BAB II

KAJIAN PUSTAKA

A. Kajian Teoritis

1. Asisten Belajar

Asisten belajar merupakan suatu entitas atau sistem yang dirancang untuk mendampingi peserta didik dalam proses memahami materi. Perkembangan teknologi seperti kecerdasan buatan memungkinkan asisten belajar tidak hanya digunakan untuk mengerjakan tugas, tetapi juga bisa digunakan sebagai pendukung aktivitas akademik yang mampu membantu meningkatkan hasil belajar (Wardani et al., 2024). Pendapat lain menyimpulkan bahwa *chatbot* sebagai asisten pembelajaran berbasis kecerdasan buatan memiliki potensi untuk meningkatkan motivasi belajar siswa (Juanta et al., 2024). Melalui pendampingan yang responsif ini, asisten belajar terbukti mampu meningkatkan motivasi serta efisiensi waktu belajar mandiri secara signifikan.

2. Metode Pomodoro

Metode pomodoro merupakan sebuah teknik manajemen waktu dalam belajar yang mementingkan efisiensi waktu, dengan tuntutan pemusatan pikiran secara penuh dalam batasan kerangka waktu yang telah ditentukan. Melalui pembagian waktu yang terstruktur ini, seseorang diarahkan untuk menggunakan waktu secara teratur antara waktu dalam belajar dan beristirahat. Pembatasan durasi tersebut sangat efektif untuk menjaga fokus dan perhatian selama proses pembelajaran (Harits et al., 2024).

Dalam penerapannya Arviani et al. (2021) menambahkan, teknik pomodoro ini pada dasarnya melatih individu untuk meningkatkan durasi konsentrasi atau fokus ketika proses belajar sedang berlangsung. Konsentrasi yang maksimal akan membantu meningkatkan pemahaman dalam proses belajar. Adanya selingan jeda istirahat secara berkala di antara sesi fokus tersebut berfungsi mencegah pengguna agar tidak mudah merasa jenuh ketika menghadapi materi pelajaran yang kompleks.

3. *Large Language Mode*l (LLM)

LLM merupakan salah satu bentuk kecerdasan buatan hasil dari perkembangan teknologi jaringan saraf tiruan dan *Natural Language Processing* (NLP) untuk memproses serta menghasilkan teks bahasa. Model ini dilatih menggunakan data teks dalam jumlah besar dengan pendekatan *deep learning* sehingga mampu mempelajari pola, struktur, dan konteks bahasa manusia. Kemampuan tersebut membuat LLM dapat digunakan dalam berbagai tugas berbasis bahasa seperti menghasilkan teks, menjawab pertanyaan, meringkas informasi, melakukan penulisan kode, hingga analisis sentimen (Rachmat & Kesuma, 2024).

Menurut Kusuma et al (2025), LLM merupakan model yang dilatih menggunakan korpus teks dengan skala besar berbasis *deep learning* yang bertujuan untuk memahami sekaligus menghasilkan teks bahasa alami. Kemampuan LLM yang dalam memberi respons dengan bahasa manusia, serta pertumbuhannya yang luar biasa cepat memiliki potensi yang tinggi

untuk digunakan di berbagai aplikasi, sistem analisis otomatis, sampai asisten virtual yang merupakan salah satu bentuk dari adopsi penggunan LLM.

a. Dataset

Dataset merupakan sebuah informasi bernilai dari proses pengolahan kumpulan objek terstruktur yang berasal dari data mentah (Hafifah et al., 2021). Dalam konteks LLM, ketersediaan sekumpulan data berfungsi sebagai komponen penting dalam proses pengembangan model. Data tersebut digunakan pada fase pelatihan lanjutan untuk membantu model mengenali pola kalimat, gaya komunikasi, serta aturan respons.

Hermawan (2024) berpendapat, bahwa kualitas dataset menjadi faktor penting dalam proses pengembangan kecerdasan buatan karena dataset digunakan sebagai dasar dalam proses pelatihan, pengujian, dan validasi model AI. Data yang digunakan akan sangat memengaruhi kualitas model kecerdasan buatan. Jika dataset yang digunakan tidak akurat atau tidak mencerminkan kebutuhan sistem, maka performa model dapat menurun atau bahkan tidak bisa di terapkan dalam sistem yang sedang di bangun.

b. *Fine-Tuning*

Fine-tuning merupakan sebuah proses optimasi yang mengubah atau menyesuaikan suatu model yang telah dilatih agar mampu mengeksekusi tugas baru yang serupa (Hafifah et al., 2021). Secara lebih spesifik, teknik ini juga didefinisikan sebagai proses penyesuaian *pre-trained* model dengan sekumpulan data atau tugas yang lebih spesifik,

untuk meningkatkan performanya di dalam domain spesifik tersebut (Ovadia et al., 2024). Melalui pendekatan ini, sistem asisten pengingat belajar dapat diwujudkan agar memiliki kemampuan untuk merespons sesuai kondisi spesifik yang dibutuhakan tanpa harus membangun model dari nol.

c. *Quantization* dan Format GGUF

Menurut Gong et al. (2024), *quantization* atau kuantisasi merupakan teknik yang digunakan untuk mengubah bobot dan aktivasi jaringan saraf menggunakan presisi bit yang lebih rendah untuk menekan kebutuhan komputasi dan memori model. Pada LLM, teknik ini sangat krusial diterapkan untuk mengurangi biaya implementasi model yang memiliki jumlah parameter sangat besar. Proses kuantisasi umumnya dilakukan dengan mengubah representasi data bobot model dari presisi tinggi, menjadi format 8-bit atau 4-bit tanpa menyebabkan penurunan performa kognitif yang terlalu signifikan. Melalui pendekatan tersebut, model LLM dapat dieksekusi secara lebih ringan dan efisien dibandingkan dengan penggunaan model berpresisi penuh.

Quantization telah menjadi teknik praktis yang mempermudah proses deployment LLM pada perangkat keras dengan sumber daya terbatas atau komputer berspesifikasi menengah. Dalam implementasi komputasi lokal, salah satu standar format yang banyak diadopsi adalah *GPT-Generated Unified Format* (GGUF) pada *framework* llama.cpp. Format GGUF digunakan untuk menyimpan model hasil kuantisasi

dengan ukuran file yang jauh lebih efisien serta mendukung berbagai skema 3-bit hingga 8-bit (Kurt, 2026). Dengan melakukan kuantisaasi dan memakai format GGUF, model bahasa untuk asisten pengingat belajar dapat dieksekusi dengan performa yang lebih optimal, responsif, dan ramah terhadap kapasitas perangkat keras.

