

BAB III

METODE PENELITIAN

3.1. Jenis dan pendekatan Penelitian

Penelitian ini merupakan penelitian terapan yang bertujuan untuk menghasilkan suatu solusi sistem parkir otomatis dalam bentuk *prototype rotary parking* berbasis mikrokontroler. Penelitian terapan dipilih karena berfokus pada perancangan dan implementasi sistem yang mampu menjawab permasalahan nyata, khususnya keterbatasan lahan parkir serta inefisiensi waktu dalam proses pencarian slot parkir pada sistem konvensional.

Metode yang digunakan dalam penelitian ini adalah pendekatan sistem kontrol sekuensial. Metode sekuensial merupakan metode pengendalian yang bekerja berdasarkan urutan langkah atau tahapan proses yang telah ditentukan secara sistematis. Dalam sistem ini, setiap proses akan dijalankan secara berurutan mulai dari tahap *input*, pemrosesan, hingga *output*, sehingga alur kerja sistem menjadi lebih terstruktur dan mudah dikendalikan. Penerapan metode ini sesuai dengan karakteristik sistem *rotary parking* yang membutuhkan urutan proses yang jelas, seperti proses deteksi slot kosong, pergerakan motor, hingga penentuan posisi akhir.

Pendekatan penelitian dilakukan secara eksperimental, yaitu dengan melakukan pengujian langsung terhadap *prototype* yang telah dirancang. Sistem yang dikembangkan memanfaatkan ESP32 sebagai pengendali utama yang mengatur alur kerja secara sekuensial, mulai dari menerima *input* pengguna, mengaktifkan proses deteksi slot menggunakan ESP32-CAM, membaca posisi menggunakan Sensor *Hall Effect* sebagai penghitung (*counter*), hingga mengontrol pergerakan motor DC melalui *driver* motor. Setiap tahapan tersebut berjalan secara berurutan sesuai dengan logika sistem yang telah diprogram.

Pengujian dilakukan untuk mengetahui performa sistem dalam mendeteksi slot parkir kosong, ketepatan posisi *rotary*, serta efisiensi waktu yang dibutuhkan dalam proses parkir. Data hasil pengujian kemudian dianalisis untuk mengevaluasi kinerja sistem secara keseluruhan. Dengan pendekatan ini, penelitian tidak hanya menekankan aspek implementasi teknis, tetapi juga memastikan bahwa sistem yang

dirancang mampu bekerja secara optimal sesuai dengan urutan proses yang telah ditentukan.

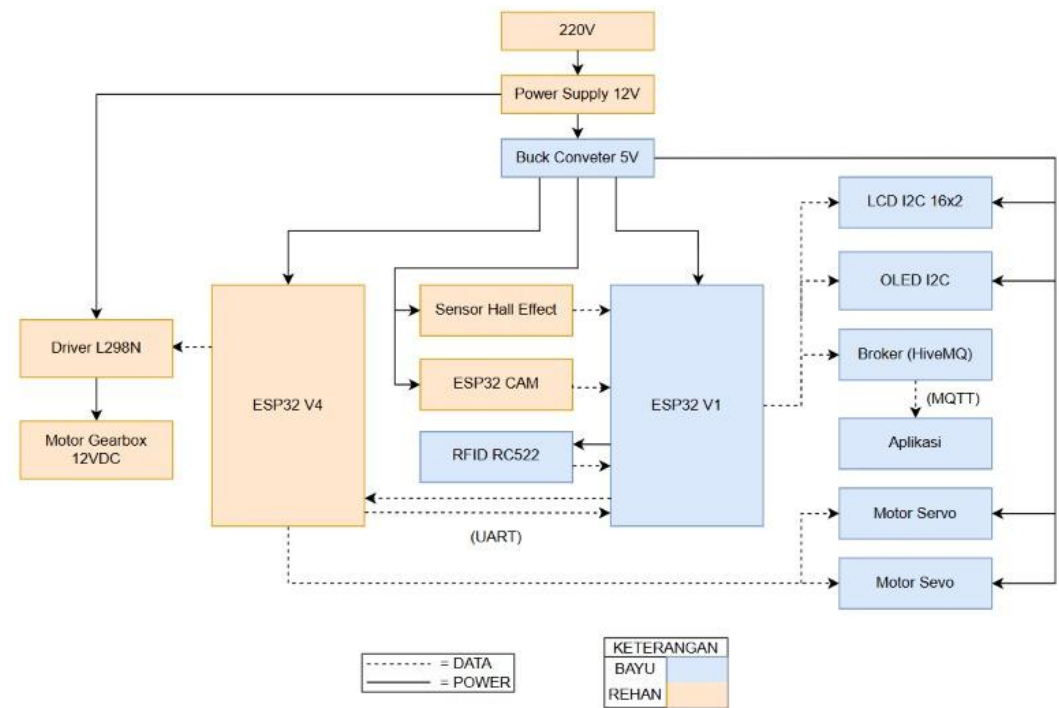


Gambar 3. 1 Diagram Alur Proses Perancangan

Dalam proses implementasi *Smart Rotary Parking*, penting untuk menetapkan tujuan dan sasaran yang ingin dicapai. Hal ini dapat dilihat pada gambar 3.1, seperti meningkatkan efisiensi, mengurangi waktu tunggu, dan mengoptimalkan penggunaan ruang parkir. Selanjutnya, kita harus membuat konsep desain sistem yang mencakup *layout*, kapasitas, jenis kendaraan yang dapat ditampung, serta teknologi yang digunakan seperti sensor, sistem kontrol, dan perangkat lunak manajemen area parkir.

3.2. Blok Diagram

Dalam perancangan *Smart Rotary Parking*, sistem dibagi menjadi dua bagian fokus penelitian utama yang saling terintegrasi secara penuh. Diagram blok pada bagian ini menguraikan arsitektur sistem kelistrikan, aliran data, serta pembagian tugas pengembangan antara penulis (Rehan) dan rekan tim (Bayu). Pemetaan ini bertujuan untuk mempermudah analisis hubungan antar-komponen serta melacak jalur komunikasi dari komponen masukan hingga keluaran sistem.:



Gambar 3. 2 Diagram Blok Sistem

Berdasarkan diagram blok Gambar 3.2 di atas, dengan pembagian tugas berdasarkan warna. Untuk warna biru untuk bagian dari Bayu Satria berupa beberapa input maupun tampilan HMI dan sistem IoT, sedangkan warna oranye/cream untuk Penulis (Rehan) yang berfokus pada sistem rotary dan *image processing* sumber daya utama sistem berasal dari tegangan 220V AC yang kemudian diturunkan menjadi 12V DC melalui komponen *Power Supply* 12V. Tegangan keluaran 12V DC ini kemudian memiliki dua percabangan jalur utama. Jalur pertama didistribusikan langsung ke Driver L298N untuk menyuplai tenaga mekanik secara maksimal ke Motor Gearbox 12VDC, sementara jalur kedua dialirkan ke modul Buck Converter 5V untuk diturunkan tegangannya guna

menyuplai daya komponen logika. Tegangan 5V hasil konversi tersebut digunakan untuk menghidupkan mikrokontroler ESP32 V4 dan ESP32 V1, beserta seluruh perangkat *input-output* yang meliputi Sensor Hall Effect, ESP32 CAM, RFID RC522, LCD I2C 16x2, OLED I2C, dan kedua unit Motor Servo. Pemisahan topologi jalur daya antara penggerak mekanik dan komponen kendali ini dirancang secara khusus untuk menjaga kestabilan logika sistem dan mencegah terjadinya kelistrikan tidak stabil (*voltage drop* atau *noise induktif*) saat beban motor DC memuncak.

Sistem kendali terdistribusi pada purwarupa ini mengandalkan komunikasi serial UART antara dua mikrokontroler utama untuk mencegah terjadinya beban komputasi berlebih (*overload*) pada satu prosesor. Bagian penulis (Rehan) ditandai dengan blok keterangan berwarna oranye yang berfokus pada sistem penggerak dan kontrol mekanik *rotary*. Bagian ini mencakup pemrosesan citra slot kosong oleh ESP32 CAM dan pencacahan posisi absolut oleh Sensor Hall Effect sebagai komponen masukan pembaca ruang parkir. Selain itu, terdapat ESP32 V4 yang bertindak sebagai mikrokontroler aktuator (*slave*), yang bertugas menerjemahkan dan meneruskan perintah untuk mengendalikan Driver L298N agar dapat memutar maupun menghentikan Motor Gearbox 12VDC secara presisi.

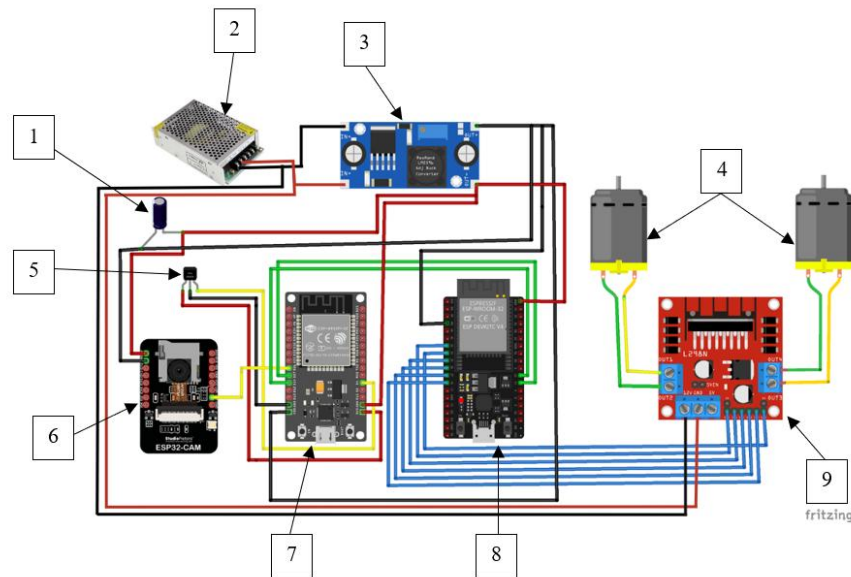
Sementara itu, bagian rekan tim (Bayu) ditandai dengan blok keterangan berwarna biru yang berfokus pada sistem akses gerbang, antarmuka pengguna (HMI), dan pemantauan *Internet of Things* (IoT). Bagian ini dikendalikan oleh ESP32 V1 sebagai pemroses logika utama (*master*). Mikrokontroler ESP32 V1 bertugas menerima dan menghimpun seluruh aliran data masukan lintas bagian, yaitu dari modul RFID RC522 miliknya sendiri, serta dari Sensor Hall Effect dan ESP32 CAM milik penulis. Hal ini menempatkan ESP32 V1 sebagai pusat pengambil keputusan (*decision maker*) dari keseluruhan alur otomasi sistem.

