BAB II
LANDASAN TEORI

Bab ini menjadi landasan konseptual yang penting untuk mendukung analisis dan pembahasan dalam penelitian. Teori-teori yang dibahas di sini sangat relevan dengan masalah yang diteliti dan telah dibuktikan secara nyata. Teori tersebut membantu menjelaskan konsep dan variabel yang terkait dengan pengembangan sistem pengenalan wajah berbasis *face tracking menggunakan framework MediaPipe* dan mikrokontroler *ESP-32*.

2.1 State of the Art

Tabel 2. 1 Perbandingan Penelitian Sebelumnya (State of the Art)

No	Peneliti, (Tahun)	Judul Penelitian	Metode	Objek Penelitian	Perbandingan yang dijadikan alasan tinjauan penelitian
1.	Joe Yuan Mambu, Andria Wahyudi, Zakarias Reinaldo dan Therok Braif (2017) *CogITo Smart Journal*, Vol. 3, No. 2, ISSN. 2541-2221. https://doi.org/10.31154/cogito.v3i2.67.164-172	*Robot Perekam Objek Berbasis Face Tracking*	*Prototype*	Wajah Manusia	Penelitian ini menggunakan metode *Viola-Jones*, dan mikrokontroler *Arduino Uno* untuk mendeteksi wajah manusia.
2	Muhammad Bagasjati Pratiwo, Mutaqin Akbar (2022)	*Penggunaan Augmented Reality untuk Pemilihan Masker Dengan*	*Waterfall*	Wajah yang memakai masker	Penelitian ini mengimplementasikan *face detection* dan *face tracking* untuk mendeteksi wajah yang menggunakan

No	Peneliti, (Tahun)	Judul Penelitian	Metode	Objek Penelitian	Perbandingan yang dijadikan alasan tinjauan penelitian
	Antivirus : Jurnal Ilmiah Teknik Informatika Vol. 16, No. 1, ISSN. 1978 -5232 https://doi.org/10.35457/antivirus.v16i1.1961	*Metode Face Tracking*			masker.
3	Dewangga Mantara Saktia, Wahyu Sudoro Murtia, Ayu Kurniasaria, Jamal Rosida (2022) *Indonesian Journal of Data and Science(IJODAS)*, Vol. 3, No. 1, ISSN. 2715-9930 https://jurnal.yoctobrain.org/index.php/ijodas/article/view/38/37	*Face recognition dengan metode Haar Cascade dan Facenet*	*Haar Cascade*	Wajah Manusia	Penelitian ini menggunakan *Haar Cascade* untuk mendeteksi wajah *dan Facenet* untuk mengenali wajah yang terdeteksi.
4	Malik Abdul Ghani, Andre Rusli, Ni Made Satvika Iswari. (2019)	*Rancang Bangun Aplikasi Face Tracking dan Filter Berdasarkan Raut Wajah*	*Fisher-Yates*	Raut Wajah Manusia	Penelitian ini merancang aplikasi *face tracking* untuk di sistem operasi IOS.

No	Peneliti, (Tahun)	Judul Penelitian	Metode	Objek Penelitian	Perbandingan yang dijadikan alasan tinjauan penelitian
	Ultima Computing: Jurnal Sistem Komputer Vol. 11, No. 1, ISSN. 2355-3286 https://doi.org/10.31937/sk.v11i1.1046	*Menggunakan Algoritma Fisher-Yates Berbasis iOS*			
5	Fitry Yuliani (2022) DSpace: Universitas Islam Indonesia https://dspace.uii.ac.id/handle/123456789/40453	*Analisis Dan Implementasi Object Tracking Pada Kamera Webcam Dengan Image Processing Menggunakan Metode Mean Shift*	*Mean Shift*	Pergerakan Manusia	Penelitian ini merancang sistem pelacakan target objek dengan menggunakan kamera *webcam* yang terhubung dengan *Raspberry Pi.*
6	Dwi Novianti, Donny Muda Priyangan, Pramestiana Jounal of Computer Science and Informatics (JOCSI) Vol. 1, No. 2, ISSN: 3025-1346 https://ojs.edupartner.co.id/index.php/jocsi/article/view/38	*MEMBANGUN SISTEM APLIKASI ABSENSI BERBASIS FACE RECOGNITION MENGGUNAKAN OPENCV DIKAMPUS STMIK KALIREJO LAMPUNG*	*Waterfall*	Wajah Manusia	Penelitian ini membangun sistem absensi yang memanfaatkan *face recognition sebagai mengenalan mahasiswa di STMIK KALIREJO LAMPUNG*

No	Peneliti, (Tahun)	Judul Penelitian	Metode	Objek Penelitian	Perbandingan yang dijadikan alasan tinjauan penelitian
7	Nofal Anam (2022) Perpustakaan Universitas Dinamika http://repository.dinamika.ac.id/id/eprint/6259	*Sistem Deteksi Simbol Pada Sibi (Sistem Isyarat Bahasa Indonesia) Menggunakan MediaPipe Dan Resnet-50*	*Prototype*	Gerakan Badan dan Mimik Muka	Penelitiann ini menggunakan *MediaPipe* untuk mendeteksi gerakan badan dan mimik muka.
8	Achmad Hasyim Nur'azizan, Abdul Riqza Ardiansyah, Rasio Fernandis (2024) *Seminar Nasional Teknologi & Sains* Vol. 3, No. 1, ISSN. 2828–299X https://doi.org/10.29407/rq0kp167	*Implementasi Deteksi Bahasa Isyarat Tangan Menggunakan OpenCV dan MediaPipe*	*R&D (Research and Development)*	Gerakan Tangan	Penelitian ini mengimplementasikan *MediaPipe* untuk deteksi bahasa isyarat tangan.
9	Aldi Ramadhani (2024) Perpustakaan Universitas Dinamika http://repository.dinamika.ac.id/id/eprint/7795	*Sistem Kontrol Robot Mobil Melalui Deteksi Bentuk Gestur Jari Tangan Menggunakan Mediapipe*	*Prototype*	Gestur Jari Tangan	Penelitian ini mendeteksi gestur jari tangan dengan *MediaPipe* untuk mengontrol Robot Mobil.
10	Jiankang Deng, Jia Guo, Jing Yang,	*ArcFace: Additive Angular Margin Loss*		Pengenalan wajah (*face recognitio*	Penelitian ini menggunakan model ArcFace sebagai metode identifikasi

No	Peneliti, (Tahun)	Judul Penelitian	Metode	Objek Penelitian	Perbandingan yang dijadikan alasan tinjauan penelitian
	Niannan Xue, Irene Kotsia, Stefanos Zafeiriou IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. 44, Issue. 10, https://doi.org/10.48550/arXiv.1801.07698	*for Deep Face Recognition*		*n*) menggunakan model deep learning berbasis embedding wajah.	wajah yang juga diterapkan pada penelitian ini melalui *framework* DeepFace.
11	Mochamad Aditya Rizky Fahreza, Abd Rabi Wahyu Dirgantara	Pengembangan Sistem Pintu Portal Miniatur dengan Pengenalan Wajah Menggunakan FaceNet dan MediaPipe	*Prototype*	Pengenalan wajah (*face recognition*) menggunakan model FaceNet	Penelitian ini menggunakan model FaceNet sebagai metode identifikasi wajah
12 2.	Mukhlis Aburizal Khordowi	Rancang Bangun Sistem Face Tracking Menggunakan *Framework* MediaPipe Dan Mikrokontroler ESP-32	*Prototype*	Wajah Manusia	Penelitian ini merancang dan membangun sistem face tracking dan face recognition menggunakan *Framework* MediaPipe dan Mikrokontroler ESP-32 untuk memantau dan mengidentifikasi orang yang memasuki area pengawasan secara otomatis.

