BAB II

LANDASAN TEORI

2.1 Tinjauan Pustaka

2.1.1 Gas Buang Kendaraan Bermotor

Emisi gas buang dari kendaraan bermotor mengandung berbagai polutan, seperti senyawa hidrokarbon (HC), karbon monoksida (CO), nitrogen oksida (NOx), senyawa sulfur (SOx), dan timbal (Pb), yang bercampur dengan partikel debu. Gas-gas ini bersifat beracun dan berdampak negatif terhadap kesehatan manusia. CO dan CO2 merupakan hasil utama dari pembakaran bensin yang tidak sempurna, sedangkan HC adalah hasil emisi dari bahan bakar yang tidak terbakar sepenuhnya dan keluar bersama sisa pembakaran (Susilo, 2023).

Menurut peraturan Menteri Negara Lingkungan Hidup Nomor 04 tahun 2019 tentang Ambang Batas Emisi Gas Buang Kendaraan Bermotor tipe baru, kendaraan bermotor beroda dua dengan kapasitas silinder lebih dari 50 cm^3 atau dengan desain kecepatan maksimum lebih dari 50 km/jam, apapun jenis tenaga penggeraknya, memiliki batas emisi untuk senyawa CO sebesar 5,5 gram/km, HC sebesar 1,0 gram/km, dan NOx sebesar 0,3 gram/km.

2.1.2 *Teknologi Internet of Things (IoT)*

2.1.2.1 Pengertian dan Manfaat

Internet of Things (*IoT*) merupakan gelombang ketiga dalam evolusi Internet yang mampu menghubungkan sekitar 28 miliar perangkat pada tahun 2020, mulai dari gelang pintar hingga mobil. Istilah "*IoT*" pertama kali diperkenalkan oleh

Kevin Ashton, seorang ahli teknologi asal Inggris, pada tahun 1999. Teknologi ini berpotensi besar dalam berbagai bidang, mulai dari menciptakan peluang produk baru, mengoptimalkan pabrik, hingga meningkatkan efisiensi kerja, yang pada akhirnya akan mendorong pertumbuhan pendapatan dan laba. *IoT* diharapkan dapat meningkatkan efisiensi energi, memungkinkan pemantauan jarak jauh dan pengendalian aset fisik, serta meningkatkan produktivitas melalui aplikasi seperti keamanan rumah dan pemantauan kesehatan pabrik. Saat ini, *IoT* telah digunakan di berbagai pasar, termasuk perawatan kesehatan, peralatan rumah tangga, konstruksi, ritel, manufaktur dan energi, mobilitas dan transportasi, logistik, serta komunikasi.

Perangkat semakin terdigitalisasi dan terhubung, membentuk jaringan antara mesin, manusia, dan Internet, yang mengarah pada penciptaan ekosistem baru yang mengarah pada peningkatan produktivitas, efisiensi energi yang lebih baik, dan keuntungan yang lebih tinggi. Sensor membantu mengenali keadaan sehingga dapat memprediksi kebutuhan manusia berdasarkan informasi yang dikumpulkan secara kontekstual. Perangkat pintar ini tidak hanya mengumpulkan informasi tentang lingkungan sekitar tetapi juga mampu mengambil keputusan tanpa campur tangan manusia. Teknologi *IoT* diterapkan dalam kehidupan kita sehari-hari seperti membuka pintu tanpa kunci; dalam pengenalan kartu, penguncian otomatis, sistem deteksi kendaraan, sistem pembayaran tol; dan untuk pelacakan hewan, kontrol akses, sistem pembayaran, kartu pintar nirsentuh, perangkat anti maling, pembaca kolom kemudi, dll. Komponen utama *IoT* akan berasal dari perangkat yang terhubung ke web, menyediakan platform umum di mana mereka dapat

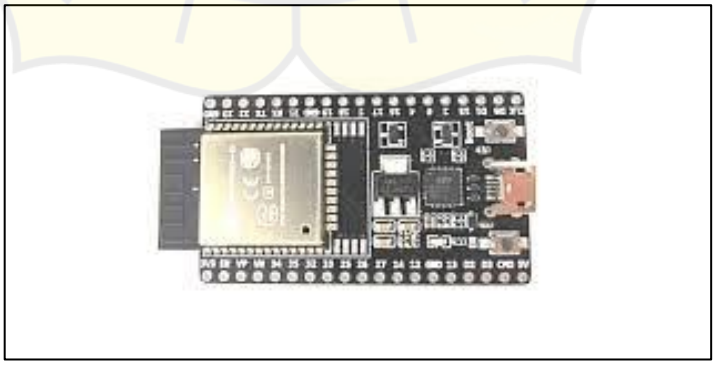
berkomunikasi dan mengembangkan aplikasi baru untuk menarik pengguna baru (Tripathy & Anuradha, 2018).

2.1.2.2 Microcontroller dan Sensor

2.1.2.2.1 ESP32 VROOM-32

ESP32 WROOM-32 merupakan modul MCU yang luar biasa kuat dan serbaguna, yang menyatukan Wi-Fi, Bluetooth, dan Bluetooth *Low Energy* (LE) dalam satu perangkat. Modul ini dirancang untuk mendukung berbagai aplikasi, mulai dari jaringan sensor dengan konsumsi daya rendah hingga tugas-tugas yang lebih berat, seperti *encoding* suara, *streaming* musik, dan *decoding* MP3.