Hubungan integrasi antara bagian penulis dan rekan tim terlihat jelas pada alur sinkronisasi kerjanya yang membentuk sistem siklus tertutup (*closed-loop*). Operasional sistem dimulai ketika ESP32 V1 memvalidasi data masukan dari RFID RC522. Apabila sistem mendeteksi kebutuhan untuk memutar keranjang parkir, ESP32 V1 (Master) akan mengirimkan instruksi via UART kepada ESP32 V4

(Slave) untuk segera dieksekusi oleh sistem penggerak mekanik. Selama motor pada bagian penulis berputar, pulsa magnetik dari Sensor Hall Effect dan hasil pemrosesan status citra dari ESP32 CAM akan dikirimkan kembali secara terus-menerus (*real-time*) sebagai data umpan balik (*feedback*) kepada ESP32 V1. Berdasarkan data umpan balik sensor dari bagian penulis inilah, ESP32 V1 dapat memberikan komando berhenti secara akurat. Setelah memvalidasi dan memproses bahwa keranjang telah berada pada posisi yang tepat, ESP32 V1 akan menampilkan status alat secara lokal pada LCD I2C 16x2 dan OLED I2C, mengirimkan data pembaruan ke Broker (HiveMQ) melalui protokol MQTT agar dapat dipantau melalui Aplikasi, serta mengatur pergerakan Motor Servo untuk membuka palang pintu masuk atau keluar

3.3. Perancangan Elektrikal

Perancangan elektrikal sistem prototipe ini menguraikan integrasi perkabelan antar-komponen, distribusi daya, serta jalur pertukaran data antarmuka kendali. Sistem ini menerapkan arsitektur *dual-microcontroller* menggunakan dua unit ESP32 guna memisahkan beban komputasi pemrosesan citra dengan kendali gerak mekanik motor secara mandiri. Penataan jalur perkabelan, pembagian kabel catu daya, dan koneksi antarmuka pin kontrol antar-modul disajikan secara menyeluruh pada gambar berikut :



Gambar 3. 3 Rangkaian Skematik Alat

Tabel 3. 1 Daftar komponen rangkaian skematik

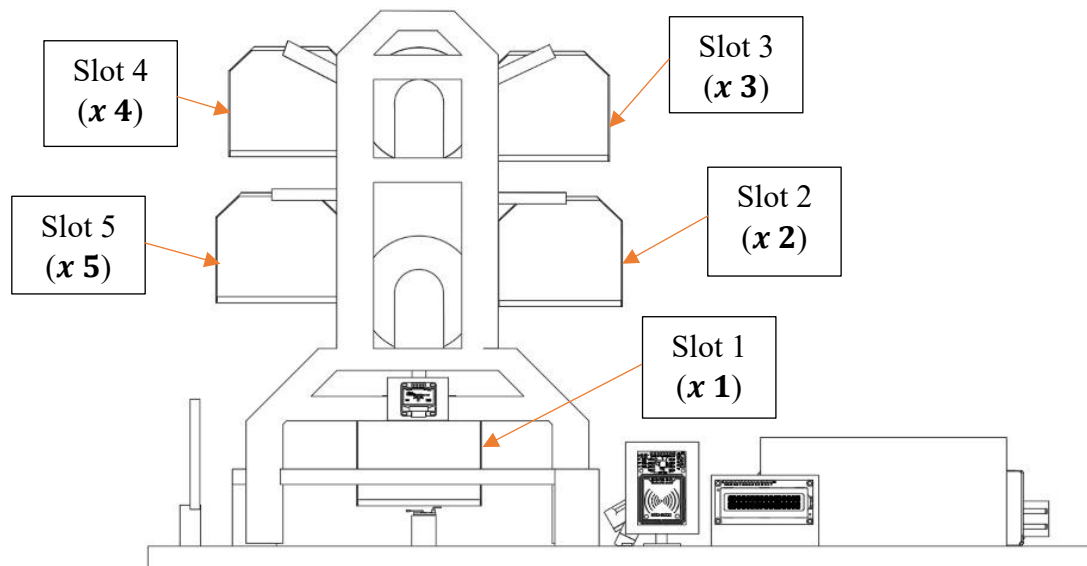
Nomor	Komponen
1	Kapasitor 1000 μ F
2	Power Supply 12VDC
3	DC Buck Converter
4	Motor DC 12V
5	Hall Effect Sensor
6	ESP32 CAM
7	ESP32 Devkit 1 (Master)
8	ESP32 Devkit 4 (Slave)
9	Driver Motor L298N

Berdasarkan Gambar 3. 3, tegangan utama 12V DC dari adaptor dialirkan ke modul *step-down converter* LM2596 untuk diturunkan menjadi 5V DC teregulasi guna menyuplai daya pin VIN ESP32 Master, ESP32 Slave, modul ESP32-CAM, serta sensor *Hall Effect* A3144 dengan rel *ground* yang digabungkan bersama (*common ground*). Pada jalur daya modul ESP32-CAM, dipasang kapasitor elektrolit 1000 μ F secara paralel antara rel VCC dan GND yang berfungsi sebagai penyangga tegangan lokal untuk meredam lonjakan arus sesaat saat kamera OV2640 dan pemancar data aktif sehingga mikrokontroler tidak mengalami penurunan tegangan (*voltage brownout*). ESP32 Master bertindak sebagai pengendali utama yang membaca pulsa magnetik dari sensor *Hall Effect* melalui

pin GPIO interupsi serta bertukar data secara serial UART dengan ESP32-CAM untuk verifikasi ketersediaan slot kosong. Setelah selisih langkah perpindahan (Δx) ditentukan, ESP32 Master mengirimkan instruksi ke ESP32 Slave via komunikasi serial UART, kemudian ESP32 Slave mengalirkan enam sinyal kontrol logika/PWM menuju modul driver L298N (ENA, IN1, IN2, IN3, IN4, ENB) untuk mengatur arah putaran serta modulasi kecepatan dua motor DC gearbox JG25-370 12V secara presisi saat memosisikan keranjang parkir.

3.4. Perancangan Mekanik

Perancangan mekanik prototipe *smart rotary parking* mengadopsi mekanisme sirkulasi vertikal yang dirancang untuk mengoptimalkan efisiensi lahan parkir. Rangka utama prototipe menggunakan struktur akrilik dan profil aluminium guna menopang 5 buah unit keranjang gantung. Seluruh keranjang ditopang oleh penopang kayu yang menempel pada *timing belt* yang digerakkan oleh sepasang motor DC gearbox JG25-370 12V dengan reduksi rasio putaran tinggi untuk menghasilkan torsi stabil. Setiap keranjang dirancang dengan sistem penggantung bebas berbasis bantalan (*bearing*) agar posisi lantai keranjang tetap horizontal secara alami akibat gaya gravitasi saat mekanisme berputar.



Gambar 3. 4 Posisi Slot pada posisi awal

Berdasarkan perancangan 3D rangka vertikal, mekanisme prototipe menempatkan Slot 1 pada posisi paling bawah sebagai titik acuan awal (*Home*

Position) yang berhadapan langsung dengan akses pintu gerbang masuk/keluar, bidang pandang lensa ESP32-CAM, serta sensor *Hall Effect*. Urutan penomoran slot bergerak secara berurutan mengikuti arah sirkulasi gerak, dimulai dari Slot 2 di sisi kanan bawah, Slot 3 di sisi kanan atas menuju puncak rotasi, Slot 4 di sisi kiri atas pada lintasan menurun, hingga Slot 5 di sisi kiri bawah tepat satu langkah sebelum kembali ke posisi dasar. Orientasi lintasan ini membentuk perputaran siklik tertutup satu arah ($1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 1$), sehingga perhitungan langkah perpindahan dari posisi awal (x_1) ke posisi target (x_2) selalu menghasilkan langkah maju positif, termasuk perpindahan dari Slot 5 ke Slot 1 yang dihitung tepat 1 langkah ($\Delta x = 1$). Sistem penopang ganda dengan dua poros vertikal dan gantungan berbasis bantalan bebas menjaga pelat lantai setiap keranjang tetap datar secara alami akibat gaya gravitasi, sementara motor DC gearbox JG25-370 dengan reduksi rasio tinggi mampu memberikan torsi penahan yang stabil untuk mengatasi perbedaan beban antara sisi kanan yang menaik dan sisi kiri yang menurun saat sensor *Hall Effect* memicu penghentian posisi di depan gerbang.

3.4.1 Realisasi Kerangka dan Mekanik Rotary

Kerangka utama penyusun prototipe ini menggunakan bahan kayu jati belanda material yang dirakit secara presisi untuk menopang seluruh beban dinamis saat sistem beroperasi. Pada bagian dasar alat, digunakan papan alas tripleks datar yang dilapisi stiker berwarna putih yang berukuran cukup lebar. Papan dasar ini berfungsi untuk mendistribusikan beban secara merata ke permukaan lantai, sehingga maket tidak mudah bergeser atau terguling saat beban mekanik sedang berputar.



Gambar 3. 5 Realisasi prototipe

Di atas papan dasar tersebut, didirikan rangka tegak lurus berwarna hitam yang terbuat dari kayu yang bertindak sebagai tiang tumpuan utama (*main support frame*). Struktur rangka ini dirancang sedemikian rupa untuk memiliki kekakuan. Berdasarkan hasil realisasi akhir, purwarupa ini memiliki dimensi keseluruhan dengan panjang 63 cm, lebar 43 cm, dan tinggi 48 cm.

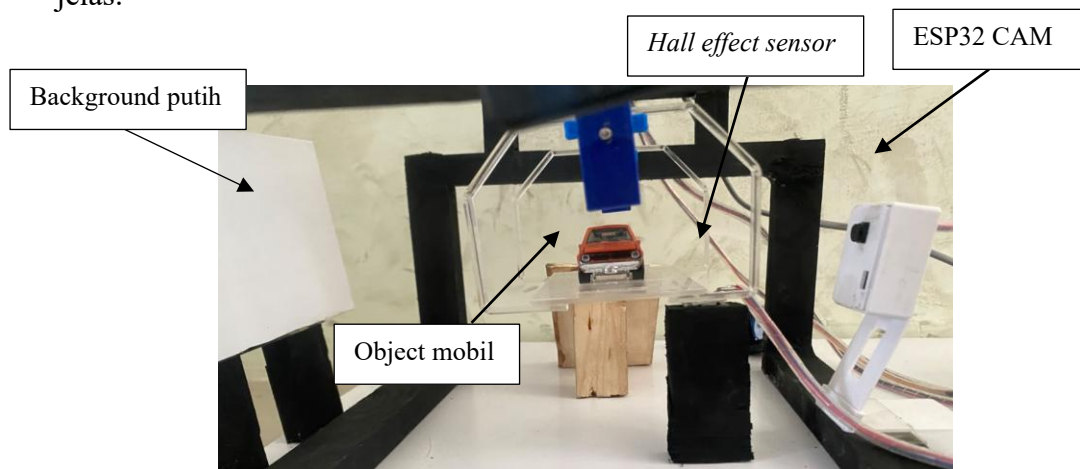
Mekanisme perputaran vertikal ditopang oleh poros utama berbahan logam (*metal shaft*) dengan Panjang 30cm yang melintang secara horizontal menembus kedua sisi rangka utama. Poros ini dihubungkan secara langsung ke aktuator penggerak, yaitu sebuah motor DC *gearbox* 12V yang terpasang kokoh pada bagian atas rangka. Pemilihan peletakan motor di bagian atas struktur bertujuan untuk mengoptimalkan transfer torsi secara efisien ke poros pemutar dan memudahkan proses pemeliharaan. Digunakan komponen pemutar yaitu *pulleys* dengan diameter 10 cm yang disambungkan dengan belt dengan keliling 52cm dan lebar 6 cm.

Struktur lengan pemutar direalisasikan menggunakan material penyangga silang berwarna biru yang terikat kuat pada poros utama. Pada ujung masing-masing lengan tersebut, dipasangkan as atau poros gantung kecil sebagai tempat meletakkan slot parkir (keranjang) yang terbuat dari bahan akrilik transparan. Pemasangan keranjang akrilik ini menggunakan sistem engsel gantung. Sistem mekanik ini memastikan orientasi rantai keranjang parkir akan selalu berada dalam keadaan mendatar (sejajar dengan tanah) akibat gaya gravitasi bumi, terlepas dari bagaimana posisi lengan *rotary* sedang berputar. Selain itu, penggunaan material

akrilik yang transparan sengaja dipilih untuk meminimalisir halangan cahaya masuk, sehingga mempermudah kerja lensa ESP32-CAM dalam memindai objek kendaraan di dalam slot parkir secara visual.