Berdasarkan hasil studi terhadap penelitian-penelitian sebelumnya, diketahui bahwa penelitian Sakti et al. (2022) telah berhasil menerapkan *face recognition* menggunakan metode Haar Cascade dan FaceNet dengan tingkat akurasi sebesar

80%. Namun, penelitian tersebut berfokus pada proses pengenalan wajah dan belum mengintegrasikan mekanisme *face tracking* maupun pengendalian kamera secara otomatis. Penelitian Fahreza (2024) juga telah mengembangkan sistem pengenalan wajah menggunakan MediaPipe dan FaceNet pada sistem pintu portal otomatis. Penelitian tersebut berhasil menerapkan pengenalan wajah sebagai mekanisme autentikasi akses, namun belum menerapkan *face tracking* untuk mengikuti pergerakan wajah serta belum mengintegrasikan mekanisme kamera *pan-tilt* sebagai pendukung proses pengenalan wajah. Sementara itu, penelitian Mambu et al. (2017) telah mengembangkan sistem *face tracking* menggunakan metode Viola-Jones dengan bantuan servo sebagai penggerak kamera. Akan tetapi, sistem yang dikembangkan masih terbatas pada pergerakan kamera satu sumbu (*pan*) dan belum dilengkapi dengan kemampuan *face recognition* untuk mengidentifikasi identitas pengguna.

Berdasarkan penelitian-penelitian tersebut, masih terdapat *research gap* dalam pengembangan sistem yang mengintegrasikan *face tracking* dan *face recognition* secara terpadu pada satu sistem. Sebagian penelitian hanya berfokus pada proses pengenalan wajah tanpa mekanisme *face tracking*, sedangkan penelitian lainnya telah menerapkan *face tracking* tetapi belum mampu mengenali identitas pengguna. Selain itu, penerapan *face tracking* menggunakan *Framework* MediaPipe yang dipadukan dengan *face recognition* menggunakan *Framework* DeepFace dengan model ArcFace serta pengendalian kamera *pan-tilt* berbasis mikrokontroler ESP-32 masih terbatas.

Oleh karena itu, penelitian ini mengusulkan rancang bangun sistem pengenalan wajah berbasis face tracking menggunakan *Framework* MediaPipe dan mikrokontroler ESP-32. Sistem yang dikembangkan mampu mendeteksi dan mengikuti pergerakan wajah secara real-time melalui mekanisme kamera pan-tilt, kemudian melakukan proses face recognition menggunakan *Framework* DeepFace dengan model ArcFace untuk mengenali wajah yang telah terdaftar pada data referensi serta membedakannya dengan Orang Tidak Dikenal (OTK). Dengan demikian, penelitian ini memberikan kontribusi berupa integrasi teknologi face

tracking, face recognition, dan pengendalian kamera aktif dalam satu sistem pengenalan wajah yang bekerja secara real-time.

2.2 Pengolahan Citra Digital

Pengolahan citra digital (*digital image processing*) merupakan suatu proses yang bertujuan untuk memanipulasi dan menganalisis citra dengan bantuan komputer. Menurut Dristyan et al. (2026), image processing atau pemrosesan gambar adalah suatu disiplin ilmu yang berfokus pada pemrosesan citra sebagai input dan output dari berbagai proses. Penerapan metode image processing dapat mengubah citra menjadi bentuk yang lebih sederhana untuk beberapa keperluan tertentu atau dalam mengekstrak informasi berharga dari dalam citra. Secara sederhana, image processing merupakan proses memanipulasi atau mengubah citra digital menggunakan teknik matematis dan algoritma komputer (Dristyan et al., 2026).

Secara langsung komputer tidak dapat merepresentasikan citra analog, sehingga citra analog harus dikonversi terlebih dahulu menjadi citra digital. Komputer dapat memproses secara langsung gambar atau citra yang dihasilkan dari peralatan digital seperti kamera DSLR karena pada peralatan digital terdapat sampling dan kuantisasi, sedangkan pada peralatan analog tidak dilengkapi sistem sampling dan kuantisasi(Andono et al., 2018). Sistem sampling adalah sistem yang dapat mengubah citra kontinu menjadi citra digital, di mana citra kontinu tersebut diperoleh dari sistem optik yang menerima sinyal analog. Cara mengubah citra kontinu menjadi citra digital yaitu dengan membagi citra analog tersebut menjadi M baris dan N kolom, semakin besar nilai M dan N maka semakin halus citra digital yang dihasilkan. Pertemuan dari baris dan kolom tersebut disebut dengan piksel. Sedangkan sistem kuantisasi adalah sistem yang mengubah intensitas analog menjadi intensitas diskrit (Andono et al., 2018)

2.3 *Computer Vision*

Computer vision merupakan salah satu cabang ilmu dalam bidang kecerdasan buatan (*Artificial Intelligence*) yang memungkinkan komputer untuk memperoleh, memproses, dan memahami informasi dari citra atau video. Teknologi ini bertujuan

untuk meniru kemampuan penglihatan manusia dalam mengenali dan menginterpretasikan objek, pola, maupun peristiwa yang terdapat dalam suatu gambar atau lingkungan visual (Dristyan et al., 2026).

Dalam implementasinya, *computer vision* banyak digunakan dalam berbagai bidang, seperti sistem keamanan, sistem pengawasan video, pengenalan wajah (*face recognition*), pelacakan objek (*object tracking*), serta interaksi antara manusia dan komputer (Cahyaningtyas et al., 2025). Proses dalam *computer vision* umumnya melibatkan beberapa tahapan, seperti akuisisi citra, pemrosesan citra, ekstraksi fitur, dan interpretasi hasil.