Pada inti modul ini terdapat *chip* ESP32-D0WDQ6, yang dirancang untuk menjadi skalabel dan adaptif. *Chip* ini memiliki dua inti CPU yang dapat dikendalikan secara individu, dengan frekuensi *clock* CPU yang dapat diatur dari 80 MHz hingga 240 MHz. Selain itu, chip ini juga memiliki koprosesor berdaya rendah yang dapat digunakan sebagai pengganti CPU untuk menghemat daya saat melakukan tugas-tugas yang tidak memerlukan daya komputasi tinggi, seperti pemantauan *peripheral* (Espressif Systems, 2023).



Gambar 2. 1 ESP32 VROOM-32

2.1.2.2.2 MQ-135

Sensor MQ-135 memiliki fitur yang mencakup area deteksi yang luas, tanggapan cepat dengan sensitivitas tinggi, kestabilan yang baik dan masa pakai yang panjang, serta rangkaian penggerak yang simpel. Sensor ini diaplikasikan dalam perangkat kontrol kualitas udara di bangunan atau kantor, dan sangat ideal untuk mendeteksi berbagai zat seperti NH3, NOx, alkohol, benzena, asap, CO2, dan lainnya. (HANWEI ELECTRONICS, n.d.-b).

Gambar 2. 2 MQ-135

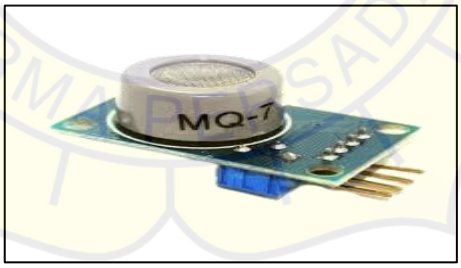
2.1.2.2.3 MQ-9

Sensor MQ-9 memiliki fitur yang mencakup sensitivitas tinggi terhadap karbon monoksida dan CH4, serta LPG yang stabil dan memiliki umur yang panjang. Penerapannya adalah dalam peralatan deteksi gas untuk karbon monoksida dan CH4, serta LPG di rumah, industri, atau kendaraan. Nilai resistansi MQ-9 berbeda-beda tergantung pada jenis dan konsentrasi gas yang berbeda. Oleh karena itu, saat menggunakan komponen ini, sangat penting untuk melakukan penyesuaian sensitivitas(HANWEI ELETRONICS, n.d.).

Gambar 2. 3 MQ-9

2.1.2.2.4MQ-7

Sensor MQ-7 memiliki fitur seperti *sensitivitas* tinggi terhadap karbon monoksida dan masa pakai yang panjang serta stabil. Sensor ini biasanya digunakan dalam perangkat deteksi gas untuk karbon monoksida (CO) di rumah, industri, atau kendaraan. Lapisan sensitif sensor MQ-7 terbuat dari SnO2 yang stabil, memberikan stabilitas jangka panjang yang luar biasa. Umur pakainya bisa mencapai 5 tahun dalam kondisi penggunaan yang tepat. Pengaturan sensitivitas sangat penting karena resistansi MQ-7 bervariasi tergantung pada jenis dan konsentrasi gas yang terdeteksi (HANWEI ELECTRONICS, n.d.-a).

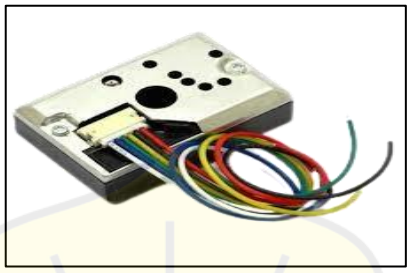


Gambar 2. 4 MQ-7

2.1.2.2.5Sharp GP2Y1010AU0F

GP2Y1010AU0F adalah sensor debu yang menggunakan sistem deteksi optik. Di dalam perangkat ini, sebuah diode pemancar inframerah (IRED) dan sebuah phototransistor diatur secara diagonal. Sensor ini mendeteksi cahaya yang

dipantulkan oleh partikel debu di udara. Sensor ini sangat efektif untuk mendeteksi partikel-partikel halus seperti asap rokok. Selain itu, sensor ini mampu membedakan antara asap dan debu rumah berdasarkan pola pulsa tegangan yang dihasilkan.n (SHARP, 2016).



Gambar 2. 5 GP2Y1010AU0F

2.1.3 *Chatbot*

2.1.3.1 Pengertian Chatbot

Chatbot didefinisikan sebagai program komputer yang mengolah input bahasa alami dari pengguna untuk menghasilkan tanggapan yang cerdas dan relevan, yang kemudian dikirim kembali ke pengguna. Saat ini, chatbot yang didukung oleh mesin berbasis aturan atau kecerdasan buatan (*AI*) berinteraksi dengan pengguna terutama melalui antarmuka teks. Chatbot ini adalah program mandiri yang dapat diintegrasikan dengan berbagai platform pesan yang tersedia untuk pengembang melalui API, seperti Facebook Messenger, Slack, Skype, Microsoft Teams, dan lainnya (Khan & Das, 2018).