4. *Internet of Things* (IoT)

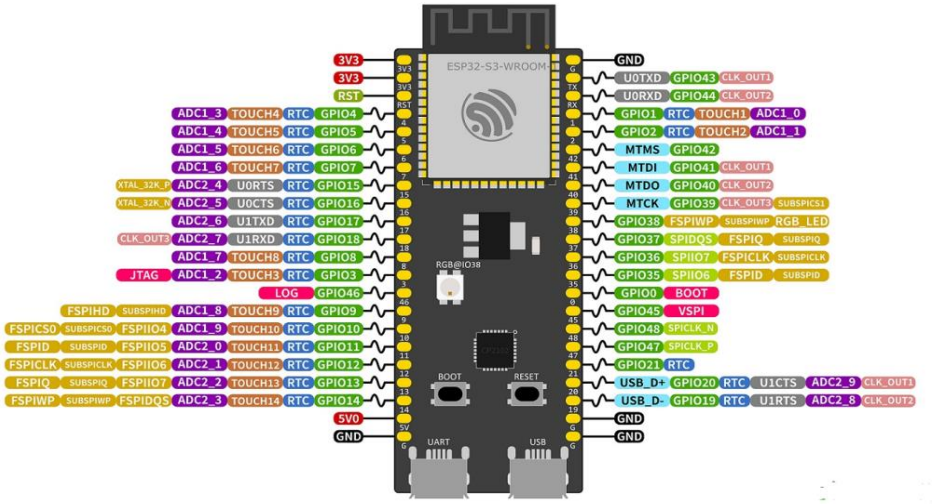
IoT merupakan teknologi yang memungkinkan berbagai perangkat dan sistem saling terhubung melalui jaringan internet sehingga dapat melakukan pertukaran data antar perangkat (Selay et al., 2022). Iot juga dikenal sebagai sebuah sistem *embedded* yang dibuat dengan tujuan memperluas pemanfaatan konektvitas internet sebagai perangkat yang dapat terus-menerus terhubung. Selain itu, IoT juga dikembangkan dengan memanfaatkan berbagai komponen seperti sensor, *Wireless Sensor Network*, RFID, serta *smart object* lainnya (Susanto et al., 2022). Kemampuan tersebut menjadi alasan kenapa IoT banyak diterapkan pada berbagai bidang seperti pendidikan, kesehatan, industri, maupun perangkat pintar otomatisasi.

a. Mikrokontroler ESP32

Mikrokontroler ESP32 merupakan perangkat mikrokontroler yang banyak digunakan sering dijumpai dalam pengembangan sistem IoT karena kemampuannya melakukan pemrosesan data dan komunikasi jaringan. ESP32 termasuk jenis *System on Chip* (SoC) yang telah dilengkapi WiFi, Bluetooth, serta berbagai peripheral pendukung (Nizam et al., 2022). Selain itu, ESP32 juga memiliki prosesor *dual core* dengan

komputasi tinggi yang mendukung penerapan sistem monitoring secara *realtime* (Ar, 2025).

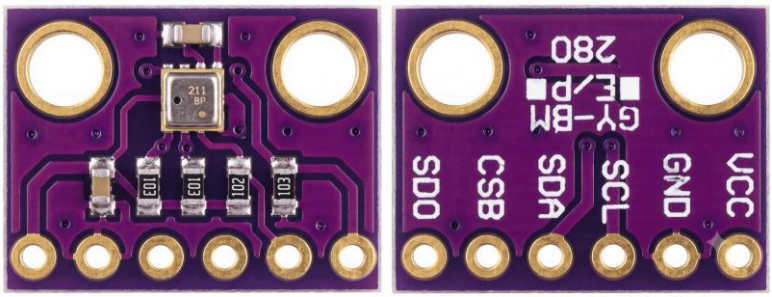
Dalam penelitian ini, perangkat yang digunakan adalah papan pengembangan ESP32-S3-DevKitC-1. Modul ini telah mendukung komunikasi WiFi dan *Bluetooth Low Energy* sehingga sesuai digunakan pada sistem IoT yang membutuhkan pertukaran data secara nirkabel. Selain itu, ESP32-S3 juga memiliki kemampuan komputasi yang lebih baik untuk mendukung pemrosesan data dan beban kerja komputasi berbasis jaringan saraf tiruan. Papan pengembangan ini menyediakan berbagai pin *General Purpose Input Output* (GPIO) yang digunakan sebagai penghubung dengan modul modul lain (Espressif Systems, 2026). Penelitian ini menggunakan varian ESP32-S3-DevKitC-1 N16R8 yang menyediakan kapasitas memori lebih besar untuk mendukung kebutuhan sistem. Bentuk fisik dan pin GPIO-nya terdapat pada Gambar 2.1.



Gambar 2.1 ESP32-S3-DevKitC-1
Sumber: Espressif Systems (2026)

b. BMP280

Sensor BMP280 merupakan sensor digital yang digunakan untuk mengukur barometrik atau tekanan udara dan suhu lingkungan. Penerapan teknologi *Micro Electro Mechanical Systems* (MEMS) membuat sensor ini memiliki ukuran yang kecil, hemat daya dan ringan sehingga cocok untuk berbagai kebutuhan perangkat elektronik (Marzuarman et al., 2024). Selain itu, berkat konsumsi daya BMP280 yang rendah sensor ini dapat diaplikasikan pada perangkat *mobile* seperti ponsel, GPS, atau jam tangan yang memerlukan efisiensi penggunaan energi (Manalu & Gunoto, 2023). Bentuk BMP280 tertera pada Gambar 2.2.

