

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **2.1 Penelitian Terkait**

Penelitian yang dilakukan oleh Rifki Alfarez Riantama, Titin Fatimah (2022) dengan judul “Sistem Monitoring Dan Pemberian Pakan Ikan Otomatis Menggunakan ESP32CAM Berbasis Web” membahas bahwa pemberian pakan ikan merupakan kegiatan rutin yang sering terlewatkan oleh pembudidaya ikan karena lupa atau adanya kesibukan lainnya yang sulit ditinggalkan maka untuk mengatasi masalah ini dibuatlah sebuah alat pemberi pakan ikan otomatis. Pemberian pakan dilakukan secara otomatis berdasarkan waktu yang telah ditentukan atau disetting sebelumnya. Maka dirancanglah sebuah sistem monitoring dan pemberian pakan ikan otomatis menggunakan esp32cam berbasis web [3]

Penelitian yang dilakukan oleh Muhammad Haikal, Mursyidah, Muhammad Nasir (2023) dengan judul “Penerapan *IoT* (Internet Of Thing) Pada Sistem Monitoring Dan Kontrol Aquarium Berbasis Web Service” membahas bahwa monitoring dan kontrol dalam aquarium salah satunya yaitu memantau kondisi pakan ikan dan kondisi kekeruhan air pada aquarium. Metode biasa yang saat ini digunakan adalah dengan melakukan monitoring dan kontrol secara manual yaitu dengan datang langsung ke aquarium yang dilakukan oleh pemilik ikan hias itu sendiri. hal tersebut tentu timbul permasalahan yaitu dibutuhkan banyak tenaga dan waktu untuk menjaga dan merawat kesehatan air serta pakan aquarium setiap saat. Sehingga para

pemilik tidak dapat meninggalkan aktivitas tersebut, yang menyebabkan kontrol dan pengamatan tidak efektif dan efisien lagi. Oleh karena itu dibangunnya sebuah penerapan Internet of Thing pada sistem monitoring dan kontrol aquarium berbasis web service [4].

Penelitian yang dilakukan oleh Arkan Fadillah, Purwanto, Utomo Budiyo, Safrina Amini (2023) dengan judul “Sistem Monitoring dan Pemberian Pakan Ikan Hias Otomatis Berbasis Web” Membahas bahwa masalah yang sering terjadi adalah pemberian pakan ikan hias tidak tepat waktu bahkan terlupakan karena kesibukan dari penghobi ikan hias tersebut dan monitoring suhu air tidak dilakukan secara rutin dan *realtime*. Untuk mengatasi masalah pada pemberian pakan dan monitoring suhu air dirancanglah sebuah sistem pemberian pakan hias otomatis dan monitoring suhu berbasis *internet of things* sehingga dapat membantu pekerjaan penggemar atau penghobi ikan hias, Tujuan penelitian adalah membuat sistem monitoring dan pemberian pakan ikan hias otomatis dengan sensor suhu DS18B20, ESP32Cam dan mikrokontroler NodeMCU ESP8266 berbasis web Berdasarkan hasil pengujian terhadap sistem pemberian pakan otomatis ini sistem berhasil memberikan pakan ikan hias sesuai jadwal dan mengirim foto dari habitat ikan hias ke web server [5].

Penelitian yang dilakukan oleh Iksal Rachman, Muhamad Rijal Budiman, Ahmad Muhajir, Rizal Mi'raz (2023) dengan judul “*Smart Aquarium berbasis IoT dengan Web Server dan intelligent OTA*” membahas bahwa Pemeliharaan ikan sebagai makhluk hidup merupakan hal yang tidak

mudah apalagi dengan jenis ikan yang cukup mahal dan unik di dalam jenisnya. Hal yang harus di perhatikan dalam merawat ikan adalah pemberian pakan secara tepat waktu memerhatikan kualitas air dan monitoring karakter atau tingkah laku ikan setiap waktunya. Maka dari itu di buat System otomatisasi menggunakan esp 32 dan esp 32 cam sebagai mikrokontroler yang terkoneksi dengan jaringan internet agar dapat mengirim dan menerima data dengan *website* yang telah di buat. Intruksi pengendalian dan monitoring ikan melalui kamera seacara realtime di tampilkan melalui tampilan *website* sehingga pengguna dapat menggunakan system smart aquarium dimanapun berada [6].

Penelitian yang dilakukan oleh Alif Bahsyar, Rini Handayani S.T., M.T., Marlinda Ike S.T., M.T. dengan judul “Perancangan Dan Pembangunan Antar Muka Sistem Pemberi Pakan Ikan Otomatis Berbasis Web” (2017) membahas bahwa memelihara ikan dalam kolam memerlukan tindakan yang berupa pemberian makan ikan dan perawatan kolam itu sendiri. Pemilik kolam harus senantiasa memantau kondisi kolam, dan memberikan dosis makanan yang tepat untuk ikan. Tetapi saat pemilik kolam tidak berada di lokasi dan bepergian dalam waktu yang cukup lama, maka pemberian makan dan perawatan kolam pun kurang terkontrol. Sehingga dibutuhkan sebuah alat yang dapat membantu para peternak ikan kolam dalam memberikan pakan dan merawat kolam ikan itu sendiri, sehingga dalam proyek akhir ini akan dibangun sebuah antarmuka sistem pemberi pakan ikan otomatis berbasis *website*. Dimana sistem tersebut akan ditambahkan ke alat *prototype* pemberi

pakan ikan, sistem ini juga akan menyediakan fitur “*monitoring* dan *controlling*” yang akan mempermudah para peternak mengawasi ikan dan mengontrol kolam ikan mereka [7].

