BAB I

PENDAHULUAN

A. Latar Belakang

Pembangunan populasi motor yang melimpah di beberapa kota besar di Indonesia menghasilkan semua jenis masalah, salah satunya adalah lahan parkir terbatas dan manajemen parkir tidak efektif. Tipe parkir konvensional yang tetap digunakan pada saat ini cenderung dapat menyebabkan terjadinya antrian panjang, kesulitan tracking motor, serta rentan pada kebocoran pendapatan retribusi parkir. Selain itu, rendahnya integrasi antara sistem parkir dengan profil pengguna palsu turut menjengahkan pengawasan dan pengendalian atas kendaraan berlapis tanpa hak (Arifin, Prasetyo, Alfaiz, Rachmanto, Juwana, Widhiyanuariyawan,& Yohana,(2025).

Di tengah persaingan itu, implementasi sistem E-Parkir atau parkir elektronik menjadi solusi semakin tepat. Sistem ini memungkinkan pengelolaan parkir yang lebih canggih, efektif, dan transparan melalui teknologi digital. Akan tetapi, kebanyakan sistem E-Parkir masih bergantung pada kartu khusus atau aplikasi mobile yang belum terintegrasi dengan identitas resmi pemilik kendaraan.

Untuk mendorong efektifitas dan keamanan sistem parkir, pengembangan E-Parkir lot-based menggunakan SIM card sebagai sarana autentikasi adalah pilihan innovatif. Kartu SIM, sebagai dokumen resmi dan pribadi per pengendara, dapat dilibatkan untuk mengenal pengguna langsung tanpa kartu cadangan. Sistem ini

juga mampu dimasukkan pada *database* kependudukan ataupun kepolisian untuk meningkatkan akuratnya dan kemanfaatannya.

Melalui metode ini, tidak hanya efisiensi parkir tetapi juga pengawasan, keamanan, dan akuntabilitas parkir umum dapat ditingkatkan. Dengan demikian, penelitian dan pengembangan sistem E-Parkir berbasis lahan dengan menggunakan kartu SIM menjadi suatu keharusan untuk memfasilitasi transformasi digital dalam manajemen transportasi dan parkir di Indonesia (Andeta, 2024).

B. Rumusan Masalah

1. Bagaimana merancang sistem E-Parkir berbasis lot yang dapat mengidentifikasi pengguna melalui kartu SIM ?
2. Bagaimana sistem ini dapat meningkatkan efisiensi dan keamanan pengelolaan parkir dibandingkan sistem parkir konvensional ?

C. Tujuan penelitian

1. Mengembangkan sistem E-Parkir berbasis lot yang terintegrasi dengan kartu Surat Izin Mengemudi (SIM) sebagai media autentikasi pengguna.
2. Menganalisis peningkatan efisiensi dan keamanan dalam manajemen parkir melalui penerapan sistem ini.

D. Batasan Masalah

1. Sistem E-Parkir yang dikembangkan hanya mencakup area parkir berbasis lot.
2. Identifikasi pengguna dilakukan menggunakan kartu SIM.

E. Manfaat Penelitian

Sistem E-Parkir berbasis lot yang dikembangkan melalui penelitian ini diharapkan mampu memberikan solusi nyata terhadap permasalahan parkir di

berbagai kawasan, terutama di lingkungan padat seperti kampus, pusat perbelanjaan, atau kawasan perkantoran. Penggunaan kartu SIM sebagai alat autentikasi diharapkan dapat mempermudah proses identifikasi pengguna, serta meningkatkan keamanan dan akuntabilitas sistem parkir. Selain itu, hasil dari penelitian ini dapat menjadi pertimbangan bagi pemerintah daerah maupun pengelola fasilitas publik dalam merancang sistem parkir yang lebih efisien, modern, dan terintegrasi dengan kebijakan transportasi cerdas.

BAB II

TINJAUAN PUSTAKA

A. Kajian teori

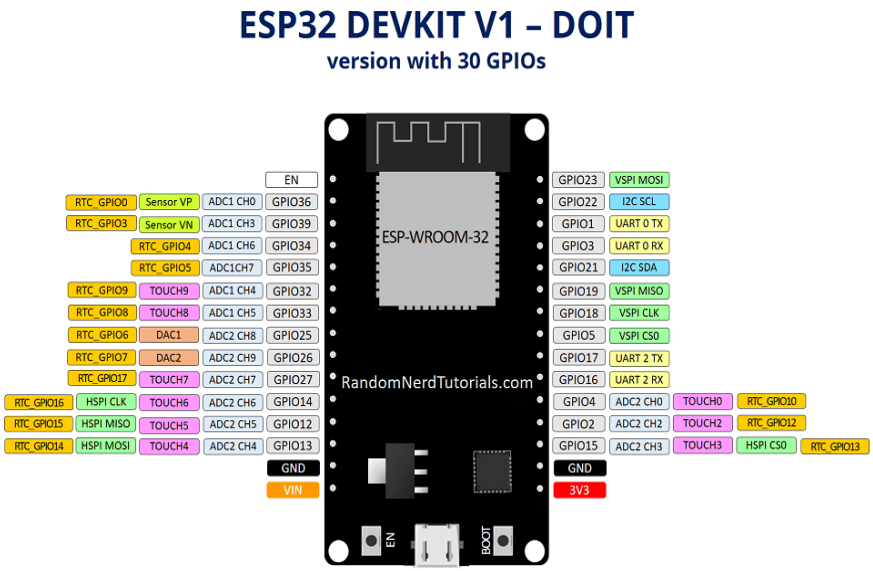
1. IOT *(Internet* Of *Things)*

Konsep teknologi *Internet* of *Things (IoT)* menghubungkan berbagai perangkat fisik ke internet agar dapat berkomunikasi, mengirimkan, dan menerima informasi secara otomatis tanpa campur tangan langsung dari manusia. Perangkat-perangkat ini dilengkapi dengan sensor, aktuator, dan *modul* komunikasi yang memungkinkan pengumpulan dan pertukaran informasi secara real-time. Dalam konteks sistem parkir, *IoT* memungkinkan pengelolaan yang lebih pintar melalui sensor parkir yang dapat mendeteksi keberadaan kendaraan, pembaca kartu atau sensor identifikasi pengguna, serta integrasi dengan *server* pusat untuk memproses data dan mengambil keputusan. Penerapan *IoT* dapat membuat sistem parkir menjadi lebih efisien, akurat, dan terotomatisasi, karena semua proses mulai dari deteksi kendaraan, pencatatan identitas pengguna, hingga pembayaran dapat dilakukan secara digital dan saling terhubung (Utomo, Teguh Prasetyo, 2019).

2. *ESP32*

ESP32 adalah sebuah mikrokontroler yang dikembangkan oleh Espressif Systems, dirancang khusus untuk aplikasi *Internet* of *Things (IoT)* karena memiliki kemampuan komputasi yang tinggi, konsumsi daya rendah, serta konektivitas nirkabel yang lengkap. Modul ini sudah dilengkapi dengan *Wi-Fi* dan *Bluetooth*, sehingga sangat ideal untuk digunakan dalam sistem yang membutuhkan

komunikasi data jarak jauh secara real-time, seperti E-Parkir berbasis *IoT*. Selain itu, *ESP32* memiliki banyak pin input/output (I/O), mendukung sensor dan perangkat eksternal seperti pembaca *RFID*, sensor ultrasonik, serta *display* digital. Dalam pengembangan sistem E-Parkir, *ESP32* dapat berfungsi sebagai otak pengendali yang mengatur pembacaan kartu (misalnya kartu SIM berbasis *RFID)*, memproses data kendaraan yang masuk dan keluar, serta mengirimkan data ke *server* atau aplikasi pengguna melalui jaringan *Wi-Fi*. Dengan fleksibilitas dan kemampuannya, *ESP32* menjadi pilihan yang sangat tepat untuk prototipe maupun implementasi sistem parkir pintar berbasis lot (putra, 2025).



Tabel 2. 1 Spesifikasi *ESP* 32

PIN GPIO	Fungsi	Keterangan	Spesifikasi
GPIO0	Boot mode, ADC2_CH1, Touch1	Mikrokontroler	Xtens dual-core 32-bitlx6
GPIO1	TX0 (UART0)	Kecepatan Clock	Hingan 240 MHz

GPIO2	ADC2_CH2, Touch2	Memori internal	520 KB (tergantuang modul)
GPIO3	RX0 (UART0)	We-fi	IEEE 802.11 b/g/n(mode AP,STA,AP + STA)
GPIO4	ADC2_CH0, Touch0	Bluetooth	Blue v 4.2 BR/ERD dan BLE
GPIO5	SPI CS, PWM, ADC2_CH12	Jumlah GPIO	30 GPIO (multi fungsi.input /output ,PWW,I2C,SPI,UART, ADC,dll)
GPIO12	Touch5, ADC2_CH5	ADC (Analong/input	15 chennel, 12 bit resolution
GPIO13	Touch4, ADC2_CH4	DAC (Analong output)	2 chennel (DAC1 di GPIO25,DAC2, di GPIO26)
GPIO14	SPI CLK, ADC2_CH6	Touch sensor	10 chennel sentuh kapasitif (TOUCHO-TOUC9)
GPIO15	ADC2_CH3, Touch3	SPI	2 bus :HSPI & VSPI
GPIO16	UART2 RX	I2C	2 bus, default: SDA (GPIO21), SCL (GPIO22)
GPIO17	UART2 TX	UART	Hingga 3 antarmuka (UART0, UART1,UART2,)
GPIO18	SPI CLK, PWM, ADC2_CH9	Sensor intrernal	Sensor Hall effect sensor suhu internal
GPIO19	SPI MISO	Tegangan oprasi	3.3V logika (butuh level shifter jika pakai 5V input)
GPIO21	I2C SDA	Tegangan input (VIN)	5V (melalui USB atau pin VIN)
GPIO22	I2C SCL	Port USB	Micro USB (dengan chip USB-to-Serial, umumnya CP21202
GPIO23	SPI MOSI	Mode Hemat Energi	Deep Sleep, light sleep, dan modem sleep
GPIO 25	DAC1, ADC2_CH8	Dimensi modul	Seketir 51 mm (tergantung versi board)
GPIO 26	DAC2, ADC2_CH9	Kompatibilitas software	Arduino IDE, platformi0,ESP-IDF (Espressisif IOT

			Development Framework
GPIO 27	Touch7, ADC2_CH7		
GPIO 32	ADC1_CH6		
GPIO 33	ADC1_CH5, Touch8		
GPIO 34	ADC1_CH6		
GPIO 35	ADC1_CH7		
GPIO 36	ADC1_CH0 (SVP)		
GPIO 39	ADC1_CH3 (SENSOR_VN)		

3.Stiker RFID

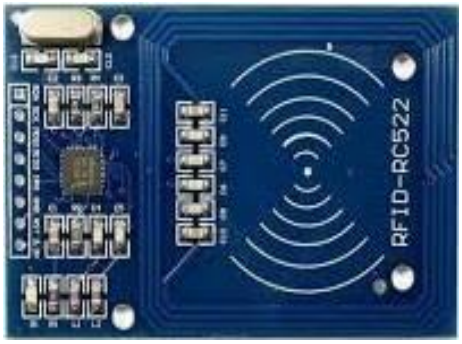
Stiker *RFID RC522 13.56 MHz* adalah jenis *tag RFID* pasif yang bekerja pada frekuensi tinggi (13.56 MHz) dan biasa digunakan untuk sistem identifikasi otomatis seperti akses kontrol, sistem parkir, absensi, hingga e-money. Stiker ini berbentuk tipis dan fleksibel, sehingga mudah ditempel pada berbagai permukaan seperti kartu, helm, kaca mobil, atau barang lainnya. Di dalamnya terdapat sebuah *chip RFID dan* antena kecil berbentuk spiral yang memungkinkan stiker tersebut menerima sinyal dari pembaca *RFID* seperti *modul RC522. Ketika* didekatkan ke pembaca, stiker akan diaktifkan oleh sinyal elektromagnetik dari *modul*, kemudian mengirimkan data unik yang tersimpan di dalamnya, seperti *UID (Unique Identifier)* atau data tambahan tergantung jenis chip yang digunakan. Karena bersifat pasif, stiker ini tidak memerlukan baterai untuk beroperasi, dan memiliki jarak baca yang relatif pendek, yaitu sekitar 2–5 cm. Stiker *RFID* ini menggunakan standar ISO 14443A dan sangat kompatibel dengan *modul RC522 yang* umum digunakan dalam proyek-proyek mikrokontroler seperti *Arduino* atau *ESP32.*



Gambar 2. 2 Stiker *RFID*

4. RFID Reader

RFID Reader adalah penghubung antara perangkat lunak aplikasi dan antena yang memancarkan gelombang radio ke tag *RFID.* Gelombang radio yang dipancarkan antena akan merambat ke ruang sekitarnya. Hasilnya, data dapat dikirim secara nirkabel ke tag *RFID* yang terletak di sebelah antenna ID-12 yang merupakan Reader yang khusus mendeteksi tag *RFID* dengan frekuensi 125 kHz. Tag *RFID* yang kompatibel dengan ID-12 termasuk GK4001 dan EM40001 Akibatnya data dapat berpindah secara wireless ke tag *RFID* yang berada berdekatan dengan antenna. *RFID* header merujuk pada deretan pin (pin header**)** yang terdapat pada *modul RFID,* seperti *modul RC522*, yang berfungsi sebagai antarmuka fisik untuk menghubungkan *modul* tersebut ke mikrokontroler (misalnya *ESP32, Arduino,* dll). Pin header ini terdiri dari beberapa pin dengan fungsi spesifik yang harus dihubungkan sesuai dengan protokol komunikasi yang digunakan—biasanya *SPI* (Serial Peripheral Interface) (Salsabila, Anwar & Syahputra, 2024.