2.1.3.2 ChatGPT dan NLP

ChatGPT, sebuah model bahasa kecerdasan buatan (AI) canggih yang dikembangkan oleh OpenAI, telah menjadi aplikasi konsumen tercepat dalam sejarah yang menjangkau 100 juta pengguna hanya dalam 2 bulan. Popularitasnya

dapat dikaitkan dengan kemampuannya untuk mendemokratisasi model *Natural Language Processing* (*NLP*) untuk pengguna sehari-hari. *NLP,* atau pemrosesan bahasa alami, adalah cabang dari AI yang berfokus pada memungkinkan komputer untuk berinteraksi dengan manusia menggunakan bahasa sehari-hari. Tujuan utama *NLP* adalah memungkinkan komputer menafsirkan, memahami, dan merespons bahasa manusia dengan cara yang bermakna dan berguna. Sebelumnya, tugas-tugas di bidang ini – mulai dari analisis sentimen hingga penerjemahan bahasa – memerlukan kumpulan data yang kuat dan keahlian dalam pembelajaran mesin dan ilmu data agar dapat dilaksanakan dengan sukses.

Namun, kemunculan *ChatGPT* dan *API* terkaitnya telah merevolusi lanskap *NLP*. Dengan demokratisasi model *NLP*, siapa pun, termasuk pengguna biasa, kini dapat menghasilkan teks mirip manusia dari perintah tanpa memerlukan pengetahuan ilmu data atau pembelajaran mesin yang luas.

Kemunculan *ChatGPT* telah membuat tugas – tugas *NLP* yang sebelumnya rumit menjadi lebih mudah diakses dan ramah pengguna, menjembatani kesenjangan antara teknologi mutakhir dan masyarakat umum. Pemrogram dan pengembang memperhatikan dan mengintegrasikan kekuatan *GPT* ke dalam aplikasi mereka sendiri untuk menjadikan mereka lebih pintar. Faktanya, banyak startup yang memiliki pendanaan besar (*Typeface, Jasper AI, Copy.ai)* menggunakan *ChatGPT* dan model bahasa besar (*LLM*) lainnya sebagai landasan produk mereka, baik itu Meringkas teks, mencari informasi, atau membuat *chatbot* (Habib et al., 2024).

2.1.3.3 ChatGPT API

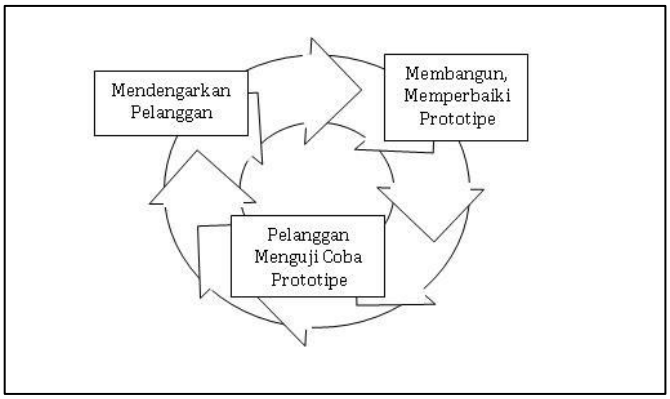
Awalnya, sebagian besar model GPT menangani teks tidak terstruktur, yang disajikan kepada model sebagai *string* “token”. Namun, model ChatGPT bekerja secara berbeda, menggunakan *string* pesan serta *metadata*. Meskipun masukan disajikan ke model sebagai *string* "token" untuk diproses, format mentah yang digunakan adalah apa yang disebut *OpenAI* sebagai *Chat Markup Language* (*ChatML*). OpenAI menggunakan sistem bayar sesuai pemakaian untuk 1.000 token. Jumlah total token yang diproses dalam *query* (*prompt* dan hasil) tidak boleh melebihi panjang konteks maksimum model. Untuk sebagian besar model, jumlahnya adalah 4.096 token atau sekitar 3.000 kata (1 token setara dengan 4 karakter atau 0,75 kata untuk teks bahasa Inggris) (Habib et al., 2024).

2.1.4 Langkah Pengembangan Metode *Prototype*

2.1.4.1 Pengertian Metode Prototype

Prototyping adalah proses merancang sebuah prototipe, yaitu model dari produk yang mungkin belum memiliki semua fitur produk akhir, tetapi sudah mengandung fitur utama. Prototipe ini biasanya digunakan untuk pengujian material sebelum memasuki tahap produksi sesungguhnya. Melalui metode ini, pengembang dan pelanggan dapat berkolaborasi selama proses pembuatan produk.

Prototyping perangkat lunak merupakan salah satu metode siklus hidup sistem yang didasarkan pada konsep pemodelan fungsional. Tujuannya adalah mengembangkan model menjadi sistem *final* dengan lebih cepat dan biaya lebih rendah dibandingkan metode tradisional. Proses ini mencakup cara pembuatan dan penggunaan prototipe.



Gambar 2. 6 Ilustrasi Model Prototype

Berikut adalah tahapan pengembangan prototipe:

1. **Mendengarkan Pelanggan**: Tahap ini melibatkan pengumpulan kebutuhan sistem dengan mendengarkan keluhan dan masukan dari pelanggan. Memahami cara kerja sistem saat ini dan masalah yang ada merupakan langkah awal untuk merancang solusi yang sesuai.
2. **Perancangan dan Pembuatan Prototipe**: Berdasarkan kebutuhan yang telah diidentifikasi, tim melakukan perancangan dan pembuatan prototipe. Prototipe ini disesuaikan dengan kebutuhan sistem yang telah diidentifikasi sebelumnya.
3. **Pengujian**: Pada tahap ini, prototipe diuji oleh pelanggan atau pengguna. Penilaian dilakukan untuk mengidentifikasi kesenjangan antara kebutuhan pelanggan dan prototipe yang ada. Pengembang kemudian mendengarkan umpan balik pelanggan untuk menyempurnakan prototipe tersebut. (Hasanah & Untari, 2020).