3.4.2 Realisasi Modul Visi ESP32-CAM

Untuk mendeteksi apakah sebuah slot parkir sedang kosong atau sudah ada mobilnya, sistem ini menggunakan modul kamera ESP32-CAM (dengan sensor OV2640 2MP). Modul kamera ini diposisikan diam di bagian bawah rangka alat, dengan lensa yang mengarah langsung ke keranjang parkir yang sedang berada di titik X 1 (titik akses keluar-masuk kendaraan). Pemilihan posisi di bawah ini sangat menguntungkan karena kamera bisa melihat langsung kondisi di dalam keranjang. Apalagi keranjang parkir pada alat ini dibuat menggunakan bahan akrilik bening (transparan). Bahan akrilik ini sangat membantu kerja kamera karena cahaya dari sekitar alat bisa masuk dengan mudah, sehingga tidak banyak bayangan gelap yang menutupi mobil miniatur di dalamnya. Hasil tangkapan gambar pun menjadi lebih jelas.



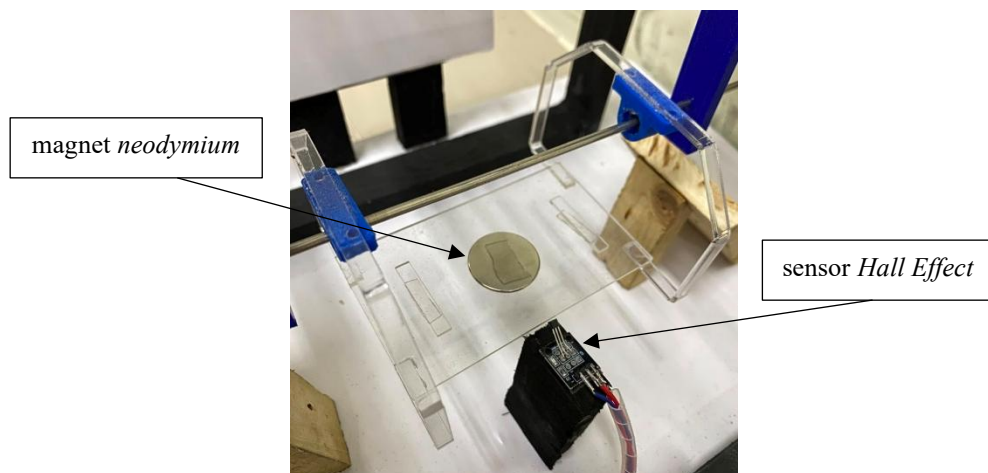
Gambar 3. 6 Penempatan angle kameran dan background

Agar gambar yang dihasilkan tidak buram, modul ESP32-CAM dipasang padaudukan yang kuat dan tidak ikut bergetar saat motor *rotary* di atas alat sedang berputar. Kalau kamera ikut bergetar, gambar bisa menjadi blur (*motion blur*) dan membuat sistem salah menebak kondisi slot. Selain itu, jarak antara kamera dan keranjang juga sudah diatur sedemikian rupa agar pas dengan titik fokus lensa. Dengan jarak yang pas ini, kotak area pembacaan gambar (*Region of Interest / ROI*)

yang diproses oleh program hanya akan terfokus pada lantai keranjang saja. Hal ini sangat penting agar tiang atau lengan penyangga alat tidak ikut masuk ke dalam area pembacaan yang bisa membuat kamera salah mendeteksi status "Kosong" (K) atau "Terisi" (T).

3.4.3 Realisasi Penempatan Sensor Hall Effect dan Magnet

Agar keranjang parkir bisa berhenti dengan presisi atau pas di depan pintu akses keluar-masuk, sistem ini menggunakan kombinasi sensor *Hall Effect* dan magnet *neodymium*. Sensor *Hall Effect* berfungsi sebagai saklar magnetik yang akan memberikan sinyal ketika ada medan magnet yang mendekat. Pada alat ini, modul sensor *Hall Effect* dipasang secara permanen pada tiang penyangga yang berada paling bawah dari rangkaian seperti pada Gambar 3.7. Sementara itu, magnet *neodymium* ditempelkan pada masing-masing bawah slot di bawah akrilik agar dapat mengenai langsung sensor *Hall Effect* seperti Gambar 3.7 berikut.



Gambar 3. 7 Penempatan Magnet

Jarak celah antara sensor di penyangga dan magnet di lengan pemutar diatur sangat dekat, yaitu kurang dari 0,5 cm. Pengaturan jarak yang rapat ini sangat penting agar saat keranjang berputar dan melintas di depan sensor, medan magnetnya cukup kuat untuk dideteksi tanpa adanya pembacaan yang terlewat. Setiap kali magnet melewati sensor, sensor akan langsung mengirimkan sinyal digital ke mikrokontroler ESP32 Master. Sinyal inilah yang diolah oleh program sebagai penghitung (*counter*) untuk mengetahui posisi keranjang nomor berapa yang sedang berada di bawah. Dengan acuan data dari sensor ini, mikrokontroler

bisa memerintahkan dinamo untuk mengerem tepat waktu, sehingga keranjang tidak berhenti meleset dari posisi yang seharusnya.

3.4.4 Realisasi Panel Kelistrikan dan Pengkabelan (Box Kontrol)

Seluruh komponen "otak" dari alat ini, seperti mikrokontroler, modul pengendali motor, dan sistem pembagi daya listrik, dikumpulkan dan disusun rapi di dalam sebuah kotak panel kontrol. Penempatan di dalam kotak ini bertujuan untuk melindungi komponen elektronik dari debu, kotoran, sentuhan fisik dari luar, serta mencegah terjadinya korsleting akibat kabel yang berantakan.

Di dalam panel kontrol ini, jalur kelistrikan sengaja dibagi menjadi dua jalur utama agar sistem bisa bekerja dengan stabil dan tidak saling mengganggu. Jalur Daya Dinamo (12 VDC) dengan Tegangan listrik sebesar 12 Volt dari adaptor disalurkan secara langsung menuju *Driver Motor* L298N. Jalur ini dikhususkan untuk menyuplai tenaga ke dinamo motor DC *gearbox* karena motor membutuhkan arus yang cukup besar, terutama saat mulai berputar awal (*starting*). Jalur Logika dan Sensor (5 VDC), komponen seperti ESP32 tidak bisa menerima tegangan 12 Volt secara langsung. Oleh karena itu, tegangan 12 Volt diturunkan terlebih dahulu menggunakan modul *Step-Down* LM2596 hingga menjadi 5 Volt yang stabil. Tegangan 5 Volt inilah yang kemudian digunakan untuk menghidupkan ESP32 Master, ESP32 Slave, kamera ESP32-CAM, layar, dan sensor *Hall Effect*.

Untuk menghubungkan komunikasi antar-mikrokontroler (kabel TX dan RX), digunakan kabel *jumper* yang ditata serapi mungkin. Hal yang paling penting dalam perakitan kelistrikan ini adalah menyatukan seluruh titik *Ground* (GND) dari semua komponen menjadi satu jalur (*common ground*). Penyatuan titik *ground* ini wajib dilakukan agar level tegangan antar komponen menjadi sama, sehingga pertukaran data serial antara Master, Slave, dan Kamera bisa berjalan lancar tanpa mengalami *error*.

3.5 Hasil Perancangan Program Pada Alat

Peneliti dalam perancangan program ini menggunakan perangkat lunak Arduino IDE sebagai aplikasi utama untuk mengembangkan sistem kendali pada alat yang inovatif ini. Alat ini dirancang dengan tujuan untuk memudahkan pengguna maupun sistem dalam melakukan tugas otomasi parkir secara presisi dan

efisien. Melalui penggunaan Arduino IDE, peneliti dapat membuat dan menyusun kode program (berbasis bahasa C/C++) yang memungkinkan alat ini berfungsi dengan baik. Karena arsitektur sistem ini dibagi menjadi dua bagian, program juga dipecah menjadi dua *file* terpisah untuk masing-masing mikrokontroler, yaitu kode untuk ESP32 *Master* (berfokus pada pembacaan sensor dan logika visual) dan ESP32 *Slave* (berfokus pada eksekusi kendali motor).

3.5.1 Konfigurasi Pin

Konfigurasi Pin adalah proses mengatur dan menghubungkan koneksi elektrik antara alat Smart Rotary Parking dengan perangkat lainnya. Dalam tabel 3.2 ini terdapat informasi mengenai fungsi dan konfigurasi pin pada alat tersebut:

Tabel 3. 2 Konfigurasi Pin

Mikrokontroler	Komponen	Pin Input / Output	Fungsi
ESP32 (Master)	Sensor <i>Hall</i>	Pin 4 (Input)	Mendeteksi lintasan magnet untuk menghitung posisi absolut <i>platform</i> parkir secara presisi.
	Komunikasi <i>Serial</i> (ke ESP32-CAM)	Pin 26 (UART)	Mengirimkan perintah <i>trigger</i> dan menerima hasil deteksi visual ketersediaan slot.
	Komunikasi <i>Serial</i> (ke ESP32 Slave)	Pin 17 (TX2)	Mengirimkan instruksi pergerakan mekanik dan jumlah slot.
ESP32 (Slave)	Komunikasi <i>Serial</i> (ke ESP32 Master)	Pin 16 (RX2)	Mengirimkan konfirmasi dari output yang telah dilakukan.
	Driver L298N (ENA)	Pin 32	Mengontrol dan membatasi kecepatan maksimal dinamo motor DC agar perputaran keranjang tetap stabil.
	Driver L298N (IN1)	Pin 25	Mengontrol arah putaran dinamo motor DC (searah jarum jam / <i>Forward</i>).

Driver L298N (IN2)	Pin 26	Mengontrol arah putaran dinamo motor DC (berlawanan jarum jam / <i>Reverse</i>).
Driver L298N (ENB)	Pin 33	Mengontrol dan membatasi kecepatan maksimal dinamo motor DC agar perputaran keranjang tetap stabil.
Driver L298N (IN3)	Pin 27	Mengontrol arah putaran dinamo motor DC (searah jarum jam / <i>Forward</i>).
Driver L298N (IN4)	Pin 14	Mengontrol arah putaran dinamo motor DC (berlawanan jarum jam / <i>Reverse</i>).

Keberhasilan operasional prototipe *Smart Rotary Parking* sangat bergantung pada tingkat akurasi dalam memetakan konfigurasi pin. Apabila setiap pin dialokasikan dan disambungkan secara presisi sesuai dengan peruntukannya, keseluruhan sistem alat ini dijamin dapat bekerja dengan stabil. Ketepatan instalasi kelistrikan ini pada akhirnya mampu menghadirkan sebuah mekanisme parkir yang efisien, andal, serta menjamin tingkat keamanan tertinggi bagi para penggunanya.