Gambar 2. 3 *RFID* Reader

Tabel 2. 2 Spesifikasi *RFID* Header

Keterangan	Spesifikasi
SDA (SS/CS)	Pin *chip* select untuk mengaktifkan komunikasi SPI
SCK	Serial *Clock*, digunakan untuk sinkronisasi data
MOSI	*Master Out Slave In*, jalur data dari mikrokontroler ke *RFID*
MISO	*Master In Slave Out*, jalur data dari *RFID* ke mikrokontroler
IRQ	*Interrupt Request*, opsional dan jarang digunakan
GND	sambungan ke ground mikrokontroler
RST	untuk me-reset modul *RFID*
3.3V	Tegangan catu daya untuk *RFID*

5. *LCD OLED* 128x32

Secara umum, *display OLED* merupakan *modul* aktuator yang berfungsi untuk menampilkan huruf dan karakter (alfanumerik). Modul ini dapat digunakan untuk menampilkan pesan sederhana seperti 'Halo, Semesta', hasil pengukuran sensor seperti 'Temperatur: 29,30 °C', atau pesan kesalahan (error handling)

Dan juga merupakan kronim dari *Organic Light-Emitting Diode*, dimana lapisan film tipis (thin film) electroluminescent nya terbuat dari bahan organik yang dapat memancarkan cahaya ketika diberikan arus listrik. *OLED* ini kemudian dirangkai sedemikian rupa sehingga membentuk sebuah *display* dengan ketelitian pixel 128 x 64. Display *OLED* memiliki ukuran yang sangat kecil yaitu

panjangnya sekitar 0.96 inchi, meskipun demikian karakter yang ditampilkan pada *OLED* dapat dengan mudah dibaca (readable) dikarenakan *OLED* memiliki tingkat kontras yang tajam. *OLED* dapat diakses atau dihubungkan dengan mikrokontroller melalui komunikasi Serial Peripheral Interface *(SPI)* atau komunikasi Inter Integrated Circuit *(I2C).*



Gambar 2. 4 Lcd oled 128x32

Tabel 2. 3 Spesifikasi lcd oled 128x32

NO	Nama Pin	Fungsi
1	GND	*Ground*
2	VCC	Tegangan input
3	SCL	*CLOCK Line*(I2C)
4	SDA	Data Line(I2C)

6. Motor *Servo*

Motor *servo* adalah sebuah perangkat sebagai aktuator putar (motor) yang dirancang dengan sistem kontrol umpan balik loop tertutup (*servo*), sehingga dapat di atur untuk menentukan dan memastikan posisi sudut dari poros output motor. Potensiometer untuk memaksimalkan putaran pada motor servo, untuk arah putaran dan sudut dari sumbu motor servo dapat diatur menggunakan sinyal *PWM* (Pulse Width Modulation) (Amin, M., & Novelan, 2020).

Terdapat dua tipe motor *servo*, yaitu motor *servo* AC dan DC. Jika dibedakan berdasarkan rotasinya, terdapat dua jenis motor *servo*, yaitu 180º dan 360º. Dalam penelitian ini, motor *servo* yang digunakan adalah motor *servo* standar (*servo* 180º) (Fonna, Husaini, & Indrawati, (2020).

Motor *servo* standar (servo 180º) merupakan jenis yang paling lazim digunakan, dengan poros output yang hanya dapat berputar 90º ke kanan dan 90º ke kiri. Artinya, total putarannya cuma 180º. Karena motor ini hanya bergerak ke kanan, kembali ke tengah, dan ke kiri saja, tidak dapat mencapai satu putaran penuh (Deddy Pradana, 2021).



Gambar 2. 5 Motor Servo

Tabel 2. 4 Spesifikasi Motor *Servo*

Spesifikasi	Keterangan
Berat	9g
Rentang rotasi	180°
Tegangan oprasi	4.8V-6V
Suhu oprasi	-30°C to + 60°C
Kecepatan	0.1 sec/60° (4.8v) 0.08sec/60° (6v)
Rotasi maksimum	1.6 kg/cm(4.8v),2.2 kg/cm(6v)
Sistem kontrol	Pwm (*pulse Width Modulation*)

7. *Push Button*

Saklar Tombol Tekan, atau lebih dikenal dengan sebutan *Push Button Switch*, adalah alat sederhana yang biasanya dipakai dalam rangkaian elektronik untuk menghidupkan atau memutuskan aliran listrik saat ditekan. dalam mode kerja tekan dan buka kunci (tanpa mengunci). Saat dalam mode kerja *unlocked*, saklar berfungsi sebagai alat untuk menghubungkan atau memutuskan aliran listrik ketika tombol ditekan. Ketika tombol tidak ditekan (dilepaskan), saklar akan kembali ke keadaan normal (Siregar, 2017).

Push Button Switch berfungsi sebagai alat penghubung atau pemutus dengan hanya dua kondisi: *On* dan *Off* (1 dan 0). Konsep *On* dan *Off* ini sangat krusial karena hampir semua perangkat listrik yang memerlukan pasokan energi listrik memerlukan pengaturan antara *On* dan *Off*.

Push Button Switch adalah perangkat paling mendasar yang sering digunakan untuk memulai dan menghentikan operasi mesin di berbagai industri, karena cara kerjanya yang *unlock* dan langsung terkait dengan pengguna. Meski seberapa canggihnya suatu mesin, sistem kerjanya tidak akan dapat berfungsi tanpa adanya saklar seperti *Push Button Switch* atau perangkat sejenis yang mengatur kondisi *On* dan *Off* (Siburian, Ringgas, 2024).



Gambar 2. 6 *Push Button*

8. Buzzer

Buzzer adalah sebuah komponen elektronik yang berfungsi mengubah getaran listrik menjadi getaran suara. Pada dasaranya prinsip kerja *buzzer* hampir sama dengan loud speaker, jadi *buzzer* juga terdiri dari kumparan yang terpasang pada diafragma dan kemudian kumparan tersebut dialiri arus sehingga menjadi electromagnet, kumparan tadi akan tertarik ke dalam atau keluar, tergantung dari arah dan polaritas magnetnya, karena kumparan dipasang pada diafragma maka setiap gerakan kumparan akan menggerakan diafragma secara bolak-balik sehingga membuat udara bergetar yang akan menghasilkan suara. Buzzer biasa digunakan sebagai indicator bahwa proses telah selesai atau terjadi suatu kesalahan pada sebuah alat (alarm). Oleh karena itu *buzzer* banyak digunakan sebagai alarm peringatan karena suara yang di keluarkannya sangatlah bising ditelinga (Hamdani, Puspita, & Wildan, 2019).

Buzzer alarm adalah perangkat elektronik yang menghasilkan suara nyaring atau getaran sebagai tanda peringatan atau pemberitahuan. Alat ini biasa digunakan dalam berbagai aplikasi, seperti sistem keamanan, peralatan rumah tangga, kendaraan, atau perangkat industri, untuk memberi sinyal adanya bahaya, kesalahan, atau kondisi tertentu yang memerlukan perhatian. Bunyi yang dihasilkan dapat berupa bip terus-menerus, pola suara intermiten, atau nada khusus tergantung pada fungsinya. Contoh penggunaannya termasuk alarm kebakaran, bel pintu, notifikasi pada mesin ATM, atau peringatan di pabrik. Buzzer alarm dirancang untuk mudah dikenali sehingga dapat segera menarik perhatian dan memicu respons cepat dari pengguna atau sistem (RAMADHAN, 2023).



Gambar 2. 7 Buzzer

B. Kajian Hasil Penelitian Terdahulu

Penelitian terdahulu bertujuan untuk mendapatkan bahan refferensi serta bahan perbandingan dan acuan. Maka penelitian akan mencantumkan beberapa hasil penelitan terdahulu sebagai berikut:

1. Penelitian yang di lakukan oleh (TATANG ADI JULIANTO, 2022) dengan judul penelitian : "RANCANG BANGUN SISTEM PENERANGAN AREA PARKIR OTOMATIS BERBASIS INTERNET OF THINGS (IOT) MENGGUNAKAN *ESP32*" bertujuan untuk memberikan upaya penghematan daya listrik dengan memanfaatkan konsep peredupan dengan menggunakan *modul* AC Light Dimmer. Hal ini membuat penggunaan daya listrik dapat dioptimalkan ketika terdapat pengguna yang membutuhkan sehingga daya listrik dapat dihemat penggunaannya. Lampu menyala disesuaikan dengan keadaan yang terdapat di lapangan didukung dengan sensor cahaya (LDR) dan sensor jarak ultrasonik (HC-SR04). Sensor ultrasonik mendapatkan nilai presisi sebesar 99,57% dan persentase akurasi sensor ultrasonik sebesar 99,26%. Pemantauan data dari daya listrik pada sistem penerangan dapat dilihat melalui aplikasi Blynk yang dikirimkan melalui *ESP32* menggunakan teknologi *Internet* Of *Things.* Dengan metode ini sangat dirasakan manfaatnya untuk menghemat penggunaan

daya listrik sebesar 34 wH dengan beban lampu 10 watt. Kata kunci: *NodeMCU ESP32*, Blynk, Area Parkir, AC Light Dimmer.

2. Penelitian yang di lakukan oleh (Arfilanti Devani, Sektio Nugroho, Wasi'atul Jannah, Inyah Hartami Sarti, 2024) dengan judul penelitian : "PERANCANGAN SISTEM INFORMASI TEMPAT PARKIR BERBASIS APLIKASI WEB DI UNIVERSITAS ISLAM BALITAR" bertuujuan penelitian ini yaitu guna membuat sistem informasi perparkiran berbasis web, mempelajari cara menjalankan sistem, dan mempelajari analisis dan pengujian program untuk perparkiran saat ini di UNISBA. Hasil penelitian menunjukkan bahwa desain sistem informasi perparkiran berbasis web bisa membantu pengemudi mengakses informasi tentang lokasi parkir dari pintu masuk hingga keluar. Implementasi sistem informasi mencakup pengembangan perangkat lunak, perangkat keras, basis data, dan antarmuka untuk aplikasi yang dibuat dengan menggunakan metode *prototyping* model dan menghasilkan sebuah aplikasi e-parkir yang dapat mengelola data kendaraan keluar masuk pada lokasi parker

3. Penelitian yang di lakukan oleh (Moch Farid Fauzi, Tofa Nurcholis, Jeki Kuswanto, Ferian Fauzi Abdulloh, Yusuf Amri Amrulloh, 2022) dengan judul penelitian : "Inovasi Sistem Pembayaran E-Parkir Cashless Dengan Teknologi Hybrid Payment System Berbasis *QRIS*" bertujuan sebagai salah satu jalur pendapatan daerah atas usaha pemerintah daerah dalam menyediakan fasilitas parkir yang ditujukan untuk memenuhi kepentingan masyarakat. Penarikanya retribusi parkir selama ini berjalan kurang efektif dan rentan terjadi kebocoran, ada sebagian oknum petugas parkir yang tidak menyetorkan uang parkir sesuai

dengan kenyataanya, sehingga merugikan potensi pendapatan daerah. Research and Development merupakan metode penelitian yang dipergunakan dalam penelitian ini, sedangkan metode pengembangannya menggunakan prototyp. User Acceptance Test juga dilaksanakan untuk mengukur seberapa tinggi penerimaan terhadap sistem oleh para pengguna aplikasi tersebut. Pengujian dilaksanakan secara komprehensif dan mendapatkan prosentase sebesar 87%, yang menandakan bahwa sistem E-Parking terbukti bisa diterima dan dipergunakan dengan baik oleh pengguna

BAB III

METODE PENELITIAN

A. Jenis Penelitian

Metode yang digunakan pada penelitian ini adalah Research and Development Penelitian ini bertujuan untuk merancang, membuat, dan menguji suatu sistem baru (yaitu sistem parkir elektronik berbasis *IoT* dengan identifikasi menggunakan kartu SIM). Fokus utamanya adalah pada proses pengembangan teknologi atau sistem, bukan sekadar pengujian hipotesis atau studi fenomena.

B. Lokasi dan Waktu

1. Lokasi

Lokasi pelaksanaan penelitian akan dilaksanakan di laboratorium Teknik Elektro kampus II Universitas Muhammadiyah Parepare, dikarenakan adanya alat serta bahan yang di butuhkan untuk pemasangan dan pengujian alat.

2. Waktu

Tabel 3. 1 Jadwal Pelaksanaan Penelitian

NO	Uraian	Bulan					
		1	2	3	4	5	6
1	Studi Literatur						
2	Perancangan Perangkat Keras						
3	Perancangan Perangkat Lunak						
4	Perancangan Mekanik						
5	Realisasi Sistem Mekanik						
6	Realisasi Perangkat Keras Dan Perangkat Lunak						
7	Pengujian Tahap Awal (Komponen)						
8	Integrasi Rangkaian Ke Mekanik						
9	Pengujian Tahap Ke Dua (Blok Diagram)						
10	Pengujian Sistem Secara Keseluruhan						
11	Analisis Dan Evaluasi						
12	Penyusunan Laporan						

C. Alat Dan Bahan

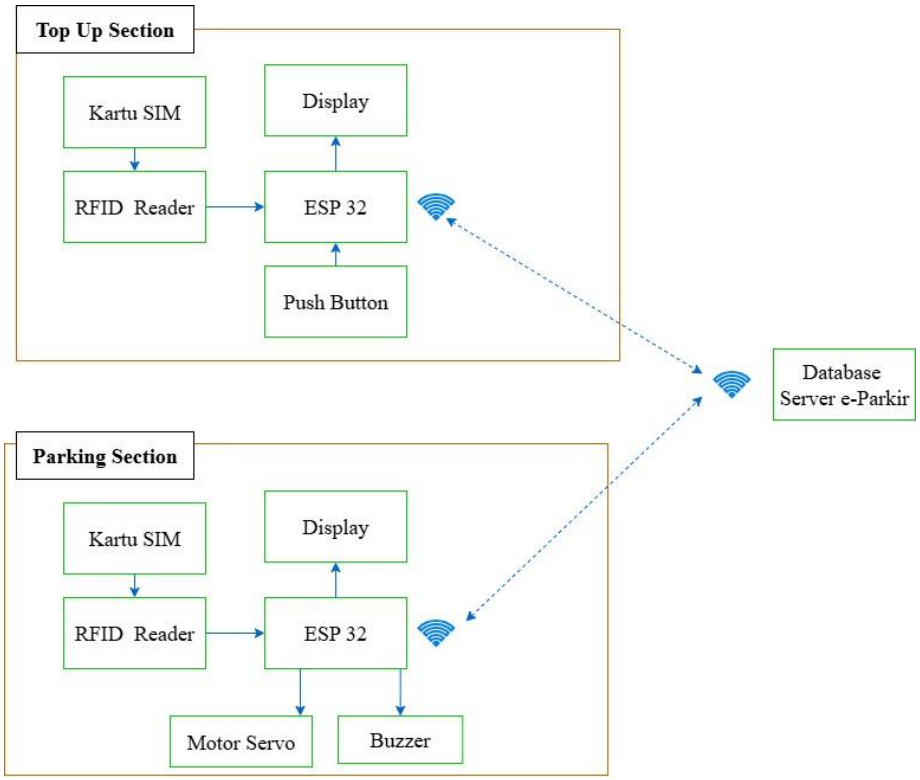
Alat dan bahan yang dibutuhkan dalam pembuatan alat dapat dilihat pada Tabel 3. 2 berikut ini:

Tabel 3. 2 Alat Dan Bahan

No	Uraian	Jumlah
1	ESP 32	2
2	RFID Reader	2
3	LCD 20x4	2
4	I2C Driver LCD	2
5	Motor Servo	1
6	Buzzer	1
7	Push Button	1
8	Stiker RFID	3

D. Rancangan Sistem

Rancangan sistem pada penelitian ini disajikan dalam bentuk blok diagram yang ditunjukkan pada gambar dibawah ini:



Gambar 3. 1 Blok Diagram

Pada gambar di atas merupakan blok diagram Sistem e-Parkir Berbasis Iot Menggunakan Surat Izin Mengemudi (SIM). Adapun komponen penyusun terdiri dari :

A. Top Up Section

Top Up Section berfungsi sebagai bagian pengisian e-Money pada kartu SIM.

