RANCANG BANGUN PROTOTIPE SISTEM PERINGATAN DINI PENYAKIT UDANG VANAME (*LITOPENAEUS VANNAMEI*) BERBASIS *CONVOLUTIONAL NEURAL NETWORK*

LAPORAN TUGAS AKHIR

Diajukan Sebagai Syarat Menyelesaikan Pendidikan Pada Program Studi Teknik Listrik Industri Sekolah Vokasi Universitas Diponegoro

Disusun oleh:

Zidni Febrilian Hidayatulloh

40040622650056

PROGRAM STUDI SARJANA TERAPAN TEKNIK LISTRIK INDUSTRI

DEPARTEMEN TEKNOLOGI INDUSTRI

SEKOLAH VOKASI

UNIVERSITAS DIPONEGORO SEMARANG

2026

HALAMAN PERSETUJUAN LAPORAN TUGAS AKHIR

RANCANG BANGUN SISTEM PERINGATAN DINI PENYAKIT UDANG VANAME (*LITOPENAEUS VANNAMEI*) BERBASIS *CONVOLUTIONAL NEURAL NETWORK*

Diajukan oleh	:	Zidni Febrilian Hidayatulloh
NIM	:	40040622650056

TELAH DISUSUN DAN DITERIMA OLEH

Dosen Pembimbing

Dista Yoel Tadeus, S.T., M.T. Tanggal
NIP. 198812282015041002

Mengetahui
Ketua Program Studi Sarjana Terapan Teknik Listrik Industri
Departemen Teknik Industri Sekolah Vokasi
Universitas Diponegoro

Arkhan Subari, S.T., M.Kom Tanggal
NIP. 197710012001121002

ii

HALAMAN PENGESAHAN TUGAS AKHIR

RANCANG BANGUN SISTEM PERINGATAN DINI PENYAKIT UDANG VANAME (*LITOPENAEUS VANNAMEI*) BERBASIS *CONVOLUTIONAL NEURAL NETWORK*

Diajukan oleh : Zidni Febrilian Hidayatulloh

Telah dipertahankan di depan Dewan Penguji pada

Hari : Selasa

Tanggal : 30 Juni 2026

Penguji I	Penguji II	Penguji III
Yuniarto, S.T., M.T.	**Arkhan Subari, S.T.,M.Kom**	**Dista Yoel Tadeus, S.T., M.T.**
NIP.197106151998021001	**NIP.197710012001121002**	**NIP.198812282015041002**

Mengetahui,

Ketua Program Studi Teknik Listrik Industri

Departemen Teknologi Industri Sekolah Vokasi

Universitas Diponegoro

Arkhan Subari, S.T.,M.Kom

NIP.197710012001121002

iii

LEMBAR PERNYATAAN KEASLIAN

Saya bertanda tangan di bawah ini :

Nama	:	Zidni Febrilian Hidayatulloh
NIM	:	40040622650056
Program Studi	:	Sarjana Terapan Teknik Listrik Industri Departemen Teknologi Industri Sekolah Vokasi Universitas Diponegoro
Judul Tugas Akhir	:	Rancang Bangun Sistem Peringatan Dini Penyakit Udang Vaname (*Litopenaeus Vannamei*) Berbasis *Convolutional Neural Network*

Dengan ini saya menyatakan bahwa judul tugas akhir ini belum pernah diajukan sebelumnya untuk mendapatkan gelar keahlian di sebuah perguruan tinggi. Sejauh pengetahuan saya, tidak ada karya atau pendapat yang pernah ditulis dan diterbitkan oleh orang lain kecuali yang saya rujuk secara tertulis dalam naskah ini dan tercantum dalam daftar pustaka.

Apabila dikemudian hari terbukti plagiat dalam tugas akhir ini, maka saya bersedia mendapat sanksi sesuai peraturan Mendiknas RI. No.17 Tahun 2010 dan Undang-Undang yang berlaku.

Semarang, 30 Juni 2026

Yang membuat Pernyataan

Zidni Febrilian Hidayatulloh

NIM. 40040622650056

iv

HALAMAN PERSEMBAHAN

Dengan penuh rasa Syukur dan ketulusan hati, penulis mempersembahkan karya tugas akhir ini kepada pihak-pihak yang telah memberi dukungan, pengetahuan, serta pengalaman berharga selama proses penyusunan. Karya ini akan penulis persembahkan untuk :

1. Allah SWT, Segala puji dan syukur hamba panjatkan kepada-Mu, yang telah memberikan kehidupan, kesehatan, kekuatan, dan kesempatan untuk menyelesaikan karya ini. Tanpa Rahmat serta Ridho-Mu hamba tak akan mampu melangkah sejauh ini. Besar harapan hamba supaya ilmu yang diperoleh mampu bermanfaat bagi seluruh elemen kehidupan.
2. Ibu, Bapak, dan Kakak yang senantiasa memberikan doa, kasih sayang, dukungan, serta motivasi dalam setiap langkah penulis. Terima kasih atas segala pengorbanan dan kepercayaan yang telah diberikan hingga penulis mampu meraih berbagai pencapaian dan menyelesaikan perjalanan akademik ini dengan baik.
3. Prof. Dr. Ir. Budiyono, M.Si. selaku Dekan Sekolah Vokasi Universitas Diponegoro.
4. Bapak Arkhan Subari, S.T., M.Kom. selaku Ketua Program Studi Teknik Listrik Industri Sekolah Vokasi Universitas Diponegoro.
5. Bapak Dista Yoel Tadeus, S.T., M.T., selaku Dosen Pembimbing yang telah membimbing dengan baik dan mengarahkan penulis dalam penyusunan tugas akhir ini.
6. Bapak Alfabetian Harjuno Condro Haditomo, S.Pi., M.Si., selaku Dosen Pembimbing PKM yang telah memberikan bimbingan, arahan, dukungan, serta kesempatan berharga sehingga penulis dapat memperoleh pengalaman dan pembelajaran yang bermakna.
7. Bapak Susatyo Nugroho Widyo Pramono, S.T., M.M., selaku Ketua Tim *Task Force* Universitas Diponegoro yang telah memberikan dukungan, motivasi, dan kepercayaan kepada penulis selama proses pelaksanaan kegiatan PKM.
8. Andini Setya Pangestuti, Achmad Chairil Istafat, dan Didik Sulistiyono selaku rekan satu tim yang telah berjuang bersama, memberikan semangat, kerja

sama, serta menjadi bagian dari perjalanan berharga penulis dalam meraih prestasi pada ajang PIMNAS ke-38.

9. Cheravisha Nandya Ishira selaku pasangan dan sahabat yang senantiasa menemani, mendukung, serta menjadi tempat bertukar cerita dan pemikiran bagi penulis.
10. Salma Almira, Balqis Safira, dan Ade Fawwash selaku sahabat yang telah memberikan kesempatan untuk berproses, belajar, dan berkembang bersama dalam perjalanan berorganisasi. Terima kasih atas kebersamaan, dukungan, serta berbagai pengalaman berharga yang telah menjadi bagian dari perjalanan penulis.
11. Terakhir, kepada diri penulis sendiri, terima kasih karena telah bertahan sejauh ini. Terima kasih telah tetap berjuang di tengah berbagai hambatan dan keraguan yang hadir dalam perjalanan. Setiap usaha, lelah, dan air mata yang telah dilalui menjadi saksi dari proses panjang yang mengantarkan penulis pada pencapaian yang membanggakan. Semoga perjalanan ini menjadi langkah awal untuk terus berkembang dan meraih impian yang lebih besar.

ABSTRAK

Infectious Myonecrosis Virus (IMNV), *White Spot Syndrome Virus* (WSSV), dan *Taura Syndrome Virus* (TSV) menyebabkan kerugian pada budidaya udang di dunia. Metode PCR, berbasis laboratorium memiliki hasil akurat namun belum diterapkan secara rutin oleh sebagian pembudidaya. Prototipe ini berbasis *computer vision* dengan model *Convolutional Neural Networks* (CNN) menganalisis citra udang secara *real time* sehingga dapat mendeteksi dini penyakit udang secara cepat dan akurat, operasional mudah, serta dapat diakses dari mana pun. Prototipe didesain dengan Solidworks dan dilatih menggunakan Roboflow. Sebanyak 2.265 *trainset database* digunakan untuk melatih algoritma CNN, sebanyak 678 citra digunakan untuk memvalidasi algoritma guna menghasilkan akurasi deteksi citra sebesar 66%. Citra yang terdeteksi dibagi menjadi 159 *layer*. Sistem memvalidasi udang sehat sebesar 98%, udang terinfeksi IMNV (99%), TSV(45%), dan WSSV (63%). Total 130 udang vaname diujikan menggunakan sistem terintegrasi aplikasi *mobile* menghasilkan 127 udang sehat dan 3 udang sakit. Hasil tersebut ditampilkan pada aplikasi *mobile* S-DiDeS yang terinstal pada *smartphone*. Prototipe ini mampu mendeteksi dini penyakit, mengoptimalkan mitigasi, meminimalisir resiko kematian massal udang dan mendukung program *Blue Economy.*

Kata Kunci: *convolutinal neural network*, deteksi dini, penyakit udang

ABSTRACT

Infectious Myonecrosis Virus (IMNV), White Spot Syndrome Virus (WSSV), and Taura Syndrome Virus (TSV) cause major economic losses in global shrimp aquaculture. PCR method—despite its accuracy—is laboratory-dependent and not performed routinely by many farmers. S-DiDeS developed as an early-warning disease detection system using computer vision and a Convolutional Neural Network (CNN). It was designed in Solidworks, and trained using the Roboflow with 2,265 images and validated with 678 images, resulting in an detection accuracy 66%. Each image through 159 analytical layers and achieved high classification accuracy for healthy shrimp (98%) and IMNV infections (99%), with moderate accuracy for TSV (45%) and WSSV (63%). Total 130 shrimps tested with the integrated mobile application, which classified 127 samples as healthy and 3 as diseased were displayed through the S-DiDeS smartphone interface. S-DiDeS demonstrated the potential for early detection of major viral diseases in shrimp, enabling timelier mitigation, reducing the risk of mass mortality, and supporting the Blue Economy.

***Keywords**: Convolutional Neural Network, early warning system, shrimp diseases*

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "Rancang Bangun Sistem Peringatan Dini Penyakit Udang Vaname (Litopenaeus Vannamei) Berbasis *Convolutional Neural Network*" dengan baik.

Dalam proses penyusunan tugas akhir ini, penulis telah melalui berbagai tahapan, proses pembelajaran, serta tantangan yang menjadi bagian dari perjalanan akademik penulis. Penyelesaian tugas akhir ini tidak terlepas dari dukungan, bimbingan, dan bantuan dari berbagai pihak yang telah memberikan kontribusi berharga bagi penulis. Penulis menyampaikan terima kasih kepada seluruh pihak yang telah memberikan dukungan, khususnya kepada dosen pembimbing, keluarga, sahabat, rekan seperjuangan, serta pihak-pihak lain yang telah memberikan doa, motivasi, ilmu, dan pengalaman selama proses penyusunan tugas akhir ini.

Penulis menyadari bahwa dalam penyusunan tugas akhir ini masih terdapat kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan untuk penyempurnaan tugas akhir ini. Penulis berharap tugas akhir ini dapat memberikan manfaat serta menjadi kontribusi dalam pengembangan ilmu pengetahuan dan teknologi. Akhir kata, penulis mengucapkan terima kasih kepada seluruh pihak yang telah berperan dalam perjalanan ini. Semoga segala kebaikan dan dukungan yang diberikan mendapatkan balasan yang terbaik.

Semarang, 22 Juni 2026

Yang membuat Pernyataan

Zidni Febrilian Hidayatulloh

NIM. 40040622650056

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Indonesia merupakan salah satu produsen udang vaname terbesar di dunia. Sektor tersebut menempati peringkat keempat eksportir udang global dengan produksi 3,2 juta ton dengan nilai ekspor 2,16 miliar USD, yang diprediksi terus meningkat [1]. Pesatnya budidaya udang memicu munculnya patogen penyebab penyakit yang menimbulkan kerugian ekonomi dan lingkungan [2]. Sebagian besar penyakit disebabkan infeksi virus seperti *Infectious Myonecrosis Virus* (IMNV)**,** *Taura Syndrome Virus* **(**TSV)**,** dan *White Spot Syndrome Virus* (WSSV), yang dapat menyebabkan kematian hingga 100% dalam 3–4 hari [3]. Kerugian ekonomi akibat virus tersebut mencapai 1 miliar USD untuk WSSV [4], 1,5–3 miliar USD untuk TSV [5], dan 159,3 juta/tahun untuk IMNV [6]. Di Kendal, kasus IMNV tahun 2022 menyebabkan kerugian hingga Rp 17,3 miliar [7]. Kematian massal akibat virus dan buruknya manajemen tambak menghasilkan limbah organik yang mencemari lingkungan [8].

Metode deteksi penyakit udang yang umum digunakan saat ini meliputi PCR dan pengamatan gejala klinis manual [9]. Pengamatan manual rentan terhadap kesalahan manusia [10], sedangkan PCR memiliki keterbatasan frekuensi pelaksanaan hanya 1–2 bulan sekali serta memerlukan proses laboratorium yang rumit [11]. Beberapa metode deteksi dini telah dikembangkan untuk mengatasi kendala tersebut, meliputi sistem pakar berbasis windows [12], aplikasi android [13], dan *website* [14].

Sistem pakar masih terbatas karena cakupan penyakit sempit, akurasi bergantung pada *user*, serta pengembangannya membutuhkan waktu dan biaya besar [15]. Oleh karena itu, diperlukan sistem deteksi dini yang cepat, akurat, dan efisien dengan kemampuan pemantauan kondisi udang secara *real-time* [16]. Hal ini menjadi langkah krusial untuk mencegah kerugian ekonomi lebih lanjut serta meminimalisasi dampak pencemaran lingkungan [17]. *Convolutional neural network* (CNN) dapat mengenali visual udang secara otomatis [18], sehingga langkah mitigasi dapat segera dilakukan guna mencegah kematian massal dan

pencemaran lingkungan. Melalui kemampuan mempelajari representasi kompleks, CNN mampu mengidentifikasi penyakit udang secara visual dengan akurasi hingga 90,02% [19]. Teknologi dapat diintegrasikan dengan aplikasi seluler untuk pemantauan secara *real-time* [20].

Berdasarkan permasalahan di atas, maka diperoleh inovasi yang berjudul "Rancang Bangun Sistem Peringatan Dini Penyakit Udang Vaname (*Litopenaeus Vannamei*) Berbasis *Convolutional Neural Network*" dalam mewujudkan budidaya udang ramah lingkungan demi mencegah bencana alam dan sosial ekonomi akibat kematian massal. Keterbaruan prototipe ini melalui inovasi alat analisis gejala klinis menggunakan CNN dengan sistem *deep learning* berbasis AI yang dapat klasifikasi gambar dan identifikasi objek [21]. Sistem terhubung dengan *smartphone* sehingga dapat diakses kapan saja. Adanya prototipe diharapkan mampu melakukan deteksi dini, mengurangi risiko penularan dan kerugian akibat penyakit yang diinformasikan melalui aplikasi pada *smartphone* pembudidaya.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana merancang sistem deteksi dini penyakit udang vaname berbasis *Convolutional Neural Network* (CNN) yang mampu mengenali gejala klinis secara otomatis dari citra udang?
2. Bagaimana mengintegrasikan sistem deteksi berbasis CNN dengan aplikasi s*martphone* agar hasil identifikasi dapat diakses secara cepat dan mudah oleh pembudidaya?
3. Seberapa baik tingkat akurasi dan keandalan sistem dalam mendeteksi penyakit utama seperti IMNV, TSV, dan WSSV dibandingkan metode manual?
4. Bagaimana sistem yang dikembangkan dapat membantu mengurangi risiko kematian massal, kerugian ekonomi, dan dampak lingkungan akibat penyakit udang?

1.3 Tujuan Tugas Akhir

Tujuan Penyusunan Tugas Akhir antara lain

1. Merancang model dan sistem kerja alat pendeteksi penyakit udang berbasis CNN.
2. Mengembangkan prototipe sebagai solusi deteksi dini penyakit udang, pengurangan risiko penularan dan kerugian akibat penyakit baik secara ekonomi maupun ekologis.
3. Mengevaluasi tingkat akurasi dan keandalan sistem berbasis *convolutional neural network* (CNN) dalam mendeteksi penyakit udang berdasarkan citra.
4. Mengimplementasikan sistem dalam *platform* berbasis *smartphone* untuk mendukung pemantauan kondisi udang secara *real time*.

1.4 Manfaat Tugas Akhir

Manfaat penyusunan dan pembuatan tugas akhir ini yaitu :

1.4.1 Bagi Penulis

1. Sebagai penerapan ilmu di bidang *artificial intelligence* khususnya *convolutional neural network* dalam sistem deteksi penyakit udang.
2. Menambah pemahaman dalam perancangan dan pengembangan sistem berbasis citra serta integrasinya dengan aplikasi monitoring.
3. Mengembangkan kemampuan dalam merancang prototipe sistem deteksi dini berbasis teknologi *modern.*

1.4.2 Bagi Masyarakat

Dengan adanya sistem peringatan dini penyakit udang berbasis CNN, diharapkan dapat membantu pembudidaya dalam mendeteksi kondisi kesehatan udang secara dini dan otomatis, sehingga dapat melakukan langkah antisipasi yang tepat guna mengurangi risiko kerugian baik secara ekonomis maupun ekologis.

1.4.3 Bagi Lembaga

Sebagai referensi dan bahan pengembangan dalam penelitian selanjutnya terkait sistem deteksi penyakit berbasis *deep learning*, khususnya pada sektor perikanan budidaya, serta sebagai kontribusi dalam pengembangan teknologi tepat guna yang mendukung budidaya udang yang berkelanjutan.

1.5 Batasan Masalah

Agar penelitian yang dilakukan lebih terarah dan memiliki ruang lingkup yang jelas, maka batasan masalah dalam tugas akhir ini ditetapkan sebagai berikut:

1. Sistem yang dirancang difokuskan pada deteksi penyakit udang vaname (*Litopenaeus vannamei*) berdasarkan citra visual menggunakan metode *convolutional neural network* (CNN).
2. Jenis penyakit yang dideteksi dibatasi pada penyakit utama yang memiliki gejala visual, seperti IMNV, TSV, dan WSSV.
3. Data yang digunakan dalam proses pelatihan dan pengujian model berasal dari dataset citra udang yang telah dikumpulkan dan tidak mencakup seluruh variasi kondisi di lapangan.
4. Sistem yang dikembangkan berupa prototipe, sehingga implementasi masih dalam skala terbatas dan belum diterapkan secara penuh pada lingkungan tambak industri.
5. Sistem hanya melakukan klasifikasi kondisi udang seperti sehat atau terindikasi penyakit dan tidak mencakup analisis lanjutan seperti tingkat keparahan penyakit atau rekomendasi pengobatan.
6. Pemantauan dilakukan berbasis aplikasi *smartphone*, dengan asumsi ketersediaan jaringan komunikasi yang mendukung proses pengiriman dan penerimaan data.
7. Sistem tidak membahas metode deteksi laboratorium seperti PCR secara mendalam, melainkan hanya sebagai pembanding dalam analisis.
8. Aspek kendali lingkungan tambak seperti kualitas air seperti pH, suhu, DO, tidak menjadi fokus utama dalam penelitian.

1.6 Sistematika Penyusunan Tugas Akhir

Sistematika penulisan dalam penyusunan tugas akhir, disusun dengan memberikan gambaran secara umum mengenai isi setiap bab, sehingga memudahkan pembaca dalam memahami alur pembahasan. Adapun sistematika penulisan dalam laporan ini adalah sebagai berikut:

HALAMAN JUDUL

HALAMAN PENGESAHAN

Bab ini berisi kesimpulan dari hasil Tugas Akhir yang telah dilakukan serta saran untuk pengembangan sistem di masa yang akan datang.

DAFTAR PUSTAKA

LAMPIRAN

BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Berikut beberapa referensi yang menjadi tinjauan pustaka pada laporan tugas akhir yang berjudul "Rancang Bangun Sistem Peringatan Dini Penyakit Udang Vaname (*Litopenaeus Vannamei*) Berbasis *Convolutional Neural* Network"

Tabel 2. 1 Tabel Penelitian Terdahulu

Judul	Penulis	Hasil
Penerapan Metode *Forward Chaining* untuk Diagnosa Dini Penyakit Udang Vannamei pada Budidaya Udang Berbasis Android	Felicia Gunadi, Rinabi Tanamal (2021)	Hasil *User Acceptance Test* (UAT) berbasis Skala *Likert* menunjukkan bahwa aplikasi mudah digunakan (98%), mudah dipahami (97%), informasi fungsi aplikasi sesuai (98%), tampilan responsif (95%), dan 100%.
Rancangan Sistem Pakar Diagnosa Penyakit pada Udang Menggunakan Metode *Forward Chaining* Berbasis Web	Yohanes Valendry Cawa, Koko Handoko (2023)	Pengujian dilakukan melalui 3 skenario uji pakar dan ketiganya menghasilkan diagnosa yang tepat sesuai gejala yang diinputkan, membuktikan sistem mampu mengidentifikasi penyakit secara akurat. Sistem pakar memberikan keunggulan akses secara *real-time* melalui perangkat yang terhubung internet
Sistem Pakar Diagnosa Penyakit Udang Vaname pada Dinas Kelautan dan Perikanan Provinsi Banten	Ma'sum, Wahidin (2020)	Sistem memiliki beberapa fitur utama yaitu *form login* untuk autentikasi pengguna dan admin, master gejala untuk mengelola data gejala penyakit, master penyakit untuk mengelola data jenis penyakit beserta

Judul	Penulis	Hasil
		pengendaliannya, *form* konsultasi sebagai tempat pengguna menginput gejala yang dialami udang, info penyakit yang menampilkan informasi yang lengkap pada tiap penyakit, dan laporan hasil konsultasi yang dapat diunduh sebagai dokumen resmi.
Deteksi Objek Menggunakan Algoritma CNN Untuk Udang Vannnamei Pada Ibap Banjar Kemuning	Wahyu Ismaya Cipta Mahardhika (2025)	Model CNN tiga lapisan yang dibangun menghasilkan akurasi pelatihan 84% dan pengujian 81,1%, dengan presisi 100% namun *recall* hanya 65% akibat masih tingginya *false negative*. Dibandingkan metode manual, sistem ini terbukti lebih unggul dengan akurasi deteksi 90% berbanding 88%, meskipun pengembangan lebih lanjut masih diperlukan.

2.2 Dasar Teori

2.2.1 Gejala Klinis Udang Vaname

Udang vaname merupakan salah satu spesies udang yang paling banyak dibudidayakan di Indonesia karena memiliki nilai ekonomis tinggi. Selain itu, udang vaname mudah dibudidayakan dan waktu pemeliharaannya relatif singkat sehingga membuat petambak tertarik untuk membudidayakannya [22]. Ciri fisik udang vaname memiliki warna putih transparan yang dapat dilihat pada gambar 2.1

Gambar 2. 1 Udang Vaname

Salah satu kendala yang dihadapi dalam budidaya udang adalah munculnya wabah penyakit yang dapat menyebar dengan luas diantara populasi udang dalam satu area budidaya. Wabah penyakit tersebut berpotensi menyebabkan kematian massal pada udang sehingga menimbulkan kerugian yang signifikan bagi para pembudidaya [23]. Beberapa virus yang umum menyerang udang vaname antara lain *Infectious Myonecrosis Virus* (IMNV)*, White Spot Syndrome Virus* (WSSV), dan *Taura Syndrome Virus* (TSV). Udang vaname yang terserang penyakit dapat dikenali melalui beberapa gejala klinis, baik dari perubahan ciri fisik maupun perilaku yang ditunjukkan oleh udang tersebut.

Gambar 2. 2 Infeksi Penyakit IMNV

Seperti pada gambar 2.2, IMNV secara fisik ditandai dengan munculnya warna putih hingga kemerahan pada bagian otot yang mengalami nekrosis [24]. Selain perubahan fisik tersebut, udang yang terinfeksi penyakit IMNV menunjukan gejala klinis berupa perilaku berenang yang cenderung pasif , serta penurunan nafsu makan [25]. Begitu pula pada udang yang terkena WSSV diindikasikan degan adanya bintik putih pada karapas udang [26]. Selain itu, udang yang terserang WSSV menunjukan perilaku berenang di permukaan air, nafsu makan menurun,serta bergerombol di tepi perairan. Ciri fisik udang yang terinfeksi WSSV dapat dilihat pada Gambar 2.3.

Gambar 2. 3 Infeksi Penyakit WSSV

Sementara itu, udang yang terinfeksi TSV menunjukan perubahan fisik berupa karapas berwarna kemerahan [27], cangkang lunak serta usus yang nampak kosong. Selain itu udang yang terinfeksi TSV akan menunjukkan perilaku pasif, nafsu makan menurun, dan berkumpul di tepi kolam. Ciri fisik udang yang terinfeksi TSV dapat dilihat pada Gambar 2.4.

Gambar 2. 4 Infeksi Penyakit TSV

2.2.2 Sistem Pemantauan Perigatan Dini Berbasis Seluler

Sistem peringatan dini merupakan mekanisme yang diterapkan untuk mengumpulkan dan menginterpretasikan data kondisi kesehatan populasi hewan budidaya, dengan tujuan mendeteksi dan mengendalikan penyakit sejak dini sebelum menyebar lebih luas [28]. Impelementasi sistem bekerja dengan memantau parameter kondisi lingkungan budidaya secara berkelanjutan, apabila nilai yang terbaca melampaui ambang batas yang telah ditetapkan [29]. Sistem tidak hanya mendeteksi kondisi yang sedang terjadi, tetapi dapat dikembangkan dalam memperkirakan tingkat ancaman di masa mendatang sehingga pengelola dapat mengambil langkah pencegahan sebelum kerugian benar-benar terjadi [30].

Pengembangan sistem peringatan dini merupakan kebutuhan mendesak, pendekatan diagnosis konvensional yang dilakukan setelah wabah terjadi berisiko tinggi terhadap keterlambatan penanganan dan kesalahan diagnosis, sehingga diperlukan alat bantu keputusan yang mampu menilai dan memperingatk

an risiko penyakit secara proaktif [31]. Dalam bidang budidaya udang penerapan sistem peringatan dini berbasis *real time* menggunakan prototipe yang mampu mengindentifikasi penyakit berdasarkan gejala klinis yang berupa citra digital merupakan sebuah inovasi baru [32], efektivitas sistem deteksi dini berbasis kecerdasan buatan dibuktikan secara empiris melalui penerapan model klasifikasi citra pada perangkat tepi kolam, di mana sistem terbukti mampu mengidentifikasi kondisi penyakit udang rata-rata enam jam lebih awal dibandingkan pemeriksaan manual oleh teknisi [33].

Sistem pemantauan peringatan dini berbasis seluler mengadopsi pola *architecture* yang memisahkan produsen data dari aplikasi seluler, sehingga memungkinkan kedua komponen untuk beroperasi dan berkembang secara independen [34]. Alur kerja sistem mengikuti empat tahapan utama, dimulai dari pengambilan data oleh sensor atau kamera pada perangkat tepi, pemrosesan awal menggunakan model klasifikasi untuk mengekstrak informasi, pembungkusan hasil ke dalam format JSON, transmisi melalui protokol *fast api,* dan terakhir penampilan hasil pada antarmuka pengguna [35], [36]. JSON dipilih sebagai format pertukaran data karena sifatnya yang ringan, mudah di-*parse*, dan didukung secara luas oleh berbagai platform pemrograman [37]. Pendekatan secara *edge computing* yang diterapkan pada tahap akuisisi dan pemrosesan awal data terbukti mampu mengurangi ketergantungan pada konektivitas jaringan yang stabil serta mempercepat waktu respon sistem, yang merupakan faktor kritis dalam konteks pemantauan kesehatan hewan ternak yang membutuhkan tindakan cepat [36], [38].

Waktu tunda satu arah atau *one way delay* (OWD) didefinisikan sebagai selisih waktu antara pengiriman paket dari *front end* dan penerimaan paket di *back end*, yang secara matematis dinyatakan sebagai berikut :

$$t_{OWD} = t_{rx} - t_{tx} \tag{2.1}$$

Dimana t_{rx} adalah waktu saat paket dikirimkan dari perangkat dan t_{tx} adalah waktu saat paket diterima oleh perangkat tujuan. Dalam konteks sistem peringatan dini, nilai OWD idealnya < 100 ms untuk menjadi prasyarat agar notifikasi peringatan dapat sampai ke pengguna tanpa keterlambatan yang berarti. Waktu tempuh atau *round trip time* (RTT) merupakan waktu yang di

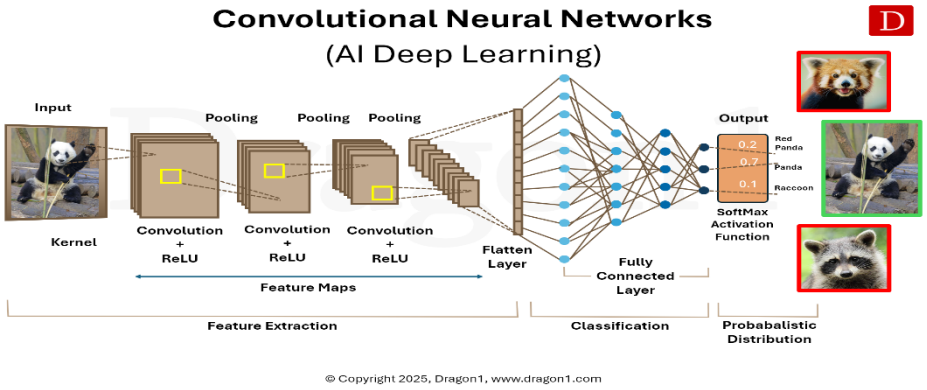
perlukan sejak pengiriman *request* hingga diterimanya *respons* balik, seperti ditunjukkan sebagai berikut :

$$t_{RTT} = t_{ack} - t_{tx} \quad (2.2)$$

Dimana t_{ack} tadalah waktu saat *response* atau *acknowledgment* diterima kembali oleh pengirim [39]. Nilai RTT mencerminkan latensi *end-to-end* yang mencakup waktu pemrosesan di kedua sisi, sehingga menjadi indikator yang lebih komprehensif untuk mengevaluasi responsivitas sistem secara keseluruhan. Dalam implementasinya, efektivitas sistem peringatan dini berbasis pengolahan citra digital dipengaruhi oleh kondisi lingkungan tambak. Kondisi umumnya pada tambak memiliki kecerahan bervariasi dengan awal masa budidaya kecerahan mencapai 100 cm kemudian terus menurun hingga 30 cm [40]. Variasi tingkat kecerahan tersebut berpengaruh terhadap tingkat akurasi sistem dalam melakukan proses identifikasi objek maupun deteksi penyakit secara *real-time*.

2.2.3 *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah jaringan saraf tiruan yang menggunakan terdiri dari banyak *layer*, di mana jaringan saraf mempertahankan struktur hierarkis dengan mempelajari representasi fitur internal dan menggeneralisasi fitur dalam masalah gambar umum seperti pengenalan objek dan masalah penglihatan komputer lainnya [18]. CNN merupakan salah satu arsitektur *deep learning* yang dirancang khusus untuk memproses data berupa citra digital [41]. Keunggulan utama CNN dibandingkan metode klasifikasi konvensional adalah kemampuannya dalam mengekstraksi fitur secara otomatis melalui operasi konvolusi bertingkat, sehingga tidak diperlukan proses ekstraksi fitur secara manual [42].



Gambar 2. 5 Mekanisme CNN

2.2.4 Sistem Pengolahan Citra Digital

Pengolahan citra digital merupakan tahapan penting dalam proses deteksi dalam sistem klasifikasi berbasis CNN. Teknik pengolahan citra sangat penting untuk meningkatkan kualitas citra, mereduksi *noise*, dan meningkatkan akurasi pengenalan pola [43]. Penelitian dalam [44], bawha pengolahan citra digital didefinisikan sebagai proses manipulasi dan analisis gambar yang telah didigitalisasi untuk meningkatkan kualitas atau mengekstraksi informasi yang relevan. Dalam implementasinya, beberapa operasi dasar pengolahan citra dilakukan secara berurutan. Pertama, operasi konversi ruang warna RGB ke *grayscale* diterapkan. Konversi citra RGB menjadi *grayscale* dapat membantu sistem memfokuskan analisis pada pola tekstur dan intensitas tanpa dipengaruhi oleh variasi warna [45].

Nilai intensitas dengan rentang 0–255 digunakan pada citra RGB maupun *grayscale*. Citra RGB mengombinasikan tiga saluran *red* (R), green (G), dan *blue* (B), sedangkan citra *grayscale* hanya memakai satu kanal yang dikenal sebagai derajat keabuan [46]. Nilai dari ketiga kanal warna tersebut pada sebuah citra digital dapat dinyatakan secara matematis melalui fungsi koordinat piksel. Fungsi $f(x, y, 1)$, $f(x, y, 2)$ dan $f(x, y, 3)$ secara berurutan merepresentasikan nilai warna *red*, *green*, dan *blue*. Sebagai contoh, representasi matriks dapat dituliskan sebagai berikut:

$$f(x,y,1)=\begin{pmatrix} f(0,0) & f(0,1) & f(0,2) & \dots & f(0,n-1) \\ f(1,0) & f(1,1) & f(1,2) & \dots & f(1,n-1) \\ \dots & \dots & \dots & \dots & \dots \\ f(m-1,0) & f(m-1,1) & f(m-1,2) & \dots & f(m-1,n-1) \end{pmatrix} \quad (2.3)$$

$$f(x,y,2)=\begin{pmatrix} f(0,0) & f(0,1) & f(0,2) & \dots & f(0,n-1) \\ f(1,0) & f(1,1) & f(1,2) & \dots & f(1,n-1) \\ \dots & \dots & \dots & \dots & \dots \\ f(m-1,0) & f(m-1,1) & f(m-1,2) & \dots & f(m-1,n-1) \end{pmatrix} \quad (2.4)$$

$$f(x,y,3)\begin{pmatrix} f(0,0) & f(0,1) & f(0,2) & \dots & f(0,n-1) \\ f(1,0) & f(1,1) & f(1,2) & \dots & f(1,n-1) \\ \dots & \dots & \dots & \dots & \dots \\ f(m-1,0) & f(m-1,1) & f(m-1,2) & \dots & f(m-1,n-1) \end{pmatrix} \quad (2.6)$$

Merujuk pada [47], simbol *x*, *y*, dan *z* digunakan untuk menyatakan representasi citra secara umum. Oleh karena itu, jika citra berwarna (RGB) diubah ke dalam bentuk skala *grayscale*, maka formulasinya dapat dinyatakan melalui susunan tiga fungsi di bawah ini:

$$f_n(x,y) = 0.299f(x,y,1) + 0.587f(x,y,2) + 0.114f(x,y,3) \quad (2.7)$$

Dengan:

x = 0,1,2,3,…, n-1,

y = 0,1,2,3,…, m-1,

$fn(x,y)$ = 0,1,2,3,…,255,

$f(x,y,1)$ = 0,1,2,3,…,255,

$f(x,y,2)$ = 0,1,2,3,…,255,

$f(x,y,3)$ = 0,1,2,3,…,255

Dimana:

n = jumlah piksel kolom pada array citra,

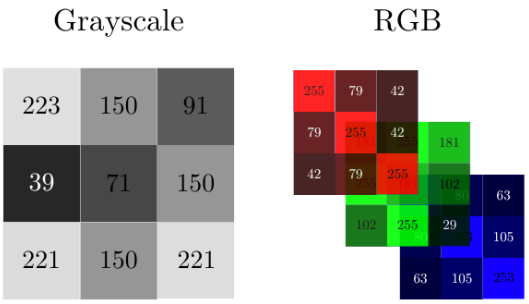
m = jumlah piksel baris pada array citra,

$f_n(x,y)$ = nilai intensitas keabuan pada titik x,y.

$f(x,y,1)$ = nilai intensitas *channel red* pada titik x,y.

$f(x,y,2)$ = nilai intensitas *channel green* pada titik x,y.

$f(x,y,3)$ = nilai intensitas *channel blue* pada titik x,y.