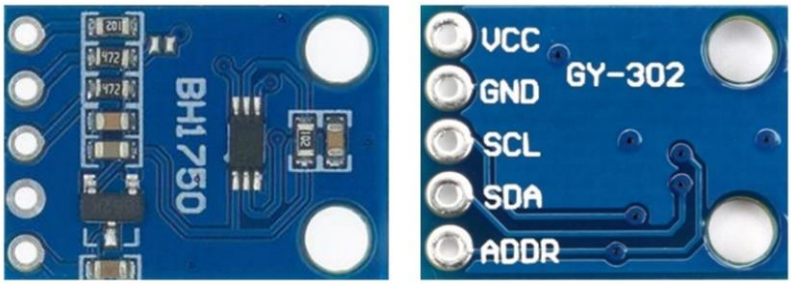


Gambar 2.2 BMP280
Sumber: Marzuarman et al. (2024)

c. BH1750

Sensor BH1750 merupakan sensor cahaya digital yang digunakan untuk mengukur intensitas cahaya pada lingkungan sekitar. Sensor ini menggunakan protokol komunikasi I2C dengan memanfaatkan pin SDA dan SCL untuk proses pertukaran data dengan mikrokontroler. Selain itu, BH1750 mampu mengubah nilai pencahayaan analog menjadi data digital lux sehingga lebih mudah digunakan pada sistem monitoring berbasis

mikrokontroler (Wulantika et al., 2023). Bentuk fisik sensor BH1750 dapat dilihat pada Gambar 2.3.



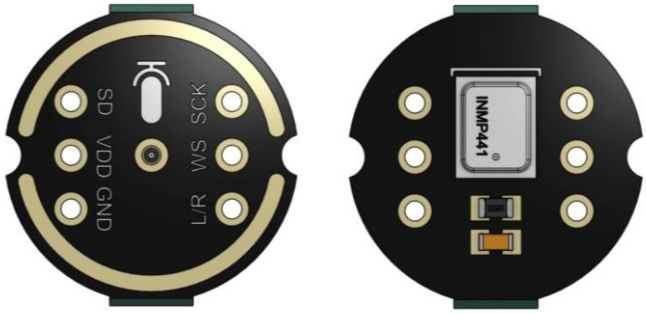
Gambar 2.3 BH1750
Sumber: Marzuarman et al. (2024)

Sensor BH1750 merupakan sensor untuk mengukur intensitas cahaya dalam satuan lux dengan memanfaatkan fotodioda yang sensitif terhadap cahaya. Cahaya yang diterima sensor akan diubah menjadi arus listrik, kemudian dikonversi menjadi data digital yang merepresentasikan tingkat pencahayaan lingkungan. Sensor ini memiliki tingkat akurasi pengukuran sekitar ±20% yang diklaim mampu mengukur intensitas cahaya dengan tepat pada rentang 1 lux hingga 65535 lux sehingga dapat digunakan pada berbagai kondisi pencahayaan, mulai dari lingkungan gelap hingga sangat terang (Marzuarman et al., 2024). Dalam penelitian ini, sensor BH1750 digunakan untuk memantau kondisi pencahayaan lingkungan belajar pengguna pada sistem asisten pengingat belajar.

d. INMP441

INMP441 merupakan mikrofon digital berbasis *Micro Electromechanical System* (MEMS) yang dirancang untuk kebutuhan pengukuran dan pemrosesan suara. Mikrofon ini termasuk jenis

omnidirectional atau mampu menerima gelombang suara dari berbagai arah dengan sensitivitas yang konsisten (Hazazi et al., 2025). Kemampuan tersebut, membuat mikrofon ini juga sesuai digunakan sebagai sensor untuk proses monitoring tingkat kebisingan suara lingkungan. Adapun bentuk fisiknya tertera pada Gambar 2.4.



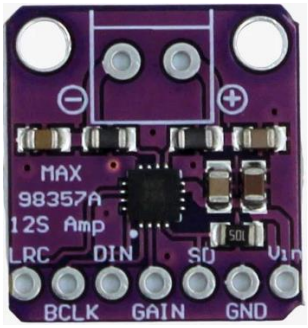
Gambar 2.4 INMP441
Sumber: Tyas & Irawan (2026)

Penggunaan jalur komunikasi I2S pada INMP441 memungkinkan data suara digital dikirim secara langsung ke mikrokontroler tanpa memerlukan *Analog to Digital Converter* (ADC) tambahan. Sensor ini juga memiliki tingkat *noise* yang rendah dan memiliki sensitivitas tinggi sehingga sesuai digunakan pada sistem *voice recognition*, perekaman suara, maupun perangkat IoT berbasis audio (Tyas & Irawan, 2026). Penelitian ini menggunakan INMP441 untuk mendeteksi tingkat kebisingan suara lingkungan belajar pengguna pada sistem asisten pengingat belajar.

e. MAX98357

Menurut Tyas & Irawan (2026) MAX98357 merupakan modul audio *amplifier* (penguat suara) berbasis I2S yang juga berfungsi sebagai

Digital to Analog Converter (DAC). Modul ini mampu menerima sinyal audio digital secara langsung dari mikrokontroler seperti ESP32 tanpa memerlukan rangkaian konverter atau DAC tambahan. Selain itu, MAX98357 dapat memperkuat keluaran suara sehingga mampu menggerakkan speaker 3W/8Ω secara langsung. Modul audio MAX98357 disertakan pada Gambar 2.5.



Gambar 2.5 MAX98357
Sumber: Tyas & Irawan (2026)

MAX98357 digunakan sebagai *low power amplifier* pada sistem yang menggunakan protokol komunikasi I2S, yang merupakan standar antarmuka digital untuk transmisi sinyal audio digital antar perangkat elektronik (Yauri & Espino, 2024). Modul MAX98357, digunakan di penelitian ini untuk menghasilkan efek suara pendukung pada sistem asisten pengingat belajar berdasarkan kondisi pemicu tertentu.

f. TFT ILI9341

Modul layar TFT ILI9341 merupakan layar LCD berbasis *Thin Film Transistor* (TFT) yang digunakan untuk menampilkan informasi visual dengan kualitas warna dan resolusi yang tinggi. Teknologi TFT pada layar ini memungkinkan setiap piksel dikendalikan secara aktif

sehingga mampu menampilkan teks, gambar, grafik, maupun animasi dengan kualitas visual yang baik dan menjadi alasan mengapa banyak digunakan pada sistem *embedded* dan IoT. Layar ini menggunakan jalur komunikasi SPI yang mendukung proses transfer data antara mikrokontroler dan display secara cepat dan stabil (Syam et al., 2024). Bentuk fisiknya dapat dilihat pada Gambar 2.6.