- *RFID* Reader untuk membaca identitas kartu.
- *ESP32* untuk koneksi ke *database server* yang berisi data kepemilikan kartu dan isi *saldo* terakhir dan nilai *saldo* yang akan ditopup.

- Display untuk menampilkan identitas kartu, *saldo* terakhir, dan pilihan nilai *saldo* yang akan di top up.
- Push Button sebagai tombol pilihan nilai *saldo*.

B. Parking Section

- *RFID* Reader untuk membaca identitas kartu
- *ESP32* untuk koneksi ke *database server,* mengecek kartu terdaftar atau tidak, mengecek kecukupan *saldo* dan memerintahkan servo untuk membuka/menutup palang parkir.
- Display untuk menampilkan identitas kartu dan *saldo* terakhir setelah pemotongan jasa parkir.
- Buzzer untuk indikator suara pada saat kartu di tag.

E. Teknik Pengumpulan Data

Teknik pengumpulan data yang dilakukan pada penelitian ini terdiri dari beberapa tahapan, yaitu :

1. *Study literature,* tahapan ini dilakukan untuk memahami konsep dasar sistem yang dirancang meliputi *hardware* dan *software.*
2. *Pengumpulan alat dan bahan yang dibutuhkan.*
3. *Perakitan.*
4. *Pengujian masing-masing komponen/modul*
5. *Finalisasi* system dengan uji keseluruhan system.

BAB IV

HASIL DAN PEMBAHASAN

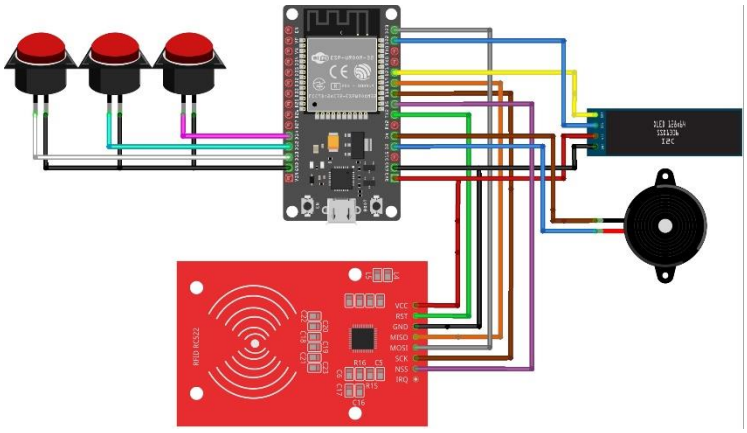
A. Perancangan Perangkat Keras

1. Rangkaian Topup

Pada parancangan sistem perangkat keras, dilakukan analisis terhadap kebutuhan teknis dan fungsional yang diperlukan untuk memasikan prangkat yang di kembangkan, dapat mendukung kinerja sistem secara opimal, dengan mempertimbangkan aspek efesiensi, serta kemampuan untuk beradaptasi dengan prangkat lunak yang ada, rancangan ini menjelaskan tentang sistem rangkain yang dibuat sistem per bagian dari sistem rangkain topup dan rangkain e-parkir.

Perancangan perangkat keras pada rangkaian sistem topup e-parkir berbasis *IoT* mengunakan kartu Surat Izin Mengemudi (SIM) memerlukan parangkat eletronika dengan mikrokntroler *ESP32* sebagain kontrol inti yang mengatur komunikasi antara perangkat dan menghubungkan sistem. Pada proses identifikasi mengguna . Yang mengunakan *modul RFID RC522.* Modul ini membaca data dari kartu sim berbasis *RFID* kemudian akan di proses untuk megakses informasi dan *saldo* pengguna. Tampilan data dilakukan melelui *modul* oled yang masing-masing terhunung dengan 12c, sehingan memudahkan bagi pengguna dalam menempilkan status *saldo* dan topup secara langsung. Selain itu ,ada tiga tobol input yang dinggunakan sebagai antarmuka pengguna untuk memili nilai *top-up* , menkonfirmasi teransaksi pegisian *saldo* dan membatalkannya. Buzzer aktif dingunakan untuk membelikan suara ketika tombol ditekan. Seluruh komunikasi

komponen dipasang pada pin GPIO *ESP32* sesuai fungsinya, dan daya diperoleh dari port USB mikrokontroler *ESP32*. Dengan sistem ini, sistem dapat melakukan proses *top-up saldo* parkir secara otomatis, cepat, dan terintegrasi dengan sistem *IoT* secara real time



Gambar 4. 1 Rangkaian Topup

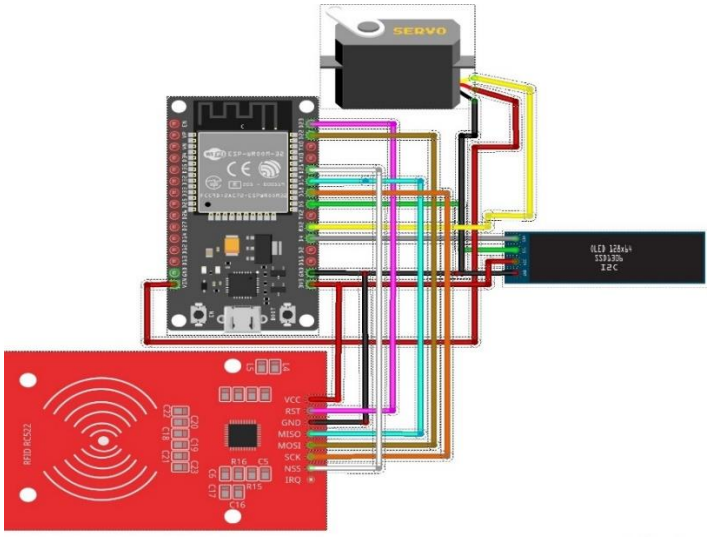
2. Rangakain E-parkir

Perancangan perangkat keras untuk sistem akses e-parkir elektronik memanfaatkan teknologi *Internet* of *Things (IoT),* yang dipadukan dengan *modul RFID* dan motor servo untuk mengendalikan palang parkir otomatis. Sistem ini dikendalikan oleh mikrokontroler *ESP32* yang bertindak sebagai pusat kendali. Identifikasi pengguna dengan *modul RFID RC522* yang membaca data dari kartu SIM pengguna yang terintegrasi dengan chip *RFID*. Komunikasi antara *RFID* dan *ESP32* dilakukan menggunakan protokol *SPI* (Serial Peripheral Interface) dengan koneksi pin: SDA, SCK, MOSI, MISO, dan RST.

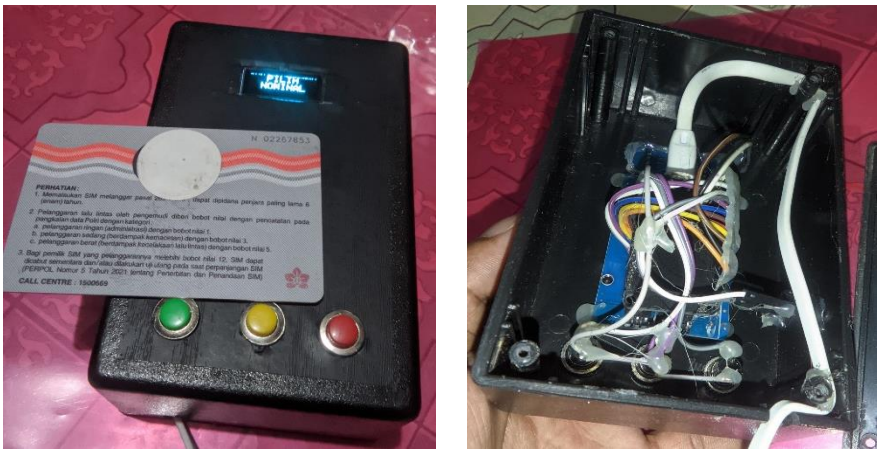
ESP32 mengenali dan memvalidasi data pengguna, lalu mengirimkan perintah ke motor servo *MG996R* untuk membuka palang parkir secara otomatis. Jika kartu *saldo* tidak mencukupi biaya parkir, maka servo tidak akan bergerak ke

posisi terbuka dan sebaliknya jika kartu *saldo* mencukupi biaya parkir maka servo akan bergerak ke posisi terbuka. Pesan status sistem ini akan ditampilkan di layar *OLED* 128x32 melalui komunikasi *I2C* dengan pin SDA dan SCL, yang menunjukkan "Kartu Dikenali", "Topup dulu", atau "Saldo terakhir".

ESP32 berfungsi sebagai unit pusat untuk distribusi daya dan kontrol ke semua komponen berdasarkan konfigurasi GPIO. Dengan integrasi ini, pengguna hanya perlu menempelkan kartu SIM di dekat box.



Gambar 4. 2 Rangkaian E-parkir



Gambar 4. 3 Realisasi Rangkaian Topup



Gambar 4. 4 Realisasi Rangkaian E-parkir

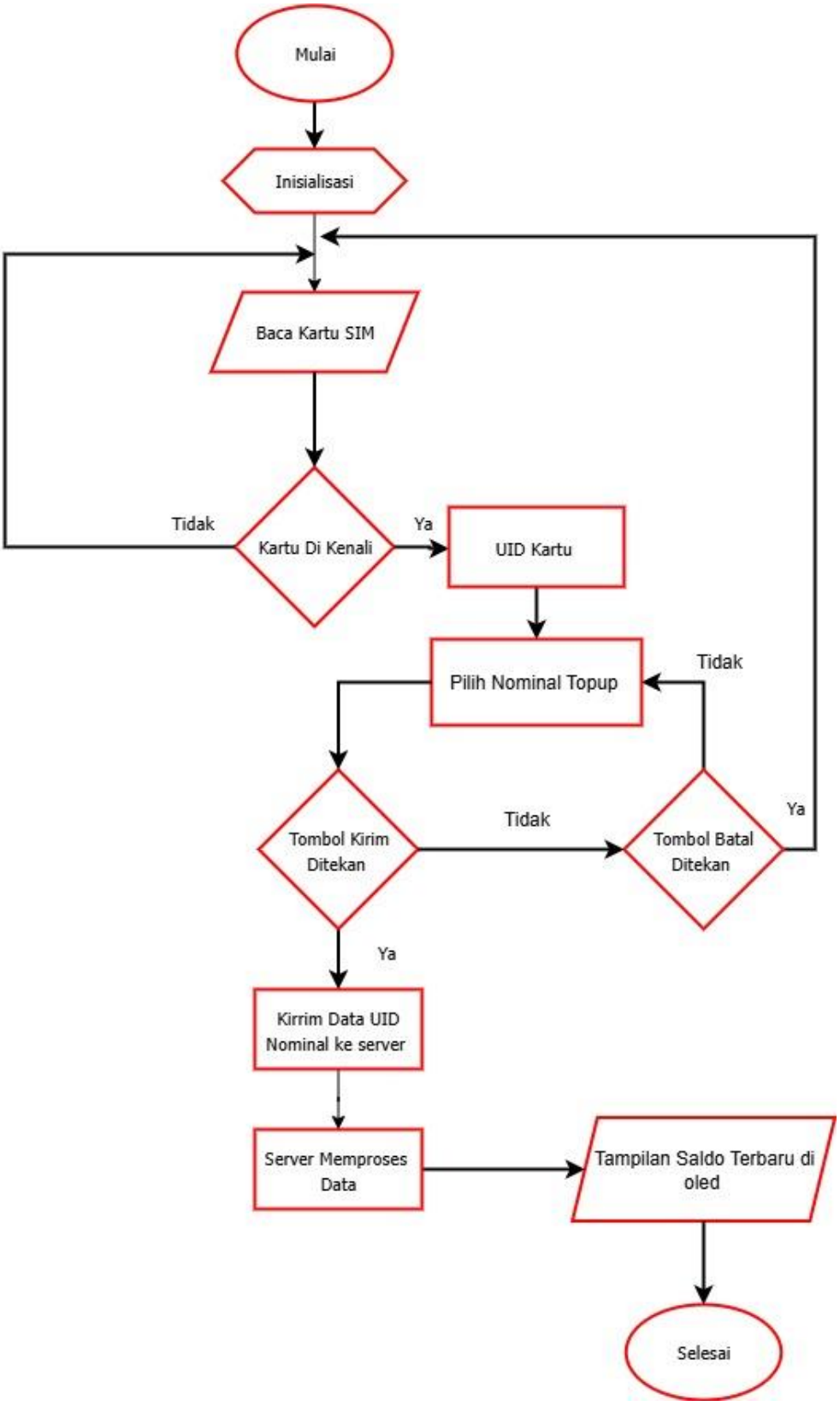
B. Perancangan Perangkat Lunak

Perancangan perangkat lunak pada Sistem E-Parkir berbasis *IoT* menggunakan kartu SIM sangat penting karena mengendalikan semua fungsi perangkat keras yang dirancang. Perangkat lunak ini dikembangkan menggunakan *ESP32* sebagai otak sistem.