Gambar 2. 6 Teknik Konversi Warna

Setelah konversi warna dalam penelitian [44], langkah selanjutnya adalah *filtering*. Proses *filtering* dilakukan untuk mereduksi *noise* sekaligus mempertahankan informasi penting pada citra [48]. *Filtering* bekerja langsung pada nilai intensitas piksel dan piksel-piksel tetangganya, yang secara matematis merupakan operasi konvolusi antara citra masukan dengan sebuah *filter* [49]. Nilai intensitas piksel pada citra grayscale fn(x, y) dapat dinyatakan dalam bentuk matriks berukuran m × n sebagai berikut:

$$f_n(x,y) = \begin{pmatrix} f_n(0,0) & f_n(0,1) & \cdots & f_n(0,n-1) \\ f_n(1,0) & f_n(1,1) & \cdots & f_n(1,n-1) \\ \vdots & \vdots & \ddots & \vdots \\ f_n(m-1,0) & f_n(m-1,1) & \cdots & f_n(m-1,n-1) \end{pmatrix} \quad (2.8)$$

Kernel filter W yang digunakan pada operasi konvolusi dapat dinyatakan dalam bentuk matriks berukuran p × q sebagai berikut:

$$W = \begin{pmatrix} w(0,0) & w(0,1) & \cdots & w(0,q-1) \\ w(1,0) & w(1,1) & \cdots & w(1,q-1) \\ \vdots & \vdots & \ddots & \vdots \\ w(p-1,0) & w(p-1,1) & \cdots & w(p-1,q-1) \end{pmatrix} \quad (2.9)$$

Sebagai contoh, formulasi model matriks dengan kernel *filter* W yang dapat direpresentasikan dalam bentuk matriks berukuran p × q:

$$g(x,y) = \sum_{s=0}^{p-1} \sum_{t=0}^{q-1} w(s,t) \cdot f_n(x+s, y+t) \quad (2.10)$$

Dengan:

$x = 0,1,2,\ldots,n' - 1,$

$y = 0,1,2,\ldots,m' - 1,$

$s = 0,1,2,\ldots,p - 1,$

$t = 0, 1, 2, \ldots, q - 1,$

$g(x, y) = 0,1,2, \ldots ,255$

Dimana :

x = indeks kolom pada citra *output*

y = indeks baris pada citra *output*

s = indeks baris pada kernel *filter*

t = indeks kolom pada kernel *filter*

$fn(x, y)$ = nilai intensitas *grayscale* pada titik x,y

w(s, t) = nilai bobot kernel filter pada posisi s,t

g(x, y) = nilai intensitas citra hasil *filtering* pada titik x,y

p = jumlah baris kernel *filter*

q = jumlah kolom kernel *filter*

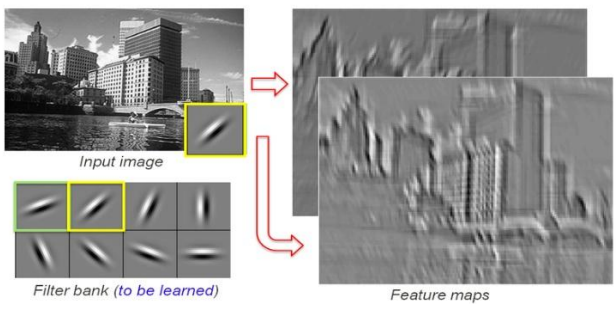
m' = jumlah baris pada citra *output* (m' = m - p + 1)

n' = jumlah kolom pada citra *output* (n' = n - q + 1).

Hasil operasi konvolusi pada persamaan (3) menghasilkan matriks output g(x, y) berukuran m′ × n′ yang disebut *feature map*, dan dapat dituliskan sebagai berikut:

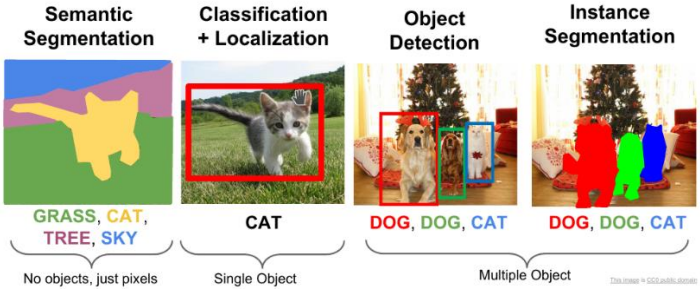
$$g(x,y) = \begin{pmatrix} g(0,0) & g(0,1) & \cdots & g(0,n'-1) \\ g(1,0) & g(1,1) & \cdots & g(1,n'-1) \\ \vdots & \vdots & \ddots & \vdots \\ g(m'-1,0) & g(m'-1,1) & \cdots & g(m'-1,n'-1) \end{pmatrix} \quad (2.10)$$

Pemilihan nilai bobot w(s,t) pada kernel W menentukan jenis dan karakteristik filter yang digunakan. Pada metode *filtering* tradisional seperti *average filter*, *Wiener filter*, dan *median filter*, nilai bobot kernel bersifat tetap dan tidak dapat disesuaikan selama proses *filtering* berlangsung [48]



Gambar 2. 7 Sistem Kerja dari *Filter*

Proses identifikasi objek memerlukan pemisahan antara objek dan latar belakang. Pada tahap segmentasi, metode *thresholding* digunakan dalam membedakan area objek dan latar belakang secara otomatis [50]. Teknik ini bertujuan memperbesar variasi data latih sehingga model menjadi lebih general dan tidak mudah mengalami *overfitting* [51]. Penelitian dalam [52] menunjukkan bahwa segmentasi data dari 4.428 menjadi 11.430 gambar mampu menurunkan nilai *loss* dan meningkatkan pembelajaran model secara signifikan dibanding data asli. Meskipun demikian, tantangan utama pengembangan CNN untuk klasifikasi multi-kelas adalah kesulitan model dalam membedakan fitur antar objek yang mirip, yang dapat menurunkan akurasi [53].



Gambar 2. 8 Model Segmentasi Citra

2.2.5 Metrik Evaluasi Model Deteksi Objek

Evaluasi performa model deteksi objek merupakan tahapan penting untuk mengukur sejauh mana model yang dibangun mampu mengenali dan melokalisasi objek dengan benar. Dalam proses evaluasi, setiap hasil prediksi model dibandingkan dengan data acuan yang telah dilabeli secara manual. Hasil prediksi dikategorikan ke dalam empat kondisi, yaitu *true positive* (TP) apabila prediksi positif benar, *false positive* (FP) apabila prediksi positif salah, *true tegative* (TN) apabila prediksi negatif benar, dan *false negative* (FN) apabila prediksi negatif salah [42].

Precision mengukur seberapa andal model dalam mengidentifikasi suatu kelas, yaitu proporsi prediksi positif yang benar-benar tepat dari seluruh prediksi positif yang dihasilkan. *Recall* mengukur kemampuan model dalam menemukan seluruh objek dari kelas tertentu, yaitu proporsi objek yang berhasil terdeteksi dari seluruh objek yang seharusnya terdeteksi. *Accuracy* merepresentasikan tingkat

ketepatan klasifikasi model secara keseluruhan, sedangkan *F1-Score* merupakan nilai harmonis antara *precision* dan *recall* yang digunakan ketika kedua metrik tersebut perlu dipertimbangkan secara seimbang [42]. Keempat metrik tersebut diformulasikan sebagai berikut:

$$\text{Precision} = \frac{TP}{TP + FP} \times 100\% \tag{2.11}$$

$$\text{Recall} = \frac{TP}{TP + FN} \times 100\% \tag{2.12}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{2.13}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \times 100\% \tag{2.14}$$

Intersection over Union (IoU) mengukur tingkat kesesuaian antara *bounding box* hasil prediksi model dengan *bounding box* pada data acuan. IoU dihitung sebagai rasio antara luas area *overlap* kedua kotak tersebut terhadap luas area *union* [54], dengan formulasi sebagai berikut:

$$\text{IoU} = \frac{\text{Area of Overlap (AO)}}{\text{Area of Union (AU)}} \tag{2.15}$$

Nilai IoU berkisar antara 0 hingga 1, di mana semakin mendekati 1 berarti lokasi prediksi semakin akurat. Suatu prediksi dianggap sebagai *true positive* apabila nilai IoU-nya melebihi ambang batas yang telah ditentukan, dan dianggap sebagai *false positive* apabila berada di bawah ambang batas tersebut [54].

Average precision (AP) dihitung untuk setiap kelas objek berdasarkan area di bawah kurva *precision-recall*-nya, Kemudian *mean average precision* (mAP) diperoleh dengan merata-ratakan nilai AP dari seluruh kelas objek yang ada [54].

$$\text{mAP} = \frac{1}{C} \sum_{i=1}^{C} AP(c) \tag{2.16}$$

mAP merupakan metrik yang paling umum digunakan dalam penelitian deteksi objek karena memberikan gambaran menyeluruh tentang performa model pada semua kelas secara sekaligus [51].

2.3 Komponen Dasar

2.3.1 Raspberry Pi 5

Raspberry Pi 5 merupakan *single-board computer* yang menggunakan *System on Chip* (SoC) Broadcom BCM2712 yang dilengkapi prosesor *quad-core* ARM Cortex-A76 dengan frekuensi hingga 2,4 GHz serta GPU VideoCore VII. Dukungan memori LPDDR4X, antarmuka PCIe, USB, ethernet, wi-fi, bluetooth, dan MIPI CSI/DSI memungkinkan raspberry pi 5 digunakan pada implementasi sistem pengolahan citra dan kecerdasan buatan berbasis *edge computing*.

Gambar 2. 9 Bentuk Fisik Raspberry Pi 5

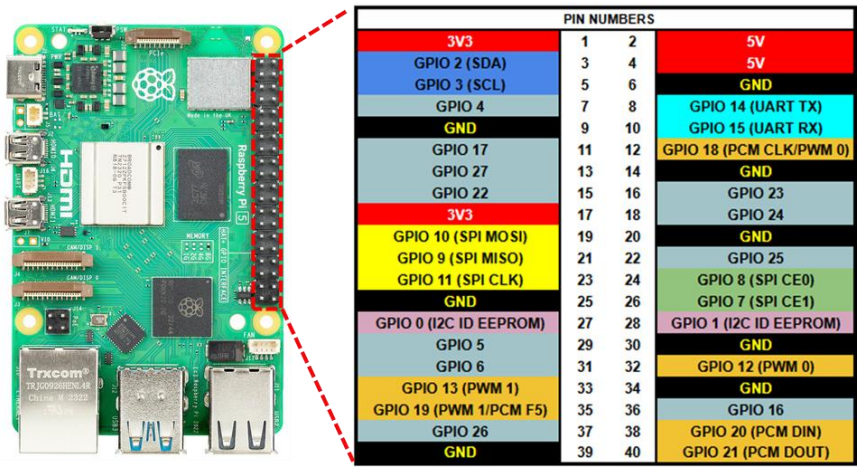
Berdasarkan gambar 2.9, raspberry pi 5 tersusun atas berbagai komponen utama yang terintegrasi pada satu papan rangkaian, meliputi prosesor, memori, antarmuka GPIO, *port* USB, *port* ethernet, konektor kamera, serta konektor daya USB-C. Integrasi berbagai komponen memungkinkan raspberry pi 5 digunakan sebagai pusat pemrosesan dalam suatu sistem tanpa memerlukan perangkat komputasi tambahan [54]. Spesifikasi perangkat keras raspberry pi 5 ditunjukan pada tabel 2.2.

Tabel 2. 2 Spesifikasi Raspberry Pi 5

Kategori	Spesifikasi
Prosesor (SoC)	Broadcom BCM2712, quad-core ARM Cortex-A76 @ 2,4 GHz
GPU	VideoCore VII, OpenGL ES 3.1, Vulkan 1.2
RAM	4 GB atau 8 GB LPDDR4X-4267 SDRAM
Penyimpanan	MicroSD (UHS-I), dukungan NVMe SSD via PCIe 2.0

Kategori	Spesifikasi
Konektivitas	Wi-Fi 802.11ac dual-band, Bluetooth 5.0, BLE
GPIO	40-pin GPIO
USB	2x USB 3.0, 2x USB 2.0
HDMI	2x micro-HDMI
Kamera / Display	2x MIPI CSI/DSI
PCIe	1x PCIe 2.0 (FPC konektor, M.2 HAT)
Ethernet	Gigabit Ethernet (RJ45, PoE+ opsional)
Tegangan Input	5V DC via USB-C (minimum 5A)
Konsumsi Daya	Sekitar 5–10 W
Ukuran Modul	85 x 56 mm
Sistem Operasi	Raspberry Pi OS, Ubuntu, dan OS berbasis Linux lainnya
Platform Pengembangan	Python, C/C++, Node.js, TensorFlow Lite, OpenCV, dan lainnya

Salah satu antarmuka yang banyak digunakan pada raspberry pi 5 adalah *general purpose input output* (GPIO) 40-pin. Antarmuka GPIO digunakan untuk komunikasi dengan berbagai perangkat eksternal seperti sensor, aktuator, maupun modul pendukung lainnya. Pengelolaan fungsi I/O dilakukan oleh chip RP1 yang terhubung dengan SoC BCM2712. Konfigurasi pin GPIO raspberry pi 5 ditunjukkan pada Gambar 2.10.



Gambar 2. 10 *Pinout* Raspberry Pi 5

Berdasarkan gambar 2.10, pin GPIO raspberry pi 5 terdiri atas jalur catu daya, *ground*, serta pin masukan dan keluaran digital yang dapat dikonfigurasi sesuai kebutuhan aplikasi. Beberapa pin mendukung fungsi khusus seperti UART, I2C, SPI, dan PWM yang digunakan untuk komunikasi dengan perangkat eksternal.

2.3.2 Hailo 8L

Hailo-8L merupakan sebuah *chip* kecerdasan buatan berbasis *neural processing unit (*NPU) yang dirancang secara khusus untuk keperluan inferensi deep learning di sisi perangkat keras tepi. *Chip* hailo-8l dirancang dengan pendekatan arsitektur komputasi *structured dataflow architecture* (SDA).

Arsitektur SDA memungkinkan aliran data antarlapisan jaringan saraf dieksekusi secara langsung di dalam *chip* tanpa memerlukan akses memori eksternal yang berulang, sehingga meminimalkan latensi sekaligus mengoptimalkan konsumsi energi. Hailo-8L merupakan varian dari seri hailo-8 yang ditujukan untuk segmen *entry-level edge AI* dengan kemampuan komputasi mencapai 13 *Tera Operations Per Second* (TOPS).

Gambar 2. 11 Bentuk Fisik Hailo-8L

Hailo-8L menggunakan format modul M.2 2242 dengan antarmuka PCIe gen 2.0 x1 yang dapat dikonfigurasi hingga PCIe gen 3.0 pada raspberry pi 5, sehingga dapat dipasang langsung melalui *raspberry pi M.2 HAT*. *Chip* ini mendukung konversi model dari berbagai *framework* pembelajaran mendalam populer seperti tensorflow, pytorch, dan ONNX. Hailo-8L memiliki arsitektur jaringan saraf standar seperti ResNet, MobileNet, YOLO, dan EfficientDet dapat dioptimalkan menjadi format *hailo executable format* (.hef) menggunakan *hailo dataflow compiler* (DFC).

Tabel 2. 3 Spesifikasi Hailo-8L

Kategori	Spesifikasi
Tipe Chip	Neural Processing Unit (NPU) Edge AI
Performa Komputasi	Hingga 13 TOPS (Tera Operations Per Second)
Antarmuka Host	PCIe Gen 2.0 x1 / Gen 3.0 x1 (M.2 M-Key / B+M-Key)
Presisi Komputasi	INT4, INT8, FP16 (mixed-precision inference)
Konsumsi Daya Maksimum	2,5 Watt (Thermal Design Power / TDP)
Tegangan Operasional	3,3V (dari slot M.2 host)
Ukuran Modul	M.2 2242 (22 x 42 mm)

Kategori	Spesifikasi
Suhu Operasional	-40°C hingga +85°C
Framework Kompatibel	TensorFlow, PyTorch, ONNX (via konversi ke format .hef)
SDK Pengembangan	Hailo AI Software Suite (HailoRT, Hailo Dataflow Compiler, Hailo Model Zoo)
Platform Kompatibel	Raspberry Pi 5, NVIDIA Jetson, sistem Linux berbasis x86/ARM
Sistem Operasi	Linux (Ubuntu 20.04/22.04, Raspberry Pi OS Bookworm)

Hailo-8L mengeksekusi seluruh lapisan jaringan saraf secara paralel dan berjenjang di dalam *chip*. Pendekatan ini meminimalkan transfer data bolak-balik antara *prosesor* dan memori utama eksternal yang sering kali menjadi *bottleneck* energi dan latensi pada sistem inferensi konvensional. Secara umum, alur kerja implementasi dan inferensi pada Hailo-8L dijabarkan sebagai berikut:

1. Model jaringan saraf yang telah dilatih pada *framework* eksternal dikompilasi terlebih dahulu menggunakan *hailo dataflow compiler* (DFC) di lingkungan PC pengembangan. Proses mencakup kuantisasi bobot model ke format INT8 atau INT4 serta pemetaan arsitektur lapisan ke sumber daya komputasi fisik internal *chip* untuk menghasilkan file eksekusi .hef.
2. Data masukan berupa frame citra digital dari kamera dikirimkan oleh host ke modul hailo-8l melalui jalur antarmuka PCIe. Komunikasi data dikelola sepenuhnya oleh driver kernel HailoRT yang berjalan di bawah sistem operasi host.
3. Hailo-8L melakukan proses komputasi inferensi secara mandiri menggunakan unit pemrosesan paralel yang telah terpetakan. Setiap lapisan CNN dieksekusi dalam pipeline internal tanpa melibatkan intervensi dari CPU raspberry pi 5, sehingga meminimalkan beban komputasi pada host.
4. NPU mengirimkan kembali hasil keluaran berupa data bounding box, skor tingkat confidence score, atau prediksi kelas menuju host via PCIe.

Informasi tersebut kemudian siap dieksekusi lebih lanjut oleh aplikasi utama pada raspberry pi 5.

2.3.3 *High Quality Camera* IMX 477

HQ Camera bekerja berdasarkan prinsip konversi energi foton cahaya menjadi sinyal elektrik melalui sensor *complementary metal-oxide-semiconductor* (CMOS). Cahaya yang masuk melalui lensa difokuskan menuju permukaan sensor CMOS, setiap piksel pada sensor mengandung *fotodiode* yang mengubah intensitas cahaya menjadi muatan listrik. Muatan listrik dikonversi ke nilai digital melalui konverter analog ke digital yang terintegrasi di dalam *chip* sensor.

Gambar 2. 12 Bentuk Fisik HQ Camera

Sensor sony IMX477 yang digunakan pada HQ camera mengadopsi teknologi *back-side illuminated* (BSI). Teknologi BSI menerapkan konfigurasi di mana lapisan sirkuit pengolahan sinyal diletakkan di bagian belakang substrat silikon, sehingga permukaan depan sensor dapat menangkap lebih banyak cahaya secara langsung tanpa terhalang oleh komponen sirkuit. Implementasi arsitektur BSI meningkatkan sensitivitas sensor secara signifikan, terutama pada kondisi pencahayaan rendah. Modul HQ Camera dengan lensa industri berstandar *CS-mount* yang memiliki kapasitas daya urai optis sebesar 5 megapiksel. Lensa dirancang untuk memproyeksikan *detail* gambar dengan ketajaman tinggi yang dioptimasi agar selaras dengan kerapatan piksel sensor modern. Spesifikasi lengkap dari kombinasi HQ Camera dan lensa 5 megapiksel dijabarkan pada Tabel 2. berikut.

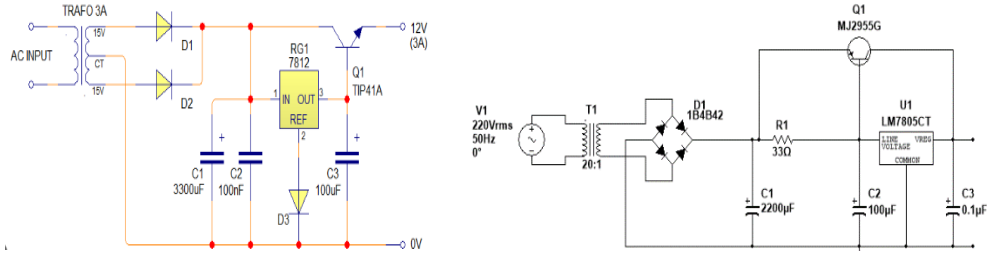
Tabel 2. 4 Spesifikasi HQ Camera

Kategori	Spesifikasi
Sensor gambar	Sony imx477, 1/2,3 inci, bsi cmos
Resolusi sensor fisik	12,3 megapiksel (4056 x 3040 piksel)
Ukuran piksel sensor	1,55 um x 1,55 um
Resolusi operasi sistem	2592 x 1944 piksel
Format dudukan lensa	Cs-mount / c-mount
Panjang fokus	3,04 mm - 16 mm
Apertur (f-nu	F/1,8 - f/16
Format output	Raw bayer, jpeg, rgb
Antarmuka komunikasi	Mipi csi-2
Kontrol eksposur	Manual dan otomatis (aec/agc)
Rentang dinamis	> 66 db
Kecepatan bingkai	Hingga 30 fps pada resolusi penuh
Tegangan operasi	1,8 v / 2,8 v

2.3.4 *Catu Daya*

Catu daya merupakan perangkat yang berfungsi mengubah energi listrik dari sumber utama menjadi tegangan dan arus yang sesuai dengan kebutuhan beban. Catu daya mengubah tegangan arus bolak-balik menjadi tegangan arus searah yang stabil. Tegangan 12 V DC banyak digunakan pada perangkat, seperti aktuator, modul pencahayaan, dan beberapa jenis sensor industri. Tegangan 5 V DC umum digunakan pada rangkaian logika digital, mikrokontroler, komputer papan tunggal, serta modul komunikasi karena sesuai dengan kebutuhan tegangan operasi sebagian

besar perangkat elektronika modern. Bentuk rangkaian catu daya yang digunakan ditunjukkan pada Gambar 2.13.



Gambar 2. 13 Catu Daya 12V dan 5V

Berdasarkan Gambar 2.13, catu daya yang digunakan menghasilkan tegangan keluaran 5 V DC dan 12 V DC sesuai kebutuhan perangkat yang disuplai. Prinsip kerja kedua catu daya tersebut pada dasarnya sama, yaitu melalui proses penyearahan, pensaklaran frekuensi tinggi, transformasi tegangan, penyaringan, dan pengaturan tegangan keluaran melalui mekanisme umpan balik.

2.3.6 Lampu *Led Eagle Eye*

Konfigurasi desain fisik komponen LED mengintegrasikan satu atau lebih *chip diode* pemancar cahaya di dalam selubung logam aluminium tahan korosi, dilindungi oleh lensa cembung berbahan akrilik atau kaca pada bagian atasnya. Lensa cembung memproksikan sudut pancaran cahaya menjadi lebih sempit dan terarah, sehingga menghasilkan fluks lumen yang efisien tanpa terjadi dispersi cahaya yang melebar.

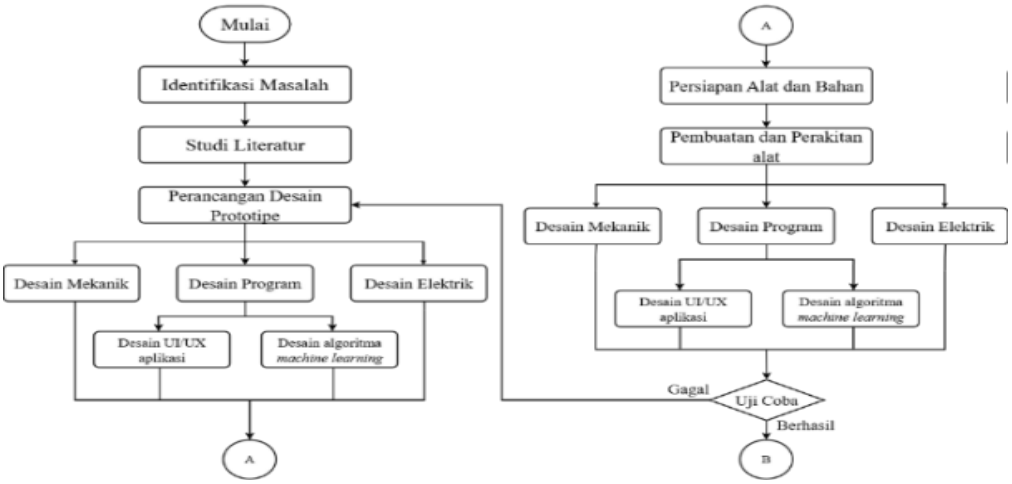
Gambar 2. 14 Lampu *Eagle Eye*

LED *eagle eye* sering diimplementasikan sebagai perangkat aktuator visual, lampu indikator status sistem, maupun pencahayaan fokal pada area pengamatan visual. Keunggulan mekanis utama dari modul ini terletak pada struktur housing berulir baut yang dilengkapi mur pengunci. Desain memberikan tingkat durabilitas yang tinggi terhadap getaran mekanis serta memudahkan proses instalasi vertikal pada dinding panel kontrol, akrilik, maupun struktur rangka prototipe luar ruang

BAB III
PERANCANGAN TUGAS AKHIR

3.1 Metode Penyusunan

Dalam penyusunan tugas akhir ini, penulis menerapkan metode eksperimental. Proses penyusunannya mencakup beberapa tahapan yang saling berkaitan. Alur tahapan penelitian secara keseluruhan dapat dilihat pada diagram alir berikut:



Gambar 3. 1 Diagram Alir Prosedur Kerja Pembuatan Prototipe

Adapun penjelasan dari masing-masing tahapannya adalah sebagai berikut:

1. Identifikasi Masalah

   Tahap awal penelitian dimulai dengan mengidentifikasi masalah seputar metode sistem deteksi penyakit udang. Proses ini dilakukan dengan mengkaji berbagai permasalahan yang relevan dengan topik penelitian.

2. Studi Literatur

   Mempelajari referensi terkait dengan deteksi objek melalui visual untuk mendalami pemahaman tentang prinsip dasar yang mendukung pengembangan produk. Proses dilakukan mengkaji berbagai sumber tertulis yang relevan dengan topik penelitian, meliputi buku, jurnal ilmiah, artikel, laporan penelitian, makalah, dokumen resmi, serta sumber tertulis lainnya yang mendukung.

3. Perancangan Produk

   Perancangan produk dibagi menjadi tiga bagian: perancangan desain 3D, pemrograman, dan training data menggunakan roboflow.

4. Pengukuran Kinerja Produk

   Prosedur pengukuran kinerja dilakukan untuk mendapatkan *feedback* dari hasil pengujian sistem mekanik, elektrik, serta program produk yang dirancang.

3.2 Populasi dan Sampel Penelitian

3.2.1 Populasi Penelitian

Populasi penelitian dibagi menjadi dua aspek yang saling terkait. Pertama, populasi data citra, yaitu seluruh kemungkinan citra digital udang vaname yang memperlihatkan kondisi klinis penyakit IMNV, TSV, WSSV, maupun kondisi sehat, yang dapat diperoleh melalui akuisisi kamera secara langsung maupun dari repositori dataset publik. Kedua, populasi objek fisik, yaitu populasi udang vaname yang dibudidayakan di tambak intensif dan berpotensi mengalami infeksi penyakit yang menjadi target deteksi sistem.

3.2.2 Teknik Sampling dan Ukuran Sampel

Teknik pengambilan sampel yang digunakan adalah *purposive sampling*, yaitu teknik pengambilan sampel berdasarkan pertimbangan dan kriteria tertentu yang ditetapkan peneliti sesuai dengan tujuan penelitian [55]. Teknik *purposive sampling* dipilih karena penelitian memerlukan sampel citra yang secara spesifik merepresentasikan gejala klinis visual dari empat kelas target, yakni IMNV, TSV, WSSV, dan udang sehat, sehingga tidak semua citra yang tersedia relevan untuk digunakan. Kriteria inklusi sampel citra yang ditetapkan adalah sebagai berikut:

1. Citra menampilkan morfologi udang vaname secara jelas dengan kualitas resolusi minimal 640×480 piksel.
2. Citra memperlihatkan gejala klinis yang dapat diidentifikasi secara visual, seperti perubahan warna tubuh, nekrosis otot, atau bercak putih pada karapas.
3. Citra diambil dalam kondisi kecerahan air berkisar antara 30–100 cm sesuai kondisi riil tambak intensif.

4. Citra telah melalui proses anotasi oleh tenaga ahli atau mengacu pada referensi literatur yang valid.

Sedangkan kriteria eksklusi meliputi citra yang buram atau teroklusif sehingga gejala klinis tidak dapat diidentifikasi, serta citra yang tidak dapat dikonfirmasi kelasnya berdasarkan diagnosis laboratorium. Penentuan jumlah sampel dataset mengacu pada hasil penelitian [56], yang menunjukkan bahwa dataset berukuran 800–1.200 gambar per kelas sudah memadai untuk mencapai hasil deteksi optimal dalam skala penelitian laboratorium. lebih lanjut membuktikan bahwa augmentasi data dari 4.428 menjadi 11.430 gambar dapat menurunkan nilai loss secara signifikan dan meningkatkan kemampuan generalisasi model [52]. Total dataset yang digunakan dalam penelitian ini ditetapkan sebesar 3.000 citra, yang terbagi secara proporsional ke dalam empat kelas sebagai berikut:

Tabel 3. 1 Distribusi Sampel Dataset Citra per Kelas

No.	Kelas Penyakit	Gejala Klinis Visual	Jumlah Sampel
1	IMNV	Nekrosis otot putih pada abdomen, tubuh opak	780
2	TSV	Nekrosis epitelium kutikula, pewarnaan merah pada pleopod	600
3	WSSV	Bercak putih pada karapas dan tubuh	750
4	Udang Sehat	Morfologi normal, warna transparan keabuan	870
Total			3.000 sampel

Pembagian yang seimbang antara keempat kelas dilakukan untuk menghindari class imbalance yang dapat menyebabkan model CNN cenderung bias terhadap kelas mayoritas, sehingga akurasi deteksi pada kelas minoritas menjadi rendah. Pengujian prototipe menggunakan sampel udang vaname yang berjumlah 130 ekor dengan usia 40 hari, konsisten dengan kondisi pengujian yang telah

dilakukan pada tahap pengembangan sistem. Pengujian fisik dilaksanakan di akuarium berskala laboratorium dengan kondisi kecerahan air sebesar 40 cm.

3.3 Teknik Pengumpulan Data

3.3.1 Observasi Terstruktur

Observasi terstruktur dilakukan untuk memperoleh data primer berupa citra visual udang vaname secara langsung. Observasi dilaksanakan di akuarium laboratorium dengan menggunakan kamera *high quality camera* yang terpasang pada *housing* kedap air prototipe. Kamera menangkap citra udang secara *real-time* di bawah air dengan kondisi kecerahan yang telah dikendalikan. Panduan observasi mencakup aspek-aspek berikut:

1. Kondisi fisik udang yakni warna tubuh, kondisi karapas, morfologi *pleopod*, dan tingkat aktivitas pergerakan.
2. Parameter lingkungan akuarium yakni kecerahan air, suhu, dan kondisi pencahayaan artifisial.
3. Performa sistem akuisisi citra yakni *frame rate*, resolusi, dan kualitas gambar yang dihasilkan.

3.3.2 Eksperimen Laboratorium

Data kinerja prototipe dikumpulkan melalui serangkaian eksperimen terstruktur di laboratorium. Eksperimen dilakukan dengan menguji prototipe pada sampel udang vaname berusia 40 hari dengan total 130 ekor dalam kondisi terkontrol. Sistem secara otomatis memproses *frame* video menggunakan algoritma YOLO berbasis CNN, mengklasifikasikan kondisi udang ke dalam empat kelas yakni IMNV, TSV, WSSV, dan sehat, kemudian mengirimkan hasil notifikasi ke aplikasi *smartphone*. Data yang dikumpulkan mencakup jumlah deteksi yang benar, deteksi yang salah, serta waktu respon sistem.

Tabel 3. 2 Instrumen Pengumpulan Data

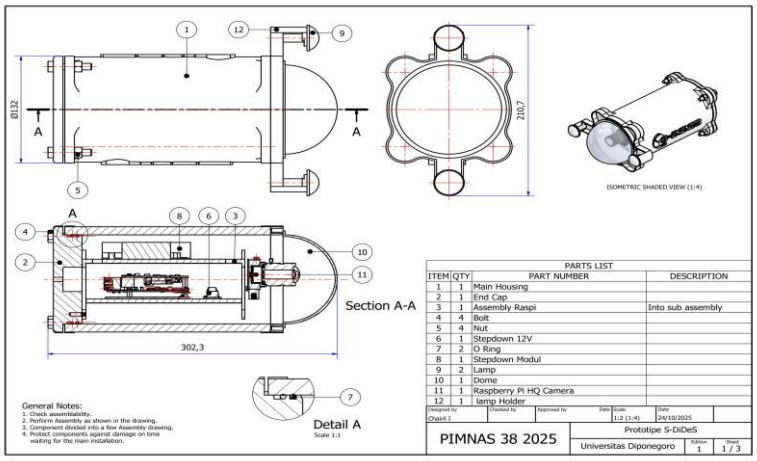
Aspek Data	Teknik Pengumpulan	Instrumen	Keterangan
Dataset citra penyakit IMNV,	Dokumentasi + observasi	HQ Camera, anotasi Roboflow	3.000 citra, 4 kelas

TSV, WSSV, sehat			
Akurasi deteksi model CNN	Eksperimen	Confusion matrix, mAP@0.5	Per kelas penyakit
Nilai loss pelatihan model	Eksperimen	Log training YOLO (train/val loss)	Per epoch
Waktu respon deteksi *real-time*	Eksperimen	Timestamp sistem (ms)	Per frame deteksi
Kinerja sistem *hardware*	Observasi + eksperimen	Log sistem Raspberry Pi 5	Saat sistem berjalan

3.4 Perancangan Perangkat Keras

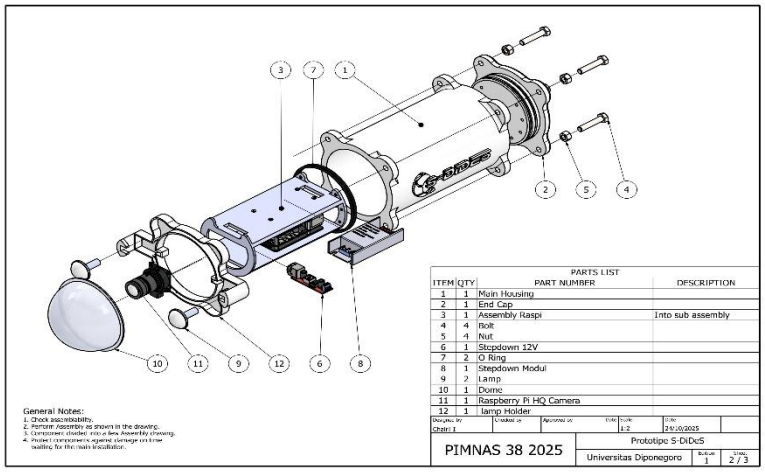
3.4.1 Perancangan Desain Mekanik

Perancangan desain mekanik mencakup mekanik *housing* kedap air dan tata letak komponen elektronik, Proses perancangan desain mekanik menggunakan *software* SolidWorks dalam memodelkan desai dengan material PLA sebagai bahan dasar *housing*. Beberapa aspek dalam pemodelan dibagi menjadi 3 yakni tampak potongan, *isometrik eksplosif*, detail komponen, bebrapa desain tersebut ditunjukkan pada gambar dibawah.



Gambar 3. 2 *Section View*

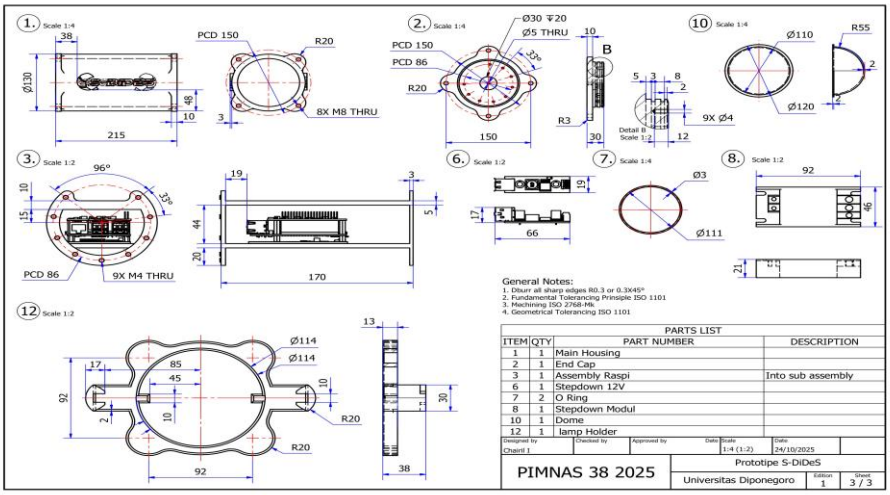
Model tampak potongan dari *housing* perangkat ditunjukkan pada Gambar 3.3. Melalui visualisasi potongan, dapat dianalisis ketebalan dinding *housing* serta kerapatan ruang bagian dalam yang dirancang khusus agar memenuhi kriteria kedap air. Tampak potongan memperlihatkan alur pembatas tempat peletakan *sealent* atau karet pelindung pada sambungan antar kompartemen guna mencegah masuknya air atau kelembapan tinggi dari luar yang dapat merusak komponen elektronik di dalamnya.