3. 5. 2. Implementasi Program Smart Rotary Parking

Perancangan logika dasar pada program ini bertujuan untuk mengendalikan sistem perparkiran agar beroperasi secara otomatis dan efisien. Rincian penjelasan untuk setiap bagian sintaks atau instruksi program yang terlampir dan dapat dilihat pada Tabel 3.3 berikut:

Tabel 3. 3 Logika Dasar Program Master

Baris	Logika Dasar Program Master
1-9	Mengimpor <i>library</i> yang diperlukan untuk program, seperti SPI.h dan MFRC522.h untuk modul RFID, WiFi.h dan PubSubClient.h untuk komunikasi IoT/MQTT, serta Wire.h , LiquidCrystal_I2C.h , Adafruit_GFX.h , dan

	Adafruit_SSD1306.h untuk mengontrol antarmuka layar LCD dan OLED.
10-24	Mendefinisikan parameter konfigurasi WiFi dan <i>broker</i> MQTT (mqtt_server , mqtt_port , mqtt_topic). Selain itu, mendeklarasikan objek WiFiClient dan variabel string untuk menyimpan status akses yang akan dipublikasikan ke aplikasi.
25-37	Mendeklarasikan pin yang terhubung ke perangkat keras. SS_PIN dan RST_PIN untuk modul RFID, serta pinHall untuk mendeteksi putaran keranjang. Bagian ini juga menetapkan pin RX/TX kustom untuk jalur komunikasi Serial1 (ke ESP32-CAM) dan Serial2 (ke ESP32 <i>Slave</i>).
38-51	Menetapkan konfigurasi antarmuka I2C kustom pada pin 13 dan 22. Mendeklarasikan objek lcd dengan alamat <i>0x27</i> dan objek oled untuk layar SSD1306 beresolusi 128x64 dengan alamat <i>0x3C</i> .
52-70	Menginisialisasi <i>database array</i> slotMemory untuk mengelola ketersediaan 6 slot parkir. Mendeklarasikan berbagai variabel <i>flag</i> dan penanda waktu (<i>millis</i>) untuk melacak status proses masuk/keluar serta status pembacaan RFID.
72-126	Memuat kumpulan fungsi jaringan dan komunikasi. Fungsi kirimKeSlave() mengirim perintah ke mikrokontroler aktuator. Fungsi setupWiFi() menangani koneksi <i>router</i> , sedangkan publishStatus() dan mqttReconnect() bertugas menyusun data berformat JSON lalu mengirimkannya secara <i>real-time</i> ke <i>broker</i> MQTT.
128-181	Memuat kumpulan fungsi antarmuka pengguna (UI). jagaRFID() untuk memeriksa kerusakan pembacaan pada sensor kartu secara otomatis. tampilLCD() , lcdIdleCheck() , dan updateOLED() mengontrol logika pembaruan layar (<i>refresh</i>) yang dirancang anti-kedip agar layar hanya diperbarui jika terdapat perubahan data.
183-238	Memuat fungsi prosesCheckIn() dan prosesCheckOut() . Fungsi ini memvalidasi apakah UID kartu sudah terdaftar di slotMemory atau belum. Jika validasi berhasil, status sistem diubah untuk memicu motor dan menunggu konfirmasi ruang parkir.
240-279	Fungsi setup() yang akan dieksekusi sekali saat sistem dinyalakan. Berfungsi untuk menginisialisasi kecepatan seluruh jalur komunikasi <i>Serial</i> , menyalakan sensor RFID melalui bus SPI, mengatur koneksi I2C untuk layar, menyambungkan jaringan WiFi, dan mencetak pesan siaga awal ke layar.

281-402	Fungsi loop() yang berjalan secara terus-menerus. Bertugas menjaga kestabilan koneksi MQTT, membaca input transisi dari sensor <i>Hall Effect</i> untuk menghitung pergerakan fisik slot, menerjemahkan sinyal visual ('K/'T') dari ESP32-CAM, dan mengeksekusi logika <i>tap</i> kartu
---------	--

Tabel 3. 4 Logika Dasar Program Slave

Baris	Logika Dasar Program Slave
1-19	Mengimpor <i>library</i> utama Arduino.h dan ESP32Servo.h . Mendefinisikan pin komunikasi RX_MASTER dan TX_MASTER , serta mendeklarasikan pin perangkat keras untuk pengoperasian dua sisi motor DC (Kiri: ENA, IN1, IN2 ; Kanan: ENB, IN3, IN4) dan pin pengendali motor servo (PIN_SERVO_IN, PIN_SERVO_OUT).
21-26	Mendeklarasikan objek servoMasuk dan servoKeluar , serta menetapkan konstanta sudut posisi gerbang yaitu SUDUT_TUTUP (0°) dan SUDUT_BUKA (90°)
28-66	Fungsi setup() yang dieksekusi sekali saat sistem dinyalakan. Menginisialisasi komunikasi Serial Monitor dan Serial2 dengan <i>baud rate</i> 9600, menetapkan seluruh pin motor DC sebagai OUTPUT dengan kondisi awal mati (LOW), mengalokasikan timer ESP32 untuk servo agar sistem tidak mengalami <i>crash/restart</i> , serta mengunci posisi kedua gerbang di sudut tertutup.
68-91	Fungsi loop() yang berjalan secara terus-menerus memantau aliran data dari kabel komunikasi Master melalui fungsi Serial2.available(). Karakter yang masuk dibaca per karakter dan disatukan hingga mendeteksi karakter baris baru (\n), kemudian dibersihkan menggunakan fungsi .trim() sebelum diteruskan ke fungsi eksekusi.
93-138	Fungsi eksekusiPerintah() yang berisi kumpulan logika kondisional (<i>if-else</i>) untuk menterjemahkan instruksi string dari <i>Master</i> : " MOTOR_ON ": Mengaktifkan pin <i>driver</i> L298N agar dinamo kiri dan kanan berputar mencari slot. " MOTOR_OFF ": Mematikan pin <i>driver</i> secara mendadak (<i>fast stop</i>) saat slot ditemukan. " GATE_IN " / " GATE_OUT ": Menggerakkan motor servo ke sudut buka, menahannya selama 5 detik menggunakan fungsi <i>delay</i> , lalu menutupnya kembali secara otomatis.

Tabel 3. 5 Logika Dasar Program ESP32 CAM

Baris	Logika Dasar Program ESP32-CAM
1-18	Mengimpor <i>library</i> utama <code>esp_camera.h</code> dan mendefinisikan keseluruhan pemetaan pin perangkat keras (GPIO) yang terhubung ke modul kamera jenis AI-Thinker.
20-23	Mendeklarasikan parameter utama metode pemrosesan citra. Menentukan koordinat pemotongan area fokus pembacaan (<i>Region of Interest</i> / ROI). Selain itu, menetapkan variabel <code>BATAS_THRESHOLD = 170</code> sebagai garis batas intensitas pemisah warna gelap/terang, serta <code>BATAS_PERSentase = 50.0</code> sebagai syarat minimum persentase luasan objek untuk validasi ketersediaan slot.
34-60	Fungsi <code>setup()</code> yang dieksekusi satu kali saat alat dihidupkan. Menginisialisasi komunikasi Serial pada <i>baud rate</i> 115200 bps untuk pengiriman data ke Master, serta mengatur konfigurasi kamera ke resolusi QQVGA (160x120 piksel) dengan format <code>PIXFORMAT_GRAYSCALE</code> guna menekan beban komputasi dan memori RAM. Jika kamera gagal diinisialisasi, sistem diprogram untuk melakukan <i>restart</i> secara otomatis setelah 3 detik.
62-79	Fungsi <code>loop()</code> diawali dengan mengambil tangkapan gambar (<i>frame</i>) ke dalam <i>buffer</i> memori fb. Sistem melakukan perulangan bersarang (<i>nested loop</i>) sumbu X dan Y untuk memindai piksel secara spesifik hanya di dalam area kotak ROI. Koordinat 2D kemudian dikonversi menjadi indeks <i>array</i> 1D untuk mengekstrak nilai intensitas warna piksel tunggal (0 = Hitam Murni, 255 = Putih Murni).
81-87	Menerapkan metode <i>Inverted Thresholding</i> . Jika nilai piksel yang terbaca lebih kecil dari <code>BATAS_THRESHOLD (< 170)</code> , maka titik tersebut dihitung dan diakumulasikan sebagai objek gelap (bodi mobil/penghalang). Setelah pemindaian piksel selesai, memori tangkapan gambar segera dibebaskan menggunakan perintah <code>esp_camera_fb_return(fb)</code> guna mencegah modul kamera mengalami kehabisan memori (<i>crash</i> / <i>brownout</i>).
89-99	Tahap klasifikasi akhir (<i>Pixel Counting</i>). Sistem menghitung persentase kumpulan piksel gelap terhadap total luas area ROI. Apabila nilai tersebut mencapai 50% atau lebih, maka ditetapkan karakter 'T' (Terisi), dan jika sebaliknya ditetapkan karakter 'K' (Kosong). Karakter ini kemudian dikirimkan ke ESP32 Master melalui jalur Serial dengan diakhiri karakter <i>newline</i> (<code>println</code>), diikuti oleh jeda waktu eksekusi (<i>delay</i>) selama 300 ms untuk mencegah penumpukan data.

3.6. Metode Pemrosesan Citra pada ESP32-CAM

Pada penelitian ini, metode pengolahan citra yang digunakan adalah Metode Pemrosesan Citra Ringan (*Lightweight Image Processing*) berbasis *Thresholding* dan Analisis Region of Interest (ROI) menggunakan teknik Penghitungan Piksel (*Pixel Counting*). Pendekatan ini dipilih karena menyesuaikan dengan keterbatasan memori dan komputasi pada modul ESP32-CAM, sehingga tidak memungkinkan penggunaan metode kecerdasan buatan yang kompleks seperti deep learning. Proses pemrosesan citra pada sistem ini dilakukan secara berkesinambungan, di mana kamera OV2640 diprogram untuk menangkap gambar secara berkala dengan interval setiap satu detik. Resolusi citra diatur pada tingkat yang relatif rendah, seperti QQVGA (160×120) atau QVGA (320×240), dengan tujuan meminimalkan beban memori buffer agar siklus pengolahan citra berjalan lancar tanpa kelambatan (*lag*) dan sistem selalu mendapatkan pembaruan kondisi slot parkir secara *real-time*.



Gambar 3. 4 Hasil dari Thresholding

Setelah citra berhasil ditangkap pada setiap detiknya, sistem langsung menjalankan tahap konversi dari citra berwarna (RGB) menjadi citra skala abu-abu (*grayscale*). Konversi ini sangat penting untuk menekan beban memori secara drastis, mengingat citra RGB membutuhkan tiga komponen warna (merah, hijau, biru) yang memakan total 3 byte per piksel, sedangkan citra *grayscale* hanya membutuhkan 1 byte per piksel. Pengurangan ukuran data ini dapat dilakukan tanpa menghilangkan informasi bentuk objek kendaraan. Selanjutnya, citra abu-abu tersebut melalui proses *thresholding* atau binerisasi untuk mengubahnya menjadi murni hitam dan putih. Sistem bekerja dengan menentukan nilai ambang batas intensitas cahaya (*threshold*), di mana piksel yang lebih terang dari batas tersebut diubah menjadi putih (255), sedangkan yang lebih gelap diubah menjadi hitam (0).

Tahap ini sangat krusial untuk memisahkan objek kendaraan, yang nantinya tampak sebagai area berpiksel putih, dari latar belakangnya.

Untuk mempercepat kinerja agar pembacaan tiap detik tidak memberatkan prosesor, analisis tidak dilakukan pada keseluruhan bingkai gambar, melainkan dibatasi hanya pada koordinat kotak slot parkir saja melalui tahap Region of Interest (ROI). Di dalam area pembacaan yang telah dipotong (crop) tersebut, sistem menerapkan teknik pixel counting untuk menghitung jumlah persentase kepadatan piksel putih. Nilai persentase ini kemudian dibandingkan dengan ambang batas kepadatan yang telah ditentukan, misalnya 70%. Jika kepadatan piksel putih melebihi ambang batas tersebut, maka pada detik itu slot disimpulkan dalam keadaan "Terisi" oleh kendaraan. Sebaliknya, jika persentase piksel putih berada di bawah batas, maka area slot dinyatakan "Kosong".



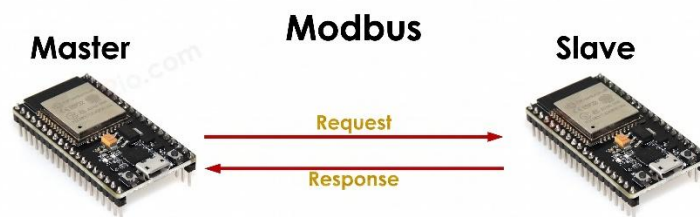
Gambar 2. 11 Hasil dari metode ROI

Hasil klasifikasi ketersediaan slot yang diperbarui setiap detiknya ini kemudian dikirimkan secara kontinu ke mikrokontroler ESP32 Master. Data real-time tersebut dijadikan acuan utama oleh Master dalam mengambil keputusan pergerakan dinamo pada sistem rotary parking. Apabila rentetan data menunjukkan status "Terisi", Master akan menginstruksikan motor untuk terus berputar mencari ruang lain. Namun, begitu status berubah menjadi "Kosong", Master akan bersiap menghentikan pergerakan tepat pada saat sensor *Hall Effect* memvalidasi bahwa posisi keranjang telah sejajar sempurna pada titik akses. Meskipun metode pemrosesan citra ringan ini sangat unggul dalam hal efisiensi memori, kemudahan implementasi, dan kecepatan deteksi, kinerjanya tetap rentan terhadap fluktuasi pencahayaan atau bayangan luar. Oleh karena itu, kondisi lingkungan dengan

cahaya yang stabil sangat diperlukan agar batas threshold dapat bekerja konsisten dalam memisahkan objek.