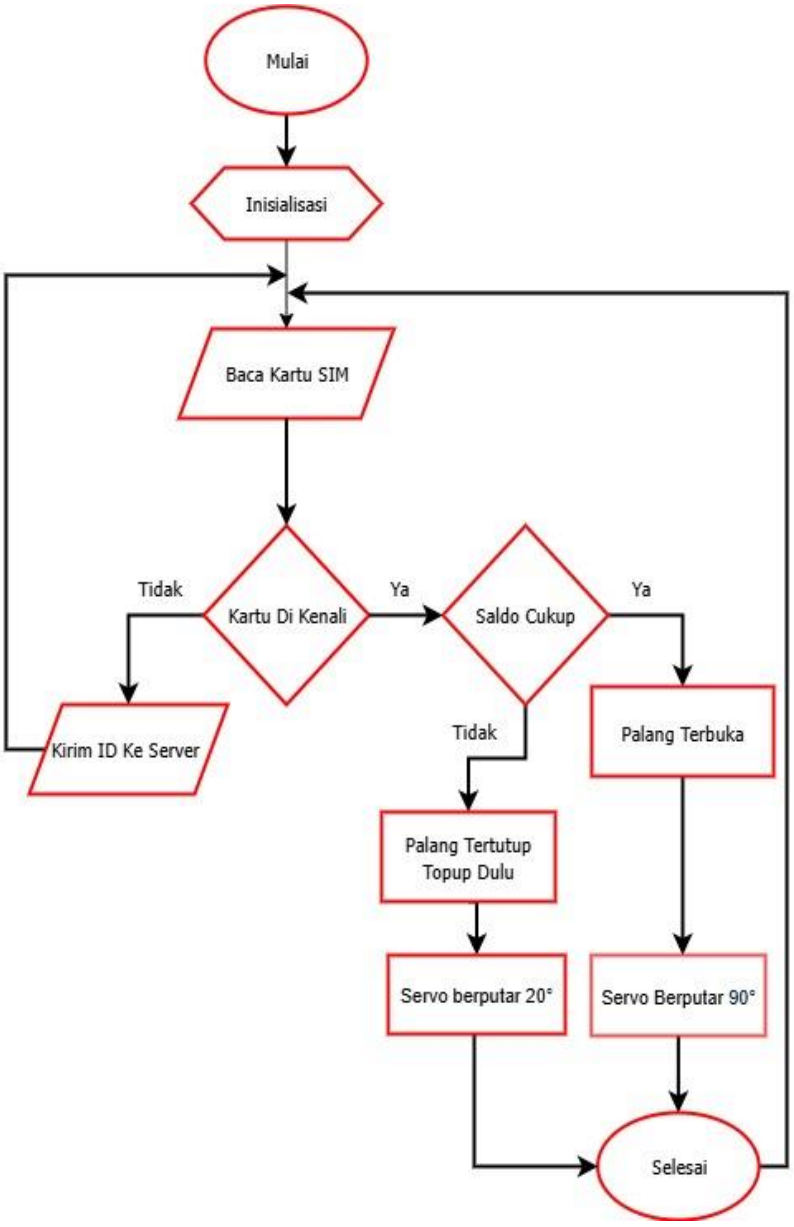
Perangkat lunak sistem ini dibagi menjadi dua yaitu *flowchart* dan uraian program. Tujuan utama perancangan, ini adalah untuk menangani seluruh proses kerja sistem, mulai dari membaca kartu *RFID* hingga memproses data , serta mengendalikan aktuator berupa motor servo Struktur perangkat lunak terbagi menjadi dua bagian utama, yaitu program untuk sistem *top-up* dan program untuk sistem e-parkir. Keduanya saling terintegrasi melalui komunikasi jaringan *Wi-Fi* dan *database server* lokal menggunakan *XAMPP* sambil menampilkan informasi secara real-time ke layar *OLED I2C.*

1. Flowchart

Flowchart adalah grafik yang digunakan untuk menggambarkan alur kerja atau proses dalam bentuk simbol-simbol grafis untuk menunjukkan Langkah-langkah aliran kontrol yang berperan dalam proses. Flowchart memudahkan penerimaan pengertian suatu sistem dalam cara yang lebih visual dan terstruktur. Diagram alir sistem kerja di tunjukkan pada gambar 4.5 dan gambar 4.6.



Gambar 4. 5 Flowchart Topup



Gambar 4. 6 Flowchart E-Parkir

2. Uraian Program

Adapun sketch program yang di gunakan untuk menjalankan fungsi dari sistem yang dibuat sesuai dengan *flowchart* yaitu:

a. Program Topup

```
//-------------------------------------
#include <WiFi.h>
#include <HTTPClient.h>
const char* ssid = "V2026"; // ganti dengan ssid wifi yg digunakan
const char* password = "2221235678"; // ganti pasword

const char* url = "http://192.168.166.230/eParkir/topup.php";
// 192.168.1.64 diganti dengan ipv4 wifi
WiFiClient client;
HTTPClient http;
//-------------------------------------
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 5
 #define RST_PIN 17
MFRC522 rfid(SS_PIN, RST_PIN);
String id;
String staBaca;
```

Pada baris pertama, library WiFi.h dan *HTTPClient.h* digunakan untuk menghubungkan *ESP32* ke jaringan nirkabel dan melakukan komunikasi *HTTP* dengan *server.* Variabel ssid dan password diisi dengan nama serta kata sandi WiFi yang digunakan agar perangkat dapat terkoneksi. Alamat URL http://192.168.166.230/eParkir/topup.php merupakan *endpoint pada server* lokal *(XAMPP* atau hosting) yang digunakan untuk *proses top-up saldo* kartu *RFID.* Selanjutnya, library *SPI.h* dan *MFRC522.h* digunakan untuk berkomunikasi dengan *modul RFID* tipe *MFRC522. Pin* SS_PIN diatur pada pin GPIO 5 dan RST_PIN pada GPIO 17, yang merupakan jalur komunikasi antara *ESP32 dan modul RFID.* Objek rfid dibuat untuk menangani fungsi-fungsi pembacaan kartu. Variabel id disiapkan untuk menyimpan *data UID* kartu *RFID yang* terbaca, sedangkan variabel staBaca kemungkinan digunakan untuk menandai status pembacaan kartu. Dengan konfigurasi ini, *ESP32* dapat membaca ID kartu *RFID,* lalu mengirimkan data tersebut ke *server* melalui protokol *HTTP* untuk keperluan *top-up saldo*.

```
//-----------------------------------
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels
#define OLED_RESET -1 // Reset pin # (or -1 if sharing Arduino
reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D
for 128x64, 0x3C for 128x32
Adafruit_SSD1306  display(SCREEN_WIDTH,  SCREEN_HEIGHT,  &Wire,

OLED_RESET);
```

Library Wire.h digunakan sebagai *driver* komunikasi *I2C*, sedangkan Adafruit_GFX.h berfungsi sebagai library grafis yang menyediakan fungsi dasar seperti menggambar teks, garis, maupun bentuk. Library Adafruit_SSD1306.h khusus digunakan untuk mengontrol *OLED SSD1306.*

Parameter SCREEN_WIDTH dan SCREEN_HEIGHT masing-masing diatur sebesar 128 dan 32 untuk menyesuaikan ukuran layar *OLED yang* digunakan. *Pin reset OLED (OLED_RESET)* diset -1 karena *OLED* ini berbagi jalur reset dengan mikrokontroler, sehingga tidak memerlukan pin khusus. *Alamat I2C* dari layar diatur pada 0x3C, yang merupakan alamat default untuk *modul OLED* berukuran 128x32 (sedangkan untuk *OLED 128x64* biasanya 0x3D).

Objek *display* kemudian dibuat menggunakan konstruktor Adafruit_SSD1306, dengan parameter resolusi layar, komunikasi *I2C (&Wire), dan* konfigurasi pin reset. Dengan konfigurasi ini, programmer dapat menampilkan teks, grafik, maupun informasi lain di layar *OLED* sesuai kebutuhan aplikasi, misalnya menampilkan status *saldo* pada sistem e-Parkir.

```
//-----------------------------------
#define LBz 2
#define HBz 4
int i=0;
```

```
String pilNom[] = {"20.000","30.000","50.000","100.000"};
String saldo="";
```

Baris #define LBz 2 dan #define HBz 4 digunakan untuk mendeklarasikan konstanta berupa nomor pin pada mikrokontroler, yang dalam hal ini kemungkinan besar terhubung ke *buzzer* sebagai indikator suara. Dengan cara ini, pemanggilan pin menjadi lebih mudah karena menggunakan nama yang lebih bermakna daripada angka langsung. Selanjutnya, int i = 0; merupakan variabel penghitung (counter) atau indeks yang biasanya dipakai untuk menelusuri data dalam array atau mengatur pilihan nominal. Kemudian, String pilNom[] = {"20.000","30.000","50.000","100.000"}; adalah sebuah array string yang berisi daftar pilihan nominal *saldo* yang dapat dipilih pengguna, yaitu Rp20.000, Rp30.000, Rp50.000, dan Rp100.000. Terakhir, String *saldo* = ""; digunakan sebagai variabel untuk menyimpan nilai *saldo* awal pengguna yang pada saat program pertama kali dijalankan masih kosong. Struktur ini memudahkan sistem dalam mengelola pilihan *top-up* sekaligus menyimpan informasi *saldo* yang akan diproses lebih lanjut.

```
void setup() {
 Serial.begin(9600);
 pinMode(16, OUTPUT); digitalWrite(16, HIGH); // Vcc rfid
 pinMode(15, OUTPUT); digitalWrite(15, LOW); // GND rfid

 pinMode(13, INPUT_PULLUP); // tombol nominal
 pinMode(12, INPUT_PULLUP); // tombol kirim
 pinMode(14, INPUT_PULLUP); // tombol batal

 pinMode(LBz, OUTPUT);
 pinMode(HBz, OUTPUT);
 digitalWrite(LBz, LOW);
```

Fungsi setup() digunakan untuk melakukan inisialisasi awal perangkat keras sebelum sistem berjalan. Pertama, komunikasi serial diaktifkan dengan Serial.begin(9600); agar mikrokontroler dapat mengirim dan menerima data

melalui monitor serial. Selanjutnya, pin 16 dikonfigurasi sebagai output dan diberi logika HIGH untuk menyuplai tegangan VCC ke *modul RFID,* sementara pin 15 juga diatur sebagai output dengan logika LOW sehingga berfungsi sebagai ground bagi *modul* tersebut. Kemudian, tiga buah tombol input dikonfigurasi menggunakan mode INPUT_PULLUP, masing-masing pada pin 13 sebagai tombol pemilihan *nominal top-up, pin 12* sebagai tombol konfirmasi atau kirim, serta pin 14 sebagai tombol batal. Setelah itu, *pin buzzer yang* telah didefinisikan sebelumnya melalui LBz dan HBz diatur sebagai *output, dan buzzer* awalnya diset dalam kondisi LOW dengan perintah digitalWrite(LBz, LOW); agar tidak aktif saat sistem pertama kali dinyalakan. Dengan inisialisasi ini, perangkat keras utama sistem *top-up,* yaitu *RFID,* tombol*, dan buzzer,* siap beroperasi sesuai fungsinya.

```
SPI.begin();
 rfid.PCD_Init();
 //------------------------------
 if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS))
 {
 Serial.println(F("SSD1306 allocation failed"));
 for(;;);
 }
 display.clearDisplay();
 display.setTextSize(2);
 display.setTextColor(SSD1306_WHITE);
```

SPI.begin(); dipanggil untuk memulai komunikasi menggunakan protokol *SPI yang* diperlukan oleh *modul RFID agar* dapat berkomunikasi dengan mikrokontroler. Selanjutnya, rfid.PCD_Init(); digunakan untuk menginisialisasi *modul RFID MFRC522* sehingga siap melakukan proses pembacaan kartu. Setelah itu, dilakukan inisialisasi pada *modul OLED* dengan perintah *display.begin*(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS). Apabila

inisialisasi *OLED* gagal, sistem akan menampilkan pesan kesalahan *"SSD1306 allocation failed"* pada serial monitor, kemudian masuk ke dalam perulangan for(;;); yang menghentikan seluruh *proses program*. *Jika OLED* berhasil terhubung, maka layar akan dikosongkan terlebih dahulu menggunakan *display.clearDisplay*();, kemudian ukuran teks diatur menjadi lebih besar melalui *display.setTextSize*(2); agar tampilan informasi lebih jelas, dan warna teks diatur menjadi putih menggunakan *display.setTextColor*(SSD1306_WHITE); . Dengan konfigurasi ini, *modul RFID* sudah siap membaca kartu, sementara *OLED* dapat menampilkan informasi kepada pengguna dengan tampilan yang lebih terbaca.

```
void standby(void)
{
 i=0; id="";
 staBaca="standby";
 display.clearDisplay();
 display.setCursor(45, 0); display.print("SCAN");
 display.setCursor(20, 18); display.print("SIM ANDA");
 display.display();
}
```

Fungsi standby() digunakan untuk mengatur kondisi sistem kembali ke mode siaga setelah proses inisialisasi selesai atau setelah suatu transaksi berakhir. Pada fungsi ini, variabel penghitung i direset ke nilai awal 0, kemudian variabel id dikosongkan, dan variabel status staBaca diisi dengan string "standby" sebagai penanda bahwa sistem sedang menunggu interaksi pengguna. Selanjutnya, layar *OLED* dibersihkan dengan perintah *display.clearDisplay*(); agar tampilan sebelumnya dihapus. Kemudian, posisi kursor diatur dengan *display.setCursor*(45, 0); untuk menuliskan teks "SCAN" di bagian atas layar, dan *display.setCursor*(20,

