BAB II

TINJAUAN PUSTAKA

2.1 Teori Terkait

Penelitian oleh Riadi, (2023) mengembangkan prototipe sistem monitoring air berbasis *Internet of Things* (IoT) yang mampu mengukur konsumsi air dan menghitung biaya penggunaannya. Sistem ini menggunakan sensor IoT yang terintegrasi dengan aplikasi *web* untuk memberikan informasi *real-time* kepada pengguna mengenai volume air yang digunakan serta analisis biaya terkait.[4]

Amertha, (2022) merancang sistem monitoring dan kontrol tangki air menggunakan Raspberry Pi yang terhubung dengan bot Telegram. Sistem ini dapat mendeteksi kekeruhan dan volume air dalam tangki, serta mengontrol pompa air melalui perintah yang dikirimkan via Telegram. Penggunaan Raspberry Pi memungkinkan pengolahan data secara lokal sebelum dikirimkan ke pengguna[5].

Firdhouzi, (2020) mengembangkan sistem monitoring kualitas air dengan menggunakan arsitektur jaringan IoT. Sistem ini memanfaatkan protokol komunikasi seperti AMQP dan *WebSocket* untuk mentransfer data secara *real-time* ke antarmuka *web*. Parameter yang dipantau meliputi kekeruhan dan pH air, yang penting untuk memastikan kualitas air tetap terjaga.[6]

Batubara, (2020) merancang sistem monitoring kanal air sebagai sarana penanggulangan banjir. Sistem ini menggunakan NodeMCU ESP8266 untuk mengirimkan data tinggi permukaan air ke *server* melalui metode HTTP POST.

Data yang dikumpulkan digunakan untuk memberikan peringatan dini kepada masyarakat melalui aplikasi android ketika potensi banjir terdeteksi.[7]

Nainggolan, (2022) mengembangkan sistem monitoring kekeruhan air dan suhu pada akuarium ikan cupang berbasis *web*. Sistem ini menggunakan sensor turbidity dan suhu yang terhubung dengan Arduino, serta menampilkan data secara *real-time* melalui antarmuka *web*. Tujuan utama dari sistem ini adalah untuk memudahkan pemilik akuarium dalam memantau kondisi air secara efisien.[8]

Berdasarkan tinjauan dari penelitian-penelitian sebelumnya, dapat disimpulkan bahwa integrasi antara perangkat keras dan perangkat lunak dalam sistem monitoring air telah banyak dilakukan dengan berbagai pendekatan. Namun, proyek *"Pembuatan Website Monitoring Kran Air Otomatis Menggunakan Framework CodeIgniter"* menawarkan keunggulan dengan menggabungkan Raspberry Pi Pico W sebagai pengontrol utama, sensor flow Aichi untuk mendeteksi aliran air, ball valve sebagai pengatur buka tutup kran, serta *relay* 2 channel sebagai pengontrol aktuator. Selain itu, penggunaan framework CodeIgniter memungkinkan pengembangan antarmuka *web* yang responsif dan interaktif, sehingga pengguna dapat memantau dan mengendalikan penggunaan air secara real-time dari jarak jauh. Keunggulan ini menjadikan sistem lebih efisien dan *user-friendly* dibandingkan dengan pendekatan sebelumnya.

2.2 Landasan Teori

2.2.1 Internet Of Things

IoT yang merupakan singkatan dari *Internet of Things*, merupakan kemajuan teknologi yang memungkinkan pengguna memantau dan mengontrol perangkat yang terhubung ke internet dari jarak jauh. Dengan memanfaatkan sensor, teknologi ini mampu melacak dan memantau berbagai kondisi vital seperti tingkat kelembaban, suhu udara, kebocoran air, hingga bahaya kebakaran di dalam rumah atau suatu ruangan [9]. Dalam konteks smarthome, alat berfungsi untuk mengumpulkan, mengolah, menyimpan, dan mendistribusikan data terkait kondisi tanaman, serta status kran otomatis. Perangkat sepert, mikrokontroler, dan aktuator kran otomatis digunakan untuk mengatur penyiraman tanaman secara otomatis, baik melalui kontrol manual menggunakan aplikasi atau sistem yang beroperasi mandiri berdasarkan parameter yang telah ditentukan.