3.7. Arsitektur Komunikasi Serial (Master, Slave, dan ESP32-CAM)

Pada perancangan *Smart Rotary Parking* ini, sistem kendali tidak dipusatkan pada satu mikrokontroler tunggal guna mendistribusikan beban kerja komputasi (*workload distribution*). Mengingat sistem harus memproses koneksi jaringan IoT, pembacaan sensor secara presisi, pengolahan citra visual, dan pengendalian aktuator daya tinggi secara bersamaan, pemusatan tugas pada satu pusat kendali berisiko menyebabkan *delay* atau gagal sistem. Oleh karena itu, sistem dipecah menjadi tiga unit pemroses utama: ESP32 *Master* (pengendali pusat), ESP32-CAM (subsistem visi), dan ESP32 *Slave* (eksekutor mekanik). Ketiga unit ini saling bertukar informasi secara asinkron menggunakan protokol komunikasi *Universal Asynchronous Receiver-Transmitter* (UART).



Gambar 3. 8 Komunikasi Master Slave antar ESP32

Dalam pelaksanaannya, jalur komunikasi pertama terjalin antara ESP32 *Master* dan ESP32-CAM yang difokuskan untuk pertukaran data hasil deteksi slot parkir. Alur komunikasi ini dirancang menggunakan pendekatan berbasis pemicu (*trigger-based*). Saat sensor pendeteksi posisi membaca bahwa keranjang parkir telah berada tepat di depan kamera, ESP32-CAM akan menangkap gambar, menjalankan algoritma pengolahan citra, dan mengirimkan keputusan akhirnya kembali ke *Master*. Data yang dikirimkan melalui jalur serial ini berupa satu karakter tunggal: huruf 'K' apabila slot terdeteksi kosong, atau huruf 'T' apabila slot terdeteksi terisi oleh kendaraan. Karakter inilah yang kemudian dijadikan acuan oleh ESP32 *Master* untuk menghentikan atau melanjutkan perputaran *rotary*.

Tabel 3. 6 Arsitektur komunikasi sistem

Komponen Pengendali	Komponen Tujuan / Aktuator	Protokol / Jalur Komunikasi	Keterangan
ESP32 <i>Master</i>	ESP32-CAM	Serial UART (<i>Serial1</i>)	engirimkan <i>trigger</i> pemindaian dan menerima hasil deteksi visual berupa karakter biner ('K' atau 'T').
ESP32 <i>Master</i>	ESP32 <i>Slave</i>	Serial UART (<i>Serial2</i> , 9600 bps)	Pertukaran data perintah logika berbentuk <i>String</i> (contoh: "MOTOR_ON", "GATE_IN") melalui pin RX2/TX2
Sensor <i>Hall Effect</i>	ESP32 <i>Master</i>	Logika Digital (<i>I/O Digital</i>)	Mengirimkan sinyal masukan berupa logika HIGH atau LOW sebagai pemicu (<i>trigger</i>) maupun penghitung (<i>counter</i>) posisi absolut.
ESP32 <i>Slave</i>	Driver Motor L298N	PWM & Logika Digital	Mengirimkan sinyal PWM untuk mengatur batas kecepatan dinamo, serta sinyal logika digital untuk mengatur arah putaran (maju/mundur).

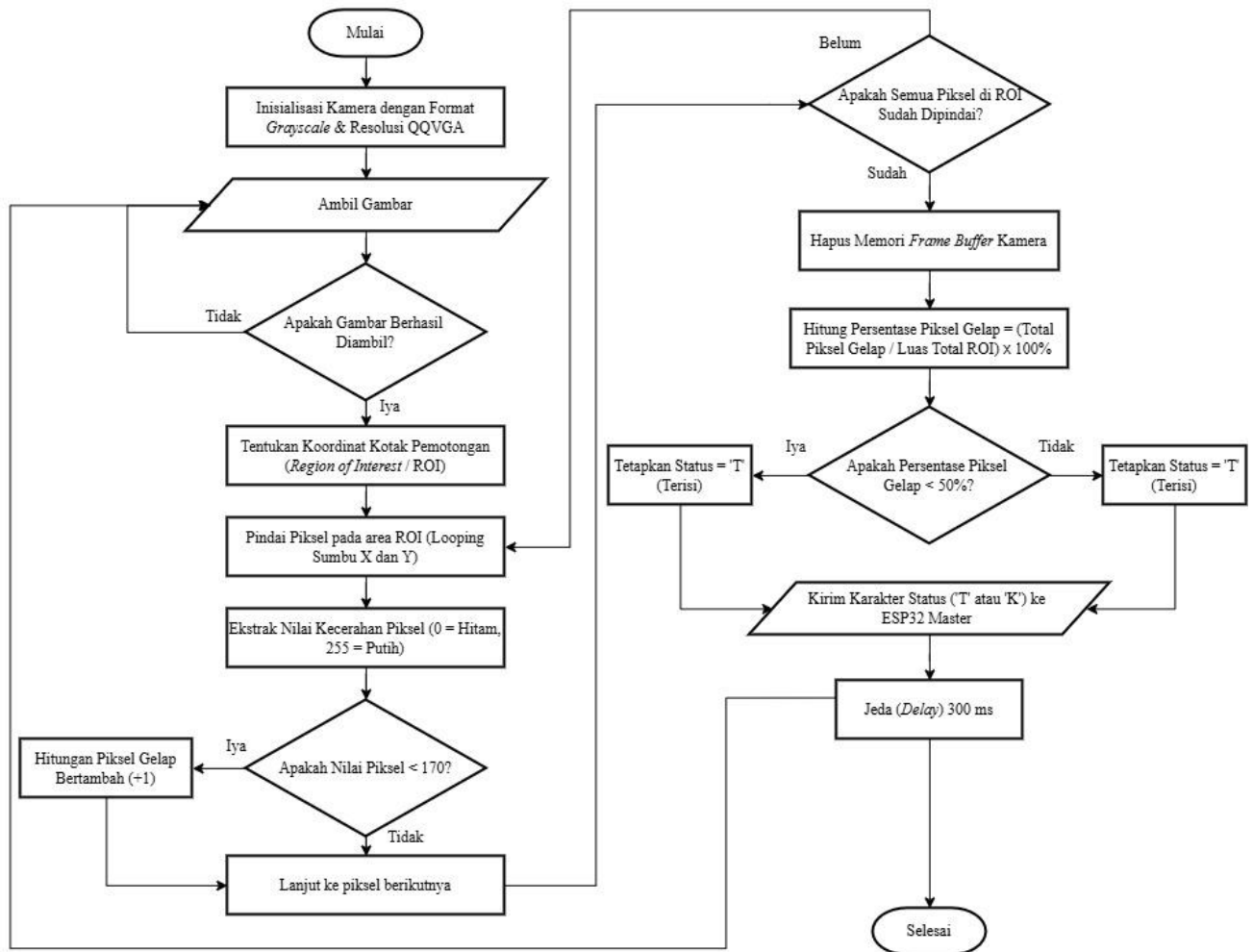
Sementara itu, Dari Tabel 3.6 jalur komunikasi kedua terjadi antara ESP32 *Master* dengan ESP32 *Slave* menggunakan komunikasi perangkat keras UART pada *Serial2* (melalui Pin TX2 dan RX2) dengan kecepatan *baud rate* 9600 bps. Pada arsitektur ini, ESP32 *Master* bertindak murni sebagai pengambil keputusan logis (*decision maker*). Ketika kondisi tertentu terpenuhi, *Master* akan mengirimkan instruksi berwujud teks (*String*) melalui kabel komunikasi. ESP32 *Slave* yang secara terus-menerus memantau jalur komunikasi (berperan sebagai *listener*) akan langsung menterjemahkan instruksi tersebut dan mengeksekusi aktuator fisik. Rincian instruksi data *string* yang dipertukarkan antara *Master* dan *Slave* dapat dilihat pada tabel berikut:

3.8. Alur Kerja Sistem

Alur kerja operasional *Smart Rotary Parking System* dirancang menggunakan prinsip otomasi terdistribusi dengan pendekatan kendali siklus tertutup (*closed-loop control*). Secara menyeluruh, sistem ini mengintegrasikan interaksi dinamis antara tiga unit komputasi utama, yaitu ESP32 Master sebagai pusat pengendali logika, modul ESP32-CAM sebagai subsistem visi komputer, serta ESP32 Slave sebagai pengendali aktuator mekanik dan palang pintu. Alur kerja sistem secara umum dibagi menjadi dua siklus operasi utama, yaitu siklus penyimpanan kendaraan (*Tap-In*) dan siklus pengambilan kendaraan (*Tap-Out*). Setiap siklus dikoordinasikan secara sekuensial berdasarkan umpan balik (*feedback*) dari sensor *Hall Effect* untuk penghitungan langkah gerak mekanik, verifikasi status ketersediaan ruang berbasis citra digital secara *real-time*, sinkronisasi data pada memori non-volatil Flash NVS (*Preferences*), hingga pembaruan status sistem ke antarmuka HMI (LCD dan OLED) serta *broker* MQTT. Untuk memberikan pemahaman yang terperinci dan sistematis, pembahasan alur kerja ini diuraikan ke dalam beberapa subbab yang mencakup mekanisme pemrosesan citra pada modul kamera, dilanjutkan dengan diagram alir operasional sistem pada siklus penyimpanan (*Tap-In*).

3.8.1 Alur Kerja Image Processing

Metode pemrosesan citra pada modul ESP32-CAM dirancang menggunakan pendekatan *Lightweight Image Processing* (pemrosesan citra ringan) guna menyesuaikan kapasitas memori dan komputasi prosesor pada sistem *embedded*. Pendekatan ini berfokus pada efisiensi eksekusi binerisasi melalui metode *Inverted Thresholding* dan pemotongan area *Region of Interest* (ROI) yang kemudian divalidasi menggunakan teknik *pixel counting* (penghitungan piksel) secara berkelanjutan. Untuk memberikan visualisasi yang sistematis mengenai tahapan komputasi mulai dari akuisisi citra hingga transmisi status ketersediaan slot menuju ESP32 Master, alur logika pemrosesan citra disajikan pada diagram alir (Gambar 3.9) berikut:



Gambar 3. 9 Flowchart Pemrosesan Citra ESP32-CAM

Berdasarkan *Flowchart* pada Gambar 3.9, siklus pemrosesan citra diawali dengan tahap inisialisasi konfigurasi perangkat keras kamera OV2640. Format piksel diatur secara spesifik pada skala abu-abu (*grayscale*) dengan resolusi rendah QQVGA (160×120 piksel) guna meminimalkan alokasi memori *buffer* serta mempercepat laju pembacaan *frame*. Setelah inisialisasi berhasil, kamera melakukan akuisisi gambar ke dalam memori *frame buffer*. Sistem memverifikasi ketersediaan data *frame*. Apabila proses pengambilan gambar gagal, sistem secara otomatis mengulangi proses akuisisi. Tahapan selanjutnya adalah penentuan koordinat pemotongan area fokus pembacaan atau *Region of Interest* (ROI). Dengan membatasi area analisis hanya pada penampang lantai keranjang akrilik tempat kendaraan bertumpu, sistem mengabaikan piksel di luar keranjang (seperti

