

Sistem Inspeksi Visual Hasil *Stamping* Berbasis YOLOv8 pada Raspberry Pi 4B Terintegrasi ESP32 dan Firebase

Rani Ariesta Nur Aprilia¹⁾, Tatyantoro Andrasto²⁾, Arief Arfriandi³⁾

¹⁾²⁾³⁾Teknik Elektro, Fakultas Teknik, Universitas Negeri Semarang

¹⁾raniaariesta@students.unnes.ac.id, ²⁾tatyantoro@mail.unnes.ac.id, ³⁾afriandi@mail.unnes.ac.id



*Rani Ariesta Nur Aprilia

Histori Artikel:

Submit: 2026-06-15

Diterima: 2026-06-18

Dipublikasikan: 2026-06-19

Kata Kunci:

Edge Computing; ESP32;

Firebase; Inspeksi Visual;

Raspberry Pi 4 Model B;

YOLOv8.

ABSTRAK

Kemajuan Industri 4.0 mendorong kebutuhan sistem inspeksi visual otomatis untuk menggantikan proses inspeksi manual yang rentan terhadap kesalahan operator pada industri percetakan, khususnya dalam mendeteksi cacat hasil *stamping*. Penelitian ini bertujuan mengimplementasikan sistem inspeksi visual berbasis YOLOv8 pada Raspberry Pi 4B sebagai perangkat *edge computing* yang terintegrasi dengan ESP32 dan Firebase *Realtime Database* untuk pemantauan berbasis IoT secara *real-time*. Sistem dirancang menggunakan kamera Pi CSI OV5647 untuk akuisisi citra, model YOLOv8 untuk mengklasifikasikan produk ke dalam kategori *Good* dan *Not Good*, ESP32 sebagai pengendali servo pemilah produk melalui protokol HTTP GET, serta Firebase *Realtime Database* sebagai media penyimpanan dan pemantauan berbasis *cloud* melalui REST API. Pengujian dilakukan pada prototipe mesin *stamping* otomatis di lingkungan laboratorium dengan total 907 *frame* yang diproses secara kontinu. Hasil pengujian menunjukkan bahwa sistem berhasil mendeteksi dan mengklasifikasikan hasil *stamping* dengan nilai *confidence* rata-rata 0,7097 untuk kategori *Good* dan 0,7398 untuk kategori *Not Good*. Raspberry Pi 4B mampu menjalankan inferensi secara *real-time* dengan rata-rata *latency* 372,18 ms dan kecepatan pemrosesan 2,69 FPS, serta penggunaan CPU, RAM, dan suhu prosesor yang stabil masing-masing sebesar 42,53%, 23,08%, dan 62,27°C. Seluruh data hasil klasifikasi berhasil dikirimkan ke ESP32 dan Firebase tanpa kegagalan pengiriman, dengan rata-rata latensi masing-masing 294,46 ms dan 1456,60 ms. Hasil penelitian menunjukkan bahwa Raspberry Pi 4B mampu berfungsi sebagai perangkat pemrosesan utama sistem inspeksi visual berbasis YOLOv8 yang terintegrasi dengan teknologi IoT untuk pemantauan *real-time*.

Jurnal Pendidikan Sains dan Komputer is licensed under a Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0).

LATAR BELAKANG

Kemajuan Industri 4.0 mendorong sektor manufaktur beralih dari proses inspeksi manual menuju sistem otomatisasi berbasis teknologi cerdas. Namun, proses inspeksi yang masih dilakukan secara manual rentan terhadap kesalahan akibat keterbatasan persepsi manusia, seperti posisi *stamping* yang miring, tidak sesuai kolom, atau menutupi informasi penting pada produk, sehingga dapat mengganggu standar kualitas dan alur produksi (Kim 2026). Kondisi tersebut diperkuat oleh laporan bahwa tingkat akurasi inspeksi visual manual di lingkungan industri hanya mencapai sekitar 80%, sehingga masih terdapat kemungkinan produk cacat lolos ke tahap produksi berikutnya (Sundaram and Zeid 2023). Oleh karena itu, sistem inspeksi visual otomatis diperlukan untuk meningkatkan konsistensi, kecepatan, dan akurasi proses pengendalian kualitas (Luo and Wu 2025). Salah satu proses yang memerlukan inspeksi kualitas secara konsisten adalah proses *stamping* pada mesin percetakan, di mana hasil cetak harus memenuhi standar posisi dan keterbacaan yang ketat sebelum produk diteruskan ke tahap berikutnya. Penelitian ini menggunakan prototipe mesin *stamping* otomatis sebagai objek implementasi sistem inspeksi visual yang dikembangkan.

Perkembangan kecerdasan buatan dan teknologi pemrosesan citra mendukung penerapan sistem inspeksi



visual otomatis secara luas melalui pendekatan *machine vision* yang dikombinasikan dengan metode *deep learning* (Luo & Wu, 2025). Pendekatan *deep learning* berbasis *computer vision* mampu mengenali dan mengklasifikasikan objek secara *real-time* dengan akurasi tinggi (Setyanto, Sasongko, and Kim 2025), mendorong pengembangan algoritma deteksi objek modern seperti *You Only Look Once* (YOLO) yang menggunakan pendekatan *single-shot end-to-end* untuk mendeteksi banyak objek secara cepat dan akurat. Di antara berbagai versi YOLO yang tersedia, YOLOv8 dipilih dalam penelitian ini karena memiliki performa deteksi yang lebih unggul dibandingkan versi sebelumnya (Hayati, Singasatia, and Muttaqin 2023).

