih bersifat konvensional.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang masalah diatas, dapat dikemukakan pertanyaan penelitian sebagai berikut:

1. Bagaimana cara merancang dan membangun sistem identifikasi kendaraan dengan pengenalan citra pelat dan pembacaan RFID ?
2. Bagaimana cara membuat aplikasi web sebagai *user interface* sistem parkir ?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah, maka tujuan penelitian ini adalah :

1. Merancang sistem identifikasi kendaraan dengan pengenalan citra pelat dan pembacaan RFID.
2. Membuat aplikasi web sebagai *user interface*.

1.4 Batasan Masalah

Batasan masalah pada penelitian ini adalah :

1. Alat yang dibuat bersifat prototype.
2. Sistem parkir yang dibuat ditujukan untuk diterapkan di lingkungan berpenghuni tetap seperti apartemen atau perkantoran.
3. Karakter pada pelat nomor harus sesuai dengan yang digunakan Samsat.
4. Tidak melakukan analisis lebih lanjut pada deteksi nomor pelat kendaraan.

1.5 Manfaat Penelitian

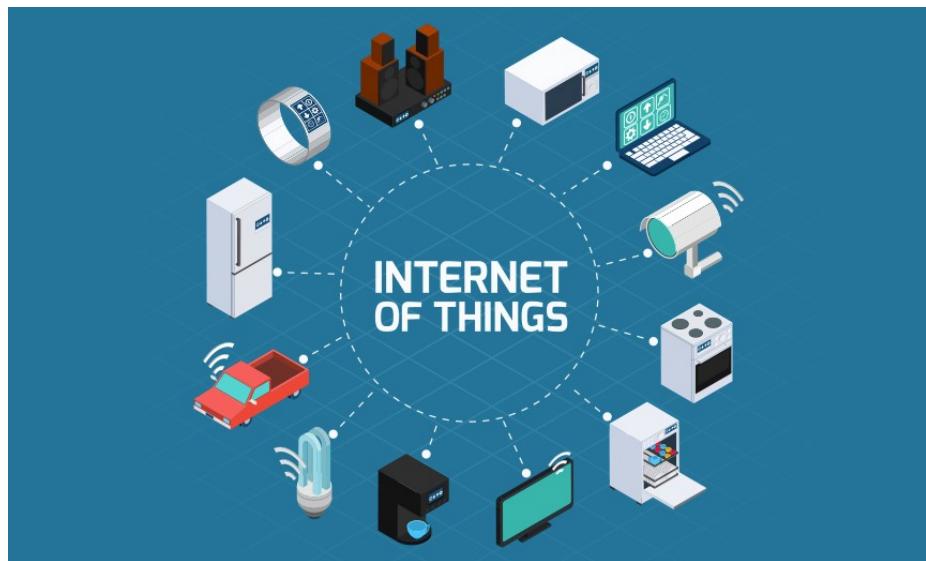
Hasil penelitian ini diharapkan dapat bermanfaat :

1. Menghemat waktu dan bahan bakar.
2. Terciptanya alat yang dapat menopang kemajuan industri.
3. Mengurangi antrian Panjang yang disebabkan oleh pencatatan nomor pelat dan pembayaran parkir yang masih konvensional.
4. Mempermudah pengelolah parkir untuk memantau sisa kapasitas lahan parkir yang tersedia.
5. Membantu upaya pemerintah dalam pembangunan *smart city* di Indonesia.

BAB II

TINJAUAN PUSTAKA

2.1 Internet of Things (IOT)



Gambar 2.1: Internet of Things

Internet of Things adalah skenario dari suatu objek yang dapat melakukan suatu pengiriman data/informasi melalui jaringan tanpa campur tangan manusia. *IoT* sangat erat hubungannya dengan komunikasi mesin ke mesin (M2M) tanpa campur tangan manusia ataupun komputer yang lebih dikenal dengan istilah cerdas (*smart*) (Limantara dkk. 2017).

Cara Kerja *Internet of Things* yaitu dengan memanfaatkan sebuah argumentasi pemrograman yang dimana tiap-tiap perintah argumennya itu menghasilkan sebuah interaksi antara sesama mesin yang terhubung secara otomatis tanpa campur tangan manusia dan dalam jarak berapa pun. Internetlah yang menjadi penghubung di antara kedua interaksi mesin tersebut, sementara manusia hanya bertugas sebagai pengatur dan pengawas bekerjanya alat tersebut secara langsung. Tantangan terbesar dalam mengkonfigurasi *Internet of Things* ialah menyusun jaringan komunikasinya sendiri, yang dimana jaringan tersebut sangatlah kompleks, dan memerlukan sistem keamanan yang ketat. Selain itu biaya yang mahal sering menjadi penyebab kegagalan yang berujung pada gagalnya produksi.

Metode yang digunakan oleh *Internet of Things* adalah nirkabel atau pengendalian secara otomatis tanpa mengenal jarak. Pengimplementasian *Internet of Things* sendiri biasanya selalu mengikuti keinginan si developer dalam mengembangkan sebuah aplikasi yang ia ciptakan, apabila aplikasinya itu diciptakan guna membantu monitoring sebuah ruangan maka pengimplementasian *Internet of Things* itu sendiri harus mengikuti alur diagram pemrograman mengenai sensor dalam sebuah rumah, berapa jauh jarak agar ruangan dapat dikontrol, dan kecepatan jaringan internet yang digunakan

Banyak manfaat yang didapatkan dari *Internet of Things*. Pekerjaan yang kita lakukan menjadi cepat, mudah, dan efisien. Kemunculan *Internet Of Things (IoT)* memungkinkan perangkat komputer secara otomatis dapat melakukan kontrol terhadap suatu sistem dan memungkinkan pula untuk memberi aksi ke sistem terhadap kejadian yang terjadi pada sistem yang dikontrol secara realtime (Ichwana dkk. 2018).

2.2 Automatic License Plate Recognition (ALPR)

Automatic License Plate Recognition adalah teknologi yang menggunakan pengenalan karakter pada gambar untuk membaca plat registrasi kendaraan. ALPR digunakan oleh polisi di beberapa negara di dunia untuk tujuan penegakan hukum, termasuk untuk memeriksa apakah kendaraan terdaftar atau tidak.

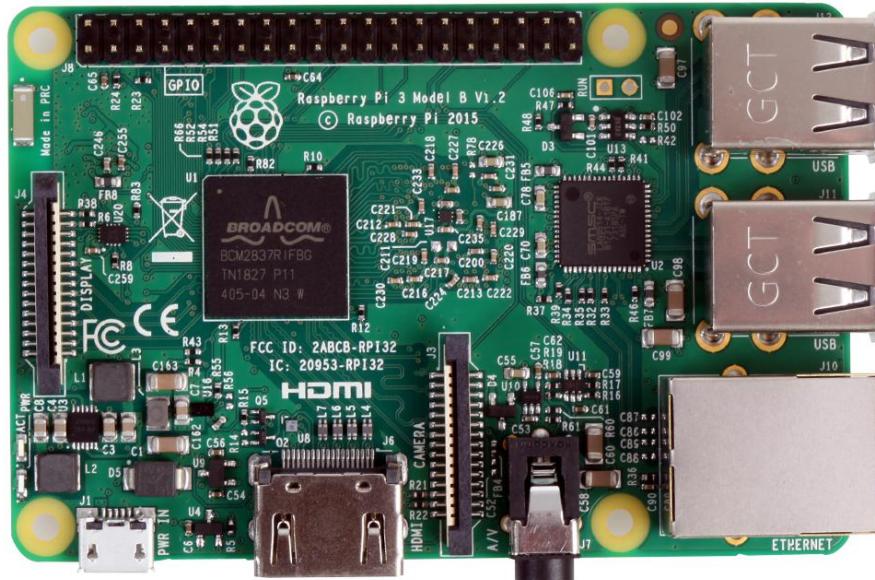
Pengenalan plat nomor otomatis dapat digunakan untuk menyimpan gambar yang diambil oleh kamera serta teks dari plat nomor. Umumnya sistem menggunakan pencahaayaan inframerah untuk memungkinkan kamera mengambil gambar kapan saja, siang atau malam hari. Selain itu, teknologi ALPR juga harus memperhitungkan variasi nomor plat dari suatu negara karena bentuk dan ukuran nomor plat di satu negara dengan negara lainnya kemungkinan sangat berbeda.

ALPR menjadi tren baru dalam otomatisasi sistem transportasi. Pencatatan pelat nomor kendaraan bisa dilakukan tanpa campur tangan manusia. Meskipun teknologi tersebut telah ditetapkan di negara-negara maju, negara-negara berkembang seperti Indonesia belum menerapkan teknologi tersebut karena berbagai alasan (Budianto 2018).

2.3 Raspberry Pi

Raspberry Pi adalah komputer mini yang dirancang dan diproduksi di Inggris dengan tujuan awal untuk menyediakan perangkat komputasi yang murah untuk pendidikan. Raspberry Pi ditemukan pertama kali di University of Cambridge laboratory pada tahun 2006. Raspberry Pi dirilis secara komersial pada februari 2012. Sejak saat itu *board* Raspberry Pi telah melalui sejumlah revisi dan tersedia dalam 2 model yaitu model A dan model B (Wicaksono 2018).

Secara kasar ditengah semua model Raspberry Pi terdapat sebuah semikonduktor persegi atau yang dikenal sebagai *integrated circuit* atau *chip*. *Integrated Circuit* adalah *sistem-on-chip* modul yang menyediakan kemampuan untuk pemrosesan umum (*general purpose*), render grafis, dan *input/output* (Wicaksono 2018).



Gambar 2.2: *Raspberry Pi model 3B*

Raspberry pi 3 adalah model terbaru Raspberry Pi. Raspberry Pi 3 menggunakan *processor* terbaru yaitu Broadcom BCM283764 bit. BCM283764 lebih cepat dari pada BCM2836. Raspberry Pi 3 juga merupakan model pertama yang memiliki *built-in wireless* (mampu terhubung ke jaringan WIFI dan juga memiliki perangkat *Bluetooth*). Raspberry Pi model 3B bisa dilihat pada gambar 2.2. Berikut

merupakan spesifikasi dari Raspberry Pi 3 :

- SoC: Broadcom BCM2837
- CPU: 4x ARM Cortex-A53, 1.2GHz
- GPU: Broadcom VideoCore IV
- RAM: 1GB LPDDR2 (900 MHz)
- Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
- Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
- Storage: microSD
- GPIO: 40-pin header
- Ports: HDMI, 3.5mm analogue audio-video jack, 4x USB 2.0, Ethernet, CameraSerial Interface (CSI), Display Serial Interface (DSI)

2.4 Module Sensor

Sensor adalah sesuatu yang digunakan untuk mendeteksi adanya perubahan lingkungan fisik atau kimia.

2.4.1 RFID MFRC522

Radio Frequency Identification (RFID) adalah teknologi untuk mengidentifikasi dan mengendalikan data dari jarak jauh menggunakan transmisi gelombang radio. RFID menggunakan sarana transponder atau RFID tag untuk menyimpan dan mengambil data dari jarak jauh. RFID tag mirip dengan penggunaan barcode yang melekat pada sebuah objek yang menyimpan identifikasi data obyek (Singgeta dkk. 2018).