2.1.4.2 Keunggulan dan Kekurangan Model Prototipe

Keunggulan model prototipe terletak pada komunikasi yang efektif antara pengembang dan pelanggan. Dengan pendekatan ini, pengembang dapat lebih baik dalam mengidentifikasi kebutuhan pelanggan, menghemat waktu pengembangan sistem, dan memudahkan penerapan karena pengguna sudah memahami apa yang diharapkan. Pelibatan pelanggan dalam proses pengembangan sistem membantu pengembang mengetahui produk yang diinginkan pelanggan.

Namun, model prototipe juga memiliki beberapa kekurangan. Risiko tinggi dapat muncul karena masalah yang tidak terstruktur dengan baik, perubahan signifikan dari waktu ke waktu, dan ketidakpastian kebutuhan data. Keterlibatan pengguna sangat penting, dan sistem harus menyediakan dialog online antara klien dan komputer. Hubungan klien dengan komputer mungkin tidak selalu mencerminkan teknik desain yang baik dan kurang fleksibel terhadap perubahan. Meskipun pengguna melihat peningkatan pada setiap versi prototipe, mereka mungkin tidak menyadari bahwa versi tersebut dibuat tanpa memperhatikan kualitas dan pemeliharaan jangka panjang. (Hasanah & Untari, 2020).

2.1.5 Pemodelan Sistem UML

2.1.5.1 Pengertian dan Sejarah Singkat UML

UML merupakan hasil kerja sama kolektif dari banyak praktisi, ahli metodologi, pemikir, dan penulis. OMG memfasilitasi kontribusi-kontribusi ini dan menyatukannya menjadi sebuah metamodel yang kuat, menghasilkan notasi pemodelan standar industri yang dikenal sebagai UML. UML pertama kali diperkenalkan sekitar tahun 1995 sebagai kombinasi dari tiga metode populer saat itu: Booch, Object Modeling Technique, dan Objectory. Seiring waktu, beberapa

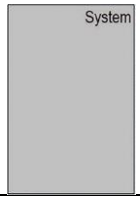
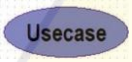
metode lain ditambahkan ke dalam UML, yang akhirnya menghasilkan versi UML 1.4 yang populer. Pada sekitar tahun 2004, UML 2.0 dirilis dengan formalitas yang lebih tinggi, mencakup 13 diagram formal dan perubahan terkait pada metamodel, yang mendukung inisiatif seperti Model-Driven Architecture (MDA). Satu dekade kemudian, UML 2.5, yang mencakup 14 diagram, dianggap sebagai bahasa pemodelan standar untuk Rekayasa Perangkat Lunak (*SE*). (Unhelkar, 2018).

2.1.5.2 *Use case* Diagram

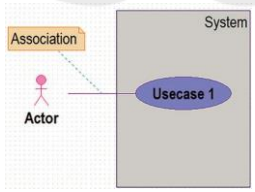
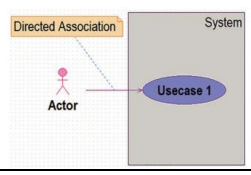
Use case diagram adalah model kebutuhan sistem pada tingkat tinggi yang digunakan untuk memvisualisasikan kasus penggunaan, industri terkait, dan interaksinya. Diagram ini bukanlah kasus penggunaan itu sendiri, melainkan representasi *visual* dari seorang aktor dan sekumpulan *use case* terkait. Model *visual* dari *use case* membantu dalam memahami proses bisnis dan memfasilitasi komunikasi dengan pemangku kepentingan. Menentukan dan mendokumentasikan *use case* yang ditunjukkan dalam *use case* diagram adalah inti dari pemodelan persyaratan.

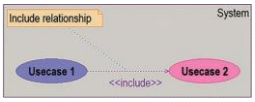
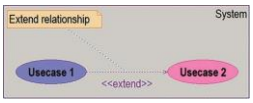
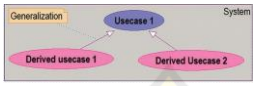
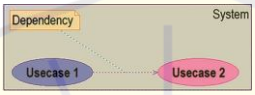
Use case diagram memiliki perilaku statis karena mereka membantu mengatur dan mengevaluasi persyaratan sistem dalam ruang masalah. Aspek perilaku dari persyaratan tidak ditampilkan dalam *use case* diagram. Karena hubungan antara dua *use case* atau antara aktor dan *use case* tidak mencerminkan konsep waktu, diagram use case digolongkan sebagai diagram statis. Jadi, penting untuk diingat bahwa *use case* diagram menggambarkan keseluruhan aliran atau perilaku sistem. (Unhelkar, 2018).

Tabel 2. 1 komponen Use Case Diagram

No	Nama komponen	*UML Notation*	Tujuan
1	*System Boundary*		• Mewakili ruang lingkup sistem. • Merangkum rangkaian lengkap fungsi sistem.
2	*Actors*		• Pengguna yang berinteraksi dengan sebuah sistem. • Aktor bisa berupa individu, organisasi, atau sistem eksternal yang berinteraksi dengan aplikasi atau sistem Anda. • Aktor dalam usecase diagram berinteraksi dengan usecase. • Pengembang bertanggung jawab untuk mempertimbangkan aktor mana yang berdampak pada fungsionalitas yang mereka modelkan.
3	*Usecases*		• Representasi visual dari berbagai fungsionalitas bisnis dalam sistem. • Memastikan bahwa proses bisnis bersifat diskrit. • Buatlah daftar fungsi bisnis diskrit dalam pernyataan masalah yang diberikan. - Mengidentifikasi usecase adalah sebuah penemuan.