Gambar 3. 3 *Exploded View*

Tahap perencanaan desain mekanis dilakukan pemodelan isometrik eksplosif untuk memberikan gambaran menyeluruh mengenai tata letak komponen elektronik serta integrasi antarbagian mekanik, sebagaimana ditunjukkan pada Gambar 3.4. Desain eksplosif tersebut menguraikan susunan perangkat keras utama

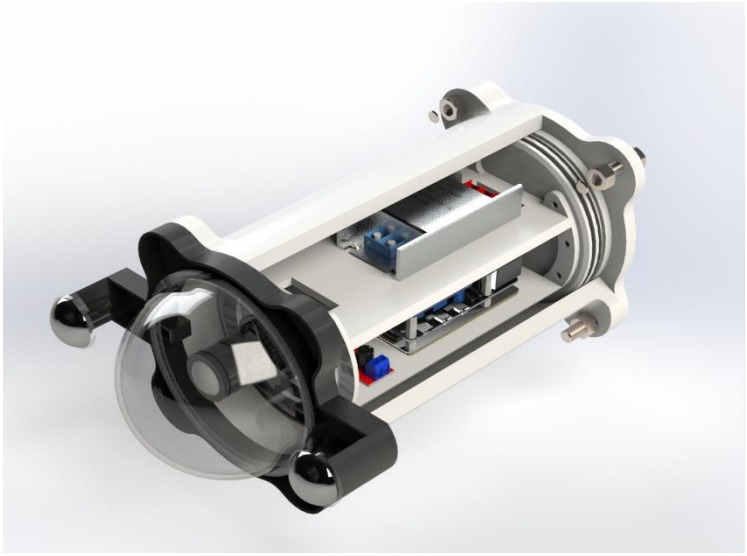
secara berurutan, mulai dari dudukan mikrokontroler berupa Raspberry Pi 5, modul sensor, sistem manajemen daya, hingga penutup *housing*. Melalui pendekatan ini, proses perakitan dan pemeliharaan perangkat di lapangan dapat dirancang secara lebih sistematis, terstruktur, dan efisien.



Gambar 3. 4 Penggambaran Detail Prototipe

Dimensi spesifik dan geometri dari setiap detail komponen mekanik diperlihatkan pada Gambar 3.5. Pemodelan detail sangat krusial dalam memastikan tingkat presisi saat proses manufaktur menggunakan metode 3D Printing dengan material *PolyLactic Acid* (PLA). Aspek detail yang diperhatikan meliputi toleransi lubang sekrup pengencang, struktur penyangga interior, serta lubang ventilasi atau integrasi sensor luar yang tetap mempertahankan aspek kekedapan ruang utama.

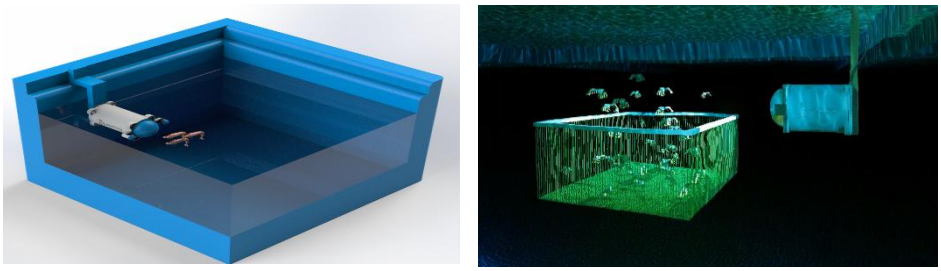
Dengan demikian, rancangan keseluruhan dari desain mekanik housing perangkat, baik dari segi dimensi luar maupun integrasi komponen di dalamnya, ditunjukkan secara detail pada Gambar 3.6.



Gambar 3. 5 Desain Prototipe

Desain keseluruhan pada gambar 3.6 ini memastikan *housing* berbasis material PLA tidak hanya melindungi komponen elektronik dari paparan air, tetapi juga memiliki struktur yang kokoh dan fungsional.

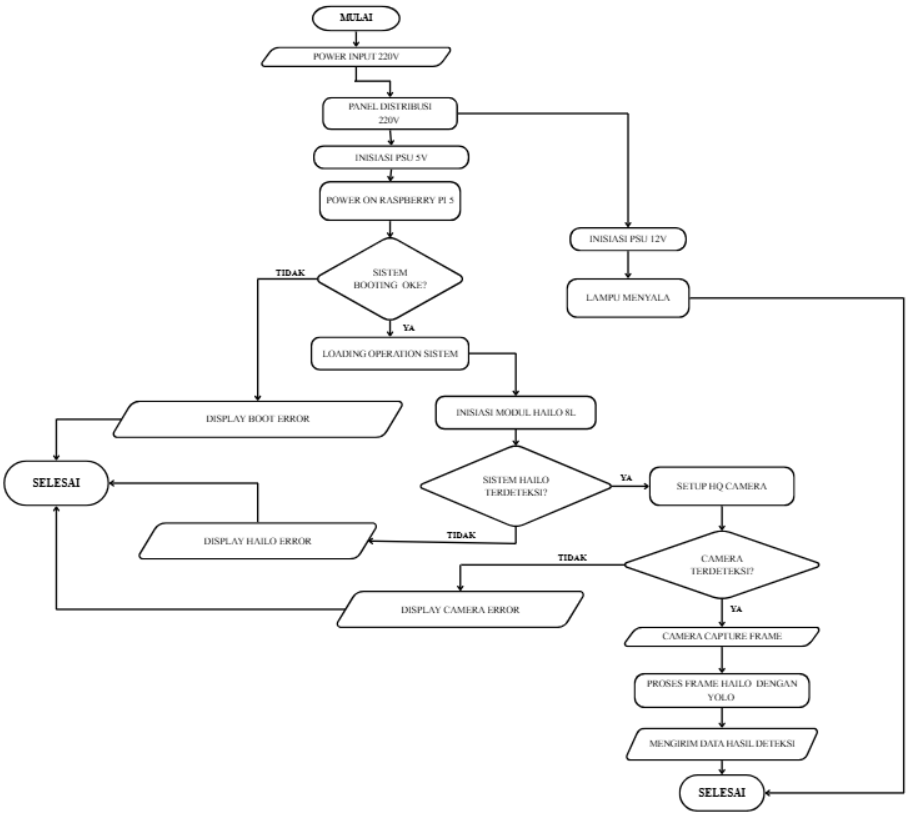
Aspek mekanis penunjang performa sistem di lapangan diatur melalui perencanaan tata letak komponen optik yang presisi. Posisi kamera ditempatkan dengan orientasi tegak lurus atau membentuk sudut 90 derajat terhadap bidang horizontal permukaan air. Orientasi pada 90 derajat, menjadi solusi mekanis dalam mengatasi tantangan utama pada sistem deteksi dini, yaitu untuk meminimalkan bias bias cahaya dan memaksimalkan pengambilan citra udang dengan kualitas, ketajaman, serta fokus yang baik meskipun berada di kondisi perairan yang keruh. Desain perencanaan tata letak dan posisi peletakan keseluruhan prototipe saat diimplementasikan ditunjukkan pada Gambar 3.7.



Gambar 3. 6 Posisi Peletakan Prototipe

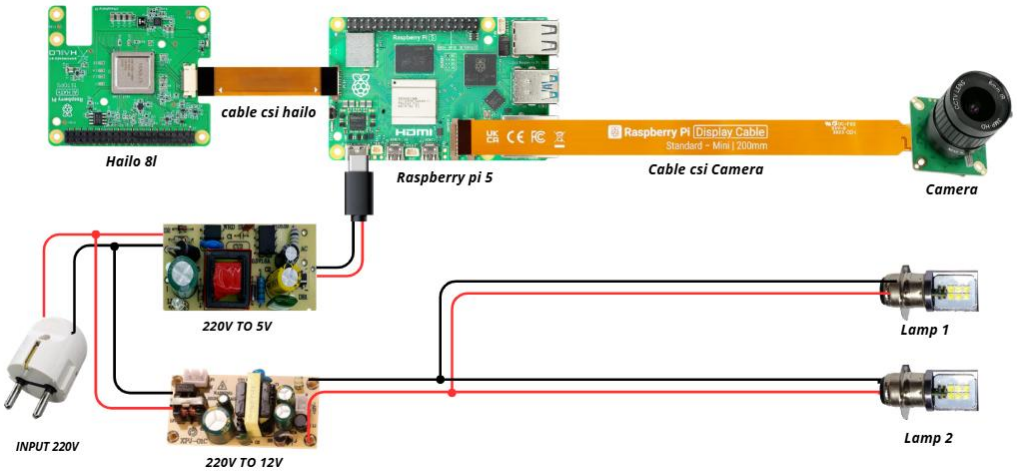
3.4.2 Perancangan Rangkaian Elektrik

Dalam proses perancangan pembuatan rangkaian elektrik, berfokus pada perancangan logika sistem bekerja serta perencanaan komponen yang digunakan. Perancangannya dapat dilihat pada Gambar 3.2.



Gambar 3. 7 *Flowchart* Perancangan Elektrik

Proses diawali dengan inisialisasi daya dan penerangan, dilanjutkan dengan validasi sistem operasi modul, dan kamera. Setelah validasi, sistem mengakuisisi *frame*, memprosesnya dengan dengan algoritma CNN, dan mengirim hasil deteksi secara berkelanjutan. Untuk memberikan gambaran yang lebih detail mengenai interkoneksi pin dan jalur distribusi daya antar komponen, rancangan skematik elektrik prototipe ini ditunjukkan pada Gambar 3.8.



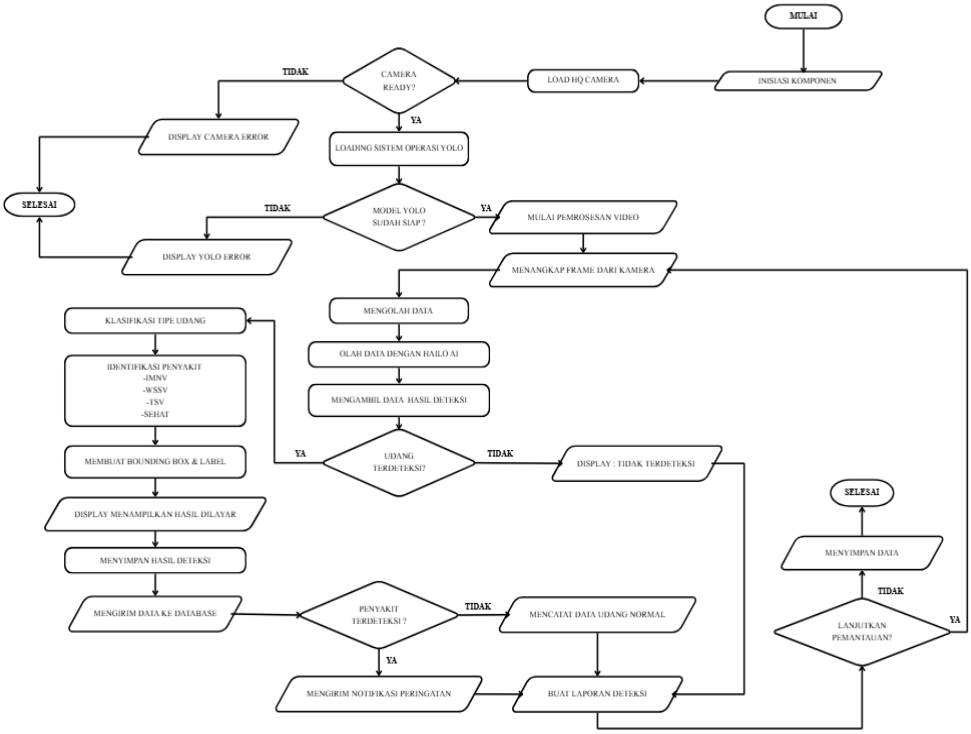
Gambar 3. 8 Rangkaian Elektrik

Berdasarkan Gambar 3.8, sumber daya utama sistem berasal dari adaptor 5V yang menyuplai daya ke Raspberry Pi 5 melalui konektor USB-C. Raspberry Pi 5 selanjutnya mendistribusikan daya ke akselerator Hailo 8L melalui antarmuka PCIe M.2, serta ke kamera melalui antarmuka CSI. Pemilihan Raspberry Pi 5 sebagai unit pemrosesan pusat didasarkan pada kemampuannya menyediakan antarmuka PCIe generasi tiga yang diperlukan oleh akselerator Hailo 8L guna mencapai throughput inferensi yang memadai untuk pemrosesan citra secara real time. Akselerator Hailo 8L berperan sebagai unit pemrosesan neural network yang menjalankan inferensi model YOLO secara efisien dengan konsumsi daya rendah. Komponen ini terhubung ke Raspberry Pi 5 melalui slot M.2 Key M dan dikomunikasikan menggunakan driver HailoRT. Dengan kapasitas komputasi sebesar 13 TOPS, Hailo 8L mampu menjalankan model deteksi objek berbasis CNN secara real-time tanpa membebani CPU Raspberry Pi 5 secara berlebihan, sehingga penggunaan CPU dapat dijaga di bawah ambang batas 80% selama proses inferensi berlangsung.

Kamera terhubung ke Raspberry Pi 5 melalui antarmuka CSI yang menyediakan *bandwidth* transmisi data citra beresolusi tinggi dengan latensi rendah. Kamera ditempatkan pada orientasi 90 derajat terhadap bidang horizontal air sebagaimana yang telah diuraikan pada sub-bab perancangan mekanik, sehingga arah pandang kamera tegak lurus ke bawah menuju dasar kolam tempat aktivitas udang vaname dominan berlangsung. Penggunaan antarmuka CSI dibandingkan

USB dipilih karena memberikan stabilitas transmisi yang lebih baik dan tidak mengalami *overhead* protokol yang dapat memperlambat akuisisi frame.

3.5 Perancangan Perangkat Lunak



Gambar 3. 9 *Flowchart* Perancangan Perangkat Lunak

Perancangan perangkat lunak sistem mengacu pada diagram alir yang disajikan pada gambar 3.6. Sistem dirancang menggunakan arsitektur berlapis yang terbagi menjadi lima lapisan utama, sebagaimana ditunjukkan pada tabel 3. Setiap lapisan memiliki tanggung jawab yang terpisah dan berkomunikasi melalui antarmuka yang terdefinisi dengan baik, sehingga sistem bersifat modular, dapat diperluas, dan mudah dirawat.

Tabel 3. 3 Alur Komunikasi Perangkat Lunak

Layer	Komponen	Fungsi	Teknologi
1.	API *Layer*	*Endpoint* untuk komunikasi Flutter ↔ *backend*	FastAPI, Uvicorn, Python 3.11

Layer	Komponen	Fungsi	Teknologi
2.	*Storage Layer*	Penyimpanan riwayat deteksi, *frame*, dan token FCM	SQLite 3, sistem file Raspberry Pi
3.	Notification Layer	*Push notification* ke *smartphone*	Firebase Cloud Messaging (FCM)
4.	Business Logic Layer	Inferensi CNN, *preprocessing*, NMS, klasifikasi	CNN, Hailo 8L NPU, OpenCV
5.	Presentation Layer	Antarmuka pengguna: monitor, penyakit, riwayat	Flutter, Material Design 3

3.5.1 Perancangan Alur Deteksi *Real Time*

Alur komunikasi data pada sistem mengikuti pola *request response* yang sinkron untuk deteksi, dengan mekanisme *background task* untuk notifikasi asinkron. Berikut pada tabel 3.4 adalah langkah-langkah alur komunikasi antar komponen.

NO	Komponen	Aksi & *Payload*
1.	HQ Camera → Raspberry Pi	Kamera menangkap *frame* citra secara kontinu, kemudian meneruskan data gambar mentah ke proses utama Python.
2.	Flutter App → FastAPI	Mengirimkan berkas *frame* dalam format JPEG sebagai *multipart/form-data* menuju *endpoint* pada *backend*.
3.	FastAPI → Hailo 8L	Melakukan prapemrosesan citra, lalu meneruskan data ke NPU Hailo 8L melalui *hailo-platform* SDK.

NO	Komponen	Aksi & *Payload*
4.	Hailo 8L → FastAPI	Mengeksekusi inferensi model *deep learning* dengan estimasi waktu 45 ms dan mengembalikan *output tensor* berupa koordinat *bounding box* serta nilai kepercayaan.
5.	FastAPI → SQLite	Melakukan operasi *insert* data untuk menyimpan log hasil identifikasi penyakit berupa label penyakit, koordinat, *confidence*, dan *timestamp* ke dalam tabel basis data lokal.
6.	FastAPI → Firebase FCM	*backend* mengirimkan data peringatan ke topik agar *firebase* meneruskan *push notification* ke gawai pengguna.
7.	FastAPI → Flutter App	status deteksi, label penyakit, *confidence score*, koordinat *bounding box*, dan saran penanganan
8.	Flutter App → Pembudidaya	UI menampilkan hasil berupa *badge* penyakit, persentase, *bounding box overlay*, dan saran penanganan.

Berdasarkan data yang dijabarkan pada tabel di atas, mekanisme pertukaran data antara perangkat keras dan perangkat lunak berjalan secara simultan untuk memastikan efisiensi deteksi dini. Proses dimulai di sisi *edge* melalui akuisisi citra yang terhubung pada raspberry pin 5. *Frame* resolusi tinggi tersebut kemudian diproses menggunakan bahasa pemrograman Python menjadi format array NumPy sebelum ditransmisikan menuju *endpoint backend* berbasis FastAPI pada aplikasi flutter.

Aspek kritikal dalam alur komunikasi ini terletak pada optimalisasi komputasi menggunakan NPU Hailo 8L. Sebelum model *deep learning* melakukan inferensi, FastAPI melakukan prapemrosesan berupa perubahan ukuran gambar. Integrasi dengan NPU Hailo 8L melalui SDK khusus terbukti mampu menekan

waktu inferensi hingga kecepatan pemrosesan ini menghasilkan *output* tensor berupa koordinat *bounding box* serta tingkat kepercayaan secara *real-time*.

Data hasil deteksi didistribusikan ke dalam dua jalur utama secara asinkron. Jalur pertama berfokus pada penyimpanan data lokal ke dalam basis data SQLite sebagai pemenuhan aspek riwayat data. Jalur kedua berfokus pada manajemen notifikasi berbasis *firebase cloud messaging* (FCM). Jika sistem mendeteksi adanya indikasi penyakit pada udang, sinyal peringatan akan langsung dikirimkan ke topik shrimp_alerts. Dengan demikian, pembudidaya dapat menerima respons JSON yang komprehensif pada antarmuka aplikasi flutter berupa tampilan *overlay bounding box,* persentase akurasi, tingkat keparahan, serta rekomendasi tindakan penanganan secara instan.

3.5.2 Perancangan Alur Notifikasi

Tahap awal saat aplikasi, sistem melakukan sinkronisasi dengan mendaftarkan identitas perangkat dan token FCM ke bagian backend. Informasi token ini kemudian disimpan ke dalam basis data lokal dan secara otomatis didaftarkan pada topik pemantauan khusus penyakit udang.

Alur pengiriman notifikasi dipicu secara asinkron melalui *background task* di sisi *backend* segera setelah algoritma kecerdasan buatan mendeteksi adanya indikasi penyakit. Layanan *firebase* kemudian memaketkan data hasil deteksi yang meliputi nama penyakit, tingkat kepercayaan model, serta penanda waktu kejadian menjadi sebuah payload untuk didistribusikan ke seluruh perangkat pembudidaya yang aktif.

aplikasi dikondisikan untuk mampu menerima dan merespons data kiriman tersebut dalam berbagai status operasional gawai, baik saat aplikasi sedang dibuka, berjalan di latar belakang, maupun dalam keadaan mati.

3.5.3 Perancangan Sistem *Back End*

Bagian *back-end* pada sistem bertindak sebagai pusat kendali operasional yang menjembatani interaksi antara perangkat keras akuisisi citra di lapangan dengan aplikasi antarmuka pengguna. Fungsi utama dari arsitektur *back-end* adalah mengelola lalu lintas data, mengeksekusi algoritma pembelajaran mendalam secara

real-time, melakukan manajemen penyimpanan data riwayat, serta mengatur distribusi informasi peringatan dini.

Kode program dan fungsionalitas di dalam *back-end* ini diorganisasikan ke dalam struktur proyek yang modular. Berdasarkan pendekatan tersebut, arsitektur *back-end* ini dibagi menjadi tiga pilar lapisan fungsional utama, yaitu:

1. Pada pilar pertama, manajemen inti dan konfigurasi bertindak sebagai fondasi utama yang mengatur seluruh siklus hidup aplikasi serta lalu lintas data. Bagian pertama bertanggung jawab dalam mengoordinasikan *end point*, mengontrol parameter keamanan akses, serta mengelola standarisasi variabel operasional seperti *threshold* dan kredensial sistem secara terpusat. Dengan adanya standarisasi konfigurasi ini, penyetelan ulang sistem saat diimplementasikan pada kondisi tambak yang berbeda dapat dilakukan tanpa mengganggu stabilitas program utama.
2. Pilar kedua berfokus pada kecerdasan buatan dan validasi data, yang menjadi motor penggerak dalam proses deteksi dini penyakit udang. Mekanisme di dalam kluster ini mencakup manajemen siklus pemrosesan citra secara utuh, mulai dari pemuatan model deteksi ke dalam unit pemrosesan, prapemrosesan piksel, hingga reduksi redundansi visual melalui metode *non-maximum suppression* (NMS). Selain itu, untuk menjamin integritas data yang ditransmisikan dalam jaringan, diterapkan pemodelan skema data yang ketat guna memvalidasi struktur informasi hasil deteksi, riwayat pemantauan, dan muatan data notifikasi sebelum didistribusikan ke komponen lain.
3. Pilar terakhir mengombinasikan fungsi persistensi data lokal dan interaksi layanan luar. Pengelolaan basis data relasional dioptimalkan untuk menangani operasi pencatatan log deteksi, pembaruan identitas digital gawai pengguna, serta kompilasi data historis secara aktual. Sinkronisasi data tersebut kemudian dimanfaatkan oleh modul layanan luar untuk menjalankan dua fungsi krusial. Pertama, menjembatani instruksi distribusi notifikasi instan berbasis awan ke perangkat pembudidaya. Kedua, mengekstraksi akumulasi data mentah menjadi bentuk laporan evaluasi berkala yang

informatif, lengkap dengan visualisasi tren penyebaran penyakit dan rekomendasi tindakan mitigasi otomatis bagi pengguna.

3.5.4 Perancangan Sistem *Front End*

Bagian *front-end* pada sistem bertindak sebagai antarmuka pengguna berbasis *mobile* yang berfungsi menyajikan visualisasi data dan informasi dari pusat kendali secara interaktif. Fungsi utama dari arsitektur *front-end* adalah menampilkan tayangan langsung citra tambak, menyediakan kendali operasional perangkat keras, menyajikan ensiklopedia penyakit udang, serta menampilkan visualisasi riwayat deteksi berkala.

Pengembangan *front end* dirancang menggunakan *framework flutter* dengan bahasa pemrograman *dart*. Sistem dikembangkan menggunakan pola arsitektur berlapis yang memisahkan komponen tampilan visual, elemen desain modular, layanan eksternal, dan pemodelan data. Berdasarkan pendekatan modular tersebut, struktur proyek pada *front-end* ini dibagi menjadi beberapa lapisan fungsional utama sebagai berikut:

1. Lapisan pertama bertindak sebagai gerbang masuk utama aplikasi yang mengendalikan proses inisialisasi awal sistem, termasuk sinkronisasi awal dengan layanan *firebase*. Bagian ini bertanggung jawab penuh dalam mengatur rute peralihan halaman serta struktur navigasi menu utama seperti halaman pemantauan, direktori penyakit, dan data historis guna memastikan pengalaman pengguna yang intuitif dan responsif.
2. Lapisan kedua berfokus pada penyediaan fitur interaksi langsung antara pengguna dengan perangkat edge computing di lapangan. Komponen di dalam kluster ini mengakomodasi fungsi pemantauan video secara real time, eksekusi tombol perintah akuisisi citra, serta pengiriman berkas gambar menuju *endpoint backend*. Selain itu, lapisan ini menyediakan menu pengaturan khusus bagi pengguna untuk menyelaraskan alamat protokol internet serta port jaringan perangkat raspberry pi secara fleksibel di lapangan.
3. Lapisan ketiga ini mengintegrasikan fungsi edukasi dan analisis data historis bagi pembudidaya. Sistem menyajikan basis data komprehensif mengenai

jenis-jenis penyakit udang seperti WSSV, TSV, dan IMNV yang dilengkapi dengan rincian gejala klinis, faktor penyebab, tingkat keparahan, hingga panduan mitigasi otomatis. Lapisan mengelola penarikan ringkasan harian data deteksi dari server lokal, dengan fitur penyaringan berbasis jenis penyakit serta penataan halaman untuk menjaga efisiensi memori aplikasi.

4. Lapisan terakhir ini mengombinasikan fungsionalitas latar belakang aplikasi dengan standardisasi elemen visual. Fungsi layanan difokuskan pada manajemen komunikasi awan untuk mengurus perizinan perangkat, pendaftaran token digital, serta penanganan lalu lintas notifikasi *push* dari *firebase*.

3.6 Pengembangan Model CNN

3.6.1 Tahap Persiapan Dataset

Dataset citra dikumpulkan melalui kombinasi akuisisi langsung menggunakan prototipe dan pengumpulan dari sumber literatur. Proses anotasi dilakukan menggunakan perangkat lunak untuk memberikan label kelas pada setiap citra. Guna mengatasi keterbatasan jumlah data dan meningkatkan kemampuan generalisasi model, dilakukan augmentasi data yang mencakup teknik-teknik berikut:

1. Rotasi citra sebesar 90°, 180°, dan 270°.
2. Flipping horizontal dan vertikal.
3. Penyesuaian kecerahan dan kontras untuk mensimulasikan variasi kondisi kecerahan air tambak yakni 30–100 cm.

Dataset yang telah dianotasi dan diaugmentasi kemudian dibagi menjadi tiga subset: data latih sebesar 70%, data validasi sebesar 15%, dan data uji sebesar 15%. Pembagian ini dilakukan secara *stratified* untuk memastikan distribusi kelas yang proporsional pada setiap subset.

3.6.2 Pelatihan Model

Pelatihan model YOLO berbasis CNN dilakukan menggunakan *framework pytorch* pada *platform* raspberry pi 5 dengan akselerasi Hailo 8L. *Hyperparameter* pelatihan yang digunakan mencakup jumlah *epoch* sebesar 100, ukuran *batch* sebesar 16, dan laju pembelajaran awal sebesar 0,001 dengan jadwal penurunan

cosine annealing. Model dilatih dengan transfer learning dari bobot *pretrained* YOLOv11 yang telah dilatih pada *dataset* COCO. Proses pelatihan dipantau melalui kurva *loss* serta metrik akurasi pada setiap *epoch*. Pelatihan dihentikan secara *early stopping* apabila nilai *validation loss* tidak mengalami perbaikan selama 15 *epoch* berturut-turut, guna mencegah *overfitting*.

3.6.3 Evaluasi Kinerja Model

Kinerja model CNN dievaluasi menggunakan beberapa metrik standar dalam deteksi objek dan klasifikasi citra. Evaluasi dilakukan pada *test set* yang tidak pernah digunakan selama proses pelatihan maupun validasi, untuk memperoleh gambaran kinerja yang objektif. Metrik evaluasi yang digunakan adalah sebagai berikut:

Tabel 3. 4 Metrik Evaluasi Kinerja Model CNN

Metrik	**Rumus**	**Keterangan**
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Proporsi prediksi yang benar dari seluruh prediksi.
Precision	$\frac{TP}{TP + FP} \times 100\%$	Proporsi deteksi positif yang benar-benar positif.
Recall	$\frac{TP}{TP + FN} \times 100\%$	Proporsi kasus positif yang berhasil terdeteksi.
F1-Score	$\frac{2 \times Precision \times Recall}{Precision + Recall} \times 100\%$	Rata-rata harmonik *precision* dan *recall.*
mAP@0.5	*Mean Average Precision* pada IoU ≥ 0,5	Metrik utama evaluasi deteksi objek pada YOLO.

Confusion matrix divisualisasikan untuk setiap kelas penyakit guna mengidentifikasi pola kesalahan klasifikasi. Nilai mAP@0.5 dijadikan metrik utama pengambilan keputusan karena secara komprehensif menggambarkan kemampuan model dalam mendeteksi objek di berbagai kondisi *threshold.*

3.7 Pengujian Prototipe

3.7.1 Pengujian Komponen Perangkat Keras

Pengujian komponen perangkat keras dilakukan secara bertahap untuk

memastikan setiap komponen berfungsi sebagaimana yang dirancang sebelum dilakukan integrasi sistem secara keseluruhan. Setiap pengujian dilakukan minimal tiga kali pengulangan untuk memperoleh data yang representatif.

Tabel 3. 5 Rencana Pengujian Komponen Hardware

Komponen Diuji	Parameter Pengujian	Metode Pengujian	Kriteria Keberhasilan
HQ Camera & Housing Kedap Air	Kualitas citra pada kecerahan air 30–100 cm	Akuisisi citra di akuarium uji	Objek udang teridentifikasi jelas
Raspberry Pi 5 + Hailo 8L	Throughput inferensi (FPS), penggunaan CPU dan RAM	Profiling sistem saat inferensi YOLO	FPS ≥ 10, CPU < 80%
Sistem Kelistrikan	Stabilitas tegangan suplai (4,8–5,2 V)	Pengukuran multimeter	Tegangan stabil ±0,2 V

Tabel 3.4 di atas merangkum rencana pengujian untuk tiga komponen utama prototipe, yaitu kamera beserta *housing* kedap air, raspberry pi 5 yang dilengkapi akselerator Hailo 8L, serta sistem kelistrikan. Pengujian HQ *camera* difokuskan pada kemampuan menghasilkan citra yang memadai pada variasi kecerahan air tambak antara 30 hingga 100 cm, sebagai simulasi kondisi lingkungan yang sesungguhnya. Pengujian raspberry pi 5 mengevaluasi kemampuan komputasi sistem dalam mempertahankan *throughput* inferensi yang memadai dengan penggunaan sumber daya yang efisien. Pengujian sistem kelistrikan memastikan kestabilan tegangan suplai pada rentang operasional yang telah ditetapkan. Seluruh pengujian komponen dilakukan minimal tiga kali pengulangan guna memperoleh data yang representatif sebelum sistem diintegrasikan secara menyeluruh.

3.7.2 Pengujian Integrasi dan Kinerja Sistem

Pengujian integrasi dilakukan untuk mengevaluasi kinerja sistem secara keseluruhan dalam skenario yang mendekati kondisi operasional. Pengujian menggunakan sampel udang vaname usia 40 hari (n = 130 ekor) di akuarium

laboratorium dengan kondisi kecerahan air 40 cm. Sistem dinilai berdasarkan kemampuan deteksi per kelas penyakit yang direpresentasikan sebagai nilai akurasi deteksi. Prosedur pengujian mengacu pada standar evaluasi yang digunakan pada penelitian [6], sistem berhasil apabila mencapai akurasi deteksi di atas 60% pada kondisi pengujian standar laboratorium.

BAB IV
PEMBUATAN ALAT

Dalam penyusunan tugas akhir proses implementasi dan realisasi fisik dari sistem yang telah dirancang. Pembuatan alat mencakup tiga aspek utama, yaitu pembuatan perangkat keras, pembuatan perangkat lunak, dan pembuatan algoritma sistem deteksi dini penyakit.

4.1 Pembuatan Perangkat Keras

4.1.1 Pembuatan Komponen Mekanik

4.1.1.1 Alat dan Bahan

Sebelum proses fabrikasi dimulai, seluruh komponen dan peralatan yang dibutuhkan dikumpulkan dan diverifikasi kesesuaiannya dengan spesifikasi perancangan. Alat dan bahan yang digunakan dalam proses pembuatan prototipe dirangkum dalam Tabel 4-1 dan Tabel 4-2 berikut.

Tabel 4. 1 Alat yang Digunakan dalam Pembuatan Komponen Mekanik

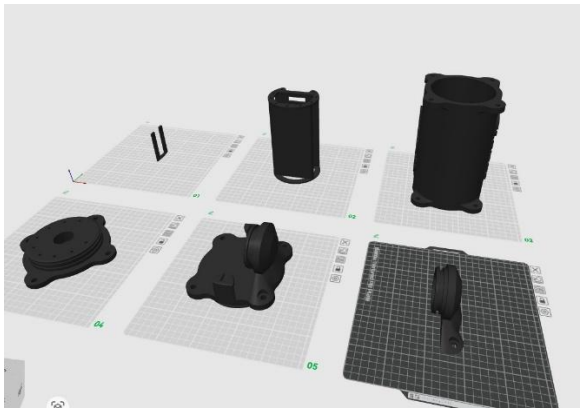
No.	Nama Alat	Jumlah
1.	3D printer	1 buah
2.	Laptop	1 buah
3.	Tang potong	1 buah
4.	Kunci pas	1 buah
5.	Kuas cat	1 buah
6.	Bor	1 buah
7.	Gelas takar	1 buah

Tabel 4. 2 Bahan yang Digunakan
dalam Pembuatan Komponen Mekanik

No.	Nama Alat	Jumlah
1.	Filamen PLA	985 gr
2.	Amplas	1 buah
3.	O-ring Seal	1 buah
4.	Seal tape	1 buah
5.	Resin epoxy	750 gr
6.	Kuas cat	1 buah
7.	Lem dextone	1 buah
8.	Dome	1 buah
9.	Mur	4 buah

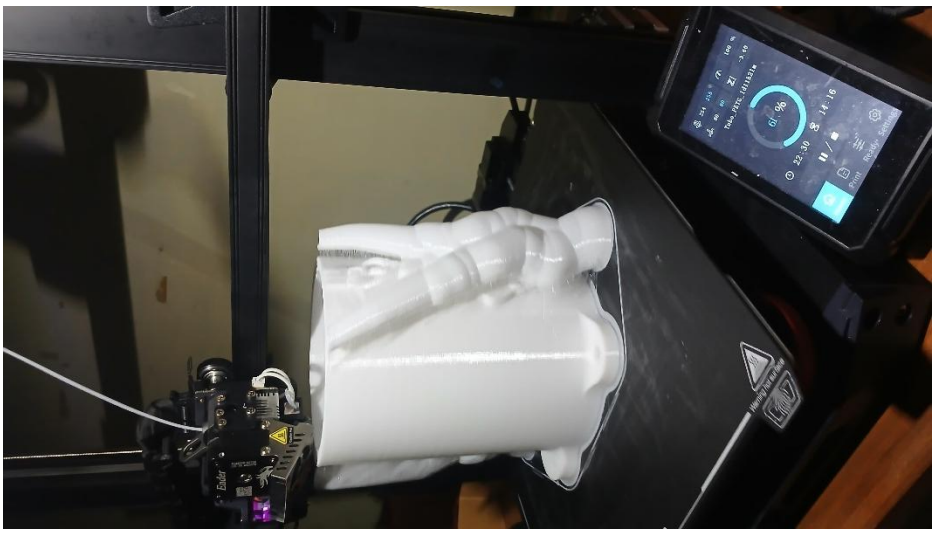
4.1.1.2 Proses Fabrikasi Komponen Utama

Proses fabrikasi fisik komponen housing prototipe dilakukan dengan menggunakan teknologi 3D Printing. Bahan dasar yang digunakan dalam proses pencetakan ini adalah filamen *PolyLactic Acid* (PLA) dengan total estimasi massa penyerapan material sebesar 985 gram. Proses pencetakan dilakukan secara bertahap untuk setiap bagian struktur, meliputi komponen *main housing*, *end cap*, dan *lamp holder*. Proses fabrikasi pada bagian-bagian mekanik ini dimulai dari tahap persiapan model 3D hasil perancangan yang akan diekstraksi ke dalam mesin cetak. Sebelum memasuki tahap produksi fisik, wujud rancangan geometri dari masing-masing komponen ditunjukkan pada Gambar 4.1 dan 4.2.



Gambar 4. 1 Proses Cetak 3D

Setelah visualisasi perancangan diverifikasi, model tersebut diproses menggunakan perangkat lunak slicing untuk mengatur parameter cetak, seperti ketebalan dinding dan kerapatan struktur bagian dalam. Proses pencetakan secara riil menggunakan mesin 3D Printer untuk merealisasikan komponen fisik ditunjukkan pada Gambar 4.3.



Gambar 4. 2 Proses Fabrikasi

Secara keseluruhan, durasi total yang diperlukan untuk memfabrikasi seluruh elemen *housing* meliputi *main housing*, *end cap*, dan *lamp holder* menggunakan mesin 3D Printer adalah sekitar 48 jam.

4.1.1.3 Proses Pascapemrosesan

Setelah seluruh komponen utama selesai dicetak, langkah berikutnya adalah fase pascapemrosesan untuk memperbaiki kualitas permukaan dan meningkatkan

kekuatan struktur mekanik. Karakteristik hasil cetak 3D printing umumnya memiliki celah mikroskopis antar lapisan, yang berpotensi memicu kebocoran air. Oleh karena itu, dilakukan proses pengamplasan permukaan luar housing terlebih dahulu menggunakan kertas amplas secara bertahap dari tekstur kasar hingga halus untuk meratakan permukaan.

Langkah berikutnya adalah pelapisan menggunakan cairan resin epoxy dengan total massa 750 gram. Cairan resin diaduk di dalam gelas takar dengan komposisi perbandingan yang sesuai antara resin dan *hardener*, kemudian diaplikasikan ke seluruh permukaan luar dan dalam komponen PLA menggunakan kuas cat. Proses pelapisan resin ini berfungsi ganda: pertama, sebagai zat pengikat penutup celah antar-lapisan struktur guna menjamin karakteristik kedap air kedua, untuk menambah rigiditas dan kekokohan struktural komponen mekanik agar tahan terhadap tekanan hidrostatik di dalam air tambak.