RFID mempunyai 2 bagian komponen utama yang tak dapat dipisahkan, yaitu:

1. RFID Tag

Merupakan sebuah perangkat yang akan diidentifikasi oleh RFID *reader* yang dapat berupa perangkat pasif maupun aktif yang berisi suatu data atau informasi. Tag RFID, dapat berupa stiker, kertas atau plastik dengan beragam ukuran . Di dalam setiap tag ini terdapat chip yang mampu menyimpan sejumlah informasi tertentu. RFID Tag berfungsi sebagai transponder (transmitter dan responder) yang berisikan data dengan menggunakan frekuensi 125 KHz. RFID tag bisa

dilihat pada gambar 2.3



Gambar 2.3: *RFID Tag*

Pada RFID tag terdapat 2 jenis yaitu *Read-Write* dan *Only Read*. Selain itu RFID tag mempunyai 2 komponen utama yang penting, antara lain:

- IC (*Integrated Circuit*) : berfungsi sebagai pemproses informasi, modulasi serta demodulasi sinyal RF, yang beroperasi dengan catudaya DC.
- ANTENNA : mempunyai fungsi untuk mengirim maupun menerima sinyal RF.

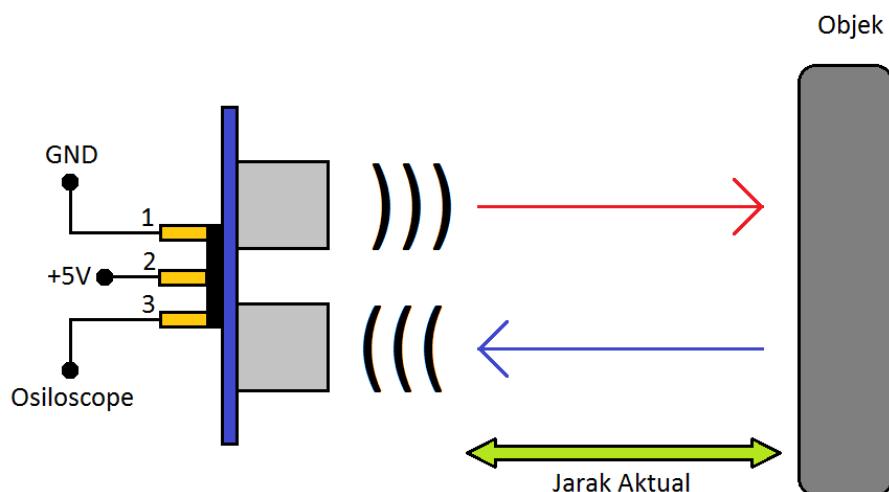
2. RFID *Reader*

Berfungsi untuk membaca data dari RFID Tag. RFID *Reader* dibedakan menjadi 2 macam, antara lain :

- Pasif : hanya bisa membaca data dari RFID tag aktif
- Aktif : dapat membaca data RFID tag pasif

2.4.2 HC-SR04

Sensor ultrasonik adalah sebuah sensor yang mengubah besaran fisis berupa bunyi menjadi besaran listrik dan sebaliknya. Cara kerja sensor ini didasarkan pada prinsip dari pantulan suatu gelombang suara sehingga dapat dipakai untuk menafsirkan jarak suatu benda dengan frekuensi tertentu. Disebut sebagai sensor ultrasonik karena sensor ini menggunakan gelombang ultrasonik (bunyi ultrasonik). Bunyi ultrasonik bisa merambat melalui zat padat, cair dan gas. Reflektivitas bunyi ultrasonik di permukaan zat padat hampir sama dengan reflektivitas bunyi ultrasonik di permukaan zat cair. Akan tetapi, gelombang bunyi ultrasonik akan diserap oleh tekstil dan busa.



Gambar 2.4: Cara Kerja Sensor Ultrasonik

Secara umum, alat ini akan menembakkan gelombang ultrasonik menuju suatu area atau suatu target. Setelah gelombang menyentuh permukaan target, maka target akan memantulkan kembali gelombang tersebut. Gelombang pantulan dari target akan ditangkap oleh sensor, kemudian sensor menghitung selisih antara waktu pengiriman gelombang dan waktu gelombang pantul diterima (Limantara dkk. 2017). Ilustrasinya bisa dilihat pada gambar 2.4.



Gambar 2.5: Sensor Ultrasonik

HC-SR04 merupakan sensor ultrasonik yang berfungsi sebagai pengirim, penerima, dan pengontrol gelombang ultrasonik. Alat ini bisa digunakan untuk mengukur jarak benda dari 2cm-4m dengan akurasi 3mm. Alat ini memiliki 4 pin, pin Vcc, Gnd, Trigger, dan Echo. Pin Vcc untuk listrik positif dan Gnd untuk ground-nya. Pin Trigger untuk trigger keluarnya sinyal dari sensor dan pin Echo untuk menangkap sinyal pantul dari benda. Sensor ultrasonik HC-SR04 bisa dilihat pada gambar 2.5.

2.4.3 SG90

SG90 adalah sebuah servo kecil dengan output power yang tinggi. Motor ini dapat berotasi sekitar 180 derajat dan bisa bekerja seperti servo lainnya hanya saja ukurannya lebih kecil (Wicaksono 2018).

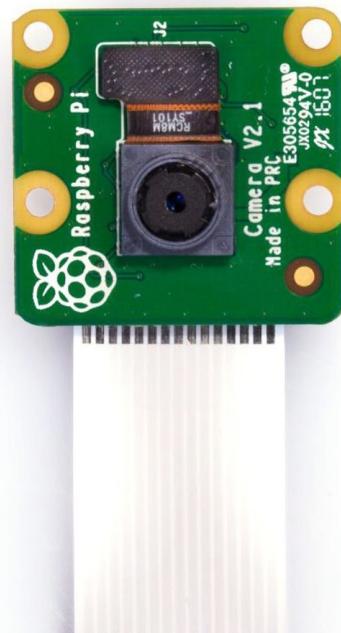


Gambar 2.6: Servo SG90

Gambar 2.6 merupakan gambar dari servo SG90.

2.4.4 Kamera Raspberry Pi v2

Modul Kamera v2 memiliki sensor Sony IMX219 8-megapiksel. Modul Kamera dapat digunakan untuk mengambil video definisi tinggi, dan juga foto. Gambar 2.7 merupakan gambar dari kamera Raspberry Pi.



Gambar 2.7: Kamera Raspberry Pi

BAB III

METODE PENELITIAN

3.1 Waktu dan Lokasi Penelitian

Penelitian ini dilaksanakan dari bulan juli 2020 sampai dengan bulan november 2020. Lokasi penelitian dilakukan di Laboratorium Rekayasa Perangkat Lunak Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Hasanuddin.

3.2 Tahapan Penelitian

Analisis Kebutuhan

Pada tahapan ini merupakan tahap awal yaitu menganalisis semua kebutuhan yang diperlukan selama meneliti seperti pengumpulan teori terkait dengan penelitian yang terdapat dalam buku atau jurnal.



Desain Sistem

Pada tahap ini dilakukan perancangan sistem yang akan dibangun seperti, rancangan sistem, rancangan skematik alat, *use case* diagram, *activity* diagram dan perancangan aplikasi web.



Implementasi

Pada tahapan ini dilakukan implementasi dari hasil perancangan pada tahapan sebelumnya. Di tahapan inilah pembangunan sistem parkir, perangkaian sensor-sensor dan *microcontroller* dilakukan.



Pengujian Sistem

Pada tahapan ini hasil dari pembuatan sistem siap untuk diimplementasikan. Sensor ditempatkan di tempat yang sudah disediakan, maka alat ini akan mengumpulkan data yang dibutuhkan seperti nomor plat, id rfid, dan jarak ultrasonik. Data yang berhasil dikumpulkan akan disimpan di database dan bisa dilihat di web.



Kesimpulan

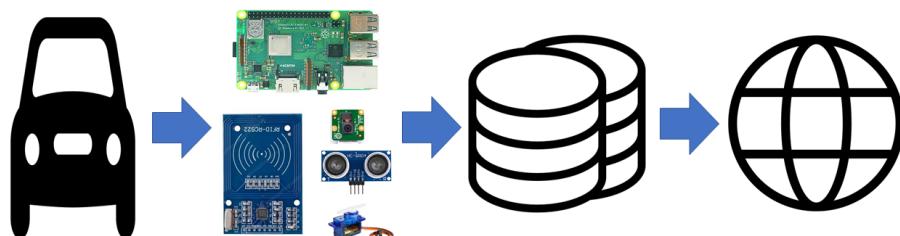
Pada tahap ini merupakan tahap penting terakhir dari penelitian ini. Pada tahapan ini hasil dari pembuatan sistem parkir sudah siap untuk diimplementasikan.

3.3 Sumber Data

Sumber data dari penelitian ini menggunakan data primer yang didapatkan secara langsung dari sensor yang digunakan dalam sistem parkir.

3.4 Rancangan Sistem

Sensor dan kamera akan terhubung dengan raspberry pi. Berikut ini adalah gambaran mengenai rancangan dari penelitian ini :



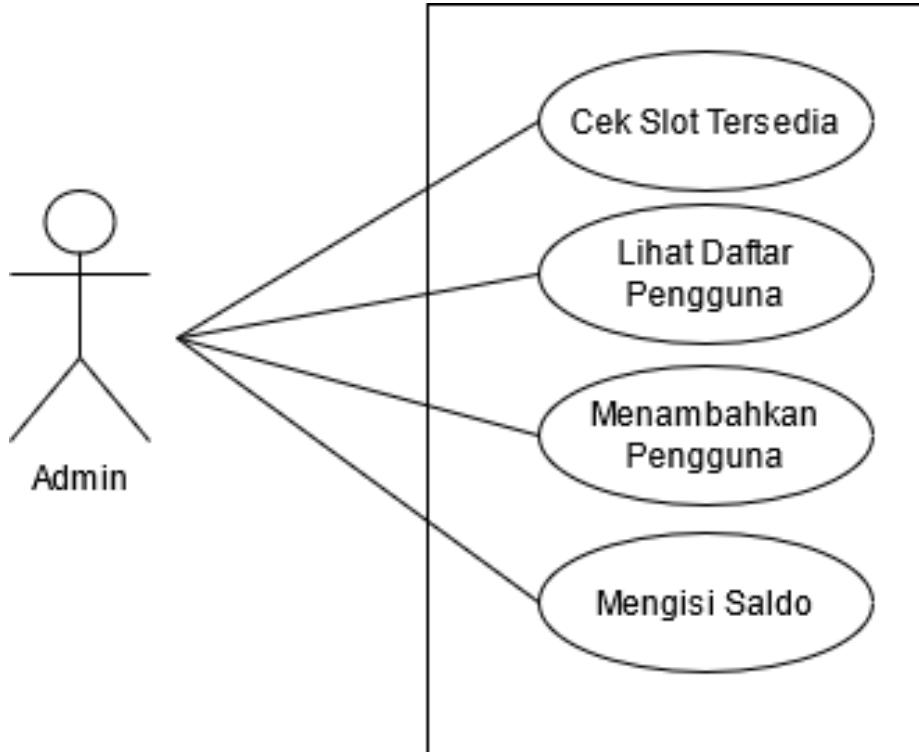
Gambar 3.1: Rancangan Sistem

Pada gambar 3.1 saat pengemudi kendaraan menempelkan tag RFID mereka ke RFID reader, kamera pada raspberry akan mengambil gambar untuk diidentifikasi nomor pelat tersebut. Hasil identifikasi akan berupa data nomor pelat kendaraan yang akan disimpan. Setelah itu servo yang berfungsi sebagai palang pintu akan terbuka.

Data yang dikumpulkan oleh sensor berupa Id RFID dan nomor pelat kendaraan dapat dilihat melalui aplikasi web.

3.5 Rancangan *Use Case Diagram*

Rancangan *use case* diagram pada penelitian ini dapat di gambarkan seperti diagram dibawah ini:



Gambar 3.2: *Use Case Diagram*

sesuai dengan gambar 3.2, sistem yang akan dibangun hanya terdapat satu jenis akun yaitu akun admin yang akan mengawasi seluruh aktivitas pemarkiran melalui aplikasi web sebagai *user interface*.