Akuisisi citra merupakan tahap awal dalam sistem *computer vision*, yaitu proses pengambilan citra atau video dari dunia nyata menggunakan perangkat input seperti kamera atau *webcam*. Citra yang diperoleh kemudian diproses melalui tahap pemrosesan citra untuk meningkatkan kualitas dan mempersiapkan citra agar lebih mudah dianalisis. Selanjutnya, tahap ekstraksi fitur dilakukan untuk mengidentifikasi karakteristik penting dari citra seperti tepi, bentuk, warna, dan tekstur objek. Hasil ekstraksi fitur tersebut kemudian diinterpretasikan oleh sistem untuk menghasilkan keputusan atau tindakan tertentu berdasarkan informasi yang diperoleh.

Perkembangan *computer vision* saat ini semakin pesat seiring dengan kemajuan teknologi *deep learning* dan ketersediaan *dataset* yang semakin besar. Berbagai *framework* dan *library* telah dikembangkan untuk mempermudah implementasi *computer vision*, salah satunya adalah *MediaPipe* yang dikembangkan oleh Google. Dalam penelitian ini, *computer vision* diterapkan untuk mendeteksi dan melacak pergerakan wajah secara *real-time*, serta mengidentifikasi identitas orang yang terdeteksi pada area pengawasan ruangan.

2.4 *Deep Learning*

Deep learning merupakan salah satu cabang dari *machine learning* yang menggunakan jaringan saraf tiruan (*artificial neural network*) dengan banyak lapisan tersembunyi (*hidden layer*) untuk mempelajari representasi data secara

hierarkis. Semakin dalam lapisan yang digunakan, semakin abstrak representasi fitur yang dapat dipelajari oleh model (LeCun et al., 2015). *Deep learning* mampu mengekstrak fitur secara otomatis dari data mentah tanpa memerlukan rekayasa fitur secara manual, sehingga sangat efektif untuk menangani data yang kompleks seperti citra, suara, dan teks.

Deep learning mampu mempelajari fitur-fitur penting dari data secara otomatis melalui proses pelatihan (*training*). Selama proses tersebut, model melakukan penyesuaian terhadap bobot jaringan menggunakan algoritma *backpropagation* dan *gradient descent* sehingga mampu mengenali pola yang terdapat pada data. Seiring dengan bertambahnya data pelatihan, model dapat meningkatkan kemampuan dalam menghasilkan prediksi atau klasifikasi yang lebih akurat (Goodfellow et al., 2016).

Dalam bidang *computer vision*, *deep learning* telah memberikan terobosan yang signifikan, khususnya dalam tugas-tugas seperti klasifikasi citra, deteksi objek, dan pengenalan wajah. Dalam penelitian ini, *deep learning* dimanfaatkan melalui penggunaan model *pre-trained* ArcFace yang diakses melalui *framework* DeepFace untuk melakukan *face recognition* tanpa perlu melakukan pelatihan model dari awal.

2.5 *Face Detection*

Face detection memegang peranan penting dalam analisis citra berbasis wajah dan menjadi salah satu permasalahan utama dalam bidang *computer vision*. Performa berbagai aplikasi berbasis wajah, mulai dari pengenalan dan verifikasi wajah konvensional hingga aplikasi modern seperti pengelompokan, penandaan, dan pencarian wajah, sangat bergantung pada akurasi dan efisiensi proses deteksi wajah. Beberapa detektor wajah frontal telah berhasil dikembangkan pada masa awal, salah satu yang paling populer adalah detektor Viola-Jones (V-J) karena keunggulannya dalam hal efisiensi. Meskipun detektor tersebut telah banyak mengalami pengembangan, performanya masih belum memuaskan dalam berbagai kondisi nyata akibat variasi tampilan yang besar, seperti perbedaan sudut pandang,

pencahayaan, *occlusion*, ekspresi wajah, dan kondisi pengambilan gambar (Yan et al., 2014).

Dalam *computer vision*, sistem dirancang untuk mengambil keputusan ketika sensor mendeteksi keberadaan suatu objek fisik nyata, salah satunya adalah wajah, dengan memanfaatkan pengenalan pola wajah yang memiliki bentuk unik pada setiap individu. Identifikasi wajah pada dasarnya merupakan gabungan antara proses deteksi wajah (*face detection*) dan proses pengenalan wajah (*face recognition*). *Face detection* berperan dalam proses *face localization*, yaitu proses untuk menentukan ukuran dan posisi wajah pada suatu citra. Dengan adanya *face detection*, posisi wajah pada citra tidak perlu berada pada lokasi tertentu, sehingga memberikan kemudahan dalam proses pengambilan citra (Subita et al., 2023).

2.6 *Face Tracking*

Face tracking merupakan salah satu teknik dalam bidang *computer vision* yang digunakan untuk mendeteksi sekaligus mengikuti pergerakan wajah manusia dalam suatu citra atau video secara berkelanjutan. Berbeda dengan *face detection* yang hanya berfokus pada pendeteksian wajah pada suatu *frame*, *face tracking* memiliki kemampuan untuk melacak posisi wajah dari waktu ke waktu sehingga objek tetap berada dalam pengamatan sistem (Cahyaningtyas et al., 2025).

Dalam implementasinya, *face tracking* bekerja dengan cara mendeteksi posisi awal wajah, kemudian menentukan perubahan posisi berdasarkan pergerakan objek pada setiap *frame*. Proses ini memungkinkan sistem untuk mengetahui arah pergerakan wajah, baik ke kiri, kanan, atas, maupun bawah (Kusumanto et al., 2013).

2.7 *Face Recognition*

Face Recognition merupakan teknologi yang digunakan untuk mengenali atau mengidentifikasi identitas seseorang berdasarkan karakteristik wajah yang dimilikinya. Teknologi ini merupakan pengembangan dari *Face Detection* yang tidak hanya berfungsi untuk mendeteksi keberadaan wajah pada suatu citra atau video, tetapi juga mampu melakukan proses identifikasi dengan membandingkan wajah yang terdeteksi terhadap data wajah yang telah tersimpan pada basis data.

Pada proses *Face Recognition*, sistem akan menangkap citra wajah menggunakan kamera, kemudian mengekstraksi karakteristik unik dari wajah tersebut untuk dijadikan sebagai representasi identitas (Dwi Novianti et al., 2024). Selanjutnya, karakteristik wajah yang diperoleh akan dibandingkan dengan data wajah yang telah tersimpan dalam basis data untuk menentukan tingkat kemiripan. Apabila ditemukan kecocokan dengan data yang tersimpan, sistem akan menampilkan identitas pengguna yang sesuai. Sebaliknya, apabila tidak ditemukan kecocokan, maka wajah tersebut akan dikategorikan sebagai orang yang tidak dikenal.

