

BAB II

TINJAUAN PUSTAKA

A. Kajian Teoritis

1. Internet of Things

Menurut Mambang (2021) IoT merupakan seperangkat teknologi yang saling berkaitan dan terhubung ke internet, sehingga manusia tidak perlu mengontrol alat secara manual atau langsung. Menurut Choudhary (2024) *Internet of Things* merupakan hubungan perangkat keras dengan menghubungkan antar perangkat dan perangkat lunak dengan internet, sehingga dapat saling bertukar informasi secara otomatis. Menurut Oliveira (2024) *Internet of Things (IoT)* merupakan suatu paradigma yang menghubungkan berbagai perangkat sehingga dapat bekerja sama.

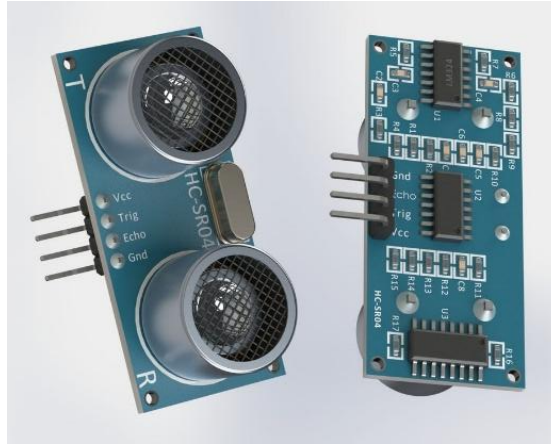
Berdasarkan dari beberapa kesimpulan di atas dapat disimpulkan bahwa *Internet of Things* adalah teknologi informasi yang memiliki manfaat mempermudah dalam kehidupan manusia. Dengan memanfaatkan internet maka perangkat dapat dikendalikan tanpa harus dikontrol secara manual dan dapat secara *real-time* memberikan informasi kepada pengguna. *Internet of Things* memanfaatkan internet untuk kehidupan sehari-hari sehingga menjadi bantuan besar dalam kehidupan di masa sekarang ini yang serba digital. Dengan

adanya *Internet of Things* maka dapat membantu kehidupan selama terkoneksi dengan internet.

2. Sistem Monitoring

Sistem monitoring merupakan sistem pemantauan yang dilakukan guna mengetahui suatu keadaan dengan berkala. Monitoring dilakukan guna untuk mendapatkan hasil yang nantinya akan menjadi bahan tujuan atau menjadi bahan evaluasi dalam suatu kondisi tertentu. Sistem monitoring ini tidak hanya memungkinkan pengguna untuk melihat data secara *real-time*, tetapi juga dapat dilengkapi dengan fitur *early warning* sistem untuk memberikan pemberitahuan dini ketika level air mencapai batas kritis (Al-Ikhsan & Sulaiman, 2026). Sistem monitoring menjadi hal yang dibutuhkan karena dalam melakukan monitoring sesuatu kondisi harus berdasarkan data yang ada sesuai dengan ketentuan yang ada dengan kondisi keadaan tersebut.

3. Sensor Ultrasonik HC-SR04



Gambar 2. 1 Sensor *ultrasonic HC-SR04*

Sumber : (Projects, 2020)

Menurut Hasruddin (2023) sensor *ultrasonic* adalah sensor yang dapat mengubah besaran fisis menjadi besaran listrik. Sensor ultrasonik menjadi sensor yang bermanfaat dalam mengubah suara menjadi menjadi listrik tanpa adanya sentuhan dengan objek langsung. Menurut Muttaqin (2021) Sensor ultrasonik dapat mengubah suara menjadi listrik ataupun sebaliknya dengan prinsip pemantulan gelombang suara yang berkisar 40 kHz. Sensor ultrasonik menjadi bagian komponen penting dalam perancangan *Internet of Things* yang membantu memberikan informasi melalui sinyal.