3.6 Instrumen Penelitian

Instrumen penelitian pada penelitian ini meliputi kebutuhan perangkat keras dan kebutuhan perangkat lunak.

1. Kebutuhan perangkat keras
 - Raspberry Pi
 - Sensor RFID MFRC522

- Sensor Ultrasonik HC-SR04
- Servo SG90
- Camera Raspberry Pi v2 8mp
- Kabel Jumper
- Bread Board
- Memori Card

2. Kebutuhan perangkat lunak

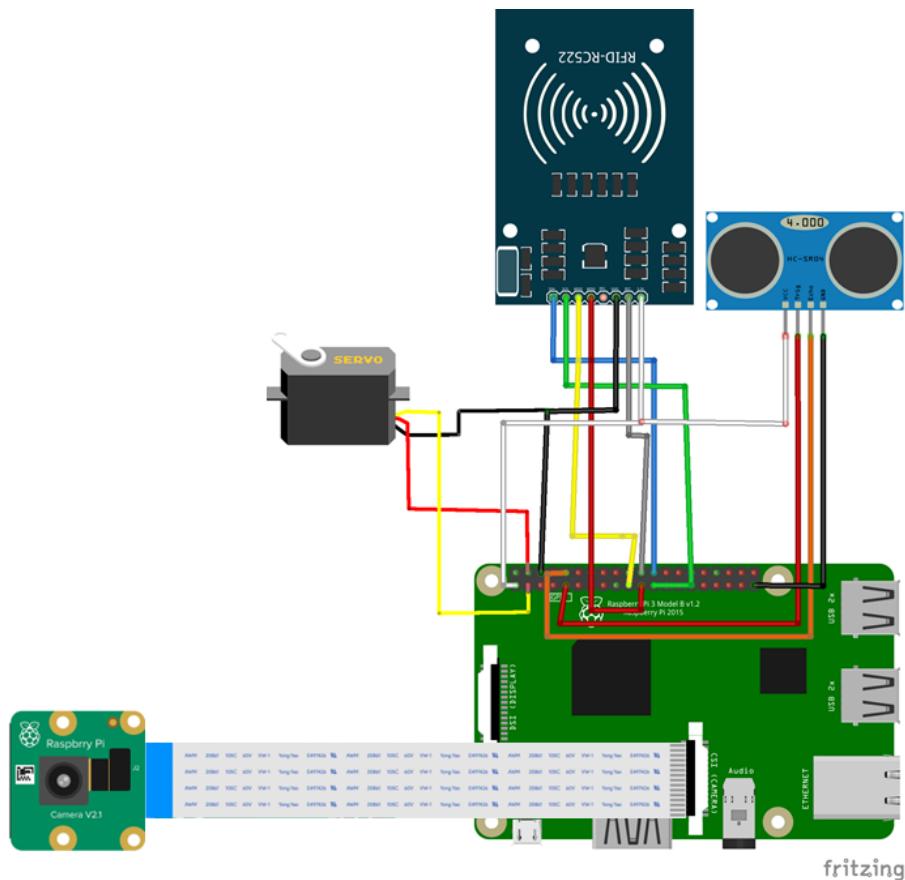
- *Operating System (OS)* Raspbian
- *Python Programming Language* (bahasa pemrograman yang digunakan)
- Visual Studio Code
- Web Browser

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil Rancangan Sistem Identifikasi Kendaraan Pada Pemarkiran Dengan Pengenalan Citra Dan Pembacaan RFID

4.1.1 Hasil Perancangan Perangkat Keras



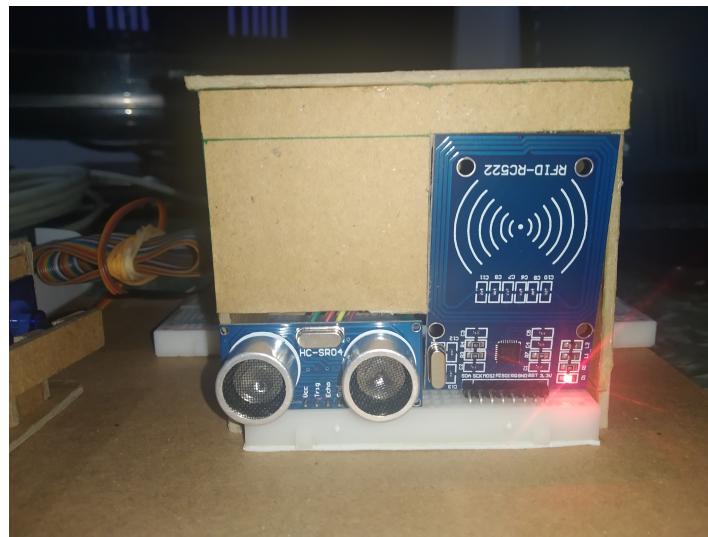
Gambar 4.1: Rangkaian Skematik Alat



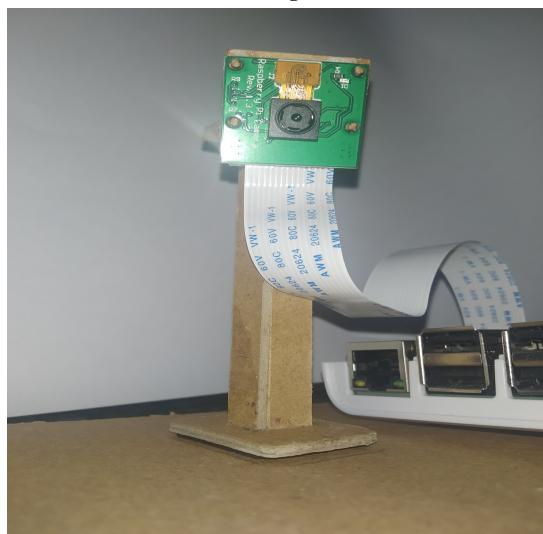
Gambar 4.2: Hasil Rancangan dan Pemasangan Alat