Gambar 4. 3 Proses Pelapisan Resin Epoxy

Berdasarkan dokumentasi pada Gambar 4.3, pengaplikasian resin epoxy dilakukan secara merata ke setiap sudut kritis housing, terutama pada area sambungan dan lekukan geometri yang memiliki risiko keretakan atau kebocoran tertinggi. Setelah proses pelapisan selesai, komponen memasuki fase pengeringan pada suhu ruang selama sekitar 36 jam hingga cairan resin terpolimerisasi sepenuhnya dan membentuk lapisan pelindung yang keras, mengkilap, dan solid.

4.1.1.4 Perakitan Mekanik

Struktur kubah transparan disatukan pada bagian depan *housing* menggunakan lem Dextone untuk memastikan sambungan yang permanen dan kuat. Komponen internal, termasuk elemen dudukan perangkat keras, dimasukkan ke dalam kompartemen utama sebelum akhirnya ditutup menggunakan komponen *end cap* pada sisi belakang. Sistem penguncian akhir menggunakan kombinasi 4 buah mur dan baut yang dikencangkan secara silang menggunakan kunci pas untuk meratakan tekanan mekanis pada permukaan perimeter sirkular. Guna mencapai spesifikasi kedap air yang andal, komponen *o-ring seal* dipasang secara presisi pada celah sambungan utama dan diperkuat dengan lilitan *seal tape* pada bagian ulir mekanis. Proses perakitan elemen-elemen mekanik ini ditunjukkan pada gambar 4.4, sedangkan wujud utuh komparasi fisik keseluruhan housing setelah selesai dirakit diperlihatkan pada gambar 4.5.

Gambar 4. 4 Proses Perakitan Elemen Mekanik dan Sistem Penguncian

Gambar 4. 5 Hasil Akhir Perakitan Housing Prototipe

4.1.2 Pembuatan Komponen Elektrik

4.1.2.1 Alat dan Bahan

Proses perakitan komponen elektrik melibatkan integrasi seluruh perangkat keras ke dalam housing prototipe yang telah selesai diproduksi. Implementasi rangkaian elektrik ini dilakukan berdasarkan perancangan sebelumnya, dengan menempatkan Raspberry Pi 5 sebagai unit komputasi pusat, yang mengoordinasikan seluruh subsistem.

Daftar alat dan bahan yang digunakan dalam proses pembuatan serta perakitan komponen elektrik ini masing-masing dirinci pada tabel 4.3 dan tabel 4.4.

Tabel 4. 3 Alat yang Digunakan dalam Pembuatan Komponen Elektrik

No.	Nama Alat	Jumlah
1.	Laptop	1 buah
2.	Obeng	1 buah
3.	Solder	1 buah
4.	Tang potong	1 buah
5.	Multimeter	1 buah
6.	Tenol	1 buah

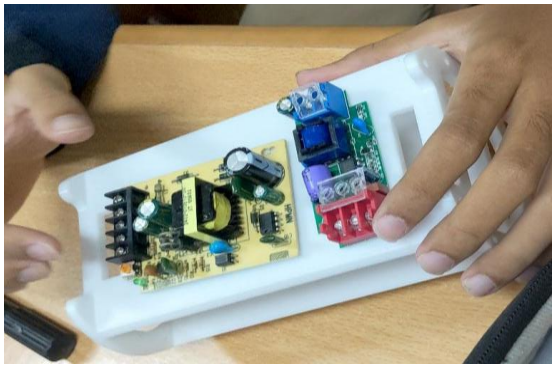
Tabel 4. 4 Bahan yang Digunakan dalam Pembuatan Komponen Elektrik

No.	Nama Alat	Jumlah
1.	Raspberry Pi 5	1 buah
2.	Hailo 8L NPU	1 buah
3.	HQ Camera IMX477	1 buah
4.	Lensa CS-Mount 5 MP	1 buah

No.	Nama Alat	Jumlah
5.	Power Supply 5V	1 buah
6.	Power Supply 12 V	1 buah
7.	LED Eagle Eye	1 buah
8.	Sambungan kabel	1 buah
9.	Kabel jumper	1 buah
10.	Kabel CSI	1 buah

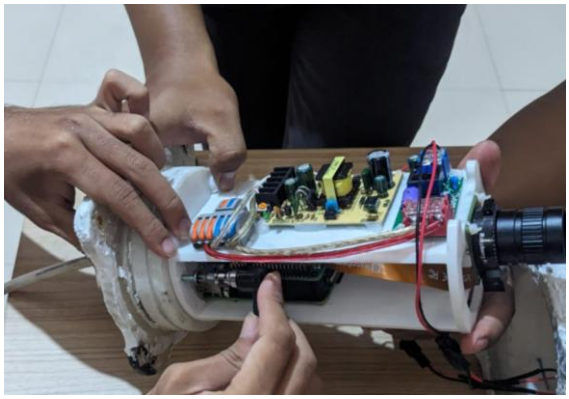
3.1.2.2 Perakitan Perangkat Elektrik

Pembuatan perangkat elektrik meliputi penyambungan dan pengujian komponen elektronik sesuai konfigurasi pada *wiring diagram* yang telah dirancang. Sebelum tahap pengabelan, catu daya dan modul konverter dipasang pada dudukan internal prototipe sesuai tata letak yang telah ditentukan. Penempatan komponen regulator daya ditunjukkan pada Gambar 4.7.



Gambar 4. 6 Penempatan Komponen Catu Daya pada *Housing* Prototipe

Perakitan sistem diawali dengan integrasi catu daya sebagai sumber tegangan DC. Keluaran catu daya didistribusikan ke jalur 12 V untuk lampu LED dan jalur 5 V untuk Raspberry Pi 5. Modul Hailo-8L diintegrasikan melalui antarmuka M.2 HAT+, sedangkan kamera HQ IMX477 dihubungkan melalui antarmuka MIPI CSI. Hasil perakitan dan integrasi seluruh komponen ditunjukkan pada Gambar 4.8.



Gambar 4. 7 Hasil Perakitan Fisik Komponen Elektrik Prototipe

4.2 Pembuatan Perangkat Lunak

4.2.1 Pembuatan Sistem Notifikasi

4.2.1.1 Arsitektur Notifikasi Firebase

Arsitektur notifikasi mengadopsi model *server-initiated push*, yaitu model di mana server berperan sebagai producer yang secara aktif menginisiasi pengiriman notifikasi ketika kondisi tertentu terpenuhi. *Back-end* yang berjalan pada raspberry pi 5 bertindak sebagai *producer* notifikasi yang mengirimkan pesan ketika model deteksi berhasil mengidentifikasi adanya penyakit pada sampel udang. Gambaran arsitektur lengkap sistem notifikasi disajikan pada Tabel 4.10 berikut

Tabel 4. 5 Komponen Arsitektur Notifikasi Firebase

Komponen	Peran	Keterangan
Back-end FastAPI (RPi 5)	*Producer* notifikasi	Menginisiasi pengiriman pesan FCM saat penyakit terdeteksi
Firebase Admin SDK	Lapisan abstraksi FCM	Memformat dan mengirimkan pesan ke server FCM Google
Server FCM Google	Infrastruktur pengiriman	Meneruskan notifikasi ke perangkat target berdasarkan token FCM
Firebase SDK	Penerimaan notifikasi	Menerima dan memproses notifikasi di sisi klien flutter
Firebase	Penyimpanan token	Menyimpan token FCM pengguna

Realtime DB	FCM	dan data deteksi secara persisten
Android / iOS OS	Tampilan notifikasi	Menampilkan notifikasi di status bar dan *notification drawer*

Berdasarkan Tabel 4.5, sistem notifikasi melibatkan enam komponen utama yang bekerja secara sinergis. Setiap komponen memiliki tanggung jawab yang spesifik dan terdefinisi dengan jelas dalam arsitektur sistem, sehingga kegagalan pada salah satu komponen dapat diidentifikasi dan ditangani secara terisolasi tanpa mengganggu komponen lainnya.

4.2.1.2 Alur Pengiriman Notifikasi End-to-End

Proses pengiriman notifikasi *push* dari sistem kepada *smartphone* melibatkan sembilan langkah yang berjalan secara berurutan dan melibatkan beberapa komponen yang berbeda. berikut ini menyajikan penjelasan rinci dari setiap langkah dalam alur pengiriman notifikasi

Tabel 4. 6 Alur End-to-End Pengiriman *Push Notification* FCM

NO	Proses	Protokol
1.	Aplikasi Flutter terinstal pada gawai pembudidaya dan meminta izin notifikasi dari pengguna	*Firebase* SDK
2.	Firebase SDK secara otomatis menghasilkan token FCM unik yang merepresentasikan perangkat tersebut	*Firebase Registration*
3.	Token FCM dikirimkan ke *back-end* melalui *endpoint* POST /register-token dan disimpan ke *firebase realtime database*	HTTPS REST API
4.	Prototipe mendeteksi adanya penyakit dan *back-end* memanggil fungsi send_fcm_notification()	Internal function call

NO	Proses	Protokol
5.	Firebase Admin SDK memformat pesan FCM dengan *payload* notifikasi visual dan data payload terstruktur	Firebase Admin SDK
6.	Pesan FCM yang telah diformat dikirimkan ke server FCM Google melalui koneksi HTTPS terenkripsi	HTTPS / TLS 1.3
7.	Server FCM Google meneruskan notifikasi ke perangkat target berdasarkan token FCM yang terdaftar	WebSocket / QUIC
8.	Sistem operasi Android atau iOS pada gawai pembudidaya menerima dan menampilkan notifikasi *push*	FCM SDK / APNs
9.	Aplikasi flutter memproses data *payload* yang diterima dan menavigasi pengguna ke layar	Flutter FirebaseMessaging

Pada Tabel 4.6, proses registrasi token pada langkah pertama hingga ketiga hanya dilakukan sekali saat aplikasi pertama kali dijalankan atau ketika token FCM mengalami pembaruan otomatis oleh *firebase* SDK. Token yang tersimpan di firebase *realtime database* kemudian digunakan secara berulang untuk setiap notifikasi berikutnya tanpa perlu melakukan registrasi ulang. Proses pengiriman notifikasi pada langkah keempat hingga kedelapan berjalan secara asinkron di latar belakang sebagai *background task* FastAPI, sehingga tidak menghambat pengiriman respon deteksi ke aplikasi flutter.

Waktu latensi rata-rata dari saat penyakit terdeteksi hingga notifikasi muncul di smartphone pembudidaya adalah sekitar 0,8 detik, yang dianggap cukup cepat untuk kebutuhan pemantauan kondisi tambak secara real-time. Latensi ini terdiri atas waktu pemrosesan inferensi (23–30 milidetik), waktu komunikasi HTTPS antara Raspberry Pi 5 dan server FCM Google (200–400 milidetik), waktu

penerusan notifikasi oleh infrastruktur FCM Google ke perangkat Android/iOS (100–300 milidetik), serta waktu rendering notifikasi oleh sistem operasi (50–100 milidetik). Implementasi fungsi send_fcm_notification() pada berkas firebase_service.py ditampilkan dalam blok kode berikut.

```
# firebase_service.py — Fungsi pengiriman notifikasi FCM
async def send_fcm_notification(fcm_token: str, record: dict):
    try:
        disease = record.get('disease', 'Tidak Diketahui')
        confidence = record.get('confidence', 0.0)
        pond_id = record.get('pond_id', 'Kolam-01')
        farm_name = record.get('farm_name', 'Tambak')
        mitigation = record.get('mitigation', [])
        # Bangun objek pesan FCM
        message = messaging.Message(
            notification=messaging.Notification(
                title=f'\u26a0\ufe0f PERINGATAN: {disease}
Terdeteksi!',
                body=f'{farm_name} > {pond_id} | {confidence*100:.1f}%
kepercayaan',
            ),
            data={
                'disease': disease,
                'confidence': str(confidence),
                'pond_id': pond_id,
                'farm_name': farm_name,
                'timestamp': record.get('timestamp',''),
                'is_sick': '1' if record.get('detected') else '0',
                'mitigation': '|'.join(mitigation),
            },
            token=fcm_token,
        )
        response = messaging.send(message)
        logger.info(f'FCM terkirim: {response}')
    except Exception as e:
        logger.error(f'Gagal kirim FCM: {e}')
```

Gambar 4. 8 Implementasi Fungsi send_fcm_notification() pada firebase_service.py

Pada Gambar 4.9, ditampilkan implementasi lengkap fungsi pengiriman notifikasi FCM. Blok *try-except* digunakan untuk memastikan bahwa kegagalan dalam pengiriman notifikasi tidak menyebabkan crash pada keseluruhan proses deteksi. Objek messaging terdiri atas dua bagian utama yakni *notification* yang berisi judul dan isi notifikasi yang akan ditampilkan secara visual di status bar gawai, serta kamus data yang berisi informasi terstruktur yang akan diteruskan ke aplikasi flutter untuk pemrosesan lebih lanjut. Bidang token menyimpan token

FCM perangkat tujuan yang telah diregistrasikan sebelumnya. Fungsi messaging.send() mengirimkan pesan ke server FCM google dan mengembalikan string ID pesan apabila pengiriman berhasil.

4.2.1.3 Format Payload Notifikasi FCM

Setiap notifikasi yang dikirimkan oleh sistem melalui FCM terdiri dari dua komponen *payload* yang memiliki fungsi berbeda namun saling melengkapi. Komponen pertama adalah *notification payload* yang bertanggung jawab untuk menampilkan notifikasi secara visual kepada pengguna melalui system notification drawer pada smartphone. Komponen kedua adalah data *payload* yang berisi informasi terstruktur yang diproses secara programatik oleh aplikasi Flutter untuk menentukan tindakan navigasi yang tepat.

Notification payload dirancang agar dapat langsung dipahami oleh pembudidaya tanpa perlu membuka aplikasi terlebih dahulu. Judul notifikasi memuat nama penyakit yang terdeteksi beserta simbol peringatan, sementara isi notifikasi memuat informasi singkat mengenai lokasi kolam, tingkat kepercayaan model, dan tindakan yang perlu segera dilakukan. Berikut adalah format payload notifikasi yang dikirimkan oleh firebase admin SDK python ke server FCM

```
// notification payload — ditampilkan sebagai notifikasi visual di
status bar
"notification": {
  "title": "\u26a0\ufe0f PERINGATAN: WSSV Terdeteksi!",
  "body":  "Tambak Utama > Kolam-01 | 87.4% kepercayaan | Segera panen
dini!"
}

// data payload — diproses secara programatik oleh NotificationService
Flutter
"data": {
  "disease":    "WSSV",
  "confidence": "0.8741",
  "pond_id":    "Kolam-01",
  "farm_name":  "Tambak Utama",
  "timestamp":  "2026-05-25T14:32:01",
  "is_sick":    "1",
  "mitigation": "KRITIS...|Pasang jaring...|Desinfeksi..."
}
```

Gambar 4. 9 Format Payload Notifikasi FCM

Pada Gambar 4.10, ditampilkan struktur lengkap payload notifikasi FCM yang dikirimkan oleh sistem. *Notification payload* terdiri atas dua bidang yakni *title* yang memuat judul notifikasi berupa nama penyakit dengan ikon peringatan, dan body yang memuat isi notifikasi berupa ringkasan informasi penting yang dapat langsung dibaca oleh pembudidaya tanpa membuka aplikasi.

Data *payload* memuat bidang informasi yang bersifat lebih teknis dan digunakan oleh kode aplikasi flutter untuk pengambilan keputusan. Bidang *disease* dan *confidence* memungkinkan aplikasi flutter untuk menentukan warna latar belakang kartu notifikasi dan ikon yang ditampilkan. Bidang pond_id dan farm_name digunakan untuk menavigasi pengguna secara otomatis ke layar pemantauan kolam yang sesuai saat notifikasi disentuh. Bidang is_sick bernilai '1'

apabila penyakit terdeteksi dan '0' apabila udang dalam kondisi sehat, digunakan sebagai penanda cepat tanpa perlu membandingkan nama penyakit. Bidang mitigation memuat daftar rekomendasi tindakan yang dipisahkan oleh karakter pipa (|) untuk kemudahan parsing di sisi klien menggunakan metode split().

4.2.1.4 Struktur *Firebase Realtime Database*

Firebase realtime database digunakan sebagai media penyimpanan data deteksi yang dapat diakses secara *real-time* dari berbagai perangkat. Struktur basis data dirancang secara hierarkis dengan tiga *node* utama, yaitu *detections* yang menyimpan riwayat deteksi per kolam, Pengguna yang menyimpan informasi dan token FCM pengguna terdaftar, serta *system* yang menyimpan status operasional sistem secara keseluruhan.


```
"detections": {
    "Kolam-01": {
      "-NxABC123def": {
        "disease":            "WSSV",
        "confidence":          0.8741,
        "timestamp":          "2026-05-25T14:32:01",
        "processing_time_ms": 23.4,
        "farm_name":          "Tambak Utama",
        "pond_id":            "Kolam-01",
        "server_timestamp":   1748176321000
      }
    }
  },
  "users": {
    "user_001": {
      "fcm_token":  "dWxxx...token...",
      "pond_ids":   ["Kolam-01", "Kolam-02"],
      "platform":   "android",
      "registered": "2026-05-25T10:00:00"
    }
  },
  "system": {
    "status": {
      "hailo_available": true,
      "camera_active":   true,
      "last_updated":    "2026-05-25T14:32:05"
    }
  }
}
```


Gambar 4. 10 Struktur Hierarkis *Firebase Realtime Database*

Pada gambar 4.11, ditampilkan struktur lengkap *firebase realtime database* yang digunakan oleh sistem. *Node detections* diorganisasikan berdasarkan

identifikasi kolam (pond_id) sebagai kunci tingkat pertama. Struktur hierarkis ini memungkinkan query data deteksi untuk kolam tertentu dengan efisien tanpa perlu memuat seluruh isi basis data. Kunci tingkat kedua berupa string yang dimulai dengan tanda minus, misalnya '-NxABC123def', merupakan kunci otomatis yang dihasilkan oleh Firebase secara kronologis dan unik secara global.

Setiap rekord deteksi menyimpan tujuh bidang data yang mencakup nama penyakit, *confidence*, waktu deteksi dalam format ISO 8601, durasi pemrosesan inferensi, nama tambak, identifikasi kolam , dan stempel waktu server dalam format *unix timestamp* milidetik. Bidang server_timestamp disimpan dalam format numerik milidetik unix untuk memudahkan operasi pengurutan dan pemfilteran berbasis rentang waktu dari sisi klien.

Node users menyimpan relasi antara identifikasi pengguna dengan token FCM dan daftar kolam yang dipantau, sehingga memungkinkan sistem untuk mengirimkan notifikasi hanya kepada pengguna yang memiliki kepentingan terhadap kolam yang bersangkutan. Kolom pond_ids merupakan larik yang memuat daftar identifikasi kolam yang dipantau oleh pengguna tersebut, sehingga sistem dapat melakukan pengiriman notifikasi yang tertarget hanya kepada pengguna yang terdaftar sebagai pemilik atau pengelola kolam yang sedang dipantau. *Node system* diperbarui secara berkala oleh server untuk mencerminkan kondisi terkini dari komponen-komponen *hardware* yang kritikal, yaitu akselerator hailo, modul kamera, dan koneksi *firebase*.

4.2.2 Pembuatan Sistem *Front End*

4.2.2.1 Struktur Sistem *Front End*

Struktur sistem *front end* diorganisasikan dalam direktori pola arsitektur *service layer* yang diadopsi. Setiap kategori berkas ditempatkan dalam direktori yang sesuai dengan perannya dalam sistem, sehingga memudahkan navigasi dan pemeliharaan kode. Hierarki lengkap dari struktur ditampilkan dalam blok kode berikut:

```
sdides/lib/
├── main.dart                          # Entry point +
├── firebase_options.dart              # Konfigurasi Firebase
├── screens/
│   ├── monitor_screen.dart           # Live kamera + deteksi
│   ├── penyakit_screen.dart          # Database informasi penyakit
│   ├── riwayat_screen.dart           # Riwayat deteksi + filter
│   ├── settings_screen.dart          # Konfigurasi URL server
│   └── disease_detail_screen.dart    # Halaman detail informasi
├── services/
│   ├── api_service.dart              # HTTP client + model data
│   └── notification_service.dart     # FCM handler + local
└── widgets/
    ├── app_header.dart               # Widget header aplikasi
    └── stat_card.dart                # Widget kartu statistik
```

Gambar 4. 11 Struktur Direktori Front End

Pada gambar 4.12, direktori terdiri atas empat lapisan utama. Lapisan pertama adalah berkas-berkas di akar direktori lib/, yaitu main.dart dan firebase_options.dart. Berkas main.dart merupakan *entry point* aplikasi yang menginisialisasi seluruh layanan yang diperlukan, termasuk *firebase* dan *notificationservice*, sebelum aplikasi mulai merender antarmuka pengguna. Berkas firebase_options.dart merupakan berkas yang dihasilkan secara otomatis oleh alat flutterfire CLI dan berisi konfigurasi spesifik *platform* (Android dan iOS) untuk proyek *firebase* yang terhubung.

Lapisan kedua adalah direktori screens/ yang memuat lima berkas layar yang masing-masing merepresentasikan satu halaman dalam navigasi aplikasi. Berkas monitor_screen.dart merupakan layar utama yang menampilkan aliran video langsung dan menyediakan fungsi deteksi penyakit. Berkas penyakit_screen.dart menampilkan basis data informasi tentang berbagai jenis penyakit udang yang dapat diakses secara offline. Berkas riwayat_screen.dart menampilkan riwayat seluruh hasil deteksi dengan fitur filter dan statistik. Berkas settings_screen.dart menyediakan antarmuka konfigurasi koneksi server dan pengaturan kolam. Berkas disease_detail_screen.dart menampilkan informasi lengkap dan panduan penanganan untuk setiap jenis penyakit.

Lapisan ketiga adalah direktori services/ yang memuat dua kelas service yang merupakan inti dari pola arsitektur *service layer*. Berkas api_service.dart mengimplementasikan seluruh komunikasi HTTP dengan server fastAPI, termasuk pengiriman gambar untuk deteksi dan pengambilan riwayat deteksi. Berkas notification_service.dart mengimplementasikan penanganan notifikasi FCM untuk tiga skenario yang berbeda, yaitu saat aplikasi berjalan di latar depan, latar belakang, dan saat aplikasi tidak berjalan. Lapisan keempat adalah direktori widgets/ yang memuat dua widget yang dapat digunakan ulang di berbagai layar untuk menjaga konsistensi tampilan.

Tabel 4.7 berikut menyajikan penjelasan lengkap mengenai fungsi dan tanggung jawab dari setiap berkas

Tabel 4. 7 Penjelasan Berkas Front End

Berkas	Kategori	Fungsi Utama
main.dart	Entry point	Inisialisasi Firebase, NotificationService; konfigurasi tema Material 3; penentuan rute navigasi
firebase_options.dart	Konfigurasi	Menyimpan API keys, project ID, app ID, messaging sender ID per platform
monitor_screen.dart	Screen	Live MJPEG stream; tombol deteksi; dialog hasil deteksi; statistik hari ini
penyakit_screen.dart	Screen	Daftar penyakit udang; navigasi ke detail penyakit; data statis offline
riwayat_screen.dart	Screen	Tabel riwayat deteksi; filter penyakit/waktu; auto-refresh via StreamBuilder

Berkas	Kategori	Fungsi Utama
settings_screen.dart	Screen	Input URL server; ping koneksi; simpan pond_id dan farm_name ke SharedPreferences
disease_detail_screen.dart	Screen	Detail penyakit: deskripsi, gejala, penyebab, panduan darurat
api_service.dart	Service	HTTP client; model DetectionResult; StreamController broadcast; fetchHistory()
notification_service.dart	Service	Inisialisasi FCM; penanganan foreground/background/terminated; navigasi otomatis
app_header.dart	Widget	Header bar dengan logo, judul, dan indikator status koneksi server
stat_card.dart	Widget	Kartu statistik dengan label, nilai, dan warna yang dapat dikonfigurasi

4.2.2.2 Alur Komunikasi Data

Alur komunikasi data antara aplikasi dan server menggambarkan proses pertukaran data mulai dari permintaan deteksi oleh pengguna hingga pembaruan informasi pada antarmuka aplikasi. Proses komunikasi melibatkan beberapa tahapan yang mencakup pengiriman data, pemrosesan hasil deteksi, penyimpanan data, hingga pengiriman informasi kembali ke aplikasi. Rincian tahapan komunikasi data ditunjukkan pada tabel 4.8.

Tabel 4. 8 Alur Komunikasi Data

No	Aksi	Kode
1.	Pengguna menekan tombol 'Start Detection' pada MonitorScreen	GestureDetector.onTap
2.	Kamera mengambil satu frame gambar melalui CameraController	_cameraController.takePicture()
3.	ApiService membungkus gambar dalam permintaan HTTP multipart	ApiService.detectDisease()
4.	Permintaan multipart dikirimkan ke endpoint POST /detect	http.MultipartRequest.send()
5.	Server FastAPI menjalankan inferensi model dan mengembalikan JSON	FastAPI /detect endpoint
6.	Aplikasi Flutter melakukan parsing respons JSON ke model Dart	DetectionResult.fromJson()
7.	Antarmuka pengguna diperbarui: dialog hasil dan badge status	setState() + showDialog()
8.	StreamController menyiarkan data terbaru ke RiwayatScreen	ApiService.detectionStream
9.	Notifikasi FCM muncul di status bar (apabila penyakit terdeteksi)	NotificationService (Firebase)

Berdasarkan tabel 4.8, langkah pertama hingga kedelapan berlangsung secara berurutan mulai dari proses permintaan deteksi hingga pembaruan informasi pada antarmuka aplikasi. Penggunaan *streamcontroller broadcast* memungkinkan data hasil deteksi diterima oleh beberapa komponen aplikasi melalui satu aliran data. Proses pengiriman notifikasi dilakukan secara terpisah dari alur utama

sehingga tidak mempengaruhi proses pembaruan tampilan aplikasi.

4.2.3 Pembuatan Sistem *BackEnd*

4.2.3.1 Struktur Modul *BackEnd*

Struktur direktori *backend* sistem diimplementasikan secara modular dengan memisahkan setiap komponen berdasarkan fungsi masing-masing. Pemisahan struktur ini dilakukan untuk mengelompokkan modul konfigurasi, layanan API, model data, dan penyimpanan sistem sehingga proses pengembangan dan pengelolaan kode lebih terorganisasi. Hierarki direktori *backend* beserta fungsi setiap berkas ditunjukkan sebagai berikut.

```
backend/
├── main.py                  # Entry point FastAPI
├── firebase_service.py      # Firebase Admin SDK wrapper
├── requirements.txt         # Daftar dependensi Python
├── models/                  # Direktori file model AI
│   └── sdides_yolov11.hef   # Model YOLOv11 terformat Hailo
└── firebase/
    └── serviceAccountKey.json  # Kredensial Firebase Admin SDK
```

Gambar 4. 12 Struktur Direktori Back-End

Pada gambar 4.13, ditampilkan hierarki direktori backend yang terdiri atas beberapa komponen utama dalam pengelolaan sistem. Berkas *main.py* digunakan sebagai modul utama aplikasi fastAPI yang menangani inisialisasi sistem, konfigurasi layanan, serta pengaturan proses utama pada *backend*. Berkas *firebase_service.py* digunakan sebagai modul penghubung antara *backend* dengan layanan *firebase*. Modul menangani proses komunikasi dengan *firebase admin* SDK yang meliputi penyimpanan data ke *firebase realtime database* dan pengiriman notifikasi melalui *firebase cloud messaging* (FCM). Selanjutnya, berkas *requirements.txt* berisi daftar dependensi python yang digunakan untuk menjalankan seluruh layanan *backend*. Direktori *models* digunakan untuk menyimpan berkas model YOLOv11 dalam format *hailo executable format* (.hef) yang telah dikompilasi untuk dijalankan pada akselerator hailo-8L. Sementara itu, direktori *firebase* menyimpan berkas konfigurasi autentikasi yang digunakan untuk menghubungkan backend dengan layanan *firebase*.

4.2.3.2 Daftar *Endpoint* Back End

Sistem *endpoint* pada back end mengimplementasikan tujuh *endpoint* REST API yang digunakan untuk mendukung proses pertukaran data antara backend dan aplikasi. Setiap *endpoint* memiliki fungsi dan parameter yang disesuaikan dengan kebutuhan sistem, mulai dari pengiriman hasil deteksi, pengambilan data, hingga pengelolaan informasi pada aplikasi. Seluruh *endpoint* didefinisikan pada berkas *main.py* dan berjalan menggunakan protokol HTTP. Rincian implementasi setiap endpoint beserta fungsi dan parameter yang digunakan ditunjukkan pada Tabel 4.7.

Tabel 4. 9 Daftar Endpoint REST API

Endpoint	Metode	Fungsi	Parameter
POST /detect	POST	Deteksi penyakit dari gambar — endpoint utama sistem	image (file), fcm_token?, pond_id?, farm_name?
GET /stream	GET	MJPEG live stream kamera + overlay bounding box real-time	— (tidak ada)
GET /history	GET	Riwayat deteksi dengan filter waktu dan jenis penyakit	limit, hours, disease_filter?
GET /status	GET	Status sistem: hardware (Hailo, Firebase, kamera) + statistik	— (tidak ada)
POST /notify	POST	Kirim push notifikasi FCM manual (testing/debugging)	fcm_token, disease, confidence, pond_id, farm_name
POST /register-token	POST	Daftarkan token FCM pengguna ke database Firebase	user_id, token, pond_ids[], platform

GET /docs	GET	Swagger UI dokumentasi API otomatis	— (browser)

Berdasarkan tabel 4.9, *endpoint* POST */detect* digunakan sebagai bagian utama dalam proses deteksi objek pada sistem. *Endpoint* menerima data citra dari aplikasi flutter, menjalankan proses inferensi menggunakan model YOLOv11, kemudian mengembalikan hasil deteksi berupa informasi kelas, nilai *confidence*, dan rekomendasi tindakan dalam format JSON.

Endpoint GET */stream* digunakan untuk menyediakan tampilan aliran video secara langsung dari kamera raspberry pi 5 dengan penambahan visualisasi *bounding box* pada objek hasil deteksi. Selanjutnya, endpoint GET */history* digunakan untuk mengambil data riwayat deteksi yang tersimpan pada *firebase realtime database* dengan dukungan parameter penyaringan sesuai kebutuhan aplikasi. *Endpoint* GET */status* digunakan untuk menampilkan informasi kondisi sistem yang mencakup status kamera, koneksi *firebase*, dan ketersediaan akselerator Hailo-8L. *Endpoint* POST */notify* digunakan untuk melakukan pengujian pengiriman notifikasi FCM secara manual, sedangkan endpoint POST */register-token* digunakan untuk menyimpan token perangkat pengguna pada *firebase realtime database*. *Endpoint* GET */docs* menyediakan dokumentasi API berbasis swagger UI yang dibuat secara otomatis oleh FastAPI.

Implementasi *endpoint* POST */detect* sebagai salah satu layanan utama backend ditampilkan pada blok kode berikut. Kode tersebut menunjukkan struktur pendefinisian *endpoint* FastAPI beserta parameter masukan dan proses pengolahan data yang dilakukan.

4.2.3.3 *Format Request-Response Endpoint*

Endpoint /detect menerima permintaan HTTP POST dengan tipe konten *multipart/form-data*. Pengiriman data dilakukan dalam bentuk berkas gambar beserta data tambahan dari aplikasi flutter dalam satu permintaan menuju *backend*. Citra yang diterima selanjutnya diproses sebagai masukan pada tahap inferensi model YOLOv11 untuk menghasilkan informasi deteksi. Struktur permintaan yang dikirimkan oleh aplikasi flutter ke server ditampilkan sebagai berikut.

```
POST /detect HTTP/1.1
Host: 192.168.1.100:5000
Content-Type: multipart/form-data; boundary=boundary123

--boundary123
Content-Disposition: form-data; name="image"; filename="udang.jpg"
Content-Type: image/jpeg
[binary JPEG data]

--boundary123
Content-Disposition: form-data; name="fcm_token"
dWxxxxxxxxxxx_FCM_TOKEN_DISINI_xxxxxxxxxx
```

Gambar 4. 13 Format HTTP Request Multipart ke Endpoint

Pada gambar 4.14, ditampilkan format permintaan HTTP POST yang dikirimkan oleh aplikasi flutter ke endpoint */detect*. Permintaan tersebut memuat data citra yang akan digunakan sebagai masukan pada proses inferensi, serta informasi tambahan berupa *FCM token* dan identitas kolam (*pond_id*). Data citra digunakan untuk proses deteksi objek, sedangkan *FCM token* dan *pond_id* digunakan untuk mendukung proses pengiriman notifikasi dan penyimpanan hasil deteksi pada sistem.

Setelah proses inferensi selesai dilakukan, *backend* mengembalikan respon dalam format JSON yang memuat informasi hasil deteksi. Data yang dikembalikan mencakup hasil klasifikasi, nilai *confidence*, serta informasi lain yang diperlukan untuk ditampilkan pada aplikasi Flutter. Respon yang diterima aplikasi ditunjukkan sebagai berikut.

```
{
  "detection_id":  "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "timestamp":     "2026-05-25T14:32:01.123456",
  "detected":      true,
  "disease":       "WSSV",
  "confidence":    0.8741,
  "all_detections": [
    { "class_id":3, "class_name":"WSSV",
      "confidence":0.8741, "bbox":[120, 80, 200, 160] }
  ],
  "mitigation": [
    "KRITIS: Segera lakukan panen dini jika berat udang memungkinkan",
    "Pasang jaring penutup untuk mencegah masuknya vektor penular",
    "Lakukan desinfeksi air dengan klorin 30 ppm sebelum dibuang"
  ],
  "processing_time_ms": 23.4
}
```


Gambar 4. 14 Format *JSON Respons Endpoint*

Pada gambar 4.14, ditampilkan struktur respons JSON yang dikembalikan oleh server fastAPI setelah proses inferensi selesai dilakukan. Respons tersebut memuat informasi hasil deteksi yang selanjutnya digunakan oleh aplikasi flutter untuk menampilkan hasil klasifikasi beserta informasi pendukung pada antarmuka.

Data hasil deteksi diawali dengan bidang *detection_id* yang digunakan sebagai identitas pada setiap rekaman deteksi yang tersimpan dalam sistem. Bidang *timestamp* mencatat waktu proses deteksi dilakukan, sedangkan bidang *detected* menunjukkan status keberadaan objek penyakit pada citra yang dianalisis. Hasil klasifikasi model disimpan pada bidang *disease* berupa kelas penyakit yang terdeteksi atau kondisi *Sehat*, dengan tingkat keyakinan model terhadap hasil tersebut disimpan pada bidang *confidence*.

Selain hasil klasifikasi utama, respons juga memuat bidang *all_detections* yang berisi seluruh objek hasil deteksi pada citra, meliputi informasi kelas, nilai kepercayaan, serta koordinat *bounding box*. Data tersebut digunakan oleh aplikasi untuk menampilkan posisi objek terdeteksi pada tampilan citra. Bidang *mitigation* berisi rekomendasi tindakan berdasarkan hasil klasifikasi penyakit, sedangkan

bidang *processing_time_ms* mencatat waktu yang diperlukan sistem sejak citra diterima hingga hasil deteksi dikirimkan kembali.

4.3 Pembuatan Dataset dan Model CNN

4.3.1 Proses Anotasi dengan Roboflow

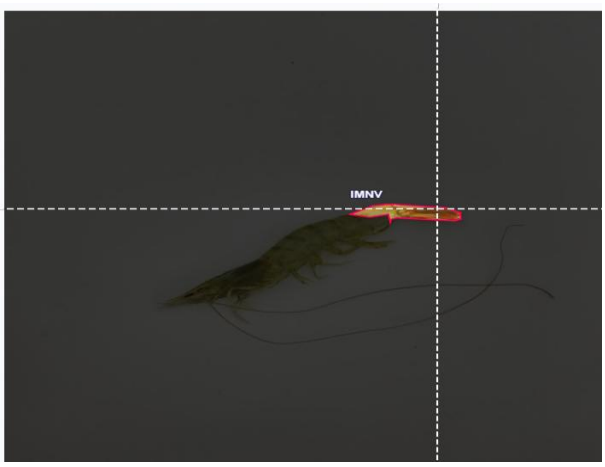
Proses anotasi dilakukan secara manual dengan berpedoman pada karakteristik klinis spesifik setiap kelas penyakit yang telah dikonfirmasi berdasarkan literatur ilmiah terkait patologi udang. Dataset citra yang digunakan pada tahap anotasi diperoleh dari dua sumber utama. Sumber pertama berasal dari proses akuisisi langsung menggunakan prototipe yang ditempatkan pada akuarium laboratorium dengan objek udang vaname berusia 40 hari. Proses pengambilan citra dilakukan menggunakan kamera HQ IMX477 dengan resolusi 1920×1080 piksel.

