Rancang Bangun Sistem *Sortir* Barang Otomatis Berbasis IoT Menggunakan *Android Scanner* dan ESP32 Melalui Sinkronisasi Dinamis *Web Dashboard*

Afif Adam M[1*], Dony Ibnu P[2], Ibnu Khairul H[3], Pramono[4]

[1]Teknik Informatika/Ilmu Komputer
Universitas Duta Bangsa Surakarta
[1*]230103184@mhs.udb.ac.id

[2]Teknik Informatika/Ilmu Komputer
Universitas Duta Bangsa Surakarta
[2]230103261@mhs.udb.ac.id

[3]Teknik Informatika/Ilmu Komputer
Universitas Duta Bangsa Surakarta
[3]230103227@mhs.udb.ac.id

[4]Teknik Informatika/Ilmu Komputer
Universitas Duta Bangsa Surakarta
[4]pramono@udb.ac.id

***Abstrak*— Penelitian ini bertujuan untuk merancang prototipe sistem sortir barang otomatis yang efisien dengan memanfaatkan teknologi IoT berbasis mikrokontroler ESP32. Sistem ini menggunakan kamera *smartphone* sebagai pemindai QR *Code* yang datanya diproses untuk menggerakkan mekanisme sortir berupa konveyor dan motor *servo*. Inovasi utama dalam penelitian ini adalah fleksibilitas konfigurasi sistem melalui antarmuka *web*, di mana pengguna dapat menentukan dan mengubah logika pemisahan berdasarkan kode awalan (*prefix*) QR *Code* secara dinamis tanpa perlu mengubah kode program pada perangkat keras. Enam sensor inframerah digunakan untuk memvalidasi posisi barang pada lima jalur penyimpanan (*Storage* A, B, C, D, dan jalur *fail-safe* E) serta mengirimkan data tersebut ke *dashboard* web sebagai fungsi *monitoring*. Pengujian dilakukan terhadap 78 sampel barang dengan variasi *prefix* QR *Code*, baik yang terdaftar maupun tidak. Hasil pengujian menunjukkan *Sorting Accuracy Rate* (SAR) sebesar 82,1% dengan *Misplacement Rate* (MR) sebesar 17,9%, serta rata-rata *latency response* sebesar 0,62 detik (deviasi standar 0,51 detik). Mekanisme *fail-safe* berhasil mengisolasi 75% barang dengan QR *Code* tidak terdaftar atau rusak ke jalur darurat tanpa intervensi operator, menunjukkan bahwa jalur ini bekerja pada mayoritas kasus namun masih memiliki celah yang perlu disempurnakan.**
Kata kunci— ESP32, QR *Code*, Sistem *Sortir*, IoT, Monitoring Web.

***Abstract*— This research aims to design an efficient automatic goods sorting system prototype utilizing IoT technology based on the ESP32 microcontroller. The system employs a smartphone camera as a QR Code scanner, processing the data to trigger sorting mechanisms consisting of a conveyor and servo motors. The primary innovation of this study is the system's configuration flexibility through a web interface, allowing users to define and modify sorting logic based on QR Code prefixes dynamically without altering the hardware's source code. Six infrared sensors are used to validate item placement across five storage paths (Storage A, B, C, D, and the fail-safe path E) and transmit this data to a web dashboard for monitoring purposes. Testing was conducted on 78 item samples with both registered and unregistered QR Code prefixes. Results indicate a Sorting Accuracy Rate (SAR) of 82.1% with a Misplacement Rate (MR) of 17.9%, and an average latency response of 0.62 seconds (standard deviation 0.51 seconds). The fail-safe mechanism successfully isolated 75% of items with unregistered or damaged QR Codes to the emergency path without operator intervention, indicating the path performs reliably in most cases while still showing gaps that warrant further refinement.**
Keywords— ESP32, QR Code, Sorting System, IoT, Web Monitoring.

I. PENDAHULUAN

Di era industri modern, efisiensi operasional dalam manajemen logistik menjadi faktor penentu daya saing sebuah entitas bisnis. Salah satu proses yang paling krusial adalah penyortiran barang berdasarkan kategori tertentu [1]. Penggunaan tenaga manusia dalam proses ini memiliki keterbatasan signifikan, seperti tingkat kelelahan yang memicu kesalahan identifikasi, kecepatan yang tidak konsisten, serta biaya operasional jangka panjang yang tinggi[3]. Oleh karena itu, diperlukan sistem otomasi yang mampu bekerja secara berkelanjutan dengan tingkat akurasi yang stabil melalui integrasi teknologi robotik dan kecerdasan buatan [1], [3]. Integrasi antara perangkat keras dan sistem *monitoring* berbasis *Internet of Things* (IoT*)* juga terbukti mampu mengoptimalkan performa serta analisis akurasi penanganan logistik secara *real-time* [4].

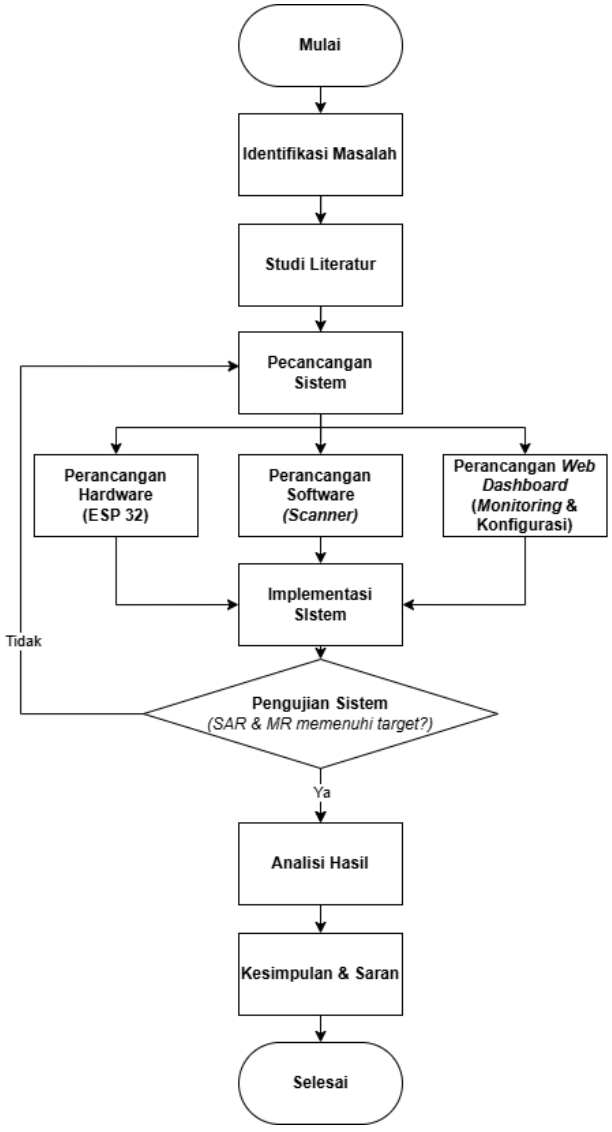
Penggunaan QR *Code* sebagai identitas unik barang telah menjadi standar global karena kemudahannya dalam penyimpanan informasi. Beberapa penelitian terdahulu telah menerapkan