Meskipun menawarkan performa yang baik, implementasi *deep learning* pada sistem inspeksi visual tetap menghadapi tantangan keterbatasan sumber daya komputasi pada perangkat *embedded*. Model yang digunakan tidak hanya harus akurat, tetapi juga ringan dan efisien agar dapat dijalankan pada perangkat *edge AI* (Singh and Singh 2023). Raspberry Pi 4B dipilih sebagai perangkat *edge computing* utama dalam penelitian ini karena menawarkan fleksibilitas *deployment* yang lebih sederhana dibandingkan perangkat lain seperti NVIDIA Jetson Nano yang memerlukan konversi model ONNX dan optimasi TensorRT sebelum digunakan (Azab, Ehab, and Shihata 2026). Kemampuan integrasi Raspberry Pi dengan model *deep learning* untuk deteksi objek secara *real-time* tanpa ketergantungan pada server eksternal telah dibuktikan sebelumnya (Nugraha, Regasari, and Putri 2026).

Hasil klasifikasi dari sistem inspeksi visual perlu diteruskan ke sistem kendali dan pemantauan secara *real-time* melalui integrasi Internet of Things (IoT). ESP32 dipilih sebagai modul komunikasi karena dilengkapi kemampuan komunikasi nirkabel *Wi-Fi* dan mendukung protokol HTTP (Azhar and Nurpulaela 2024). Data hasil klasifikasi dikirimkan ke ESP32 untuk mengendalikan servo yang memilah produk *Good* dan *Not Good*, serta diunggah ke Firebase *Realtime Database* untuk pemantauan berbasis *cloud*. Firebase *Realtime Database* merupakan layanan basis data berbasis *cloud* dari Google yang menyimpan dan menyinkronkan data secara *real-time* melalui koneksi internet, sehingga hasil inspeksi dapat dipantau kapan saja melalui *dashboard* berbasis web (Margapuri and Neilsen 2021). Pengiriman data ke Firebase dilakukan menggunakan REST API yang mengirimkan data dalam format JSON, sedangkan pengiriman data hasil klasifikasi ke ESP32 melalui protokol HTTP GET. Kombinasi arsitektur ini memungkinkan sistem inspeksi beroperasi sepenuhnya secara otomatis dan terhubung dengan ekosistem IoT.

Penelitian ini bertujuan untuk: (1) mengimplementasikan sistem inspeksi visual berbasis YOLOv8 pada Raspberry Pi 4B yang mampu mendeteksi dan mengklasifikasikan kualitas hasil *stamping* ke dalam kategori *Good* dan *Not Good* secara *real-time*; (2) menganalisis kinerja Raspberry Pi 4B selama menjalankan YOLOv8, meliputi *latency* inferensi, FPS, serta penggunaan CPU dan memori; dan (3) merancang serta menguji sistem komunikasi data antara Raspberry Pi 4B, ESP32 dan Firebase *Realtime Database* menggunakan protokol HTTP dan REST API sebagai jalur pengiriman hasil klasifikasi menuju sistem IoT dan kendali pada prototipe mesin *stamping* otomatis. Perlu dicatat bahwa model YOLOv8 yang digunakan dalam penelitian ini telah melalui proses *training* sebelumnya, sehingga penelitian ini berfokus pada implementasi dan evaluasi sistem secara keseluruhan, bukan pada proses pelatihan model.

Penelitian ini berfokus pada implementasi YOLOv8 pada Raspberry Pi 4B sebagai perangkat pemrosesan utama sistem inspeksi visual hasil *stamping*. Model YOLOv8 yang digunakan telah dilatih sebelumnya, sehingga penelitian ini tidak membahas *dataset*, proses pelatihan dan validasi model, maupun parameter evaluasi model deteksi. Fokus penelitian diarahkan pada evaluasi kinerja Raspberry Pi 4B dalam menjalankan inferensi YOLOv8 serta integrasi sistem dengan ESP32 dan Firebase *Realtime Database*.

STUDI LITERATUR

Beberapa penelitian terdahulu telah mengkaji penggunaan YOLOv8 pada perangkat *edge* berbasis Raspberry Pi untuk keperluan deteksi objek secara *real-time*. Implementasi *deep learning* pada perangkat *edge* membutuhkan perangkat yang mampu menjalankan inferensi secara lokal tanpa ketergantungan server eksternal. Raspberry Pi merupakan komputer mini berbasis *single-board computer* yang dikembangkan oleh Raspberry Pi Foundation, dilengkapi dengan 40 pin GPIO, prosesor ARM, serta kompatibilitas terhadap berbagai sistem operasi Linux (Ahmed, Aboluhom, and Kandilli 2024). Di antara seluruh variannya, Raspberry Pi 4B menawarkan peningkatan signifikan dengan prosesor *Quad-core Cortex-A72* 1.5 GHz, kapasitas memori hingga 8 GB LPDDR4, serta *port* CSI untuk

akuisisi citra berkecepatan tinggi dan latensi rendah (Isnaen and Stefanie 2023), menjadikannya pilihan utama sebagai perangkat *edge computing* untuk sistem inspeksi visual berbasis *deep learning*.

Pada penelitian (Nugraha, Regasari, dan Putri 2026) sistem deteksi pelanggaran alat pelindung diri berbasis Raspberry Pi 4 dengan multi-kamera menggunakan YOLOv8, membuktikan bahwa Raspberry Pi 4B mampu menjalankan inferensi YOLOv8 secara kontinu dalam skenario industri dengan notifikasi *real-time* tanpa ketergantungan pada server eksternal. (Noon et al. 2025) mengembangkan sistem kendali gerbang otomatis berbasis pengenalan stiker kendaraan secara *real-time* menggunakan YOLOv8 versi ringan yang dioptimasi melalui kuantisasi dan pemangkasan parameter pada Raspberry Pi 4; sistem diuji pada 506 citra dalam berbagai kondisi pencahayaan dan mencapai mAP@0.5 sebesar 99% dengan kecepatan pemrosesan 3–5 FPS, membuktikan bahwa YOLOv8 teroptimasi dapat berjalan efektif pada perangkat *edge* meskipun masih dipengaruhi kondisi pencahayaan yang kurang optimal. Sementara itu, (Okano et al. 2025) mengimplementasikan YOLOv8 Nano dan YOLOv8 Small pada Raspberry Pi untuk inspeksi kesesuaian *visual* industri pada pipa poliamida hasil ekstrusi di sektor otomotif, dengan waktu pemrosesan sekitar 470 ms per citra dan nilai presisi serta mAP yang tetap tinggi, membuktikan kelayakan YOLOv8 sebagai solusi inspeksi kualitas otomatis di lingkungan industri tanpa infrastruktur komputasi berkinerja tinggi, yang menjadi acuan perbandingan performa dalam penelitian ini.