Dalam implementasi *face recognition* modern, pendekatan yang umum digunakan adalah memanfaatkan model *pre-trained*, yaitu model *deep learning* yang telah dilatih sebelumnya menggunakan *dataset* wajah berskala besar sehingga dapat langsung digunakan tanpa perlu melakukan proses pelatihan ulang dari awal (Goodfellow et al., 2016). Pendekatan ini sangat efisien karena pengembang tidak memerlukan sumber daya komputasi yang besar maupun *dataset* yang sangat banyak untuk membangun sistem *face recognition* yang akurat.

Salah satu *framework* yang menyediakan akses terhadap berbagai model *pre-trained* untuk *face recognition* adalah *DeepFace*. *DeepFace* merupakan *framework face recognition* berbasis Python yang dikembangkan oleh Sefik Ilkin Serengil, yang menyediakan antarmuka sederhana untuk mengakses berbagai *model face recognition pre-trained* seperti *VGG-Face*, *FaceNet*, *DeepID*, dan *ArcFace* (S. I. Serengil & Ozpinar, 2020). Dalam penelitian ini, *DeepFace* digunakan dengan model *ArcFace* sebagai model *face recognition* utama.

ArcFace merupakan model *face recognition* yang dikembangkan oleh Deng et al. (2019) dan menjadi salah satu model dengan akurasi tertinggi dalam pengenalan wajah. *ArcFace* menggunakan pendekatan *Additive Angular Margin Loss* untuk memaksimalkan jarak antar kelas wajah yang berbeda sekaligus meminimalkan jarak dalam kelas wajah yang sama, sehingga menghasilkan representasi fitur wajah yang lebih diskriminatif dan akurat (Deng et al., 2022).

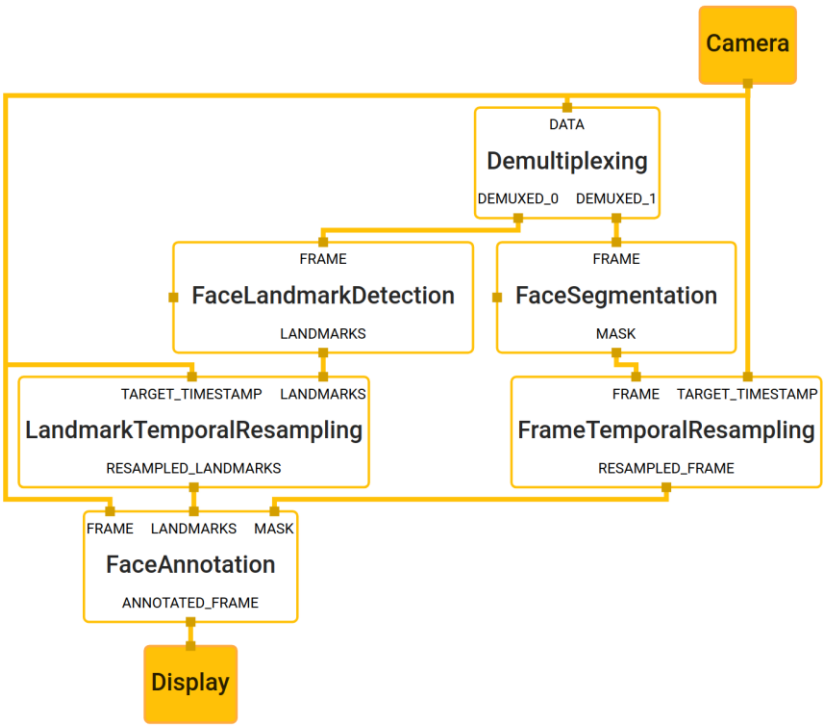
2.8 MediaPipe

MediaPipe merupakan sebuah *framework open-source* yang dikembangkan oleh Google untuk membangun aplikasi berbasis *computer vision* dan *machine learning* secara *real-time*. *MediaPipe* menyediakan berbagai solusi siap pakai (*pre-trained models*) yang dapat digunakan untuk mendeteksi dan melacak berbagai objek, seperti wajah, tangan, pose tubuh, serta *gesture* secara efisien (Zhang et al., 2020).

MediaPipe bekerja dengan memanfaatkan *pipeline* pemrosesan data yang terdiri dari serangkaian komponen yang disebut sebagai "*calculator*", dimana setiap komponen bertugas untuk memproses data secara bertahap mulai dari *input* citra hingga menghasilkan *output* berupa informasi deteksi (Lugaresi et al., 2019). Dengan arsitektur ini, *MediaPipe* mampu melakukan pemrosesan citra secara cepat dan optimal sehingga cocok digunakan untuk aplikasi *real-time* (Ardiansyah et al., 2024).

Salah satu fitur utama MediaPipe yang digunakan dalam penelitian ini adalah *Face Landmark Detection*. MediaPipe mengimplementasikan deteksi *landmark* wajah bersamaan dengan segmentasi potret (*portrait segmentation*) melalui sebuah *graph* pemrosesan yang dirancang untuk meminimalkan beban komputasi. Untuk menjalankan kedua proses tersebut secara bersamaan, MediaPipe menerapkan strategi *demultiplexing*, yaitu memisahkan aliran *frame* masukan menjadi dua subset yang berbeda, dimana setiap subset diproses secara terpisah untuk menjalankan masing-masing tugas secara bergantian (Lugaresi et al., 2019).

Hasil deteksi *landmark* dan segmentasi yang diperoleh dari subset *frame* tersebut kemudian diinterpolasi secara temporal untuk menghasilkan output pada seluruh *frame*. Selanjutnya, anotasi dari kedua proses tersebut digabungkan dan ditampilkan pada *frame* kamera secara tersinkronisasi melalui *annotation node*. Alur pemrosesan *face landmark detection* dan segmentasi pada MediaPipe dapat dilihat pada Gambar 2.1 (Lugaresi et al., 2019).



Gambar 2. 1 Face Landmark Detection and Segmentation using MediaPipe (Sumber: (Lugaresi et al., 2019))

Pipeline ini juga dapat dipercepat menggunakan komputasi berbasis GPU, sehingga seluruh alur data dan proses komputasi dapat berjalan sepenuhnya di GPU tanpa perpindahan data ke CPU yang dapat menghambat performa sistem (Lugaresi et al., 2019).

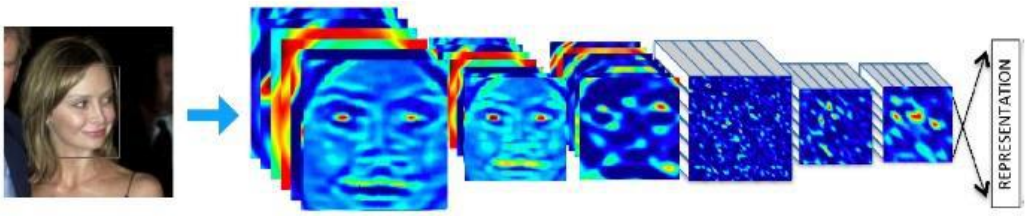
2.9 DeepFace

DeepFace merupakan *framework face recognition* berbasis Python yang dikembangkan oleh Sefik Ilkin Serengil, yang menyediakan antarmuka sederhana untuk mengakses berbagai model *face recognition* berbasis *deep learning* secara langsung. *Framework* ini mendukung berbagai model *pre-trained* seperti VGG-Face, FaceNet, DeepID, dan ArcFace, sehingga memudahkan pengembang dalam mengimplementasikan sistem pengenalan wajah tanpa perlu membangun model dari awal (S. I. Serengil & Ozpinar, 2020).