sistem penyortiran berbasis mikrokontroler menggunakan pemindai kode barang untuk meningkatkan efisiensi [2], [8]. Namun, sistem *sortir* otomatis yang ada saat ini seringkali bersifat kaku *(hard-coded)*, dimana perubahan aturan *sortir* memerlukan pemrograman ulang pada perangkat keras [8], [9]. Hal ini menjadi kendala bagi gudang atau unit produksi yang memiliki variasi barang yang sering berubah [2]. Selain itu, kebutuhan akan transparansi data menuntut adanya sistem pemantauan jarak jauh yang dapat diakses secara *real-time* [4].

Penelitian ini mengusulkan solusi berupa sistem *sortir* berbasis *Internet of Things* (IoT) menggunakan ESP32. Mikrokontroler ini dipilih karena memiliki modul Wi-Fi internal yang memungkinkan komunikasi nirkabel dengan *server web* melalui pengiriman data JSON yang ringan [13]. Dengan memanfaatkan kamera *smartphone* sebagai pemindai QR *Code* sementara (untuk kebutuhan prototipe) berbasis *Flutter* yang mengintegrasikan *Google ML Kit* [15] serta integrasi *database* MySQL [12], sistem ini memungkinkan pengguna untuk melakukan kontrol penuh atas logika penyortiran melalui *dashboard web*. Fleksibilitas ini memastikan sistem dapat beradaptasi dengan berbagai jenis QR *Code* baru hanya melalui *input* di sisi perangkat lunak tanpa perlu menghentikan operasional perangkat.

II. METODOLOGI PENELITIAN

Alur tahapan penelitian ini dirancang secara sistematis sebagaimana direpresentasikan pada diagram alir Gambar 1. Penelitian diawali dengan identifikasi masalah terhadap keterbatasan sistem sortir konvensional yang bersifat *hard-coded*, dilanjutkan dengan studi literatur yang mencakup kajian teknologi IoT, ESP32, dan sistem sortir berbasis mikrokontroler [2], [8], [11]. Tahap selanjutnya adalah perancangan sistem yang meliputi subsistem perangkat keras, perangkat lunak, dan *web dashboard* sebagai pusat konfigurasi logika sortir secara dinamis [4], [6], yang kemudian diwujudkan melalui implementasi dan perakitan sistem secara terintegrasi. Pengujian dilakukan dalam tiga tahap, yaitu *unit testing*, pengujian integrasi, dan *system testing*; apabila hasil belum memenuhi target, proses dikembalikan ke tahap perancangan untuk perbaikan [2]. Tahap akhir meliputi analisis hasil secara kuantitatif berdasarkan metrik SAR, MR, dan *latency response*, serta penyusunan kesimpulan dan saran pengembangan [4].

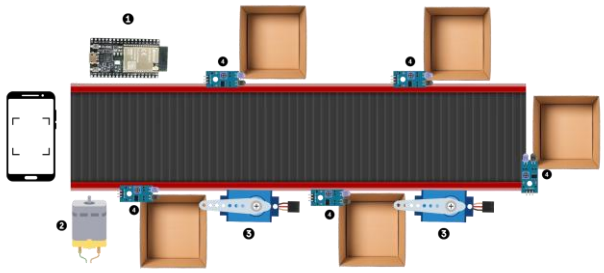


Gambar 1. Diagram alir penelitian

A. *Desain Sistem*

Sistem yang diimplementasikan dalam penelitian ini dibangun menggunakan arsitektur *hybrid* yang mengintegrasikan dua subsistem utama secara *real-time*, yaitu subsistem

perangkat keras (*hardware*) dan subsistem perangkat lunak (*software*). Tata letak fisik komponen hardware utama meliputi mikrokontroler ESP32, motor DC penggerak konveyor, motor *servo* SG90 sebagai aktuator pembelok, sensor inframerah FC-51 pada setiap jalur penyimpanan, serta posisi smartphone sebagai unit pemindai QR *Code* digambarkan pada Gambar 2. Integrasi antara subsistem *hardware* tersebut dengan subsistem *software* dan basis data terpusat, yang menjembatani interaksi antara komponen fisik di lapangan dengan sistem *monitoring* berbasis *web*, (dijelaskan lebih lanjut melalui diagram blok pada Gambar 3) dan terbukti mampu mengoptimalkan performa penanganan logistik serta analisis akurasi data secara berkelanjutan [4].



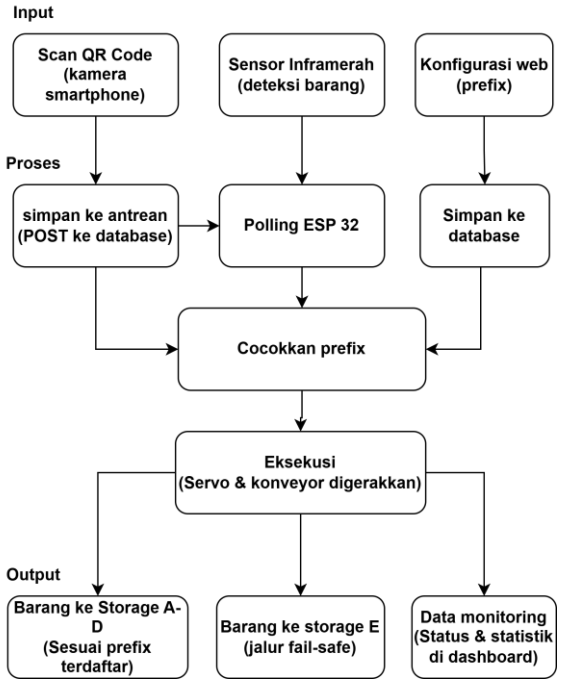
Gambar 2. Desain subsistem *hardware.*

Subsistem *hardware* mencakup: (1) ESP32 sebagai mikrokontroler utama yang mengendalikan seluruh aktuator; (2) motor DC dengan *driver* L298N sebagai penggerak konveyor *belt*; (3) dua buah motor *servo* SG90 sebagai aktuator pembelok barang ke jalur *Storage* A, B, C, dan D; (4) enam buah sensor inframerah sebagai pendeteksi kehadiran barang di setiap jalur penyimpanan.