18); untuk menampilkan teks “SIM ANDA” di bagian bawah. Perintah *display.display*(); dipanggil untuk menampilkan perubahan tersebut ke layar *OLED.* Dengan demikian, fungsi ini berperan penting sebagai tampilan awal atau layar utama yang memberikan instruksi kepada pengguna untuk menempelkan kartu *RFID* sebelum sistem dapat memproses langkah selanjutnya.

```
void bacaRFID()
{
 if(rfid.PICC_IsNewCardPresent() && rfid.PICC_ReadCardSerial())
 {
 id="";
 for (byte i = 0; i < rfid.uid.size; i++)
 {
 id=id+String(rfid.uid.uidByte[i],HEX);
 }
 id.toUpperCase();
 rfid.PICC_HaltA();
 rfid.PCD_StopCrypto1();
 //buzzer-----------------------------
 tone(HBz, 1500, 100);
 //-----------------------------------
 display.clearDisplay();
 display.setCursor(35, 0); display.print("PILIH");
 display.setCursor(25, 18); display.print("NOMINAL");
 display.display();
 }
}
```

Fungsi *bacaRFID*() bertugas untuk membaca data dari kartu *RFID yang* ditempelkan pada *modul*. Pertama, program melakukan pengecekan dengan rfid.PICC_IsNewCardPresent() dan rfid.PICC_ReadCardSerial() untuk memastikan ada kartu baru yang terdeteksi dan data serialnya dapat dibaca. Jika kondisi tersebut terpenuhi, variabel id dikosongkan terlebih dahulu, kemudian dilakukan perulangan for untuk mengambil setiap byte dari *UID (Unique Identifier)* kartu menggunakan rfid.uid.uidByte[i]. *Nilai UID* tersebut dikonversi menjadi string dalam format heksadesimal dan disimpan ke dalam variabel id. Selanjutnya,

perintah id.toUpperCase(); digunakan agar seluruh karakter *UID* ditampilkan dalam huruf kapital, sehingga lebih konsisten dalam penyimpanan maupun perbandingan dengan *database.* Setelah proses pembacaan selesai, kartu dihentikan dengan rfid.PICC_HaltA(); dan enkripsi komunikasi dihentikan dengan rfid.PCD_StopCrypto1();. Sebagai indikator bahwa kartu berhasil terbaca, sistem mengaktifkan *buzzer* dengan tone(HBz, 1500, 100); yang menghasilkan bunyi singkat. Terakhir, layar *OLED* diperbarui untuk menampilkan instruksi kepada pengguna berupa tulisan “PILIH” pada bagian atas dan “NOMINAL” di bagian bawah, sehingga pengguna diarahkan untuk memilih jumlah *saldo* yang akan ditambahkan. Dengan demikian, fungsi ini menjadi penghubung awal antara kartu *RFID dan* sistem *top-up*, sekaligus memberikan umpan balik visual dan audio kepada pengguna.

```
void kirim_data(String dt)
{
 String postData = (String)"nilai=" + dt;
 HTTPClient http;
 http.begin(url);
 http.addHeader("Content-Type","application/x-www-form
urlencoded");
 auto httpCode = http.POST(postData);
 //String payload = http.getString();
 saldo = http.getString();
 display.clearDisplay();
 display.setCursor(0, 0); display.print("SALDO ANDA");
 display.setCursor(0, 18); display.println(saldo);
 display.display();
 delay(3000);
 standby();
 //Serial.println(postData);
 //Serial.println(saldo);
 http.end();
}
```

Fungsi kirim_data(String dt) digunakan untuk mengirimkan data dari mikrokontroler ke *server* melalui protokol *HTTP*. *Data yang* akan dikirim disimpan

pada variabel postData dengan format nilai=dt, di mana dt merupakan parameter yang berisi nominal *saldo* atau data lain yang akan diproses. Selanjutnya, objek *HTTPClient* http digunakan untuk membuka koneksi ke *server* dengan perintah http.begin(url);, kemudian ditambahkan header Content-Type bertipe *application/x-www-form-urlencoded* agar data dapat diterima dengan benar *oleh server*. *Proses* pengiriman dilakukan dengan http.POST(postData);, sementara respons dari *server* dibaca menggunakan http.getString(); dan hasilnya disimpan ke dalam variabel *saldo*. Setelah itu, layar *OLED* diperbarui untuk menampilkan pesan "SALDO ANDA" diikuti dengan nilai *saldo* terbaru yang diperoleh dari *server,* sehingga pengguna dapat mengetahui jumlah *saldo* setelah *proses top-up.* Tampilan ini ditahan selama 3 detik menggunakan delay(3000); sebelum sistem kembali ke kondisi siaga dengan memanggil fungsi standby();. Pada akhir fungsi, koneksi *HTTP* ditutup dengan http.end(); agar komunikasi jaringan lebih efisien. Dengan demikian, fungsi ini menjadi penghubung antara perangkat keras sistem dengan *server,* sekaligus memberikan umpan balik langsung kepada pengguna mengenai *saldo* yang dimilikinya.

```
void loop()
{
 if (staBaca=="standby") {
 bacaRFID();
 }
 if (id != "") {
 pilihNominal();
 }
}

void pilihNominal(void)
{
 if (digitalRead(13)==LOW)
 {
 display.clearDisplay();
 display.setCursor(0, 0); display.print("TOP-UP SIM");
 display.setCursor(0, 18); display.print(pilNom[i]);
```

```
display.display();
i=i+1;
if (i>4) {
i=0;
}
```

fungsi loop(), program utama berjalan dengan logika sederhana namun terstruktur. Pertama, sistem akan memeriksa apakah status staBaca berada dalam kondisi "standby". Jika ya, maka fungsi *bacaRFID*() dipanggil untuk mendeteksi apakah ada kartu *RFID yang* ditempelkan. Selanjutnya, apabila variabel id tidak kosong atau telah berisi *UID* dari kartu yang terbaca, maka sistem akan memanggil fungsi pilihNominal() untuk memberikan opsi nominal *saldo* yang bisa dipilih pengguna. Dengan demikian, fungsi loop() menjadi pusat kendali alur sistem yang terus-menerus menunggu input pengguna.

Fungsi pilihNominal() bertugas menampilkan pilihan nominal *saldo* pada layar *OLED* berdasarkan input tombol. Dalam fungsi ini, kondisi if (digitalRead(13)==LOW) digunakan untuk mendeteksi apakah tombol pemilih nominal ditekan. Jika benar, maka layar *OLED* akan dibersihkan dengan *display.clearDisplay*();, kemudian ditampilkan teks “TOP-UP SIM” pada baris pertama, dan salah satu nilai nominal dari array pilNom[i] pada baris berikutnya. Setelah nominal ditampilkan, variabel i akan ditambah satu agar pada penekanan tombol berikutnya sistem menampilkan nominal selanjutnya. Untuk menjaga agar indeks tidak keluar dari batas array, terdapat kondisi if (i>4) yang mengembalikan nilai i menjadi 0, sehingga daftar pilihan nominal akan berulang kembali dari awal. Dengan cara ini, pengguna dapat menelusuri berbagai *nominal top-up* secara bergilir hanya dengan menekan satu tombol.

```
tone(HBz, 500, 100);
```

```
 } else
 if (digitalRead(12)==LOW)
 {
 tone(HBz, 1000, 100);
 display.clearDisplay();
 display.setCursor(0, 0); display.print("KIRIM DATA");
 display.setCursor(0, 18); display.print(" ..... ");
 display.display();
 String dtKirim=id+"#"+pilNom[i-1];
 kirim_data(dtKirim);

 } else
 if (digitalRead(14)==LOW)
 {
 standby();
 tone(HBz, 1500, 100);
 }
 delay(250);
}
```

Pada bagian akhir fungsi pilihNominal(), sistem mengatur respon terhadap tombol konfirmasi maupun pembatalan transaksi. Jika tombol konfirmasi yang terhubung pada pin 12 ditekan, *buzzer* akan berbunyi singkat dengan frekuensi 1000 Hz sebagai indikator penerimaan input. Setelah itu, layar *OLED* dibersihkan dan menampilkan pesan "KIRIM DATA" beserta indikator **"....."** yang menandakan bahwa data sedang diproses. Data yang dikirim dirangkai dalam variabel dtKirim dengan format gabungan antara *UID* kartu *RFID (*id) dan nominal *saldo* yang dipilih (pilNom[i-1]). Variabel ini kemudian diteruskan ke *server* melalui pemanggilan fungsi kirim_data(dtKirim); untuk memperbarui *saldo*. Sebaliknya, jika tombol batal pada pin 14 ditekan, maka sistem segera memanggil fungsi standby(); untuk kembali ke kondisi siaga, *dan buzzer* akan berbunyi dengan frekuensi 1500 Hz sebagai tanda pembatalan. Untuk menghindari pembacaan tombol berulang akibat *bouncing* atau penekanan cepat, fungsi ini menambahkan delay(250); sebagai jeda sebelum melanjutkan proses berikutnya. Dengan mekanisme ini, pengguna memiliki kendali penuh apakah akan melanjutkan *proses*

top-up atau membatalkannya, sementara sistem tetap memberikan umpan balik berupa tampilan *visual pada OLED dan* indikator suara dari *buzzer.*

b. Program e-parkir

```
#include <SPI.h>
#include <Wire.h>
#include <MFRC522.h>
#include <ESP32Servo.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

#include <*SPI.h*> digunakan untuk mengaktifkan protokol komunikasi *SPI yang* dibutuhkan *modul RFID.* Selanjutnya, #include <Wire.h> digunakan untuk komunikasi I²C, misalnya pada *modul OLED. Pustaka #include <MFRC522.h>* dipakai untuk mengendalikan *modul RFID MFRC522* sehingga dapat membaca *UID* kartu. Library *#include <ESP32Servo.h>* berfungsi untuk mengontrol motor servo menggunakan mikrokontroler *ESP32.* Kemudian, #include <WiFi.h> dan #include <*HTTPClient.h*> dipakai untuk menghubungkan *ESP32* dengan jaringan WiFi serta melakukan komunikasi data dengan *server* melalui protokol *HTTP.* Terakhir, #include <Adafruit_GFX.h> dan #include <Adafruit_SSD1306.h> digunakan untuk mengontrol tampilan *OLED,* baik dari segi grafis maupun teks, sehingga sistem dapat menampilkan informasi secara visual kepada pengguna. Dengan kombinasi library tersebut, program mampu menjalankan berbagai fungsi penting mulai dari pembacaan kartu *RFID,* koneksi ke *server,* hingga penyajian informasi melalui layar *OLED dan* aktuator.

```
// WiFi
const char* ssid = "V2026";
const char* password = "2221235678";
```

```
const char* urlGetSaldo =
"http://192.168.166.230/eParkir/cek_saldo.php";
const char* urlUpdateSaldo =
"http://192.168.166.230/eParkir/update_saldo.php";
```

Variabel const char* ssid = "V2026"; dan const char* password = "2221235678"; berfungsi untuk menyimpan nama jaringan WiFi (SSID) serta kata sandinya, sehingga perangkat *ESP32* dapat terhubung ke jaringan lokal. Selanjutnya, terdapat dua alamat URL yang didefinisikan, yaitu urlGetSaldo dan urlUpdateSaldo. URL *http://192.168.166.230/eParkir/cek_saldo.php* digunakan untuk mengambil informasi *saldo* terkini dari *server* ketika kartu *RFID berhasil dibaca*, sedangkan http://192.168.166.230/eParkir/update_saldo.php digu nakan untuk memperbarui *saldo* setelah proses transaksi dilakukan. Dengan adanya konfigurasi ini, perangkat dapat melakukan komunikasi dua arah dengan *server* melalui protokol *HTTP*, yaitu membaca *saldo* yang tersimpan dalam *database* serta mengirimkan data pembaruan *saldo* setelah transaksi *top-up* maupun pemotongan biaya parkir.

```
// OLED
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
&Wire, OLED_RESET);
```

Nilai #define SCREEN_WIDTH 128 dan #define SCREEN_HEIGHT 32 mendefinisikan resolusi layar *OLED* dengan lebar 128 piksel dan tinggi 32 piksel. Konstanta *OLED_RESET* diberi nilai -1, yang berarti *modul OLED* tidak menggunakan pin reset khusus, melainkan memanfaatkan reset bawaan dari mikrokontroler. Selanjutnya, dibuat objek Adafruit_SSD1306 *display(SCREEN_*

WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET); yang menginisialisasi *modul OLED* dengan library Adafruit SSD1306 menggunakan komunikasi I²C melalui interface Wire. Dengan konfigurasi ini, *OLED* dapat digunakan untuk menampilkan berbagai teks maupun grafis sederhana, seperti instruksi, informasi *saldo*, atau status koneksi, sehingga pengguna mendapatkan umpan balik visual secara langsung dari sistem.

```
// RFID
#define RFID_SS_PIN 21
#define RFID_RST_PIN 22
MFRC522 rfid(RFID_SS_PIN, RFID_RST_PIN);

// Servo
#define SERVO_PIN 16
Servo palangServo;
```

Baris *#define RFID_SS_PIN 21* dan *#define RFID_RST_PIN 22* mendefinisikan *pin pada ESP32 yang* digunakan untuk komunikasi dengan *modul RFID MFRC522, di mana pin* SS **(**Slave Select) terhubung ke pin 21 dan pin RST (Reset) terhubung ke pin 22. Selanjutnya dibuat objek *MFRC522* rfid(*RFID_SS_PIN, RFID_RST_PIN);* yang berfungsi sebagai pengendali *modul RFID* untuk membaca ID kartu atau tag yang ditempelkan.

#define SERVO_PIN 16 digunakan untuk menentukan *pin GPIO 16 pada ESP32* sebagai jalur kendali sinyal *PWM* untuk motor servo. Kemudian dibuat objek Servo palangServo; dengan bantuan library *ESP32Servo.h*, yang akan digunakan untuk mengatur sudut putaran servo sehingga palang parkir dapat bergerak membuka atau menutup secara otomatis sesuai logika program. Dengan konfigurasi ini, sistem dapat membaca kartu *RFID* pengguna dan mengatur gerakan palang servo berdasarkan status *saldo* yang terdeteksi.