DeepFace tidak hanya menyediakan kemampuan pengenalan wajah, tetapi juga mencakup berbagai fungsi analisis berbasis wajah seperti verifikasi wajah (*face verification*), pencarian wajah (*face finding*), dan identifikasi wajah (*face*

identification). Selain itu, DeepFace juga mengintegrasikan berbagai model detektor wajah seperti OpenCV, SSD, Dlib, MTCNN, dan RetinaFace yang dapat dipilih sesuai kebutuhan sistem (S. I. Serengil & Ozpinar, 2021).

Seluruh model *face recognition* yang terdapat dalam *framework* DeepFace menggunakan modul representasi berbasis *Convolutional Neural Network* (CNN). Meskipun struktur CNN memiliki banyak *output node*, unit-unit tersebut tidak digunakan untuk tugas klasifikasi secara langsung. Sebaliknya, probabilitas dari setiap *output node* dimanfaatkan untuk menghasilkan *vector embeddings* yang merepresentasikan fitur unik dari setiap wajah, sebagaimana ditunjukkan pada (S. Serengil & Özpınar, 2024).



Gambar 2. 2 Proses Pembentukan Vector Embeddings pada DeepFace (Sumber: (S. Serengil & Özpınar, 2024))

Dalam penelitian ini, DeepFace digunakan sebagai *framework* utama untuk proses pengenalan wajah dengan memanfaatkan model ArcFace sebagai model *face recognition* yang digunakan.

2.10 ArcFace

ArcFace merupakan komponen dari proyek *InsightFace* yang dikembangkan oleh kelompok riset *Deep Inside*. Model ArcFace beserta bobot *pre-trained*-nya dapat diakses secara terbuka untuk berbagai *framework* seperti PyTorch dan MXNet (S. Serengil & Özpınar, 2024).

ArcFace menggunakan pendekatan *Additive Angular Margin Loss* sebagai fungsi kerugian (*loss function*) dalam tugas pengenalan wajah. Berbeda dengan pendekatan *SoftMax* yang umum digunakan, ArcFace mampu menghasilkan fitur yang lebih diskriminatif untuk pengenalan wajah. Pendekatan ini memiliki interpretasi geometris yang jelas karena berkorespondensi langsung dengan jarak

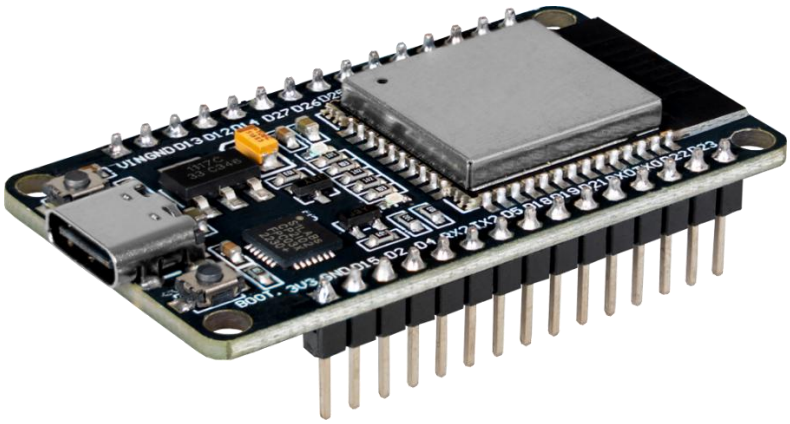
geodesik pada *hypersphere*, sehingga menghasilkan representasi fitur wajah yang lebih akurat dan terpisah antar kelas (Deng et al., 2022).

Dalam penerapannya, model ArcFace terbukti lebih andal dibandingkan model *face recognition* lainnya. Hal ini dibuktikan melalui pengujian pengenalan wajah secara *real-time* yang menunjukkan bahwa model ArcFace menghasilkan akurasi pengenalan yang lebih tinggi dan lebih konsisten, sehingga menjadikannya pilihan utama dalam sistem pengenalan wajah yang membutuhkan tingkat akurasi tinggi (Asmara et al., 2022).

2.11 *Cosine Similarity*

Cosine similarity merupakan salah satu metrik yang digunakan untuk mengukur tingkat kemiripan antara dua buah vektor berdasarkan sudut yang dibentuk oleh kedua vektor tersebut, tanpa memperhitungkan besar (*magnitude*) dari vektor itu sendiri. Semakin kecil sudut yang terbentuk antara dua vektor, maka nilai *cosine similarity* akan semakin mendekati 1, yang menandakan tingkat kemiripan yang tinggi antara kedua vektor tersebut. Sebaliknya, semakin besar sudut yang terbentuk, maka nilai *cosine similarity* akan semakin mendekati 0 atau bahkan negatif, yang menandakan kedua vektor tersebut semakin tidak mirip. Dalam konteks *face recognition*, tugas verifikasi wajah pada dasarnya adalah menghitung jarak (*distance*) antara dua vektor wajah, sehingga metrik jarak yang tepat menjadi hal yang penting bagi akurasi verifikasi wajah (Nguyen & Bai, 2010). Setiap wajah yang telah diekstraksi menjadi *embedding* melalui model *deep learning* seperti ArcFace direpresentasikan sebagai vektor angka berdimensi tinggi, sehingga *cosine similarity* dapat digunakan untuk membandingkan tingkat kemiripan antara *embedding* wajah yang terdeteksi dengan *embedding* wajah yang telah tersimpan pada *database*, guna menentukan apakah kedua wajah tersebut berasal dari individu yang sama.

2.12 Mikrokontroler ESP-32



Gambar 2. 3 Mikrokontroler ESP-32

(Sumber: https://joy-it.net)

ESP-32 merupakan mikrokontroler SoC (System on Chip) yang dikembangkan oleh Espressif Systems yang memiliki kemampuan pemrosesan yang cukup tinggi serta dilengkapi dengan fitur komunikasi nirkabel seperti Wi-Fi dan Bluetooth. ESP-32 hadir sebagai pengembangan dari generasi sebelumnya yaitu ESP-8266, dengan peningkatan signifikan pada kecepatan pemrosesan, jumlah pin I/O, serta fitur komunikasi yang lebih lengkap.