Subsistem *software* mencakup: (1) firmware ESP32 yang ditulis menggunakan Arduino IDE berbasis C++; (2) web *server* berbasis PHP yang berjalan di atas Apache; (3) database MySQL untuk menyimpan konfigurasi aturan *sortir* dan *log* data; serta (4) antarmuka *web* berbasis HTML, CSS, JavaScript, dan PHP dengan memanfaatkan aplikasi *mobile scanner* berbasis *android* dengan menggunakan *library scanner* dari *Google ML Kit.*

B. Diagram Blok dan Alur Kerja Sistem

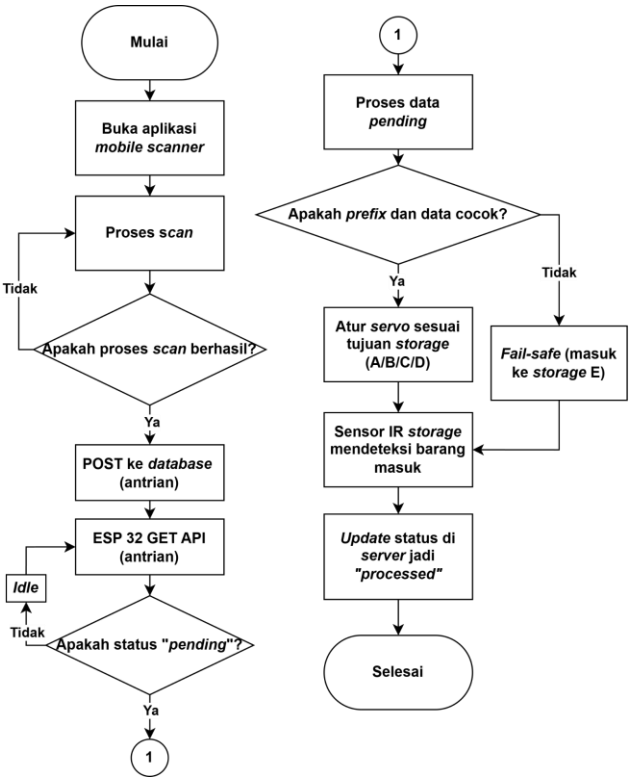
Struktur fungsional sistem kendali *sortir* otomatis ini dibagi menjadi tiga bagian utama, yaitu subsistem masukan (*input*), proses (*process*), dan keluaran (*output*). Komponen *input* bertugas menangkap data dari lingkungan melalui pemindaian QR *Code* barang dan pembacaan intensitas cahaya dari sensor inframerah. Komponen *process* berpusat pada mikrokontroler ESP32 yang mengolah data antrian, melakukan pencocokan *prefix string* secara dinamis, dan mengambil keputusan logis. Komponen *output* bertindak sebagai eksekutor mekanis yang meliputi pergerakan motor *servo* dan motor DC konveyor, serta visualisasi data pemantauan operasional pada *web dashboard*.



Gambar 3. Desain blok sistem.

Prosedur operasional dan alur keputusan data teknis di dalam sistem ini dirancang secara terstruktur mulai dari deteksi awal hingga validasi akhir di dalam gudang [11]. Secara umum, logika pemrosesan data di dalam sistem dibagi menjadi tiga fase berurutan yang saling berkesinambungan. Fase pertama dimulai dari inisialisasi perangkat dan penangkapan citra QR *Code* oleh unit pemindai mobile. Fase kedua

berpusat pada *server* dan mikrokontroler, di mana terjadi proses transmisi data via HTTP POST, penyimpanan antrean pada basis data, hingga penarikan *config* berformat JSON oleh ESP32 melalui metode *polling* berkala. Fase ketiga merupakan tahap eksekusi fisik, di mana ESP32 melakukan komparasi *string prefix* untuk mengaktifkan pin PWM *servo* yang sesuai, dilanjutkan dengan validasi *feedback* digital dari sensor inframerah sebelum data *log* akhir dikirim kembali ke *server*. Seluruh tahapan pemrosesan data, mekanisme *polling* berkala, serta validasi logika untuk memastikan keandalan sinkronisasi perangkat keras dan lunak meskipun dalam kondisi arus barang yang padat ini direpresentasikan secara visual melalui diagram alur kerja (*flowchart*) pada Gambar 4.

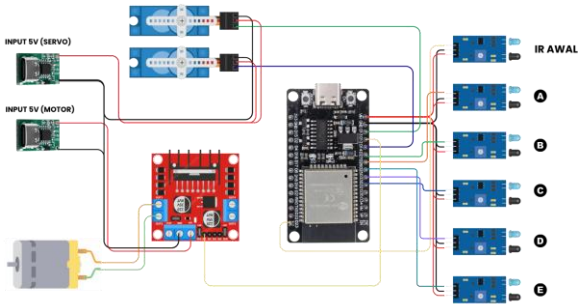


Gambar 4. Desain *flowchart* sistem.

Skenario kerja di lapangan dimulai ketika operator melakukan autentikasi pada aplikasi *mobile* untuk menginisialisasi kamera *smartphone*. Saat barang melintasi area konveyor, kamera akan memindai QR *Code* barang, lalu *library Google ML Kit* akan memproses citra tersebut dan mengirimkan hasilnya ke API *server* melalui protokol HTTP POST untuk dimasukkan ke dalam tabel antrian basis data MySQL [15].