4. ESP32



Gambar 2. 2 Modul ESP32

Sumber : PCBWay (2025)

ESP32 merupakan mikrokontroler yang sudah memiliki modul WiFi di dalam chipnya, sehingga sangat mendukung dalam pembuatan aplikasi *Internet of Things*. *ESP32* memiliki beberapa fitur sebagai berikut: prosesor *dual-core*, konektivitas Wi-Fi dan *Bluetooth* terintegrasi, jumlah pin *General Purpose Input/Output* (GPIO) yang banyak, dan konsumsi daya yang rendah (Hercog et al., 2023). *ESP32* menjadi mikrokontroler yang sangat berguna dalam penelitian ini dikarenakan sudah kompleks dan lebih canggih. Dari segi efisiensi energi, *ESP32* sudah memiliki berbagai mode hemat daya yang menjadikan perangkat tetap responsif dengan konsumsi energi yang tergolong rendah (Bhaganagare et al., 2025). *ESP32* memberikan manfaat yang lebih banyak dengan teknologi dan keunggulan yang lebih banyak dalam perancangan teknologi berbasis *Internet of Things*.

5. Buzzer



Gambar 2. 3 Buzzer

Sumber : AliExpress (2026).

Buzzer adalah salah satu komponen elektronika yang dapat mengganti sinyal listrik menjadi getaran suara (Nizam et al., 2022). *Buzzer* sering digunakan sebagai alarm atau penanda peringatan terhadap suatu kondisi yang sedang terjadi. Ketika sinyal listrik diterapkan pada *buzzer*, membran atau penggetar akan bergetar dengan frekuensi tertentu, menghasilkan bunyi atau nada yang terdengar (Anggara et al., 2023). *Buzzer* sering digunakan untuk indikator suara pada berbagai elektronik. Elektronik yang menggunakan *buzzer* adalah sistem keamanan, perangkat medis, peralatan rumah tangga, kendaraan dan sistem yang berbasis mikrokontroler. Penggunaan *buzzer* bertujuan memberikan informasi atau peringatan kepada pengguna ketika terjadi suatu kondisi tertentu, misalnya kesalahan sistem, keberhasilan sistem atau adanya bahaya.

6. Kabel Jumper



Gambar 2. 4 Kabel Jumper

Sumber : Pinterest (2006)

Dalam merancang Sistem monitoring ketinggian air berbasis IoT, dibutuhkan sebuah komponen yang dapat menggabungkan setiap bagiannya. Kabel jumper merupakan kabel yang memiliki pin yang dapat dihubungkan tanpa menggunakan perekat seperti lem atau solder. Menurut Anggara (2023) Menjelaskan bahwa kabel jumper merupakan kabel yang memiliki konektor pin di setiap ujungnya, sehingga memberi kemudahan dalam menghubungkan dua komponen yang terhubung tanpa menggunakan solder. Kabel jumper memberikan kemudahan dalam menyatukan modul dengan cara yang mudah. Dengan adanya kabel jumper ini pengguna dapat dengan mudah mengatur atau bahkan mengubah rangkaian komponen dengan cepat.

7. Servo MG996R



Gambar 2. 5 Servo MG996R

Sumber : Pinterest (2025)

Dalam pembuatan pintu yang dapat dibuka atau ditutup secara *real-time* tentunya diperlukan penggerak yang menggunakan mekanisme *closed feedback*. Penggerak ini adalah motor servo yang terdiri dari penggerak, *gear*, potensiometer dan rangkaian untuk sistem kontrol. Menurut Lesmana (2025) Motor servo adalah sistem kendali tertutup yang menggunakan umpan balik dalam mengukur gerakan dan posisi akhir. Rangkaian motor servo terdiri dari motor Dc, rangkaian roda gigi, potensiometer dan rangkaian kendali. Potensiometer berfungsi sebagai penentu batas sudut dari putaran servo. Sudut poros motor servo diatur berdasarkan lebar pulsa yang dikirimkan melalui pin sinyal pada kabel motor (biasanya warna jingga). Umumnya, motor servotidak dapat bergerak kontinu seperti motor DC lain dan hanya bergerak

mencapai sudut tertentu saja. Servo dikemas dalam bentuk kotak segi empat yang terdiri dari 3 jenis konektor kabel yaitu tegangan (Vdd), I/O kontrol, dan *grounding* (Vss) (Sanjaya et al., 2023).

8. Firebase Database



Firestore

Gambar 2. 6 Aplikasi *Firebase*

Sumber : Google (n.d.), *Firebase Brand Guidelines*.