ESP-32 menggunakan prosesor Xtensa LX6 dual-core 32-bit yang mampu beroperasi hingga kecepatan 240 MHz. Selain itu, mikrokontroler ini dilengkapi dengan berbagai antarmuka komunikasi dan periferal, seperti ADC, SPI, I2C, UART, PWM, serta sejumlah pin input/output (I/O) yang dapat digunakan untuk mengendalikan berbagai perangkat eksternal. Ketersediaan fitur tersebut menjadikan ESP-32 fleksibel untuk berbagai kebutuhan pengembangan sistem kendali maupun sistem otomasi (Wagyana, 2019).

Salah satu keunggulan utama ESP-32 adalah tersedianya modul Wi-Fi dan Bluetooth yang telah terintegrasi dalam satu chip. Fitur ini memungkinkan ESP-32 melakukan komunikasi data secara nirkabel dengan perangkat lain maupun jaringan internet tanpa memerlukan modul tambahan. Selain itu, ESP-32 juga didukung oleh berbagai platform pengembangan, seperti Arduino IDE, sehingga memudahkan proses pemrograman dan implementasi sistem.

2.13 Motor Servo

Gambar 2. 4 Motor Servo

(Sumber: https://fit.labs.telkomuniversity.ac.id)

Motor servo merupakan motor yang bekerja dengan sistem closed-loop feedback, dimana informasi posisi motor dikirimkan kembali ke rangkaian kontrol yang terdapat di dalamnya. Berbeda dengan motor DC biasa yang bekerja secara open-loop, motor servo mampu mempertahankan posisi sudut tertentu secara presisi karena adanya mekanisme umpan balik yang terus-menerus memantau dan mengoreksi posisi motor (Hilal & Manan, 2013).

Komponen utama motor servo terdiri dari motor, susunan gear, potensiometer, dan rangkaian kontrol. Motor berfungsi sebagai penggerak utama, susunan gear berfungsi untuk memperbesar torsi yang dihasilkan motor, potensiometer berperan dalam menentukan batas sudut putaran dengan cara mengukur posisi sudut poros motor, sementara rangkaian kontrol bertugas memproses sinyal masukan dan mengatur pergerakan motor sesuai perintah yang diterima.

Besar sudut putaran sumbu motor dikendalikan berdasarkan lebar pulsa yang diterima melalui kabel sinyal, yang dikenal dengan istilah Pulse Width Modulation (PWM). Sebagai contoh, pulsa selebar 1,5 ms akan menempatkan sumbu motor pada posisi tengah (90 derajat). Apabila pulsa OFF semakin lebar, maka sumbu akan bergerak searah jarum jam, dan sebaliknya jika pulsa OFF semakin sempit, sumbu akan bergerak berlawanan arah jarum jam.

Berdasarkan kemampuan pergerakan sudutnya, motor servo dibedakan menjadi dua jenis, yaitu motor servo standar dan motor servo continuous. Motor servo standar hanya mampu bergerak pada rentang sudut tertentu, umumnya antara 0 hingga 180 derajat, sedangkan motor servo continuous mampu berputar secara penuh 360 derajat tanpa batas sudut. Dalam penelitian ini, motor servo yang digunakan adalah motor servo standar yang dikonfigurasi dalam mekanisme pan-tilt untuk menggerakkan kamera secara horizontal (pan) dan vertikal (tilt) mengikuti pergerakan wajah yang terdeteksi.

2.14 Webcam

Gambar 2. 5 Webcam

(Sumber: https://logitech.com/id-id/shop/p/c270-hd-webcam.960-000584)

Webcam (*web camera*) merupakan kamera digital yang terhubung ke komputer dan mampu menangkap gambar atau video secara *real-time*. Kamera digital dan *webcam* merupakan dua perangkat yang paling umum digunakan dalam kegiatan akuisisi citra. Kamera digital biasanya memiliki kualitas sensor dan lensa yang lebih baik serta menyediakan pengaturan manual yang lebih lengkap, seperti pengaturan *aperture*, *shutter speed*, dan ISO. Perangkat ini sering digunakan pada pengambilan data yang membutuhkan kualitas tinggi dan kontrol penuh terhadap kondisi pemotretan (Yuhandri et al., 2022).

Sebaliknya, *webcam* dirancang untuk penggunaan yang lebih sederhana dan praktis. *Webcam* mudah diintegrasikan dengan komputer dan dapat diakses langsung melalui pustaka pemrograman seperti *OpenCV*. Dalam konteks pembelajaran dan praktikum *computer vision*, *webcam* menjadi pilihan utama

karena kemudahan penggunaan, biaya yang relatif murah, serta kemampuannya untuk melakukan pengambilan citra secara *real-time* (Rosnelly et al., 2020). *Webcam* USB khususnya bersifat *plug-and-play* sehingga langsung dapat digunakan tanpa memerlukan instalasi driver tambahan pada sistem operasi modern.

2.15 Python

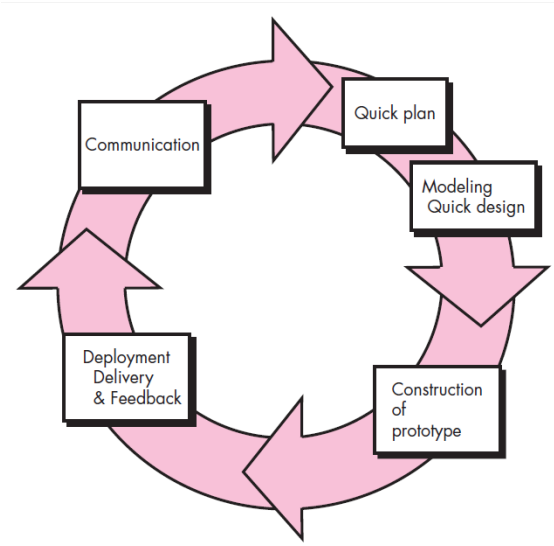
Python adalah bahasa pemrograman yang menggunakan *interpreter* dalam mengeksekusi kode programnya, sehingga kode dapat diterjemahkan dan dijalankan secara langsung tanpa perlu dikompilasi terlebih dahulu. Python bersifat *cross-platform*, artinya dapat dijalankan di berbagai sistem operasi seperti Windows, Linux, dan lainnya. Selain itu, Python mengadopsi berbagai paradigma pemrograman sekaligus, mulai dari paradigma prosedural seperti bahasa C, berorientasi objek seperti Java, hingga pemrograman fungsional seperti Lisp. Kombinasi dari berbagai paradigma tersebut menjadikan Python fleksibel dan memudahkan para pengembang dalam membangun berbagai jenis proyek (Rahman et al., 2023).

Dalam penelitian ini, Python digunakan sebagai bahasa pemrograman utama untuk menjalankan *framework* MediaPipe dalam proses deteksi dan pelacakan wajah, sekaligus sebagai perantara komunikasi antara sistem dengan mikrokontroler ESP-32.

2.16 Metode Prototyping

Metode prototyping merupakan salah satu metode pengembangan sistem yang dilakukan secara bertahap dan iteratif. Prototyping adalah sebuah metode rekayasa perangkat lunak yang digunakan untuk membangun sebuah aplikasi atau sistem ketika kebutuhan yang diperlukan belum sepenuhnya jelas (Pressman, 2005). Dalam metode ini, pengembang dan pengguna bekerja sama secara aktif untuk mendefinisikan kebutuhan sistem, merancang prototype awal, menguji, dan memperbaiki sistem secara berkelanjutan hingga sistem yang dihasilkan memenuhi kebutuhan yang diharapkan. Metode prototyping dipilih dalam penelitian ini karena sesuai dengan karakteristik penelitian pengembangan sistem yang memerlukan

proses evaluasi dan perbaikan secara iteratif. Tahapan dalam metode prototyping adalah sebagai berikut:



Gambar 2. 6 Metode Prototyping
(Sumber: (Pressman, 2005))

A. *Communication*

Pada tahap ini dilakukan proses pengumpulan informasi mengenai kebutuhan sistem yang akan dikembangkan. Informasi yang diperoleh digunakan untuk mengidentifikasi kebutuhan pengguna sebagai dasar dalam proses pengembangan sistem.

B. *Quick Plan dan Modelling Quick Design*

Pada tahap ini dilakukan analisis terhadap kebutuhan sistem yang telah diperoleh pada tahap *communication*. Analisis meliputi kebutuhan pengguna dan kebutuhan teknologi yang mendukung pengembangan sistem. Selanjutnya, hasil analisis tersebut digunakan sebagai acuan dalam merancang model atau desain awal sistem.

C. *Construction of Prototype*

Pada tahap ini dilakukan pembangunan *prototype* berdasarkan hasil analisis kebutuhan dan desain sistem yang telah ditentukan sebelumnya. Prototype

yang dihasilkan digunakan sebagai representasi awal dari sistem yang akan dikembangkan.

D. *Deployment, Delivery, and Feedback*

Pada tahap ini dilakukan pengujian dan evaluasi terhadap prototype yang telah dibangun. Pengguna memberikan umpan balik mengenai kesesuaian sistem dengan kebutuhan yang diharapkan. Jika masih terdapat kekurangan, maka dilakukan perbaikan dengan kembali ke tahap sebelumnya. Proses ini berlangsung secara iteratif hingga prototype yang dihasilkan memenuhi kebutuhan pengguna.

2.17 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) merupakan bahasa pemodelan standar yang digunakan untuk memvisualisasikan, merancang, dan mendokumentasikan suatu sistem. UML menyediakan berbagai jenis diagram yang dapat menggambarkan struktur maupun perilaku sistem, sehingga memudahkan proses analisis, perancangan, dan komunikasi antar pengembang (Rumbaugh et al., 2003). Dalam pengembangan perangkat lunak, UML banyak dimanfaatkan sebagai alat bantu dalam merancang sistem karena mampu menggambarkan alur kerja sistem sesuai dengan kebutuhan pengguna (Hidayati et al., 2023).

2.17.1 *Use Case Diagram*

Use Case Diagram merupakan salah satu diagram pada *Unified Modeling Language* (UML) yang digunakan untuk menggambarkan fungsionalitas sistem dari sudut pandang pengguna actor. Diagram ini memperlihatkan interaksi antara aktor dengan sistem melalui berbagai fungsi atau layanan yang disediakan, tanpa menjelaskan bagaimana proses internal dari fungsi tersebut dijalankan. Dengan demikian, *Use Case Diagram* membantu dalam mengidentifikasi kebutuhan fungsional sistem serta hubungan antara pengguna dan sistem (Rumbaugh et al., 2003).

Tabel 2. 2 Notasi *Use Case Diagram*

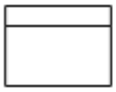
Simbol	Nama	Keterangan
	Actor	Merepresentasikan pengguna atau entitas yang berinteraksi dengan sistem.
	Use Case	Menunjukkan fungsi atau layanan yang disediakan oleh sistem.
	Association	Menunjukkan hubungan atau interaksi antara aktor dengan use case.

Use Case Diagram menggunakan beberapa notasi untuk merepresentasikan aktor, fungsi sistem, serta hubungan di antara keduanya. Setiap notasi memiliki fungsi yang berbeda dalam menggambarkan interaksi antara pengguna dengan sistem sehingga memudahkan pemahaman terhadap kebutuhan fungsional sistem. Adapun notasi yang digunakan pada *Use Case Diagram* dalam penelitian ini dapat dilihat pada Tabel 2.2.

2.17.2 *Activity Diagram*

Activity Diagram merupakan salah satu diagram pada UML yang digunakan untuk menggambarkan urutan aktivitas atau *workflow* dalam suatu sistem, baik yang berlangsung secara berurutan maupun secara bersamaan. Diagram ini memudahkan pemahaman terhadap alur kerja sistem melalui penggunaan berbagai notasi yang merepresentasikan aktivitas, percabangan, serta hubungan antar aktivitas (Rumbaugh et al., 2003).

Tabel 2. 3 Notasi *Activity Diagram*

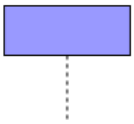
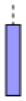
Simbol	Nama	Keterangan
	Status Awal (Start)	Menunjukkan awal aktivitas pada Activity Diagram
	Aktivitas	Aktivitas atau proses yang dilakukan oleh aktor maupun sistem
	Percabangan / Decision	Menunjukan percabangan proses berdasarkan suatu kondisi tertentu
	Penggabungan / Join	Penggabungan dimana lebih dari satu aktivitas lalu digabungkan jadi satu
	Status Akhir (End)	Menunjukkan akhir dari aktivitas pada Activity Diagram.
	Swimline	Membagi aktivitas berdasarkan aktor atau komponen yang bertanggung jawab terhadap proses yang dilakukan.

Activity Diagram menggunakan beberapa notasi atau simbol untuk merepresentasikan alur aktivitas yang terjadi di dalam sistem. Setiap simbol memiliki fungsi yang berbeda, seperti menunjukkan awal dan akhir aktivitas, proses yang dilakukan, percabangan keputusan, serta pembagian aktivitas berdasarkan aktor atau komponen yang terlibat. Adapun simbol-simbol yang digunakan pada Activity Diagram dalam penelitian ini dapat dilihat pada Tabel 2.3.