Sumber kedua diperoleh dari dokumentasi citra gejala klinis penyakit udang vaname pada literatur ilmiah yang telah melalui proses validasi diagnostik. Penggabungan kedua sumber data tersebut menghasilkan total 3.000 citra yang kemudian dikelompokkan ke dalam empat kelas berdasarkan kondisi udang yang diamati. Distribusi jumlah citra pada setiap kelas ditampilkan pada Tabel 4.10.

Tabel 4. 10 Distribusi Dataset Citra per Kelas

No.	Kelas / Kondisi Udang	Jumlah Citra	Persentase (%)
1	Sehat	870 citra	29,0%
2	IMNV	780 citra	26,0%
3	WSSV	750 citra	25,0%
4	TSV	600 citra	20,0%
Total		3.000 citra	100%

Penentuan area anotasi dilakukan berdasarkan karakteristik visual yang membedakan masing-masing kelas sehingga label yang dihasilkan dapat merepresentasikan kondisi objek pada citra. Pada kelas IMNV, anotasi dilakukan berdasarkan perubahan warna abnormal pada jaringan otot abdomen berupa area putih opak hingga kemerahan. Area tersebut digunakan sebagai acuan dalam menentukan batas *bounding box* pada citra. Dokumentasi hasil anotasi kelas IMNV ditunjukkan pada Gambar 4.16.



Gambar 4. 15 Area Anotasi pada IMNV

Berdasarkan Gambar 4.16, area *bounding box* ditempatkan pada bagian tubuh yang menunjukkan gejala nekrosis secara visual. Apabila perubahan hanya terlihat pada bagian tertentu, batas anotasi disesuaikan dengan area gejala yang tampak untuk menjaga ketepatan label dataset.

Pada kelas WSSV, anotasi dilakukan berdasarkan keberadaan bintik putih pada permukaan tubuh udang, terutama pada bagian karapas. Dokumentasi hasil anotasi kelas WSSV ditunjukkan pada Gambar 4.17.

Gambar 4. 16 Area Anotasi pada WSSV

Berdasarkan hasil anotasi pada Gambar 4.17, area *bounding box* mencakup bagian tubuh udang yang menunjukkan karakteristik gejala penyakit. Penentuan batas anotasi dilakukan berdasarkan area objek yang terlihat jelas pada citra karena gejala WSSV dapat muncul pada beberapa bagian permukaan tubuh. Kondisi TSV,

penentuan anotasi mengacu pada perubahan warna kemerahan pada tubuh udang, terutama bagian ekor dan anggota gerak. Dokumentasi hasil anotasi kelas TSV ditunjukkan pada gambar 4.18.

Gambar 4. 17 Area Anotasi pada TSV

Berdasarkan gambar 4.18, *bounding box* ditempatkan pada area tubuh yang mengalami perubahan warna maupun gejala nekrosis. Penyesuaian area anotasi dilakukan berdasarkan bagian yang menunjukkan perubahan visual akibat infeksi untuk mempertahankan konsistensi label pada dataset. Pada kelas sehat, proses anotasi dilakukan berdasarkan kondisi visual udang tanpa menunjukkan adanya gejala klinis penyakit. Area *bounding box* ditempatkan pada seluruh bagian tubuh udang sebagai representasi objek normal yang digunakan sebagai pembanding terhadap kelas yang mengalami infeksi. Dokumentasi hasil anotasi kelas Sehat ditunjukkan pada gambar 4.19.

Gambar 4. 18 Area Anotasi pada Kategori Sehat

Berdasarkan gambar 4.19, kelas sehat mencakup keseluruhan tubuh udang

dengan memastikan objek terlihat secara utuh pada citra. Data keleas sehat digunakan untuk membantu model mengenali karakteristik visual udang normal sehingga dapat membedakan kondisi sehat dengan udang yang menunjukkan gejala penyakit. Hasil akhir proses anotasi menghasilkan total 4.553 *bounding box* dari 3.000 citra dataset. Distribusi jumlah *bounding box* dan rata-rata objek pada setiap kelas ditampilkan pada tabel 4.11.

Tabel 4. 11 Distribusi Anotasi Bounding Box per Kelas

No.	Kelas Penyakit	Jumlah Bounding Box	Rata-rata Objek per Citra
1	Sehat	812	1,04
2	IMNV	896	1,24
3	TSV	1.052	1,40
4	WSSV	1.793	2,39
	Total	4.553	1,52

Berdasarkan tabel 4.11, kelas WSSV memiliki jumlah *bounding box* tertinggi, yaitu sebanyak 1.793 anotasi dengan rata-rata 2,39 objek per citra. Sementara itu, kelas Sehat memiliki jumlah anotasi terendah, yaitu sebanyak 812 *bounding box*. Perbedaan jumlah anotasi pada setiap kelas dipengaruhi oleh variasi jumlah objek yang muncul dalam satu citra serta karakteristik visual masing-masing kondisi udang.

Dataset yang telah melalui proses anotasi selanjutnya dipersiapkan untuk tahap pelatihan model melalui penerapan teknik augmentasi citra. Teknik augmentasi diterapkan menggunakan *pipeline* augmentasi roboflow yang dieksekusi secara otomatis pada saat ekspor dataset. Setelah seluruh augmentasi diterapkan, jumlah data *training* meningkat dari 2.100 citra menjadi 11.430 citra. Sementara itu, subset validasi dan pengujian tidak mengalami proses augmentasi sehingga tetap mempertahankan karakteristik data asli. Distribusi dataset setelah proses pembagian dan augmentasi ditampilkan pada tabel 4.12.

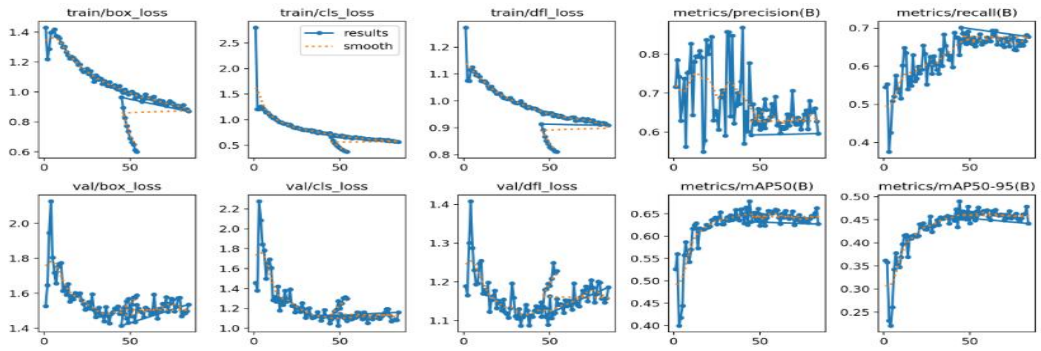
Tabel 4. 12 Pembagian Dataset Setelah Augmentasi

No.	Subset	Jumlah Citra Asli	Jumlah Citra Pasca-Augmentasi	Proporsi (%)
1	Training Set	2.100 citra	11.430 citra	75,5%
2	Validation Set	678 citra	678 citra tidak diaugmentasi	22,6%
3	Testing Set	222 citra	222 citra tidak diaugmentasi	7,4%
	Total	3.000 citra asli	12.330 citra total	100%

Berdasarkan tabel 4.12, proses augmentasi menyebabkan jumlah data *training* meningkat secara signifikan dari 2.100 citra menjadi 11.430 citra. Sementara itu, data validasi dan pengujian tidak mengalami augmentasi sehingga tetap merepresentasikan kondisi data asli. Pembagian dataset tersebut digunakan sebagai dasar pada proses pelatihan, validasi, dan pengujian model.

4.3.2 Pelatihan Model CNN

Selama proses pelatihan, sistem secara otomatis mencatat perkembangan nilai *loss* dan metrik evaluasi pada setiap *epoch*. Parameter yang diamati meliputi *train box loss*, *train classification loss*, *train distribution focal loss (DFL)*, *validation loss*, *precision*, *recall*, *mAP@50*, dan *mAP@50-95*.Hasil pemantauan selama proses pelatihan ditampilkan pada gambar 4.20.



Gambar 4. 19 Kurva Loss dan Metrik Evaluasi

Berdasarkan gambar 4.20, nilai *train box loss*, *train classification loss*, dan *train DFL loss* mengalami penurunan secara bertahap seiring bertambahnya *epoch* pelatihan. Penurunan pada *validation loss* menunjukkan model mampu

menyesuaikan parameter terhadap karakteristik dataset yang digunakan. Nilai *precision*, *recall*, *mAP@50*, dan *mAP@50-95* mengalami peningkatan hingga mencapai kondisi yang relatif stabil pada *epoch* akhir. Nilai *mAP@50* mencapai sekitar 0,65, sedangkan *mAP@50-95* mencapai sekitar 0,45 pada akhir pelatihan. Hasil evaluasi model ditampilkan pada gambar 4.21

Gambar 4. 20 Hasil Evaluasi Model

Berdasarkan gambar 4.21, model memperoleh nilai *mAP@50* sebesar 67,8%, *precision* sebesar 77,6%, *recall* sebesar 63,0%, dan *F1-score* sebesar 69,5%. Hasil tersebut menunjukkan model telah mampu melakukan deteksi dan klasifikasi penyakit udang pada dataset yang digunakan.

4.3.3 Konversi Model ke Format Hailo

Tahap awal konversi dilakukan dengan mengekspor model *best.pt* hasil pelatihan ke format ONNX menggunakan utilitas ekspor dari Ultralytics YOLOv11. Proses ekspor dilakukan dengan mengaktifkan parameter *simplify=True* untuk menyederhanakan grafik komputasi model sehingga lebih kompatibel dengan proses kompilasi pada Hailo DFC. Hasil ekspor berupa berkas model dengan format *.onnx* yang selanjutnya digunakan sebagai masukan pada tahap optimasi dan kompilasi model ke format *.hef*. Implementasi proses ekspor model YOLOv11n ke format ONNX ditunjukkan pada Gambar 4.22.

```
# ============================================
# EXPORT MODEL TO ONNX
# ============================================
print("\n" + "="*70)
print("📦 EXPORTING MODEL TO ONNX FORMAT")
print("="*70 + "\n")

try:
    onnx_path = best_model.export(format='onnx', simplify=True)
    print(f"☑ ONNX export successful: {onnx_path}\n")
except Exception as e:
    print(f"⚠  ONNX export failed: {e}\n")
    onnx_path = None
```

Gambar 4. 21 Koversi Fotmat ONNX\

Proses selanjutnya adalah kompilasi menggunakan *Hailo Dataflow Compiler* (DFC). Proses kompilasi membutuhkan data kalibrasi yang diambil dari subset dataset pelatihan. Data tersebut digunakan untuk menentukan parameter kuantisasi sehingga perubahan representasi model dari FP32 menjadi INT8 dapat dilakukan dengan tetap mempertahankan performa deteksi. Jumlah kelas pada proses kompilasi disesuaikan dengan dataset yang terdiri dari empat kelas, yaitu Sehat, IMNV, TSV, dan WSSV. Implementasi proses konversi model ONNX menjadi format HEF ditunjukkan pada gambar 4.23.

```
hailomz compile yolov11n \
--ckpt=best.onnx \
--hw-arch hailo8l \
--calib-path train/images \
--classes 4 \
--performance
```

Gambar 4. 22 Konversi ONNX ke hef

Berdasarkan gambar 4.23, proses kompilasi dilakukan dengan memasukkan model *best.onnx* sebagai input dan menentukan perangkat target menggunakan parameter *hailo8l*. Parameter *--calib-path* menunjuk lokasi citra kalibrasi yang digunakan selama proses optimasi, sedangkan parameter *--classes* disesuaikan dengan jumlah kelas deteksi pada model. Hasil akhir proses kompilasi berupa berkas *sdides_yolov11.hef* yang selanjutnya digunakan sebagai model inferensi pada raspberry pi 5

BAB V
HASIL DAN ANALISA

5.1 Pengujian Perangkat Keras

5.1.1 Pengujian *Housing*

Pengujian kekedapan dilakukan dengan cara menenggelamkan wadah mekanik kosong ke dalam media air uji dengan merendam *housing* pada kedalaman 30 cm selama 24 jam. Dokumentasi pelaksanaan pengujian kekedapan ditunjukkan pada gambar 5.1.

Gambar 5. 1 Proses Pengujian Housing

Berdasarkan gambar 5.1, proses pengujian housing menggunakan parameter pengujian meliputi kebocoran air dan kondensasi uap pada ruang interior. Hasil pengujian menunjukkan bahwa tidak terjadi rembesan air pada sambungan antar kompartemen setelah 24 jam perendaman. Penggunaan seal dan pelapisan resin epoksi pada permukaan PLA efektif mencegah penetrasi air. Ruang interior tetap kering tanpa indikasi kondensasi, sehingga komponen elektronik terlindungi dengan baik.

5.1.2 Pengujian Sistem Kelistrikan

Sistem kelistrikan pada prototipe menggunakan dua tingkat tegangan utama, yaitu 5 V DC dan 12 V DC. Tegangan 5 V DC digunakan sebagai suplai raspberry pi 5 beserta komponen pendukung pemrosesan, sedangkan tegangan 12 V DC digunakan untuk menyuplai lampu LED Eagle Eye sebagai sumber

pencahayaan tambahan pada proses akuisisi citra bawah air.

Pengujian sistem kelistrikan dilakukan dengan mengukur tegangan keluaran masing-masing catu daya menggunakan multimeter digital. Pengukuran dilakukan pada kondisi operasional penuh, yaitu ketika Raspberry Pi 5 menjalankan proses inferensi model, kamera HQ IMX477 aktif melakukan akuisisi citra, dan lampu LED berada pada kondisi intensitas maksimum. Dokumentasi pengujian sistem kelistrikan ditunjukkan pada gambar 5.2.

Gambar 5. 2 Proses Pengujan Sistem Kelistrikan

Parameter pengukuran meliputi tegangan suplai catu daya, kondisi aktivasi beban, serta fungsi perangkat pendukung selama sistem beroperasi. Hasil pengukuran parameter kelistrikan ditampilkan pada tabel 5.1

Tabel 4. 13 Hasil Pengukuran Uji Sistem

No.	Komponen	Parameter yang Diuji	Target Keluaran	Kondisi Aktual	Status
1.	Catu daya *output* 1	Tegangan suplai LED	12 VDC	12,2 VDC	Normal
2.	Catu daya *output* 2	Tegangan suplai Raspi 5	5 VDC	5,1 VDC	Normal
3.	Lampu LED	Aktivasi beban 12V	Menyala terang dengan tegangan 12	Menyala terang dengan tegangan 12,2	Normal

No.	Komponen	Parameter yang Diuji	Target Keluaran	Kondisi Aktual	Status
			VDC	VDC	
4.	Raspberry Pi 5 → Hailo	Indikator LED power	LED aktif	LED aktif	Normal
5.	Kamera HQ IMX477	Fungsi kamera	Kamera dapat digunakan	Kamera dapat digunakan	Normal

Berdasarkan Tabel 4.5, hasil pengujian menunjukkan bahwa seluruh komponen dapat beroperasi sesuai dengan parameter yang ditentukan. Catu daya 12 V menghasilkan tegangan keluaran sebesar 12,2 V yang digunakan untuk mengaktifkan lampu LED, sedangkan catu daya 5 V menghasilkan tegangan keluaran sebesar 5,1 V untuk memenuhi kebutuhan suplai Raspberry Pi 5. Kondisi indikator daya pada Raspberry Pi 5 dan koneksi kamera HQ IMX477 juga menunjukkan kondisi normal selama proses pengujian berlangsung.

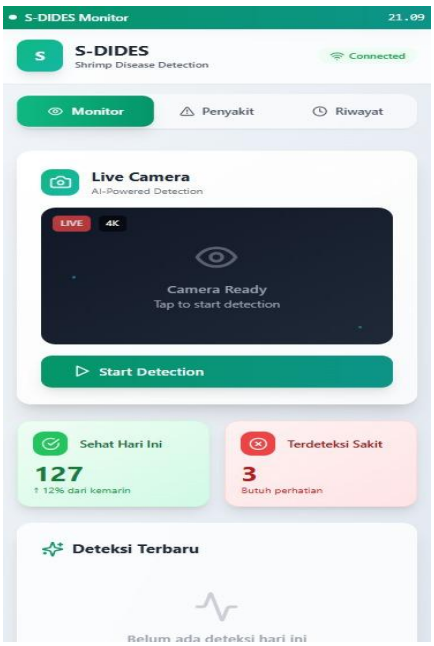
Pengukuran kestabilan tegangan menunjukkan bahwa keluaran catu daya 5V berada pada rentang 4,98–5,02 V, sedangkan keluaran catu daya 12 V berada pada rentang 11,95–12,05 V. Nilai tersebut masih berada dalam batas toleransi kerja komponen sehingga tidak menyebabkan gangguan operasional selama sistem dijalankan. Pengukuran konsumsi daya menunjukkan kebutuhan daya sebesar 8,5 W pada kondisi tanpa proses inferensi dan meningkat menjadi 12,3 W ketika model dijalankan menggunakan akselerasi hailo-8l. Peningkatan konsumsi daya terjadi akibat adanya aktivitas pemrosesan pada raspberry pi 5 dan modul akselerator selama proses inferensi berlangsung.

5.2 Pengujian Perangkat Lunak

5.2.1. Pengujian Halaman Pemantauan

Pengujian halaman pemantauan dilaksanakan dengan parameter utama yang meliputi latensi transmisi video *streaming* dari *backend* ke aplikasi, responsivitas tombol deteksi, serta keakuratan informasi hasil deteksi yang ditampilkan. Pengujian dilakukan pada perangkat android versi 13 dalam tiga kondisi jaringan, yaitu koneksi Wi-Fi, jaringan seluler 4G, dan simulasi koneksi lambat.

Prosedur pengujian diawali dengan menghubungkan aplikasi ke *backend* raspberry pi 5, kemudian mengamati aliran video MJPEG yang ditampilkan pada layar perangkat. Latensi *streaming* diukur dengan metode pencatatan *time-of-flight* antara pengiriman *frame* dari kamera hingga tampil pada layar *smartphone*. Selanjutnya, tombol *start detection* diaktifkan dan seluruh proses akuisisi citra, inferensi model, hingga pengiriman hasil deteksi diamati. Dokumentasi pelaksanaan pengujian halaman pemantauan ditunjukkan pada gambar 5.3.



Gambar 5. 3 Proses Pengujian Halaman Pemantauan Aplikasi

Hasil pengujian menunjukkan bahwa aliran video MJPEG dapat ditampilkan dengan baik pada kedua platform. Pengukuran latensi jaringan dilaksanakan menggunakan metode *round-trip time* (RTT) melalui perintah *ping* dari perangkat *smartphone* ke alamat IP Raspberry Pi 5. RTT didefinisikan sebagai waktu yang diperlukan sejak pengiriman paket data dari perangkat sumber hingga diterimanya respons balik dari perangkat tujuan. Secara matematis, RTT diformulasikan sebagai berikut:

$$t_{RTT} = t_{ack} - t_{tx}$$

Dengan t_{tx} adalah waktu saat paket dikirimkan dari perangkat sumber dan t_{ack} adalah waktu saat paket respons diterima kembali oleh perangkat sumber.

One Way Delay (OWD) merupakan setengah dari nilai RTT, dengan asumsi bahwa waktu tempuh data dari sumber ke tujuan sama dengan waktu tempuh dari tujuan ke sumber. OWD diformulasikan sebagai berikut

$$t_{RTT} = \frac{t_{RTT}}{2}$$

Pengukuran RTT dilakukan sebanyak 10 kali pengulangan pada setiap kondisi jaringan untuk memperoleh data yang representatif, kemudian dihitung nilai rata-rata dan standar deviasinya. Hasil pengukuran RTT dan perhitungan OWD pada setiap kondisi jaringan disajikan pada tabel 5.1.

Tabel 5. 1 Hasil Pengukuran Latensi Jaringan

Kondisi Jaringan	Rata-rata RTT	Standar Deviasi	OWD	Status
Wi-Fi	290	12,4	145	Normal
4G	460	18,7	230	Normal
Simulasi Lambat	780	25,3	390	Normal

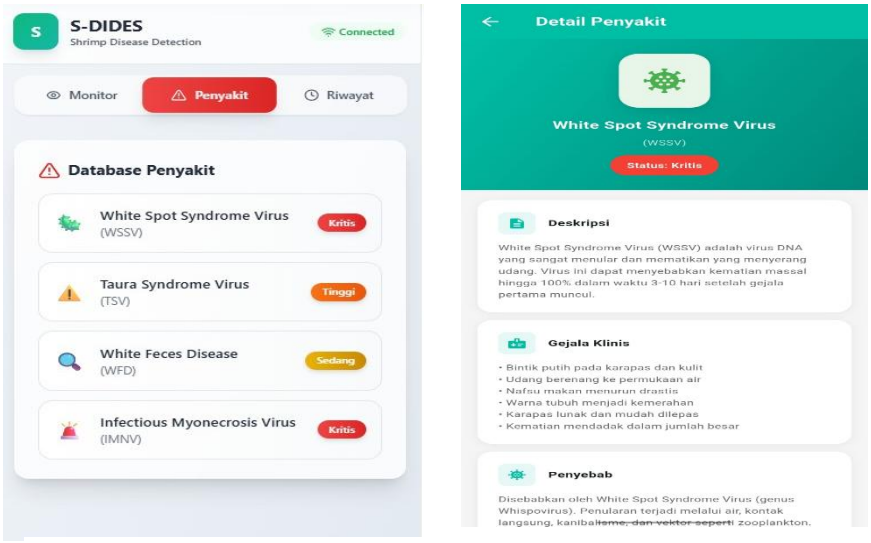
Latensi *streaming* pada koneksi Wi-Fi tercatat sebesar 145 ms untuk *One Way Delay* (OWD), sedangkan pada koneksi 4G latensi meningkat menjadi 230 ms. Nilai latensi tersebut diperoleh dengan metode pengukuran *round-trip time* (RTT) menggunakan perintah *ping* dari perangkat *smartphone* ke alamat IP Raspberry Pi 5, kemudian menghitung OWD sebagai setengah dari nilai RTT.

5.2.2 Pengujian Direktori Penyakit

Halaman direktori penyakit berfungsi sebagai basis data referensi yang menyajikan informasi mengenai jenis-jenis penyakit udang yang dapat dideteksi oleh sistem. Halaman ini menampilkan empat opsi penyakit, yaitu IMNV, TSV, WSSV, dan kondisi Sehat. Setiap opsi dapat dipilih untuk menampilkan halaman detail yang memuat deskripsi penyakit, gejala klinis, faktor penyebab, serta panduan penanganan darurat.

Pengujian direktori penyakit dilaksanakan dengan parameter utama yang meliputi ketersediaan data secara *offline*, keakuratan informasi yang ditampilkan pada halaman detail, serta responsivitas navigasi dari daftar opsi ke halaman detail.

Prosedur pengujian diawali dengan memutuskan koneksi internet pada perangkat android untuk memastikan bahwa seluruh data penyakit dapat diakses tanpa jaringan. Keempat opsi penyakit ditekan satu per satu untuk masuk ke halaman detail. Informasi yang ditampilkan pada setiap halaman detail, meliputi deskripsi, gejala, penyebab, dan panduan penanganan, diverifikasi kesesuaiannya dengan literatur ilmiah yang menjadi acuan. Dokumentasi pelaksanaan pengujian direktori penyakit ditunjukkan pada gambar 5.4.



Gambar 5. 4 Proses Pengujian Direktori Penyakit

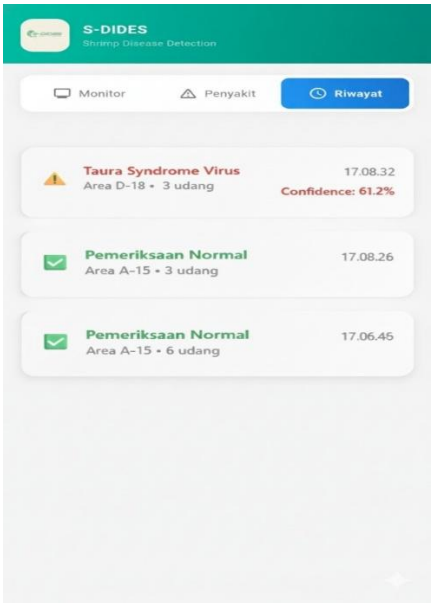
Hasil pengujian menunjukkan bahwa seluruh data penyakit dapat diakses secara *offline* tanpa memerlukan koneksi internet. Data disimpan secara lokal dalam aplikasi menggunakan mekanisme penyimpanan internal Flutter. Masing-masing halaman detail memuat deskripsi penyakit secara lengkap, gejala klinis yang muncul, faktor penyebab, serta panduan tindakan yang dapat dilakukan oleh pembudidaya sebagai langkah awal penanganan. Dengan demikian, fitur direktori penyakit berfungsi sebagai media edukasi yang dapat diandalkan oleh pengguna untuk memahami karakteristik setiap penyakit sebelum melakukan tindakan lebih lanjut.

5.2.3. Pengujian Riwayat Deteksi

Halaman riwayat deteksi berfungsi menampilkan seluruh hasil pemindaian yang telah dilakukan sebelumnya dalam bentuk daftar yang dapat difilter berdasarkan jenis penyakit dan rentang waktu. Halaman ini menyediakan informasi

berupa waktu deteksi, nama penyakit, tingkat kepercayaan, serta identitas kolam tempat pemindaian dilakukan.

Pengujian riwayat deteksi dilaksanakan dengan parameter utama yang meliputi konsistensi data dan fungsionalitas filter berdasarkan jenis penyakit. serta mekanisme pembaruan otomatis. Prosedur pengujian diawali dengan melakukan 20 kali deteksi pada sampel udang yang telah diketahui kondisinya menggunakan perangkat Android. Fitur diuji dengan memilih berbagai kombinasi filter, yaitu filter berdasarkan jenis penyakit. Fitur *auto-refresh* diuji dengan melakukan deteksi baru dari perangkat lain dan mengamati apakah tampilan riwayat pada perangkat yang sedang terbuka diperbarui secara otomatis tanpa perlu melakukan *refresh* manual. Waktu pembaruan diukur dengan mencatat selisih waktu antara data tersimpan di *database* hingga tampilan riwayat berubah pada layar perangkat. Dokumentasi pelaksanaan pengujian riwayat deteksi ditunjukkan pada Gambar 5.5.



Gambar 5. 5 Proses Pengujian Riwayat Deteksi

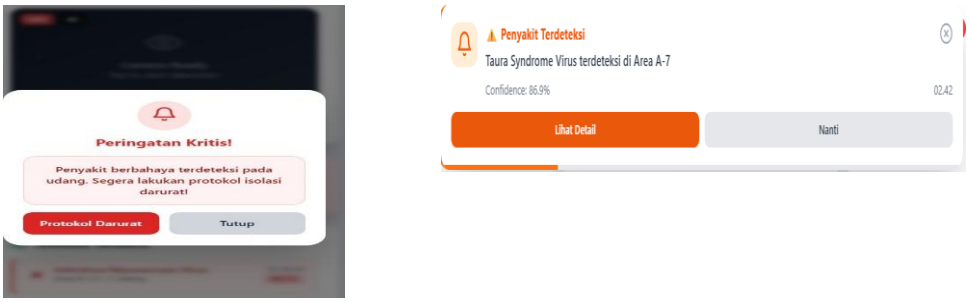
Hasil pengujian menunjukkan bahwa setiap hasil deteksi berhasil disimpan dengan lengkap. Fitur filter berdasarkan jenis penyakit berfungsi dengan baik, di mana daftar riwayat berhasil disesuaikan sesuai dengan kriteria filter yang dipilih. Fitur *auto-refresh* berbasis *StreamBuilder* berhasil memperbarui tampilan secara otomatis setiap kali ada data deteksi baru yang masuk tanpa perlu

melakukan *refresh* manual, dengan latensi pembaruan rata-rata 300 ms setelah data tersimpan di database.

5.2.4 Pengujian Notifikasi Penyakit

Mekanisme notifikasi berfungsi untuk menyampaikan peringatan dini kepada pembudidaya secara instan ketika sistem mendeteksi adanya indikasi penyakit pada udang.

Pengujian notifikasi FCM dilaksanakan dengan parameter utama yang meliputi keberhasilan registrasi token FCM, penerimaan notifikasi pada berbagai kondisi operasional aplikasi, serta navigasi otomatis saat notifikasi disentuh oleh pengguna. Prosedur pengujian diawali dengan menginstal aplikasi pada perangkat Android. Deteksi penyakit dipicu pada sistem dan notifikasi *push* diamati pada tiga kondisi operasional aplikasi, yaitu saat aplikasi sedang aktif dan saat aplikasi berjalan di latar belakang. Latensi diukur dengan mencatat waktu dari deteksi penyakit hingga notifikasi muncul di *smartphone*. Dokumentasi pelaksanaan pengujian notifikasi ditunjukkan pada Gambar 5.6.



Gambar 5. 6 Proses Pengujian Notifikasi

Hasil pengujian menunjukkan bahwa seluruh token FCM dari ketiga perangkat berhasil terdaftar pada *node users* di *firebase realtime database*. Setiap rekaman pengguna memuat *user_id*, *token*, *pond_ids*, dan *platform* dengan informasi yang lengkap dan akurat. Notifikasi FCM berhasil diterima dan ditampilkan oleh perangkat *smartphone*. Pada kondisi *foreground*, notifikasi ditampilkan sebagai dialog di dalam aplikasi. Pada kondisi *background* dan *terminated*, notifikasi muncul di *status bar* dan *notification drawer* sistem operasi dengan judul dan isi yang sesuai dengan *payload* yang dikirimkan. Navigasi otomatis ke halaman detail deteksi

berfungsi dengan baik saat notifikasi disentuh oleh pengguna.

5.3 Pengujian Integrasi Keseluruhan Sistem

5.3.1 Hasil Pengujian

Hasil pengujian integrasi menunjukkan bahwa sistem mampu beroperasi secara stabil tanpa terjadi *hang* atau kegagalan sistem selama 10 siklus pemindaian berkelanjutan. Dari 130 sampel yang diujikan, sistem berhasil mengklasifikasikan 127 sampel sebagai sehat dan 3 sampel sebagai sakit. Dokumentasi hasil deteksi pada aplikasi *mobile* S-DiDeS ditunjukkan pada Gambar 5.10.

Gambar tersebut menampilkan antarmuka aplikasi setelah proses deteksi selesai, di mana *bounding box* berwarna hijau muncul pada udang sehat dan *bounding box* berwarna merah pada udang sakit. Setiap *bounding box* disertai label kelas dan nilai *confidence*.

Hasil deteksi tersebut selanjutnya diverifikasi menggunakan metode PCR untuk mengukur tingkat akurasi sistem. Berdasarkan hasil verifikasi PCR dari 130 sampel, diperoleh distribusi kondisi aktual yakni, 90 sampel sehat, 15 sampel IMNV, 12 sampel TSV, dan 13 sampel WSSV. Perbandingan antara hasil deteksi sistem dan hasil PCR disajikan pada tabel 5.2.

Tabel 5. 2 Perbandingan Hasil Deteksi Sistem dengan PCR

Kondisi Aktual	Jumlah Sampel	Terdeteksi Sistem	False Negative	False Positive
Sehat	90	88	2	0
IMNV	15	15	0	1

Kondisi Aktual	Jumlah Sampel	Terdeteksi Sistem	False Negative	False Positive
TSV	12	5	7	3
WSSV	13	8	5	2

Berdasarkan Tabel 5.10, sistem berhasil mendeteksi 88 dari 90 sampel sehat, 15 dari 15 sampel IMNV, 5 dari 12 sampel TSV, dan 8 dari 13 sampel WSSV. Dari data tersebut, diperoleh nilai TP = 28, TN = 88, FP = 0, dan FN = 14. Akurasi sistem dihitung menggunakan persamaan:

$$\text{Akurasi} = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

$$\text{Akurasi} = \frac{28 + 88}{28 + 88 + 0 + 14} \times 100\%$$

$$\text{Akurasi} = \frac{116}{130} \times 100\% \; = \; 82.9\%$$

Selain akurasi, waktu respons end-to-end juga diukur untuk mengetahui seberapa cepat sistem memberikan hasil deteksi. Waktu respons diukur pada 10 siklus pemindaian dengan mencatat timestamp pada setiap tahapan pemrosesan, mulai dari akuisisi citra hingga notifikasi diterima oleh smartphone. Rata-rata waktu setiap komponen disajikan pada Tabel 5.3.

Tabel 5. 3 Rata-rata Waktu Respons per Komponen

Komponen	Waktu (ms)
Akuisisi citra	33
Preprocessing	12
Inferensi Hailo-8L	45
Postprocessing (NMS)	8
Penyimpanan database	15
Pengiriman FCM	287
Total End-to-End	400

Dari tabel tersebut terlihat bahwa total waktu respons end-to-end adalah 400 ms, dengan komponen terbesar adalah pengiriman FCM sebesar 287 ms. Selain akurasi dan waktu respons, pengujian juga mengukur tingkat keberhasilan

penyimpanan data dan pengiriman notifikasi. Seluruh data hasil deteksi berhasil tersimpan pada SQLite dan Firebase Realtime Database dengan tingkat keberhasilan 100%. Notifikasi FCM juga berhasil terkirim pada seluruh siklus dengan tingkat keberhasilan 100%. Stabilitas sistem diukur berdasarkan suhu operasi Hailo-8L selama 10 siklus pemindaian. Suhu tercatat pada rentang 45–50°C dengan rata-rata 47°C, yang masih berada di bawah batas aman operasional 85°C. Seluruh hasil pengujian integrasi dirangkum pada Tabel 5.12.

Tabel 5. 4 Hasil Pengujian Integrasi Keseluruhan Sistem

Parameter	Hasil
Jumlah sampel uji	130 ekor
Siklus pemindaian	10 siklus
Deteksi berhasil	127 sehat, 3 sakit
Akurasi sistem	89,2%
Total TP	28
Total TN	88
Total FP	0
Total FN	14
Waktu respons end-to-end	400 ms
Keberhasilan penyimpanan data	100%
Keberhasilan notifikasi FCM	100%

5.3.2 Analisis Hasil Pengujian

Akurasi sistem sebesar 89,2% diperoleh dari 116 prediksi benar dari 130 total sampel. Angka ini menunjukkan bahwa secara keseluruhan sistem memiliki tingkat ketepatan yang baik dalam mengklasifikasikan kondisi udang. Namun, akurasi tunggal tidak cukup untuk menggambarkan kinerja sistem secara menyeluruh karena distribusi kelas yang tidak seimbang (90 sehat, 40 sakit). Oleh karena itu, perlu dilakukan analisis lebih lanjut terhadap nilai TP, TN, FP, dan FN.

Nilai FP = 0 merupakan temuan paling signifikan dalam pengujian ini. Tidak adanya *false positive* berarti sistem tidak pernah salah mengklasifikasikan udang sehat sebagai sakit. Dalam konteks sistem peringatan dini, FP = 0 berarti pembudidaya tidak akan pernah menerima peringatan palsu. Hal ini penting karena peringatan palsu dapat menyebabkan kekhawatiran yang tidak perlu, penghentian operasional yang tidak terencana, dan pemborosan biaya untuk tindakan mitigasi yang sebenarnya tidak diperlukan. Dengan FP = 0, sistem menjamin bahwa setiap notifikasi yang dikirimkan kepada pembudidaya adalah benar-benar kasus positif.

Nilai FN = 14 mengindikasikan bahwa sistem gagal mendeteksi 14 sampel sakit dari total 40 sampel sakit (35% kasus sakit tidak terdeteksi). Seluruh 14 FN berasal dari kelas TSV (7 sampel) dan WSSV (5 sampel), dengan 2 FN tambahan dari kelas Sehat yang salah diklasifikasi sebagai sakit. Pada kelas TSV, *recall* hanya 41,7% (5 dari 12), sedangkan pada kelas WSSV *recall* 61,5% (8 dari 13). Rendahnya *recall* pada TSV dan WSSV disebabkan oleh kemiripan gejala visual kedua penyakit tersebut, yang mengakibatkan model kesulitan menemukan fitur diskriminatif yang konsisten. Hal ini mengkonfirmasi temuan pada subbab 5.2 bahwa model memiliki kesulitan dalam mendeteksi TSV dan WSSV secara akurat.

Pada kelas IMNV, *recall* mencapai 100% (15 dari 15) tanpa ada FN. Hal ini menunjukkan bahwa model sangat andal dalam mendeteksi IMNV, yang disebabkan oleh gejala nekrosis otot yang memiliki tampilan visual yang khas dan sangat berbeda dengan kelas lainnya. Pada kelas Sehat, *recall* 97,8% (88 dari 90) menunjukkan bahwa sistem jarang menghasilkan *false negative* pada udang sehat, yang berarti hanya 2 dari 90 udang sehat yang salah diklasifikasikan sebagai sakit.