2.2.2 Framework Codeigniter

CodeIgniter merupakan aplikasi *open source* berupa *framework* PHP untuk membuat *website* lebih dinamis menggunakan PHP dengan pola MVC (Model, View, Controller), memudahkan pengembang *web* untuk membuat dan membangun aplikasi *web* dengan cepat dari awal. Selain membuat *web* menjadi lebih dinamis, proses ini juga dapat membantu pengembang

membangun aplikasi *web* yang ringan dan cepat. CodeIgniter mempunyai dokumentasi dengan contoh implementasi kode yang sangat lengkap, Dokumentasi yang lengkap ini menjadi salah satu alasan kuat mengapa banyak orang memilih CodeIgniter sebagai pilihannya. [10]

2.2.3 Visual Studio Code

Visual Studio Code adalah salah satu *Integrated Development Environment* (IDE) yang sangat efektif dan populer di kalangan pengembang. IDE ini memiliki fitur-fitur yang sangat lengkap dan memudahkan pengembang dalam membuat aplikasi *mobile*. Dengan mempelajari Visual Studio Code, pemula dapat menggunakan IDE 1 ini untuk mengembangkan aplikasi mobile dengan lebih mudah dan cepat.[9]

2.2.4 Laragon

Laragon merupakan *platform* pengembangan lokal yang dirancang untuk memudahkan pengembang dalam membangun dan mengelola aplikasi *web* secara efisien. Dikenal karena instalasinya yang cepat dan konfigurasi yang minimal, Laragon menyediakan paket lengkap yang mencakup *server web* (Apache atau Nginx), database (MySQL atau MariaDB), dan dukungan untuk berbagai bahasa pemrograman seperti PHP, Node.js, Python, Ruby, dan lainnya .

2.2.5 Application Programming Interface (API)

Application Programming Interface (API) adalah antarmuka yang dibangun oleh pengembang sistem sehingga beberapa atau semua fungsi sistem dapat diakses secara terprogram. Peneliti menunjukkan bahwa pengembangan API telah berhasil, dan implementasi REST telah mempermudah pengembangan struktur API. Dengan bantuan API, user dapat register atau login ke aplikasi hanya dengan mengirimkan email ke aplikasi dan memudahkan proses transaksi donasi. API juga dapat berperan dalam memudahkan user melakukan berbagai metode pembayaran untuk membuat proses checkout menjadi lebih cepat dan mudah

2.2.6 Arduino IDE

Arduino IDE adalah perangkat lunak adalah *software* yang digunakan untuk menulis dan mengunggah program ke papan mikrokontroler seperti Arduino, ESP8266, atau Raspberry Pi Pico. Bahasa yang digunakan berbasis C/C++, dan sudah didukung berbagai perpustakaan untuk memudahkan pemrograman sensor dan aktuator. Antarmukanya sederhana, cocok untuk pemula, dan dilengkapi fitur seperti verifikasi kode, upload, dan serial monitor

2.2.7 Thonny

Thonny adalah lingkungan pengembangan terintegrasi (IDE) yang dirancang untuk pemula dalam pemrograman Python. Dikembangkan oleh Aivar Annamaa dari Universitas Tartu, Estonia, Thonny menawarkan antarmuka yang sederhana dan fitur-fitur seperti debugger visual, tampilan variabel secara *real-time*, dan kemampuan untuk menjalankan kode langkah demi langkah.

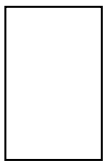
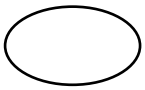
2.2.8 Micropython

Micropython adalah implementasi ringan dari Python 3 yang dioptimalkan untuk dijalankan pada mikrokontroler seperti ESP32, ESP8266, dan Raspberry Pi Pico. MicroPython memungkinkan pengembangan aplikasi IoT dengan bahasa Python yang lebih sederhana dibandingkan dengan bahasa pemrograman tingkat rendah seperti C atau C++.

2.2.9 Use Case Diagram

Dalam pengembangan sistem, *use case diagram* merupakan deskripsi fungsi dari sebuah sistem, dari sudut pandang para pengguna sistem. *Use case diagram* berjalan dengan menggunakan scenario yang merupakan deskripsi dari urutan atau langkah-langkah yang menjelaskan apa yang dilakukan oleh user terhadap sistem ataupun sebaliknya[14]. Simbol-simbol *use case diagram* bisa di lihat pada Tabel 2. 1

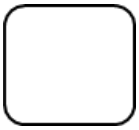
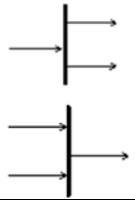
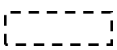
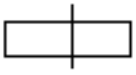
Tabel 2. 1 Tabel Diagram Usecase

No	Simbol	Nama	Keterangan
1.		System	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
2.		Actor	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
3.		Use Case	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
4.		Association	Simbol yang menghubungkan antara objek satu dengan objek lainnya
5.		Include	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
6.		Extend	Menspesifikasikan bahwa *use case* target memperluas perilaku dari use case sumber pada suatu titik yang diberikan.
7.		Generalization	Hubunagn dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya.
8.		Dependency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*)akan mempengaruhi element yang bergantung padanya elemen yang tidak mandiri.