Gambar 2.6 TFT ILI9431
Sumber: Syam et al. (2024)

Layar TFT ILI9341 mendukung resolusi 320×240 piksel serta menggunakan jalur data 8-bit dua arah untuk proses komunikasi dengan mikrokontroler ESP32. Pengontrol layar ini menggunakan format warna 16-bit RGB565 yang memungkinkan tampilan memiliki kombinasi warna hingga sekitar 65.000 pada setiap piksel. Selain itu, pengaturan alamat piksel dilakukan berdasarkan susunan baris dan kolom sehingga tampilan visual dapat ditampilkan secara lebih terstruktur, detail, dan berwarna (Espino et al., 2024). Dalam penelitian ini, TFT ILI9341 digunakan untuk menampilkan animasi mimik wajah, teks respons, serta informasi kondisi lingkungan dan progres pomodoro pada perangkat IoT.

5. *Context-Aware System*

Context-Awareness merupakan kemampuan sistem untuk memanfaatkan informasi konteks pada berbagai domain aplikasi, termasuk dalam proses penyesuaian antarmuka pengguna agar interaksi sistem menjadi lebih adaptif (Stefanidi et al., 2022). Melalui pendekatan ini, sistem tidak hanya bekerja secara statis, tetapi mampu menyesuaikan perilaku dan respons berdasarkan kondisi yang terjadi di sekitar pengguna. Penerapan konsep *context-aware* memungkinkan sistem memberikan pengalaman interaksi yang lebih responsif karena keputusan sistem dipengaruhi oleh data lingkungan maupun kondisi pengguna secara langsung.

Pada sistem IoT, konsep *context-aware* dapat diterapkan dengan memanfaatkan sensor dan mikrokontroler untuk melakukan operasi khusus berdasarkan kondisi lingkungan. Data sensor yang dikumpulkan mikrokontroler tersebut kemudian dianalisis oleh sistem untuk menentukan respons yang sesuai (Tundo et al., 2024). Dalam penelitian ini, mikrokontroler ESP32 juga berperan sebagai pusat pengumpulan data sensor lingkungan yang digunakan sebagai konteks bagi LLM dalam menghasilkan respons sistem. Berdasarkan data tersebut, asisten pengingat belajar dapat memberikan interaksi yang dinamis berupa animasi mimik wajah, efek suara, dan teks yang menyesuaikan kondisi lingkungan belajar pengguna.

6. *Human Computer Interaction* (HCI)

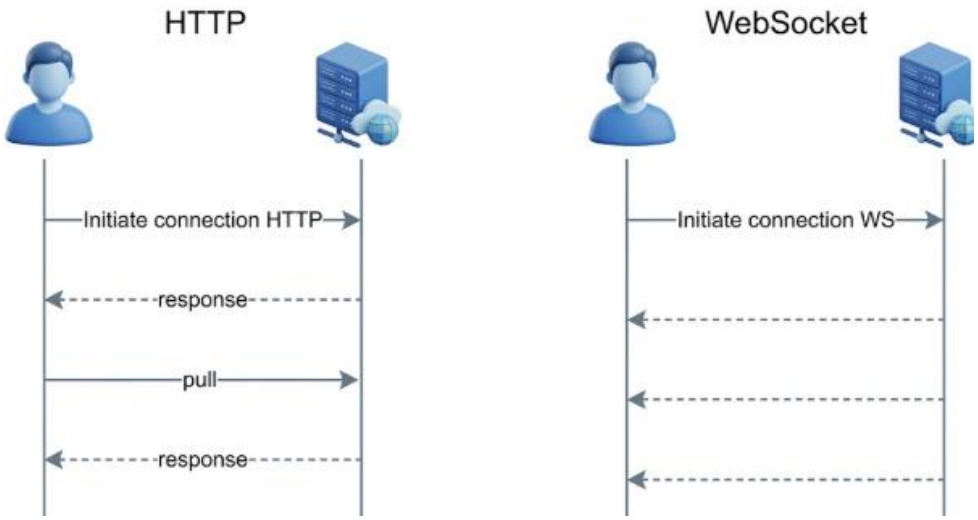
Menurut Mubarok et al. (2022), *Human Computer Interaction* (HCI) merupakan ilmu yang mempelajari hubungan interaksi antara manusia dan

komputer dalam proses penggunaan suatu sistem. Interaksi tersebut terjadi melalui komunikasi dua arah ketika pengguna memasukkan *input* ke komputer dan komputer mengembalikan *feedback* atau respons terhadap perintah yang diterima. HCI juga berkaitan dengan perancangan *user interface* dan *user experience* agar pengguna dapat berinteraksi dengan sistem secara lebih nyaman dan efektif.

HCI juga merupakan ilmu yang mempelajari hubungan antara manusia dengan teknologi komputer serta pengaruhnya terhadap aktivitas pengguna dalam menggunakan sistem komputer (Hamidah et al., 2023). Proses interaksi pada penelitian ini diwujudkan melalui sistem asisten pengingat belajar yang mampu memberikan respons dinamis berdasarkan kondisi lingkungan belajar pengguna. Respons tersebut dihasilkan menggunakan LLM yang telah disesuaikan untuk menghasilkan teks sesuai data suhu, cahaya, suara, dan fase pomodoro.

7. *Websocket*

Websocket merupakan protokol komunikasi yang memungkinkan pertukaran data dua arah secara *realtime* antara *client* dan *server* melalui koneksi yang tetap terbuka. Penggunaan *websocket* membuat aplikasi menjadi lebih responsif dan interaktif karena proses pengiriman data dapat dilakukan tanpa harus membuat koneksi baru setiap kali terjadi perubahan data (Putra et al., 2024). Mekanisme tersebut memungkinkan pertukaran data dapat terjadi secara langsung dengan *latency* yang lebih rendah.



Gambar 2.7 Mekanisme *Websocket* dan HTTP
Sumber: Nugraha (2024)

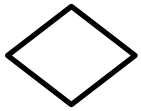
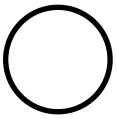
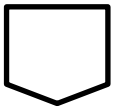
Pada Gambar 2.7, *websocket* terlihat mendukung komunikasi dua arah, protokol *socket* juga memungkinkan *server* mengirimkan data ke *client* tanpa harus menunggu *request* dari *client* terlebih dahulu. Berbeda dengan protokol HTTP yang komunikasinya bersifat satu arah dimana *client* harus selalu melakukn *request* untuk mendapatkan *response*. Karakteristik koneksi yang bersifat terus terhubung tersebut membuat *websocket* sesuai digunakan pada sistem yang membutuhkan pertukaran data secara cepat dan berkelanjutan (Nugraha, 2024).