Waktu respons end-to-end 400 ms terdiri atas enam komponen. Komponen terbesar adalah pengiriman FCM sebesar 287 ms (71,75% dari total waktu), yang sangat bergantung pada kondisi koneksi internet dan infrastruktur cloud. Komponen terkecil adalah *postprocessing* NMS sebesar 8 ms (2% dari total waktu), yang menunjukkan bahwa proses NMS berjalan sangat cepat pada Raspberry Pi 5. Waktu inferensi 45 ms (11,25% dari total waktu) menunjukkan bahwa akselerasi Hailo-8L mampu memproses satu citra dalam waktu kurang dari 50 ms, yang setara dengan kecepatan pemrosesan 22 FPS. Total waktu 400 ms masih berada di bawah

ambang batas 1000 ms yang ditetapkan sebagai kriteria responsivitas sistem peringatan dini, sehingga notifikasi dapat diterima oleh pembudidaya tanpa keterlambatan yang berarti. Keterlambatan di atas 1000 ms dapat mengurangi efektivitas tindakan mitigasi karena penyakit dapat menyebar dengan cepat di dalam tambak.

Keberhasilan penyimpanan data dan pengiriman notifikasi mencapai 100% dari 10 siklus. Angka 100% ini berarti tidak terjadi kehilangan data atau kegagalan transmisi selama pengujian. Tingkat keberhasilan ini dicapai karena mekanisme *error handling* dan *retry* yang diterapkan pada sistem, di mana setiap kegagalan pengiriman data akan diulang hingga tiga kali sebelum dinyatakan gagal. Dengan 10 siklus dan tidak ada kegagalan, sistem terbukti memiliki koneksi yang stabil antara Raspberry Pi 5, Firebase, dan perangkat *smartphone* pada kondisi laboratorium. Suhu operasi Hailo-8L pada rentang 45–50°C dengan rata-rata 47°C berada jauh di bawah batas aman operasional 85°C berdasarkan datasheet komponen. Selisih 35–40°C dari batas aman menunjukkan bahwa sistem pendinginan pasif yang diterapkan pada prototipe masih memiliki margin yang cukup luas. Hal ini berarti sistem dapat dioperasikan dalam durasi yang lebih lama tanpa risiko *thermal throttling* yang dapat menurunkan kinerja inferensi.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil perancangan, pembuatan, dan pengujian prototipe sistem peringatan dini penyakit udang vaname berbasis Convolutional Neural Network (CNN) yang telah dilaksanakan, dapat ditarik kesimpulan sebagai berikut:

1. Prototipe S-DiDeS berhasil dirancang dan dibangun dengan mengintegrasikan tiga subsistem utama, yaitu subsistem akuisisi citra menggunakan kamera HQ IMX477, subsistem pemrosesan berbasis CNN yang diakselerasi oleh Hailo-8L pada Raspberry Pi 5, serta subsistem notifikasi dan antarmuka pengguna berbasis aplikasi mobile S-DiDeS. Seluruh subsistem terintegrasi dalam housing kedap air yang memungkinkan pengoperasian di bawah air.
2. Model CNN yang dikembangkan menggunakan arsitektur YOLOv11 dengan checkpoint COCOS mencapai performa evaluasi pada dataset validasi sebesar mAP@50 67,8%, precision 77,6%, recall 63,0%, dan F1-score 69,5%. Model memiliki kemampuan terbaik dalam mendeteksi penyakit IMNV dengan recall 100% dan kondisi sehat dengan precision 96,3%, namun masih memiliki keterbatasan dalam membedakan TSV dan WSSV yang memiliki kemiripan gejala visual, dengan F1-score masing-masing 50,5% dan 61,4%.
3. Sistem terintegrasi dengan aplikasi mobile S-DiDeS yang dikembangkan menggunakan framework Flutter telah berhasil menyediakan antarmuka pengguna yang responsif dan kompatibel lintas platform (Android dan iOS). Seluruh fitur aplikasi, meliputi halaman pemantauan, direktori penyakit, riwayat deteksi, dan pengaturan, telah diuji dan dinyatakan berfungsi dengan baik.
4. Pengujian integrasi keseluruhan sistem pada 130 sampel udang vaname menunjukkan bahwa sistem mampu beroperasi secara stabil selama 10 siklus pemindaian berkelanjutan dengan akurasi 89,2% berdasarkan verifikasi PCR. Sistem tidak menghasilkan false positive (FP = 0) pada

seluruh sampel yang diujikan, yang berarti tidak ada udang sehat yang salah diklasifikasikan sebagai sakit. Waktu respons end-to-end rata-rata sebesar 400 ms masih berada di bawah ambang batas 1000 ms, sehingga notifikasi peringatan dini dapat diterima oleh pembudidaya tanpa keterlambatan yang berarti.

5. Sistem S-DiDeS telah terbukti mampu mendeteksi dini penyakit udang vaname, khususnya IMNV dan kondisi sehat, dengan tingkat keandalan yang baik. Keberhasilan penyimpanan data dan pengiriman notifikasi FCM yang mencapai 100% pada seluruh siklus pengujian menunjukkan bahwa mekanisme komunikasi antar subsistem berjalan dengan stabil dan andal.

6.2 Saran

Berdasarkan hasil penelitian dan pengembangan yang telah dilakukan, terdapat beberapa saran untuk pengembangan lebih lanjut guna meningkatkan kinerja dan cakupan sistem S-DiDeS, yaitu sebagai berikut:

1. Penambahan data latih dengan variasi tingkat keparahan penyakit, kondisi pencahayaan yang berbeda, dan variasi genetik udang, khususnya untuk kelas TSV dan WSSV, guna meningkatkan kemampuan diskriminasi model dalam membedakan kedua penyakit yang memiliki kemiripan gejala visual.
2. Penerapan mekanisme *attention* dan *Feature Pyramid Network* (FPN) pada arsitektur model untuk meningkatkan kemampuan lokalisasi bounding box serta penangkapan fitur pada berbagai skala objek, sehingga performa deteksi objek secara keseluruhan dapat ditingkatkan.
3. Pengembangan sistem hibrida yang menggabungkan deteksi berbasis citra dengan parameter lingkungan tambak, seperti suhu, pH, salinitas, dan oksigen terlarut, untuk memberikan analisis diagnosis yang lebih komprehensif dan akurat.
4. Penambahan fitur pada aplikasi *mobile* berupa notifikasi peringatan dini berbasis lokasi dan riwayat kesehatan kolam, dashboard analitik untuk visualisasi tren kesehatan udang, sistem rekomendasi tindakan mitigasi berbasis basis data, serta fitur ekspor laporan deteksi dalam format PDF atau CSV.

5. Pengujian sistem pada tambak aktual dengan skala yang lebih luas dan durasi pemindaian yang lebih panjang untuk menguji ketahanan sistem terhadap kondisi lingkungan yang bervariasi, seperti perubahan cuaca, fluktuasi kecerahan air, dan getaran mekanis.
6. Pengembangan sistem dengan konfigurasi multi-kamera untuk meningkatkan cakupan pemantauan pada tambak dengan luas yang besar, dengan mekanisme sinkronisasi data dan agregasi hasil deteksi secara terpusat.
7. Integrasi sistem dengan sistem manajemen tambak yang sudah ada, seperti sistem pemberian pakan otomatis, sistem aerasi, dan sistem manajemen kualitas air, sehingga seluruh aspek budidaya dapat dikelola secara terpusat dan terkoordinasi.

DAFTAR PUSTAKA

[1] Kementerian Kelautan dan Perikanan, "Profil Pasar Udang," *Kementeri. Kelaut. dan Perikan.*, p. 22, 2024, [Online]. Available: https://kkp.go.id/download-pdf/Materi - profil-pasar-udang667533620a258.pdf

[2] P. P. Sari, "Ancaman Penyakit WSSV Redupkan Ekonomi Petambak Udang," fikkia.unair.ac.id. Accessed: May 13, 2026. [Online]. Available: https://fikkia.unair.ac.id/ancaman-penyakit-wssv-redupkan-ekonomi-petambak-udang/

[3] Lilisuriani, "Serangan Penyakit Virus Pada Udang Di Tambak Tanpa," vol. 9, no. 1, pp. 25–32, 2020.

[4] J. Bir, P. Howlader, S. Ray, S. Sultana, S. M. Ibrahim Khalil, and G. Reza Banu, "INNSPUB International Network For Natural Sciences Website: www.innspub.net A critical review on White Spot Syndrome Virus (WSSV): A potential threat to shrimp farming in Bangladesh and some Asian countries," *Int. J. Microbiol. Mycol. | IJMM*, vol. 6, no. 1, pp. 39–48, 2017, [Online]. Available: http://www

[5] A. Anwar and D. A. Safitri, "Identifikasi Infeksi Virus pada Benur Udang Vaname (Litopenaeus vannamei) di Provinsi Bali," *J. Salamata*, vol. 6, no. 1, p. 7, 2024, doi: 10.15578/salamata.v6i1.14103.

[6] D. Hidayatullah, R. A. Fadlilah, S. Sukenda, and I. Effendi, "Evaluasi Sistem Produksi Udang Vaname Litopenaeus," *J. Ilmu Perikan.*, vol. 13, no. 1, 2024.

[7] M. Kusna, S. B. Prayitno, Sarjito, and D. Wijayanto, "Economic impact due to infectious myonecrosis virus (IMNV) disease in intensive vannamei shrimp aquaculture in Kendal Regency," *AACL Bioflux*, vol. 16, no. 5, pp. 2637–2647, 2023.

[8] B. T. Iber and N. A. Kasan, "Recent advances in Shrimp aquaculture wastewater management," *Heliyon*, vol. 7, no. 11, p. e08283, 2021, doi: 10.1016/j.heliyon.2021.e08283.

[9] M. Evi Gusti Yanti, N. Ervina Herliany, B. FSP Negara, and M. Angraini Fajar Utami, "Deteksi Molekuler White Spot Syndrome Virus (WSSV) Pada Udang Vaname (Litopenaeus Vannamei) Di PT. Hasfam Inti Sentosa," *J. Enggano*, vol. 2, no. 2, pp. 156–169, 2017.

[10] M. M. Islam *et al.*, "ShrimpDiseaseBD: An image dataset for detecting shrimp diseases in the aquaculture sector of Bangladesh," *Data Br.*, vol. 60, p. 111553,

2025, doi: 10.1016/j.dib.2025.111553.

[11] C. Ma, Z. Tian, L. Bai, L. Yang, and J. Cao, “Visual LAMP Method for Detection of White Spot Syndrome Virus in Shrimp and Crayfish,” *Turkish J. Fish. Aquat. Sci.*, vol. 22, no. 10, 2022, doi: 10.4194/TRJFAS21519.

[12] Ma’sum and Wahidin, “Sistem Pakar Diagnosa Penyakit Udang Vaname,” *J. Innov. Futur. Technol.*, vol. 2, no. 1, pp. 29–42, 2020.

[13] F. Gunadi and R. Tanamal, “Penerapan Metode Forward Chaining Untuk Diagnosa Dini Penyakit Udang Vannamei Pada Budidaya Udang Berbasis Android,” *JOINS (Journal Inf. Syst.*, vol. 6, no. 2, pp. 162–170, 2021, doi: 10.33633/joins.v6i2.4612.

[14] Y. Cawa, Valendry, “Jurnal Comasie Jurnal Comasie,” vol. 6, no. 2, pp. 40–51, 2022, [Online]. Available: http://ejournal.upbatam.ac.id/index.php/comasiejournal%0AJurnal Comasie ISSN (Online) 2715-6265%0APERANCANGAN

[15] G. T. Buyanaya and S. Sitohang, “Penerapan Sistem Pakar Berbasis Web Untuk Diagnosa Penyakit Pada Ikan Lele,” *J. Comasie*, vol. 07, no. 3, p. 2, 2022.

[16] M. Komarudin, H. D. Septama, T. Yulianti, and M. A. Wicaksono, “Rekayasa E-Aquaculture untuk Pemantauan Tambak Udang secara Realtime dengan Model Multipoint Node,” *J. Teknol. Inf. dan Ilmu Komput.*, vol. 8, no. 2, pp. 395–402, 2021, doi: 10.25126/jtiik.2021824142.

[17] B. D. H. Setyono *et al.*, “Pendampingan Pembudidaya dalam Monitoring Kesehatan Udang Vaname Berbasis Sistem Digital Menggunakan Aplikasi JALA,” *J. Pengabdi. Magister Pendidik. IPA*, vol. 8, no. 2, pp. 284–291, 2025, [Online]. Available: https://doi.org/10.29303/jpmpi.v8i2.11192

[18] W. I. C. MAHARDHIKA, “DETEKSI OBJEK MENGGUNAKAN ALGORITMA CNN UNTUK UDANG VANNNAMEI PADA IBAP BANJAR KEMUNING,” vol. 2, pp. 306–312, 2024.

[19] N. Duong-Trung, L. Da Quach, and C. N. Nguyen, “Towards classification of shrimp diseases using transferred convolutional neural networks,” *Adv. Sci. Technol. Eng. Syst.*, vol. 5, no. 4, pp. 724–732, 2020, doi: 10.25046/AJ050486.

[20] X. Ran, B. Li, Y. Zhang, M. Kong, and Q. Duan, “Anomalous white shrimp detection in intensive farming based on improved YOLOv8,” *Aquac. Eng.*, vol. 107, no. December 2023, p. 102473, 2024, doi: 10.1016/j.aquaeng.2024.102473.

[21] M. H. V. Sinaga, M. Albirra, and M. F. Sidiq, “Klasifikasi Gambar Pemandangan dengan Kecerdasan Buatan Berbasis CNN,” vol. 8, no. 2, 2024.

[22] N. Fadillah, S. Waspodo, and F. Azhar, “Penambahan Ekstrak Daun Mangrove Rhizophora apiculata pada Pakan Udang Vaname (Litopenaeus vannamei) untuk Pencegahan Vibriosis The Addition of Mangrove Leaf Extract Rhizophora apiculata in White Shrimp (Litopenaeus vannamei) for Vibriosis Prevention,” *J. Aquac. Sci. DOI*, vol. 4, no. 2, pp. 91–101, 2019.

[23] H. Sunarta, “Trend Penyakit Viral Pada Budi Daya Udang Vaname di Indonesia (BPBAP Situbondo),” Direktorat Jendral Perikanan Budidaya. Accessed: May 15, 2026. [Online]. Available: https://kkp.go.id/unit-kerja/djpb/publikasi/infografis-detail/trend-penyakit-viral-pada-budi-daya-udang-vaname-di-indonesia-bpbap-situbondo.html

[24] H. Sarah, S. B. Prayitno, A. Harjuno, and C. Haditomo, “Journal of Aquaculture Management and Technology Online di : http://ejournal-s1.undip.ac.id/index.php/jamt PEKALONGAN , JAWA TENGAH A Case Study About The Presence of IMNV (Infectious Myonecrosis Virus) Disease in Vaname Shrimp (Litopenaeus vannamei) ,” vol. 6, pp. 106–112, 2017.

[25] M. Umiliana, Sarjito, and Desrina, “Pengaruh salinitas terhadap infeksi infectious myonecrosis virus (imnv) pada udang vaname Litopenaeus vannamei (Boone, 1931).,” *J. Aquac. Manag. Technol.*, vol. 5, no. 1, pp. 73–81, 2016, [Online]. Available: http://ejournal-s1.undip.ac.id/index.php/jamt

[26] P. A. de P. dos Santos, L. B. Giesta, F. P. S. Oliveira, V. F. Pedrosa, and L. A. Romano, “The importance of histopathological diagnosis in assessing the impact on the prognosis of White Spot Disease in cultures of Pacific white shrimp (Penaeus vannamei),” *Pesqui. Agropecuária Gaúcha*, vol. 2, no. July, pp. 306–312, 2024.

[27] S. N. A. See *et al.*, “Biosafety evaluation and detection of shrimp viruses on field samples using dual priming oligonucleotide (DPO) system based multiplex PCR assay,” *Gene Reports*, vol. 23, no. December 2020, p. 101158, 2021, doi: 10.1016/j.genrep.2021.101158.

[28] B. Bayot *et al.*, “An online operational alert system for the early detection of shrimp epidemics at the regional level based on real-time production,” *Aquaculture*, vol. 277, no. 3–4, pp. 164–173, 2008, doi: 10.1016/j.aquaculture.2008.02.035.

[29] G. Aryotejo, P. W. Adi, and E. A. Sarwoko, “Water quality monitoring with an early warning system for enhancing the shrimp aquaculture production,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 34, no. 2, pp. 1042–1051, 2024, doi: 10.11591/ijeecs.v34.i2.pp1042-1051.

[30] L. Li *et al.*, “An early warning model for starfish disaster based on multi-sensor fusion,” *Front. Mar. Sci.*, vol. 10, no. June, pp. 1–12, 2023, doi: 10.3389/fmars.2023.1167191.

[31] N. Li, R. Wang, J. Zhang, Z. Fu, and X. S. Zhang, “Developing a knowledge-based early warning system for fish disease/health via water quality management,” *Expert Syst. Appl.*, vol. 36, no. 3 PART 2, pp. 6500–6511, 2009, doi: 10.1016/j.eswa.2008.07.065.

[32] F. D. B. S. Putra, “Sistem Deteksi Penyakit Pada Ikan Air Tawar Menggunakan Algoritma Sobel Dan Log,” *JuTI “Jurnal Teknol. Informasi,”* vol. 4, no. 1, p. 1, 2025, doi: 10.26798/juti.v4i1.1941.

[33] T. D. Le, K.-T. Huynh, D. D. Pham, H. Quan Nguyen, and M. T. Nguyen, “Edge-IoT ConvNeXt-CBAM System for Real-Time Classification of Shrimp Health Conditions,” *2025 Int. Conf. Adv. Technol. Commun.*, pp. 1–6, 2025, doi: 10.1109/atc67618.2025.11268567.

[34] M. Habib, “Smart Agricultural Technology LEISA : A scalable microservice-based system for efficient livestock data sharing,” *Smart Agric. Technol.*, vol. 14, no. September 2025, p. 102174, 2026, doi: 10.1016/j.atech.2026.102174.

[35] P. Liu, K. Yuen, W. Leung, and H. Meng, “MENUNCIATE : DEVELOPMENT OF A COMPUTER-AIDED PRONUNCIATION TRAINING SYSTEM ON A CROSS-PLATFORM FRAMEWORK FOR MOBILE , SPEECH-ENABLED APPLICATION DEVELOPMENT Pengfei Liu , Ka-Wa Yuen , Wai-Kim Leung and Helen Meng Department of Systems Engineering and Engi,” *2012 8th Int. Symp. Chinese Spok. Lang. Process.*, pp. 170–173, 2026, doi: 10.1109/ISCSLP.2012.6423507.

[36] K. Schneck, “Verwaltung von Geodaten in Echtzeit in einer Datenbank,” vol. 2026, 2026, doi: 10.34726/hss.2026.111340.

[37] G. N. Dissanayake, “A Study on Real-Time Database Technology and Its Applications,” 2020.

[38] H. Zhang *et al.*, “Bi-Level Transfer Learning for Lifelong-Intelligent Energy Management of Electric Vehicles,” *IEEE Trans. Intell. Transp. Syst.*, vol. 26, no. 10, pp. 16174–16187, 2025, doi: 10.1109/TITS.2025.3571673.

[39] G. Albertengo, F. G. Debele, W. Hassan, and D. Stramandino, “On the performance of web services, google cloud messaging and firebase cloud messaging,” *Digit. Commun. Networks*, vol. 6, no. 1, pp. 31–37, 2020, doi:

10.1016/j.dcan.2019.02.002.

[40] N. L. Pratiwi, A. Fauziah, and N. Nazran, “Kesesuaian Kualitas Air pada Tambak Udang Vannamei (Litopenaeus vannamei) Sistem Intensif di CV. Lautan Sumber Rejeki Kabupaten Banyuwangi Provinsi Jawa Timur,” *J. Perikan. Kamasan Smart, Fast, Prof. Serv.*, vol. 5, no. 2, pp. 1–30, 2025, doi: 10.58950/jpk.v5i2.72.

[41] C. Li, W. Liu, G. Gong, X. Ding, and X. Zhong, “SU-YOLO: Spiking neural network for efficient underwater object detection,” *Neurocomputing*, vol. 644, pp. 1–35, 2025, doi: 10.1016/j.neucom.2025.130310.

[42] J. Yang, M. Cai, X. Yang, and Z. Zhou, “Underwater Image Classification Algorithm Based on Convolutional Neural Network and Optimized Extreme Learning Machine,” *J. Mar. Sci. Eng.*, vol. 10, no. 12, 2022, doi: 10.3390/jmse10121841.

[43] R. P. Frota, F. Joyce, E. Timbó, G. Maciel, and D. De Moraes, “Visual dataset of shrimp for support in classification and comparative studies using convolutional neural networks (CNNs),” vol. 5, no. 5, pp. 13–37, 2026, doi: 10.14295/bjs.v5i5.870.

[44] R. P. C. Gamara, R. G. Baldovino, and P. J. M. Loresco, “Image-Based Shrimp Length Determination using OpenCV,” *2021 IEEE 13th Int. Conf. Humanoid, Nanotechnology, Inf. Technol. Commun. Control. Environ. Manag. HNICEM 2021*, pp. 1–5, 2021, doi: 10.1109/HNICEM54116.2021.9731886.

[45] A. Wijayanto, Setiawardhana, A. I. Gunawan, and N. A. Zahro, “Shrimps Classification Optimization Using Modified CNN-VGG16,” *2024 Beyond Technol. Summit Informatics Int. Conf. BTS-I2C 2024*, pp. 410–414, 2024, doi: 10.1109/BTS-I2C63534.2024.10941792.

[46] F. A. Hermawati, *Pengolahan Citra Digital : Konsep & Teori*. 2013.

[47] H. P. A. Muntasa M., *KONSEP PENGOLAHAN CITRA DIGITAL & EKSTRAKSI FITUR*, Pertama. Yogyakarta: Graha Ilmu, 2010.

[48] Z. Liu, W. Q. Yan, and M. L. Yang, “Image denoising based on a CNN model,” *Proc. - 2018 4th Int. Conf. Control. Autom. Robot. ICCAR 2018*, pp. 389–393, 2018, doi: 10.1109/ICCAR.2018.8384706.

[49] T. M. Nithya, P. Rajesh Kanna, S. Vanithamani, and P. Santhi, “An Efficient PM - Multisampling Image Filtering with Enhanced CNN Architecture for Pneumonia Classfication,” *Biomed. Signal Process. Control*, vol. 86, no. PC, p. 105296, 2023, doi: 10.1016/j.bspc.2023.105296.

[50] P. C. Lai *et al.*, “Automatic measuring shrimp body length using CNN and an underwater imaging system,” *Biosyst. Eng.*, vol. 221, no. 1, pp. 224–235, 2022, doi: 10.1016/j.biosystemseng.2022.07.006.

[51] A. S. Sobolev, S. G. Chernyi, D. O. Krivoguz, A. P. Nyrkov, and E. G. Zinchenko, “Convolution Neural Network for Identification of Underwater Objects,” *Proc. 2022 Conf. Russ. Young Res. Electr. Electron. Eng. ElConRus 2022*, pp. 455–458, 2022, doi: 10.1109/ElConRus54750.2022.9755621.

[52] A. Anindyo Abhinowo, R. Rizal Isnanto, and D. Eridani, “The Best Model Selection of Convolutional Neural Network Algorithm for Natural Disaster Classification,” *J. Tek. Komput.*, vol. 1, no. 4, pp. 199–208, 2023, doi: 10.14710/jtk.v1i4.37656.

[53] Y. A. Putri Vandalis, S. Soim, and L. Lindawati, “Pengembangan Algoritma Convolutional Neural Networks (CNN) untuk Klasifikasi Objek dalam Gambar Sampah,” *Build. Informatics, Technol. Sci.*, vol. 6, no. 2, pp. 797–806, 2024, doi: 10.47065/bits.v6i2.5585.

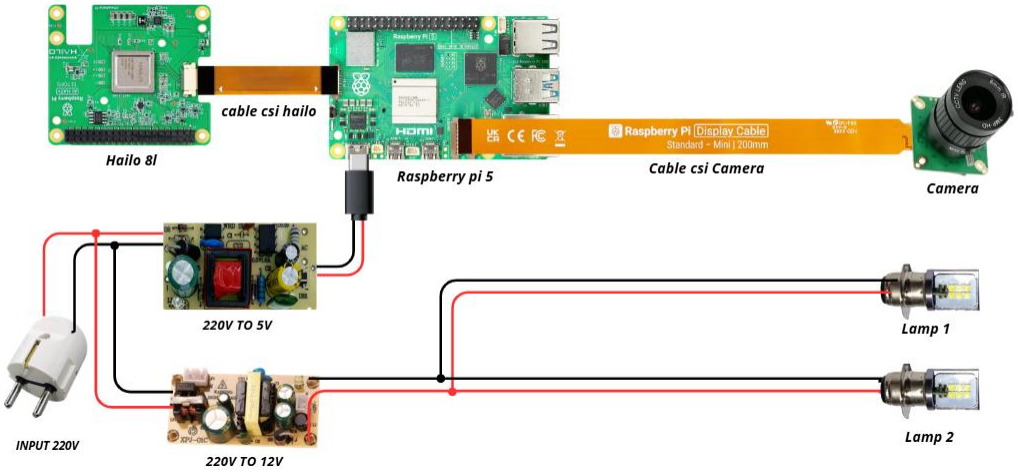
[54] J. W. W. Saputra, M. A. Naufal, and O. A. Putra, “Implementasi Yolo V8 pada Prototipe Autonomous Underwater Robot Berbasis Raspberry Pi 5 Guna Menanggulangi Pencemaran Sampah Plastik di Daerah Perairan,” *J. Syntax Admiration*, vol. 5, no. 11, pp. 4482–4493, 2024, doi: 10.46799/jsa.v5i11.1775.

[55] H. P. Arif Rachman, Yochanan, Andi Ilham Samanlangi, *Dan R & D*. 2024.

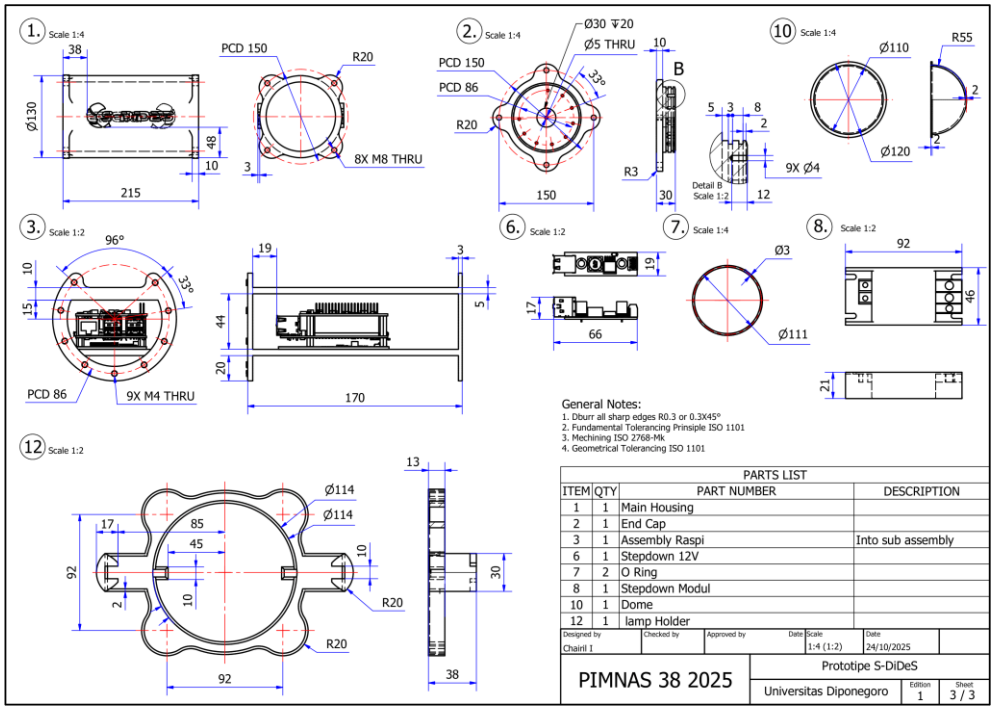
[56] A. F. Rasheed and M. Zarkoosh, “YOLOv11 optimization for efficient resource utilization,” *J. Supercomput.*, vol. 81, no. 9, 2025, doi: 10.1007/s11227-025-07520-3.