Di bidang inspeksi visual berbasis *deep learning*, YOLOv8 merupakan versi terbaru dari keluarga algoritma YOLO yang menggunakan arsitektur *backbone* CSP lebih ringan serta *decoupled head* yang memisahkan proses klasifikasi dan regresi *bounding box*, sehingga prediksi menjadi lebih stabil dan akurat dengan nilai mAP sekitar 2% lebih tinggi dibandingkan YOLOv5 (Sohan, Sairam, and Author 2024). Keunggulan arsitektur ini telah dimanfaatkan pada berbagai skenario inspeksi industri, salah satunya pada inspeksi cacat permukaan melengkung dan mengkilap menggunakan metode optimasi jalur pemindaian berbasis YOLO yang membuktikan bahwa algoritma ini dapat diadaptasi pada berbagai karakteristik permukaan produk industri (Kim 2026). Namun penelitian-penelitian tersebut secara umum masih berfokus pada performa deteksi, tanpa mengintegrasikan hasil inspeksi ke sistem kendali mesin dan IoT secara langsung.

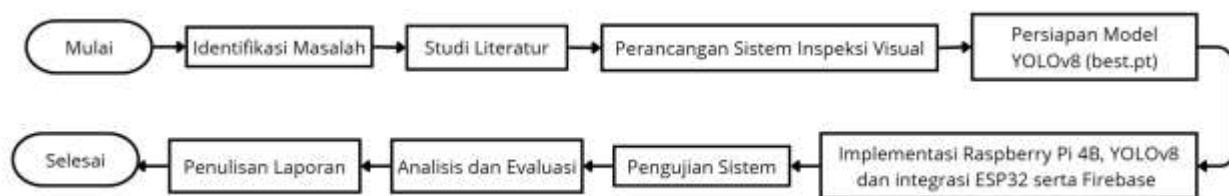
Pada aspek integrasi kontrol, ESP32 merupakan mikrokontroler *System-on-Chip* (SoC) yang dikembangkan oleh Espressif Systems, dilengkapi fitur komunikasi nirkabel *Wi-Fi* dan Bluetooth terintegrasi, konsumsi daya rendah, serta mendukung berbagai antarmuka komunikasi seperti UART, SPI, dan I2C (Nizam et al. 2022). Dalam penelitian ini, ESP32 berperan sebagai penerima data hasil klasifikasi dari Raspberry Pi 4B sekaligus pengendali servo yang memilah produk *Good* dan *Not Good* secara otomatis berdasarkan hasil inspeksi. Pengiriman data dari Raspberry Pi 4B ke ESP32 dilakukan menggunakan protokol HTTP dan TCP/IP merupakan pilihan yang andal dan ringan untuk komunikasi data (Khalid et al. 2024), di mana hasil klasifikasi dikirimkan sebagai parameter URL yang kemudian dibaca dan dieksekusi oleh ESP32 untuk menggerakkan servo pemilah produk. Firebase *Realtime* Database merupakan layanan basis data berbasis *cloud* dari Google yang menggunakan struktur data JSON dan memungkinkan sinkronisasi data secara *real-time* antara perangkat dan aplikasi yang terhubung melalui internet tanpa perlu melakukan penyegaran manual (Mahendra and Winardi 2023). Penelitian (Margapuri and Neilsen 2021) membuktikan bahwa integrasi Raspberry Pi dengan Firebase mampu mendukung sistem pemantauan dan notifikasi berbasis IoT secara *real-time* melalui pengembangan sistem keamanan berbasis IoT menggunakan sensor PIR dan kamera yang terhubung ke Firebase *Realtime* Database, dengan implementasi yang relatif sederhana dan biaya yang terjangkau.

Berbagai penelitian tersebut menunjukkan bahwa implementasi YOLOv8 pada Raspberry Pi 4B mampu mendukung deteksi dan klasifikasi objek secara *real-time* pada lingkungan *edge computing* dengan tingkat akurasi dan waktu inferensi yang memadai. Penelitian sebelumnya juga membuktikan bahwa ESP32 dan Firebase *Realtime* Database dapat digunakan sebagai solusi komunikasi data dan pemantauan berbasis IoT yang ringan serta mudah diimplementasikan. Meskipun demikian, sebagian besar penelitian masih berfokus pada aspek tertentu secara terpisah, seperti optimasi performa deteksi YOLOv8 pada Raspberry Pi, pengembangan sistem kendali berbasis mikrokontroler, atau implementasi pemantauan berbasis *cloud*. Penelitian-penelitian tersebut belum banyak

membahas integrasi menyeluruh antara sistem inspeksi visual berbasis *deep learning*, sistem kendali, dan sistem pemantauan IoT dalam satu arsitektur yang terpadu pada aplikasi inspeksi kualitas hasil *stamping*. Selain itu, sebagian besar penelitian lebih menitikberatkan pada peningkatan akurasi model deteksi, sedangkan evaluasi implementasi YOLOv8 pada Raspberry Pi 4B sebagai perangkat pemrosesan utama (*edge computing*) yang terintegrasi dengan ESP32 dan Firebase masih relatif terbatas. Oleh karena itu, penelitian ini memposisikan Raspberry Pi 4B sebagai pusat pemrosesan sistem inspeksi visual yang menjalankan inferensi YOLOv8 secara *real-time*, kemudian mengintegrasikan hasil klasifikasi dengan ESP32 untuk pengendalian servo dan Firebase *Realtime* Database untuk pemantauan berbasis *cloud*. Dengan demikian, penelitian ini tidak hanya mengevaluasi kemampuan deteksi pada perangkat *edge*, tetapi juga mengkaji kinerja sistem secara keseluruhan dari sisi pemrosesan, komunikasi data, dan integrasi IoT.