Gambar 4.3: Hasil Rancangan Servo

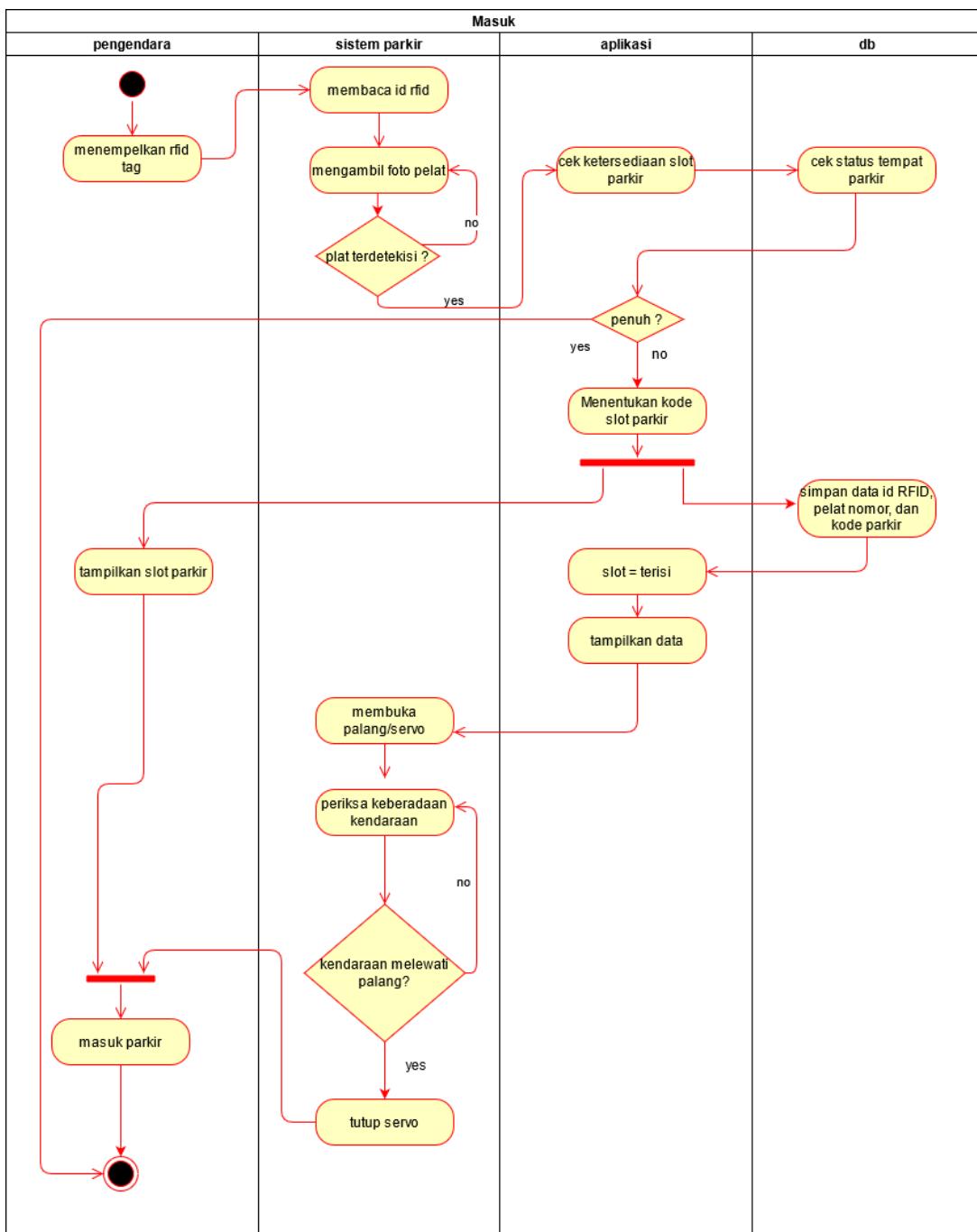


Gambar 4.4: Hasil Rancangan Ultrasonik dan RFID

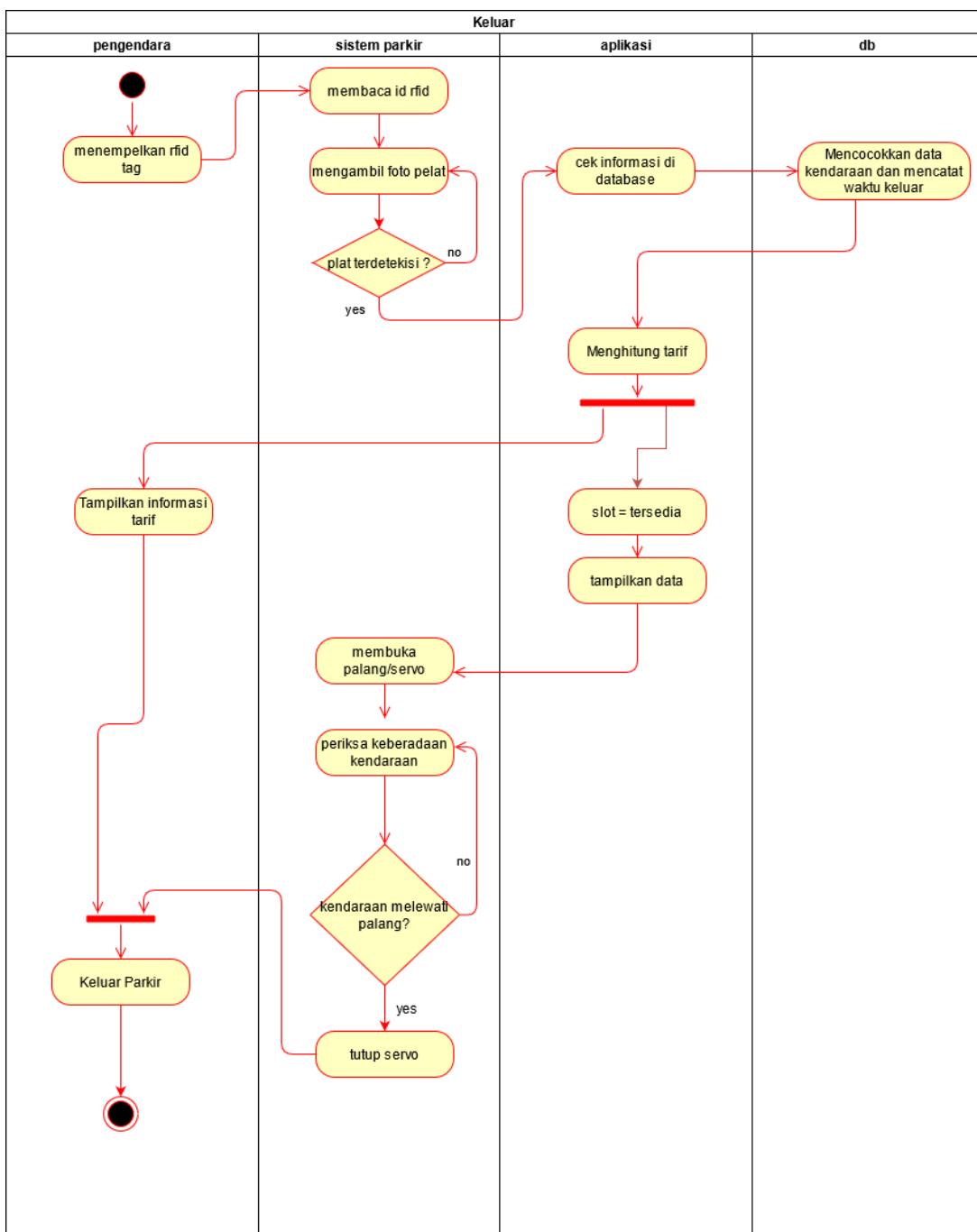


Gambar 4.5: Hasil Rancangan Kamera

Pada gambar diatas merupakan rangkaian perangkat keras untuk penelitian di mana seluruh perangkat dirangkai menjadi satu rangkaian. Gambar 4.2 merupakan hasil dari rancangan dari penelitian yang dilakukan, yang dimana masih bersifat *prototype*. Pada gambar 4.3 dapat dilihat bahwa servo digambarkan sebagai palang parkir. Pada gambar 4.4 dapat dilihat sensor ultrasonik yang akan membaca jarak dari kendaraan untuk menentukan apakah kendaraan masih ada atau sudah tidak ada dan sensor rfid untuk membaca kartu atau tag dari pengendara. Pada gambar 4.5 dapat dilihat kamera raspberry yang digunakan untuk mengambil nomor plat kendaraan. *Activity diagram* pada sistem yang dibuat bisa dilihat pada gambar 4.6 dan gambar 4.7.

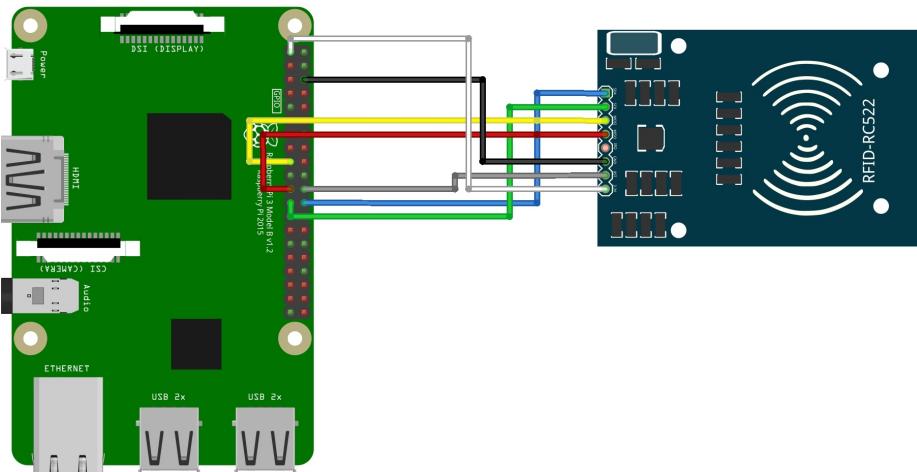


Gambar 4.6: Activity Diagram Masuk



Gambar 4.7: Activity Diagram Keluar

4.1.2 Raspberry Pi dan RFID MFRC522



Gambar 4.8: Ragkaian Raspberry Pi dan RFID

Berdasarkan gambar 4.8 Menunjukkan Raspberry Pi sebagai mikrokontroler untuk menghubungkan sensor RFID yang akan membaca kartu atau tag dari pengendara. Untuk pin pada Raspberry Pi dihubungkan pada sensor RFID dapat dilihat pada tabel 4.1.

Tabel 4.1: Rangkaian pin RFID ke Raspberry Pi

RFID	RASPBERRY PI
SDA	Pin 24
SCK	Pin 23
MOSI	Pin 19
MISO	Pin 21
GND(-)	Pin 6
RST	Pin 22
3.3v(+)	Pin 1

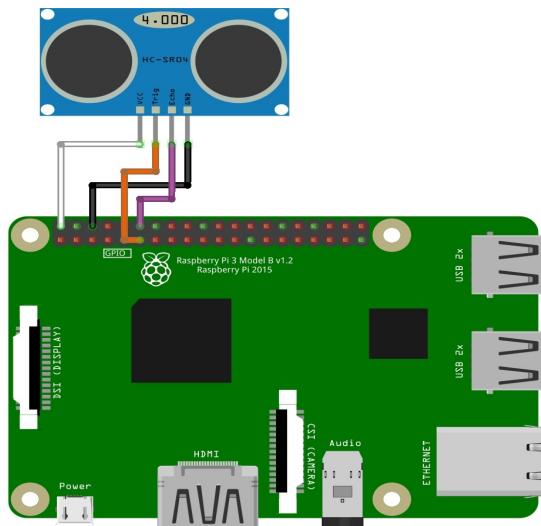
Berikut ini dilakukan pengujian jarak baca kartu RFID yang berfungsi untuk melihat jarak maksimum keterbacaan RFID dari bagian pembaca ke kartu RFID. Pengujian ini dilakukan dengan cara meletakkan kartu RFID dengan memberi jarak tertentu pada area pembacaan. Hasil pengujian dapat dilihat pada tabel 4.2.

Tabel 4.2: Hasil uji jarak baca RFID

No.	Jarak (mm)	Terbaca
1	0	Ya
2	2	Ya
3	4	Ya
4	6	Ya
5	8	Ya
6	10	Ya
7	12	Ya
8	14	Ya
9	16	Ya
10	18	Ya
11	20	Ya
12	22	Ya
13	24	Ya
14	26	Ya
15	28	Ya
16	30	Tidak
17	32	Tidak
18	34	Tidak
19	36	Tidak
20	38	Tidak

Berdasarkan data hasil pengujian sensor RFID pada tabel 4.2, pengambilan data dilakukan sebanyak dua puluh kali. Dari data hasil pengukuran yang pertama sampai ke lima belas dengan jarak 0 mm sampai 28 mm, kartu RFID dapat terbaca dengan baik dan pengukuran ke enam belas sampai dua puluh dengan jarak 30 mm sampai 38 mm, kartu sudah tidak bisa terbaca.

4.1.3 Raspberry Pi dan HC-SR04



Gambar 4.9: Ragkaian Raspberry Pi dan Ultrasonik

Gambar 4.9 merupakan gambar rangkaian Raspberry Pi dan sensor ultrasonik. Sensor ultrasonik berfungsi sebagai indikator untuk menutup palang parkir. Apabila didepan sensor ultrasonik masih ada kendaraan, maka palang parkir masih akan terbuka, sebaliknya apabila didepan sensor sudah tidak ada kendaraan, maka palang parkir akan menutup. Sensor ultrasonik yang digunakan adalah HC-SR04 yang mempunyai 4 pin yaitu pin *ground* (-), pin *echo*, pin *trigger*, dan pin *vcc* (+). Untuk pin pada Raspberry Pi dihubungkan pada sensor HC-SR04 dapat dilihat pada tabel 4.3.

Tabel 4.3: Rangkaian pin Ultrasonik ke Raspberry Pi

ULTRASONIK	RASPBERRY PI
GND(-)	Pin 6
TRIG	Pin 11
ECHO	Pin 12
VCC(+)	Pin 4

Berikut ini pengujian yang dilakukan pada sensor ultrasonik untuk mengukur tingkat kesalahan jarak yang diukur oleh sensor ultrasonik dibandingkan jarak sebenarnya dengan pengukuran secara manual:

Berdasarkan data hasil pengujian sensor ultrasonik pada tabel 4.4,

Tabel 4.4: Hasil uji jarak baca servo

Pengukuran Manual(cm)	Pengukuran Sensor(cm)			Selisih Pengukuran(cm)			Percentase Kesalahan(%)		
	Pengukuran ke-			Pengukuran ke-			Pengukuran ke-		
	1	2	3	1	2	3	1	2	3
2	2	2	2	0	0	0	0	0	0
6	6	6	6	0	0	0	0	0	0
15	15	15	15	0	0	0	0	0	0
20	20	19	19	0	1	1	0	5	5
30	29	29	29	1	1	1	3	3	3
40	38	39	38	2	1	2	5	3	5
50	49	49	49	1	1	1	2	2	2
60	57	56	57	3	4	3	5	7	5
70	68	69	68	2	1	2	3	1	3
80	78	78	78	2	2	2	3	3	3
90	89	90	88	1	0	2	1	0	2
100	96	97	96	4	3	4	4	3	4
110	107	107	108	3	3	2	3	3	2
120	115	114	115	5	6	5	4	5	4
Rata-rata kesalahan							2,34	2,44	2,71

pengambilan data dilakukan sebanyak empat belas kali dengan tiga kali pengukuran, didapatkan hasil yang berbeda antara pengukuran manual yang menggunakan penggaris dengan pengukuran oleh sensor. Dari data hasil pengukuran yang pertama didapat bahwa terdapat selisih jarak hasil pengujian oleh sensor dengan jarak sebenarnya, dilakukan pengukuran dengan rentang jarak dari 2 cm sampai 120 cm yang dimana untuk pengukuran 2 cm sampai 15 cm tidak ada selisih antara pengukuran manual dan pengukuran sensor. Pada jarak 20 cm tidak ada selisih untuk pengukuran pertama, sedangkan pengukuran kedua dan ketiga terdapat selisih sebesar 1 cm. Pada jarak 30 cm terdapat selisih 1 cm untuk semua pengukuran. Pada jarak 40 cm selisih pada pengukuran pertama sebesar 2 cm, kedua sebesar 1 cm, dan ketiga sebesar 2 cm. Pada jarak 50 cm dan 60 cm terdapat selisih sebesar 2 cm pada setiap pengukuran. Pada jarak 70 cm selisih pada pengukuran pertama sebesar 2 cm, kedua sebesar 1 cm, dan ketiga sebesar 2 cm. Pada jarak 80 cm dan 90 cm terdapat selisih sebesar 3 cm pada setiap pengukuran. Pada jarak 100 cm dan 110 cm terdapat selisih sebesar 4 cm

pada setiap pengukuran. Pada jarak 120 cm selisih pada pengukuran pertama sebesar 5 cm, kedua sebesar 6 cm, dan ketiga sebesar 5 cm.

Berdasarkan data hasil pengujian pada tabel menunjukkan bahwa semakin jauh jarak yang diukur, semakin besar kesalahan jarak yang diukur oleh sensor dari jarak sebenarnya. Perbedaan jarak hasil pengujian sensor dengan jarak sesungguhnya dapat disebabkan oleh adanya *noise* atau permukaan jarak yang diukur tidak rata.