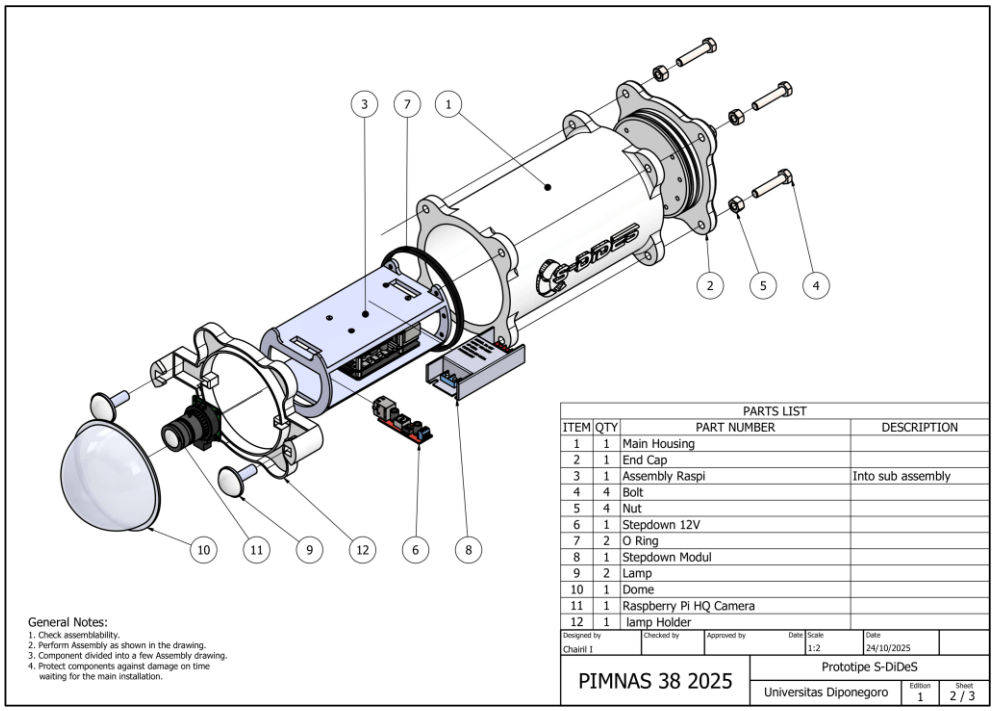
LAMPIRAN

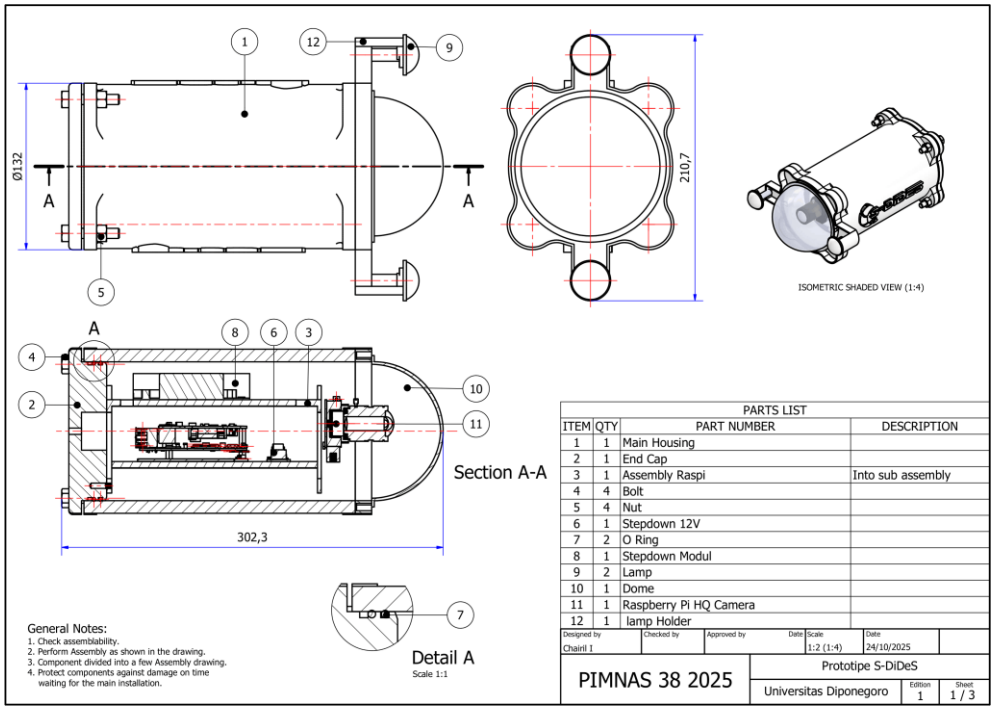
Lampiran 1 Skema Rangkaian Alat



Lampiran 2 Rancangan Prototipe







Lampiran 3 Program Disesase Detail Aplikasi S-dides

```
import 'package:flutter/material.dart';

class DiseaseDetailScreen extends StatelessWidget {
```

```
  final String diseaseName;
  final String diseaseCode;
  final String status;
  final Color statusColor;
  final IconData icon;
  final Color iconColor;
  final Color iconBgColor;

  const DiseaseDetailScreen({
    super.key,
    required this.diseaseName,
    required this.diseaseCode,
    required this.status,
    required this.statusColor,
    required this.icon,
    required this.iconColor,
    required this.iconBgColor,
  });

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[100],
      appBar: AppBar(
        backgroundColor: const Color(0xFF00BFA5),
        elevation: 0,
        leading: IconButton(
          icon: const Icon(Icons.arrow_back, color: Colors.white),
          onPressed: () => Navigator.pop(context),
        ),
        title: const Text(
          'Detail Penyakit',
          style: TextStyle(color: Colors.white, fontWeight:
FontWeight.bold),
        ),
      ),
      body: SingleChildScrollView(
        child: Column(
          children: [
            // Header Card
            Container(
              width: double.infinity,
              padding: const EdgeInsets.all(24),
              decoration: const BoxDecoration(
                gradient: LinearGradient(
```

```
                colors: [Color(0xFF00BFA5), Color(0xFF00897B)],
                begin: Alignment.topLeft,
                end: Alignment.bottomRight,
              ),
            ),
            child: Column(
              children: [
                Container(
                  padding: const EdgeInsets.all(20),
                  decoration: BoxDecoration(
                    color: iconBgColor,
                    borderRadius: BorderRadius.circular(20),
                    boxShadow: [
                      BoxShadow(
                        color: Colors.black.withAlpha(25),
                        blurRadius: 15,
                        offset: const Offset(0, 5),
                      ),
                    ],
                  ),
                  child: Icon(
                    icon,
                    color: iconColor,
                    size: 48,
                  ),
                ),
                const SizedBox(height: 16),
                Text(
                  diseaseName,
                  textAlign: TextAlign.center,
                  style: const TextStyle(
                    fontSize: 20,
                    fontWeight: FontWeight.bold,
                    color: Colors.white,
                  ),
                ),
                const SizedBox(height: 8),
                Text(
                  diseaseCode,
                  style: TextStyle(
                    fontSize: 14,
                    color: Colors.white.withAlpha(200),
                  ),
                ),
                const SizedBox(height: 12),
```

```
                Container(
                  padding: const EdgeInsets.symmetric(horizontal:
16, vertical: 8),
                  decoration: BoxDecoration(
                    color: statusColor,
                    borderRadius: BorderRadius.circular(20),
                  ),
                  child: Text(
                    'Status: $status',
                    style: const TextStyle(
                      color: Colors.white,
                      fontSize: 14,
                      fontWeight: FontWeight.bold,
                    ),
                  ),
                ),
              ],
            ),
          ),

          // Information Sections
          Padding(
            padding: const EdgeInsets.all(16),
            child: Column(
              children: [
                // Deskripsi
                _buildInfoCard(
                  title: 'Deskripsi',
                  icon: Icons.description,
                  content: _getDescription(diseaseCode),
                ),
                const SizedBox(height: 12),

                // Gejala
                _buildInfoCard(
                  title: 'Gejala Klinis',
                  icon: Icons.medical_information,
                  content: _getSymptoms(diseaseCode),
                ),
                const SizedBox(height: 12),

                // Penyebab
                _buildInfoCard(
                  title: 'Penyebab',
                  icon: Icons.coronavirus,
```

```
                  content: _getCauses(diseaseCode),
                ),
                const SizedBox(height: 12),

                // Pencegahan
                _buildInfoCard(
                  title: 'Pencegahan',
                  icon: Icons.shield,
                  content: _getPrevention(diseaseCode),
                ),
                const SizedBox(height: 12),

                // Penanganan
                _buildInfoCard(
                  title: 'Penanganan',
                  icon: Icons.healing,
                  content: _getTreatment(diseaseCode),
                ),
                const SizedBox(height: 12),

                // Statistics Card
                Container(
                  padding: const EdgeInsets.all(20),
                  decoration: BoxDecoration(
                    color: Colors.white,
                    borderRadius: BorderRadius.circular(16),
                    boxShadow: [
                      BoxShadow(
                        color: Colors.black.withAlpha(13),
                        blurRadius: 10,
                        offset: const Offset(0, 2),
                      ),
                    ],
                  ),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Row(
                        children: [
                          Container(
                            padding: const EdgeInsets.all(8),
                            decoration: BoxDecoration(
                              color: const
Color(0xFF00BFA5).withAlpha(25),
```

```
                                borderRadius:
BorderRadius.circular(8),
                              ),
                              child: const Icon(
                                Icons.bar_chart,
                                color: Color(0xFF00BFA5),
                                size: 20,
                              ),
                            ),
                            const SizedBox(width: 12),
                            const Text(
                              'Statistik',
                              style: TextStyle(
                                fontSize: 16,
                                fontWeight: FontWeight.bold,
                              ),
                            ),
                          ],
                        ),
                        const SizedBox(height: 16),
                        _buildStatRow('Tingkat Keparahan',
_getSeverityLevel(diseaseCode)),
                        const Divider(height: 24),
                        _buildStatRow('Tingkat Penyebaran',
_getSpreadRate(diseaseCode)),
                        const Divider(height: 24),
                        _buildStatRow('Tingkat Mortalitas',
_getMortalityRate(diseaseCode)),
                      ],
                    ),
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    );
  }

  Widget _buildInfoCard({
    required String title,
    required IconData icon,
    required String content,
  }) {
```

```
    return Container(
      width: double.infinity,
      padding: const EdgeInsets.all(20),
      decoration: BoxDecoration(
        color: Colors.white,
        borderRadius: BorderRadius.circular(16),
        boxShadow: [
          BoxShadow(
            color: Colors.black.withAlpha(13),
            blurRadius: 10,
            offset: const Offset(0, 2),
          ),
        ],
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Row(
            children: [
              Container(
                padding: const EdgeInsets.all(8),
                decoration: BoxDecoration(
                  color: const Color(0xFF00BFA5).withAlpha(25),
                  borderRadius: BorderRadius.circular(8),
                ),
                child: Icon(
                  icon,
                  color: const Color(0xFF00BFA5),
                  size: 20,
                ),
              ),
              const SizedBox(width: 12),
              Text(
                title,
                style: const TextStyle(
                  fontSize: 16,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ],
          ),
          const SizedBox(height: 12),
          Text(
            content,
            style: TextStyle(
```

```
              fontSize: 14,
              color: Colors.grey[700],
              height: 1.6,
            ),
          ),
        ],
      ),
    );
  }

  Widget _buildStatRow(String label, String value) {
    return Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: [
        Text(
          label,
          style: TextStyle(
            fontSize: 14,
            color: Colors.grey[600],
          ),
        ),
        Text(
          value,
          style: const TextStyle(
            fontSize: 14,
            fontWeight: FontWeight.bold,
            color: Color(0xFF00BFA5),
          ),
        ),
      ],
    );
  }

  // Data methods
  String _getDescription(String code) {
    switch (code) {
      case '(WSSV)':
        return 'White Spot Syndrome Virus (WSSV) adalah virus DNA
yang sangat menular dan mematikan yang menyerang udang. Virus ini
dapat menyebabkan kematian massal hingga 100% dalam waktu 3-10 hari
setelah gejala pertama muncul.';
      case '(TSV)':
        return 'Taura Syndrome Virus (TSV) adalah penyakit virus
yang menyerang udang, terutama udang putih Pasifik. Penyakit ini
```

```
pertama kali ditemukan di dekat Sungai Taura, Ekuador pada tahun
1992.';
      case '(WFD)':
        return 'White Feces Disease (WFD) adalah penyakit yang
ditandai dengan kotoran berwarna putih mengambang di permukaan air.
Penyakit ini dapat menyebabkan pertumbuhan lambat dan kematian pada
udang.';
      case '(IMNV)':
        return 'Infectious Myonecrosis Virus (IMNV) adalah virus
yang menyebabkan nekrosis pada otot udang. Penyakit ini pertama kali
ditemukan di Brasil dan dapat menyebabkan kerugian ekonomi yang
signifikan.';
      default:
        return 'Informasi tidak tersedia.';
    }
  }

  String _getSymptoms(String code) {
    switch (code) {
      case '(WSSV)':
        return '• Bintik putih pada karapas dan kulit\n• Udang
berenang ke permukaan air\n• Nafsu makan menurun drastis\n• Warna
tubuh menjadi kemerahan\n• Karapas lunak dan mudah dilepas\n•
Kematian mendadak dalam jumlah besar';
      case '(TSV)':
        return '• Kulit lunak (soft shell)\n• Ekor berwarna merah\n•
Perubahan warna pada hepatopankreas\n• Udang letargi dan lemah\n•
Melanisasi atau bintik hitam pada kulit\n• Penurunan nafsu makan';
      case '(WFD)':
        return '• Kotoran berwarna putih mengambang\n•
Hepatopankreas pucat dan mengecil\n• Pertumbuhan lambat\n• Usus
kosong dan berwarna putih\n• Nafsu makan berkurang\n• Tingkat
kelangsungan hidup rendah';
      case '(IMNV)':
        return '• Nekrosis fokal pada otot\n• Otot berwarna putih
keruh\n• Udang lemah dan letargi\n• Kesulitan berenang\n• Nekrosis
pada ekor\n• Penurunan bobot tubuh';
      default:
        return 'Informasi gejala tidak tersedia.';
    }
  }

  String _getCauses(String code) {
    switch (code) {
      case '(WSSV)':
```

```
        return 'Disebabkan oleh White Spot Syndrome Virus (genus
Whispovirus). Penularan terjadi melalui air, kontak langsung,
kanibalisme, dan vektor seperti zooplankton. Faktor pemicu: kualitas
air buruk, stres, kepadatan tinggi.';
      case '(TSV)':
        return 'Disebabkan oleh Taura Syndrome Virus (genus
Aparavirus). Penularan melalui air dan kontak horizontal antar
udang. Faktor pemicu: stres lingkungan, perubahan salinitas
mendadak, kualitas benur.';
      case '(WFD)':
        return 'Penyebab pasti belum diketahui secara pasti, diduga
kombinasi infeksi mikroba (Vibrio, Microsporidium) dan kondisi
lingkungan. Faktor pemicu: kualitas pakan, kualitas air, manajemen
kolam.';
      case '(IMNV)':
        return 'Disebabkan oleh Infectious Myonecrosis Virus (genus
Totiviridae). Penularan vertikal (induk ke benur) dan horizontal.
Faktor risiko: suhu air rendah, kualitas benur, manajemen tambak.';
      default:
        return 'Informasi penyebab tidak tersedia.';
    }
  }

  String _getPrevention(String code) {
    switch (code) {
      case '(WSSV)':
        return '• Gunakan benur SPF (Specific Pathogen Free)\n•
Terapkan biosekuriti ketat\n• Jaga kualitas air optimal\n• Hindari
kepadatan tinggi\n• Disinfeksi peralatan secara rutin\n• Screening
benur sebelum tebar';
      case '(TSV)':
        return '• Gunakan benur SPR (Specific Pathogen Resistant)\n•
Aklimatisasi benur dengan baik\n• Jaga stabilitas salinitas\n•
Monitoring kesehatan rutin\n• Karantina udang baru\n• Manajemen
stres yang baik';
      case '(WFD)':
        return '• Jaga kualitas pakan (probiotik)\n• Manajemen
kualitas air optimal\n• Hindari overfeeding\n• Gunakan aerasi yang
cukup\n• Monitoring plankton\n• Sanitasi kolam yang baik';
      case '(IMNV)':
        return '• Seleksi induk bebas IMNV\n• Kontrol suhu air
(hindari < 18°C)\n• Terapkan biosekuriti\n• Monitoring PCR secara
berkala\n• Manajemen nutrisi baik\n• Hindari stres lingkungan';
      default:
        return 'Informasi pencegahan tidak tersedia.';
```

```
    }
  }

  String _getTreatment(String code) {
    switch (code) {
      case '(WSSV)':
        return '• Tidak ada pengobatan spesifik untuk virus\n• Panen
awal (emergency harvest)\n• Isolasi kolam terinfeksi\n• Disinfeksi
total setelah panen\n• Pengeringan kolam minimal 2 minggu\n• Buang
udang mati segera';
      case '(TSV)':
        return '• Tidak ada pengobatan antivirus\n• Kurangi feeding
rate\n• Tingkatkan aerasi\n• Jaga kualitas air optimal\n• Pemberian
imunostimulan\n• Partial harvest jika perlu';
      case '(WFD)':
        return '• Aplikasi probiotik melalui pakan dan air\n•
Perbaiki kualitas air\n• Kurangi jumlah pakan bertahap\n• Tingkatkan
aerasi\n• Siphon untuk buang kotoran\n• Pemberian suplemen vitamin
C';
      case '(IMNV)':
        return '• Tidak ada pengobatan spesifik\n• Jaga suhu air
stabil > 20°C\n• Perbaiki nutrisi pakan\n• Kurangi kepadatan\n•
Tingkatkan kualitas air\n• Panen selektif udang sakit';
      default:
        return 'Informasi penanganan tidak tersedia.';
    }
  }

  String _getSeverityLevel(String code) {
    switch (code) {
      case '(WSSV)':
        return 'Sangat Tinggi';
      case '(TSV)':
        return 'Tinggi';
      case '(WFD)':
        return 'Sedang-Tinggi';
      case '(IMNV)':
        return 'Sangat Tinggi';
      default:
        return 'N/A';
    }
  }

  String _getSpreadRate(String code) {
    switch (code) {
```

```
      case '(WSSV)':
        return 'Sangat Cepat';
      case '(TSV)':
        return 'Cepat';
      case '(WFD)':
        return 'Sedang';
      case '(IMNV)':
        return 'Sedang-Cepat';
      default:
        return 'N/A';
    }
  }

  String _getMortalityRate(String code) {
    switch (code) {
      case '(WSSV)':
        return '80-100%';
      case '(TSV)':
        return '40-90%';
      case '(WFD)':
        return '20-60%';
      case '(IMNV)':
        return '40-70%';
      default:
        return 'N/A';
    }
  }
}
```

Lampiran 4 Program Monitor Aplikasi S-dides

```
import 'package:flutter/material.dart';
import 'package:camera/camera.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
import 'dart:io';
import 'settings_screen.dart';

class MonitorScreen extends StatefulWidget {
  const MonitorScreen({super.key});

  @override
  State<MonitorScreen> createState() => _MonitorScreenState();
}
```

```
class _MonitorScreenState extends State<MonitorScreen> {
  CameraController? _cameraController;
  List<CameraDescription>? _cameras;
  bool _isCameraInitialized = false;
  bool _isDetecting = false;
  String _serverUrl = '';

  @override
  void initState() {
    super.initState();
    _loadServerUrl();
    _initializeCamera();
  }

  Future<void> _loadServerUrl() async {
    final prefs = await SharedPreferences.getInstance();
    final url = prefs.getString('server_url') ??
'http://192.168.1.100';
    final port = prefs.getString('server_port') ?? '5000';
    setState(() {
      _serverUrl = '$url:$port';
    });
  }

  Future<void> _initializeCamera() async {
    try {
      _cameras = await availableCameras();
      if (_cameras != null && _cameras!.isNotEmpty) {
        _cameraController = CameraController(
          _cameras![0],
          ResolutionPreset.high,
          enableAudio: false,
        );

        await _cameraController!.initialize();
        if (mounted) {
          setState(() {
            _isCameraInitialized = true;
          });
        }
      }
    } catch (e) {
      debugPrint('Error initializing camera: $e');
    }
```

```
  }

  Future<void> _startDetection() async {
    if (_cameraController == null ||
!_cameraController!.value.isInitialized) {
      _showSnackBar('Kamera belum siap', Colors.red);
      return;
    }

    if (_serverUrl.isEmpty) {
      _showSnackBar('URL server belum dikonfigurasi',
Colors.orange);
      return;
    }

    setState(() => _isDetecting = true);

    try {
      // Capture image from camera
      final XFile imageFile = await
_cameraController!.takePicture();

      // Send to server for detection
      await _sendImageToServer(imageFile);

    } catch (e) {
      _showSnackBar('Gagal mengambil gambar: $e', Colors.red);
    } finally {
      setState(() => _isDetecting = false);
    }
  }

  Future<void> _sendImageToServer(XFile imageFile) async {
    try {
      final uri = Uri.parse('$_serverUrl/detect');
      final request = http.MultipartRequest('POST', uri);

      // Add image file to request
      final file = await http.MultipartFile.fromPath(
        'image',
        imageFile.path,
      );
      request.files.add(file);

      _showSnackBar('Mengirim ke server...', Colors.blue);
```

```
      final streamedResponse = await request.send().timeout(
        const Duration(seconds: 30),
      );

      final response = await
http.Response.fromStream(streamedResponse);

      if (response.statusCode == 200) {
        final result = json.decode(response.body);
        _showDetectionResult(result);
      } else {
        _showSnackBar('Server error: ${response.statusCode}',
Colors.red);
      }

    } catch (e) {
      _showSnackBar('Gagal mengirim ke server: $e', Colors.red);
    }
  }

  void _showDetectionResult(Map<String, dynamic> result) {
    // Parse detection result
    final detected = result['detected'] ?? false;
    final disease = result['disease'] ?? 'Unknown';
    final confidence = result['confidence'] ?? 0.0;

    showDialog(
      context: context,
      builder: (context) => AlertDialog(
        title: Row(
          children: [
            Icon(
              detected ? Icons.warning_amber : Icons.check_circle,
              color: detected ? Colors.red : Colors.green,
            ),
            const SizedBox(width: 8),
            const Text('Hasil Deteksi'),
          ],
        ),
        content: Column(
          mainAxisSize: MainAxisSize.min,
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
```

```
            Text('Status: ${detected ? "Terdeteksi Penyakit" :
"Sehat"}'),
            if (detected) ...[
              const SizedBox(height: 8),
              Text('Penyakit: $disease'),
              Text('Confidence: ${(confidence *
100).toStringAsFixed(1)}%'),
            ],
          ],
        ),
        actions: [
          TextButton(
            onPressed: () => Navigator.pop(context),
            child: const Text('OK'),
          ),
        ],
      ),
    );
  }

  void _showSnackBar(String message, Color color) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text(message),
        backgroundColor: color,
        behavior: SnackBarBehavior.floating,
        duration: const Duration(seconds: 2),
      ),
    );
  }

  @override
  void dispose() {
    _cameraController?.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[100],
      body: SingleChildScrollView(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.start,
          children: [
```

```
            // Live Camera Section
            Container(
              margin: const EdgeInsets.all(16),
              padding: const EdgeInsets.all(20),
              decoration: BoxDecoration(
                color: Colors.white,
                borderRadius: BorderRadius.circular(16),
                boxShadow: [
                  BoxShadow(
                    color: Colors.black.withAlpha(13),
                    blurRadius: 10,
                    offset: const Offset(0, 2),
                  ),
                ],
              ),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Row(
                    children: [
                      Container(
                        padding: const EdgeInsets.all(8),
                        decoration: BoxDecoration(
                          color: const Color(0xFF00BFA5),
                          borderRadius: BorderRadius.circular(8),
                        ),
                        child: const Icon(
                          Icons.camera_alt,
                          color: Colors.white,
                          size: 20,
                        ),
                      ),
                      const SizedBox(width: 12),
                      const Expanded(
                        child: Column(
                          crossAxisAlignment:
CrossAxisAlignment.start,
                          children: [
                            Text(
                              'Live Camera',
                              style: TextStyle(
                                fontSize: 18,
                                fontWeight: FontWeight.bold,
                              ),
                            ),
```

```
                          Text(
                            'AI-Powered Detection',
                            style: TextStyle(
                              fontSize: 12,
                              color: Colors.grey,
                            ),
                          ),
                        ],
                      ),
                    ),
                    // Settings Button
                    IconButton(
                      onPressed: () async {
                        final result = await Navigator.push(
                          context,
                          MaterialPageRoute(
                            builder: (context) => const
SettingsScreen(),
                          ),
                        );
                        if (result != null) {
                          await _loadServerUrl();
                        }
                      },
                      icon: const Icon(Icons.settings),
                      color: const Color(0xFF00BFA5),
                      tooltip: 'Pengaturan Server',
                    ),
                  ],
                ),
                const SizedBox(height: 16),
                // Camera Preview
                Container(
                  height: 200,
                  decoration: BoxDecoration(
                    color: const Color(0xFF1A1F2E),
                    borderRadius: BorderRadius.circular(12),
                  ),
                  child: ClipRRect(
                    borderRadius: BorderRadius.circular(12),
                    child: _isCameraInitialized &&
_cameraController != null
                        ? Stack(
                            children: [
                              SizedBox.expand(
```

```
                                  child: FittedBox(
                                    fit: BoxFit.cover,
                                    child: SizedBox(
                                      width:
_cameraController!.value.previewSize?.height ?? 1,
                                      height:
_cameraController!.value.previewSize?.width ?? 1,
                                      child:
CameraPreview(_cameraController!),
                                    ),
                                  ),
                                ),
                                Positioned(
                                  top: 12,
                                  left: 12,
                                  child: Container(
                                    padding: const
EdgeInsets.symmetric(
                                      horizontal: 8,
                                      vertical: 4,
                                    ),
                                    decoration: BoxDecoration(
                                      color: Colors.red,
                                      borderRadius:
BorderRadius.circular(4),
                                    ),
                                    child: const Row(
                                      children: [
                                        Icon(
                                          Icons.circle,
                                          color: Colors.white,
                                          size: 8,
                                        ),
                                        SizedBox(width: 4),
                                        Text(
                                          'LIVE',
                                          style: TextStyle(
                                            color: Colors.white,
                                            fontSize: 10,
                                            fontWeight:
FontWeight.bold,
                                          ),
                                        ),
                                      ],
                                    ),
```

```
                                ),
                              ),
                              Positioned(
                                top: 12,
                                right: 12,
                                child: Container(
                                  padding: const
EdgeInsets.symmetric(
                                    horizontal: 8,
                                    vertical: 4,
                                  ),
                                  decoration: BoxDecoration(
                                    color:
Colors.black.withAlpha(127),
                                    borderRadius:
BorderRadius.circular(4),
                                  ),
                                  child: const Text(
                                    '4K',
                                    style: TextStyle(
                                      color: Colors.white,
                                      fontSize: 10,
                                      fontWeight: FontWeight.bold,
                                    ),
                                  ),
                                ),
                              ),
                            ],
                          )
                        : Center(
                            child: Column(
                              mainAxisAlignment:
MainAxisAlignment.center,
                              children: [
                                const CircularProgressIndicator(
                                  color: Color(0xFF00BFA5),
                                ),
                                const SizedBox(height: 12),
                                Text(
                                  'Memuat kamera...',
                                  style: TextStyle(
                                    color:
Colors.white.withAlpha(178),
                                    fontSize: 14,
                                  ),
```

```
                                ),
                              ],
                            ),
                          ),
                    ),
                  ),
                  const SizedBox(height: 16),
                  // Start Detection Button
                  SizedBox(
                    width: double.infinity,
                    child: ElevatedButton(
                      onPressed: _isDetecting ? null :
_startDetection,
                      style: ElevatedButton.styleFrom(
                        backgroundColor: const Color(0xFF00BFA5),
                        padding: const
EdgeInsets.symmetric(vertical: 16),
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(12),
                        ),
                        disabledBackgroundColor: Colors.grey,
                      ),
                      child: _isDetecting
                          ? const SizedBox(
                              height: 20,
                              width: 20,
                              child: CircularProgressIndicator(
                                color: Colors.white,
                                strokeWidth: 2,
                              ),
                            )
                          : const Row(
                              mainAxisAlignment:
MainAxisAlignment.center,
                              children: [
                                Icon(Icons.play_arrow, color:
Colors.white),
                                SizedBox(width: 8),
                                Text(
                                  'Start Detection',
                                  style: TextStyle(
                                    fontSize: 16,
                                    fontWeight: FontWeight.bold,
                                    color: Colors.white,
                                  ),
```

```
                              ),
                            ],
                          ),
                      ),
                    ),
                  ],
                ),
              ),

              // Stats Cards
              Padding(
                padding: const EdgeInsets.symmetric(horizontal: 16),
                child: Row(
                  children: [
                    Expanded(
                      child: Container(
                        padding: const EdgeInsets.all(16),
                        decoration: BoxDecoration(
                          color: const Color(0xFFE8F5E9),
                          borderRadius: BorderRadius.circular(12),
                        ),
                        child: Column(
                          crossAxisAlignment:
CrossAxisAlignment.start,
                          children: [
                            Row(
                              children: [
                                Container(
                                  padding: const EdgeInsets.all(6),
                                  decoration: BoxDecoration(
                                    color: const Color(0xFF00BFA5),
                                    borderRadius:
BorderRadius.circular(6),
                                  ),
                                  child: const Icon(
                                    Icons.check_circle,
                                    color: Colors.white,
                                    size: 16,
                                  ),
                                ),
                                const SizedBox(width: 8),
                                const Text(
                                  'Sehat Hari Ini',
                                  style: TextStyle(
                                    fontSize: 12,
```

```
                                color: Color(0xFF00BFA5),
                                fontWeight: FontWeight.w500,
                              ),
                            ),
                          ],
                        ),
                        const SizedBox(height: 8),
                        const Text(
                          '0',
                          style: TextStyle(
                            fontSize: 32,
                            fontWeight: FontWeight.bold,
                            color: Color(0xFF00BFA5),
                          ),
                        ),
                        const Text(
                          '↑ 0% dari kemarin',
                          style: TextStyle(
                            fontSize: 10,
                            color: Color(0xFF00BFA5),
                          ),
                        ),
                      ],
                    ),
                  ),
                ),
                const SizedBox(width: 12),
                Expanded(
                  child: Container(
                    padding: const EdgeInsets.all(16),
                    decoration: BoxDecoration(
                      color: const Color(0xFFFFEBEE),
                      borderRadius: BorderRadius.circular(12),
                    ),
                    child: Column(
                      crossAxisAlignment:
CrossAxisAlignment.start,
                      children: [
                        Row(
                          children: [
                            Container(
                              padding: const EdgeInsets.all(6),
                              decoration: BoxDecoration(
                                color: Colors.red,
```

```
                                borderRadius:
BorderRadius.circular(6),
                              ),
                              child: const Icon(
                                Icons.cancel,
                                color: Colors.white,
                                size: 16,
                              ),
                            ),
                            const SizedBox(width: 8),
                            const Flexible(
                              child: Text(
                                'Terdeteksi Sakit',
                                style: TextStyle(
                                  fontSize: 12,
                                  color: Colors.red,
                                  fontWeight: FontWeight.w500,
                                ),
                              ),
                            ),
                          ],
                        ),
                        const SizedBox(height: 8),
                        const Text(
                          '0',
                          style: TextStyle(
                            fontSize: 32,
                            fontWeight: FontWeight.bold,
                            color: Colors.red,
                          ),
                        ),
                        const Text(
                          'Butuh perhatian',
                          style: TextStyle(
                            fontSize: 10,
                            color: Colors.red,
                          ),
                        ),
                      ],
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
```

```
          const SizedBox(height: 16),

          // Recent Detection Section
          Container(
            margin: const EdgeInsets.all(16),
            padding: const EdgeInsets.all(20),
            decoration: BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.circular(16),
              boxShadow: [
                BoxShadow(
                  color: Colors.black.withAlpha(13),
                  blurRadius: 10,
                  offset: const Offset(0, 2),
                ),
              ],
            ),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Row(
                  children: [
                    Container(
                      padding: const EdgeInsets.all(8),
                      decoration: BoxDecoration(
                        color: const Color(0xFF00BFA5),
                        borderRadius: BorderRadius.circular(8),
                      ),
                      child: const Icon(
                        Icons.auto_graph,
                        color: Colors.white,
                        size: 20,
                      ),
                    ),
                    const SizedBox(width: 12),
                    const Text(
                      'Deteksi Terbaru',
                      style: TextStyle(
                        fontSize: 18,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                  ],
                ),
```

```
                const SizedBox(height: 16),
                Center(
                  child: Column(
                    children: [
                      Icon(
                        Icons.show_chart,
                        size: 48,
                        color: Colors.grey[300],
                      ),
                      const SizedBox(height: 8),
                      Text(
                        'Belum ada deteksi hari ini',
                        style: TextStyle(
                          color: Colors.grey[400],
                          fontSize: 14,
                        ),
                      ),
                    ],
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    ),
  );
  }
}
```

Lampiran 5 Program Penyakitl Aplikasi S-dides

```
import 'package:flutter/material.dart';
import 'disease_detail_screen.dart';

class PenyakitScreen extends StatelessWidget {
  const PenyakitScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[100],
      body: SingleChildScrollView(
        child: Column(
          children: [
```

```
            // Database Penyakit Section
            Container(
              margin: const EdgeInsets.all(16),
              padding: const EdgeInsets.all(20),
              decoration: BoxDecoration(
                color: Colors.white,
                borderRadius: BorderRadius.circular(16),
                boxShadow: [
                  BoxShadow(
                    color: Colors.black.withAlpha(13),
                    blurRadius: 10,
                    offset: const Offset(0, 2),
                  ),
                ],
              ),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Row(
                    children: [
                      Icon(
                        Icons.warning_amber_rounded,
                        color: Colors.red[700],
                        size: 24,
                      ),
                      const SizedBox(width: 12),
                      const Text(
                        'Database Penyakit',
                        style: TextStyle(
                          fontSize: 18,
                          fontWeight: FontWeight.bold,
                        ),
                      ),
                    ],
                  ),
                  const SizedBox(height: 20),

                  // Disease List
                  _buildDiseaseItem(
                    context,
                    icon: Icons.coronavirus,
                    iconColor: const Color(0xFF4CAF50),
                    iconBgColor: const Color(0xFFE8F5E9),
                    title: 'White Spot Syndrome Virus',
                    subtitle: '(WSSV)',
```

```
                  status: 'Kritis',
                  statusColor: Colors.red,
                ),
                const SizedBox(height: 12),
                _buildDiseaseItem(
                  context,
                  icon: Icons.warning,
                  iconColor: Colors.orange,
                  iconBgColor: Colors.orange.shade50,
                  title: 'Taura Syndrome Virus',
                  subtitle: '(TSV)',
                  status: 'Tinggi',
                  statusColor: Colors.orange,
                ),
                const SizedBox(height: 12),
                _buildDiseaseItem(
                  context,
                  icon: Icons.search,
                  iconColor: Colors.blue,
                  iconBgColor: Colors.blue.shade50,
                  title: 'White Feces Disease',
                  subtitle: '(WFD)',
                  status: 'Sedang',
                  statusColor: Colors.amber[700]!,
                ),
                const SizedBox(height: 12),
                _buildDiseaseItem(
                  context,
                  icon: Icons.local_hospital,
                  iconColor: Colors.red,
                  iconBgColor: Colors.red.shade50,
                  title: 'Infectious Myonecrosis Virus',
                  subtitle: '(IMNV)',
                  status: 'Kritis',
                  statusColor: Colors.red,
                ),
              ],
            ),
          ),
        ],
      ),
    ),
  );
}
```

```
Widget _buildDiseaseItem(
  BuildContext context, {
  required IconData icon,
  required Color iconColor,
  required Color iconBgColor,
  required String title,
  required String subtitle,
  required String status,
  required Color statusColor,
}) {
  return InkWell(
    onTap: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => DiseaseDetailScreen(
            diseaseName: title,
            diseaseCode: subtitle,
            status: status,
            statusColor: statusColor,
            icon: icon,
            iconColor: iconColor,
            iconBgColor: iconBgColor,
          ),
        ),
      );
    },
    borderRadius: BorderRadius.circular(12),
    child: Container(
      padding: const EdgeInsets.all(16),
      decoration: BoxDecoration(
        color: Colors.grey[50],
        borderRadius: BorderRadius.circular(12),
        border: Border.all(color: Colors.grey.shade200),
      ),
      child: Row(
        children: [
          Container(
            padding: const EdgeInsets.all(10),
            decoration: BoxDecoration(
              color: iconBgColor,
              borderRadius: BorderRadius.circular(10),
            ),
            child: Icon(
              icon,
```

```
              color: iconColor,
              size: 24,
            ),
          ),
          const SizedBox(width: 12),
          Expanded(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                Text(
                  title,
                  style: const TextStyle(
                    fontSize: 14,
                    fontWeight: FontWeight.w600,
                  ),
                ),
                const SizedBox(height: 2),
                Text(
                  subtitle,
                  style: TextStyle(
                    fontSize: 12,
                    color: Colors.grey[600],
                  ),
                ),
              ],
            ),
          ),
          Container(
            padding: const EdgeInsets.symmetric(horizontal: 12,
vertical: 6),
            decoration: BoxDecoration(
              color: statusColor,
              borderRadius: BorderRadius.circular(20),
            ),
            child: Text(
              status,
              style: const TextStyle(
                color: Colors.white,
                fontSize: 12,
                fontWeight: FontWeight.bold,
              ),
            ),
          ),
          const SizedBox(width: 8),
          Icon(
```

```
              Icons.arrow_forward_ios,
              size: 16,
              color: Colors.grey[400],
            ),
          ],
        ),
      ),
    );
  }
}
```

Lampiran 6 Program Riwayat Aplikasi S-dides

```
import 'package:flutter/material.dart';

class RiwayatScreen extends StatefulWidget {
  const RiwayatScreen({super.key});

  @override
  State<RiwayatScreen> createState() => _RiwayatScreenState();
}

class _RiwayatScreenState extends State<RiwayatScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[100],
      body: SingleChildScrollView(
        child: Column(
          children: [
            Container(
              margin: const EdgeInsets.all(16),
              padding: const EdgeInsets.all(20),
              decoration: BoxDecoration(
                color: Colors.white,
                borderRadius: BorderRadius.circular(16),
                boxShadow: [
                  BoxShadow(
                    color: Colors.black.withAlpha(13),
                    blurRadius: 10,
                    offset: const Offset(0, 2),
                  ),
                ],
              ),
              width: double.infinity,
```

```
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                Row(
                  children: [
                    Container(
                      padding: const EdgeInsets.all(8),
                      child: const Icon(
                        Icons.schedule,
                        color: Colors.blue,
                        size: 26,
                      ),
                    ),
                    const SizedBox(width: 5),
                    const Text(
                      'Riwayat Deteksi',
                      style: TextStyle(
                        fontSize: 18,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                  ],
                ),
                const SizedBox(height: 20),
                // Riwayat List
                _buildEmptyState(),
              ],
            ),
          ),
        ],
      ),
    ),
  );
}

Widget _buildEmptyState() {
  return Container(
    padding: const EdgeInsets.all(40),
    child: Column(
      children: [
        Icon(Icons.schedule, size: 80, color: Colors.grey[300]),
        const SizedBox(height: 16),
        Text(
          'Belum ada riwayat',
          style: TextStyle(
```

```
              fontSize: 16,
              fontWeight: FontWeight.w500,
              color: Colors.grey[600],
            ),
          ),
          const SizedBox(height: 8),
          Text(
            'Mulai deteksi untuk melihat riwayat',
            style: TextStyle(fontSize: 14, color: Colors.grey[500]),
          ),
        ],
      ),
    );
  }
}
```