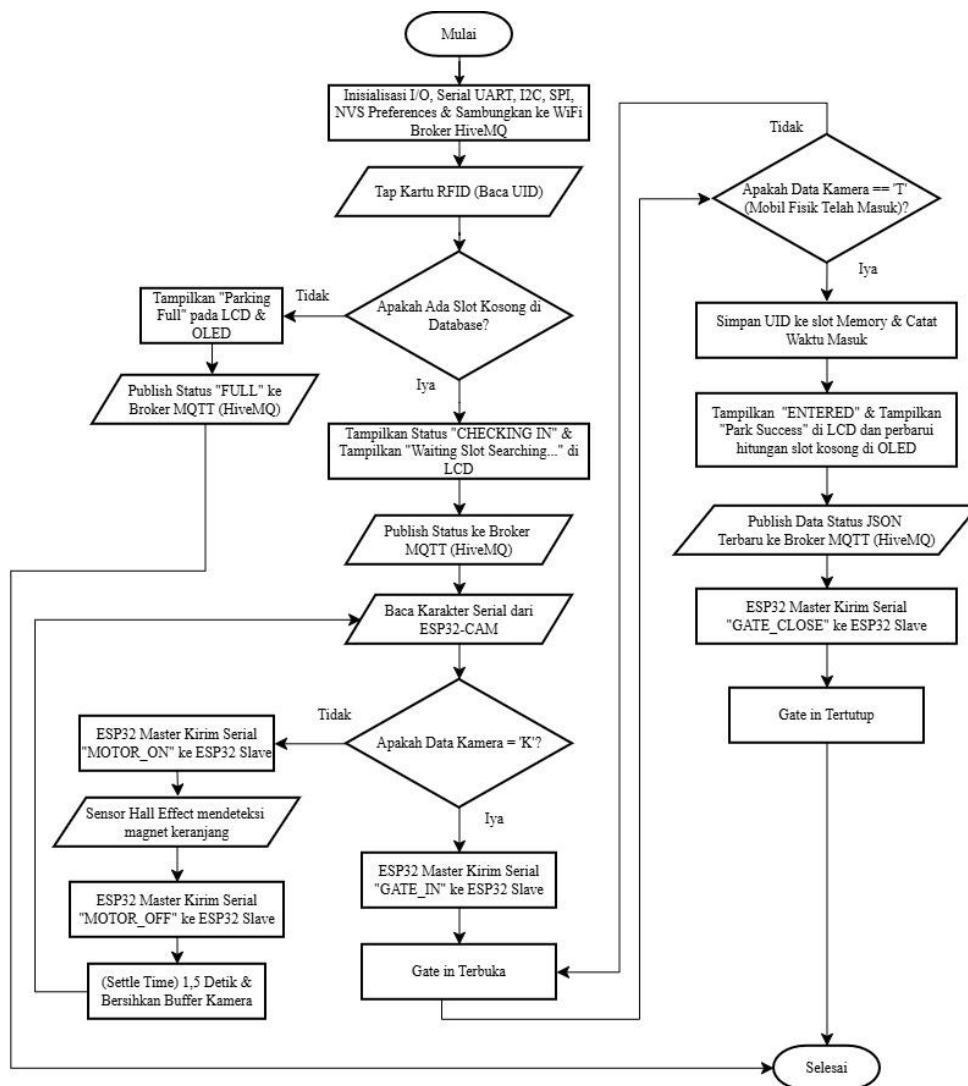
tiang dan lingkungan luar) sehingga menghemat beban kerja komputasi. Program kemudian menjalankan perulangan bertingkat untuk memindai koordinat piksel sumbu X dan Y di dalam kotak ROI serta mengekstrak nilai intensitas kecerahan tiap piksel tunggal dalam rentang 0 (hitam pekat) hingga 255 (putih terang).

Pada setiap piksel yang diperiksa, diterapkan algoritma *Inverted Thresholding* dengan parameter batas ambang `BATAS_THRESHOLD = 170`. Karena lantai keranjang memiliki latar belakang putih dengan pantulan terang, piksel yang bernilai lebih kecil dari 170 diidentifikasi sebagai objek gelap (representasi penampang bodi mobil miniatur atau bayangan pekat), sehingga variabel akumulator `total_piksel_gelap` bertambah satu (+1). Sebaliknya, piksel dengan intensitas ≥ 170 diabaikan karena diklasifikasikan sebagai warna lantai kosong. Pemindaian ini terus berulang hingga seluruh matriks piksel pada area ROI selesai diproses. Sesaat setelah proses pemindaian selesai, memori *frame buffer* kamera segera dibebaskan menggunakan instruksi `esp_camera_fb_return()` untuk mencegah terjadinya penumpukan alokasi memori (*memory leak*) yang rentan memicu kegagalan daya (*brownout/crash*) pada modul ESP32-CAM. Sistem kemudian menghitung persentase kepadatan piksel gelap terhadap luas total area ROI. Nilai persentase ini dievaluasi terhadap parameter `BATAS_PERSentase = 50.0%`. Jika akumulasi piksel gelap mencapai 50% atau lebih, sistem menyimpulkan bahwa terdapat objek kendaraan fisik di dalam keranjang dan menetapkan karakter status 'T' (Terisi). Sebaliknya, jika persentase piksel gelap berada di bawah 50%, sistem menyimpulkan keranjang dalam kondisi steril dari kendaraan dan menetapkan karakter status 'K' (Kosong).

Karakter keputusan status tersebut selanjutnya dikirimkan secara langsung menuju ESP32 Master melalui saluran komunikasi serial UART pada kecepatan *baud rate* 115200 bps. Siklus ini diakhiri dengan jeda waktu (*delay*) selama 300 ms untuk mencegah terjadinya luapan data (*data flooding*) pada antrean serial Master sebelum memulai kembali siklus akuisisi gambar berikutnya.

3.8.2 Alur Kerja Tap in

Untuk memberikan gambaran yang menyeluruh dan terintegrasi mengenai otomatisasi penyimpanan kendaraan, alur kerja sistem disusun ke dalam diagram alir siklus *Tap-In* keseluruhan. Diagram ini menguraikan interaksi terdistribusi antara ESP32 Master sebagai pengendali utama, modul visi ESP32-CAM sebagai subsistem verifikasi visual ketersediaan ruang, serta ESP32 Slave sebagai pengendali aktuatur mekanik dan gerbang. Seluruh mekanisme dirancang menggunakan pendekatan kendali siklus tertutup (*closed-loop control*), di mana transisi tahapan putaran motor maupun pembukaan palang pintu senantiasa didasarkan pada umpan balik sensorik secara *real-time*. Diagram alir proses penyimpanan kendaraan tersebut disajikan pada Gambar 3.10 berikut.



Gambar 3. 10 Flowchart *Tap in*

Berdasarkan diagram alir pada Gambar 3.10, siklus penyimpanan (*Tap-In*) diawali dengan tahap inisialisasi sistem saat mikrokontroler dinyalakan. Tahap ini mencakup konfigurasi pin masukan/keluaran (I/O), pengaktifan komunikasi serial UART ke ESP32-CAM dan Slave, bus I2C untuk antarmuka layar, bus SPI untuk modul RFID, pembacaan memori non-volatil Flash NVS (*Preferences*), serta pembentukan koneksi nirkabel Wi-Fi menuju *broker* MQTT HiveMQ. Proses operasional dimulai ketika pengguna menempelkan kartu identitas pada pemindai RFID RC522. ESP32 Master membaca kode UID kartu dan langsung mengevaluasi ketersediaan kapasitas ruang parkir pada basis data larik `slotMemory[]`. Apabila seluruh slot terisi penuh, sistem langsung menolak akses, menampilkan peringatan "*Parking Full*" pada layar LCD dan OLED, serta mempublikasikan status "*FULL*" ke *broker* MQTT sebelum siklus berakhir tanpa menggerakkan aktuator mekanik.

Sebaliknya, jika kapasitas parkir masih tersedia, sistem mengeset status operasional menjadi "*CHECKING IN*", memperbarui pesan pada LCD dengan teks "*Waiting Slot Searching...*", dan mempublikasikan data pembaruan tersebut ke *broker* MQTT. Sistem kemudian memberikan jeda 1 detik untuk membersihkan antrean data serial (*buffer flush*) pada jalur komunikasi ESP32-CAM guna mencegah pembacaan data sisa. Selanjutnya, ESP32 Master membaca karakter keputusan status ketersediaan ruang dari ESP32-CAM untuk keranjang yang sedang berada tepat di titik akses bawah.

Apabila data yang diterima dari kamera bukan karakter 'K' melainkan 'T' (menandakan bahwa keranjang yang berada di titik akses bawah terisi kendaraan), sistem memasuki *looping* pencarian mekanik. ESP32 Master mengirimkan instruksi serial "*MOTOR_ON*" ke ESP32 Slave untuk memutar motor DC *gearbox* melalui *driver* L298N. Perputaran struktur *rotary* terus dipantau oleh sensor *Hall Effect*. Saat medan magnet penanda keranjang melintas tepat di depan sensor dan memicu sinyal logika rendah (*LOW*), ESP32 Master segera mencacah posisi keranjang (`currentSlot++`), memperbarui posisinya ke Flash NVS, dan mengirimkan instruksi serial "*MOTOR_OFF*" ke Slave untuk menghentikan motor penggerak. Sesaat setelah motor berhenti, diterapkan jeda penstabilan mekanik

(*settle time*) selama 1,5 detik guna meredam osilasi inersia ayunan keranjang akrilik gantung bebas, disusul dengan pengosongan *buffer* serial kamera. Setelah struktur keranjang berada dalam kondisi statis sempurna, alur kembali ke tahap pembacaan kamera untuk memindai ulang kondisi fisik lantai keranjang. Siklus putaran bertahap ini terus berulang secara otomatis hingga kamera mendeteksi lantai keranjang kosong dan mentransmisikan karakter 'K'.

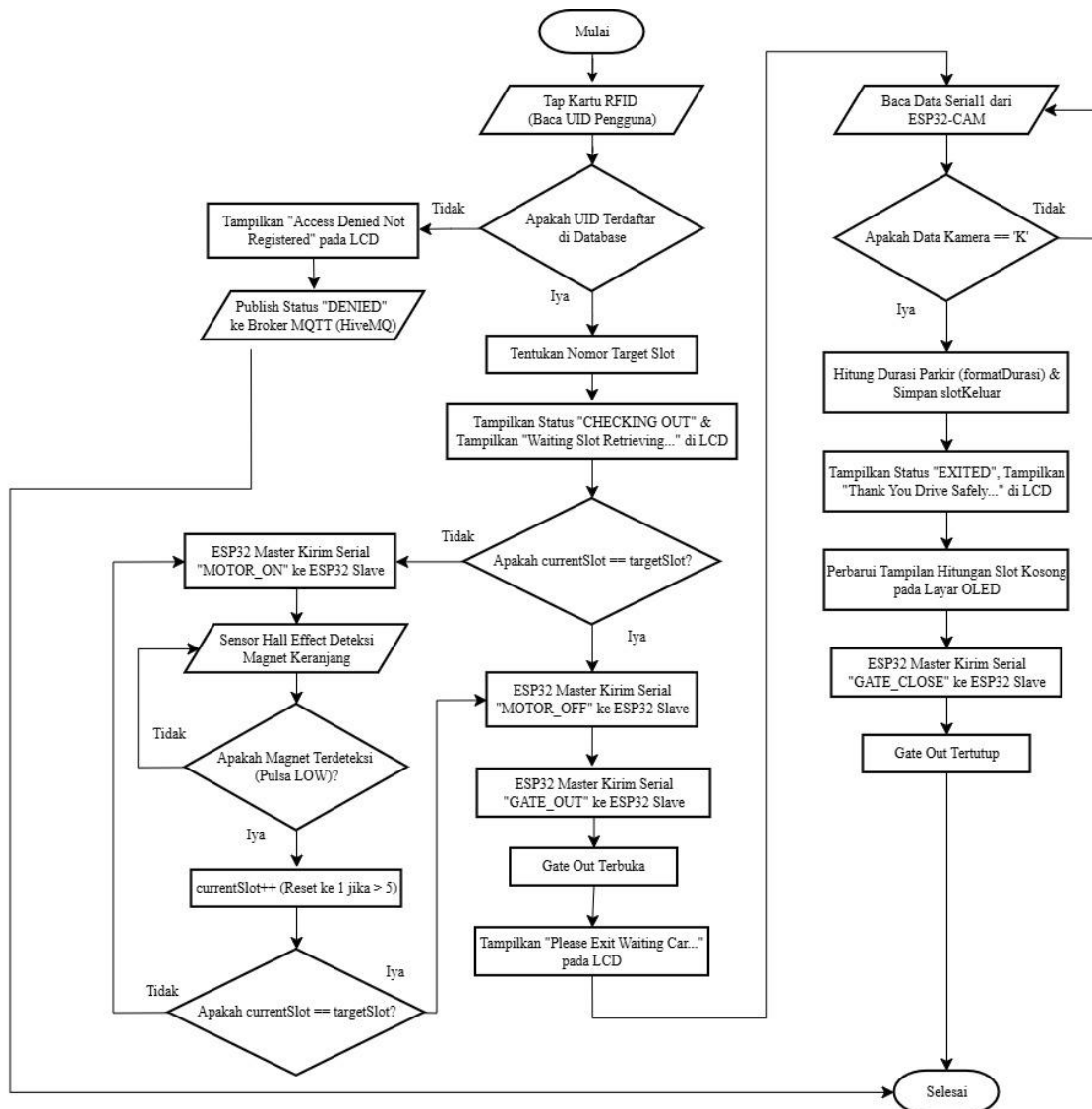
Ketika kondisi kamera bernilai 'K' terpenuhi, keranjang kosong telah siap di depan gerbang. ESP32 Master mengirimkan instruksi "GATE_IN" ke ESP32 Slave, yang direspons dengan menggerakkan motor servo masuk ke sudut buka 90° sehingga palang pintu masuk terbuka. Sistem kemudian beralih ke tahapan umpan balik visual. Pada tahapan ini, palang pintu masuk ditahan dalam kondisi terbuka sembari ESP32 Master terus membaca aliran data serial dari ESP32-CAM secara berulang. Selama kamera masih membaca kondisi kosong ('K'), palang pintu tidak akan menutup.