Basis data atau *Data Base* merupakan media penyimpanan data yang menjadi sumber berkumpulnya beberapa data yang sudah terdaftar secara otomatis. Basis data memberikan manfaat untuk menyimpan riwayat suatu keadaan dalam penelitian, pengembangan dan pelaksanaan sebuah penelitian, sehingga dapat membantu dalam evaluasi suatu penelitian. Menurut Mahendra dan Winardi (2023) menjelaskan mengenai *Firebase Realtime Database* adalah *database* yang berbasis *Cloud*, sehingga memungkinkan saling pertukaran data secara *real-time*.

Firebase Realtime Database merupakan layanan yang dirancang guna menyimpan data secara *real-time*, guna menyimpan setiap perubahan data aplikasi. *Firebase Realtime Database* selalu digunakan untuk pengembangan aplikasi atau dalam pengembangan *Internet of Things*. Hal ini dikarenakan *Firebase Realtime Database* dapat menyediakan informasi data yang cepat, sederhana dan efisien.

9. Bahasa Pemrograman C++

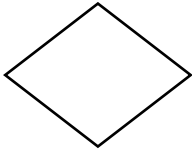
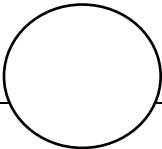
Bahasa pemrograman atau bahasa komputer merupakan bahasa yang digunakan dalam ilmu komputer yang menggunakan bahasa inggris sebagai simbolnya seperti *IF* untuk jika dan lainnya. Bahasa pemrograman C++ merupakan bahasa turunan dari bahasa C yang diberikan fitur PBO (Pemrograman Berorientasi Objek), sehingga dapat menghasilkan program yang lebih kompleks dan lebih mudah dikembangkan dibandingkan bahasa C. Bahasa pemrograman C++ merupakan bahasa program yang fleksibel dan memberikan kemudahan dalam pengembangan atau membangun aplikasi.

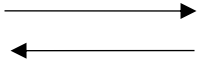
Bagi bidang *Internet of Things*, C++ adalah bahasa yang digunakan dalam memprogram mikrokontroler, seperti ESP32. Bahasa ini memungkinkan pengembang mengakses pin input/output, membaca data dari sensor, mengendalikan aktuator, serta mengelola komunikasi berbagai bagian, seperti Wi-Fi, Bluetooth, maupun MQTT.

10. *Flowchart*

Diagram alir atau *Flowchart* adalah suatu bentuk yang menggambarkan langkah-langkah secara berurutan atau alur proses dalam suatu sistem, program atau bahkan algoritma. *Flowchart* disusun menggunakan simbol-simbol standar yang saling dihubungkan dengan garis alir untuk menunjukkan hubungan dan urutan setiap proses. Dalam pengembangan perangkat lunak, *flowchart* berperan sebagai alat bantu dalam merancang logika program sebelum proses pengkodean dilakukan.

Tabel 2. 1 Simbol-simbol *Flowchart*

Simbol	Nama	Fungsi
	<i>Proses</i>	Menunjukkan suatu proses digunakan untuk melambangkan perhitungan dan perubahan nilai peubah
	<i>Operasi</i>	Menyatakan suatu proses masukan (<i>input</i>) atau keluaran (<i>output</i>) data tanpa ketergantungan terhadap peralatannya
	<i>Terminal</i>	Menunjukkan awal atau akhir suatu tahapan dalam <i>Flowchart</i>
	<i>Decision</i>	Menyatakan pengujian terhadap suatu kondisi tertentu yang akan menghasilkan
	<i>Document</i>	Menunjukkan dokumen yang di hasilkan atau dibutuhkan dalam proses
	<i>Stored Data</i>	Menggambarkan alur penyimpanan data
	<i>Connector</i>	Menyatakan keterkaitan antara satu proses dengan proses

	lainnya dalam satu halaman yang serupa
	<i>Flow</i> Menggambarkan bagaimana jalannya suatu arus proses

11. UML (Unifed Modeling Language)

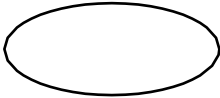
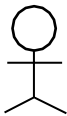
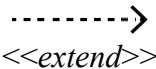
Unifed Modeling Language (UML) merupakan standar dalam bahasa pemodelan yang berguna untuk mendokumentasikan suatu sistem secara berurutan. UML memiliki beberapa jenis diantaranya *Use Case Diagram*, *Activity Diagram*, *Sequence Diagram*, *Class Diagram*, *State Diagram* dan *Deployment Diagram*. UML adalah standar Internasional dalam pemodelan yang merepresentasikan perilaku, struktur dan interaksi antar sistem. Dengan adanya UML maka dapat menjadi konsisten dalam proses analisis, perancangan, implementasi dan dokumentasi.

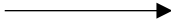
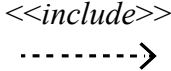
a. *Use Case Diagram*

Use Case Diagram berfungsi untuk menggambarkan perilaku dari suatu sistem melalui sudut pandang pengguna atau aktor. Menurut Gunawan (2022) tujuan *use case diagram* adalah untuk mengetahui apa saja fungsi yang tersedia dan siapa saja yang memiliki akses dalam menggunakan sistem tersebut. Sehingga, dengan *use case diagram* maka dapat mengetahui bagaimana prosesnya tanpa mengimplementasikan sistem secara langsung. Pada pengembangan sistem berbasis Internet of Things

(IoT), *Use Case Diagram* digunakan untuk memodelkan interaksi antara pengguna dengan perangkat IoT maupun aplikasi pendukung. Sebagai contoh, pada sistem monitoring suhu berbasis ESP32 dan *Firebase*, aktor yang terlibat dapat berupa Administrator dan Pengguna. Administrator memiliki hak untuk mengelola perangkat, mengatur parameter sistem, dan melihat data monitoring, sedangkan pengguna hanya dapat memantau data sensor dan menerima notifikasi apabila terjadi kondisi tertentu. Dengan adanya *Use Case Diagram*, kebutuhan fungsional sistem dapat didokumentasikan secara sistematis sehingga memudahkan proses pengembangan dan pengujian sistem.

Tabel 2. 2 Simbol *Use Case Diagram*

Simbol	Nama	Keterangan
	<i>Use Case</i>	<i>Use case</i> merupakan interaksi antara aktor dengan sistem.
	<i>Actor</i>	<i>Actor</i> yang akan melakukan interaksi dengan sistem.
	<i>Association</i>	<i>Association</i> adalah jembatan yang akan menghubungkan <i>user</i> dengan <i>use case</i> .
	<i>Extend</i>	<i>Extend</i> adalah tambahan dari <i>usecase</i> namun mandiri.

	<i>Generalization</i>	<i>Generalization</i> adalah hubungan dari beberapa <i>usecase</i> .
	<i>Include</i>	<i>Include</i> adalah relasi dari <i>usecase</i> inti terhadap <i>usecase</i> tambahan yang akan dijalankan bersama.

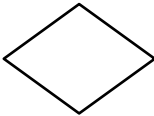
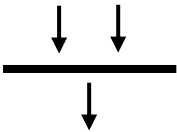
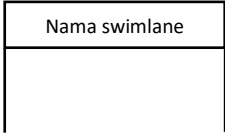
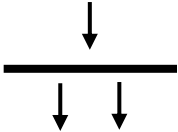
b. *Deployment Diagram*

Deployment Diagram menggambarkan arsitekur suatu hubungan antara hardware dan software secara visual dari aplikasi (Wildan et al., 2021). Dengan adanya *deployment diagram*, penelitian dapat dipahami dengan alur antar perangkat yang dipastikan dapat berjalan dengan baik. Menurut Sopriani (2023) menjelaskan bahwa *deployment diagram* adalah penggambaran dari detail komponen dari infrastruktur sistem, di mana komponen akan terletak pada mesin, server atau piranti keras.

c. *Activity Diagram*

Activity diagram adalah alur aktivitas dari sebuah sistem yang menampilkan suatu aktivitas dari dimulai, dijalankan sampai selesai. *Activity Diagram* memiliki rangkaian aktivitas dari keputusan (*decision*), percabangan (*branch*), penggabungan (*merge*) dan aktivitas yang dilakukan secara bersamaan (*parallel process*). *Activity diagram* yang dapat dilihat pada Tabel 2.3.

Tabel 2. 3 Simbol-simbol *Activity diagram*

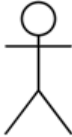
Simbol	Nama	Keterangan
	Status awal	Status awal adalah awal dari sistem.
	<i>Activity</i>	Aktivitas bertujuan untuk menggambarkan aktivitas dari sebuah sistem.
	<i>Decision</i>	Percabangan merupakan aktivitas yang memiliki lebih dari satu kondisi atau pilihan.
	<i>Join</i>	Penggabungan digunakan untuk menghubungkan aktivitas satu dengan aktivitas yang lain.
	Status akhir	Status akhir merupakan akhir dari aktivitas sistem.
	<i>Swimlane</i>	<i>Swimlane</i> merupakan penanggung jawab dari setiap aktivitas.
	<i>Line connector</i>	<i>Line Connector</i> digunakan untuk menghubungkan satu simbol dengan simbol lainnya.
	<i>Fork</i>	<i>Fork</i> digunakan untuk percabangan secara parallel dari aktivitas.

d. *Sequence Diagram*

Sequence Diagram merupakan salah satu diagram perilaku yang berguna untuk menggambarkan urutan interaksi antara aktor dan objek dalam suatu

sistem berdasarkan dimensi waktu. *Sequence Diagram* berfokus pada urutan komunikasi antar objek, dimulai dari aktor yang memberikan permintaan (*request*) kepada sistem, kemudian sistem memproses permintaan tersebut dan memberikan respons. Simbol-simbol *sequence diagram* dapat dilihat pada tabel 2.4.

Tabel 2. 4 Simbol-simbol *Sequence diagram*

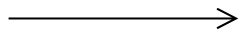
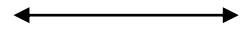
Simbol	Nama	Keterangan
	<i>Actor</i>	<i>Actor</i> merupakan pengguna yang berinteraksi secara langsung dengan sistem
	<i>Life Line</i>	Objek <i>entity</i> dan antarmuka (<i>boundary</i>) saling berinteraksi untuk mendukung jalannya sistem
	<i>Message</i>	Menjelaskan spesifikasi komunikasi antar objek yang memuat informasi mengenai aktivitas yang berlangsung
	<i>Message</i>	Menggambarkan spesifikasi komunikasi antar objek yang berisi informasi terkait aktivitas yang terjadi dalam sistem

e. *Class Diagram*

Class diagram digunakan untuk menggambarkan relasi antara komponen *hardware*, sensor, mikrokontroler dan pengguna. Melalui *Class Diagram*, pengembang dapat menggambarkan struktur data, fungsi dari setiap objek

dan hubungan antar komponen, hal ini berguna dalam proses implementasi dan pemeliharaan menjadi lebih mudah.

Tabel 2. 5 *Class Diagram*

Simbol	Nama	Keterangan
nama kelas +atribut +operasi()	<i>Class</i>	<i>Class</i> merupakan struktur dari sebuah sistem
 nama_interface	<i>Interface</i>	Menggunakan konsep <i>interface</i> yang sama pada <i>Object Oriented Programming</i> (OOP).
	<i>Association</i>	<i>Association</i> merupakan penghubung antara classifiers atau dengan yang lain.
	Asosiasi berarah / <i>directed association</i>	Hubungan dari satu kelas dengan kelas yang lain dimana pada kelas lain terdapat atribut yang sama.
	Ke bergantungan/ <i>dependency</i>	Ke bergantungan merupakan kelas yang saling bergantung.
	Agregasi/ <i>aggregation</i>	Agregasi merupakan hubungan antar kelas yang bersifat semua bagian.
	<i>Generalisasi</i>	<i>Generalisasi</i> merupakan hubungan antar kelas yang bersifat umum-khusus.

12. Eksperimen

Eksperimen adalah suatu percobaan yang dilakukan untuk mengetahui suatu hasil dari aksi dan reaksi variabel dalam kontrol pengawasan. Eksperimen

menjadi metode yang dilakukan dalam proses membangun alat yang tentunya membutuhkan hasil dari pengujian alat tersebut. Dalam metode ini, peneliti melakukan pengamatan terhadap perubahan yang terjadi setelah suatu tindakan berhasil dilakukan. Setelah itu, pengamat akan membandingkan hasilnya dengan kondisi sebelum perlakuan.

Pada Penelitian berbasis *Internet of Things*, eksperimen dilakukan guna melakukan pengujian kemampuan perangkat dalam rancangan. Tahapan eksperimen adalah perencanaan, pelaksanaan, pengumpulan data, analisis hasil dan kesimpulan. Tahapan yang sistematis membantu memastikan bahwa hasil penelitian memiliki tingkat validitas yang baik.

13. Rekayasa sistem

Rekayasa sistem merupakan serangkaian tahapan yang digunakan dalam merancang, mengembangkan dan menjalankan agar dapat bekerja secara berstruktur sesuai dengan tujuan. Rekayasa sistem tentunya tidak hanya berhubungan dengan *software* tetapi juga berhubungan dengan *hardware*, sehingga saling berhubungan satu sama lain. Rekayasa sistem dapat membangun sistem yang lebih optimal dan menghasilkan kinerja yang lebih baik sesuai kebutuhan. Pada pengembangan sistem berbasis *Internet of Things* (IoT), rekayasa sistem memiliki peran penting dalam mengintegrasikan berbagai komponen, seperti sensor, mikrokontroler, jaringan komunikasi, basis data, layanan *cloud*, dan aplikasi pengguna.

14. *Black Box Testing*

Dalam pengujian *software*, penggunaan *Blackbox testing* menjadi strategi pengujian untuk mengetahui bagaimana keadaan sistem program apakah sukses atau adanya kendala. *Black box testing* adalah salah satu metode pengujian *software* yang berfokus pada pengujian fungsionalitas tanpa memperhatikan struktur internal, kode program ataupun logika yang digunakan dalam proses pengembangannya. Dalam pelaksanaannya, *Black Box Testing* dilakukan dengan memberikan berbagai variasi data masukan kepada sistem, kemudian membandingkan hasil yang diperoleh dengan hasil yang diharapkan. Oleh karena itu, *Black Box Testing* sering digunakan pada tahap pengujian fungsional untuk memastikan seluruh fitur telah berjalan sesuai dengan kebutuhan pengguna. Pada sistem berbasis *Internet of Things (IoT)*, *Black Box Testing* digunakan untuk menguji fungsi-fungsi utama yang melibatkan interaksi antara perangkat keras, perangkat lunak, dan layanan *cloud*. Setiap fungsi diuji berdasarkan skenario yang telah dirancang tanpa melihat implementasi kode program yang digunakan.

15. *Prototype*

Prototype adalah salah satu metode pengembangan sistem yang membuat program dengan cara pendekatan secara cepat dan bertahap, sehingga dapat dievaluasi oleh pengguna dengan segera. *Prototype* menjadi model awal dari alat yang akan dibangun secara struktur, sistematis dan lebih mudah dievaluasi dalam pembuatannya. *Prototype* bersifat berulang kali atau *iterative*

dalam prosesnya, sehingga memberikan kemudahan dalam menghasilkan sistem yang lebih efektif, efisien dan sesuai kebutuhan pengguna.

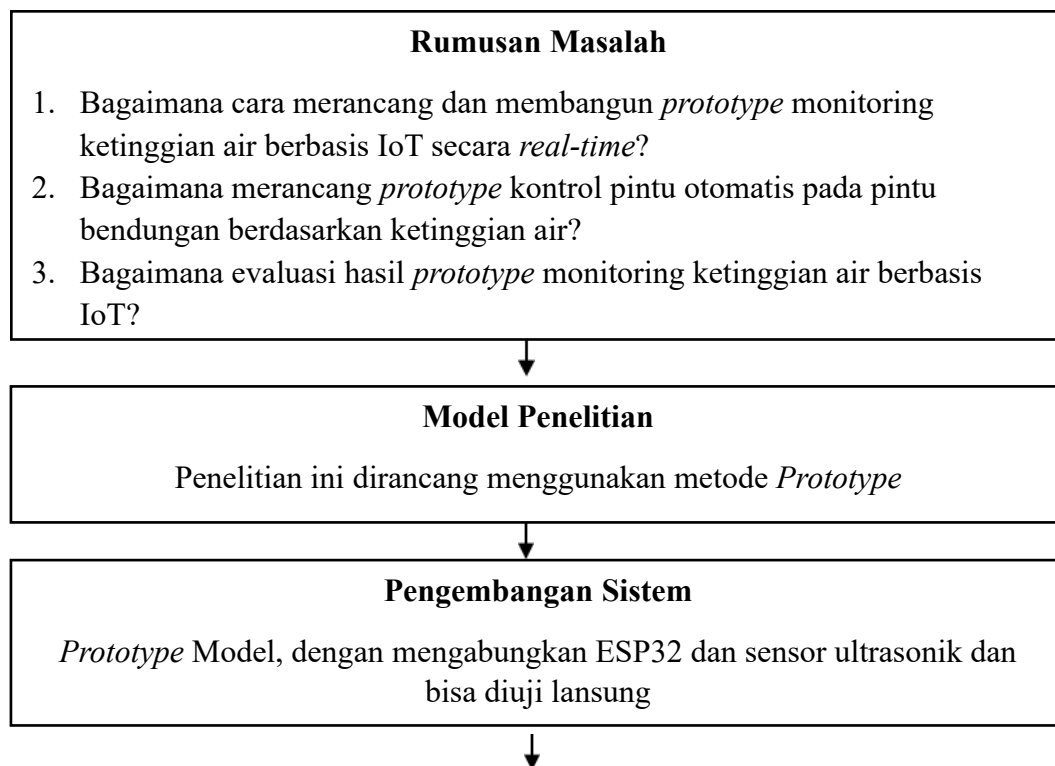
B. Kajian Empiris

Pada penelitian yang dilakukan (Baskoro et al., 2021) adalah merancang *prototype* dengan judul “Rancang Bangun *Prototype* Monitoring Ketinggian Air Pada Bendungan Berbasis *Internet of Things*”. Pada penelitian ini penulis menggunakan metode *prototype* pada penelitiannya dengan menggunakan sensor ultrasonik terhadap *prototype* monitoring ketinggian airnya. Tujuan dari penelitian ini adalah untuk membangun *prototype* yang dapat memonitoring ketinggian air pada bendungan dan memberikan informasi secara *real-time*. Hasil penelitian ini adalah berhasil menciptakan *prototype* yang dapat memonitoring ketinggian air pada bendungan dan berhasil memberikan informasi setiap waktu pada semua orang.

Pada Penelitian yang dilakukan oleh (Poetra et al., 2023) adalah merancang *prototype* dengan judul “*Prototype* Sistem Monitoring Ketinggian Air Pada Tangki Berbasis *Internet of Things*”. Pada penelitian ini penulis menggunakan metode *prototype* pada penelitiannya dengan menggunakan sensor ultrasonik untuk membaca ketinggian air pada tangki air. Tujuan dari penelitian ini adalah untuk mengetahui seberapa tinggi air di dalam tangki, sehingga dapat diketahui air tersebut meluap atau tangki kosong. Hasil dari penelitian ini adalah berhasil menciptakan *prototype* yang dapat membaca ketinggian air pada tangki, sehingga dapat memberikan informasi kepada pengguna. Dari penelitian berikut dapat disimpulkan bahwa penelitian tersebut menggunakan sensor ultrasonik untuk mengukur dan memonitoring ketinggian air.

C. Kerangka Berpikir

Perkembangan teknologi informasi mendorong perubahan pola pikir masyarakat dalam mencegah bencana yang terjadi akibat cuaca yang tidak terkendali lagi akibat Globalisasi dan kebiasaan masyarakat. Pada saat ini yang serba digital, maka kecepatan dalam mendapat informasi mengenai perubahan kondisi lingkungan menjadi hal penting untuk mencegah kerusakan maupun korban. Monitoring ketinggian air berbasis IoT menjadi salah satu hal yang dibutuhkan untuk beberapa wilayah di Indonesia yang masih mengalami banjir, dengan adanya sistem ini tidak hanya dapat mengetahui mengenai kondisi ketinggian air, namun juga dapat melakukan pencegahan dini untuk mengantisipasi terjadinya banjir pada pemukiman warga.



Implementasi

ESP32 sebagai mikrokontroler utama dan sensor ultrasonik HC-SR04 akan menjadi pendeteksi ketinggian air dan jika air melebihi batas maka akan mengaktifkan *buzzer* dan mengirimkan nontifikasi ke pengguna dengan aplikasi *Framework monitoring* secara *real-time*.



Hasil Penelitian

Prototype Monitoring Ketinggian Air Dengan *Ultrasonic Sensor Hc-Sr04*
Berbasis *Internet Of Things*