Berikut ini rumus serta perhitungan yang digunakan untuk menghitung persentase kesalahan dan rata-rata kesalahan, dari perbandingan pengukuran jarak antara sensor dengan manual:

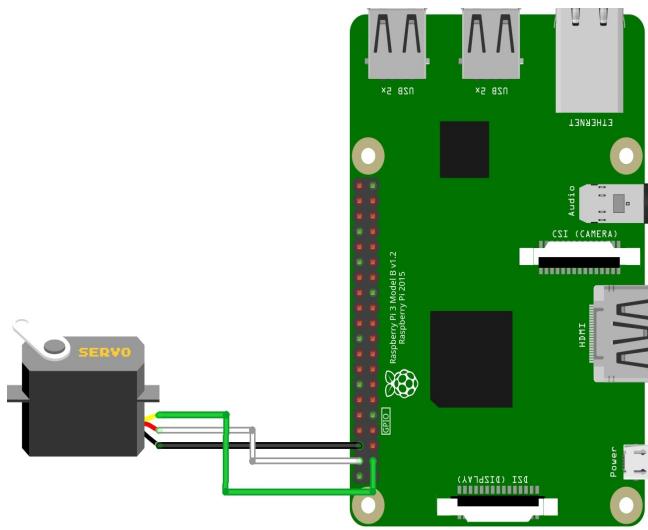
1. Persentase kesalahan

$$\% \text{kesalahan} = \frac{(jarak\text{manual} - jarak\text{sensor})}{jarak\text{manual}} \times 100\% \quad (4.1)$$

2. Rata-rata kesalahan

$$\% \text{rata-rata kesalahan} = \frac{\text{jarak}\% \text{kesalahan}}{\text{banyaknya data}} \quad (4.2)$$

4.1.4 Raspberry Pi dan SG90



Gambar 4.10: Ragkaian Raspberry Pi dan Servo

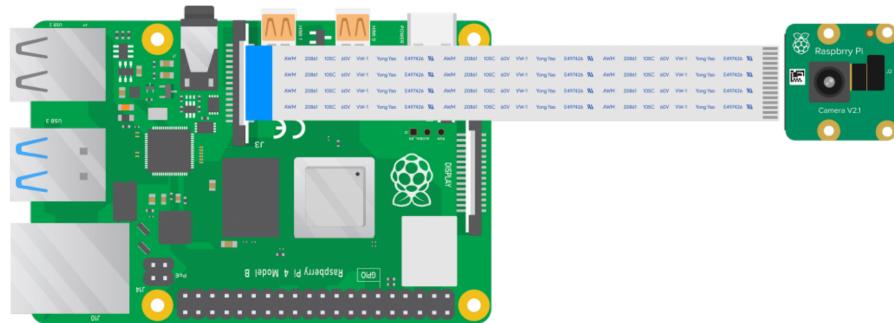
Pada penelitian ini yang akan berfungsi sebagai palang parkir adalah servo SG90. Servo yang digunakan mempunyai 3 pin yaitu pin *ground* (-), pin *pwm*, dan pin *5v* (+). Untuk pin pada Raspberry Pi dihubungkan pada servo SG90 dapat dilihat

pada tabel 4.5.

Tabel 4.5: Rangkaian pin Servo ke Raspberry Pi

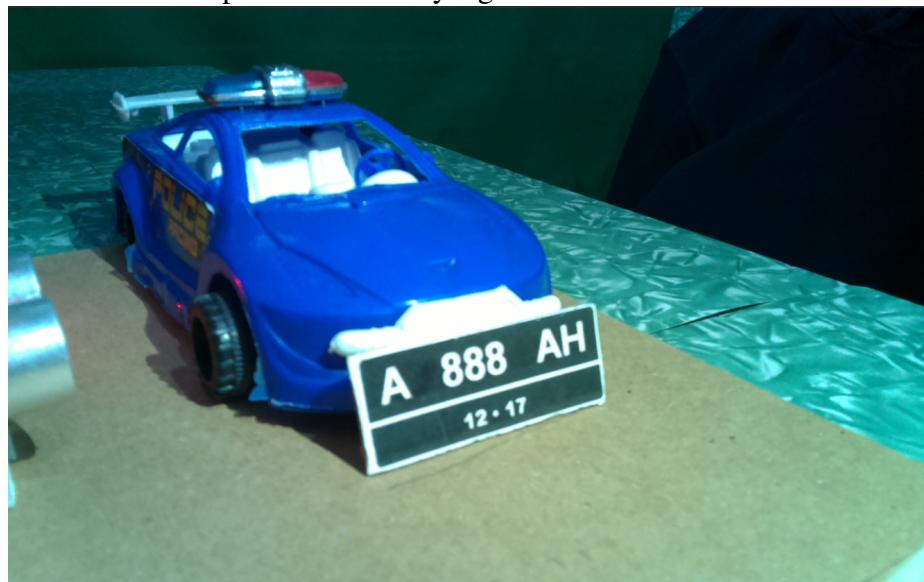
SERVO	RASPBERRY PI
PWM	Pin 3
5v(+)	Pin 4
GND(-)	Pin 6

4.1.5 Raspberry Pi dan Kamera Pi



Gambar 4.11: Ragkaian Raspberry Pi dan Kamera

Gambar 4.11 merupakan gambar skematik Raspberry Pi dengan kamera. Untuk menghubungkan Raspberry Pi dengan kamera cukup dengan menghubungkan kamera dengan port *Camera Serial Interface* (CSI) yang sudah tersedia pada Raspberry Pi. Gambar 4.12 merupakan hasil foto yang diambil oleh kamera Pi.



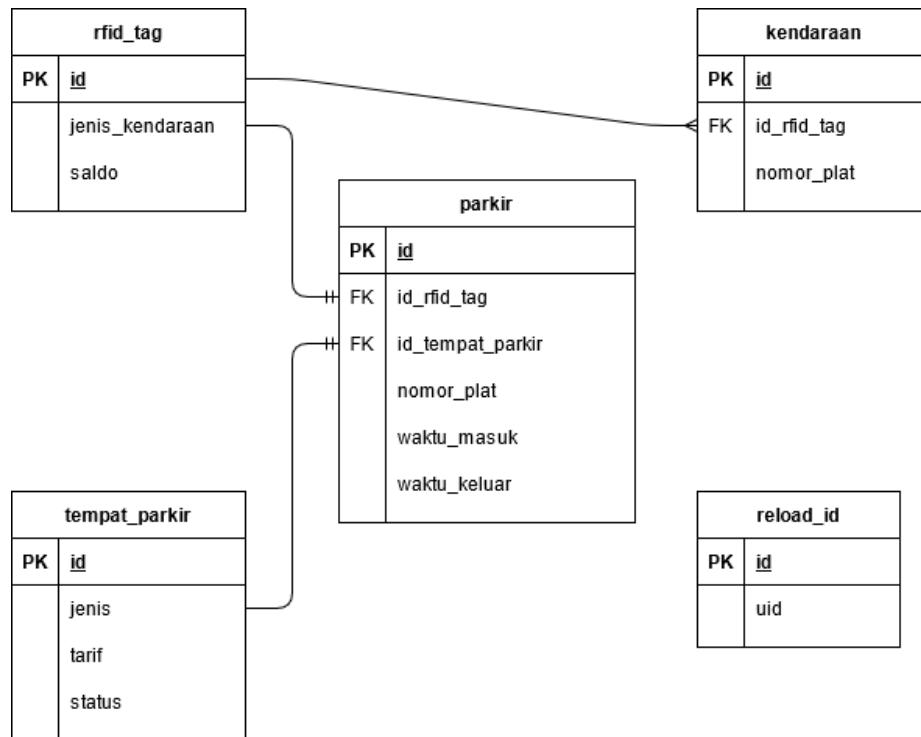
Gambar 4.12: Hasil Foto Kamera Pi

Foto yang diambil oleh kamera akan di proses untuk mengambil nomor plat kendaraan. Pengambilan nomor plat kendaraan menggunakan API((Application Programming Interface)) yang disediakan oleh (platerecognizer.com). Foto yang telah diambil oleh kamera akan dikirim dengan cara mengakses API (platerecognizer.com) yang telah dihubungkan. Setelah berhasil mengakses alamat API, permintaan tersebut akan diteruskan ke server (platerecognizer.com). Jadi, API akan memberitahukan bahwa aplikasi membutuhkan data nomor pelat sesuai gambar yang dikirim. Setelah memproses foto dan menemukan data yang diinginkan, server kembali menghubungi API. Data tersebut berupa nomor plat kendaraan. Selanjutnya, API meneruskan informasi dari server ke aplikasi kita.

4.2 Hasil Rancangan Aplikasi Web

Pada penelitian ini menggunakan web sebagai *user interface*. Web yang digunakan dibuat dengan bahasa pemrograman python dan Flask sebagai *frameworknya*.

4.2.1 Entity relationship Diagram



Gambar 4.13: ERD

Pada gambar 4.13, dapat dilihat terdapat 4 tabel yang saling berelasi antar tabel lain dan 1 tabel yang tidak mempunyai relasi antar tabel lain.

4.2.2 Struktur Database

Nama *database* : skripsi

Pada *database* aplikasi ini, tabel dibagi menjadi 5 tabel sebagai berikut:

1. Tabel rfid_tag

Nama Tabel : rfid_tag

Primary Key : id

Tabel 4.6: rfid_tag

Column	Type
id	varchar(15)
jenis_kendaraan	enum('mobil','motor')
saldo	bigint(8)

Atribut *id* pada tabel rfid_tag berfungsi sebagai kunci utama. Atribut jenis_kendaraan digunakan untuk menentukan jenis kendaraan yang digunakan

oleh pengemudi. Atribut saldo digunakan untuk menyimpan saldo dari pengemudi.

2. Tabel tempat_parkir

Nama Tabel : tempat_parkir

Primary Key : id

Tabel 4.7: tempat_parkir

Column	Type
id	varchar(3)
jenis	varchar(15)
tarif	int(5)
status	enum('Tersedia','Terpakai')

Tabel tempat_parkir mempunyai atribut *id* yang berfungsi sebagai kunci utama, atribut *id* juga berfungsi sebagai nomor slot tempat parkir. Atribut jenis digunakan untuk menentukan jenis dari slot parkir. Atribut tarif digunakan sebagai tarif per jam dari slot parkir. Atribut status digunakan untuk mengetahui apakah slot sedang tersedia atau terpakai.

3. Tabel kendaraan

Nama Tabel : kendaraan

Primary Key : id

Foreign Key : id_rfid_tag

Tabel 4.8: kendaraan

Column	Type
id	int(4)
nomor_plat	varchar(8)
id_rfid_tag	varchar(15)

Tabel kendaraan mempunyai atribut *id* yang berfungsi sebagai kunci utama. Atribut nomor_plat berfungsi untuk menyimpan nomor plat pengendara. Atribut id_rfid_tag didapat dari tabel rfid_tag.

4. Tabel parkir

Nama Tabel : parkir

Primary Key : id

Foreign Key : id_rfid_tag, id_tempat_parkir

Tabel 4.9: parkir

<i>Column</i>	<i>Type</i>
id	varchar(4)
id_rfid_tag	varchar(15)
nomor_plat	varchar(8)
id_tempat_parkir	varchar(3)
waktu_masuk	datetime
waktu_keluar	datetime

Tabel kendaraan mempunyai atribut *id* yang berfungsi sebagai kunci utama. Atribut *id_rfid_tag* didapat dari tabel *rfid_tag*. Atribut *nomor_plat* digunakan untuk menyimpan nomor plat pengendara. Atribut *id_tempat_parkir* didapat dari tabel *tempat_parkir*. Atribut *waktu_masuk* digunakan untuk mencatat waktu masuk pengendara. Atribut *waktu_keluar* digunakan untuk mencatat waktu keluar pengguna.