2.17.3 *Sequence Diagram*

Sequence Diagram merupakan salah satu *interaction diagram* pada Unified Modeling Language (UML) yang digunakan untuk menggambarkan urutan interaksi atau pertukaran pesan antar aktor maupun komponen dalam suatu sistem berdasarkan urutan waktu. Diagram ini memperlihatkan bagaimana setiap komponen saling berkomunikasi melalui pengiriman dan penerimaan pesan untuk menjalankan suatu fungsi, sehingga alur komunikasi sistem dapat dipahami tanpa menjelaskan implementasi internal dari setiap komponen (Rumbaugh et al., 2003).

Tabel 2. 4 Notasi *Sequence Diagram*

Simbol	Nama	Keterangan
	Aktor	Merepresentasikan pengguna atau entitas yang berinteraksi dengan sistem.
	Lifeline	Merepresentasikan objek atau komponen yang berpartisipasi dalam interaksi sistem.
	Activition	Menunjukkan periode ketika suatu objek sedang menjalankan proses atau operasi.
	Message	Menunjukkan pesan atau permintaan yang dikirim dari satu objek ke objek lainnya.
	Return Message	Menunjukkan balasan atau hasil dari proses yang telah dilakukan oleh suatu objek.

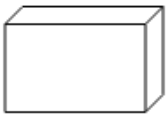
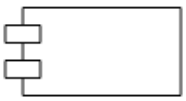
Sequence Diagram menggunakan beberapa notasi untuk menggambarkan interaksi antar objek berdasarkan urutan waktu. Pada diagram ini, sumbu vertikal menunjukkan urutan waktu pelaksanaan proses, sedangkan sumbu horizontal merepresentasikan objek atau komponen yang saling berinteraksi. Setiap objek digambarkan menggunakan *lifeline*, sementara periode ketika objek menjalankan suatu proses ditunjukkan oleh *activation*. Interaksi antar objek direpresentasikan melalui *message* sebagai pesan yang dikirim dan return message sebagai balasan dari proses yang telah dijalankan (Rumbaugh et al., 2003). Adapun notasi yang digunakan pada Sequence Diagram dalam penelitian ini dapat dilihat pada Tabel 2.4.

2.17.4 *Deployment Diagram*

Deployment Diagram merupakan diagram yang menggambarkan arsitektur hubungan antara perangkat keras (*hardware*) dan perangkat lunak (*software*) secara visual dalam suatu sistem (Wildan et al., 2021). Selain itu, *deployment diagram* juga menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi, sehingga dapat memberikan gambaran yang jelas mengenai bagaimana komponen perangkat

lunak didistribusikan dan dijalankan pada perangkat keras yang terlibat (Siswidiyanto et al., 2020).

Tabel 2. 5 Notasi *Deployment Diagram*

Simbol	Nama	Keterangan
	Node	Menunjukkan perangkat keras atau lingkungan eksekusi yang digunakan untuk menjalankan perangkat lunak
	Komponen	Menunjukkan modul perangkat lunak yang memiliki fungsi tertentu serta dapat berinteraksi dengan komponen lain dalam sistem.
	Artefak	Menunjukkan bentuk fisik perangkat lunak yang digunakan atau dihasilkan oleh sistem
	Relasi antar node	Hubungan komunikasi atau koneksi fisik yang menghubungkan dua node
	Kebergantungan	Hubungan ketergantungan antar elemen, di mana suatu node atau komponen memerlukan layanan atau fungsi dari elemen lainnya.

Adapun notasi yang digunakan pada *Deployment Diagram* dalam penelitian ini dapat dilihat pada Tabel 2.5. *Deployment Diagram* menggunakan beberapa notasi untuk menggambarkan hubungan antara perangkat keras dan perangkat lunak secara visual. Node merepresentasikan objek fisik pada saat *runtime* yang mewakili sumber daya komputasi seperti CPU, perangkat, dan memori, yang digambarkan dalam bentuk kubus dengan nama node di dalamnya. Objek atau komponen yang berada di dalam suatu node ditunjukkan dengan cara meletakkan simbol objek tersebut secara bersarang (*nested*) di dalam simbol node. Relasi antar node merepresentasikan jalur komunikasi, dimana *stereotype* dapat ditambahkan untuk membedakan jenis jalur komunikasi yang digunakan (Rumbaugh et al., 2003).

2.18 Pengujian Blackbox

Black Box Testing merupakan metode pengujian perangkat lunak yang berfokus pada pengujian fungsi sistem tanpa memperhatikan detail perangkat lunak maupun kode program yang digunakan. Pengujian dilakukan dengan memberikan berbagai masukan (*input*) sesuai skenario yang telah ditentukan, kemudian membandingkan keluaran (*output*) yang dihasilkan dengan hasil yang diharapkan. Dengan metode ini, setiap fungsi pada sistem dapat divalidasi untuk memastikan bahwa sistem telah berjalan sesuai dengan kebutuhan dan spesifikasi yang telah ditetapkan (Hardika et al., 2024).

Berbagai penelitian terdahulu menunjukkan bahwa metode *Black Box Testing* efektif digunakan untuk memverifikasi fungsionalitas sistem. Pengujian dilakukan dengan menyusun *test case* berdasarkan spesifikasi sistem, kemudian mengevaluasi apakah setiap fungsi menghasilkan keluaran yang sesuai dengan perilaku yang diharapkan (Zen et al., 2024). Oleh karena itu, metode ini banyak digunakan untuk memastikan bahwa seluruh fitur pada sistem dapat beroperasi dengan baik sesuai tujuan pengembangannya.

2.19 Gambaran Umum Perusahaan

PT. Radar Telekomunikasi Indonesia adalah perusahaan teknologi pertahanan dan telekomunikasi yang berbasis di Bandung, Jawa Barat. Perusahaan ini didirikan pada 5 Februari 2016 dan bergerak di bidang pengembangan radar, telekomunikasi, elektronika, dan sistem informasi, dengan misi untuk mengembangkan produk teknologi buatan dalam negeri dengan Tingkat Komponen Dalam Negeri (TKDN) yang tinggi.

Gambar 2. 7 Logo PT. Radar Telekomunikasi Indonesia

2.19.1 Visi dan Misi Perusahaan

A. Visi Perusahaan

Menjadi perusahaan nasional yang *unggul, inovatif, dan kompetitif* di teknologi radar, telekomunikasi, dan produk elektronik, serta mampu bersaing dengan perusahaan nasional dan internasional.

B. Misi Perusahaan

- Mengembangkan produk teknologi yang inovatif sesuai kemajuan teknologi dan kebutuhan pasar.
- Menghasilkan produk berkualitas tinggi dan berdaya saing.
- Mengembangkan produk profesional dan berkualitas tinggi di bidang telekomunikasi, elektronik pertahanan, dan layanan yang terkait dengan produk-produk tersebut
- Menjaga Hak Atas Kekayaan Intelektual (HAKI) sebagai bagian dari proses inovasi.