Begitu kendaraan dimasukkan ke dalam keranjang dan kamera mengonfirmasi kehadiran fisik objek dengan mengirimkan karakter 'T', sistem segera memvalidasi akhir proses parkir. ESP32 Master menyimpan UID kartu pengguna ke dalam larik `slotMemory[]`, mencatat penanda waktu masuk (*timestamp*), serta mengunci basis data tersebut ke dalam Flash NVS. Tampilan status pada layar LCD diperbarui menjadi "ENTERED" dan "Park Success", sisa kuota slot kosong pada layar OLED dikurangi, dan format data status JSON terbaru dipublikasikan ke *broker* MQTT HiveMQ agar tersinkronisasi dengan aplikasi pemantauan pengguna. Setelah seluruh status berhasil diperbarui, barulah ESP32 Master mengirimkan instruksi serial "GATE_CLOSE" ke ESP32 Slave untuk mengembalikan sudut motor servo ke 180° sehingga palang masuk tertutup, dan siklus penyimpanan dinyatakan selesai secara aman.

3.8.3 Alur Kerja Tap out

Untuk memberikan visualisasi yang terstruktur mengenai prosedur pemanggilan dan penjemputan kendaraan yang telah terparkir, alur kerja sistem disusun ke dalam diagram alir siklus *Tap-Out* keseluruhan. Diagram ini menjabarkan koordinasi terdistribusi antara ESP32 Master, ESP32 Slave, dan

modul visi ESP32-CAM dalam mengeksekusi validasi identitas kartu, pelacakan posisi target keranjang berbasis komputasi selisih langkah, hingga pengoperasian palang pintu keluar dengan prinsip kendali siklus tertutup (*closed-loop control*). Diagram alir proses penjemputan kendaraan tersebut disajikan pada Gambar 3.11 berikut.



Gambar 3. 11 Flowchart sistem *Tap out*

Berdasarkan diagram alir pada Gambar 3.11 siklus pengambilan kendaraan (*Tap-Out*) diawali saat pengguna menempelkan kartu identitas pada perangkat pembaca RFID RC522 untuk dibaca kode uniknya (UID). Mikrokontroler ESP32 Master langsung mengevaluasi status UID tersebut terhadap basis data larik

slotMemory[] yang tersimpan pada memori. Apabila kode UID tidak terdaftar di dalam sistem, akses penjemputan secara otomatis ditolak; layar LCD menampilkan pesan "*Access Denied Not Registered*" dan sistem mentransmisikan status "*DENIED*" menuju broker MQTT HiveMQ sebelum mengakhiri proses tanpa memicu pergerakan komponen mekanik.

Sebaliknya, apabila UID terverifikasi terdaftar, sistem menetapkan nomor keranjang tujuan (*targetSlot*) berdasarkan indeks penempatan kendaraan yang bersangkutan. ESP32 Master kemudian mengubah status operasional menjadi "*CHECKING OUT*" dan menampilkan notifikasi "*Waiting Slot Retrieving...*" pada antarmuka layar LCD. Selanjutnya, sistem mengevaluasi apakah posisi keranjang aktif saat ini sudah bersesuaian dengan target tujuan ($\text{currentSlot} == \text{targetSlot}$). Apabila posisi keranjang saat ini telah berada tepat di titik akses bawah, sistem tidak perlu memutar motor penggerak dan dapat langsung mengeksekusi tahapan pembukaan gerbang keluar.

Namun, apabila posisi keranjang belum sesuai dengan target tujuan ($\text{currentSlot} \neq \text{targetSlot}$), sistem memasuki siklus putaran mekanik. ESP32 Master mentransmisikan instruksi serial "*MOTOR_ON*" menuju ESP32 Slave untuk mengaktifkan motor DC *gearbox* melalui *driver* L298N. Selama struktur keranjang berputar secara vertikal, sensor *Hall Effect* memantau lintasan fluks magnetik penanda posisi. Ketika paparan medan magnet memicu transisi logika keluaran sensor menjadi *LOW* (0,018 V), nilai pencacah posisi keranjang diakumulasikan ($\text{currentSlot}++$ dengan pengaturan batas siklik kembali ke 1 jika melebihi 5) dan secara simultan diperbarui ke dalam Flash NVS. Evaluasi posisi kemudian dilakukan kembali; jika *currentSlot* belum menyamai *targetSlot*, motor dibiarkan terus berputar mencari keranjang berikutnya. Begitu keranjang target tiba di titik akses bawah ($\text{currentSlot} == \text{targetSlot}$), ESP32 Master segera mengirimkan instruksi serial "*MOTOR_OFF*" ke ESP32 Slave untuk memutus arus motor penggerak.

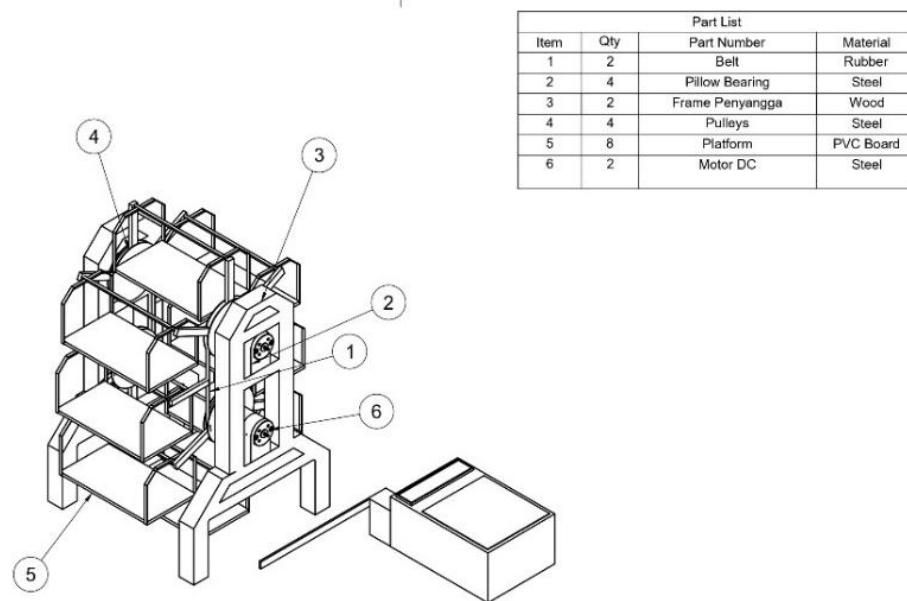
Setelah keranjang berhenti presisi pada titik akses keluar, ESP32 Master mentransmisikan instruksi "*GATE_OUT*" ke ESP32 Slave guna menggerakkan motor servo membuka palang gerbang keluar (*Gate Out Terbuka*) ke sudut 90°,

sembari menampilkan instruksi *"Please Exit Waiting Car..."* pada layar LCD. Pada tahapan ini, diterapkan mekanisme kendali umpan balik visual (*visual closed-loop verification*). Sistem menahan palang gerbang keluar dalam kondisi terbuka sembari membaca aliran data serial dari modul visi ESP32-CAM secara kontinu. Selama kamera masih mendeteksi keberadaan bodi fisik kendaraan di lantai keranjang (mentransmisikan karakter 'T'), gerbang keluar akan tetap dipertahankan terbuka guna menjamin keamanan kendaraan yang sedang bergerak keluar.

Begitu kendaraan miniatur berhasil dikeluarkan dan ESP32-CAM memvalidasi bahwa lantai keranjang telah steril dengan mengirimkan karakter 'K', sistem secara otomatis mengeksekusi pembaruan data operasional. ESP32 Master mengalkulasi durasi parkir berdasarkan selisih waktu masuk terhadap waktu aktual menggunakan fungsi `formatDurasi`, menyimpan nomor keranjang yang dikosongkan ke variabel `slotKeluar`, mengosongkan rekaman UID pada larik `slotMemory[]` dan Flash NVS, serta memperbarui sisa kuota slot parkir pada layar OLED. Status sistem kemudian ditetapkan menjadi *"EXITED"*, layar LCD menampilkan pesan *"Thank You Drive Safely..."*, dan seluruh data telemetri dipublikasikan dalam format JSON ke *broker* MQTT HiveMQ agar tersinkronisasi secara *real-time* ke aplikasi pemantauan pengguna. Setelah seluruh pembaruan status selesai diproses, ESP32 Master mentransmisikan instruksi serial *"GATE_CLOSE"* ke ESP32 Slave untuk mengembalikan motor servo keluar ke sudut tertutup 0° (*Gate Out Tertutup*), dan siklus penjemputan dinyatakan selesai.