8. *Flowchart*

Flowchart digunakan untuk menggambarkan urutan langkah atau alur proses dalam suatu sistem maupun program secara sistematis dan merupakan salah satu diagram visual yang sering digunakan (Listyoningrum et al., 2023). Diagram ini memanfaatkan simbol-simbol grafis dan panah sebagai penghubung untuk menunjukkan aliran proses, informasi, maupun

tahapan kerja dari satu langkah ke langkah berikutnya. Penggunaan *flowchart* membantu menggambarkan berbagai macam proses seperti prosedur operasional, alur kerja bisnis, algoritma pemrograman dan masih banyak lagi (Khairunisa et al., 2023). Simbol *Flowchart* tercantum pada Tabel 2.1.

Tabel 2.1 Simbol *Flowchart*

Simbol	Nama	Fungsi
	*Flow Line*	Menunjukkan arah alur proses antar simbol.
	*Terminator*	Menandai awal atau akhir proses.
	*Preparation*	Menunjukkan persiapan atau nilai awal proses.
	*Process*	Menunjukkan proses pengolahan atau eksekusi data.
	*Input/Output Data*	Menunjukkan proses masukan atau keluaran data.
	*Document*	Menunjukkan dokumen atau data tertulis.
	*Decision*	Menunjukkan percabangan berdasarkan kondisi tertentu.
	*On Page Connector*	Menghubungkan alur pada halaman yang sama.
	*Off Page Connector*	Menghubungkan alur ke halaman berbeda.

Sumber: Khairunisa et al. (2023)

9. *Unified Modeling Language* (UML)

Bahasa pemodelan grafis UML berfungsi untuk mempermudah dalam memahami rancangan perangkat lunak dengan memvisualisasikan, serta mendokumentasikan perancangan sistem perangkat lunak (Gunawan et al., 2023). UML membantu *developer* dalam memahami struktur dan alur kerja perangkat lunak sebelum implementasi dilakukan. Melalui gambaran visual berupa diagram, rancangan sistem dapat dijelaskan secara lebih terstruktur sehingga proses komunikasi antar pengembang dan *stakeholder* (pemangku kepentingan) dapat berjalan lancar.

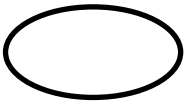
Menurut Narulita et al. (2024), UML juga berfungsi sebagai standar pembuatan *blueprint* pada pengembangan sistem *object oriented* (berorientasi objek). UML dapat digunakan untuk menggambarkan proses bisnis, rancangan *class*, basis data, hingga komponen yang dibutuhkan dalam pembangunan sistem. Dengan penggunaan UML, proses analisis dan pengembangan perangkat lunak dapat dilakukan secara lebih sistematis, terarah, dan mudah dipahami.

a. *Use Case Diagram*

Use case diagram merupakan bagian dari salah satu diagram UML yang dapat menggambarkan hubungan serta interaksi antara pengguna dengan sistem yang dikembangkan. Sistem harus berada dalam kotak sistem dan setidaknya punya satu aktor diluar kotak sistem (Hendri et al., 2022). Menurut Gunawan et al. (2023) diagram ini menampilkan aktor di luar sistem dan fitur-fitur utama yang dapat diakses atau

dijalankan oleh pengguna di dalam sistem. *Use case diagram* juga digunakan untuk merepresentasikan kebutuhan utama pengguna terhadap fungsi-fungsi yang tersedia pada sistem. Simbol *use case diagram* tercantum pada Tabel 2.2.

Tabel 2.2 Simbol *Use Case*

Simbol	Elemen	Keterangan
	Aktor	Menunjukkan pengguna, sistem lain, atau perangkat yang berinteraksi dengan sistem.
	*Use Case*	Menunjukkan fungsi atau layanan yang dapat dijalankan oleh aktor.
	*Assocation*	Menunjukkan hubungan antara aktor dengan *use case*.
	Generalisasi	Menunjukkan hubungan pewarisan atau pengkhususan antar aktor.
<<include>> 	*Include*	Menunjukkan *use case* yang selalu menjadi bagian dari *use case* lain.
<<extend>> 	*Extend*	Menunjukkan fungsi tambahan yang berjalan jika kondisi tertentu terpenuhi.

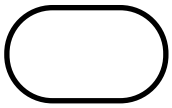
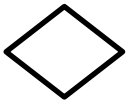
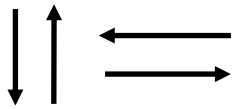
Sumber: Gunawan et al. (2023)

b. *Activity Diagram*

Activity diagram merupakan bagian dari diagram UML yang biasa dipakai sebagai penggambaran alur aktivitas atau proses kerja dalam sebuah sistem (Gunawan et al., 2023). Sedangkan menurut Narulita et al. (2024) *activity diagram* merepresentasikan urutan proses mulai dari

aktivitas awal, pengambilan keputusan, proses paralel, hingga proses berakhir di dalam sistem. Diagram ini juga digunakan untuk memvisualisasikan bagaimana sebuah *use case* dijalankan oleh aktor di dalam sistem, dan setiap *use case* setidaknya minimal harus ada satu *activity diagram*. *Activity diagram* bisa buat berdasarkan satu atau dari beberapa *use case*. Simbol *activity diagram* tercantum pada Tabel 2.3.

Tabel 2.3 Simbol *Activity Diagram*

Simbol	Elemen	Keterangan
	*Activity*	Menunjukkan aktivitas yang terjadi dalam alur sistem.
	*Action*	Menunjukkan aksi atau langkah yang dijalankan sistem.
	*Initial Node*	Menandai titik awal alur aktivitas.
	*Activity Final Node*	Menandai akhir dari seluruh alur aktivitas.
	*Decision*	Menunjukkan percabangan berdasarkan kondisi tertentu.
	*Line Connector*	Menghubungkan satu aktivitas dengan aktivitas lainnya.

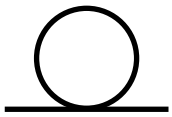
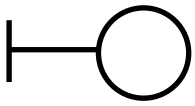
Sumber: Gunawan et al. (2023)

c. *Sequence Diagram*

Sequence diagram termasuk bagian dari salah satu diagram UML yang berfungsi untuk menggambarkan interaksi antar objek dalam sistem secara berurutan berdasarkan waktu (Gunawan et al., 2023). Diagram ini

memperlihatkan rangkaian *message* (pesan) yang terjadi antar objek dalam satu *use case*. *Sequence diagram* juga menggambarkan kolaborasi dinamis antar objek yang mengirimkan rangkaian *message* ketika sistem sedang dijalankan (Narulita et al., 2024). Simbol *sequence diagram* tercantum pada Tabel 2.4.