Tabel 2. 2 Use Case Relationships

No	*Use case Relationship*	*UML Notation*	Tujuan
1	*Association*		Asosiasi merepresentasikan hubungan antara aktor dan usecase.
2	*Directed Association*		*Directed Association* mewakili hubungan satu arah di mana aktor bertanggung jawab dalam menjalankan sebuah *use case.*

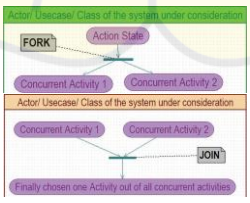
3	*Include*		Merepresentasikan situasi di mana satu *use case* menyertakan fungsionalitas dari *use case* lainnya.
4	*Extend*		Mengimplikasikan hubungan di mana *use case* yang diperluas menambahkan perilaku tambahan pada fungsionalitas yang ada dalam *use case* dasar.
5	*Generalization*		Merupakan hubungan antara use case induk dan satu atau lebih *use case* anak.
6	*Dependency*		Mendefinisikan hubungan di mana keberadaan satu use case bergantung pada keberadaan *use case* lainnya.

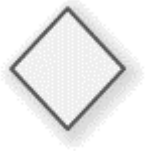
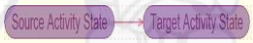
2.1.5.3 Activity Diagram

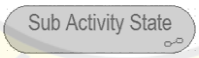
Activity Diagram memodelkan aliran atau proses dalam suatu sistem, mirip dengan diagram alur. Pemodelan aliran ini dapat diterapkan pada tingkat proses bisnis, dalam suatu use case, dan kadang-kadang di antara *use case*. Aktivitas yang ditampilkan dalam diagram dapat berada pada tingkat teknis yang terperinci atau pada tingkat bisnis yang lebih umum. Diagram aktivitas menangkap perilaku internal dalam sebuah *use case*, di antara *use case*, atau di seluruh organisasi. Diagram aktivitas tingkat tinggi digunakan sebagai diagram konteks yang menunjukkan keterkaitan antara berbagai proses bisnis. (Unhelkar, 2018).

Tabel 2. 3 Komponen Activity Diagram

No	Nama komponen	*UML Notation*	Tujuan
1	*Initial state*		Mewakili kondisi awal dari sistem yang sedang dipertimbangkan
2	*Final state*		Mewakili status penghentian sistem yang sedang dipertimbangkan.

3	*Swimlanes*		• Berisi dua partisi, di mana partisi teratas mewakili entitas seperti aktor, *use case*, kelas, dan lainnya, sementara partisi kedua berfokus pada kumpulan aktivitas yang terlibat. • Swimlane terdiri dari dua jenis yaitu swimlane vertikal dan swimlane horizontal. • Vertical swimlane - mewakili aktivitas paralel dari sebuah skenario tertentu dari sistem yang sedang dipertimbangkan. • Horizontal swimlane - mewakili aktivitas berurutan dari skenario tertentu dari sistem yang sedang dipertimbangkan.
4	*Action state*		Status tindakan mewakili operasi atau aktivitas bisnis atau proses.
5	*Object*		Entitas yang membawa data di antara dua status tindakan.
6	*Synchronization*		Sinkronisasi mewakili dua atau lebih aktivitas yang terjadi pada waktu atau kecepatan yang sama. Ini terdiri dari dua jenis yaitu: • Fork - Membagi aliran aktivitas tunggal menjadi dua atau lebih aktivitas yang berlangsung secara bersamaan. • Join - Menggabungkan dua atau lebih aktivitas bersamaan menjadi satu aliran dengan memastikan hanya satu aktivitas terjadi pada satu waktu.

7	*Decision*		• Memiliki satu input dan dua atau lebih output tergantung pada kondisi yang ditentukan. • Setiap aliran keluaran memiliki kondisi yang melekat padanya. • Jika suatu kondisi terpenuhi, aliran akan berlanjut bersama dengan keluaran yang sesuai. • Output 'lain' dapat didefinisikan di mana aliran dapat dilanjutkan jika tidak ada kondisi lain yang terpenuhi.
8	*Flow Final*		Mewakili penghentian abnormal dari sebuah jalur dalam diagram aktivitas yang tidak dianggap sebagai bagian dari sistem yang sedang dikembangkan.
9	*Transition*		Panah yang merepresentasikan pergerakan dari status aktivitas sumber ke status aktivitas target, yang dipicu oleh selesainya aktivitas pada status aktivitas sumber.
10	*Self-Transition*		Mewakili transisi internal ke status tindakan itu sendiri.
11	*Signal Send State*		• Sinyal menunjukkan bagaimana aktivitas dapat dipengaruhi secara eksternal oleh sistem. • Mereka biasanya muncul berpasangan dengan sinyal yang dikirim dan diterima. Status Kirim Sinyal mewakili tindakan pengiriman sinyal di luar dari aktivitas. • Status pengiriman sinyal tidak menunggu tanggapan apa pun dari penerima sinyal.