Pada siklus *polling* berikutnya, ESP32 akan menarik data antrian teratas, mencocokkan *prefix* kode, dan mengirimkan sinyal elektrik menuju aktuator pembelok yang sesuai. Secara bersamaan, *supervisor* dapat memantau status kapasitas setiap *storage* melalui *web dashboard* secara *real-time*. Apabila terjadi dinamika kebutuhan logistik, *administrator* dapat memodifikasi *parameter* fisik, seperti kecepatan konveyor dan sudut bukaan *servo*, serta memperbarui aturan *prefix* QR *Code* secara langsung dari *dashboard* tanpa perlu menghentikan proses operasional sistem atau melakukan pemrograman ulang pada perangkat keras [6].

C. Desain Skema IoT



Gambar 5. Desain skema elektrikal IoT.

Untuk menjamin kestabilan arus dan tegangan pada prototipe sistem *sortir* otomatis ini, perancangan skema elektrikal menerapkan teknik isolasi daya total (*power rails isolation*). Pasokan daya dibagi secara independen ke dalam tiga buah modul pencatu daya Power Supply USB-C 5V terpisah. Penerapan isolasi tiga jalur ini sangat krusial guna mengeliminasi gangguan akibat arus kejut *(inrush current)* dan lonjakan tegangan balik (*back*-EMF) yang dihasilkan oleh aktuator mekanis saat bekerja secara bersamaan [5]. Jika aktuator dan mikrokontroler berbagi jalur daya yang sama, penurunan tegangan sesaat *(voltage dip)* akan memicu proteksi internal perangkat keras berupa fenomena *microcontroller restart* atau *brownout* pada ESP32 yang berakibat pada kegagalan koneksi IoT secara mendadak [7].

Jalur daya pertama dialokasikan khusus untuk menyuplai pin VCC pada dua buah motor *servo* SG90 guna menjaga stabilitas torsi dan arus konstan saat memutar *flap* pembelok .Jalur daya kedua dihubungkan langsung menuju terminal input daya *driver* motor L298N yang mengendalikan motor DC konveyor belt dengan karakteristik beban induktif tinggi. Jalur daya ketiga dikhususkan untuk menyuplai pin VIN pada mikrokontroler ESP32 serta didistribusikan secara paralel untuk menghidupkan enam buah sensor inframerah FC-51 melalui pin keluaran tegangan ESP32.

Meskipun menggunakan tiga sumber daya 5V terpisah, seluruh kutub *negatif* atau *Ground* (GND) dari ketiga catu daya, *driver* L298N, kedua motor *servo*, dan ESP32 wajib dihubungkan menjadi satu kesatuan *(common ground)* untuk memastikan referensi sinyal PWM *(Pulse Width Modulation*) tetap sinkron dan presisi.

D. Perancangan Hardware dan Mekanikal

Konveyor *belt* dirancang menggunakan *belt* dari bahan kertas dengan panjang 100 cm melingkar dan lebar 9 cm, digerakkan oleh motor DC 5V melalui *driver* L298N yang dikendalikan oleh sinyal PWM dari ESP32 [14]. Kecepatan konveyor dapat diatur melalui *dashboard web* dengan mengubah nilai kecepatan konveyor antara 0-255.

Mekanisme pembelok barang menggunakan motor servo SG90 yang dipasang secara *vertikal* di ujung setiap jalur percabangan. Ketika QR *Code* barang terdeteksi dan barang diidentifikasi oleh sinyal inframerah, ESP32 akan mengirimkan sinyal PWM ke *servo* yang sesuai untuk memutar *flap* pembelok selama durasi tertentu, kemudian kembali ke posisi awal [16]. Posisi sudut *servo* dikalibrasi melalui konfigurasi *web* agar presisi dalam mengarahkan barang ke jalur yang tepat [10].

Sensor inframerah tipe FC-51 dipasang di ujung setiap jalur penyimpanan (*Storage* A, B, C, D) dan satu buah di jalur *fail-safe* (*Storage* E untuk barang tidak terdaftar) serta dipasang pada awal jalur untuk mendeteksi benda awal setelah *scan* dimulai. Sensor ini mendeteksi kehadiran barang berdasarkan pantulan sinar inframerah dan mengirimkan sinyal digital HIGH ke pin GPIO ESP32 ketika terdeteksi ada barang yang melewatinya.

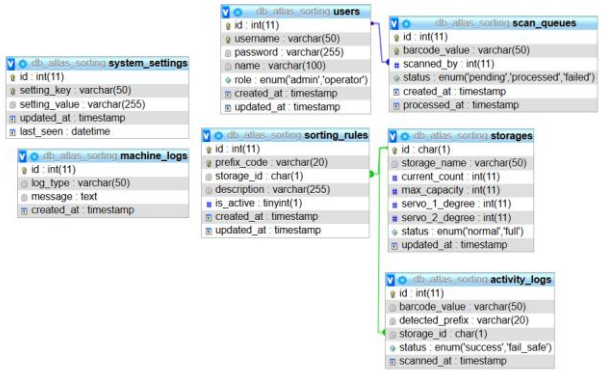
E. Arsitektur Perangkat Lunak dan Sinkronisasi Logika Dinamis

Sistem perangkat lunak pada penelitian ini dibangun menggunakan arsitektur *hybrid* terintegrasi yang menghubungkan *database* terpusat [12]. Di sisi lapangan, operator menggunakan aplikasi *mobile* berbasis *Flutter* yang mengintegrasikan *library mobile_scanner* dan *Google ML Kit* untuk mendeteksi QR *Code* secara instan dengan latensi rendah [6]. Pemilihan *Flutter* didasarkan pada performa *multi-threading* melalui *Isolates* yang memungkinkan proses *rendering* kamera berjalan paralel dengan proses *decoding* gambar oleh algoritma *Google ML Kit*. Setelah data QR *Code* terbaca, aplikasi secara otomatis mengirimkan data *string* tersebut melalui protokol HTTP POST menuju API *server* berbasis PHP *native* untuk disimpan ke dalam tabel antrian *database* MySQL [2]. Proses penangkapan data ini dirancang agar berjalan independen dari pergerakan konveyor, sehingga operator dapat melakukan pemindaian berlapis tanpa mengganggu siklus kerja perangkat keras utama [6].