Tabel 2.4 Simbol *Sequence Diagram*

Simbol	Elemen	Keterangan
	*Actor*	Menunjukkan pengguna atau pihak yang berinteraksi dengan sistem.
	*Entity Class*	Menunjukkan objek data atau informasi yang dikelola dalam sistem.
	*Boundary Class*	Menjadi penghubung antara pengguna dan sistem.
	*Control Class*	Mengatur alur proses antara *boundary* dan *entity*.
	*A Focus of Control* & *A Life Line*	Menunjukkan waktu aktif objek saat menerima atau menjalankan pesan.
	A Message	Menunjukkan pesan atau perintah yang dikirim antar objek.

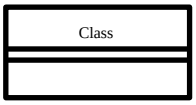
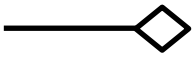
Sumber: Gunawan et al. (2023)

d. *Class Diagram*

Class diagram merupakan diagram yang menunjukkan hubungan antar *class* serta penjelasan detail mengenai atribut, aturan, dan tanggung jawab setiap entitas dalam sistem. *Class diagram* juga dapat dipahami

sebagai visualisasi struktur program yang memperlihatkan kumpulan *class* beserta relasi yang terbentuk di dalam sistem (Ramdany et al., 2024). Pendapat lain mengatakan bahwa *class* merupakan gambaran yang mewakili struktur dan prilaku dari objek. Setiap objek memiliki nilai atribut, dan juga menunjukkan hubungannya dengan *object* lain (Wayahdi & Ruziq, 2023). Simbol *class diagram* tercantum pada Tabel 2.5.

Tabel 2.5 Simbol *Class Diagram*

Simbol	Elemen	Keterangan
	*Class*	Menunjukkan blok-blok yang mendefinisikan objek dan atribut dalam sistem.
	Association	Menunjukkan hubungan setara antar *class* dalam sistem.
	*Aggregation*	Menunjukkan hubungan *class* utama dan bagian, namun objek bagian masih dapat berdiri sendiri.
	*Composition*	Menunjukkan hubungan *class* utama dan bagian, dan objek bagian tidak dapat berdiri sendiri.
	*Generalization*	Menunjukkan hubungan pewarisan dari *class* khusus ke *class* yang lebih umum.
	*Dependency*	Menunjukkan hubungan ketergantungan sementara, perubahan satu *class* dapat mempengaruhi *class* lain .

Sumber: Ramdany et al. (2024)

10. Diagram Blok

Diagram blok sistem merupakan bentuk visual yang digunakan untuk menggambarkan komponen-komponen dengan fungsi yang berbeda serta hubungan antar komponen dalam suatu sistem. Diagram blok membantu

mempermudah pemahaman, analisis, dan perancangan sistem (Nasution et al., 2023). Pendapat lain mengatakan, diagram blok menggambarkan suatu komponen yang memiliki fungsi tertentu, di mana setiap komponen saling berhubungan dan memengaruhi komponen lainnya sehingga membentuk satu kesatuan sistem yang utuh (Sepudin & Abdullah, 2022).

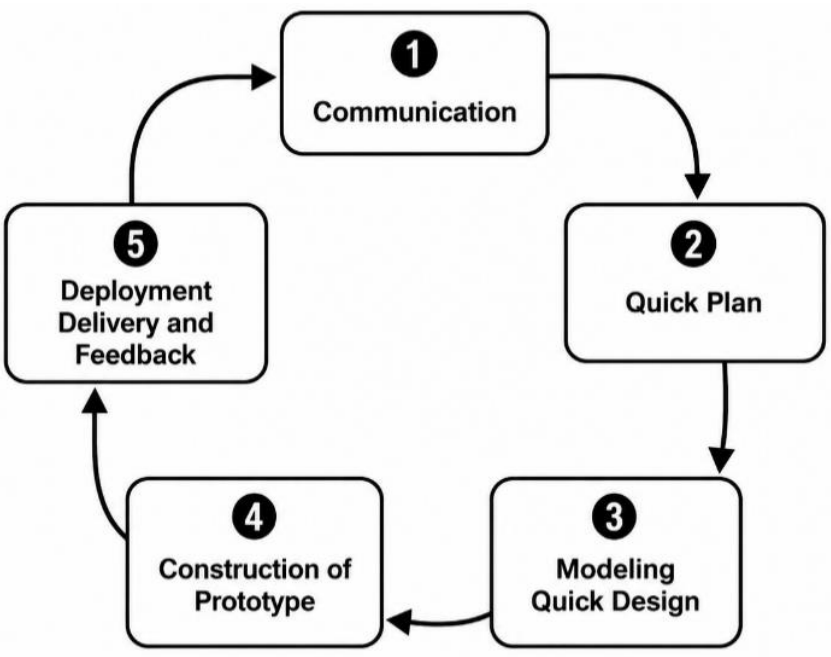
11. SQLite

SQLite merupakan sistem manajemen basis data yang digunakan untuk menyimpan dan mengelola data pada aplikasi (Yuliani, 2024). SQLite juga merupakan salah satu mesin basis data SQL yang banyak digunakan karena mendukung tipe data dasar, seperti *Text* untuk menyimpan data berbentuk teks, *Integer* untuk menyimpan data angka berbentuk bilangan bulat, dan *Real* untuk menyimpan data angka berbentuk bilangan desimal. SQLite juga tidak melakukan validasi secara ketat terhadap kesesuaian tipe data dengan tipe kolom yang telah didefinisikan, sehingga data dengan tipe yang berbeda tetap dapat disimpan pada kolom tertentu (Putra et al., 2020).

12. Metode *Prototype*

Metode *prototype* merupakan pendekatan pengembangan sistem yang dilakukan dengan membuat model awal dari sistem yang akan dibangun. Metode *prototype* bergantung pada hasil umpan balik dari pengguna akhir dalam proses pengembangannya. Umpan balik tersebut digunakan untuk menyempurnakan sistem agar lebih sesuai dengan kebutuhan pengguna, sekaligus membantu mengurangi risiko kesalahan, biaya pengembangan,

serta kemungkinan sistem tidak sesuai dengan kebutuhan yang diharapkan (Novitasari et al., 2025).



Gambar 2.8 Tahapan Metode *Prototype*
Sumber: Syarif & Risdiansyah (2024)

Pada Gambar 2.8, terdapat alur proses metode *prototype* dimulai dari tahap *communication* yang dilakukan untuk menggali kebutuhan pengguna dan memahami permasalahan yang akan diselesaikan, lalu *quick plan* digunakan untuk menyusun rencana awal pengembangan secara cepat dan efisien. Setelah itu, tahap *modeling quick design* dilakukan untuk membuat rancangan awal sistem, kemudian dilanjutkan dengan *construction of prototype* sebagai tahap pembentukan *prototype* berdasarkan rancangan yang telah dibuat. Tahap terakhir *deployment delivery and feedback*, yaitu proses penyerahan hasil *prototype* kepada pengguna untuk diuji dan diberi umpan balik sebagai dasar penyempurnaan sistem (Syarif & Risdiansyah, 2024).

13. Pengujian

a. *Black Box Testing*

Black box testing merupakan metode pengujian fungsional yang dilakukan berdasarkan spesifikasi sistem tanpa perlu mengetahui struktur internal, desain, maupun kode program yang digunakan. *Black box* testing berfokus untuk memastikan bahwa setiap fungsi, masukan, dan keluaran pada sistem telah berjalan sesuai dengan kebutuhan yang telah ditentukan (Zen et al., 2024).

Pendapat lain menjelaskan, metode *testing* ini dapat digunakan untuk mengetahui kelemahan sistem, terutama ketika sistem masih menerima masukan data yang tidak sesuai sehingga berpotensi menghasilkan data yang kurang valid. Pengujian *black box* juga biasa digunakan untuk melihat kesesuaian hasil eksekusi sistem terhadap data masukan, sekaligus menemukan kelemahan fungsi sebelum aplikasi digunakan secara langsung oleh pengguna (Febriyanti et al., 2021).

b. *Cosine Similiarity*

Menurut Sihombing & Nasution (2024) *cosine similarity* merupakan metode yang digunakan untuk menghitung tingkat kesamaan antara dua dokumen atau teks. Nugroho et al. (2021) menjelaskan bahwa *cosine similarity* dapat digunakan untuk menghitung kemiripan suatu dokumen, sehingga metode ini sesuai untuk digunakan dalam proses evaluasi kesesuaian teks atau jawaban. Rumus persamaan matematis untuk menghitung nilai kemiripan tersebut didefinisikan pada persamaan 2.1.

$$\boldsymbol{CosSim} \propto= \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{|\boldsymbol{A}||\boldsymbol{B}|} = \frac{\sum_{i=1}^{n} \boldsymbol{A_i} \times \boldsymbol{B_i}}{\sqrt{\sum_{i=1}^{n} (\boldsymbol{A_i})^2} \times \sqrt{\sum_{i=1}^{n} (\boldsymbol{B_i})^2}} \qquad (2.1)$$

Berdasarkan rumus Persamaan 2.1, variabel A merupakan Vektor A dan variabel B merupakan Vektor B yang akan dibandingkan kemiripannya. Adapun variabel A.B mendefinisikan *dot product* di antara vektor A dan vektor B. Sementara itu, variabel |A| merupakan panjang vektor A, dan variabel |B| merupakan panjang vektor B.

B. Kajian Empiris

Penelitian yang dilakukan oleh Ardilah & Affandi (2025) membahas efektivitas metode pomodoro untuk mengurangi tingkat kebosanan akademik pada proses pembelajaran siswa sekolah dasar. Penelitian tersebut menggunakan metode eksperimen dengan menerapkan teknik pomodoro dengan siklus 15 menit fokus dan 5 menit istirahat selama 6 minggu. Penyesuian teknik pomodoro terbukti efektif menurunkan tingkat kebosanan akademik dibandingan pembelajaran konvensional. Walaupun terbukti menurunkan tingkat kebosanan, penetapan waktu siklus pomodoro masih terlalu kaku.

Disisi lain perkembangan LLM yang semakin cerdas bisa dimanfaatkan untuk memberikan umpan balik respon, terkait kondisi lingkungan dan pengingat progres pembelajaran yang menerapkan metode pomodoro. Zhang et al. (2025) mengembangkan *chatbot* sadar lingkungan (*context-aware*) berbasis LLM yang memanfaatkan data sensor *smartphone* untuk memahami kondisi personal pengguna. Pendekatan *context-aware* tersebut mampu menghasilkan akurasi pengenalan konteks sebesar 84% dan memberikan pengalaman interaksi

yang lebih natural. Namun, Julfan & Haifaturrahmah (2025) menjelaskan dampak dari penggunaan gawai pintar terhadap fokus belajar siswa. Meskipun gawai memiliki dampak positif dalam mendukung pembelajaran, tetapi juga berpotensi menurunkan fokus belajar jika digunakan secara berlebihan.

Oleh karena itu, dibutuhkan perangkat keras terpisah agar proses pemantauan lingkungan dan siklus pomodoro tidak mengganggu fokus pengguna. Selviani & Wahyono (2022) mengembangkan sistem monitoring lingkungan ruang pembelajaran berbasis ESP32 yang mampu mengukur cahaya, suara, suhu, dan kelembaban udara dalam satu sistem terintegrasi. Hasil penelitian menunjukkan bahwa alat ukur yang dikembangkan memperoleh kategori "Sangat Baik" dan dinyatakan layak digunakan berdasarkan validasi ahli media dan guru fisika.

Lebih lanjut, penelitian Helmy et al. (2024) membahas penerapan teknologi *cloud computing* dan *edge computing* pada sistem IoT untuk pengendalian parameter budidaya tanaman hidroponik berbasis *machine learning*. Penelitian tersebut menunjukkan bahwa *edge computing* dapat digunakan untuk memproses langsung data dari perangkat sensor, sedangkan *cloud computing* dimanfaatkan untuk menyediakan kapasitas penyimpanan dan pemrosesan data yang lebih besar. Kombinasi kedua teknologi tersebut memungkinkan sistem berjalan lebih *scalable*, efisien, dan dapat diakses dari mana saja melalui jaringan internet.

Berdasarkan kajian penelitian terdahulu, dapat disimpulkan bahwa pengembangan asisten pintar saat ini masih memiliki keterbatasan. Penggunaan

smartphone sebagai media pengumpulan data sensor untuk membuat LLM merespons sesuai kondisi lingkungan memang mampun menjadi solusi, namun penggunaan gawai, berpotensi menyebabkan distraksi yang mengganggu proses belajar pengguna. Di sisi lain, sistem IoT berbasis sensor lingkungan umumnya masih berfokus pada fungsi monitoring dan belum banyak diintegrasikan dengan kemampuan interaksi cerdas berbasis LLM.