2.2.10 Activity Diagram

Diagram yang menggambarkan aliran kerja atau aktivitas dalam sebuah sistem pada perangkat lunak. *Activity diagram* merepresentasikan proses yang dijalankan oleh sistem, bukan tindakan yang dilakukan oleh aktor, sehingga aktivitas yang ditampilkan adalah yang dilakukan oleh sistem itu sendiri[15]. Simbol dalam *activity diagram* bisa di lihat pada Tabel 2. 2

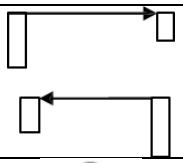
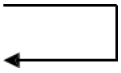
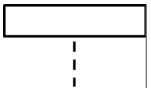
Tabel 2. 2 Tabel Activity Diagram

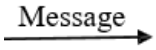
No.	Simbol	Nama	Keterangan
1.		Initial Node	Bagaimana objek dibentuk atau diawali.
2.		Final Node	Bagaimana objek dibentuk dan dihancurkan.
3.		Activity	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
4.		Action	State dari sistem yang mencerminkan eksekusi suatu aksi.
5.		Decision	Pilihan untuk mengambil keputusan
6.		Fork Node	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran.
7.		Fork/Join	Digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu
8.		Rake	Menunjukkan adanya dekomposisi
9.		Send	Tanda pengiriman
10.		Time	Tanda waktu

2.2.11 Sequence Diagram

Diagram urutan *(squencce diagram)* merupakan diagram yang menjelaskan interaksi antar objek dan menunjukkan komunikasi antara objek-objek tersebut. *squence diagram* digunakan untuk menggambarkan perilaku dalam suatu sistem serta menjelaskan bagaimana entitas dan sistem saling berinteraksi, termasuk pesan yang digunakan dalam interaksi tersebut. Pesan ditampilkan sesuai dengan urutan eksekusi[16]. Simbol *squence diagram* bisa dilihat pada Tabel 2. 3

Tabel 2. 3 Tabel Sequence Diagram

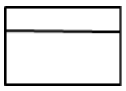
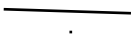
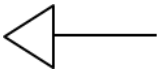
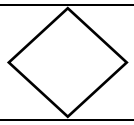
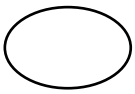
No	Simbol	Nama	Keterangan
1.		Activation	Sebagai sebuah objek yang akan melakukan sebuah aksi
2.		Message	Spesifikasi dari komunikasi antar objek yang memuat informasi – informasi tentang aktifitas yang terjadi.
3.		Actor	Menggambarkan orang yang sedang berinteraksi dengan sistem
4.		Boundary Class	Menggambarkan penggambaran dari *form*
5.		Entity Class	Mengambarkan hubungan kegiatan yang akan dilakukan
6.		Control Class	Menggambarkan penghubung antara *Boundary* dengan tabel
7.		Self Message	Mengindikasikan komunikasi kembali kedalam sebuah objek itu sendiri
8.		LifeLine	Objek entity, antar muka yang saling berinteraksi.

No	Simbol	Nama	Keterangan
9.		Message	Mengindikasikan komunikasi antara objek dengan objek

2.2.12 Class Diagram

Class diagram adalah jenis pemodelan yang digunakan untuk menggambarkan struktur basis data dan kelas objek dalam sebuah sistem. Tujuannya adalah untuk mendefinisikan kelas-kelas pada basis data. Diagram ini memperlihatkan bagaimana kelas-kelas saling berhubungan dan berinteraksi untuk menciptakan kerangka dasar dari sebuah sistem software[17]. Simbol Class Diagram bisa dilihat pada Tabel 2. 4

Tabel 2. 4 Tabel Class Diagram

No.	Simbol	Nama	Keterangan
1.		Class	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
2.		Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.
3.		Generalization	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*)
4.		Nary Association	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
5.		Colaboration	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
6.		Dependency	Operasi yang benar benar dilakukan oleh suatu objek.

No.	Simbol	Nama	Keterangan
7.		Dependency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri.