

BAB II

TINJAUAN PUSTAKA

2.1. Teori Terkait

Penelitian mengenai sistem keamanan berbasis *Internet of Things* (IoT) untuk mitigasi kebocoran gas telah banyak dikembangkan dengan berbagai pendekatan platform. Penelitian oleh Sari et al. (2026) merancang *smart detector* kebocoran gas menggunakan NodeMCU dan sensor MQ-2 yang terintegrasi dengan *Firebase* dan Android Studio, di mana sistem ini memberikan peringatan melalui suara lokal melalui DFPlayer Mini dan notifikasi Android[6]. Selanjutnya, Lestari et al. (2025) mengimplementasikan sistem *monitoring* berbasis ESP32 dengan protokol MQTT yang mengirimkan notifikasi via Telegram, dengan waktu respons rata-rata 3,1 detik dan tingkat keberhasilan deteksi sebesar 98%[1]. Penelitian lain oleh Mulwinda et al. (2026) melakukan pengujian kinerja pada kondisi lingkungan berbeda menggunakan platform Blynk, yang menunjukkan bahwa waktu respons di ruang tertutup ($\pm 3,8$ detik) lebih stabil dibandingkan ruang terbuka ($\pm 5,8$ detik[24]). Selain itu, Putra (2024) mengembangkan prototipe yang mengirimkan koordinat lokasi GPS melalui pesan WhatsApp untuk mempercepat penanganan darurat[8]. Berdasarkan tinjauan terhadap penelitian terdahulu, ditemukan beberapa celah penelitian (*research gap*) yang menjadi dasar pengembangan aplikasi FirAlert. Tabel 2.1 merangkum perbandingan tersebut.

Tabel 2. 1 Penelitian Terkait

No	Penulis, Tahun	Teknologi/ Metode	Model, Algoritma, Instrumen	Capaian Hasil Evaluasi	Research Gap
1	Putra (2024)[8]	ESP32, MQ-2, GPS, WhatsApp	Prototipe IoT Terintegrasi WhatsApp.	Mampu mengirim lokasi GPS melalui pesan WhatsApp dalam waktu 5 detik.	Notifikasi sangat bergantung pada platform pihak ketiga (WhatsApp) yang bukan jalur khusus darurat.
2	Sesanti & Rahmanto (2025)[5]	NodeMCU, MQ-2, ThingSpeak	Eksperimental, Cloud ThingSpeak	Berhasil mengirimkan data setiap 15-20 detik dengan visualisasi <i>dashboard</i> web.	<i>Monitoring</i> berbasis web portal kurang responsif untuk mobilitas tinggi pengguna <i>smartphone</i> dibandingkan aplikasi native.
3	Lestari et al. (2025)[1]	ESP32, MQTT, Telegram	Protokol MQTT, Bot Telegram	Tingkat keberhasilan deteksi sebesar 98% dengan rata-rata respons 3,1 detik.	Belum menyediakan fitur visualisasi lokasi titik bahaya pada peta digital (<i>Google Maps</i>).
4	Sari et al. (2026)[6]	NodeMCU, MQ-2, Firebase, Android Studio	Firebase Database, Android Studio	Berhasil mengirimkan notifikasi "AWAS GAS BOCOR" ke aplikasi Android secara cepat.	Pengembangan berbasis Native Android Studio kurang efisien dalam integrasi <i>multiplatform</i> dan sinkronisasi asinkron modern.
5	Mulwindu et al. (2026) [24]	ESP32, MQ-2, Blynk	R&D (<i>Planning, Production, Evaluation</i>)	Waktu respons di ruang tertutup rata-rata $\pm 3,8$ detik dan ruang	Fitur sangat terbatas pada ekosistem standar platform Blynk sehingga kustomisasi

No	Penulis, Tahun	Teknologi/ Metode	Model, Algoritma, Instrumen	Capaian Hasil Evaluasi	Research Gap
				terbuka $\pm 5,8$ detik.	UI/UX tidak maksimal.

Kebaruan (*novelty*) penelitian ini dibandingkan dengan penelitian-penelitian terdahulu terletak pada pengembangan aplikasi *monitoring* mandiri ("FirAlert") yang dibangun menggunakan *Framework* Flutter, yang memungkinkan performa antarmuka yang sangat responsif dan efisiensi pengembangan lintas platform[25]. Berbeda dengan sistem terdahulu yang sangat bergantung pada platform pesan instan pihak ketiga seperti WhatsApp atau Telegram, sistem ini mengimplementasikan FCM sebagai jalur notifikasi push terdedikasi yang menjamin peringatan bahaya tetap diterima[26].

Selain itu, keunggulan signifikan dari penelitian ini adalah integrasi *Google Maps API* secara langsung di dalam aplikasi untuk memvisualisasikan titik koordinat lokasi kebocoran gas secara presisi[27]. Hal ini mengatasi keterbatasan pada penelitian-penelitian sebelumnya yang mayoritas hanya memberikan informasi status gas atau koordinat teks tanpa visualisasi peta yang interaktif[1]. Pemanfaatan *Firebase Realtime Database* juga memastikan sinkronisasi data konsentrasi gas berjalan secara asinkron dengan latensi yang sangat rendah, didukung oleh pengujian performa mendalam mencakup penggunaan sumber daya sistem (CPU dan Memori) untuk menjamin keandalan aplikasi sebagai instrumen keselamatan bagi pengguna Dapur MBG[22].

2.2. Landasan Teori

2.2.1. Sistem *Monitoring*

Sistem *monitoring* adalah proses pengumpulan, analisis, dan penyajian informasi secara kontinu untuk mendukung pengambilan keputusan dalam pengelolaan suatu kondisi[28]. Dalam penelitian ini, sistem *monitoring* berfungsi sebagai pusat kendali untuk memantau nilai konsentrasi gas yang dikirim oleh sumber data di Dapur MBG[29]. Aplikasi berperan mentransformasi data mentah dari sumber data menjadi informasi visual yang dapat diakses dari jarak jauh, sehingga meningkatkan kewaspadaan pengguna terhadap potensi bahaya[30].

2.2.2. *Monitoring Real-Time*

Monitoring real-time didefinisikan sebagai kemampuan sistem untuk menyajikan data terbaru secara instan sesaat setelah data tersebut diterima dari sumber[5]. Konsep ini sangat krusial dalam deteksi kebocoran gas karena setiap detik keterlambatan informasi dapat meningkatkan risiko ledakan. Penelitian ini memanfaatkan pembaruan data asinkron agar status "*Safe*" "*High*" atau "*Critical*" pada aplikasi selalu sinkron dengan kondisi aktual di lapangan tanpa perlu memuat ulang (*refresh*) halaman[23].

2.2.3. *Framework Flutter*

Flutter merupakan *Framework open-source* yang dikembangkan oleh Google untuk membangun aplikasi *multiplatform* dari satu basis kode menggunakan bahasa Pemrograman Dart[31]. Flutter menggunakan kompilasi *Ahead-of-Time* (AOT) untuk performa tinggi dan fitur *Hot Reload* untuk efisiensi pengembangan[25]. Flutter dipilih dalam penelitian ini untuk membangun antarmuka aplikasi "FirAlert" yang responsif, memastikan visualisasi grafik konsentrasi gas dan indikator status bahaya berjalan mulus di perangkat Android[32].

2.2.4. *Firebase*

Firebase adalah platform *Backend-as-a-Service* (BaaS) yang menyediakan infrastruktur *cloud* siap pakai untuk mempercepat pengembangan aplikasi.

1. *Firebase Realtime Database: Merupakan database NoSQL berbasis cloud yang menyimpan data dalam format JSON dan melakukan sinkronisasi secara instan ke seluruh klien yang terhubung[33]. Layanan ini berfungsi sebagai tulang punggung sinkronisasi data konsentrasi gas antara sumber data dengan aplikasi Flutter.*
2. *Firebase Authentication: Layanan keamanan yang menangani proses pendaftaran dan login pengguna secara aman. Dalam penelitian ini, fitur ini memastikan hanya pengguna*

terverifikasi yang dapat mengakses data monitoring Dapur MBG guna menjaga privasi dan keamanan data[34].

3. *Firebase Cloud Messaging (FCM): Layanan pengiriman pesan lintas platform untuk mengirimkan push notification secara real-time. FCM digunakan untuk mendorong peringatan bahaya langsung ke perangkat pengguna dengan rata-rata waktu respons yang sangat singkat[18].*

2.2.5. Google Maps API

Google Maps API adalah sekumpulan antarmuka pemrograman untuk mengintegrasikan peta digital ke dalam aplikasi. Dalam sistem ini, API tersebut digunakan untuk memvisualisasikan titik koordinat lokasi Dapur MBG yang mengalami potensi bahaya, sehingga pengguna dapat melihat lokasi presisi titik kebocoran secara visual untuk mempermudah proses mitigasi dan evakuasi[35].

2.2.6. Internet of Things (IoT)

IoT adalah infrastruktur jaringan global yang menghubungkan objek fisik dan virtual melalui sensor serta kemampuan komunikasi internet[36], [37]. IoT berperan sebagai konsep besar yang mendasari penelitian ini, di mana data dari sensor gas di lapangan dikirimkan menuju *cloud* untuk diolah dan ditampilkan melalui aplikasi *mobile* secara terintegrasi[38].