Penyimpanan data dan konfigurasi sistem dikelola secara terpusat pada basis data MySQL yang dirancang untuk menjaga integritas data *log* operasional dan sinkronisasi parameter dinamis. Arsitektur basis data ini terdiri dari beberapa tabel utama yang saling berelasi, meliputi tabel *sorting_rules* untuk menyimpan parameter konfigurasi *prefix string* dan pemetaan *storage* tujuan, tabel *system_settings* untuk menyimpan kalibrasi fisik (kecepatan konveyor dan sudut derajat *servo*), tabel *scan_queue* sebagai antrean data QR *Code* yang belum dieksekusi, tabel *activity_logs* untuk mencatat riwayat hasil akhir setiap proses *sortir*, serta tabel *machine_logs* untuk mencatat kejadian anomali tingkat perangkat keras.

Relasi antar-tabel ini memastikan bahwa setiap perubahan logika yang diinput oleh

administrator melalui *web* akan langsung memperbarui referensi data yang akan ditarik oleh ESP32. Keterikatan struktural antar-tabel ini dikelola melalui mekanisme *foreign key* yang menghubungkan riwayat aktivitas mesin dengan aturan konfigurasinya, sehingga seluruh log data transaksi pemindaian tetap sinkron dan terlacak secara kronologis. Rancangan pemetaan entitas dan hubungan kardinalitas antar-tabel tersebut digambarkan secara detail pada *Entity Relationship Diagram* (ERD) pada Gambar 6



Gambar 6. Desain ERD database.

Untuk mengeksekusi penyortiran fisik, *firmware* ESP32 bekerja dalam *loop* utama yang secara berkala melakukan sinkronisasi data melalui mekanisme *polling* HTTP GET setiap 5 detik ke *endpoint /api/config* [13]. Melalui *request* ini, ESP32 mengambil dokumen konfigurasi berformat JSON yang berisi aturan aktif dari tabel *sorting_rules*. Algoritma pencocokan *prefix* bekerja dengan mengurutkan seluruh aturan berdasarkan panjang *string* dari yang terpanjang hingga terpendek guna menghindari konflik pembacaan pada kode wilayah logistik yang saling beririsan. Ketika data antrean terbaru diterima dari tabel *scan_queue*, ESP32 mencocokkan karakter awal string barang dengan daftar aturan tersebut; jika ditemukan kecocokan, sinyal PWM akan langsung mengaktifkan motor servo pada jalur yang sesuai, sedangkan jika seluruh baris aturan tidak terpenuhi, barang secara otomatis dialihkan ke jalur *fail-safe* (*Storage* E).

Seluruh kendali *administratif* dan pemantauan kinerja sistem berpusat pada *web dashboard* responsif yang bertindak sebagai *command center* bagi *supervisor* [1]. Fungsi *monitoring* pada *dashboard* ini menyajikan transparansi data logistik secara *real-time*, menampilkan statistik jumlah barang masuk, serta menghitung persentase tingkat kesuksesan sortir berdasarkan umpan balik (*feedback*) digital dari enam sensor inframerah FC-51 yang tercatat pada tabel *activity_logs* [6]. Selain pengawasan, *dashboard* ini memegang peranan krusial sebagai media konfigurasi parameter sistem, di mana *administrator* memiliki hak akses penuh untuk mengubah relasi *prefix* QR *Code* secara instan, menetapkan batasan kapasitas maksimal gudang demi mencegah penumpukan, mengatur kecepatan putaran konveyor melalui *duty cycle* PWM, serta mengalibrasi derajat bukaan sudut motor *servo* secara presisi langsung dari antarmuka *web* tanpa perlu melakukan modifikasi kode program atau melakukan *flash* ulang *firmware* pada mikrokontroller [2], [9].

F. Prosedur Pengujian

Pengujian sistem dilakukan dalam tiga tahap. Tahap pertama adalah pengujian unit (*unit testing*) terhadap setiap komponen secara terpisah, meliputi pengujian akurasi pembacaan QR *Code* oleh kamera *smartphone*, pengujian respon motor *servo* terhadap sinyal PWM, dan pengujian sensitivitas sensor inframerah. Tahap kedua adalah pengujian integrasi untuk memastikan komunikasi antara ESP32, *server web*, dan *database* berjalan dengan benar. Tahap ketiga adalah pengujian sistem secara keseluruhan (*system testing*) menggunakan sampel barang dengan QR *Code* yang telah dikategorikan ke dalam beberapa kelompok, termasuk kategori QR *Code* tidak terdaftar.

Metrik evaluasi dalam penelitian ini diukur secara kuantitatif berdasarkan data log objektif yang terekam pada basis data. Metrik tersebut meliputi:

a. Sorting Accuracy Rate (%)

$$\text{SAR} = \left(\frac{\Sigma Barang\ Sukses}{Total\ Sampel\ Diuji}\right) \times 100\%$$

dihitung berdasarkan rasio jumlah status *success* pada tabel *activity_logs* dibandingkan total barang yang diuji.

b. Misplacement Rate (%)

$$MR = \left(\frac{Jumlah\ Log\ CRITICAL}{Total\ Sampel\ Diuji}\right) \times 100\%$$

dihitung dari frekuensi kemunculan *log CRITICAL* pada tabel *machine_logs* akibat kesalahan pembelokan fisik.

c. Latency Response

$$TL = T_{servo\ move} - T_{ir\ active}$$

waktu komparasi data sejak IR_AWAL aktif hingga *servo* bergerak memotong jalur konveyor.

III. HASIL DAN PEMBAHASAN

Pada bagian ini, akan dibahas secara detail mengenai hasil dari seluruh tahapan pengujian yang telah dilakukan, mulai dari pengujian fungsionalitas tiap komponen (*unit testing*), integrasi sistem berbasis IoT, hingga analisis performa kuantitatif alat berdasarkan matriks evaluasi yang sudah ditentukan.