5. Tabel *reload_id*

Nama Tabel : *reload_id*

Primary Key : id

Tabel 4.10: reload_id

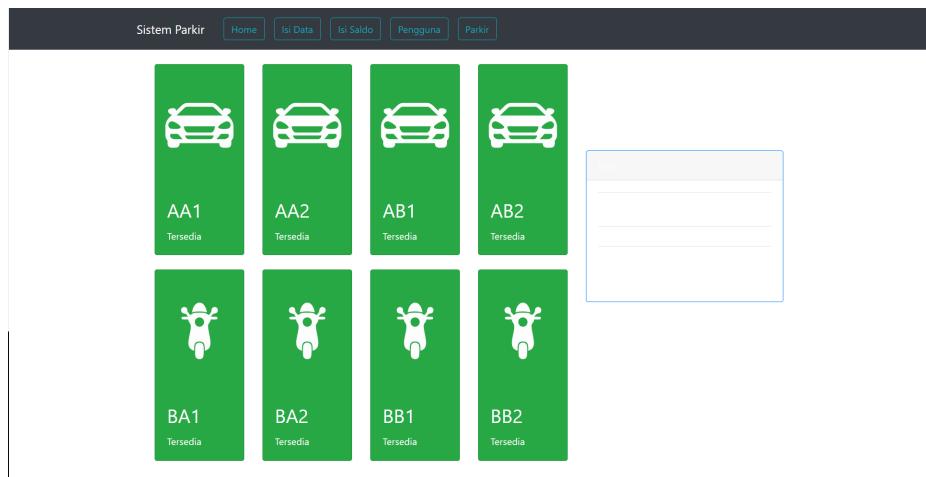
<i>Column</i>	<i>Type</i>
id	int(3)
uid	varchar(20)

Tabel *reload_id* mempunyai atribut *id* yang berfungsi sebagai kunci utama. Atribut *uid* digunakan untuk menyimpan id rfid tag.

4.2.3 Tampilan Website

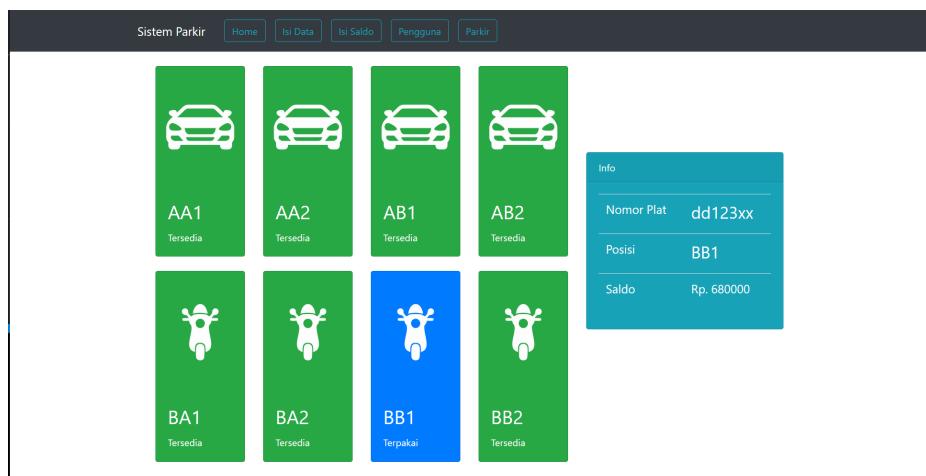
Pada tahap ini dilakukan pengimplementasian aplikasi. Berikut penjelasan mengenai tampilan aplikasi.

4.2.3.1 Tampilan Layout Parkir

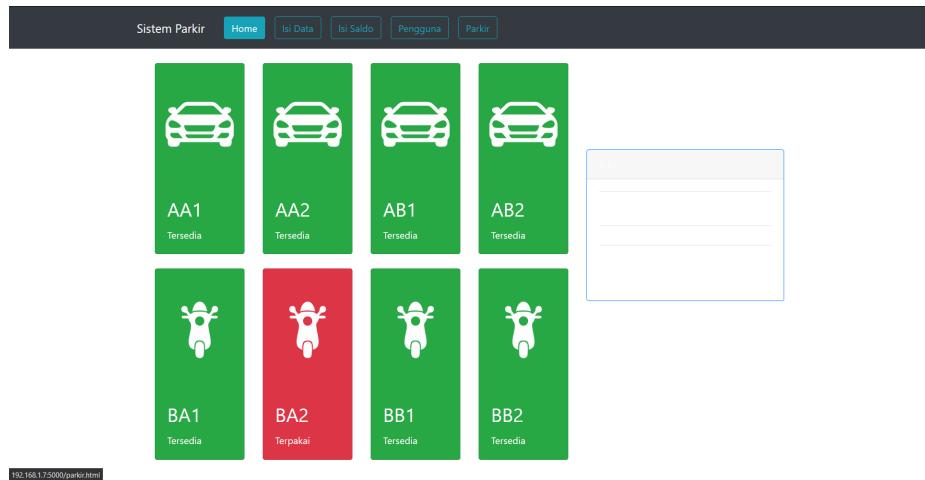


Gambar 4.14: Layout Parkir

Layout parkir berguna untuk menampilkan informasi mengenai jumlah slot parkir. Pada tampilan kita bisa melihat jumlah slot yang terisi dan terpakai. Pada gambar 4.14 terlihat jumlah slot parkir ada delapan buah dan semuanya tersedia.



Gambar 4.15: Layout Parkir Saat Pengendara Baru Masuk



Gambar 4.16: Layout Parkir Saat Ada Kendaraan

Pada gambar 4.15 terlihat warna salah satu slot parkir berubah dari warna hijau ke warna biru, ini menunjukkan slot yang harus diisi oleh pengendara pada saat baru masuk. Terlihat juga informasi seperti nomor plat, posisi parkir, dan saldo dari pengendara.

Pada gambar 4.16 terlihat warna slot parkir berubah menjadi warna merah, itu menandakan bahwa slot tersebut sedang terisi.

4.2.3.2 Tampilan Isi Data

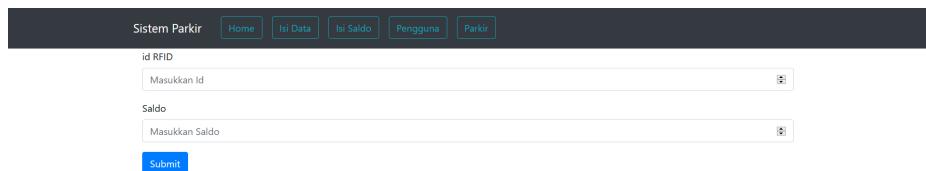
A screenshot of a "Form Isi Data" (Enter Data) page. The top navigation bar includes "Sistem Parkir", "Home", "Isi Data" (highlighted in blue), "Isi Saldo", "Pengguna", and "Parkir". The main content area contains four input fields: "id RFID" with value "9092124512", "Jenis kendaraan" with value "mobil", and "Saldo" with placeholder "Masukkan Saldo". Below these fields is a blue "Submit" button.

Gambar 4.17: Form Isi Data

Gambar 4.17 merupakan gambar *Form* isi data yang berfungsi untuk memasukkan informasi data pengendara kedalam *database*. Data yang dimasukkan pada *Form* inputan yaitu id RFID, jenis kendaraan, dan saldo. Setelah data diinput

maka data akan tersimpan pada *database*.

4.2.3.3 Tampilan Isi Saldo

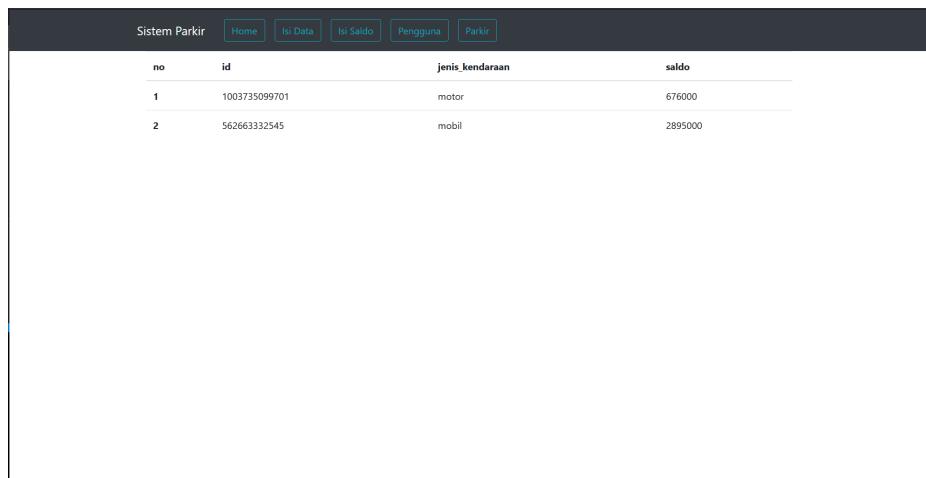


The screenshot shows a web-based application for managing parking. At the top, there's a navigation bar with the title 'Sistem Parkir' and several tabs: 'Home', 'Isi Data', 'Isi Saldo' (the active tab), 'Pengguna', and 'Parkir'. Below the tabs, there are two input fields: one for 'id RFID' with the placeholder 'Masukkan Id' and another for 'Saldo' with the placeholder 'Masukkan Saldo'. At the bottom of the form is a blue 'Submit' button.

Gambar 4.18: Form Isi Saldo

Gambar 4.18 merupakan gambar *Form* isi saldo yang berfungsi untuk menambahkan saldo pengendara kedalam *database*. Data yang dimasukkan pada *Form* inputan yaitu id RFID dan jumlah saldo yang ingin ditambahkan. Setelah data diinput maka data akan tersimpan pada *database*.

4.2.3.4 Tampilan Daftar Pengguna



The screenshot shows a table of user data. The table has four columns: 'no', 'id', 'jenis_kendaraan', and 'saldo'. There are two rows of data. Row 1: id 1003735099701, jenis_kendaraan motor, saldo 676000. Row 2: id 562663332545, jenis_kendaraan mobil, saldo 2895000.

no	id	jenis_kendaraan	saldo
1	1003735099701	motor	676000
2	562663332545	mobil	2895000

Gambar 4.19: Daftar Pengguna

Gambar 4.19 merupakan gambar Tampilan Daftar Pengguna yang berfungsi untuk melihat pengguna yang telah terdaftar. Untuk mendaftarkan pengguna bisa

dilakukan di *Form* isi data.

4.2.3.5 Tampilan Informasi

Sistem Parkir						
	Home	Isi Data	Isi Saldo	Pengguna	Parkir	
no	id_rfid_tag	nomor_plat	id_tempat_parkir	waktu_masuk	waktu_keluar	
1	1003735099701	A888AH	BA2	2021-04-15 07:23:11	2021-04-15 07:31:57	
2	1003735099701	A888AH	BA2	2021-04-15 07:34:29	2021-04-15 07:36:08	
3	1003735099701	A888AH	BA2	2021-04-15 07:37:01	2021-04-15 07:38:11	
4	562663332545	A888AH	AB1	2021-04-15 07:51:43	2021-04-15 07:52:09	
5	562663332545	B189LAU	AB2	2021-04-15 07:54:12	2021-04-15 07:54:36	
6	562663332545	B189LAU	AB2	2021-04-15 07:55:12	2021-04-15 07:55:32	

Gambar 4.20: *Informasi*

Gambar 4.20 merupakan gambar Tampilan Informasi yang berfungsi untuk menampilkan informasi dari pengendara yang sedang parkir. Tampilan ini juga bisa digunakan sebagai tampilan *history* karena menampilkan riwayat parkir semua pengendara.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang diuraikan pada bab IV, maka peneliti menarik beberapa kesimpulan sebagai berikut :

1. Sistem identifikasi kendaraan pada pemarkiran dengan pengenalan citra plat dan pembacaan rfid berhasil diimplementasikan. Rancangan alat yang dibuat masih bersifat *prototype*. Adapun alat yang digunakan yaitu Raspberry Pi, RFID, ultrasonik, servo, dan kamera.
2. Membuat aplikasi web sebagai *user interface*. Web dibuat dengan bahasa pemrograman python dan flask sebagai *framework*. Web yang dibuat memiliki beberapa fitur seperti melihat slot yang terisi dan tersedia, mendaftarkan pengguna baru, menambah saldo, dan melihat informasi mengenai slot parkir.