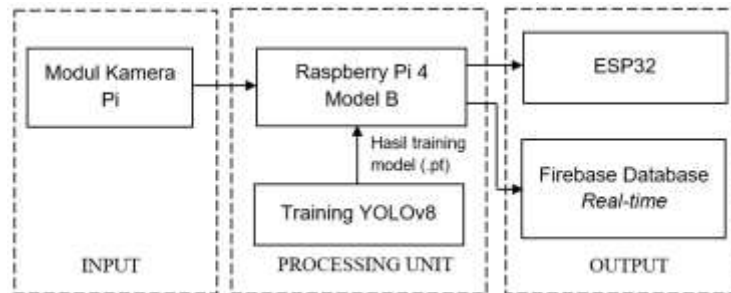
METODE

Penelitian ini menggunakan metode *Research and Development* (R&D). *Research and Development* (R&D) merupakan metode penelitian yang digunakan untuk menghasilkan produk baru atau menyempurnakan produk yang telah ada melalui serangkaian tahapan yang sistematis dan terencana (Sugiyono 2021). Oleh karena itu, metode R&D dipilih karena penelitian ini berfokus pada perancangan, implementasi, dan pengujian sistem inspeksi visual berbasis YOLOv8 pada Raspberry Pi 4B. Dalam penelitian ini, model YOLOv8 yang digunakan telah melalui proses pelatihan sebelumnya sehingga aspek pengembangan model seperti penyusunan *dataset*, jumlah data pelatihan, proses *training* dan validasi, serta evaluasi performa model deteksi tidak menjadi ruang lingkup penelitian. Fokus penelitian diarahkan pada implementasi sistem dan evaluasi kinerja Raspberry Pi 4B sebagai perangkat pemrosesan utama (*edge computing*) dalam menjalankan inferensi YOLOv8 secara *real-time*. Alur penelitian secara keseluruhan ditunjukkan pada Gambar 1.

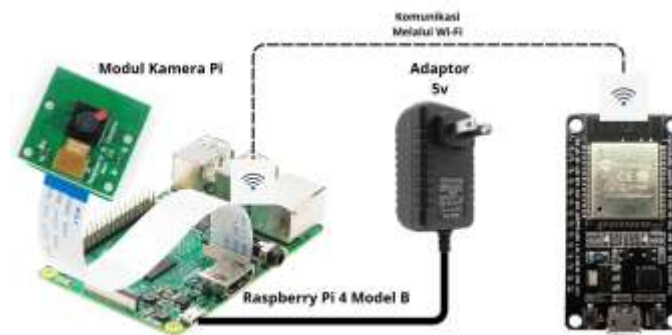


Gambar 1. Diagram Alur Penelitian

Tahap pertama adalah perancangan arsitektur sistem. Sistem dirancang dengan Raspberry Pi 4B sebagai unit pemrosesan utama yang menjalankan inferensi YOLOv8 menggunakan model hasil pelatihan sebelumnya (best.pt) berbasis framework PyTorch. Akuisisi citra dilakukan menggunakan kamera Pi CSI OV5647, sedangkan ESP32 digunakan sebagai pengendali servo pemilah produk, dan Firebase *Realtime* Database digunakan sebagai media penyimpanan serta pemantauan data secara *real-time*. Diagram blok arsitektur sistem ditunjukkan pada Gambar 2, sedangkan perancangan *hardware* secara lengkap ditunjukkan pada Gambar 3. Raspberry Pi 4B dan ESP32 dihubungkan melalui jaringan *Wi-Fi* lokal menggunakan protokol HTTP GET, sedangkan pengiriman data ke Firebase dilakukan melalui REST API dengan format JSON.