1.) Pengujian Fungsionalitas Komponen (Unit Testing)

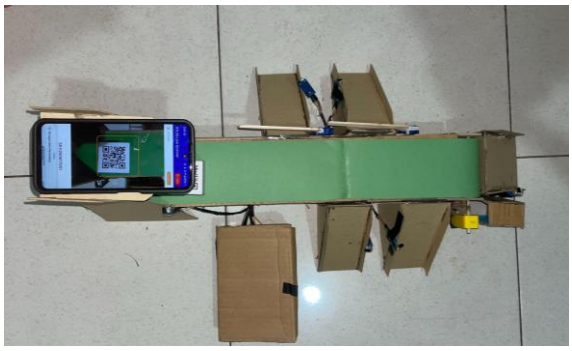
Sebelum sistem diuji secara terintegrasi, dilakukan pengujian terhadap komponen *hardware* dan *software* secara terisolasi untuk memastikan batas operasional optimal [2].

a) Pengujian *Google ML Kit Scanner (Mobile)*

Pengujian pembacaan QR *Code* oleh *Google ML Kit* dilakukan pada variasi jarak 5–20 cm dengan kondisi pencahayaan normal. Kamera *smartphone* berhasil mendeteksi seluruh sampel QR *Code* standar yang diujikan tanpa kegagalan pembacaan, dengan kecepatan respon yang dinilai memadai untuk memastikan data masukan terkirim ke *server* sebelum barang mencapai area percabangan konveyor. Antarmuka aplikasi pemindai berbasis *Flutter* ini dirancang agar mudah dioperasikan oleh petugas gudang di lapangan, seperti yang ditunjukkan pada Gambar 7.

Gambar 7. Gambar antarmuka aplikasi scanner.

Saat paket melintasi area awal konveyor, operator melakukan pemindaian menggunakan aplikasi pada *smartphone*. Proses pemindaian fisik dan interaksi *hardware* pembelok saat mendeteksi paket dapat dilihat pada Gambar 8.



Gambar 8. Dokumentasi Proses Pemindaian pada Prototipe Konveyor.

b) Kalibrasi Derajat Bukaan Motor *Servo* SG90

Posisi sudut *flap* pembelok dikonfigurasi melalui antarmuka *web dashboard* sesuai dengan tata letak fisik mekanis prototipe. Sudut *default* ditetapkan pada 0° sebagai posisi *standby* (menutup). Untuk mengalihkan barang menuju *Storage* A dan B, sudut bukaan

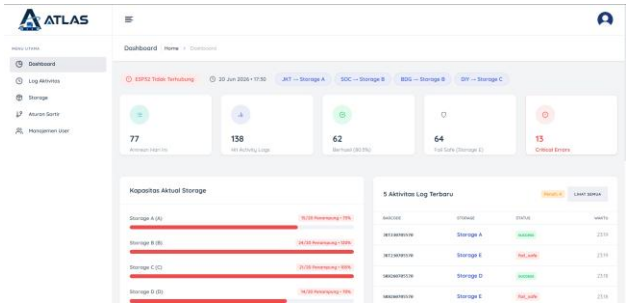
ditetapkan pada 45°, sedangkan untuk *Storage* C dan D ditetapkan pada 130°, menyesuaikan jarak dan posisi percabangan jalur konveyor. Nilai sudut ini dapat diubah sewaktu-waktu melalui *dashboard* tanpa perlu modifikasi *firmware*, yang merupakan salah satu keunggulan utama arsitektur sistem ini.

c) Respon Sensor Inframerah FC-51

Sensor Inframerah FC-51 yang telah dikalibrasi sensitivitas jaraknya pada range 2cm - 4cm dari tepi pembatas *storage* untuk menghindari *false positive* akibat pantulan cahaya luar ataupun *interferensi* dari material konveyor [2].

2.) Pengujian Sinkronisasi Logika Dinamis (JSON & Web Dashboard)

Pengujian ini memvalidasi kemampuan ESP32 dalam melakukan *polling* data konfigurasi secara berkala tanpa menyebabkan *crash* pada *memory heap*. Tabel 1 menunjukkan hasil pengujian pencocokan *prefix* QR *Code* terhadap konfigurasi aturan yang aktif pada *dashboard* web beserta respons mekanis ESP32.



Gambar 9. Antarmuka Web Dashboard Utama untuk Monitoring dan Konfigurasi Aturan JSON.

Manajemen aturan sortir ini dikendalikan penuh melalui *Web Dashboard* utama yang sekaligus berfungsi sebagai pusat *monitoring* kapasitas gudang secara *real-time*, sebagaimana diimplementasikan pada Gambar 9.

Tabel 1. Hasil Pengujian Sinkronisasi Logika Dinamis via JSON menggunakan QR *Code*

Skenario	Konfigurasi JSON (*prefix* → target)	QR *Code*	Target *Storage*	Status
1	{"prefix": "JKT", "target": "A"}	JKT230705570	A	*Success*
2	{"prefix": "SOC", "target": "B"}	SOC230807566	B	*Success*
3	{"prefix": "SKH", "target": "D"}	SKH260705570	D	*Success*
4	{"prefix": "SOC", "target": "B"}	SOCSRG260512178	B	*Success*
5	*(tidak terdaftar)*	SKG240705570	E	*Fail-safe*

Analisis prioritas *prefix* dilakukan saat sistem diberikan aturan dengan *overlap prefix*. Pada skenario 4, QR *Code SOCSRG260512178* mengandung dua *prefix* yang terdaftar sekaligus, yaitu SOC yang diarahkan ke *Storage* B dan SRG yang diarahkan ke *Storage* C. *Algoritma* pengurutan berdasarkan panjang *string* terpanjang terbukti berjalan dengan benar sistem mengevaluasi seluruh karakter awal *string* secara berurutan dari *prefix* terpanjang ke terpendek, sehingga SOC dikenali sebagai kecocokan pertama dan barang diarahkan secara presisi menuju *Storage* B sesuai konfigurasi aktif, bukan ke *Storage* C. Hal ini membuktikan bahwa sistem tidak terjebak pada kecocokan parsial yang ambigu. Sementara itu, skenario 5 menunjukkan bahwa QR *Code* dengan *prefix* yang tidak terdaftar dalam sistem (SKG240705570) secara otomatis diisolasi ke jalur *fail-safe Storage* E tanpa memerlukan intervensi operator. Hasil keseluruhan menegaskan bahwa pembacaan konfigurasi JSON secara dinamis melalui *endpoint* API mampu menggantikan metode *hard-coded* konvensional pada mikrokontroler secara adaptif [8].

Tabel 2. Data Pengukuran *Latency Response* (TL) pada Sistem.