```
// Waktu
unsigned long waktuTerakhirScan = 0;
const unsigned long jedaScan = 3000;

// Status Masuk
bool statusMasuk = false;
String lastUID = "";
```

bool statusMasuk = false; digunakan sebagai penanda kondisi kendaraan, apakah sedang masuk atau tidak**.** Nilai default false menunjukkan bahwa kendaraan belum terdeteksi masuk. Nantinya, variabel ini akan berubah sesuai hasil pembacaan *saldo* dan logika pintu masuk.

String *lastUID* = ""; berfungsi menyimpan *UID* terakhir dari kartu *RFID yang* terbaca. *UID* ini digunakan untuk membandingkan apakah kartu yang discan saat ini sama dengan kartu sebelumnya. Jika sama, sistem bisa mengabaikan pembacaan hingga melewati jedaScan, sehingga mencegah proses ganda untuk kartu yang sama.

```
void setup() {
  Serial.begin(115200);
  Wire.begin(4, 5); // SDA, SCL khusus OLED

  SPI.begin();
  rfid.PCD_Init();

  // OLED
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("OLED gagal");
    while (1);
  }
```

Fungsi setup()komunikasi serial diaktifkan dengan Serial.begin(115200) seh ingga mikrokontroler dapat berkomunikasi dengan komputer melalui Serial Monitor untuk keperluan debugging dengan kecepatan 115200 baud. Setelah itu, protokol *I2C* diinisialisasi menggunakan Wire.begin(4, 5), di mana pin GPIO4 ditetapkan sebagai SDA dan GPIO5 sebagai SCL untuk menghubungkan

mikrokontroler dengan *modul OLED.* Selanjutnya, protokol komunikasi *SPI* dimulai dengan *SPI.begin*(), yang diperlukan untuk berkomunikasi dengan *modul RFID MFRC522. Modul RFID* kemudian diinisialisasi dengan perintah rfid.PCD_Init() agar siap digunakan untuk membaca data kartu atau *tag RFID.*

Bagian terakhir adalah inisialisasi layar *OLED* melalui perintah *display.begin*(SSD1306_SWITCHCAPVCC, 0x3C). *Jika OLED* tidak terdeteksi atau gagal diinisialisasi pada alamat *I2C 0x3C,* maka program akan mencetak pesan *"OLED* gagal" pada Serial Monitor dan masuk ke dalam loop tak terbatas (while(1)), yang artinya program berhenti berjalan untuk mencegah error lebih lanjut. Dengan demikian, fungsi setup() ini berfungsi memastikan bahwa komunikasi *serial, OLED, dan RFID* siap digunakan sebelum program masuk ke fungsi utama loop().

```
  // Servo
  palangServo.setPeriodHertz(50);
  palangServo.attach(SERVO_PIN, 500, 2500);
  palangServo.write(90); // Palang terbuka awal

  tampilkanStatus("Sistem Aktif", "Palang Terbuka");

  // WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    tampilkanStatus("Menyambung WiFi", "");
    delay(500);
  }
  tampilkanStatus("WiFi Terhubung", "Dekatkan
kartu");
}
```

setup() tersebut, bagian pertama berfungsi untuk mengatur servo yang digunakan sebagai palang parkir. Servo diatur dengan frekuensi kerja sebesar 50 Hz menggunakan palangServo.setPeriodHertz(50), kemudian dihubungkan ke pin yang sudah ditentukan melalui palangServo.attach(SERVO_PIN, 500, 2500), di

mana nilai 500 dan 2500 merupakan batas pulsa minimum dan maksimum dalam mikrodetik untuk menentukan sudut putar servo. Selanjutnya, palangServo.write(90) digunakan untuk memposisikan servo pada sudut 90 derajat, yang diasumsikan sebagai posisi palang terbuka pada awal sistem dijalankan. Setelah itu, program menampilkan informasi *pada OLED* melalui fungsi tampilkanStatus("Sistem Aktif", "Palang Terbuka") sebagai indikasi bahwa sistem sudah aktif dan palang dalam kondisi terbuka.

Bagian berikutnya adalah inisialisasi koneksi WiFi. Perintah WiFi.begin(ssid, password) digunakan untuk menyambungkan mikrokontroler ke jaringan WiFi sesuai dengan SSID dan password yang telah ditentukan. Selama proses penyambungan, program akan terus menampilkan status "Menyambung WiFi*" pada OLED* melalui fungsi tampilkanStatus("Menyambung WiFi", "") sambil memberikan jeda 500 milidetik. Proses ini berlangsung hingga koneksi WiFi berhasil. Setelah koneksi terhubung, status akan diperbarui menjadi "WiFi Terhubung" dan instruksi "Dekatkan kartu" ditampilkan, yang menandakan bahwa sistem siap digunakan untuk membaca kartu *RFID* sebagai akses parkir.

```
void loop() {
  if (rfid.PICC_IsNewCardPresent() &&
rfid.PICC_ReadCardSerial()) {
    String uid = "";
    for (byte i = 0; i < rfid.uid.size; i++) {
      uid += String(rfid.uid.uidByte[i] < 0x10 ? "0"
: "");
      uid += String(rfid.uid.uidByte[i], HEX);
    }
    uid.toUpperCase();
    Serial.println("UID: " + uid);
    rfid.PICC_HaltA();
```

loop() tersebut, program berfungsi untuk mendeteksi kartu *RFID yang didekatkan ke reader. Proses dimulai dengan pengecekan* kondisi if (rfid.PICC_Is

NewCardPresent() && rfid.PICC_ReadCardSerial()), yang memastikan bahwa ad a kartu baru yang terdeteksi dan data serialnya berhasil dibaca. Jika kondisi ini terpenuhi, maka sistem akan membuat variabel string bernama uid untuk menampung nomor identitas unik *(UID)* dari kartu. *UID* dibentuk dengan melakukan perulangan for sesuai panjang data yang diterima dari kartu (rfid.uid.size). Setiap *byte UID* akan dikonversi menjadi format heksadesimal dengan menambahkan nol di depan jika nilai byte kurang dari 0x10 agar tampilannya seragam. Setelah semua byte digabung, hasil *UID* diubah menjadi huruf kapital menggunakan uid.toUpperCase(). *UID yang* sudah jadi kemudian ditampilkan pada Serial Monitor dengan perintah Serial.println*("UID: "* + uid), sehingga pengguna dapat melihat identitas kartu yang berhasil dibaca. Terakhir, rfid.PICC_HaltA() dipanggil untuk menghentikan komunikasi dengan kartu tersebut, sehingga pembacaan berikutnya hanya dilakukan ketika ada kartu baru yang didekatkan. Dengan alur ini, fungsi loop() memastikan bahwa sistem mampu

```
if (!statusMasuk) {
      // Proses Masuk
      statusMasuk = true;
      lastUID = uid;
      tampilkanStatus("Palang Tertutup", "Selamat
Berbelanja");
      palangServo.write(20); // Palang tertutup
    } else {
      // Proses Keluar
      if (uid == lastUID) {
        int saldo = getSaldo(uid);
        if (saldo >= 2000) {
          int saldoBaru = saldo - 2000;
          if (updateSaldo(uid, saldoBaru)) {
            tampilkanStatus("Saldo: Rp" +
```

fungsi loop() ini, kode digunakan untuk mengatur logika keluar-masuk kendaraan berdasarkan *data UID* kartu *RFID.* Pertama, program memeriksa kondisi if (!statusMasuk). Jika statusMasuk bernilai false, artinya kartu baru terdeteksi

sebagai proses masuk. Maka sistem akan mengubah nilai statusMasuk menjadi true, menyimpan *UID* kartu yang sedang digunakan ke variabel *lastUID*, kemudian menampilkan *status pada OLED* dengan pesan “Palang Tertutup” dan “Selamat Berbelanja”. Setelah itu, perintah palangServo.write(20) dijalankan untuk memposisikan servo pada sudut tertentu yang melambangkan palang dalam keadaan tertutup.

Sebaliknya, jika statusMasuk bernilai true, maka sistem menganggap proses yang terjadi adalah keluar. Pada kondisi ini, sistem akan membandingkan *UID* kartu yang digunakan saat keluar dengan *lastUID (UID* terakhir yang masuk). Jika keduanya sama, maka dilakukan pengecekan *saldo* melalui fungsi getSaldo(uid). Jika *saldo* yang tersimpan pada kartu mencukupi, minimal Rp2000, maka sistem menghitung *saldo* baru dengan mengurangi Rp2000 dari *saldo* awal. Setelah itu, fungsi updateSaldo(uid, *saldoBaru*) dipanggil untuk memperbarui *saldo pada database* atau penyimpanan. Jika proses pembaruan berhasil, maka *OLED* akan menampilkan pesan dengan format “Saldo: Rp...” yang menunjukkan *saldo* terbaru pengguna.

```
String(saldoBaru), "Palang Terbuka");
            palangServo.write(90); // Palang terbuka
            statusMasuk = false;   // Reset status
          } else {
            tampilkanStatus("Update Gagal", "Coba
Lagi");
          }
        } else {
          tampilkanStatus("Saldo Tidak Cukup", "Top
Up Dulu");
        }
      } else {
        tampilkanStatus("Bukan Kartu Sama", "Coba
Kartu Sebelumnya");
      }
    }
  waktuTerakhirScan = millis();
```

```
  }
```

kode ini, logika proses keluar kendaraan dilengkapi hingga akhir. Jika *saldo* mencukupi dan berhasil diperbarui melalui fungsi updateSaldo(uid, *saldoBaru*), maka sistem akan menampilkan informasi *pada OLED* berupa *saldo* terbaru serta status "Palang Terbuka". Setelah itu, perintah palangServo.write(90) dijalankan untuk memutar servo ke posisi 90 derajat yang berarti palang parkir terbuka, sehingga kendaraan dapat keluar. Kemudian variabel statusMasuk di-reset menjadi false agar sistem kembali ke kondisi siap menerima proses masuk berikutnya.

Namun, apabila pembaruan *saldo* gagal, maka *OLED* akan menampilkan pesan "Update Gagal" dengan instruksi "Coba Lagi" untuk memberi tahu pengguna bahwa transaksi tidak berhasil. Jika *saldo* yang dimiliki tidak mencukupi (kurang dari Rp2000), sistem akan menampilkan pesan "Saldo Tidak Cukup" dan "Top Up Dulu", yang berarti pengguna perlu melakukan pengisian ulang *saldo* sebelum bisa keluar. Selain itu, jika kartu yang digunakan saat keluar tidak sama dengan kartu yang digunakan saat masuk *(UID* berbeda dari *lastUID*), maka sistem menampilkan b pesan "Bukan Kartu Sama" dan "Coba Kartu Sebelumnya". Dengan demikian, kode ini memastikan bahwa proses keluar hanya dapat dilakukan oleh kartu yang sama dengan kartu masuk, *saldo* mencukupi, dan pembaruan *saldo* berhasil, sehingga keamanan dan akurasi sistem parkir tetap terjaga.

```
  if (millis() - waktuTerakhirScan > jedaScan &&
!statusMasuk) {
    tampilkanStatus("Dekatkan kartu", "Palang
Terbuka");
  }
}

void tampilkanStatus(String baris1, String baris2) {
  display.clearDisplay();
```

```
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0); display.println(baris1);
  display.setCursor(0, 16); display.println(baris2);
  display.display();

}
```

terdapat logika yang digunakan untuk mengatur tampilan pesan pada layar *OLED* berdasarkan waktu jeda dan status masuk kendaraan. Baris if (millis() - waktuTerakhirScan > jedaScan && !statusMasuk) berfungsi untuk memeriksa apakah sudah lewat waktu jeda tertentu sejak kartu terakhir kali discan serta memastikan bahwa tidak ada kendaraan yang sedang berstatus masuk. Jika kedua kondisi tersebut terpenuhi, maka fungsi tampilkanStatus() akan dipanggil untuk menampilkan pesan "Dekatkan kartu" pada baris pertama dan "Palang Terbuka" pada baris kedua.

Fungsi tampilkanStatus() sendiri dibuat untuk menyederhanakan proses penulisan teks ke *OLED*, sehingga tidak perlu menuliskan ulang perintah pengaturan tampilan setiap kali ingin menampilkan pesan baru. Di dalam fungsi tersebut, layar *OLED* terlebih dahulu dikosongkan menggunakan clearDisplay(), kemudian teks diatur ukurannya, warnanya, serta posisinya dengan setCursor(). Setelah itu, teks dari parameter baris1 dan baris2 dicetak ke layar, lalu ditampilkan secara penuh dengan perintah *display.display*(). Dengan cara ini, sistem parkir dapat memberikan informasi yang jelas dan real-time kepada pengguna, misalnya instruksi untuk mendekatkan kartu atau kondisi lain seperti *saldo* tidak cukup atau kartu tidak terdaftar.

```
int getSaldo(String uid) {
  if (WiFi.status() != WL_CONNECTED) return -1;
  HTTPClient http;
```

```
  http.begin(urlGetSaldo + String("?id=") + uid);
  int httpCode = http.GET();
  String response = http.getString();
  http.end();
  if (httpCode == 200) {
    return response.toInt();
  } else {
    return -1;
  }
}
```

Fungsi getSaldo(String uid) digunakan untuk mengambil *saldo* dari *database* melalui koneksi internet berdasarkan *UID* kartu *RFID* yang dipindai. Pertama, fungsi akan memeriksa apakah *modul ESP32* masih terhubung dengan jaringan WiFi, dan jika tidak, fungsi langsung mengembalikan nilai -1 sebagai tanda error. Jika terhubung, maka dibuat objek *HTTPClient* untuk mengirim permintaan *HTTP GET* ke *server* dengan alamat URL yang sudah ditentukan di variabel urlGetSaldo. *UID* kartu ditambahkan ke dalam URL sebagai parameter query string, misalnya ...?id=12345. Setelah permintaan dikirim, *server* akan merespons dengan data *saldo* dalam bentuk string. Nilai ini kemudian dibaca menggunakan http.getString(), lalu dikonversi menjadi bilangan integer dengan toInt() sebelum dikembalikan sebagai hasil fungsi. Jika status kode *HTTP yang* diterima bukan 200 (OK), maka fungsi juga akan mengembalikan nilai -1. Dengan cara ini, sistem bisa membedakan antara *saldo* valid dari *database* dengan kondisi gagal koneksi atau respons tidak valid.