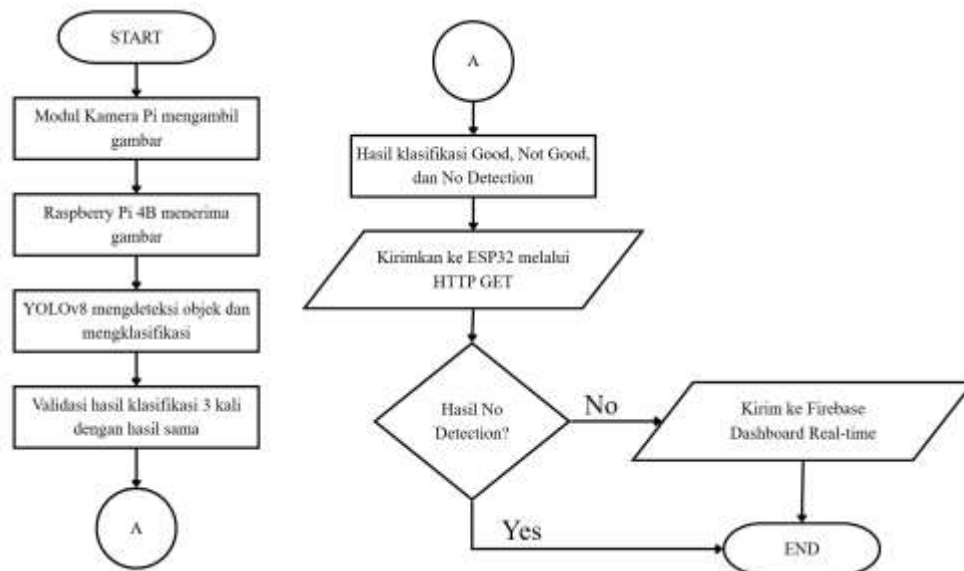


Gambar 2. Diagram Blok Sistem



Gambar 3. Perancangan Hardware

Tahap kedua adalah implementasi perangkat lunak. Program inferensi dikembangkan menggunakan bahasa Python dengan memanfaatkan *library* Ultralytics YOLOv8 untuk proses deteksi dan klasifikasi objek, Picamera2 untuk akuisisi citra melalui kamera Pi CSI, serta *library requests* untuk pengiriman data ke ESP32 dan Firebase. Seluruh program dijalankan pada Raspberry Pi 4B yang diakses melalui koneksi RealVNC. Model YOLOv8 yang digunakan merupakan model hasil training sebelumnya dalam format best.pt dengan dua kelas output, yaitu *Good* dan *Not Good*. Setelah citra diperoleh dari kamera, Raspberry Pi 4B melakukan proses inferensi menggunakan model tersebut untuk menentukan kategori hasil stamping. Untuk meningkatkan kestabilan hasil deteksi, sistem menerapkan mekanisme konfirmasi tiga frame berturut-turut ($CONFIRM_NEEDED = 3$) sebelum hasil klasifikasi dikirimkan ke ESP32 dan Firebase. Selain itu, proses inferensi dilakukan pada resolusi citra 320×240 piksel dengan *confidence threshold* sebesar 0,5, Alur kerja perangkat lunak secara keseluruhan ditunjukkan pada Gambar 4.



Gambar 4. Flowchart Sistem Inspeksi Visual

Tahap ketiga adalah pengujian sistem. Pengujian dilakukan pada prototipe mesin *stamping* otomatis di lingkungan laboratorium dengan total 907 *frame* yang diproses secara kontinu. Parameter yang diuji meliputi *latency* inferensi YOLOv8, kecepatan pemrosesan (FPS), penggunaan CPU dan RAM Raspberry Pi 4B, suhu prosesor, serta *latency* pengiriman data ke ESP32 dan Firebase. Data kinerja sistem dicatat secara otomatis menggunakan *library* *psutil* dan fungsi *time.perf_counter()* ke dalam file CSV selama sistem berjalan, kemudian dianalisis menggunakan Google Colab untuk visualisasi grafik.

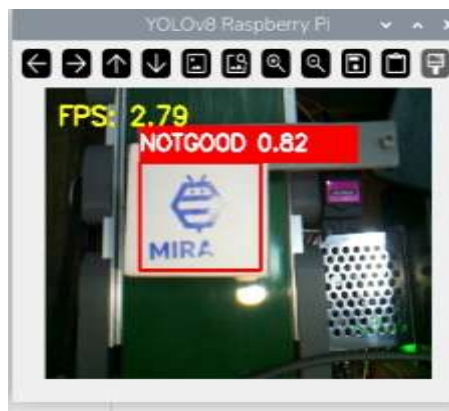
HASIL

Sistem inspeksi visual berhasil diimplementasikan pada prototipe mesin *stamping* otomatis dengan Raspberry Pi 4B sebagai unit pemrosesan utama. Kamera Pi CSI OV5647 digunakan untuk akuisisi citra hasil *stamping* pada resolusi 320×240 piksel, kemudian citra diproses oleh model YOLOv8 untuk mengklasifikasikan produk ke dalam kategori *Good* atau *Not Good*.

Pada aspek deteksi dan klasifikasi, pengujian dilakukan selama sistem memproses 907 *frame* secara kontinu. Dari total *frame* tersebut, sistem berhasil mengonfirmasi 12 sampel valid yang terdiri dari 6 sampel *Good* dan 6 sampel *Not Good* melalui mekanisme konfirmasi 3 *frame* berturut-turut sebagaimana ditunjukkan pada Tabel 1. Nilai *confidence* deteksi untuk kategori *Good* berada pada rentang 0,5099–0,8299 dengan rata-rata 0,7097, sedangkan untuk kategori *Not Good* berada pada rentang 0,6471–0,8299 dengan rata-rata 0,7398. Selisih rata-rata *confidence* antara kedua kelas yang relatif kecil mengindikasikan bahwa perbedaan karakteristik visual antara objek *Good* dan *Not Good* tidak ekstrem, sehingga model membutuhkan fitur visual yang konsisten untuk membedakan keduanya secara akurat. Hasil klasifikasi *Good* dan *Not Good* ditunjukkan pada Gambar 5 dan Gambar 6.



Gambar 5. Hasil klasifikasi Good



Gambar 6. Hasil klasifikasi Not Good

Tabel 1. Hasil pengujian deteksi dan klasifikasi YOLOv8

Frame	Hasil Deteksi	Confidance	FPS	Latency Inferensi (ms)
3	Good	0,6842	2,76	362,16
83	Not Good	0,7459	2,64	379,48
155	Good	0,5099	2,80	381,60
251	Not Good	0,8299	2,79	358,26
375	Good	0,7902	2,75	363,08
427	Not Good	0,6499	2,67	374,72
459	Good	0,7874	2,71	368,49
515	Not Good	0,7949	2,57	389,41
594	Good	0,7639	2,71	369,09
697	Not Good	0,7581	2,59	386,75
842	Good	0,7228	2,64	378,32
907	Not Good	0,6602	2,82	354,76