ID Paket	IR Aktif	Servo	Target *Storage*	TL
JKT230705570	23:19:27	23:19:28	A	1,00 s
SKH260705570	23:18:49	23:18:50	D	1,00 s
SKH250705570	23:18:15	23:18:16	D	1,00 s

SKG240705570	23:17:46	23:17:47	E	1,00 s
SRG240705570	23:17:10	23:17:10	C	0,00 s
DIY240705589	23:16:45	23:16:46	C	1,00 s
DIY240705587	23:16:10	23:16:10	C	0,00 s
SRG260606126	22:50:09	22:50:09	C	0,00 s
Rata-rata (μ)				0,62 s
Deviasi Standar (σ)				0,51 s

Berdasarkan Tabel 2, nilai rata-rata *latency response* (TL) sistem adalah sebesar 0,62 detik dengan deviasi standar 0,51 detik. Perlu dicatat bahwa pencatatan *timestamp* pada tabel *activity_logs* menggunakan resolusi satuan detik penuh *(integer)*, sehingga seluruh nilai TL yang sesungguhnya berada di bawah 1.000 milidetik akan terbaca sebagai 0,00 detik, sedangkan nilai yang melewati batas satu detik tercatat sebagai 1,00 detik. Dengan demikian, nilai TL = 0,00 detik pada tiga sampel pengujian tidak berarti respons sistem terjadi secara instan, melainkan mencerminkan bahwa latensi aktual pada sampel tersebut lebih kecil dari satu detik namun tidak dapat dikuantifikasi lebih presisi dengan *resolusi log* yang tersedia. Nilai TL yang seluruhnya konsisten pada rentang 0–1 detik di seluruh 8 sampel pengujian mengindikasikan bahwa keterlambatan pada sisi perangkat lunak dapat dikesampingkan sebagai penyebab utama kegagalan penyortiran. Untuk memperoleh data latensi yang lebih akurat pada pengembangan selanjutnya, disarankan meningkatkan resolusi pencatatan *timestamp* menjadi satuan milidetik (ms).

3.) Analisis Kuantitatif Performa Sistem (System Testing)

Pengujian performa secara kontinu dilakukan dengan menjalankan siklus *sortir* terhadap 78 sampel barang secara acak untuk mengukur keandalan logika dan mekanis [4]. Setiap sampel diberi QR *Code* dengan variasi prefix yang mencakup kategori regional umum (*makro*), regional spesifik (*mikro*) yang saling beririsan, serta sejumlah kode yang sengaja tidak didaftarkan untuk menguji fungsi *fail-safe*. Tabel 3 menyajikan rekapitulasi hasil pengujian berdasarkan jalur penyimpanan tujuan.

Tabel 3. Rekapitulasi Hasil Pengujian Sortir terhadap 78 Sampel Barang

Storage	Jumlah Diuji	Sukses	Gagal	Akurasi (%)
A	12	12	0	100,0
B	24	19	5	79,2
C	19	16	3	84,2
D	15	11	4	73,3
E	8	6	2	75,0
Total	**78**	**64**	**14**	**82,1**

Berdasarkan data pada Tabel 3, sistem mencatatkan *Sorting Accuracy Rate* (SAR) sebesar 82,1% dan *Misplacement Rate* (MR) sebesar 17,9% dari 78 sampel yang diuji, sesuai dengan formula yang dijabarkan pada bagian metodologi penelitian. *Storage* A mencatatkan akurasi sempurna sebesar 100% tanpa satu pun kejadian salah arah. *Storage* D mencatatkan akurasi terendah (73,3%), sementara *Storage* B mencatatkan jumlah kejadian gagal absolut tertinggi (5 dari 24 sampel). *Storage* C berada pada posisi menengah dengan akurasi 84,2%.

Seluruh 14 kejadian *misplacement* yang teridentifikasi pada Tabel 3 memerlukan penelusuran lanjutan pada tabel *machine_logs* untuk memastikan penyebab pastinya. Berdasarkan pola pada *activity_logs*, beberapa kejadian gagal menunjukkan dua entri log untuk *barcode* yang sama dengan *storage* akhir yang berbeda dari hasil sebelumnya, mengindikasikan kemungkinan deteksi ganda akibat tumpang-tindih posisi fisik barang di atas konveyor. Namun, tanpa data *machine_logs* berstatus *CRITICAL*, atribusi penyebab kegagalan sepenuhnya pada faktor mekanis belum dapat dipastikan secara mutlak..

IV. KESIMPULAN

Berdasarkan hasil perancangan, implementasi, dan analisis kuantitatif yang telah dilakukan terhadap prototipe Sistem Kendali *Sortir* Barang Otomatis Berbasis *Internet of Things* (IoT) ini, Sistem telah berhasil mengintegrasikan

perangkat keras berbasis mikrokontroler ESP32 dengan *web dashboard* secara sinkron menggunakan protokol HTTP. Implementasi format data JSON terbukti efektif dalam memfasilitasi perubahan aturan pemisahan barang (logika *sortir*) secara dinamis di sisi perangkat lunak. Pendekatan ini berhasil mengatasi keterbatasan sistem *sortir* konvensional yang bersifat *hard-coded*, sehingga konfigurasi parameter tujuan distribusi dapat diperbarui secara *real-time* tanpa memerlukan proses pemrograman ulang (*flash firmware*) pada perangkat keras di lapangan.

Melalui pengujian performa secara kontinu, sistem menunjukkan tingkat keandalan logika yang tinggi dengan capaian *Sorting Accuracy Rate* (SAR) sebesar 82,1% dari total 78 sampel barang yang diuji. Pengukuran waktu respon menunjukkan nilai rata-rata *latency response* yang rendah, yakni sebesar 0,62 detik dengan deviasi standar 0,51 detik, yang mengonfirmasi bahwa aspek komputasi sistem berjalan dengan efisien.

Sistem mencatatkan *Misplacement Rate (MR)* sebesar 17,9%. Pola pada *activity_logs* mengindikasikan kemungkinan deteksi ganda akibat tumpang-tindih posisi fisik barang di atas konveyor sebagai salah satu faktor penyebab, namun atribusi penyebab kegagalan secara pasti memerlukan verifikasi lebih lanjut terhadap tabel *machine_logs* berstatus *CRITICAL* yang belum tersedia pada analisis ini. Tidak ditemukan indikasi kegagalan logika pencocokan *prefix* atau kesalahan pembacaan data JSON pada seluruh kasus *misplacement* yang tercatat.