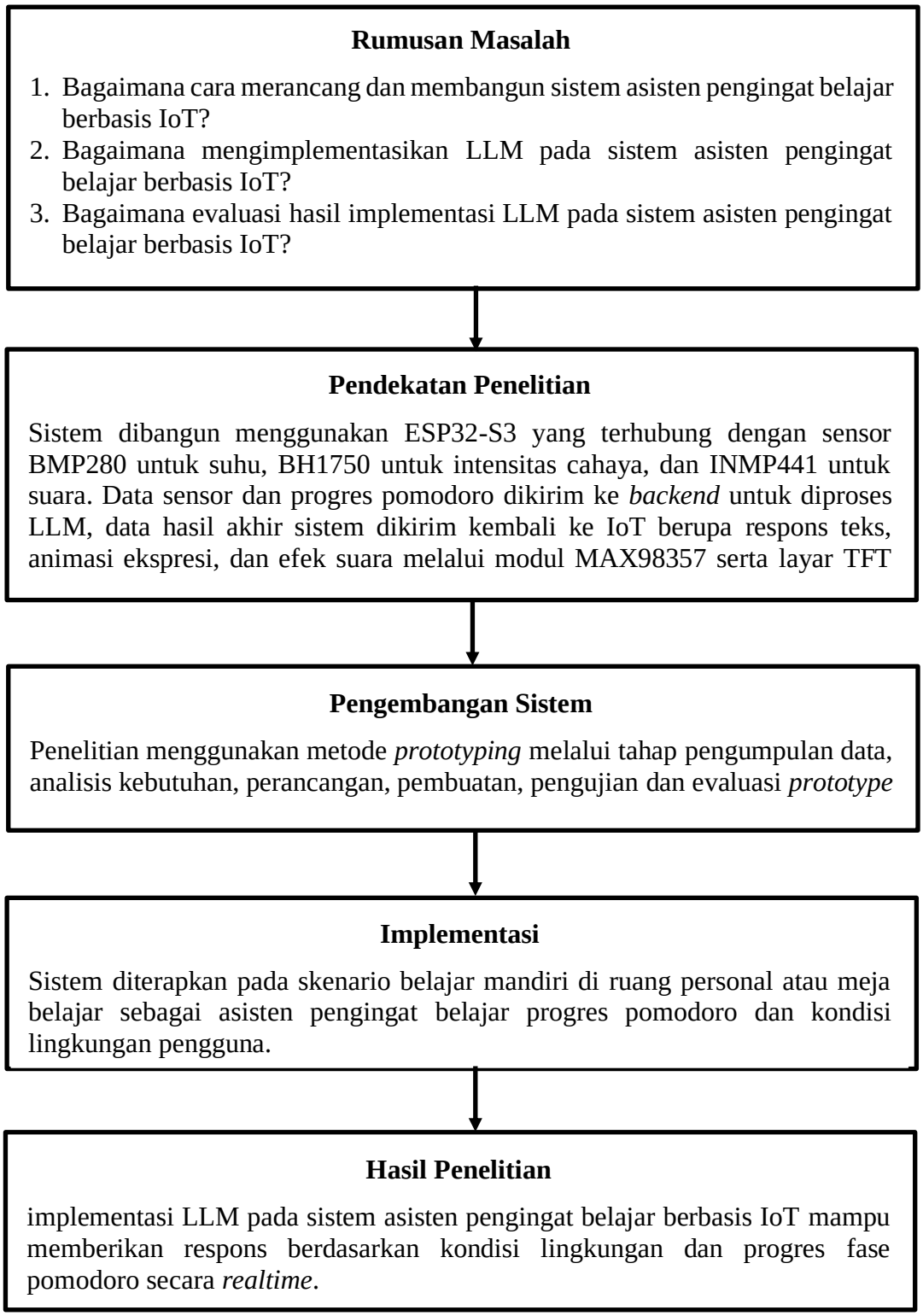
Penelitian ini hadir dengan mengimplementasikan LLM pada sistem asisten pengingat belajar berbasis IoT yang mampu memantau kondisi lingkungan belajar, progres belajar berdasarkan fase pomomodoro, sekaligus memberikan respons yang sesuai. Sistem memanfaatkan ESP32 dan sensor lingkungan untuk mendeteksi kondisi suhu, cahaya, serta suara pada ruangan, kemudian data tersebut dikirim menuju *server backend* untuk diproses oleh model LLM terkuantisasi. Respons sistem disampaikan melalui teks, animasi mimik wajah, dan efek suara pendukung agar mampu memberikan respons yang menarik selama *timer* pomodoro berjalan, tanpa menyebabkan distraksi tambahan akibat penggunaan *smartphone* atau *voice output* yang berlebihan.

C. Kerangka Berpikir

Permasalahan utama dalam penelitian ini adalah tingginya distraksi digital yang dapat menurunkan fokus belajar, terutama ketika proses belajar dilakukan secara mandiri di ruang personal. Perangkat digital memang memudahkan akses materi pembelajaran, tetapi juga berpotensi menjadi sumber gangguan melalui notifikasi, media sosial, dan aplikasi hiburan. Metode pomodoro banyak digunakan untuk membantu manajemen fokus melalui

pembagian waktu belajar dan istirahat. Namun, penerapan pomodoro pada *timer* konvensional masih cenderung statis. Oleh karena itu, penelitian ini mengembangkan asisten pengingat belajar terintegrasi IoT dan LLM yang menggunakan ESP32, sensor suhu, sensor cahaya, dan sensor suara untuk membaca kondisi lingkungan sebagai konteks respons sistem.

Hasil dari penelitian ini adalah berupa asisten pengingat belajar yang mampu memberikan respons *realtime* berupa teks, perubahan mimik wajah, dan efek suara pendukung sesuai kondisi lingkungan serta siklus pomodoro. Sistem ini diharapkan dapat menjadi asisten pengingat belajar berbasis IoT yang mampu memberikan respons secara lebih adaptif berdasarkan siklus pomodoro dan kondisi lingkungan dibandingkan penggunaan *smartphone* sebagai media utama interaksi. Kerangka berpikir penelitian ini ditunjukkan pada Gambar 2.9.



Gambar 2.9 Kerangka Berpikir