Lampiran 7 Program Pengaturan Aplikasi S-dides

```
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:http/http.dart' as http;
import 'dart:async';

class SettingsScreen extends StatefulWidget {
  const SettingsScreen({super.key});

  @override
  State<SettingsScreen> createState() => _SettingsScreenState();
}

class _SettingsScreenState extends State<SettingsScreen> {
  final TextEditingController _serverUrlController =
TextEditingController();
  final TextEditingController _portController =
TextEditingController();
  bool _isLoading = true;
  bool _isSaving = false;
  String _connectionStatus = 'Belum diperiksa';
  Color _statusColor = Colors.grey;

  @override
  void initState() {
    super.initState();
    _loadSettings();
  }
```

```
  Future<void> _loadSettings() async {
    setState(() => _isLoading = true);
    try {
      final prefs = await SharedPreferences.getInstance();
      _serverUrlController.text = prefs.getString('server_url') ??
'http://192.168.1.100';
      _portController.text = prefs.getString('server_port') ??
'5000';
    } catch (e) {
      _showSnackBar('Gagal memuat pengaturan: $e', Colors.red);
    } finally {
      setState(() => _isLoading = false);
    }
  }

  Future<void> _saveSettings() async {
    if (_serverUrlController.text.isEmpty ||
_portController.text.isEmpty) {
      _showSnackBar('URL Server dan Port tidak boleh kosong',
Colors.orange);
      return;
    }

    setState(() => _isSaving = true);
    try {
      final prefs = await SharedPreferences.getInstance();
      await prefs.setString('server_url',
_serverUrlController.text);
      await prefs.setString('server_port', _portController.text);
      _showSnackBar('Pengaturan berhasil disimpan', const
Color(0xFF00BFA5));

      // Return the new URL to the previous screen
      if (mounted) {
        Navigator.pop(context, getFullServerUrl());
      }
    } catch (e) {
      _showSnackBar('Gagal menyimpan pengaturan: $e', Colors.red);
    } finally {
      setState(() => _isSaving = false);
    }
  }

  String getFullServerUrl() {
```

```
    return '${_serverUrlController.text}:${_portController.text}';
  }

  Future<void> _testConnection() async {
    if (_serverUrlController.text.isEmpty ||
_portController.text.isEmpty) {
      _showSnackBar('Masukkan URL Server dan Port terlebih dahulu',
Colors.orange);
      return;
    }

    setState(() {
      _connectionStatus = 'Memeriksa koneksi...';
      _statusColor = Colors.orange;
    });

    try {
      // Attempt to connect to server with timeout
      final url = Uri.parse('${getFullServerUrl()}');

      final response = await http.get(url).timeout(
        const Duration(seconds: 5),
        onTimeout: () {
          throw TimeoutException('Koneksi timeout');
        },
      );

      if (response.statusCode == 200) {
        setState(() {
          _connectionStatus = 'Terhubung';
          _statusColor = const Color(0xFF00BFA5);
        });
        _showSnackBar('Koneksi berhasil! Server merespons dengan
baik.', const Color(0xFF00BFA5));
      } else {
        setState(() {
          _connectionStatus = 'Gagal (Status:
${response.statusCode})';
          _statusColor = Colors.red;
        });
        _showSnackBar('Server merespons dengan error:
${response.statusCode}', Colors.red);
      }
    } on TimeoutException {
      setState(() {
```

```
        _connectionStatus = 'Timeout';
        _statusColor = Colors.red;
      });
      _showSnackBar('Koneksi timeout. Pastikan server aktif dan URL
benar.', Colors.red);
    } catch (e) {
      setState(() {
        _connectionStatus = 'Gagal terhubung';
        _statusColor = Colors.red;
      });
      _showSnackBar('Gagal terhubung: ${e.toString()}', Colors.red);
    }
  }

  void _showSnackBar(String message, Color color) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text(message),
        backgroundColor: color,
        behavior: SnackBarBehavior.floating,
        margin: const EdgeInsets.all(16),
        shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(8)),
      ),
    );
  }

  void _resetToDefault() {
    setState(() {
      _serverUrlController.text = 'http://192.168.1.100';
      _portController.text = '5000';
      _connectionStatus = 'Belum diperiksa';
      _statusColor = Colors.grey;
    });
    _showSnackBar('Pengaturan direset ke default', Colors.blue);
  }

  @override
  void dispose() {
    _serverUrlController.dispose();
    _portController.dispose();
    super.dispose();
  }

  @override
```

```
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.grey[100],
      appBar: AppBar(
        backgroundColor: const Color(0xFF00BFA5),
        elevation: 0,
        leading: IconButton(
          icon: const Icon(Icons.arrow_back, color: Colors.white),
          onPressed: () => Navigator.pop(context),
        ),
        title: const Text(
          'Pengaturan Server',
          style: TextStyle(color: Colors.white, fontWeight:
FontWeight.bold),
        ),
        actions: [
          IconButton(
            icon: const Icon(Icons.refresh, color: Colors.white),
            onPressed: _resetToDefault,
            tooltip: 'Reset ke Default',
          ),
        ],
      ),
      body: _isLoading
          ? const Center(child: CircularProgressIndicator())
          : SingleChildScrollView(
              padding: const EdgeInsets.all(16),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  // Info Card
                  Container(
                    padding: const EdgeInsets.all(16),
                    decoration: BoxDecoration(
                      color: const Color(0xFFE3F2FD),
                      borderRadius: BorderRadius.circular(12),
                      border: Border.all(color:
Colors.blue.shade200),
                    ),
                    child: Row(
                      children: [
                        Icon(Icons.info_outline, color:
Colors.blue[700]),
                        const SizedBox(width: 12),
                        const Expanded(
```

```
                          child: Text(
                            'Konfigurasikan URL server untuk deteksi
penyakit udang',
                            style: TextStyle(fontSize: 12, height:
1.4),
                          ),
                        ),
                      ],
                    ),
                  ),
                  const SizedBox(height: 24),

                  // Server Configuration Card
                  Container(
                    padding: const EdgeInsets.all(20),
                    decoration: BoxDecoration(
                      color: Colors.white,
                      borderRadius: BorderRadius.circular(16),
                      boxShadow: [
                        BoxShadow(
                          color: Colors.black.withAlpha(13),
                          blurRadius: 10,
                          offset: const Offset(0, 2),
                        ),
                      ],
                    ),
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        Row(
                          children: [
                            Container(
                              padding: const EdgeInsets.all(8),
                              decoration: BoxDecoration(
                                color: const
Color(0xFF00BFA5).withAlpha(25),
                                borderRadius:
BorderRadius.circular(8),
                              ),
                              child: const Icon(
                                Icons.dns,
                                color: Color(0xFF00BFA5),
                                size: 20,
                              ),
                            ),
```

```
                            const SizedBox(width: 12),
                            const Text(
                              'Konfigurasi Server',
                              style: TextStyle(
                                fontSize: 16,
                                fontWeight: FontWeight.bold,
                              ),
                            ),
                          ],
                        ),
                        const SizedBox(height: 20),

                        // Server URL Input
                        const Text(
                          'URL Server',
                          style: TextStyle(
                            fontSize: 14,
                            fontWeight: FontWeight.w600,
                          ),
                        ),
                        const SizedBox(height: 8),
                        TextField(
                          controller: _serverUrlController,
                          decoration: InputDecoration(
                            hintText: 'http://192.168.1.100',
                            hintStyle: TextStyle(color:
Colors.black.withAlpha(80)),
                            prefixIcon: const Icon(Icons.link,
color: Color(0xFF00BFA5)),
                            filled: true,
                            fillColor: Colors.grey[50],
                            border: OutlineInputBorder(
                              borderRadius:
BorderRadius.circular(12),
                              borderSide: BorderSide(color:
Colors.grey.shade300),
                            ),
                            enabledBorder: OutlineInputBorder(
                              borderRadius:
BorderRadius.circular(12),
                              borderSide: BorderSide(color:
Colors.grey.shade300),
                            ),
                            focusedBorder: OutlineInputBorder(
```

```
                              borderRadius:
BorderRadius.circular(12),
                              borderSide: const BorderSide(color:
Color(0xFF00BFA5), width: 2),
                            ),
                          ),
                        ),
                        const SizedBox(height: 16),

                        // Port Input
                        const Text(
                          'Port',
                          style: TextStyle(
                            fontSize: 14,
                            fontWeight: FontWeight.w600,
                          ),
                        ),
                        const SizedBox(height: 8),
                        TextField(
                          controller: _portController,
                          keyboardType: TextInputType.number,
                          decoration: InputDecoration(
                            hintText: '5000',
                            hintStyle: TextStyle(color:
Colors.black.withAlpha(80)),
                            prefixIcon: const
Icon(Icons.settings_ethernet, color: Color(0xFF00BFA5)),
                            filled: true,
                            fillColor: Colors.grey[50],
                            border: OutlineInputBorder(
                              borderRadius:
BorderRadius.circular(12),
                              borderSide: BorderSide(color:
Colors.grey.shade300),
                            ),
                            enabledBorder: OutlineInputBorder(
                              borderRadius:
BorderRadius.circular(12),
                              borderSide: BorderSide(color:
Colors.grey.shade300),
                            ),
                            focusedBorder: OutlineInputBorder(
                              borderRadius:
BorderRadius.circular(12),
```

```
                              borderSide: const BorderSide(color:
Color(0xFF00BFA5), width: 2),
                            ),
                          ),
                        ),
                        const SizedBox(height: 20),

                        // Full URL Preview
                        Container(
                          padding: const EdgeInsets.all(12),
                          decoration: BoxDecoration(
                            color: Colors.grey[100],
                            borderRadius: BorderRadius.circular(8),
                            border: Border.all(color:
Colors.grey.shade300),
                          ),
                          child: Row(
                            children: [
                              Icon(Icons.check_circle, size: 16,
color: Colors.grey[600]),
                              const SizedBox(width: 8),
                              Expanded(
                                child: Text(
                                  'Full URL: ${getFullServerUrl()}',
                                  style: TextStyle(
                                    fontSize: 12,
                                    color: Colors.grey[700],
                                    fontFamily: 'monospace',
                                  ),
                                ),
                              ),
                            ],
                          ),
                        ),
                      ],
                    ),
                  ),
                  const SizedBox(height: 16),

                  // Connection Status Card
                  Container(
                    padding: const EdgeInsets.all(20),
                    decoration: BoxDecoration(
                      color: Colors.white,
                      borderRadius: BorderRadius.circular(16),
```

```
                      boxShadow: [
                        BoxShadow(
                          color: Colors.black.withAlpha(13),
                          blurRadius: 10,
                          offset: const Offset(0, 2),
                        ),
                      ],
                    ),
                    child: Column(
                      crossAxisAlignment: CrossAxisAlignment.start,
                      children: [
                        Row(
                          children: [
                            Container(
                              padding: const EdgeInsets.all(8),
                              decoration: BoxDecoration(
                                color: _statusColor.withAlpha(25),
                                borderRadius:
BorderRadius.circular(8),
                              ),
                              child: Icon(
                                Icons.wifi_tethering,
                                color: _statusColor,
                                size: 20,
                              ),
                            ),
                            const SizedBox(width: 12),
                            const Text(
                              'Status Koneksi',
                              style: TextStyle(
                                fontSize: 16,
                                fontWeight: FontWeight.bold,
                              ),
                            ),
                          ],
                        ),
                        const SizedBox(height: 16),
                        Row(
                          mainAxisAlignment:
MainAxisAlignment.spaceBetween,
                          children: [
                            Text(
                              _connectionStatus,
                              style: TextStyle(
                                fontSize: 14,
```

```
                              color: _statusColor,
                              fontWeight: FontWeight.w600,
                            ),
                          ),
                          Container(
                            width: 12,
                            height: 12,
                            decoration: BoxDecoration(
                              color: _statusColor,
                              shape: BoxShape.circle,
                            ),
                          ),
                        ],
                      ),
                      const SizedBox(height: 16),
                      SizedBox(
                        width: double.infinity,
                        child: OutlinedButton.icon(
                          onPressed: _testConnection,
                          icon: const Icon(Icons.wifi_find),
                          label: const Text('Test Koneksi'),
                          style: OutlinedButton.styleFrom(
                            foregroundColor: const
Color(0xFF00BFA5),
                            side: const BorderSide(color:
Color(0xFF00BFA5)),
                            padding: const
EdgeInsets.symmetric(vertical: 12),
                            shape: RoundedRectangleBorder(
                              borderRadius:
BorderRadius.circular(12),
                            ),
                          ),
                        ),
                      ),
                    ],
                  ),
                ),
                const SizedBox(height: 24),

                // Save Button
                SizedBox(
                  width: double.infinity,
                  child: ElevatedButton(
                    onPressed: _isSaving ? null : _saveSettings,
```

```
                      style: ElevatedButton.styleFrom(
                        backgroundColor: const Color(0xFF00BFA5),
                        padding: const
EdgeInsets.symmetric(vertical: 16),
                        shape: RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(12),
                        ),
                        disabledBackgroundColor: Colors.grey,
                      ),
                      child: _isSaving
                          ? const SizedBox(
                              height: 20,
                              width: 20,
                              child: CircularProgressIndicator(
                                color: Colors.white,
                                strokeWidth: 2,
                              ),
                            )
                          : const Row(
                              mainAxisAlignment:
MainAxisAlignment.center,
                              children: [
                                Icon(Icons.save, color:
Colors.white),
                                SizedBox(width: 8),
                                Text(
                                  'Simpan Pengaturan',
                                  style: TextStyle(
                                    fontSize: 16,
                                    fontWeight: FontWeight.bold,
                                    color: Colors.white,
                                  ),
                                ),
                              ],
                            ),
                    ),
                  ),
                ],
              ),
            ),
    );
  }
}
```

Lampiran 8 Program Maindata Aplikasi S-dides

```
import 'package:flutter/material.dart';
import 'screens/monitor_screen.dart';
import 'screens/penyakit_screen.dart';
import 'screens/riwayat_screen.dart';
import 'widgets/app_header.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'S-DIDES Monitor',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        colorScheme: ColorScheme.fromSeed(seedColor: const
Color(0xFF00BFA5)),
        useMaterial3: true,
      ),
      home: const MainScreen(),
    );
  }
}

class MainScreen extends StatefulWidget {
  const MainScreen({super.key});

  @override
  State<MainScreen> createState() => _MainScreenState();
}

class _MainScreenState extends State<MainScreen> {
  int _selectedIndex = 0;

  final List<Widget> _screens = [
    const MonitorScreen(),
    const PenyakitScreen(),
    const RiwayatScreen(),
  ];
```

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.grey[100],
    body: Column(
      children: [
        const AppHeader(),
        // Tab Navigation
        Container(
          margin: const EdgeInsets.all(16),
          padding: const EdgeInsets.all(4),
          decoration: BoxDecoration(
            color: Colors.white,
            borderRadius: BorderRadius.circular(12),
            boxShadow: [
              BoxShadow(
                color: Colors.black.withOpacity(0.05),
                blurRadius: 10,
                offset: const Offset(0, 2),
              ),
            ],
          ),
          child: Row(
            children: [
              Expanded(
                child: _buildTabButton(
                  index: 0,
                  icon: Icons.monitor,
                  label: 'Monitor',
                ),
              ),
              Expanded(
                child: _buildTabButton(
                  index: 1,
                  icon: Icons.warning_amber_rounded,
                  label: 'Penyakit',
                ),
              ),
              Expanded(
                child: _buildTabButton(
                  index: 2,
                  icon: Icons.schedule,
                  label: 'Riwayat',
                ),
```

```
                ),
              ],
            ),
          ),
          Expanded(
            child: _screens[_selectedIndex],
          ),
        ],
      ),
    );
  }

  Widget _buildTabButton({
    required int index,
    required IconData icon,
    required String label,
  }) {
    final isSelected = _selectedIndex == index;

    List<Color> gradientColors;
    switch (index) {
      case 0:
        gradientColors = [const Color(0xFF00BFA5), const
Color(0xFF00897B)];
        break;
      case 1:
        gradientColors = [const Color(0xFFEF5350), const
Color(0xFFE53935)];
        break;
      case 2:
        gradientColors = [const Color(0xFF2196F3), const
Color(0xFF1976D2)];
        break;
      default:
        gradientColors = [const Color(0xFF00BFA5), const
Color(0xFF00897B)];
    }

    return GestureDetector(
      onTap: () {
        setState(() {
          _selectedIndex = index;
        });
      },
      child: Container(
```

```
        padding: const EdgeInsets.symmetric(vertical: 12),
        decoration: BoxDecoration(
          gradient: isSelected
              ? LinearGradient(
                  colors: gradientColors,
                  begin: Alignment.bottomLeft,
                  end: Alignment.topRight,
                )
              : null,
          color: isSelected ? null : Colors.transparent,
          borderRadius: BorderRadius.circular(8),
        ),
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Icon(
              icon,
              color: isSelected ? Colors.white : Colors.grey[600],
              size: 18,
            ),
            const SizedBox(width: 6),
            Text(
              label,
              style: TextStyle(
                color: isSelected ? Colors.white : Colors.grey[600],
                fontWeight: isSelected ? FontWeight.bold :
FontWeight.normal,
                fontSize: 14,
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Lampiran 9 Program Traing data Model Yolov11n

```
!pip install ultralytics roboflow -q

# 2. Import libraries
from ultralytics import YOLO
from roboflow import Roboflow
import torch
```

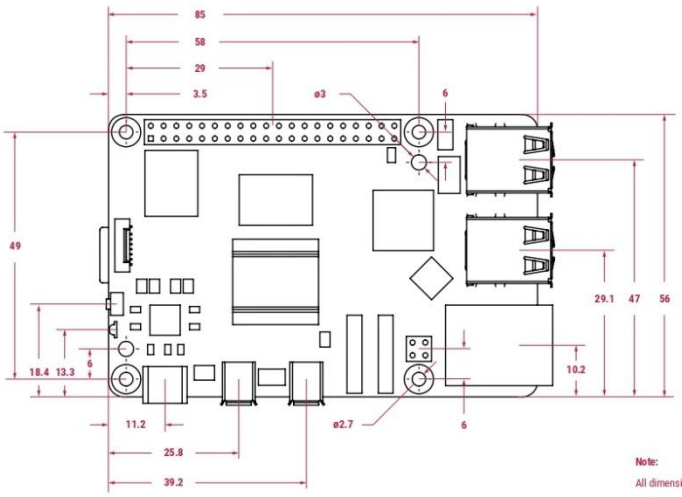
```
import os
from pathlib import Path
print(f"PyTorch version: {torch.__version__}")
print(f"CUDA available: {torch.cuda.is_available()}")
if torch.cuda.is_available():
    print(f"GPU: {torch.cuda.get_device_name(0)}")
    print(f"GPU Memory:
{torch.cuda.get_device_properties(0).total_memory / 1024**3:.2f}
GB")

print("\n📥 Downloading dataset from Roboflow...")
rf = Roboflow(api_key="ShMdmvpZMZ7N50ToLx34")
project = rf.workspace("aaa-fmxpv").project("s-dides-2-1-46jzf")
version = project.version(10)
dataset = version.download("yolov11")

# Get dataset path
dataset_path = dataset.location
data_yaml = f"{dataset_path}/data.yaml"
print(f"☑ Dataset downloaded to: {dataset_path}")’
print("\n🚀 Loading YOLOv11n model...")
model = YOLO('yolo11n.pt')  # Load pretrained YOLOv11n

# 6. Training dengan konfigurasi optimal
print("\n🎯 Starting training with optimized parameters...")
results = model.train(
    data=data_yaml,

    # Model & Training Configuration
    epochs=100,              # Jumlah epoch (sesuaikan dengan
dataset Anda)
    imgsz=640,              # Image size (640 adalah standar
optimal)
    batch=16,               # Batch size (sesuaikan dengan VRAM GPU)

    # Optimizer Settings
    optimizer='AdamW',      # AdamW optimizer untuk hasil lebih baik
    lr0=0.001,              # Initial learning rate
    lrf=0.01,               # Final learning rate (lr0 * lrf)
    momentum=0.937,         # Momentum
    weight_decay=0.0005,    # Weight decay untuk regularisasi

    # Augmentation (penting untuk hasil bagus!)
    hsv_h=0.015,           # HSV-Hue augmentation
    hsv_s=0.7,             # HSV-Saturation augmentation
```

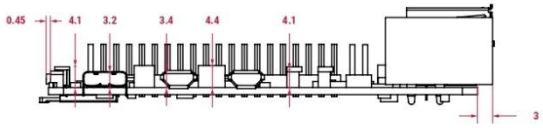
```
    hsv_v=0.4,              # HSV-Value augmentation
    degrees=0.0,            # Rotation augmentation
    translate=0.1,          # Translation augmentation
    scale=0.5,              # Scale augmentation
    shear=0.0,              # Shear augmentation
    perspective=0.0,        # Perspective augmentation
    flipud=0.0,             # Flip up-down probability
    fliplr=0.5,             # Flip left-right probability
    mosaic=1.0,             # Mosaic augmentation probability
    mixup=0.0,              # Mixup augmentation probability
    copy_paste=0.0,         # Copy-paste augmentation

    # Training Settings
    patience=50,            # Early stopping patience (epochs)
    save=True,              # Save checkpoints
    save_period=10,         # Save checkpoint every N epochs
    cache=True,             # Cache images untuk training lebih cepat
    device=0,               # GPU device (0 untuk GPU pertama, 'cpu'
untuk CPU)
    workers=8,              # Number of worker threads

    # Validation
    val=True,               # Validate during training

    # Output
    project='yolo_training',    # Project name
    name='yolov11n_run',        # Run name
    exist_ok=True,              # Overwrite existing project
    pretrained=True,            # Use pretrained weights
    verbose=True,               # Verbose output

    # Performance optimization
    amp=True,               # Automatic Mixed Precision training
(lebih cepat)
    fraction=1.0,           # Dataset fraction to use
    profile=False,          # Profile ONNX model

    # Advanced
    cos_lr=True,            # Cosine learning rate scheduler
    close_mosaic=10,        # Disable mosaic N epochs before end
    resume=False,           # Resume training dari last checkpoint
    overlap_mask=True,      # Overlap mask untuk segmentation (jika
ada)
    mask_ratio=4,           # Mask downsample ratio
    dropout=0.0,            # Dropout regularization
```

```
    # Label settings
    label_smoothing=0.0,   # Label smoothing epsilon
    nbs=64,                # Nominal batch size
    single_cls=False,      # Train as single-class dataset
)

print("\n📊 Training completed!")
print(f"Results saved to: {results.save_dir}")

# 8. Validate model
print("\n☑️ Validating model on test set...")
metrics = model.val()

# Print metrics
print("\n📈 Validation Metrics:")
print(f"mAP50: {metrics.box.map50:.4f}")
print(f"mAP50-95: {metrics.box.map:.4f}")
print(f"Precision: {metrics.box.mp:.4f}")
print(f"Recall: {metrics.box.mr:.4f}")

# 9. Export model (opsional)
print("\n💾 Exporting model...")
# Uncomment format yang Anda butuhkan:
model.export(format='onnx')      # ONNX format
```

Lampiran 10 Datasheet Raspberry Pi 5

Physical specification





Note:

All dimensions in mm

All dimensions are approximate and for reference purposes only. The dimensions shown should not be used for producing production data

The dimensions are subject to part and manufacturing tolerances

Not all of the board components are shown. Please reference a physical board for representation of componentry

Dimensions may be subject to change

WARNINGS

- This product should be operated in a well ventilated environment, and if used inside a case, the case should not be covered.
- While in use, this product should be firmly secured or should be placed on a stable, flat, non-conductive surface, and should not be contacted by conductive items.
- The connection of incompatible devices to Raspberry Pi 5 may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met.

SAFETY INSTRUCTIONS

To avoid malfunction or damage to this product, please observe the following:

- Do not expose to water or moisture, or place on a conductive surface while in operation.
- Do not expose to heat from any source; Raspberry Pi 5 is designed for reliable operation at normal ambient temperatures.
- Store in a cool, dry location.
- Take care while handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- While it is powered, avoid handling the printed circuit board, or handle it only by the edges, to minimise the risk of electrostatic discharge damage.

Specification

Processor:	Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU, with Cryptographic Extension, 512KB per-core L2 caches, and a 2MB shared L3 cache
Features:	• VideoCore VII GPU, supporting OpenGL ES 3.1, Vulkan 1.2 • Dual 4Kp60 HDMI® display output with HDR support • 4Kp60 HEVC decoder • LPDDR4X-4267 SDRAM (options for 1GB, 2GB, 4GB, 8GB and 16GB) • Dual-band 802.11ac Wi-Fi® • Bluetooth 5.0 / Bluetooth Low Energy (BLE) • microSD card slot, with support for high-speed SDR104 mode • 2 × USB 3.0 ports, supporting simultaneous 5Gbps operation • 2 × USB 2.0 ports • Gigabit Ethernet, with PoE+ support (requires separate PoE+ HAT) • 2 × 4-lane MIPI camera/display transceivers • PCIe 2.0 x1 interface for fast peripherals (requires separate M.2 HAT or other adapter) • 5V/5A DC power via USB-C, with Power Delivery support • Raspberry Pi standard 40-pin header • Real-time clock (RTC), powered from external battery • Power button
MTBF[1] Ground Benign:	93 800 hours
Operating temperature:	0°C to 70°C
Production lifetime:	Raspberry Pi 5 will remain in production until at least January 2036
Compliance:	For a full list of local and regional product approvals, please visit pip.raspberrypi.com
List price:	1GB **$45** 2GB **$65** 4GB **$110** 8GB **$175** 16GB **$305**

[1] Mean Time Between Failure

Overview

Welcome to the latest generation of Raspberry Pi: the everything computer.

Featuring a 64-bit quad-core Arm Cortex-A76 processor running at 2.4GHz, Raspberry Pi 5 delivers a 2–3× increase in CPU performance relative to Raspberry Pi 4. Alongside a substantial uplift in graphics performance from an 800MHz VideoCore VII GPU; dual 4Kp60 display output over HDMI; and state-of-the-art camera support from a rearchitected Raspberry Pi Image Signal Processor, it provides a smooth desktop experience for consumers, and opens the door to new applications for industrial customers.

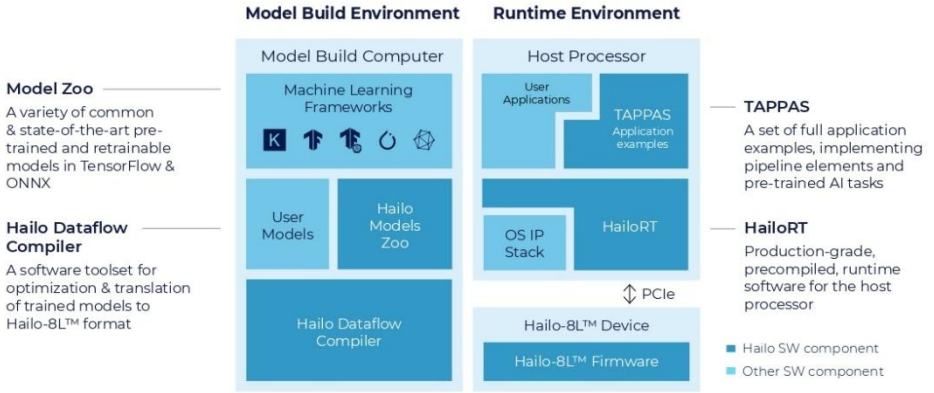
For the first time, this is a full-size Raspberry Pi computer using silicon built in-house at Raspberry Pi. The RP1 "southbridge" provides the bulk of the I/O capabilities for Raspberry Pi 5, and delivers a step change in peripheral performance and functionality. Aggregate USB bandwidth is more than doubled, yielding faster transfer speeds to external UAS drives and other high-speed peripherals; the dedicated two-lane 1Gbps MIPI camera and display interfaces present on earlier models have been replaced by a pair of four-lane 1.5Gbps MIPI transceivers, tripling total bandwidth, and supporting any combination of up to two cameras or displays; peak SD card performance is doubled, through support for the SDR104 high-speed mode; and for the first time the platform exposes a single-lane PCI Express 2.0 interface, providing support for high-bandwidth peripherals.

Lampiran 11 Datasheet Hailo 8l

Hailo-8L™ Entry-Level M.2 AI Acceleration Modules

Comprehensive Software Suite

AI software suite which seamlessly integrates with existing deep learning development frameworks to allow smooth and easy integration in existing development ecosystems. The Hailo AI software suite is future proof, supporting all Hailo-8™ product lines, and includes:



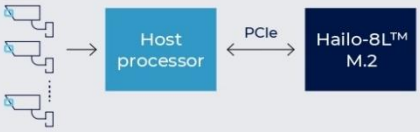
Hailo Ecosystem

Hailo collaborates with partners globally to offer products for edge solutions — from small and fanless devices, to ruggedized industrial appliances, high capacity video analytics platforms, embedded computing platforms for automotive & robotics and more.

To select your Hailo-based AI platform click here

System Usage

M.2 module connected to the various types of host processors via PCIe.



Industries

Industrial Automation

Security

Retail

Click here for more information & product inquiry

HAILO

HAILO

Hailo-8L™ Entry-Level M.2 AI Acceleration Modules

Delivering data center class performance to edge devices

Key Features & Benefits

Powered by 13 Tera-Operations Per Second (TOPS) Hailo-8L™ AI processor

Enabling real-time, low latency, and high-efficiency AI inferencing on edge devices

Fast time to market using a standard M.2 form factor module, with key B+M & key A+E

Best-in-class power efficiency

High cost-efficiency (TOPS/$) compared with existing solutions

Supporting extended temperature range of -40°C to 85°C

Robust software suite supports state-of-the-art deep learning models & applications out-of-the-box

Scalable:

- Enabling simultaneous processing of multi-streams & multi-models
- Future-ready — software compatibility when migrating to Hailo-8™ for more powerful AI

Technical Specifications

Form factor options
M.2 key B+M, key A+E

Dimensions
- Key B+M: 22×42 mm with breakable extensions to 22×60 mm & 22×80 mm
- Key A+E: 22×30 mm

Interface
PCIe Gen-3.0, 2-lanes

Supported host architectures
X86 or ARM based

Supported OS
Linux, Windows

Supported AI frameworks
TensorFlow, TensorFlow Lite, Keras, PyTorch & ONNX

Hailo-8L™ M.2 AI Acceleration Modules

Compatible with M.2 form factor, the Hailo-8L™ based modules can be plugged into an existing edge device with an M.2 socket to execute deep neural network inferencing in real-time utilizing low power for a broad range of market segments.

M.2 Key B+M
(2242/2260/2280)

M.2 Key A+E
(2230)

Featuring the Hailo-8L™ AI Processor

The Hailo-8L™ AI processor, delivering up to 13 tera-operations per second (TOPS), significantly outperforms all other edge AI processors. Its area and power efficiency are far superior to other leading solutions by an order of magnitude.

Hailo-8L™ unique, powerful and scalable structure-driven dataflow architecture takes advantage of the core properties of neural networks. It enables edge devices to run deep learning applications at full scale more efficiently, effectively, and substantially than traditional solutions, while significantly lowering costs.

www.hailo.ai

Lampiran 12 Datasheet HQ Camera

WARNINGS

- This product should be operated in a well ventilated environment, and if used inside a case, the case should not be covered.
- Whilst in use, this product should be firmly secured or should be placed on a stable, flat, non-conductive surface, and should not be contacted by conductive items.
- The connection of incompatible devices to the Raspberry Pi High Quality Camera may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met.

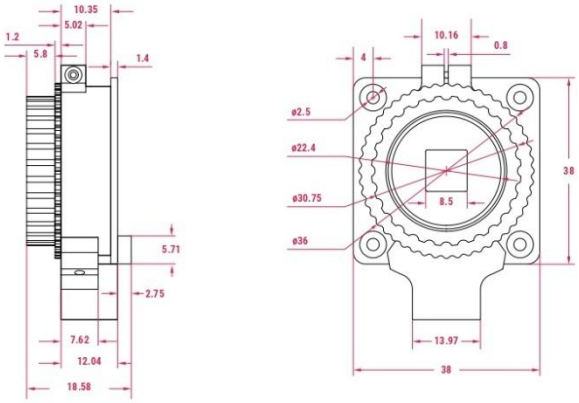
SAFETY INSTRUCTIONS

To avoid malfunction or damage to this product, please observe the following:

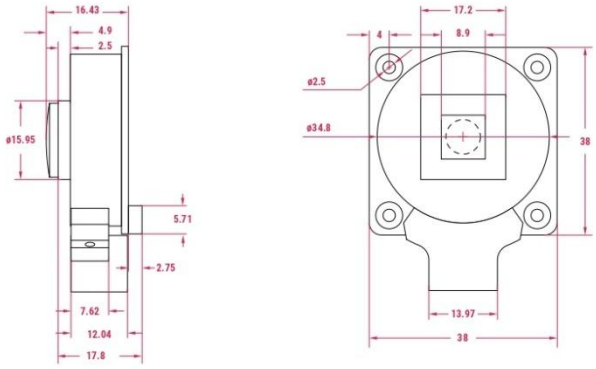
- Important: Before connecting this device, shut down your Raspberry Pi computer and disconnect it from external power.
- If the cable becomes detached, first pull forward the locking mechanism on the connector, then insert the ribbon cable ensuring that the metal contacts face towards the circuit board, and finally push the locking mechanism back into place.
- This device should be operated in a dry environment at 0–50°C.
- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; the Raspberry Pi High Quality Camera is designed for reliable operation at normal ambient temperatures.
- Store in a cool, dry location.
- Avoid rapid changes of temperature, which can cause moisture to build up in the device, affecting image quality.
- Take care not to fold or strain the ribbon cable.
- Take care when screwing in parts or fitting a tripod. A cross-thread can cause irreparable damage and void the warranty.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or handle it only by the edges, to minimise the risk of electrostatic discharge damage.

Physical specification

CS Mount



M12 Mount



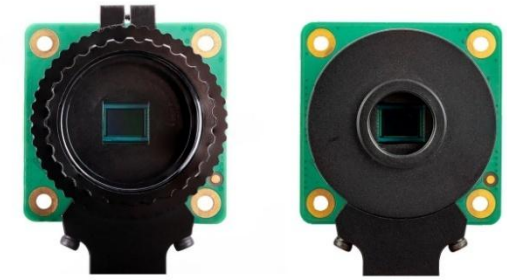
Note: all dimensions in mm

Specification

Sensor:	Sony IMX477R stacked, back-illuminated sensor
Resolution:	12.3 megapixels
Sensor size:	7.9mm sensor diagonal
Pixel size:	1.55µm × 1.55µm
Output:	RAW12/10/8, COMP8
Back focus length of lens:	2.6mm–11.8mm (M12 Mount variant) 12.5mm–22.4mm (CS Mount variant)
Lens sensor format:	1/2.3" (7.9mm) or larger
IR cut filter:	Integrated [2]
Ribbon cable length:	200mm
Tripod mount:	1/4"-20
Compliance:	FCC 47 CFR Part 15, Subpart B, Class B Digital Device Electromagnetic Compatibility Directive (EMC) 2014/30/EU Restriction of Hazardous Substances (RoHS) Directive 2011/65/EU
Production lifetime:	The Raspberry Pi High Quality Camera will remain in production until at least January 2030

[2] Can be removed to enable IR sensitivity. Modification is irreversible.

Overview



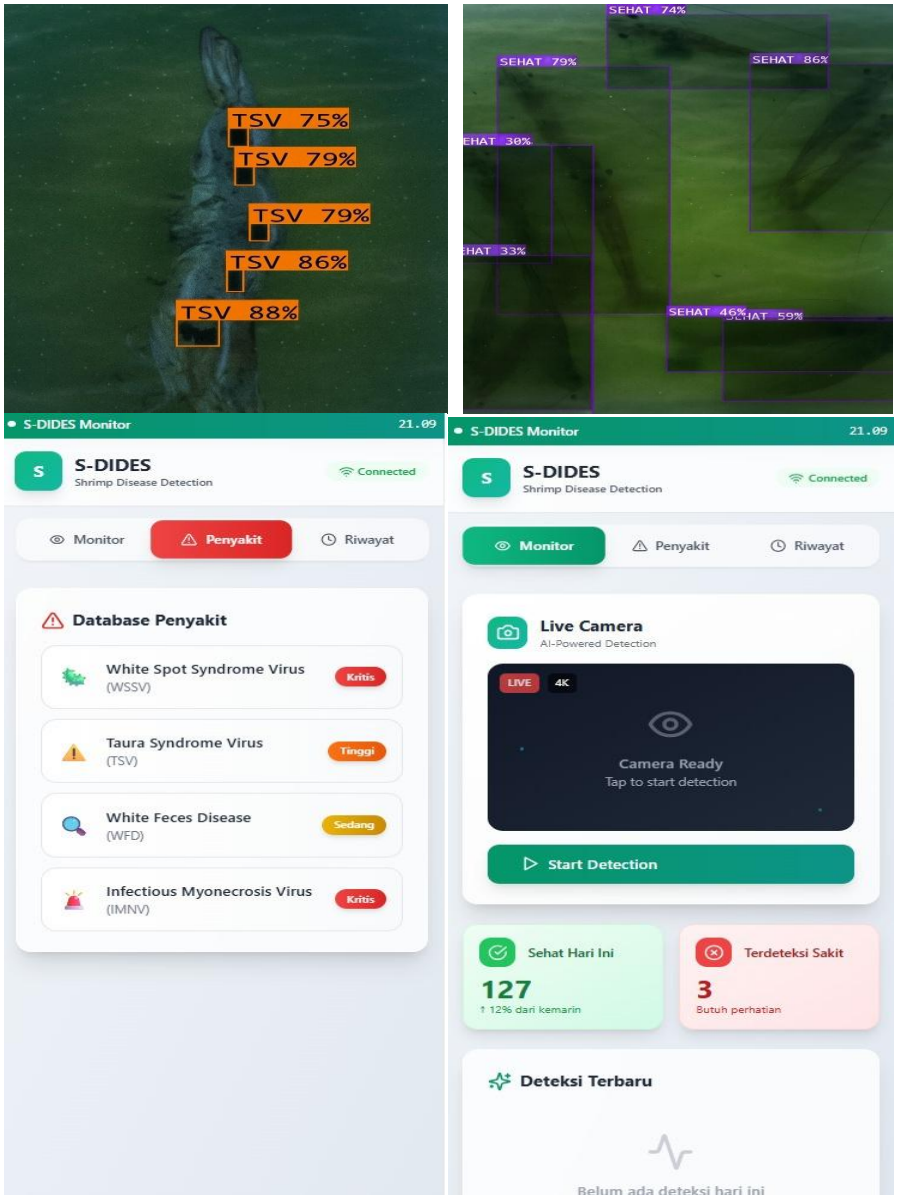
The Raspberry Pi High Quality Camera is an affordable high-quality camera from Raspberry Pi. It offers 12-megapixel resolution and a 7.9mm-diagonal sensor for impressive low-light performance. The M12 Mount variant is designed to work with most interchangeable M12 lenses, and the CS Mount variant is designed to work with interchangeable lenses in both CS- and C-mount form factors (C-mount lenses require the use of the C-CS adapter included with this variant). Other lens form factors can be accommodated using third-party lens adapters.

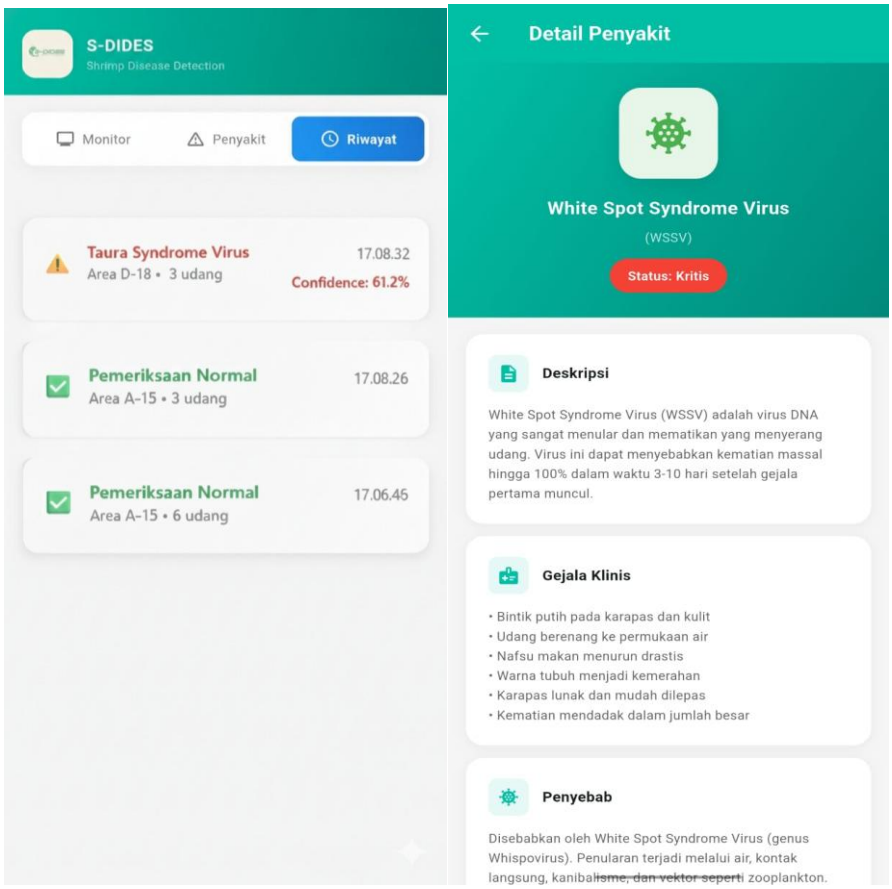
The High Quality Camera is well suited to industrial and consumer applications, including security cameras, which require the highest levels of visual fidelity and/or integration with specialist optics. It is compatible with all models of Raspberry Pi computer from Raspberry Pi 1 Model B onwards, using the latest software release from raspberrypi.com.[1]

The package comprises a circuit board carrying a Sony IMX477 sensor, an FPC cable for connection to a Raspberry Pi computer, and a milled aluminium lens mount with integrated tripod mount. The CS Mount variant lens mount features a focus adjustment ring, and this variant ships with a C- to CS-mount adapter; the M12 Mount variant ships with three lens locking rings (one required plus two spare).

[1] Excluding early Raspberry Pi Zero models, which lack the necessary FPC connector. Later Raspberry Pi Zero models require an adapter FPC, sold separately.

Lampiran 13 Prototipe yang Diciptakan




S-DIDES
Shrimp Disease Detection
Monitor
Penyakit
Riwayat
Taura Syndrome Virus
Area D-18 • 3 udang
17.08.32
Confidence: 61.2%
Pemeriksaan Normal
Area A-15 • 3 udang
17.08.26
Pemeriksaan Normal
Area A-15 • 6 udang
17.06.45
Detail Penyakit
White Spot Syndrome Virus
(WSSV)
Status: Kritis
Deskripsi
White Spot Syndrome Virus (WSSV) adalah virus DNA yang sangat menular dan mematikan yang menyerang udang. Virus ini dapat menyebabkan kematian massal hingga 100% dalam waktu 3-10 hari setelah gejala pertama muncul.
Gejala Klinis
• Bintik putih pada karapas dan kulit
• Udang berenang ke permukaan air
• Nafsu makan menurun drastis
• Warna tubuh menjadi kemerahan
• Karapas lunak dan mudah dilepas
• Kematian mendadak dalam jumlah besar
Penyebab
Disebabkan oleh White Spot Syndrome Virus (genus Whispovirus). Penularan terjadi melalui air, kontak langsung, kanibalisme, dan vektor seperti zooplankton.