Selain hasil klasifikasi utama, terdapat kondisi di mana sistem tidak menghasilkan keluaran deteksi (*No Detection*). Salah satu kondisi yang diamati ditunjukkan pada Gambar 7, yaitu ketika hasil *stamping* berada dalam posisi terbalik sehingga objek tidak berhasil dikenali oleh model YOLOv8. Kondisi ini menunjukkan bahwa hasil deteksi masih dipengaruhi oleh orientasi objek saat proses inspeksi berlangsung. Perubahan orientasi dapat menyebabkan tampilan objek berbeda dari pola yang dikenali oleh model, sehingga sistem gagal mendeteksi objek dan menghasilkan *false negative*. Selain itu, faktor lain seperti pencahayaan, posisi kamera, dan kualitas citra juga

dapat memengaruhi keberhasilan deteksi karena berpengaruh terhadap kejelasan area *stamping* yang diamati. Hasil pengujian ini menunjukkan bahwa meskipun sistem mampu bekerja secara *real-time*, performa deteksi tetap dipengaruhi oleh kondisi pengambilan citra. Oleh karena itu, pengembangan selanjutnya dapat dilakukan dengan menambahkan variasi orientasi objek dan kondisi pencahayaan agar hasil deteksi lebih stabil dan konsisten.



Gambar 7. Keadaan No detection

Pada aspek kinerja perangkat, pengujian dilakukan selama proses inferensi, komunikasi data ke ESP32, dan pengiriman data ke Firebase berlangsung secara bersamaan. Parameter yang diamati meliputi *latency* inferensi, FPS, penggunaan CPU, penggunaan RAM, dan suhu prosesor. Data diperoleh secara otomatis menggunakan pustaka psutil dan fungsi `time.perf_counter()` ke dalam file CSV selama sistem memproses 907 *frame*. Hasil pengujian kinerja Raspberry Pi 4B ditunjukkan pada Tabel 2.

Tabel 2. Hasil Pengujian Kinerja Raspberry Pi 4 Model B

Parameter	Minimum	Maksimum	Rata-rata
<i>Latency</i> Inferensi (ms)	354,76	389,41	372,18
FPS (<i>frame</i> /detik)	2,57	2,82	2,69
Penggunaan CPU (%)	21,6	52,7	42,53
Penggunaan RAM (%)	21,50	24,60	23,08
Suhu Prosesor (°C)	56,7	65,2	62,27

Pada aspek komunikasi data, seluruh data hasil klasifikasi berhasil dikirimkan ke ESP32 dan Firebase *Realtime* Database tanpa kegagalan pengiriman, sebagaimana ditunjukkan pada Tabel 3. Rata-rata waktu pengiriman ke ESP32 sebesar 294,46 ms dengan rentang 67,55–354,68 ms, sedangkan rata-rata waktu pengiriman ke Firebase sebesar 1.456,60 ms dengan rentang 901,60–2.163,85 ms.

Tabel 3. Hasil pengiriman data ESP32 dan Firebase

<i>Frame</i>	Status Kirim	Waktu ESP32 (ms)	Waktu Firebase (ms)	Status ESP32	Status Firebase
3	<i>Good</i>	67,55	901,60	Berhasil	OK
83	<i>Not Good</i>	171,15	1382,20	Berhasil	OK
155	<i>Good</i>	316,44	1424,26	Berhasil	OK
251	<i>Not Good</i>	331,72	2163,85	Berhasil	OK
375	<i>Good</i>	312,85	1750,10	Berhasil	OK

427	<i>Not Good</i>	345,63	968,99	Berhasil	OK
459	<i>Good</i>	328,17	1283,19	Berhasil	OK
515	<i>Not Good</i>	318,90	1251,18	Berhasil	OK
594	<i>Good</i>	337,46	2173,63	Berhasil	OK
697	<i>Not Good</i>	324,58	2078,30	Berhasil	OK
842	<i>Good</i>	348.43	952,15	Berhasil	OK
907	<i>Not Good</i>	354.68	1149,71	Berhasil	OK

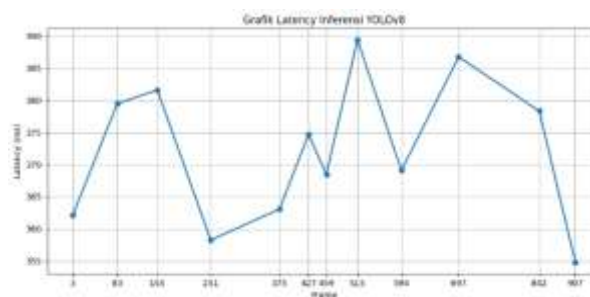
PEMBAHASAN

Sistem inspeksi visual berbasis YOLOv8 pada Raspberry Pi 4B berhasil mendeteksi dan mengklasifikasikan hasil *stamping* secara *real-time*. Mekanisme konfirmasi 3 *frame* berturut-turut yang diterapkan terbukti meningkatkan kestabilan klasifikasi, karena sistem tidak mengirimkan data berdasarkan satu *frame* tunggal yang rentan terhadap deteksi *false positive*. Interval *frame* yang panjang antar pengiriman mencerminkan kombinasi waktu tunggu objek memasuki area pengamatan, akumulasi konfirmasi, dan periode *No Detection*, bukan indikasi kegagalan sistem.

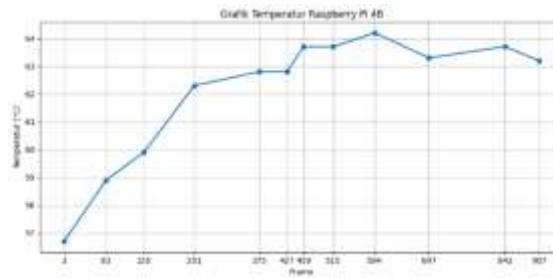
Selisih rata-rata *confidence* antara kategori *Good* 0,7097 dan *Not Good* 0,7398 yang relatif kecil mengindikasikan bahwa perbedaan karakteristik visual antar kelas tidak ekstrem, sehingga kualitas pencahayaan dan posisi kamera menjadi faktor penting dalam menjaga konsistensi deteksi.