5.2 Saran

Adapun saran yang diberikan kepada peneliti berikutnya apabila ingin mengembangkan penelitian ini agar menjadi lebih baik adalah sebagai berikut :

1. Melakukan penelitian dan analisis lebih lanjut pada deteksi nomor pelat kendaraan karena pada penelitian ini, peneliti masih menggunakan API dari pihak ketiga.
2. Menambahkan fitur tambah slot parkir agar sistem parkir menjadi lebih fleksibel.
3. Menambahkan sensor disetiap slot parkir untuk mengetahui apakah pengendara sudah parkir ditempat yang ditentukan.

DAFTAR PUSTAKA

- Azli Yahya, P.C. (2017). “4th Industrial Revolution: The Future of Machining”. en. In: *4th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*. Semarang: Institute of Electrical and Electronics Engineers, p. 3.
- Budianto, A. (2018). “Automatic License Plate Recognition: A Review with Indonesian Case Study”. en. In: *Scientific Journal of Informatics* 5.2, pp. 258–270.
- Ichwana, D. dkk. (2018). “Sistem Cerdas Reservasi dan Pemantauan Parkir pada Lokasi Kampus Berbasis Konsep Internet of Things”. en. In: *Jurnal Teknologi dan Sistem Komputer* 6.ue 2, pp. 57–63.
- Limantara, A.D., Y.C. Purnomo, and S.W. Mudjanarko (2017). “Pemodelan Sistem Pelacakan Lot Parkir Kosong Berbasis Sensor Ultrasonic dan Internet of Things (Iot) Pada Lahan Parkir Diluar Jalan”. pt. In: *Seminar Nasional Sains dan Teknologi*, p. 1.
- Rahayu, N. (2019). “Mei 7). Mengenal Revolusi Industri dari 1.0 hingga 4.0”. io. Retrieved from Warta Ekonomi: URL: <https://www.wartaekonomi.co.id/read226785/mengenal-revolusi-industri-dari-10-hingga-40.html>.
- Singgeta, R.L., P.D. Manembu, and M.D. Rembet (2018). “Sistem Pengamanan Pintu Rumah Dengan RFID Berbasis Wireless ESP8266”. en. Seminar Nasional Riset dan Teknologi Terapan, pp. 87–97.
- Suharman and W. Murti (2019). “Kajian Industri 4.0 Untuk Penerapannya di Indonesia”. no. In: *Jurnal Manajemen Industri dan Logistik* 03.1, p. 1.
- Wicaksono, M.F. (2018). “Mudah Belajar Raspberry Pi”. id. Bandung: INFORMATIKA.

LAMPIRAN

1. RFID

```
# SDA connects to Pin 24.  
# SCK connects to Pin 23.  
# MOSI connects to Pin 19.  
# MISO connects to Pin 21.  
# GND connects to Pin 6.  
# RST connects to Pin 22.  
# 3.3v connects to Pin 1.  
  
import RPi.GPIO as GPIO  
from mfrc522 import SimpleMFRC522  
GPIO.setwarnings(False)  
  
reader = SimpleMFRC522()  
  
def tempel():  
    print("tempel kartu")  
    id = reader.read()  
    return id  
    # print("id kartu adalah = {}".format(id))  
  
GPIO.cleanup()
```

2. Servo

```
# pin kuning (pwm) di pin 3 raspi  
# pin orange (5v) di pin 4 raspi  
# pin coklat (gnd) di pin 6 raspi  
  
import RPi.GPIO as GPIO  
import sys  
import time
```

```

import ultra # ultra.py
GPIO.setwarnings(False)

def bukaServo():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(3,GPIO.OUT)
    q=GPIO.PWM(3,50)
    # print('awal 0 derajat')
    q.start(2.5) #90 derajat

    #q=GPIO.PWM(3,50)
    q.ChangeDutyCycle(7.5) #90 derajat
    time.sleep(1)

def tutupServo():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(3,GPIO.OUT)
    q=GPIO.PWM(3,50)

    q.start(2.5) #90 derajat

    q.ChangeDutyCycle(2.5) #0 derajat
    time.sleep(1)

GPIO.cleanup()

```

3. Ultrasonik

```

import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)
#GPIO.setmode(GPIO.BOARD)

def jarakMobil(min, max):

```

```

GPIO.setmode(GPIO.BOARD)

TRIG = 11
ECHO = 12

jarakMinimal = min
jarakMaksimal = max

GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)

while True:
    GPIO.output(TRIG,False)
    print('Pengukuran dimulai')
    time.sleep(2)
    GPIO.output(TRIG,True)
    time.sleep(0.00001)
    GPIO.output(TRIG,False)

    while GPIO.input(ECHO) == 0:
        pulse_start = time.time()

    while GPIO.input(ECHO) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start

    distance = pulse_duration * 17150
    distance = round(distance, 2)

    if distance > jarakMinimal and distance
        < jarakMaksimal :
        return True

GPIO.cleanup()

```

4. Kamera

```
import picamera
import time
import requests

def plate_recognition(rfid_tag) :
    img_name = f"/static/imgpelat/{rfid_tag}.jpg"

    camera = picamera.PiCamera()

    looping = True

    while looping:
        camera.start_preview()
        time.sleep(1)

        print("sedang mengambil gambar")
        camera.capture(img_name)
        print("gambar diambil")

        camera.stop_preview()

    api_token = '6955f6366338a6078826f7d89031c17179986082'

    with open(img_name, 'rb') as fp:
        response = requests.post(
            'https://api.platerecognizer.com/\
                v1/plate-reader/',
            files=dict(upload=fp),
            headers={'Authorization': 'Token ' \
                + api_token})
        result = response.json()
```

```

    try :
        plate = result.get("results")[0].get("plate")
        looping = False
    except IndexError :
        print("mengambil gambar lagi")
        looping = True

    camera.close()
    print(plate.upper())
    return plate.upper()

```

5. Main

```

import RPi.GPIO as GPIO
import servo # servo.py
import rfid # rfid.py
import ultra # ultra.py
import time
import kamerarevisi as kamera
import requests
GPIO.setwarnings(False)

while True:
    #baca tag rfid
    idRfid, textRfid = rfid.tempel()
    print("id kartu adalah = ", idRfid)
    print("text kartu adalah = ", textRfid)

    # ambil foto
    nomor_pelat = kamera.plate_recognition(idRfid)

    # kirim data
    data = {
        "id_rfid_tag":idRfid,
        "nomor_plat":nomor_pelat

```

```

        }

    response = requests.post(
        'http://localhost:5000/apimasuk',
        data=data
    )

    if responsejson['status'] == 'terdaftar' \
        and responsejson['tempat_parkir'] == 'tersedia':
        print('masuk atau keluar')
        nomor_pelat = ""

        # buka palang
        servo.bukaServo()

        # periksa keberadaan mobil
        ultra.jarakMobil(7, 400)

        # tutup palang
        time.sleep(1)
        servo.tutupServo()

    GPIO.cleanup()

    elif responsejson['status'] == 'terdaftar' \
        and responsejson['tempat_parkir'] ==\
        'tidak tersedia':
        print('id terdaftar dan parkir tidak tersedia')
    else :
        print('id tidak terdaftar')

```

6. Web

```

from flask import Flask, render_template, url_for, \
    request, redirect, jsonify
from flask_mysqldb import MySQL

```

```

from flask_api import FlaskAPI, status, exceptions
from random import randint
from math import ceil
import datetime
app = Flask(__name__)

app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'raspberry'
app.config['MYSQL_DB'] = 'skripsi'

mysql = MySQL(app)

app.route("/", methods=['GET', 'POST'])
def myapp():
    return render_template('main_layout.html')
#    return redirect('/index.html')

app.route("/parkir.html", methods=['GET', 'POST'])
def layout_parkir():
    plat = ""
    posisi_parkir = ""
    saldo = ""
    table_parkir = load_table_parkir()
    table_rfid_tag = load_table_rfid_tag()
    tempat_parkir = load_table_tempat_parkir()
    text_keterangan = ""
    keterangan = ""

    if len(table_parkir) > 0 :
        last = table_parkir[-1]

        waktu_sekarang = datetime.datetime.now()

```

```

if last["waktu_keluar"] == None:
    waktu_masuk = datetime.datetime.\n
        strptime(str(last["waktu_masuk"]), \
        '%Y-%m-%d %H:%M:%S')
    waktu_masuk_5detik = waktu_masuk \
        + datetime.timedelta(seconds=5)
if waktu_sekarang < waktu_masuk_5detik:
    plat = last["nomor_plat"]
    posisi_parkir = last["id_tempat_parkir"]
    for data in table_rfid_tag:
        if data["id"] == last["id_rfid_tag"]:
            saldo = f"Rp. {data['saldo']}"
            text_keterangan="Posisi"
            keterangan = posisi_parkir
            break

else :
    waktu_keluar = datetime.datetime.\n
        strptime(str(last["waktu_keluar"]), \
        '%Y-%m-%d %H:%M:%S')
    waktu_keluar_5detik = waktu_keluar \
        + datetime.timedelta(seconds=5)
    if waktu_sekarang < waktu_keluar_5detik:
        # print(last)
        tarif = hitung_tarif2(last, last, tempat_parkir)
        plat = last["nomor_plat"]
        posisi_parkir = last["id_tempat_parkir"]
        for data in table_rfid_tag:
            if data["id"] == last["id_rfid_tag"]:
                saldo = f"Rp. {data['saldo']}"
                text_keterangan = "Tarif"
                keterangan = tarif
                break

```

```

        return render_template('index.html', tempat_parkir \
= tempat_parkir, keterangan=keterangan, text_keterangan\
=text_keterangan, posisi_parkir=posisi_parkir, \
plat=plat, saldo=saldo, time = \
str(datetime.datetime.now()))

def tentukan_tempat_parkir(tempat_parkir, jenis_kendaraan):
    slot_tersedia = 0
    tempat_parkir_tersedia = []
    for tempat in tempat_parkir:
        if tempat["jenis"] == jenis_kendaraan and\
            tempat["status"] == "Tersedia":
            slot_tersedia = slot_tersedia + 1
            tempat_parkir_tersedia.append(tempat)

    if slot_tersedia == 0 :
        return "Penuh"
    index = randint(0,slot_tersedia-1)
    return tempat_parkir_tersedia[index]

app.route("/pengguna")
def pengguna():
    pengguna = load_table_rfid_tag()
    return render_template("pengguna.html", result=pengguna \
,panjangResult=len(pengguna))

def load_table_rfid_tag():
    cur = mysql.connection.cursor()
    cur.execute("select * from rfid_tag")
    tuple_rfid_tag = cur.fetchall()
    mysql.connection.commit()
    cur.close()