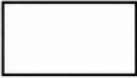
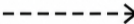
2.2.7. Use case Diagram

Use case diagram merupakan representasi visual yang mengilustrasikan aktivitas yang dilakukan oleh sistem dari perspektif pengguna, dengan fokus utama pada tindakan yang dilakukan oleh sistem dan bukan cara sistem melakukan tindakan tersebut[39]. Fungsi utama diagram ini adalah untuk mengidentifikasi fungsionalitas yang tersedia dalam sistem serta aktor-aktor (pengguna atau sistem lain) yang memiliki hak akses terhadap fungsi tersebut. Komponen utama dalam diagram ini meliputi aktor, use case (deskripsi fungsionalitas), dan *boundary* (batas sistem). Dalam penelitian pengembangan aplikasi FirAlert, *use case diagram* digunakan sebagai instrumen untuk memetakan kebutuhan pengguna secara prioritas agar pengembangan aplikasi tepat sasaran[40].

Implementasi *use case diagram* pada sistem deteksi dini kebocoran gas ini sangat krusial untuk mendefinisikan batas operasional aplikasi, seperti proses autentikasi akun (*login* dan *register*), pemantauan kadar gas secara *real-time*, hingga penerimaan notifikasi bahaya. Melalui pemodelan ini, pengembang dapat memastikan bahwa hanya pengguna yang terverifikasi yang dapat mengakses data *monitoring* Dapur MBG, sehingga integritas

dan keamanan informasi tetap terjaga sejalan dengan tujuan penelitian[41].

Tabel 2. 2 Use case Diagram

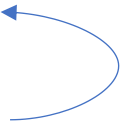
No	Simbol	Nama	Deskripsi	Keterangan Penggunaan
1		<i>Actor</i>	Mewakili pengguna yang berinteraksi dengan sistem.	Digunakan sebagai aktor <i>User</i> pada aplikasi FirAlert.
2		Use case	Menunjukkan layanan atau fungsi sistem.	Digunakan pada fitur <i>Registrasi</i> , <i>Login</i> , <i>Monitoring</i> , <i>Notifikasi</i> , dan <i>Lokasi</i> .
3		<i>System Boundary</i>	Menunjukkan batas sistem. Menunjukkan ketergantungan layanan.	Digunakan sebagai batas sistem aplikasi FirAlert.
4		<i>Association</i>	Menghubungkan <i>actor</i> dengan use case.	Menunjukkan hubungan <i>User</i> dengan fitur-fitur aplikasi.
5		<i>Dependency</i>	Menunjukkan ketergantungan layanan.	Digunakan untuk menunjukkan ketergantungan terhadap <i>Firebase Authentication</i> , <i>Firebase Realtime Database</i> , <i>Firebase Cloud Messaging (FCM)</i> , dan <i>Google Maps API</i> .

2.2.8. Activity Diagram

Activity diagram menggambarkan urutan kerja untuk setiap skenario penggunaan atau kasus penggunaan (*use case*), yang membantu dalam menyusun alur operasional sistem secara lebih mendalam. Diagram ini berfungsi untuk mendeskripsikan alur kerja (*workflow*) atau aliran proses aksi dalam suatu sistem, mulai dari titik awal hingga titik akhir. Penggunaan diagram ini sangat membantu dalam memahami proses yang terjadi secara detail, sehingga dapat digunakan untuk menganalisis dan mengembangkan strategi guna meningkatkan efisiensi proses perangkat lunak. Dalam pengembangan aplikasi FirAlert berbasis Flutter, diagram ini membantu memodelkan alur logika operasional, mulai dari inisialisasi koneksi hingga proses pengecekan data di *database*[42].

Dalam penelitian ini, *activity diagram* berperan untuk memvisualisasikan logika sistem deteksi dini secara sistematis, di mana aplikasi secara otomatis akan mengecek pembaruan data di *Firebase* dan menentukan keputusan logis apakah notifikasi bahaya perlu dimunculkan atau tidak berdasarkan ambang batas konsentrasi gas yang terdeteksi. Dengan demikian, alur informasi dari sumber data hingga ke antarmuka pengguna dapat berjalan secara runtut dan responsif[43].

Tabel 2. 3 Activity Diagram

No	Simbol	Nama	Deskripsi	Keterangan
1.		<i>Initial Node</i>	Menunjukkan awal proses.	Digunakan ketika aplikasi mulai dijalankan.
2.		<i>Activity</i>	Menunjukkan aktivitas sistem.	Digunakan pada Buka Aplikasi, <i>Dashboard Monitoring</i> , <i>Monitoring Data Gas</i> , <i>Update Dashboard</i> , dan Kirim Notifikasi.
3.		<i>Decision Node</i>	Menunjukkan percabangan berdasarkan kondisi.	Digunakan pada Status <i>Login</i> dan kondisi Nilai Gas.
4.		<i>Control Flow</i>	Menunjukkan alur aktivitas.	Menghubungkan seluruh aktivitas dalam sistem.
5.		<i>Loop</i>	Menunjukkan aktivitas yang dilakukan berulang.	Digunakan pada proses <i>monitoring</i> data secara <i>real-time</i> .