Perbedaan latensi antara pengiriman ke ESP32 dan Firebase disebabkan oleh perbedaan arsitektur komunikasi. Pengiriman ke ESP32 melalui jaringan *Wi-Fi* lokal menghasilkan rata-rata 294,46 ms yang cukup responsif untuk kendali servo secara otomatis. Sementara pengiriman ke Firebase melalui REST API dan internet publik menghasilkan rata-rata 1456,60 ms dengan variasi lebih besar akibat ketergantungan pada kondisi jaringan eksternal. Meskipun demikian, seluruh data berhasil tersimpan dengan status OK, sehingga latensi Firebase masih dalam kisaran yang dapat diterima untuk kebutuhan monitoring *real-time* berbasis *cloud*.

Latency inferensi rata-rata 372,18 ms lebih baik dibandingkan hasil (Okano et al. 2025) yang melaporkan sekitar 470 ms per citra pada implementasi YOLOv8 serupa di Raspberry Pi. Nilai FPS rata-rata 2,69 *frame*/detik memang di bawah standar video *real-time* (≥ 24 FPS), namun memadai untuk kebutuhan inspeksi *stamping* pada prototipe ini mengingat objek bergerak lambat di *conveyor*, sejalan dengan pernyataan (Noon et al. 2025) bahwa 3–5 FPS sudah efektif untuk sistem *edge computing*. Penggunaan CPU rata-rata 42,53% menunjukkan beban komputasi yang signifikan namun stabil, dan suhu prosesor rata-rata 62,27°C masih jauh di bawah batas *throttling* 80°C sehingga tidak terjadi *overheating* selama pengujian berlangsung.



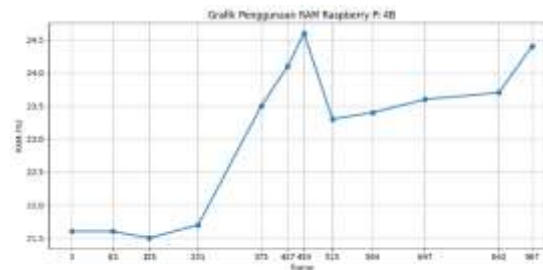
Gambar 8. Grafik latency inferensi YOLOv8 pada Raspberry Pi 4 B



Gambar 9. Grafik temperature Raspberry Pi 4B



Gambar 10. Grafik penggunaan CPU Raspberry Pi 4B



Gambar 11. Grafik Penggunaan RAM Raspberry Pi 4B

Grafik *latency* inferensi pada Gambar 11. menunjukkan distribusi yang stabil sepanjang 907 *frame* tanpa tren kenaikan, mengindikasikan tidak adanya degradasi performa akibat akumulasi beban proses selama inferensi kontinu berlangsung. Grafik suhu prosesor pada Gambar 12 memperlihatkan pola kenaikan bertahap di awal pengujian yang kemudian menstabilkan diri pada rentang 60–65°C, yang merupakan respons termal wajar pada perangkat *embedded* tanpa pendingin aktif. Grafik penggunaan CPU pada Gambar 13 menunjukkan lonjakan sesekali yang berkorelasi dengan momen pengiriman data ke ESP32 dan Firebase secara bersamaan, sehingga menjelaskan selisih antara nilai rata-rata (42,53%) dan nilai maksimum (52,7%). Grafik penggunaan RAM pada Gambar 14 menampilkan pola yang sangat stabil tanpa tren kenaikan sepanjang pengujian, mengonfirmasi tidak adanya *memory leak* pada program inferensi meskipun sistem berjalan selama ratusan *frame*.

Meskipun sistem mampu menjalankan inferensi secara *real-time* dan menghasilkan klasifikasi yang stabil melalui mekanisme konfirmasi tiga *frame* berturut-turut, hasil pengujian menunjukkan bahwa performa sistem masih dapat dipengaruhi oleh kondisi objek yang diamati. Kegagalan deteksi pada objek dengan orientasi tertentu menunjukkan bahwa keberhasilan deteksi tidak hanya bergantung pada kemampuan Raspberry Pi 4B dalam menjalankan YOLOv8, tetapi juga dipengaruhi oleh posisi box saat pengambilan gambar. Selain itu, faktor seperti pencahayaan, posisi kamera, dan kualitas citra juga dapat memengaruhi hasil deteksi dan klasifikasi. Oleh karena itu, penerapan sistem inspeksi visual pada lingkungan industri perlu memperhatikan kondisi pengambilan citra agar hasil deteksi tetap konsisten dan akurat.