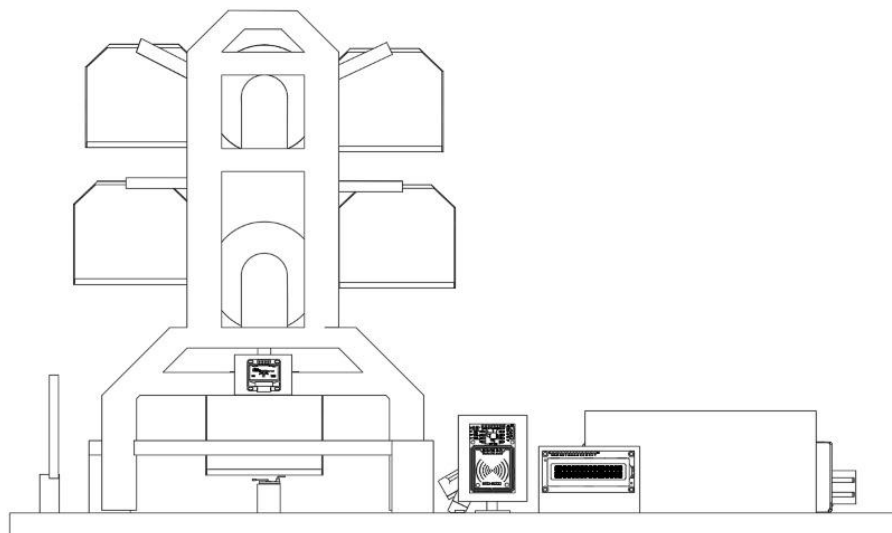
3.9. Gambar 3D

Visualisasi dalam bentuk gambar 3D pada gambar berikut berperan penting dalam menampilkan desain alat secara menyeluruh, meliputi bentuk, dimensi, serta tampilan estetika, sehingga memudahkan dalam memahami fungsi dan cara kerja alat yang dirancang. Sementara itu, detail proyeksi memberikan informasi yang lebih spesifik terkait ukuran, toleransi, serta spesifikasi teknis dari masing-masing komponen, yang sangat diperlukan dalam tahap pembuatan dan perakitan. Adapun gambar ditunjukkan sebagai berikut:



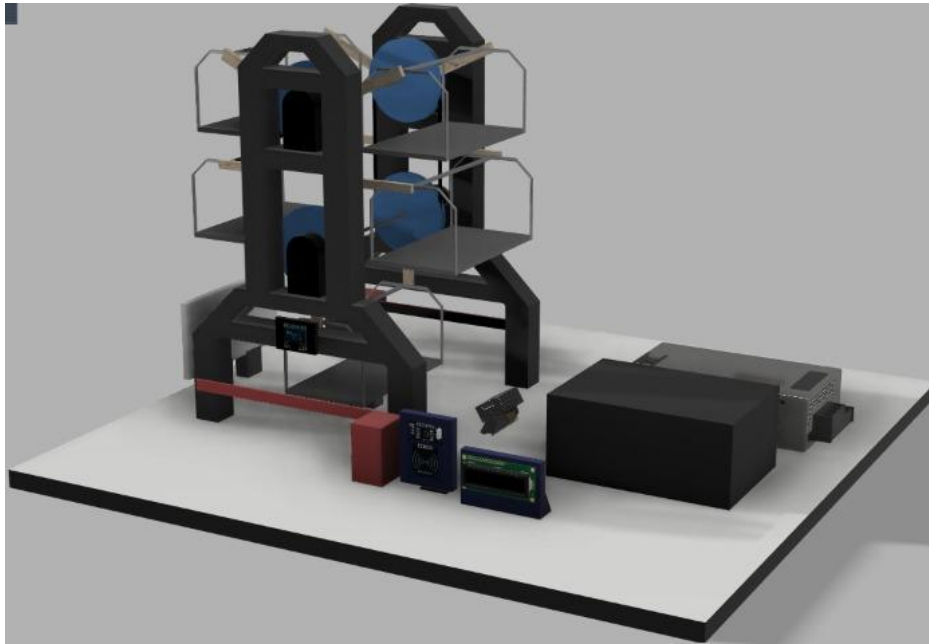
Gambar 3. 12 Komponen Pada Alat

Gambar 3. 12 merupakan gambar 3D tampak dari sudut kiri atas yang menguraikan bagian-bagian penyusun struktur mekanik prototipe beserta daftar komponennya, yang meliputi *belt*, *pillow bearing*, rangka penyangga berbahan kayu, *pulleys*, keranjang parkir (*platform slot*) berbahan *PVC board*/akrilik, serta motor DC penggerak. Gambar ini berfungsi memudahkan pemahaman mengenai tata letak serta hubungan mekanis antar-komponen saat proses perakitan berlangsung.



Gambar 3. 13 Detil Proyeksi

Gambar 3. 13 menyajikan gambar kerja proyeksi ortogonal dari sudut pandang depan sejajar yang menampilkan beberapa komponen dalam bentuk 2D. Gambar proyeksi ini berfungsi sebagai acuan teknis pemotongan bahan, penentuan toleransi jarak celah gerak, serta kepresisian perakitan struktur vertikal.



Gambar 3. 14 Proyeksi 3D

Gambar 3. 14 memperlihatkan visualisasi 3D berwarna dari model prototipe dalam kondisi utuh dan terintegrasi dengan mekanisme pintu gerbang. Tampilan ini berfungsi memberikan gambaran nyata bentuk visual alat, penempatan keranjang akrilik gantung, serta keselarasan posisi lantai dasar terhadap titik akses masuk/keluar kendaraan. Ketiga gambar tersebut merupakan aspek visualisasi yang berperan penting dalam menyampaikan konsep desain alat secara efektif dan membantu pembaca memahami prinsip kerja serta proses pembuatan alat yang dirancang dalam tugas akhir ini.

3.10. Spesifikasi dan Fitur

Pada bagian ini dijelaskan spesifikasi umum dari sistem *rotary parking* yang telah dirancang. Spesifikasi tersebut mencakup karakteristik teknis dari keseluruhan alat, seperti ukuran fisik, kebutuhan daya, kemampuan operasional, serta kinerja sistem dalam menjalankan proses parkir otomatis. Informasi ini bertujuan untuk memberikan gambaran menyeluruh mengenai performa dan

kapasitas sistem yang dikembangkan. Adapun spesifikasi sistem ditunjukkan pada Tabel berikut.

Tabel 3. 7 Spesifikasi Rancang Bangun Alat

Spesifikasi	Value
Dimensi	63 cm x 43 cm x 48 cm
Input Daya	12 VDC
Berat	3 kg
Waktu <i>Image Processing</i>	0,3 s
Pembacaan Sensor <i>Hall Effect</i>	5 mm
Kapasitas Parkir	5 slot
Waktu Proses Parkir	1-2 menit

Pada tabel tersebut pemahaman mengenai spesifikasi dan fitur sistem *Rotary Parking* akan memberikan gambaran yang lebih jelas tentang bagaimana alat ruang parkir ini dapat meningkatkan pengalaman parkir secara keseluruhan.

Spesifikasi pada *table* berikut , dapat dipahami bagaimana setiap komponen bekerja baik secara individual maupun sebagai bagian dari sistem secara keseluruhan. Tabel berikut menyajikan ringkasan spesifikasi serta fungsi dari masing-masing komponen yang digunakan dalam sistem *rotary parking*.

Tabel 3. 8 Spesifikasi Komponen Lengkap

Komponen	Spesifikasi	Fungsi
Power Supply / Adaptor	12V DC	Menyediakan sumber daya listrik untuk seluruh sistem
Sensor <i>Hall Effect</i>	Jarak deteksi 0,1 – 0,5 cm	Mendeteksi posisi slot sebagai <i>counter rotary</i>
Push Button (Tactile Switch)	4 pin, 6x6 mm	Memberikan input manual untuk memulai sistem

ESP32	Tegangan operasi 3.3V, WiFi & Bluetooth	Mikrokontroler utama untuk mengontrol seluruh sistem
ESP32-CAM	Resolusi kamera OV2640 (2MP), Tegangan 5V	Melakukan pengambilan gambar dan deteksi slot parkir
Driver L298N	Tegangan operasi 5V–35V, Arus max 2A/channel	Mengontrol arah dan kecepatan motor DC
Motor DC Gearbox	Tegangan 12V	Menggerakkan mekanisme <i>rotary parking</i>
Modul Step-down	Input 12V, Output 5V / 3.3V	Menurunkan tegangan sesuai kebutuhan komponen

3.11. Fitur

Pada bagian ini dijelaskan berbagai fitur yang dimiliki oleh sistem *rotary parking* yang telah dirancang. Fitur-fitur ini menunjukkan keunggulan sistem dalam aspek otomatisasi, efisiensi, serta kemampuan integrasi antar komponen. Sistem yang dikembangkan menggabungkan teknologi mikrokontroler, sensor, serta pengolahan citra untuk menghasilkan mekanisme parkir yang lebih efektif dibandingkan sistem konvensional. Adapun fitur sistem ditunjukkan pada Tabel berikut.

Tabel 3. 9 Fitur pada Alat

Fitur	Value
Otomatisasi Sistem Parkir	Sistem bekerja secara otomatis menggunakan kontrol berbasis <i>sequential</i> tanpa intervensi manual secara langsung
Deteksi Slot Berbasis Kamera	Menggunakan ESP32-CAM untuk mengidentifikasi slot parkir kosong secara visual

Efisiensi Penggunaan Ruang	Mengoptimalkan ruang parkir dengan konsep <i>rotary</i>
Sistem Penentuan Posisi	Menggunakan sensor <i>infrared</i> sebagai <i>counter</i> untuk menentukan posisi slot secara akurat
Antarmuka Pengguna Sederhana	Menggunakan <i>push button</i> sebagai <i>input</i> awal sistem
Kecepatan Proses Parkir	Waktu proses parkir berkisar $\pm 1 - 2$ menit pada skala <i>prototype</i>
Integrasi Sistem <i>Embedded</i>	Seluruh komponen terintegrasi dalam satu sistem berbasis ESP32
Stabilitas Pergerakan	Pergerakan <i>rotary</i> dikontrol bertahap untuk menghindari hentakan mekanis
Fleksibilitas Pengembangan	Sistem dapat dikembangkan untuk kapasitas slot yang lebih besar

3.12. Teknik Pabrikasi

Dalam proses pembuatan *prototype* sistem *rotary parking*, diperlukan beberapa komponen utama, alat perkakas, serta bahan habis pakai yang mendukung proses perancangan dan perakitan alat. Setiap elemen memiliki peran penting agar sistem dapat dibangun dengan baik dan berfungsi sesuai dengan yang direncanakan. Adapun klasifikasi kebutuhan dalam proses pabrikasi ditunjukkan pada Tabel berikut.

Tabel 3. 10 Alat dan Bahan Teknik Pabrikasi

Komponen Utama		
ESP32	Sensor <i>Infrared</i>	Buck Converter
ESP32-CAM	Push Button	Bearing
Motor DC Gearbox	Motor Servo	Shaft
Driver Motor L298N	Power Supply	Rangka Rotary
Alat Perkakas		
Bor listrik	<i>Cutter</i>	Lem tembak
Obeng	Gerinda	Multimeter

Solder dan timah Tang	Penggaris	Klem
Bahan Habis Pakai		
Kabel jumper	Lem kayu	Heatshrink
Kabel listrik Akrilik	Baut dan mur	Pilox

Pada tahap awal pabrikan, seluruh komponen, alat, dan bahan yang dibutuhkan harus dipastikan tersedia serta dalam kondisi layak digunakan. Proses pembuatan dimulai dari tahap perancangan desain menggunakan model 3D untuk memvisualisasikan bentuk dan struktur sistem secara menyeluruh. Selanjutnya dilakukan proses pembuatan rangka mekanik yang meliputi pemotongan bahan, pembentukan struktur *rotary*, serta pemasangan komponen mekanik seperti poros dan *bearing* agar sistem dapat berputar dengan stabil.

Tahap berikutnya adalah instalasi sistem kelistrikan (*wiring*), di mana seluruh komponen elektronik seperti ESP32, ESP32-CAM, sensor *infrared*, driver motor, serta aktuator dihubungkan sesuai dengan rancangan rangkaian. Proses ini memerlukan ketelitian agar tidak terjadi kesalahan koneksi yang dapat mempengaruhi kinerja sistem. Setelah itu dilakukan tahap pemrograman (*coding*) pada mikrokontroler untuk mengatur logika kerja sistem, termasuk proses deteksi slot parkir, pengendalian motor, serta alur kerja berbasis *sequential control*.

Tahap akhir merupakan proses perakitan (*assembling*), yaitu menggabungkan seluruh bagian mekanik dan elektronik menjadi satu kesatuan sistem yang utuh. Pada tahap ini juga dilakukan pengujian awal untuk memastikan setiap komponen bekerja dengan baik dan sistem dapat berjalan sesuai dengan yang direncanakan. Selama proses pabrikan berlangsung, aspek keselamatan kerja perlu diperhatikan dengan menggunakan alat pelindung diri serta memastikan lingkungan kerja dalam kondisi aman dan nyaman.