```
bool updateSaldo(String uid, int newSaldo) {
  if (WiFi.status() != WL_CONNECTED) return false;
  HTTPClient http;
  http.begin(urlUpdateSaldo);
  http.addHeader("Content-Type", "application/x-www-
form-urlencoded");
  String data = "id=" + uid + "&saldo=" +
String(newSaldo);
  int httpCode = http.POST(data);
  http.end();
return (httpCode == 200);
```

```
}
```

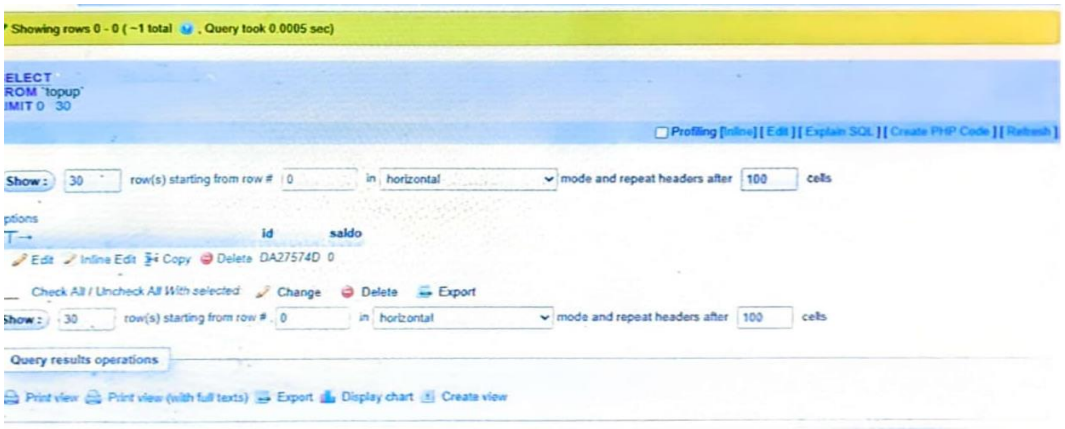
Fungsi updateSaldo(String uid, int newSaldo) digunakan untuk memperbarui *saldo* kartu *RFID* ke *database* melalui koneksi internet. Pertama, fungsi akan memastikan bahwa *ESP32* masih dalam kondisi terhubung ke jaringan WiFi. Jika tidak terhubung, maka fungsi langsung mengembalikan nilai false sebagai tanda gagal. Jika terhubung, objek *HTTPClient* dibuat untuk mengirim permintaan *HTTP POST* ke *server* dengan alamat yang sudah ditentukan di variabel urlUpdateSaldo. Sebelum mengirim data, ditambahkan header Content-Type dengan nilai application/x-www-form-urlencoded *agar server* tahu bahwa data dikirim dalam format form standar. Kemudian dibuat string data yang berisi pasangan kunci-nilai, yaitu id=<uid>&*saldo*=<newSaldo>, *di mana UID* kartu dan *saldo* baru dikirimkan ke *server*. Setelah data dikirim menggunakan http.POST(data), fungsi memeriksa status kode yang dikembalikan *server*. *Jika* kode tersebut bernilai 200 (berhasil), fungsi mengembalikan true, jika tidak maka false. Dengan mekanisme ini, sistem parkir dapat memastikan bahwa setiap transaksi seperti *top-up* atau pengurangan *saldo* tercatat dengan benar *di database*.

C. Pengujian

1. Pengujian Kartu Tidak Terisi Saldo

Pengujian ini dilakukan untuk mengetahui bagaimana sistem e-Parkir akan merespons jika pengguna mencoba menggunakan kartu *RFID yang saldonya* tidak mencukupi atau *saldonya* kurang dari batas minimum yang ditentukan, yaitu Rp2.000. Pengujian dilakukan dengan menempatkan kartu *RFID yang* sebelumnya telah terdaftar di basis data tetapi tidak memiliki *saldo* atau *saldonya* nol.

Ketika kartu ditempelkan pada *modul RFID MFRC522,* sistem akan membaca *UID* kartu dan meminta *saldo* dari *server* melalui koneksi *Wi-Fi. Server* akan mengembalikan informasi *saldo,* yang kemudian akan diproses oleh sistem itu sendiri. Jika *saldo* yang dikembalikan kurang dari Rp2.000, sistem akan menampilkan pesan di layar *OLED yang* bertuliskan "Saldo Tidak Cukup" pada baris pertama dan "Top Up Dulu" pada baris kedua. Selanjutnya, motor servo tidak akan aktif, sehingga palang parkir tetap tertutup. Hal ini membuktikan bahwa sistem melakukan validasi yang tepat terhadap *saldo* minimum untuk keluar dari area parkir.



Gambar 4. 7 Kartu Tidak Terisi Saldo



Gambar 4. 8 Kartu Tidak Terisi Saldo

Hasil pengujian yang telah dilakukan menunjukkan bahwa sistem e-Parkir memiliki kemampuan yang baik untuk memvalidasi *saldo* kartu *RFID. Ketika* kartu *RFID* dengan *saldo* nol ditempelkan ke alat, sistem secara otomatis meminta *server* untuk memeriksa *saldo* yang tersimpan di basis data.

Hasil pengujian menunjukkan bahwa sistem secara konsisten menolak keluar jika *saldo* di bawah ambang batas minimum yang ditetapkan, yaitu Rp2.000. Hal ini dapat diamati dari dua aspek utama berikut: pertama, tampilan *OLED* menampilkan pesan "Saldo Tidak Cukup" dan "Top up Dulu" yang terbaca jelas oleh pengguna sebagai peringatan atau notifikasi. Kedua, motor servo tidak bergerak untuk membuka, sehingga akses tetap terhalang bagi kendaraan.

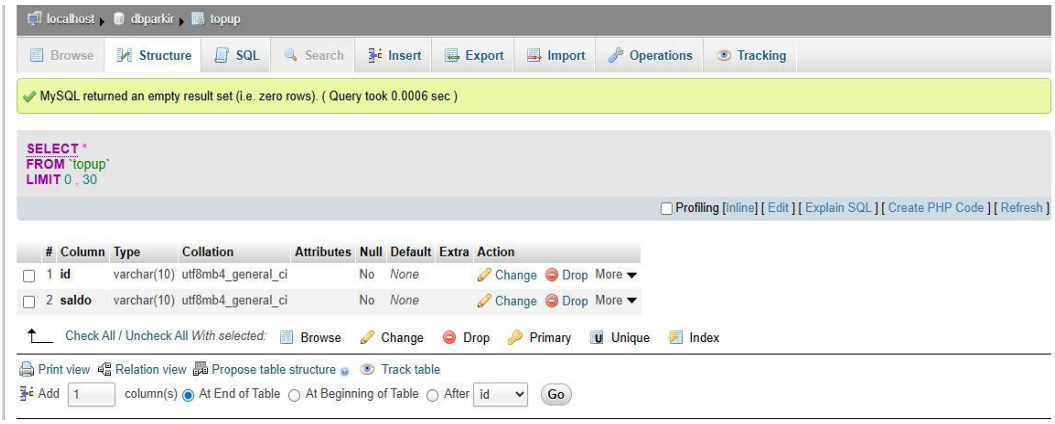
Keberhasilan sistem dalam mendeteksi kondisi keseimbangan yang tidak cukup untuk menjamin keluar menunjukkan bahwa sistem terintegrasi dengan baik antara komponen perangkat keras *(ESP32, RFID, servo, OLED) dan* perangkat lunak *(server-side PHP dan basis data MySQL).*

2. Pengujian Kartu Tidak Terdaftar

Pengujian ini bertujuan untuk mengevaluasi bagaimana sistem e-Parkir bereaksi menggunakan kartu *RFID yang* tidak terdaftar (yaitu, tidak memiliki entri *UID* dalam *basis data server) dan* memeriksa apakah sistem mampu mendaftarkan kartu baru secara otomatis saat pengisian ulang. Dalam skenario khusus ini, kartu *RFID* baru ditempelkan pada mesin pengisian ulang. Sistem membaca *UID* kartu menggunakan *modul RFID MFRC522 dan* mengirimkannya ke *server* melalui metode *HTTP POST* ke titik akhir topup.php beserta jumlah *saldo* yang dipilih oleh pengguna.

Jika UID tersebut tidak tercatat dalam tabel pengguna basis data (pengguna*), server* akan secara otomatis menambahkan satu catatan lagi dengan *UID yang* diberikan dan jumlah *saldo* awal sesuai dengan pilihan pengisian ulang. Setelah itu, sistem menampilkan *saldo* yang berhasil diisi ulang pada layar *OLED yang* menandakan pengenalan kartu dan pendaftaran berhasil. Kemudian, dari titik tersebut, kartu yang terdaftar dapat langsung digunakan saat masuk dan keluar gerbang parkir, sama seperti kartu lain yang telah terdaftar sebelumnya.

Pengujian ini telah menunjukkan bahwa sistem memang menyediakan mekanisme yang sangat mudah untuk registrasi otomatis di mana pengguna baru tidak perlu melakukan registrasi.



Gambar 4. 9 Kartu Tidak Terdaftar

Pengujian ini bertujuan untuk mengetahui bagaimana sistem e-Parkir bereaksi ketika pengguna mencoba menggunakan kartu *RFID yang* belum terdaftar di basis data dan menguji fitur registrasi otomatis tersebut saat melakukan pengisian *saldo*. Secara subjektif, pengguna pertama kali menguji kartu *RFID* untuk melakukan pengisian *saldo* tanpa registrasi manual sebelumnya. Secara objektif, sistem membaca *UID* dari kartu *RFID* menggunakan *modul MFRC522 dan*

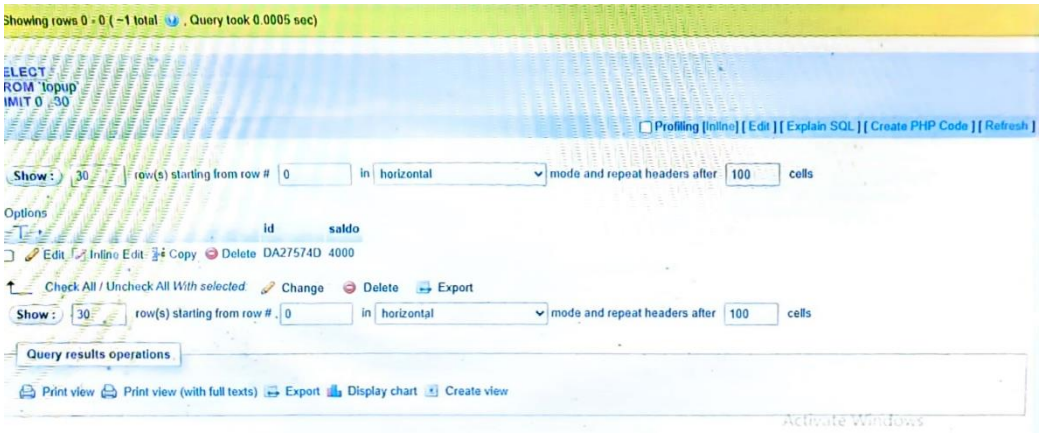
kemudian mengirimkannya ke *server* menggunakan protokol *HTTP POST* ke titik akhir topup.php beserta jumlah pengisian *saldo* yang dipilih. *Jika UID* tidak ditemukan *oleh server di basis data,* secara otomatis, *UID* tersebut akan ditambahkan sebagai entri baru dan akan menyimpan *saldo* awal sesuai nominal yang dipilih. Setelah proses berhasil, layar *OLED* menampilkan informasi "Saldo Anda:", yang berarti kartu telah aktif dan siap digunakan. Dari sudut pandang penilaian, proses ini berhasil menunjukkan bahwa sistem mampu melakukan identifikasi, registrasi, dan pengisian *saldo* dalam satu langkah otomatis.

Tabel 4. 1 Pengujian Kartu Tidak Terdaftar

No.	Jenis Kartu	UID Kartu	Respon Sistem	Status Akses	Keterangan
1	Terdaftar	DA27574D	Saldo dibaca: Rp20.000. Akses diberikan.	Terbuka	Palang terbuka, saldo berkurang Rp2.000
2	Terdaftar	DA27574D	Saldo dibaca: Rp10.000. Akses diberikan.	Terbuka	Saldo mencukupi
3	Tidak Terdaftar	BA27574D	UID tidak dikenali. Registrasi otomatis dimulai.	Tertutup	Kartu harus top-up terlebih dahulu untuk akses
4	Tidak Terdaftar	BA27574D	UID tidak dikenali. Tidak ada data di database.	Tertutup	Sistem menolak akses
5	Terdaftar (Saldo < 2000)	DA27574D	Saldo: Rp1.000. Akses ditolak.	Tertutup	Pesan "Saldo tidak cukup" muncul di OLED

3. Pengujian Topup Saldo

Pengujian ini bertujuan untuk memastikan bahwa sistem mampu melakukan proses pengisian *saldo (top-up)* dengan benar pada kartu *RFID yang* digunakan dalam sistem e-Parkir. Langkah pertama dalam pengujian ini adalah menempelkan kartu *RFID* ke alat *top-up yang* telah terhubung ke *database server* melalui jaringan WiFi. Setelah kartu terdeteksi, pengguna dapat memilih nominal *saldo* yang diinginkan, seperti Rp20.000, Rp30.000, Rp50.000, atau Rp100.000, dengan menggunakan tombol yang tersedia. Setelah nominal dipilih, pengguna menekan tombol "Kirim" untuk mengonfirmasi *top-up.* Sistem akan menggabungkan *data UID* dari kartu *RFID dan nominal saldo* yang dipilih, kemudian mengirimkannya ke *server* menggunakan metode *HTTP POST*. *Server* akan memproses data tersebut dengan cara mencatat atau memperbarui entri *saldo* pada tabel users *di database.* Setelah proses berhasil, *server* akan mengirimkan kembali nilai *saldo* terbaru kepada perangkat. Nilai *saldo* ini kemudian ditampilkan secara langsung pada layar *OLED* sebagai bukti bahwa *proses top-up* berhasil dilakukan. Berdasarkan hasil pengujian yang dilakukan, semua *nominal top-up* berhasil ditambahkan ke dalam *saldo* kartu sesuai permintaan, serta tampil secara real-time di layar *OLED, yang* menandakan sistem telah berjalan sesuai dengan fungsinya.



Gambar 4. 10 Topup Saldo