```

```

result = []
for data in tuple_rfid_tag:
    rfid_tag = {
        "id":"",
        "jenis_kendaraan":"",
        "saldo":""
    }
    rfid_tag["id"] = data[0]
    rfid_tag["jenis_kendaraan"] = data[1]
    rfid_tag["saldo"] = data[2]
    result.append(rfid_tag)
return result

def load_table_tempat_parkir():
    cur = mysql.connection.cursor()
    cur.execute("select * from tempat_parkir")
    tuple_tempat_parkir = cur.fetchall()
    mysql.connection.commit()
    cur.close()

    result = []
    for data in tuple_tempat_parkir:
        tempat_parkir = {
            "id":"",
            "jenis":"",
            "tarif":"",
            "status":""
        }
        tempat_parkir["id"] = data[0]
        tempat_parkir["jenis"] = data[1]
        tempat_parkir["tarif"] = data[2]
        tempat_parkir["status"] = data[3]

```

```

        result.append(tempat_parkir)

    return result

app.route("/table_parkir")
def table_parkir():
    table_parkir = load_table_parkir()
    return render_template("table_parkir.html", \
                           result=table_parkir ,panjangResult=len(table_parkir))

def load_table_parkir():
    cur = mysql.connection.cursor()
    cur.execute("select * from parkir")
    tuple_parkir = cur.fetchall()
    mysql.connection.commit()
    cur.close()

    result = []
    for data in tuple_parkir:
        parkir = {
            "id":"",
            "id_rfid_tag":"",
            "nomor_plat":"",
            "id_tempat_parkir":"",
            "waktu_masuk":"",
            "waktu_keluar":""
        }
        parkir["id"] = data[0]
        parkir["id_rfid_tag"] = data[1]
        parkir["nomor_plat"] = data[2]
        parkir["id_tempat_parkir"] = data[3]
        parkir["waktu_masuk"] = data[4]
        parkir["waktu_keluar"] = data[5]
        result.append(parkir)

```

```

        return result

def update_status_tempat_parkir(tempat_parkir, status):
    cur = mysql.connection.cursor()
    # print("ini status = ",status)
    # print(type(status))
    # print("ini tempat_parkir = ",tempat_parkir)
    # print(type(tempat_parkir))
    cur.execute(f"update tempat_parkir set status='{status}' \
                where id='{tempat_parkir}';")
    mysql.connection.commit()
    cur.close()

def save_table_parkir(kendaraan, tempat_parkir, waktu_masuk \
                      = "NULL", waktu_keluar = "NULL"):
    id_rfid_tag = kendaraan["id_rfid_tag"]
    nomor_plat = kendaraan["nomor_plat"]

    cur = mysql.connection.cursor()
    # masuk
    if waktu_masuk != "NULL":
        id_tempat_parkir = tempat_parkir["id"]
        cur.execute(f"insert into parkir(id_rfid_tag, \
                    nomor_plat, id_tempat_parkir, waktu_masuk, \
                    waktu_keluar) values ('{id_rfid_tag}', \
                    '{nomor_plat}', '{id_tempat_parkir}', \
                    {waktu_masuk}, NULL) ;")

    # keluar
    elif waktu_keluar != "NULL":
        cur.execute(f"update parkir set waktu_keluar\
                    ={waktu_keluar} where id_rfid_tag={id_rfid_tag}\
                    AND id_tempat_parkir='{tempat_parkir}'\
                    AND waktu_keluar is NULL;")

```

```

mysql.connection.commit()
cur.close()

def save_table_kendaraan(kendaraan):
    nomor_plat = kendaraan["nomor_plat"]
    id_rfid_tag = kendaraan["id_rfid_tag"]

    cur = mysql.connection.cursor()
    cur.execute(f"insert into kendaraan(nomor_plat, \
        id_rfid_tag) values ('{nomor_plat}', \
        '{id_rfid_tag}');")
    mysql.connection.commit()
    cur.close()

# @app.route("/test_api", methods=["POST", "GET"])
# def test_api():
#     no_id = request.form["id_rfid_tag"]
#     dd = request.form["nomor_plat"]
#     cek_rfid_tefdaftar=load_table_rfid_tag()
#     for data in cek_rfid_tefdaftar:
#         if data["id"] == no_id:
#             r = {'status':'terdaftar'}
#             return jsonify(r), 200

#     print("dari app id tdk terdaftar")
#     r = {'status':'tidak terdaftar','test':'juga'}
#     return jsonify(r)

app.route("/apimasuk", methods=["POST", "GET"])
def api_masuk():
    kendaraan = {
        "id_rfid_tag":"",
        "nomor_plat":",

```

```

        "jenis_kendaraan": ""

    }

kendaraan["id_rfid_tag"] = request.form["id_rfid_tag"]
kendaraan["nomor_plat"] = request.form["nomor_plat"]

# table rfid_tag
table_rfid_tag = load_table_rfid_tag()

# cek id terdaftar
terdaftar = False
for data in table_rfid_tag:
    if data["id"] == kendaraan["id_rfid_tag"]:
        terdaftar = True
        break

# id tidak terdaftar
if not terdaftar:
    cur = mysql.connection.cursor()
    cur.execute("update reload_id set uid = %s \
                where id = 1;", (kendaraan["id_rfid_tag"],))
    # cur.execute("delete from user_id;")
    mysql.connection.commit()
    cur.close()

    r = {'status': 'tidak terdaftar', \
          'tempat_parkir': 'tidak tersedia'}
    return jsonify(r)

# cek jenis kendaraan
for rfid_tag in table_rfid_tag:
    id_rfid = rfid_tag["id"]
    jenis_kendaraan = rfid_tag["jenis_kendaraan"]
    if kendaraan["id_rfid_tag"] == id_rfid:

```

```

        kendaraan["jenis_kendaraan"] = jenis_kendaraan
        break

# cek apakah kendaraan masuk atau keluar
table_parkir = load_table_parkir()
cek_keluar = False
for data in table_parkir:
    if data["id_rfid_tag"] == kendaraan["id_rfid_tag"]\ 
        and data["waktu_keluar"] == None:
        cek_keluar = True
    break

activity = ""
# Parkir Keluar
if cek_keluar:
    # cek ini saldo
    tarif = hitung_tarif(kendaraan, table_parkir)

    # update saldo kendaraan
    update_saldo(kendaraan, tarif)

tempat_parkir = []
for data in table_parkir:
    # cocokkan waktu keluar
    if data["id_rfid_tag"] == kendaraan["id_rfid_tag"]\ 
        and data["waktu_keluar"] == None:
        tempat_parkir.append(data)
    break

# update status tempat parkir di table
# tempat_parkir menjadi tersedia
update_status_tempat_parkir(tempat_parkir[0]\ 
    ['id_tempat_parkir'], "Tersedia")

```

```

# update waktu keluar
waktu_keluar = "now()"
save_table_parkir(kendaraan, tempat_parkir[0]\
['id_tempat_parkir'], waktu_keluar=waktu_keluar)
activity = "keluar"

# Parkir Masuk
else:
    # menentukan tempat parkir
    table_tempat_parkir = load_table_tempat_parkir()
    tempat_parkir = tentukan_tempat_parkir\
        (table_tempat_parkir,kendaraan["jenis_kendaraan"])

    # cek parkir penuh
    if tempat_parkir == "Penuh":
        r = {'status':'terdaftar', 'tempat_parkir'\
            :'tidak tersedia'}
        return jsonify(r)

    # update table parkir
    waktu_masuk = "now()"
    save_table_parkir(kendaraan, tempat_parkir, \
        waktu_masuk=waktu_masuk)

    # update status tempat parkir di table
    # tempat_parkir menjadi terpakai
    update_status_tempat_parkir(tempat_parkir["id"], \
        "Terpakai")

    # update table kendaraan
    save_table_kendaraan(kendaraan)
    activity = "masuk"

```

```

r = {'status':'terdaftar', 'tempat_parkir':'tersedia', \
      'aktivitas':activity}

return jsonify(r)

# ini keluar

def hitung_tarif(kendaraan, table_parkir):
    # print(kendaraan)

    table_tempat_parkir = load_table_tempat_parkir()
    tarif = 0

    for data in table_parkir:
        for tempat_parkir in table_tempat_parkir:
            if data["id_rfid_tag"] == kendaraan["id_rfid_tag"]\

                and data["waktu_keluar"] == None and \

                data["id_tempat_parkir"] == \
                tempat_parkir["id"]:

                waktu_masuk = datetime.datetime.strptime\
                    (str(data["waktu_masuk"]), \
                     '%Y-%m-%d %H:%M:%S')

                waktu_keluar = datetime.datetime.now()

                durasi = waktu_keluar - waktu_masuk

                durasikonversi = ceil(durasi.total_seconds()\
                    /3600)

                tarif = durasikonversi * int(tempat_parkir\
                    ["tarif"])

                return tarif

    return tarif

def hitung_tarif2(kendaraan, data, table_tempat_parkir):
    tarif = 0

    waktu_masuk = datetime.datetime.strptime(str\
        (data["waktu_masuk"]), '%Y-%m-%d %H:%M:%S')

```

```

waktu_keluar = datetime.datetime.now()
durasi = waktu_keluar - waktu_masuk
durasikonversi = ceil(durasi.total_seconds()/3600)
for tempat_parkir in table_tempat_parkir:
    if tempat_parkir["id"] == data["id_tempat_parkir"]:
        tarif = durasikonversi * int(tempat_parkir\
            ["tarif"])
        break
    return tarif

# ini keluar
def update_saldo(kendaraan, tarif):
    id_rfid = kendaraan["id_rfid_tag"]
    cur = mysql.connection.cursor()
    cur.execute(f"select saldo from rfid_tag \
        where id = {id_rfid};")
    saldoAwal = cur.fetchall()
    # return render_template('test.html', \
    #     saldoAwal = saldoAwal[0][0])

    saldoSekarang = saldoAwal[0][0] - int(tarif)

    cur.execute(f"update rfid_tag set saldo=\
        {saldoSekarang} where id={id_rfid};")
    mysql.connection.commit()
    cur.close()

app.route("/isi_data", methods=["POST", "GET"])
def isi_data():
    if request.method == "POST":
        idRFID = request.form["idRFID"]
        jenis_kendaraan = request.form["jenis_kendaraan"]
        saldo = request.form["saldo"]

```

```

        cur = mysql.connection.cursor()
        cur.execute(f"insert into rfid_tag(id, \
                jenis_kendaraan, saldo) values \
                ({idRFID}, '{jenis_kendaraan}', {saldo});")
        mysql.connection.commit()
        cur.close()
        return redirect(url_for("layout_parkir"))
    return render_template('isi_data.html')

app.route("/reload_id", methods=["GET", "POST"])
def proses():
    cur = mysql.connection.cursor()
    cur.execute("select * from reload_id ;")
    user = cur.fetchone()
    cur.close()

    # print(user[1])
    # print(type(user))
    # return jsonify({'id':user})

    if user == None:
        return jsonify({'id':'otomatis terisi'})

    return jsonify({'id':user[1]})

app.route("/isi_saldo", methods=["POST", "GET"])
def isi_saldo():
    if request.method == "POST":
        idRFID = request.form["idRFID"]
        saldoIsi = request.form["saldo"]

        cur = mysql.connection.cursor()

```

```
cur.execute(f"select saldo from rfid_tag where id =\\
    {idRFID};")
saldoAwal = cur.fetchall()
# return render_template('test.html', saldoAwal =\
#     saldoAwal[0][0])

saldoSekarang = int(saldoIsi) + saldoAwal[0][0]

cur.execute(f"update rfid_tag set saldo=\\
    {saldoSekarang} where id={idRFID};")
mysql.connection.commit()
cur.close()
return redirect(url_for("layout_parkir"))
return render_template('isi_saldo.html')

if __name__ == '__main__':
    app.run(debug=True, host="0.0.0.0")
```