			• Ia akan berakhir dengan sendirinya dan meneruskan kontrol eksekusi ke tindakan berikutnya. • Status Kirim Sinyal memiliki notasi sebagai segi lima cembung.
12	*Signal Accept State*		• Status tindakan yang pemicunya adalah peristiwa sinyal secara informal disebut Status Terima Sinyal. • Hal ini berhubungan dengan Signal Send State. • Signal Accept State dengan sisi masuk berarti aksi dimulai setelah status aksi sebelumnya selesai. • Signal Accept State tanpa sisi masuk tetap diaktifkan setelah menerima peristiwa. • Ia tidak berhenti setelah menerima sebuah event dan mengeluarkan sebuah nilai, tetapi terus menunggu event lainnya.
13	*Subactivity*		Subaktivitas adalah status tindakan yang dapat menjadi skenario aktivitas utama lainnya di masa mendatang.

2.1.6 *Software* dan Pemrograman Terkait

2.1.6.1 Database

2.1.6.1.1 Pengertian Database dan DBMS

Manajemen data yang efektif seringkali memerlukan penggunaan database komputer. Basis data adalah struktur komputasi yang terintegrasi dan dibagikan untuk menyimpan kumpulan berikut:

- Data pengguna akhir: Fakta mentah yang menarik bagi pengguna akhir.
- Metadata: Data tentang data, yang mengintegrasikan dan mengelola data pengguna akhir.

Metadata menggambarkan karakteristik data dan hubungan yang menghubungkan data dalam basis data. Misalnya, metadata menyimpan informasi seperti nama setiap item data, jenis nilai (angka, tanggal, atau teks) yang disimpan, dan apakah item data tersebut dapat kosong. Metadata memberikan informasi tambahan yang memperluas nilai dan penggunaan data, memberikan gambaran lebih lengkap tentang data dalam basis data. Karena karakteristik ini, database sering digambarkan sebagai "kumpulan data eksplisit."

Database Management System (DBMS) adalah sekumpulan program yang mengelola struktur basis data dan mengontrol akses ke data yang tersimpan. Secara semantik, basis data seperti lemari arsip elektronik yang sangat terorganisir, dengan perangkat lunak kuat (*DBMS*) yang membantu mengelola isi lemari arsip tersebut. (Coronel & Morris, 2019).

2.1.6.1.2 *NoSQL* Database

NoSQL adalah *unfortunate nam*e yang diberikan pada serangkaian teknologi database non-relasional yang dikembangkan untuk mengatasi tantangan yang ditimbulkan oleh *Big Data*. Nama tersebut sangat disayangkan karena tidak menggambarkan apa sebenarnya teknologi *NoSQL*, melainkan tidak menggambarkan apa itu teknologi *NoSQL*. Bahkan, dari namanya juga tidak menjelaskan secara jelas apa sebenarnya teknologi itu. Nama ini dipilih sebagai hashtag Twitter untuk memudahkan koordinasi pertemuan pengembang guna

mendiskusikan ide tentang teknologi database non-relasional yang sedang dikembangkan oleh organisasi seperti Google, Amazon, dan Facebook untuk memecahkan masalah yang mereka hadapi ketika kumpulan data mereka mencapai skala yang sangat besar.

Istilah *NoSQL* tidak pernah dimaksudkan untuk menyiratkan bahwa produk dalam kategori ini tidak mendukung *SQL*. Faktanya, banyak produk semacam itu mendukung bahasa kueri yang meniru *SQL* secara signifikan. Meskipun belum ada yang membuat sistem *NoSQL* yang mengimplementasikan *SQL* standar, mengingat banyaknya pengguna *SQL*, manfaat membuat produk semacam itu jelas. Baru-baru ini, beberapa pengamat industri mencoba menjelaskan bahwa *NoSQL* bisa berarti "bukan hanya *SQL*" (Coronel & Morris, 2019).

2.1.6.1.3 Firebase

Platform Firebase membantu *developer* mengembangkan aplikasi berkualitas tinggi yang berfokus pada pengguna. Firebase adalah platform pengembangan aplikasi web dan *seluler* yang ditawarkan oleh Google. Ini adalah solusi lengkap untuk semua kebutuhan pengembangan aplikasi web dan *seluler* berkualitas tinggi Anda. Firebase menyediakan berbagai produk canggih untuk pengembangan dan pengujian aplikasi secara real-time dengan fokus pada pengalaman pengguna. Produk-produk ini mencakup Realtime Database, Crash Reporting, Cloud Firestore, Cloud Storage, Cloud Functions, Authentication, Storage, Android Test Lab, dan Performance Monitoring untuk iOS. Dengan Firebase, pengembang dapat dengan mudah mengelola data, melaporkan kerusakan, mengotentikasi pengguna, dan memantau kinerja aplikasi mereka secara efektif. (Singh & Tanna, 2018).

2.1.6.1.4 *Realtime* Database

Untuk aplikasi *real-time*, kita memerlukan database *real-time*. Firebase Realtime Database adalah solusi NoSQL berbasis cloud yang menawarkan sinkronisasi data secara langsung dengan semua klien yang terhubung. Database ini memastikan setiap perubahan data segera diperbarui di semua perangkat secara real-time. Berbeda dengan model permintaan-respons biasa, database Firebase menggunakan mekanisme yang menyinkronkan data di semua perangkat yang terhubung dalam hitungan milidetik. Salah satu fitur utama lainnya adalah fungsionalitas *offline*. Firebase SDK menyimpan data pada *disk*; Jadi meskipun pengguna kehilangan koneksi internet, aplikasi tetap responsif. Secara otomatis menyinkronkan data setelah koneksi tersambung kembali. Didukung oleh platform iOS, Android, Web, C++ dan Unity (Singh & Tanna, 2018).

2.1.6.2 Android

Android adalah sistem operasi seluler *open source* yang berbasis Linux, menawarkan keamanan, modularitas, dan produktivitas tinggi pada perangkat seluler. Dikembangkan dan dikelola oleh *Open Handset Alliance (OHA)*, yang didirikan pada tahun 2007 dengan Google sebagai anggota inti, aliansi ini terdiri dari banyak perusahaan perangkat keras dan perangkat lunak terkemuka. Android awalnya dikembangkan oleh Android Inc., sebuah perusahaan yang diakuisisi oleh Google pada tahun 2005. Setelah diakuisisi, Google menjadikan Android sebagai open source, yang kemudian mendapatkan popularitas besar. Pada tahun 2016, Android menguasai sekitar 85% pangsa pasar (Cardle, 2017).

2.1.6.3 Kotlin

Kotlin adalah bahasa pemrograman berbasis JVM yang dikembangkan oleh JetBrains, perusahaan di balik IntelliJ IDEA, IDE yang terkenal dan kuat untuk pengembangan Java. Android Studio, IDE Android resmi, berdasarkan IntelliJ. Kotlin dibuat dengan pengembang Java sebagai target utama dan dengan IntelliJ sebagai IDE pengembangan utama. Ada dua fitur hebat untuk pengembang Android:

1. Kotlin sangat intuitif dan mudah dipahami oleh pengembang Java. Sebagian besar sintaksnya mirip dengan Java, sehingga pengembang yang sudah berpengalaman dengan Java dapat mempelajari konsep-konsep dasarnya dengan cepat.
2. Integrasi sepenuhnya gratis dengan IDE Harian. Android Studio dapat memahami, mengkompilasi, dan menjalankan kode Kotlin. Dukungan untuk bahasa ini berasal dari perusahaan pengembang IDE (Leiva, 2015).

2.1.6.4 Nodejs

Node.js adalah *runtime* JavaScript yang dibangun di atas mesin JavaScript, berfungsi sebagai kerangka kerja *open-source* untuk manajemen sisi server. Node.js dikenal karena ringan, efisien, dan *kompatibel* dengan berbagai *platform* seperti Windows, Linux, dan macOS. Dikembangkan oleh Ryan Dahl pada tahun 2009, Node.js memungkinkan penggunaan JavaScript untuk skrip sisi server, yang sebelumnya hanya digunakan untuk skrip sisi klien. Inovasi ini memperkenalkan konsep penggunaan satu bahasa pemrograman untuk keseluruhan aplikasi web. Node.js menawarkan berbagai keuntungan, di antaranya adalah:

1. Pemrograman berbasis peristiwa: Node.js menggunakan pemrograman berbasis peristiwa, artinya menggunakan tindakan interaktif pengguna, seperti klik mouse dan penekanan tombol, untuk mengubah keadaan objek.
2. I/O *non-blocking*: I/O *non-blocking* atau I/O *non-synchronous*, berarti I/O *asynchronous*. Proses *asynchronous* menunggu proses yang sedang berjalan selesai dan memblokirnya. Di sisi lain, proses *asynchronous* tidak perlu menunggu hingga proses selesai, sehingga membuatnya cepat dan andal.
3. *Single threading*: *Single threading* berarti JavaScript dijalankan dalam satu *event loop*. Meskipun proses asinkron memungkinkan kita menjalankan beberapa proses secara bersamaan, tampaknya semua proses ini berjalan di thread khusus masing-masing. Namun, Node.js menangani *async* sedikit berbeda. Perulangan peristiwa di Node.js mengaktifkan fungsi panggilan balik berikutnya yang dijadwalkan untuk dijalankan ketika peristiwa terkait terjadi (Sharma, 2018).

2.1.6.5 Arduino IDE

Pada tahun 2003, seorang mahasiswa bernama Hernando Barragan menulis tesis tentang perangkat keras yang menjelaskan IDE dan integrasinya dengan papan sirkuit dan mikrokontroler. Dengan masukan dari peneliti lain, konsep ini berkembang untuk memungkinkan pengembang menulis beberapa baris kode saja untuk mereproduksi koneksi sederhana komponen perangkat keras. Hal ini memungkinkan interaksi seperti menyalakan LED, menerima input dari tombol, memutar suara, dll.

Jadi meskipun bahasa yang digunakan dalam IDE adalah C/C++, namun IDE dapat menyediakan *API* yang memungkinkan pengkabelan dan kode menjadi

sangat sederhana dan didasarkan pada bagaimana komponen elektronik dihubungkan ke rangkaian (Ramos, 2014).

2.1.6.6 Android Studio

Pada tahun 2018, Google memperkenalkan Android Jetpack ke komunitas pengembang. Dirancang untuk membuat pengembangan aplikasi Android yang modern dan andal menjadi lebih cepat dan mudah, Jetpack menyertakan serangkaian alat, pustaka, dan prinsip arsitektur. Komponen utama Android Jetpack mencakup Lingkungan Pengembangan Terintegrasi (IDE) Android Studio, Komponen Arsitektur Android, dan Pedoman Arsitektur Aplikasi Modern, semuanya disertakan dalam versi *Things* terbaru ini (Smyth, 2019).

2.1.6.7 Visual Studio Code

Memilih *editor* mana yang digunakan *developer* adalah masalah yang sangat pribadi. Alasan memilih satu *editor* dibandingkan editor lainnya bergantung pada serangkaian *atribut* yang sering kali terkait dengan tugas yang mereka lakukan sehari-hari. *developer* mencari fitur, pintasan, cuplikan kode, warna, dan banyak lagi untuk membantu mereka bekerja secara efektif.

Mengubah preferensi *editor developer* tidaklah mudah. Setiap perubahan pada *editor* akan menyebabkan penurunan produktivitas secara langsung. Terakhir, perlu waktu untuk membiasakan diri dengan fitur-fitur yang disediakan dan mengintegrasikannya secara alami ke dalam “aliran” pemrograman. Oleh karena itu, pengembang memerlukan tingkat “kebaikan” khusus untuk mengubah editor.

Oleh karena itu, keberhasilan Visual Studio Code sebagian besar tercermin dalam fitur dan fungsinya. Meskipun baru dirilis secara resmi selama tiga tahun

(pratinjau publik dimulai pada bulan April 2016), editor ini dengan cepat menjadi salah satu *editor* teratas dalam hal popularitas, setara dengan Sublime Text, Atom, dan UltraEdit di posisi pertama (Johnson, 2019).

2.2 Kajian Penelitian Terdahulu

Tabel 2. 4 Penelitian Terkait

No	Judul	Author	Tahun	Klasifikasi
1	ALAT UJI EMISI PORTABEL KENDARAAN BERMOTOR	Pandu Aldhareva, Risfendra	2020	JTEV, Volume 6 Number 1 (2020), ISSN: 2302-3309
Hasil: Pengukuran gas CO menggunakan alat uji emisi portabel disimpan di kartu memori yang mencatat nomor registrasi serta tanggal dan waktu pengukuran. Kadar gas CO kendaraan lebih rendah jika menggunakan sistem mesin injeksi dengan bahan bakar seperti pertamax atau pertalite. Sebaliknya, kadar gas CO akan lebih tinggi jika menggunakan sistem mesin karburator dengan bahan bakar premium (Aldhareva & Risfendra, 2020).				
2	IMPLEMENTASI ARDUINO DALAM RANCANG BANGUN ALAT UJI EMISI KENDARAAN BERMOTOR BERBASIS ANDROID	Ikhsan, Irwanto	2017	JURNAL ILMU KOMPUTER, Vol. 6, No. 1, April 2017, E-ISSN: 2579 - 3918
Hasil: Alat uji emisi gas buang yang dikembangkan mampu mengukur kadar emisi kendaraan 2-tak dan 4-tak, sesuai dengan standar pengukuran yang ditetapkan dalam Peraturan Menteri Negara Lingkungan Hidup Nomor 05 Tahun 2006. Secara keseluruhan, alat ini telah terbukti berfungsi dengan baik dan sesuai harapan. Keunggulannya adalah alat ini dapat digunakan secara portabel dan				

hasil pengukurannya dapat ditampilkan pada perangkat Android (Ikhsan & Irwanto, 2017).				
3	RANCANG BANGUN ALAT UKUR UJI EMISI KENDARAAN GAS KARBON MONOKSIDA (CO), KARBONDIOKSIDA (CO2), DAN HINDROKARBON (HC) BERBASIS IOT	Angelia Maharani Purba, Efandsah Perdana Siregar	2023	TrekRiTel, Volume 3, Nomor 1, April 2023, ISSN 2776 - 5946
Hasil: Hasil pengukuran ditampilkan dalam satuan ppm pada aplikasi Telegram melalui konektivitas IoT. Rancang bangun alat ini diharapkan dapat berkontribusi dalam upaya mengurangi polusi udara dan meningkatkan kualitas udara serta lingkungan secara keseluruhan (Purba & Siregar, 2023).				
4	RANCANG BANGUN ALAT UKUR UJI EMISI GAS KARBON MONOKSIDA (CO) BERBASIS MIKROKONTROLLER	Semuel Kete Sarungallo, I Gusti Putu Raka Agung, Lie Jasa	2017	Teknologi Elektro, Vol. 16, No1, Januari-April 2017, p-ISSN:1693 – 2951; e-ISSN: 2503-2372
Hasil: Alat ukur emisi gas karbon monoksida yang dirancang menggunakan Arduino Uno R3 dan sensor MQ-7 mampu mendeteksi kadar gas CO dalam rentang 20-2000 ppm. Pengujian menunjukkan bahwa alat ini memiliki margin kesalahan sebesar 2,7%, yang dianggap dapat diterima. Oleh karena itu, dapat disimpulkan bahwa alat ini berfungsi dengan baik dan efektif dalam mengukur kadar gas CO, yang merupakan gas berbahaya bagi kesehatan dan lingkungan, terutama yang dihasilkan oleh kendaraan bermotor (Sarungallo et al., 2017).				

5	RANCANG BANGUN ALAT UJI EMISI GAS BUANG KENDARAAN BERBASIS ARDUINO AT MEGA 2560	F.A. Elhaq, S. Supriyadi dan A. Burhanudin	2020	SENS 5, Vol. 5 No. 1, 17 Desember 2020, ISBN: 978-623-6602-31-7
Hasil: Alat uji emisi yang berbasis Arduino AT Mega 2560 dan dilengkapi dengan sensor MQ-135, MQ-2, dan MQ-7 mampu mendeteksi gas CO2, O2, HC, dan CO pada emisi gas buang kendaraan. Pengujian menunjukkan bahwa alat ini dapat mengukur gas emisi dengan akurat, meskipun terdapat beberapa fluktuasi pada beberapa sensor. Penelitian ini menyimpulkan bahwa alat uji emisi ini efektif untuk mengukur kadar gas emisi kendaraan bermotor dengan biaya yang lebih ekonomis (Elhaq et al., 2020).				