Penerapan algoritma pencocokan *prefix* berbasis *string* terpanjang terbukti presisi dalam membedakan kode wilayah logistik yang saling beririsan (seperti *prefix makro* SOC dan *prefix mikro* SKH). Selain itu, integrasi sensor inframerah FC-51 di setiap gerbang bersama jalur penyimpanan darurat (*Storage* E) menghasilkan tingkat keberhasilan isolasi sebesar 75,0% (6 dari 8 sampel) untuk barang dengan QR *Code* yang rusak atau tidak terdaftar. Dua kasus kegagalan isolasi yang teridentifikasi menunjukkan bahwa mekanisme *fail-safe*, meskipun bekerja pada mayoritas kasus, masih memerlukan penyempurnaan lebih lanjut.

Untuk pengembangan selanjutnya, disarankan meningkatkan *resolusi timestamp log* menjadi satuan milidetik (ms) untuk memantau *true latency* secara lebih presisi, serta mengoptimalkan mekanisme konveyor guna menjaga jarak aman (gap) antar-barang secara konsisten.

REFERENSI

[1] T. J. Lukka, T. Tossavainen, J. V. Kujala, and T. Raiko, *"ZenRobotics Recycler – Robotic Sorting using Machine Learning,"* in Proc. Int. Conf. Sensor-Based Sorting (SBS), Aachen, Germany, Mar. 2014, p. 1. [Online]. Available: https://users.ics.aalto.fi/praiko/papers/SBS14.pdf

[2] P. Pramana and R. Mukhaiyar, *"Rancang Bangun Alat Penyortir Barang menggunakan Barcode Berbasis Mikrokontroler," Ranah Research: Journal of Multidisciplinary Research and Development*, vol. 4, no. 2, pp. 156–163, Mar. 2022, doi: 10.38035/RRJ.V4I2.454.

[3] T. Dewi, P. Risma, and Y. Oktarina, *"Fruit sorting robot based on color and size for an agricultural product packaging system," Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 4, pp. 1438–1445, Aug. 2020, doi: 10.11591/EEI.V9I4.2353.

[4] S. Zeng, R. Xue, and S. Wang, *"Research on Automatic Sorting System Based on RFID," in Proc. 2017 Int. Conf. Computational Science and Engineering (ICCSE 2017)*, Atlantis Press, 2017, doi: 10.2991/ICCSE-17.2017.39.

[5] A. H. Okilly, N. Kim, and J. Baek, *"Inrush Current Control of High Power Density DC–DC Converter," Energies*, vol. 13, no. 17, p. 4301, Aug. 2020, doi: 10.3390/EN13174301.

[6] Google for Developers, *"Scan barcodes with ML Kit on Android,"* Google LLC. [Online]. Available: https://developers.google.com/ml-kit/vision/barcode-scanning/android. [Accessed: 21-Jun-2026].

[7] Espressif Systems, *"ESP32 Technical Reference Manual,"* Espressif Systems Co., Ltd., Shanghai, China, Tech. Ref. Manual ver. 5.1, 2023. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[8] C. Mega, *"Rancang Bangun Sistem Sortir Barang Menggunakan QR Code Berbasis Arduino Mega,"* in Proc. Seminar Nasional Teknologi Informasi (SENAFTI), Universitas Budi Luhur, 2023. [Online]. Available: https://senafti.budiluhur.ac.id/senafti/article/download/1642/925. [Accessed: 21-Jun-2026].

[9] E. Budiarto, *"Rancang Bangun Automatic Wheel Sorter Konveyor pada Jasa Ekspedisi Menggunakan Quick Response Code,"* Laporan Akhir, Politeknik Manufaktur Bandung, 2025. [Online].

Available: https://repositori.polman-bandung.ac.id/id/eprint/833/. [Accessed: 21-Jun-2026].

[10] S. M. Kulkarni, S. Umadi, S. S. Airani, A. R. M. Raikar, and M. M. Raikar, *"Grading and classification of tomatoes using ESP32-CAM and impulse cloud based IoT platform for sustainability,"* Procedia Computer Science, vol. 260, pp. 938–946, 2025, doi: 10.1016/j.procs.2025.03.277.

[11] Y. H. Prasetyo, L. Ixbal, U. B. Gea, I. Dwisaputra, and Y. F. Arriyani, *"Sistem Sortir Paket Otomatis dengan Conveyor dan Kamera ESP32-CAM,"* in Proc. Seminar Nasional Inovasi Teknologi Terapan (SNITT), Polman Babel, 2025. [Online]. Available: http://snitt.polman-babel.ac.id/index.php/snitt/article/view/598. [Accessed: 21-Jun-2026].

[12] C. Austin, M. Mulyadi, and S. Octaviani, *"Implementasi IoT dengan ESP32 untuk Pemantauan Kondisi Suhu Secara Jarak Jauh Menggunakan MQTT pada AWS,"* Jurnal Elektro, vol. 15, no. 2, 2022. [Online]. Available: https://ejournal.atmajaya.ac.id/index.php/JTE/article/view/5141. [Accessed: 21-Jun-2026].

[13] I. R. Nugraha, W. H. N. Putra, and E. Setiawan, *"A comparative study of HTTP and MQTT for IoT applications in hydroponics,"* Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), vol. 8, no. 1, pp. 119–126, 2024, doi: 10.29207/resti.v8i1.5561.

[14] B. Dhiya' Ushofa, L. Anifah, I. G. P. A. Buditjahjanto, and Endryansyah, *"Sistem Kendali Kecepatan Putaran Motor DC pada Conveyor dengan Metode Kontrol PID,"* Jurnal Teknik Elektro, vol. 11, no. 2, 2022. [Online]. Available: https://ejournal.unesa.ac.id/index.php/JTE/article/download/48616/40560. [Accessed: 21-Jun-2026].

[15] A. R. Hakim, K. Harefa, and B. Widodo, *"Pengembangan Sistem Informasi Akademik Berbasis Android Menggunakan Flutter di Politeknik,"* SCAN: Jurnal Teknologi Informasi dan Komunikasi, vol. 14, no. 3, pp. 27–32, 2019. [Online]. Available: http://ejournal.upnjatim.ac.id/index.php/scan/article/view/1684. [Accessed: 21-Jun-2026].

[16] R. Al Hayubi, S. Aulia, D. A. Gunawan, S. Hidayatullah, and D. Aribowo, *"Implementasi Sistem Penggerak Servo SG90 Berbasis Arduino Uno dengan Kontrol Sudut Dinamis,"* Mars: Jurnal Teknik Mesin, Industri, Elektro dan Ilmu Komputer, vol. 2, no. 6, pp. 130–140, Dec. 2024, doi: 10.61132/mars.v2i6.535