KESIMPULAN

Penelitian ini berhasil mengimplementasikan sistem inspeksi visual berbasis YOLOv8 pada Raspberry Pi 4B yang mampu mendeteksi dan mengklasifikasikan kualitas hasil *stamping* ke dalam kategori *Good* dan *Not Good* secara *real-time* pada prototipe mesin *stamping* otomatis, dengan nilai *confidence* rata-rata 0,7097 untuk kategori *Good* dan 0,7398 untuk kategori *Not Good*. Pengujian kinerja Raspberry Pi 4B menunjukkan bahwa perangkat mampu menjalankan inferensi YOLOv8 secara kontinu dengan rata-rata *latency* 372,18 ms, kecepatan pemrosesan 2,69 FPS, penggunaan CPU rata-rata 42,53%, penggunaan RAM rata-rata 23,08%, dan suhu prosesor rata-rata 62,27°C seluruhnya berada dalam batas operasional yang stabil tanpa degradasi performa selama 907 *frame* pengujian. Sistem komunikasi data antara Raspberry Pi 4B, ESP32, dan Firebase *Realtime* Database menggunakan protokol HTTP GET dan REST API berhasil mengirimkan seluruh data hasil klasifikasi dengan rata-rata latensi pengiriman ke ESP32 sebesar 294,46 ms dan ke Firebase sebesar 1.456,60 ms, membuktikan bahwa integrasi teknologi *edge computing* berbasis AI dan IoT dapat menjadi solusi efektif dan berbiaya rendah untuk sistem inspeksi visual secara *real-time*. Meskipun demikian, penelitian ini masih terbatas pada pengujian menggunakan prototipe mesin *stamping* di lingkungan laboratorium. Oleh karena itu, penelitian selanjutnya dapat diarahkan pada implementasi sistem di lingkungan industri yang sebenarnya dengan kondisi operasional yang lebih beragam untuk mengevaluasi kinerja sistem secara lebih menyeluruh.

REFERENSI

- Ahmed, Abdulatif, Ali Aboluhom, and Ismet Kandilli. 2024. "Face Recognition Using Deep Learning on Raspberry Pi." (September):3020–30.
- Azab, Eman, Mohamed Ehab, and Lamia Shihata. 2026. "Optimizing Computer Vision for Edge Deployment in Industry 4.0: A Framework and Experimental Evaluation." 1–29.
- Azhar, Muhammad Fathor, and Lela Nurpulaela. 2024. "IMPLEMENTASI PENGGUNAAN ESP32 SEBAGAI IOT PADA PROJECT SMART CHARGER DI PT. PASIFIK SATELIT NUSANTARA BEKASI." 8(4):7248–53.
- Hayati, Nurhaliza Juliyani, Dayan Singasatia, and Muhamad Rafi Muttaqin. 2023. "KOMPUTA : Jurnal Ilmiah Komputer Dan Informatika OBJECT TRACKING MENGGUNAKAN ALGORITMA YOU ONLY LOOK ONCE (YOLO) v8 UNTUK MENGHITUNG KENDARAAN KOMPUTA : Jurnal Ilmiah Komputer Dan Informatika." 12(2):91–99.
- Isnaen, Muhammad Kuku, and Arnisa Stefanie. 2023. "MATA DENGAN ARSITEKTUR YOLOv8 MENGGUNAKAN OFTALMOSKOP." *Jurnal Mahasiswa Teknik Informatika* 7(3):1885–89.
- Khalid, Wajahat, Mohsin Jamil, Ashraf Ali Khan, and Qasim Awais. 2024. "Open-Source Internet of Things-Based Supervisory Control and Data Acquisition System for Photovoltaic Monitoring and Control Using HTTP and TCP/IP Protocols."
- Kim, Min-gyu. 2026. "Scan Path Optimization and YOLO-Based Detection for Defect Inspection of Curved and Glossy Surfaces."
- Luo, Qiuming, and Kunzhong Wu. 2025. "A RISC-V Based Image Acquisition System Using CIS for Industrial Print Quality Inspection." *Journal of Engineering Research* (October). doi:10.1016/j.jer.2025.10.002.
- Mahendra, Agum, and Slamet Winardi. 2023. "Perancangan Realtime Database Firebase Untuk IoT Dan Unity Menggunakan Metode SDLC." XIV(2):72–82.
- Margapuri, Venkat, and Mitchell Neilsen. 2021. "PiBase : An IoT-Based Security System Using Google Firebase and Raspberry Pi."
- Nizam, Muhammad, Haris Yuana, Fakultas Teknologi Informasi, Universitas Islam, Balitar Blitar, and Magnetic Door

- Switch. 2022. "MIKROKONTROLER ESP 32 SEBAGAI ALAT MONITORING PINTU BERBASIS WEB." 6(2):767–72.
- Noon, Seroosh Karim, Ali Hassan Noor, Abdul Mannan, Miqdam Arshad, Turab Haider, and Muhammad Abdullah. 2025. "Real-Time Vehicle Sticker Recognition for Smart Gate Control with YOLOv8 and Raspberry Pi 4." 1–14.
- Nugraha, Rahmawan Pramananda, Rekyan Regasari, and Mardi Putri. 2026. "Sistem Deteksi Pelanggaran Penggunaan Alat Pelindung Diri Berbasis Raspberry Pi 4 Dengan Multi-Kamera Dan Notifikasi Real-Time Menggunakan Algoritma Yolov8n." 10(2):1–8.
- Okano, Marcelo T., William Aparecido, Celestino Lopes, and Sergio Miele Ruggero. 2025. "Edge AI for Industrial Visual Inspection : YOLOv8-Based Visual Conformity Detection Using Raspberry Pi."
- Setyanto, Arief, Theopilus Bayu Sasongko, and I. N. K. E. E. Kim. 2025. "Knowledge Distillation in Object Detection for Resource-Constrained Edge Computing." *IEEE Access* 13(January):18200–214. doi:10.1109/ACCESS.2025.3534020.
- Singh, Raghubir, and Sukhpal Singh. 2023. "Internet of Things and Cyber-Physical Systems Edge AI : A Survey." *Internet of Things and Cyber-Physical Systems* 3(February):71–92. doi:10.1016/j.iotcps.2023.02.004.
- Sohan, Mupparaju, Thotakura Sairam, and Corresponding Author. 2024. "A Review on YOLOv8 and Its Advancements."
- Sugiyono. 2021. *Metode Penelitian Kuantitatif, Kualitatif, Dan R&D*. Cetakan Ke. Bandung: Alfabeta.
- Sundaram, Sarvesh, and Abe Zeid. 2023. "Artificial Intelligence-Based Smart Quality Inspection for Manufacturing."