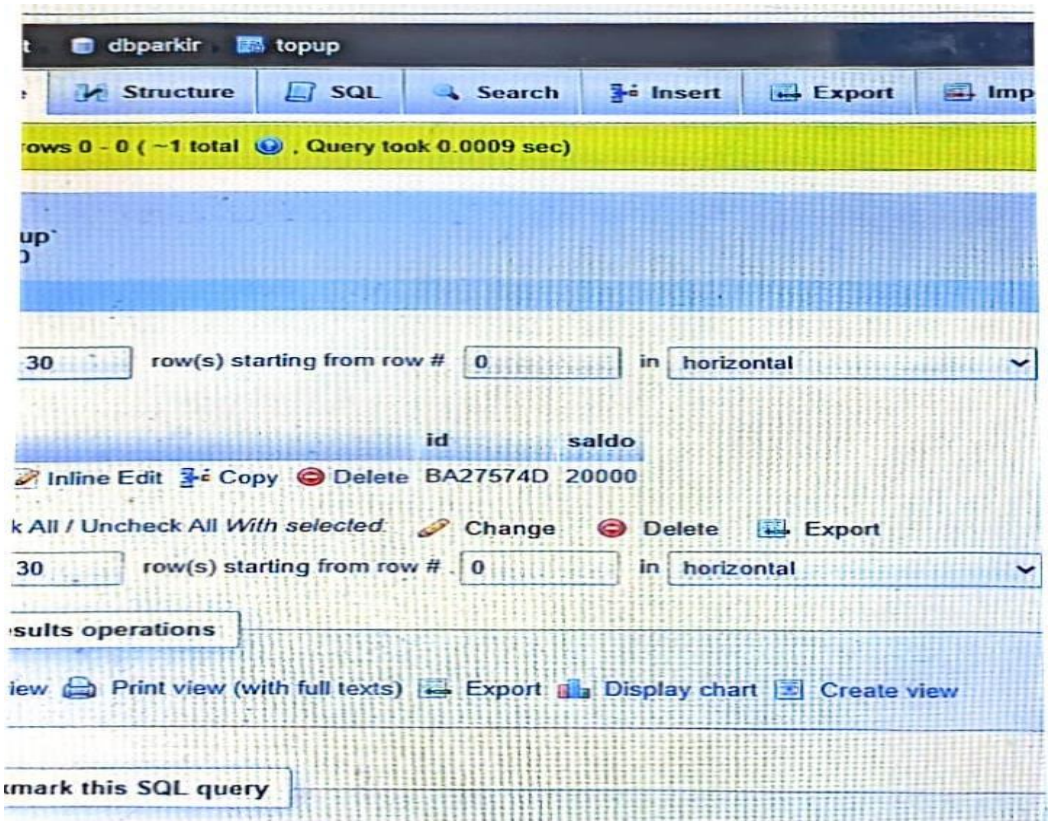
Berdasarkan hasil pengujian, sistem e-Parkir berhasil menjalankan fungsi *top-up saldo* dengan baik dan sesuai dengan yang diharapkan. Seluruh nominal yang dipilih pengguna (Rp20.000, Rp30.000, Rp50.000, dan Rp100.000) dapat dikirimkan ke *server* melalui metode *HTTP POST dan* diproses secara otomatis. Server merespons dengan melakukan pembaruan data *saldo pada database* sesuai *UID yang* dikirim, kemudian mengembalikan nilai *saldo* terbaru. Nilai tersebut ditampilkan secara *real-time pada OLED, yang* memberikan umpan balik visual langsung kepada pengguna. Keberhasilan ini menunjukkan bahwa integrasi antara perangkat keras *(RFID reader,* tombol *input, OLED display),* jaringan WiFi, *dan backend database* sudah berjalan stabil dan sinkron. Selain itu, fitur ini juga mendukung efisiensi waktu dan kemudahan bagi pengguna, karena tidak diperlukan prosedur manual dalam pengisian *saldo*. Pengujian ini membuktikan bahwa sistem mampu menangani skenario operasional utama dalam pengelolaan *saldo* e-Parkir secara otomatis dan responsif.

Tabel 4. 2 Pengujian Topup Saldo

No	UID Kartu	Nominal Dipilih	Input Tombol	Output OLED	Server Database
1	DA27574D	Rp20.000	Tombol Kirim	"Saldo Anda: Rp20.000"	20.000
2	DA27574D	Rp30.000	Tombol Kirim	"Saldo Anda: Rp50.000"	50.000
3	DA27574D	Rp50.000	Tombol Kirim	"Saldo Anda: Rp100.000"	100.000
4	DA27574D	Rp100.000	Tombol Kirim	"Saldo Anda: Rp200.000"	200.000

4. Pengujian Pengurangan Saldo

Pengujian ini bertujuan untuk memastikan sistem dapat secara otomatis memotong *saldo* ketika pengguna meninggalkan area parkir. *Kartu RFID* berisi *saldo* dari pengisian ulang sebelumnya ditunjukkan kepada pembaca di gerbang keluar. Sistem membaca *UID* kartu dan mengirimkannya ke *server* menggunakan *HTTP GET* untuk memeriksa *saldo* yang tersimpan di basis data. *Jika saldo pada kartu minimal Rp2.000, server akan mengizinkan keluar.* Sistem kemudian memotong biaya sebesar Rp2.000 dan memperbarui *saldo* yang sesuai di basis data. Saldo baru ditampilkan di layar *OLED, dan* palang pintu otomatis terbuka oleh motor servo untuk menandakan bahwa kendaraan dapat keluar. Sistem akan menunggu beberapa saat sebelum kembali ke mode siaga untuk membaca kartu berikutnya.



Gambar 4. 11 Pengurangan Saldo

Pengujian mengonfirmasi bahwa sistem menyelesaikan verifikasi dan pengurangan *saldo* secara otomatis dan tepat waktu. Setiap kali pengguna menempelkan kartu *RFID* terdaftar pada pembaca sistem langsung mengecek *saldo* dan memeriksa kecukupan *saldo*. Proses pengurangan *saldo* dilakukan dengan benar sesuai dengan nominal Rp2.000 yang telah ditetapkan. *Layar OLED* menampilkan *saldo* terbaru yang telah dikurangi sebagai bentuk tanggapan visual untuk pengguna. Palang secara otomatis membuka akses keluar setelah *saldo* dikurangi sehingga menunjukkan proses akses keluar berhasil. Mikrokontroler *ESP32* berkomunikasi secara responsif dengan *server database* dan komponen output motor servo dalam sistem. Hasil pengujian menunjukkan bahwa sistem

berhasil membuktikan kemampuannya untuk berfungsi sebagai solusi e-Parkir otomatis yang efisien di dunia nyata.

Tabel 4. 3 Pengujian Pengurangan Saldo

No	UID Kartu	Saldo Awal	Status Masuk	Aksi	Saldo Sesudah	Output OLED	Database
1	DA27574D	Rp20.000	Belum Masuk	Tap kartu (Masuk)	Rp20.000	"Palang Tertutup"	20.000
2	DA27574D	Rp20.000	Sudah Masuk	Tap kartu (Keluar)	Rp18.000	"Saldo: Rp18.000"	18.000
3	DA27574D	Rp18.000	Belum Masuk	Tap kartu (Masuk)	Rp18.000	"Palang Tertutup"	18.000
4	DA27574D	Rp18.000	Sudah Masuk	Tap kartu (Keluar)	Rp16.000	"Saldo: Rp16.000"	16.000
5	DA27574D	Rp16.000	Belum Masuk	Tap kartu (Masuk)	Rp16.000	"Palang Tertutup"	16.000
6	DA27574D	Rp16.000	Sudah Masuk	Tap kartu (Keluar)	Rp14.000	"Saldo: Rp14.000"	14.000

BAB V

PENUTUP

A. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem e-Parkir berbasis *RFID* dengan fitur *top-up dan* pengurangan *saldo* otomatis menggunakan *ESP32*, dapat disimpulkan beberapa hal berikut:

1. Sistem berhasil melakukan proses *top-up saldo* dengan nominal yang bervariasi (Rp20.000, Rp30.000, Rp50.000, dan Rp100.000) secara real-time ke dalam *database* menggunakan koneksi WiFi. *UID* kartu *RFID yang* belum terdaftar dapat secara otomatis ditambahkan ke *database* saat *proses top-up* dilakukan.
2. Sistem juga berhasil melakukan pengurangan *saldo* secara otomatis ketika pengguna keluar dari area parkir, dengan memotong *saldo* sebesar Rp2.000 apabila *saldo* cukup. Pengurangan *saldo* ditampilkan secara langsung *di OLED dan* diiringi dengan aksi pembukaan palang otomatis.
3. Sistem juga dapat mendeteksi kartu yang tidak terdaftar, serta mencegah akses keluar jika *saldo* tidak mencukupi, yang menunjukkan bahwa validasi keamanan sistem sudah berjalan sesuai fungsi.

B. Saran

Sistem dapat dikembangkan lebih lanjut dengan penambahan fitur notifikasi melalui aplikasi mobile atau SMS gateway, sehingga pengguna dapat menerima informasi *saldo* atau riwayat parkir secara langsung.

DAFTAR PUSTAKA

Amin, M., & Novelan, M. S. (2020). Sistem Cerdas Kontrol Kran Air Menggunakan Mikrokontroler *Arduino* dan Sensor Ultrasonic. *Jurnal Nasional Informatika Dan Teknologi Jaringan, 2*, 1-5.

Andeta, A. (2024). *RANCANGAN STRATEGI PENGUATAN TRANSAKSI QRIS PADA RETRIBUSI PARKIR DI KOTA SEMARANG* (Doctoral dissertation, Universitas Islam Sultan Agung Semarang).

Arifin, Z., Prasetyo, S. D., Alfaiz, N. F., Rachmanto, R. A., Juwana, W. E., Widhiyanuariyawan, D., & Yohana, E. (2025). *Transisi Energi Bersih: Strategi dan Infrastruktur Stasiun Pengisian Kendaraan Listrik*. MEGA PRESS NUSANTARA.

Deddy Pradana, R. (2021). *PEMILAH SAMPAH PINTAR BERDASARKAN LOGAM DAN NONLOGAM DI KANTOR FAKULTAS TEKNIK* (Doctoral dissertation, Universitas Muhammadiyah Ponorogo).

Devani, A., Nugroho, S., Jannah, W., & Santi, I. H. (2024). PERANCANGAN SISTEM INFORMASI TEMPAT PARKIR BERBASIS APLIKASI WEB DI UNIVERSITAS ISLAM BALITAR. *JATI (Jurnal Mahasiswa Teknik Informatika)*, *8*(3), 4329-4335.

Fauzi, M. F., Nurcholis, T., Kuswanto, J., Abdulloh, F. F., & Amrulloh, Y. A. (2022). Inovasi Sistem Pembayaran E-Parkir Cashless Dengan Teknologi Hybrid Payment System Berbasis *QRIS*. *Jurnal Infomedia: Teknik Informatika, Multimedia, dan Jaringan*, *7*(2), 77-81.

Fonna, M. Z., Husaini, H., & Indrawati, I. (2020). Penerapan Iot *(Internet* Of *Things)* Untuk Pemberian Pakan Ikan Pada Aquarium. *Jurnal Teknologi Rekayasa Informasi dan Komputer*, *3*(2).

Hamdani, R., Puspita, H., & Wildan, D. R. (2019). Pembuatan sistem pengamanan kendaraan bermotor berbasis radio frequency identification *(RFID)*. *INDEPT: Jurnal Industri, Elektro dan Penerbangan*, *8*(2).

Jaya, A. K. (2021). *RANCANG BANGUN ALAT UNTUK MENINGKATKAN PRODUKTIFITAS EBAH MADU BERBASIS FUZZY LOGIC* (Doctoral dissertation, Universitas Muhammadiyah Gresik).

JULIANTO, T. A. (2022). *Rancang Bangun Sistem Penerangan Area Parkir Otomatis Berbasis Internet of Things (Iot) Menggunakan Esp32* (Doctoral dissertation, Universitas Islam Sultan Agung).

Putra, D. E., Kom, S., Kom, M., Amnur, H., Kom, S., Flomina, K., & Kom, M. (2025). *Konsep Dasar Internet Of Things (IoT) dengan Mikrokontroler Esp32*.

RAMADHAN, R. P. (2023). *Rancang Bangun Sistem Pengamanan Kebocoran Gas LPG Dan Perbandingan Kinerja Menggunakan Sensor TGS 2610 Dan MQ-5 Berbasis IOT* (Doctoral dissertation, UNIVERSITAS ISLAM NEGERI SULTAN SYARIF KASIM RIAU).

Saghoa, Y. C., Sompie, S. R., & Tulung, N. M. (2018). Kotak Penyimpanan Uang Berbasis Mikrokontroler *Arduino* Uno. *Jurnal Teknik Elektro dan Komputer*, *7*(2), 167-174.

Salsabila, S., Anwar, A., & Syahputra, G. (2024). Implementation of *RFID-Based* Attendance Integrated with Management Systems and Notifications via WhatsApp. *Journal of Artificial Intelligence and Software Engineering*, *4*(2), 85-93.

SIBURIAN, RINGGAS. "Rancang Bangun Sistem Informasi Pasien Berbasis Wemos D1 Mini Layar Display Untuk Rumah Sakit." (2024)

Siregar, M. I. S. (2017). *Perancangan Peralatan Sistem Keamananan Elektronik di Shelter BTS Secara Real Time Melalui SMS Berbasis Mikrokontroller ATMega16 DAN Module GSM* (Doctoral dissertation).

Utomo, Teguh Prasetyo. "Potensi implementasi *Internet* of *Things (IoT)* untuk perpustakaan." *Buletin Perpustakaan* 2.1 (2019): 1-18.