2.2.9. Sequence Diagram

Sequence diagram adalah diagram UML yang digunakan untuk menggambarkan urutan interaksi antar objek dalam sistem berdasarkan dimensi waktu. Fungsi utama diagram ini adalah menunjukkan bagaimana setiap komponen perangkat lunak berkomunikasi satu sama lain untuk menjalankan fungsionalitas tertentu dalam suatu periode waktu yang sekuensial. Komponen

utamanya meliputi *lifeline* (keberadaan objek), *activation* (waktu eksekusi), dan *message* (pertukaran informasi antar objek)[44]. Diagram ini sangat diperlukan dalam penelitian ini untuk memvisualisasikan proses transmisi data yang terjadi antara aplikasi Flutter, Firebase Realtime Database, dan layanan FCM.

Pada pengembangan aplikasi FirAlert, *sequence diagram* digunakan untuk memastikan bahwa seluruh tahapan pengiriman peringatan dini mulai dari pembaruan nilai gas di cloud hingga munculnya notifikasi pada perangkat Android pengguna berjalan dengan urutan yang tepat. Hal ini memungkinkan peneliti untuk menganalisis potensi tunda waktu (*latency*) pada setiap titik interaksi objek, guna mencapai tujuan pengiriman notifikasi yang bersifat *real-time* secara efektif[45].

Tabel 2. 4 Sequence Diagram

No	Simbol	Nama	Deskripsi	Keterangan
1.		<i>Object</i>	Objek yang terlibat dalam komunikasi.	<i>Firestore Realtime Database</i> , Flutter <i>Controller</i> , FCM, dan Android.
2.		<i>Lifeline</i>	Menunjukkan keberadaan objek selama proses berlangsung.	Digunakan pada seluruh objek.
3.		<i>Message</i>	Pesan antar objek.	Digunakan untuk pengiriman data <i>monitoring</i> dan <i>request</i> notifikasi.
4.		<i>Return Message</i>	Respon dari suatu proses.	Digunakan pada proses <i>Update Dashboard</i> dan <i>Push Notification</i> .
5.		<i>Combined Fragment</i>	Menunjukkan percabangan kondisi.	Digunakan pada kondisi <i>Critical/High</i> dan <i>Safe</i> .
6.		<i>Self Message</i>	Objek memproses dirinya sendiri.	Digunakan pada validasi nilai gas oleh Flutter <i>Controller</i> .

2.2.10. Class Diagram

Class diagram adalah jenis diagram UML yang menggambarkan struktur statis sistem dengan menunjukkan kelas-

kelas yang ada, atributnya, metode (*method*), serta hubungan antar kelas tersebut. Fungsi utamanya adalah memberikan gambaran mengenai arsitektur data dan logika internal perangkat lunak. Komponen utama dalam diagram ini terdiri atas nama kelas, atribut (data yang disimpan), dan operasi atau fungsi yang dapat dijalankan. Diagram ini digunakan untuk merancang model data pada Flutter agar sinkronisasi informasi dengan basis data NoSQL *Firestore* tetap terjaga integritasnya[46].

Penerapan *class diagram* dalam penelitian ini berfungsi untuk mendefinisikan entitas data penting seperti profil pengguna, data riwayat *monitoring*, dan koordinat lokasi bahaya yang akan divisualisasikan melalui *Google Maps API*. Dengan struktur kelas yang efisien, proses konversi data dari format JSON yang diterima dari *Firestore* menjadi objek Dart pada Flutter dapat dilakukan dengan penggunaan memori yang optimal, mendukung performa aplikasi yang ringan selama proses pemantauan berlangsung[47].

Tabel 2. 5 Class Diagram

No	Simbol	Nama	Deskripsi	Keterangan
1.		<i>Class</i>	Mewakili struktur kelas.	Digunakan pada <i>User Model</i> , <i>Gas Sensor Model</i> , <i>Notification Service</i> , dan <i>Location Model</i> .
2.		<i>Association</i>	Menunjukkan hubungan antar kelas.	Digunakan untuk menghubungkan kelas pada sistem.
3.	Label	<i>Relationship</i>	Menjelaskan jenis hubungan antar kelas.	Digunakan pada <i>monitor</i> , <i>triggers</i> , dan <i>has location</i>
4.	+	<i>Attribute</i>	Menyimpan data dalam kelas.	Contohnya <i>uid</i> , <i>email</i> , <i>latitude</i> , <i>longitude</i> , dan <i>token</i> .
5.	()	<i>Operation</i>	Fungsi yang dimiliki kelas.	Contohnya <i>login()</i> , <i>logout()</i> , <i>init FCM()</i> , <i>update Marker()</i> , dan <i>show Notification()</i> .