## 2.2 Landasan Teori

### 2.2.1 Ikan Air Tawar

Ikan air tawar adalah jenis ikan yang menjalani sebagian atau seluruh siklus hidupnya di habitat air tawar.[8]. Pada dasarnya ikan air tawar lebih mudah dibandingkan dengan ikan air laut tetapi terdapat kendala utama yang terjadi pada ikan air tawar yaitu diperlukannya waktu dan biaya yang cukup tinggi [9].



Gambar 2. 1 Ikan air tawar

### 2.2.2 Sistem Monitoring

Sistem *Monitoring* merupakan suatu proses untuk mengumpulkan data dari berbagai sumber daya. Biasanya data yang dikumpulkan merupakan data yang *realtime*. Secara umum tujuan *monitoring* adalah untuk mendapatkan data - data atau pandangan agar diperoleh umpan balik bagi kebutuhan tertentu.

### 2.2.3 Website

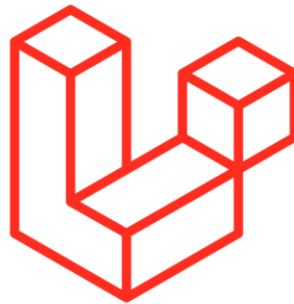
*Website* merupakan kumpulan informasi yang terdiri dari halaman *web* yang saling terhubung satu sama lain yang disediakan secara perorangan, kelompok, atau pun organisasi.[10] Situs *web* yang baik menampilkan visual yang menarik dan berfungsi sesuai dengan kebutuhan pengguna .

### 2.2.4 Framework

*Framework* adalah komponen pemrograman yang telah siap digunakan kapan saja, sehingga memudahkan programmer tidak membuat skrip yang sama dalam tugas yang sama. *Framework* biasanya digunakan untuk efisiensi dan meningkatkan produktivitas dalam pembuatan perangkat lunak.

### 2.2.5 Laravel

*Laravel* adalah sebuah *framework* PHP yang dirilis dibawah lisensi MIT, dibentuk dengan model MVC (*Model View Controller*) yang ditulis dengan bahasa PHP yang dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pembuatan awal dan pemeliharaan dengan aplikasi yang banyak menyediakan sintaks yang jelas dan menghemat waktu .



Gambar 2. 2 Logo *laravel*

### 2.2.6 Database

Database atau basis data adalah kumpulan data yang terstruktur dan terorganisir secara sistematis agar dapat diakses, dikelola, dan diperbarui dengan mudah. Database digunakan untuk menyimpan informasi dalam berbagai bentuk seperti data pengguna, transaksi, produk, atau informasi lainnya [11].

### 2.2.7 Visual Studio Code

*Visual Studio Code* (VS Code) adalah editor kode sumber (source code editor) yang dikembangkan oleh *Microsoft*. *VS Code* banyak digunakan oleh programmer untuk menulis, mengedit, dan menjalankan berbagai bahasa pemrograman seperti *Python*, *JavaScript*, *PHP*, *Java*, *C++*, dan lainnya [12].



Gambar 2. 3 Logo *Visual Studio Code*

### 2.2.8 UML

UML adalah bahasa grafik/visualisasi yang digunakan untuk memvisualisasikan, mendefinisikan, membangun, dan mendokumentasikan sistem pengembangan perangkat lunak berbasis berorientasi objek. Pemodelan perancangan sistem aplikasi alopel ini memodelkan beberapa diagram UML, antara lain *Use Case*, *Activity* diagram, *Class* diagram, dan *Sequence* diagram [13].






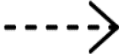



Gambar 2. 4 Logo UML (Unified Modeling Language)

### 2.2.9 Use Case Diagram




*Use Case* Diagram adalah deskripsi tentang bagaimana pengguna (aktor) berinteraksi dengan sistem untuk mencapai suatu tujuan tertentu. Dalam pengembangan perangkat lunak, *Use Case*

digunakan untuk memahami kebutuhan sistem dan menggambarkan bagaimana suatu fitur akan digunakan. Simbol *Use Case* diagram bisa dilihat pada tabel

Tabel 2. 1 *UseCase* Diagram

No	Gambar	Nama	Keterangan
1		Actor	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
2		Dependency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri ( <i>independent</i> ) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri ( <i>independent</i> ).
3		Generalization	Hubungan dimana objek anak ( <i>descendent</i> ) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk ( <i>ancertor</i> ).
4		Include	Menspesifikasikan bahwa <i>use case</i> sumber secara eksplisit.
5		Extend	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
6		Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		System	Menspesifikasikan paket yang menampilkan sistem secara terbatas.









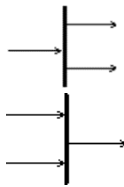

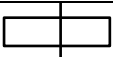
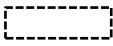
No	Gambar	Nama	Keterangan
8		UseCase	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
9		Collaboration	Interaksi aturan - aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen - elemennya (sinergi).
10		Note	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

### 2.2.10 Activity Diagram

*Activity Diagram* adalah salah satu diagram dalam *UML* (*Unified Modeling Language*) yang digunakan untuk menggambarkan alur kerja atau proses dalam sebuah sistem. Diagram ini menunjukkan bagaimana suatu aktivitas dimulai, dikerjakan, dan berakhir [14]. komponen simbol dalam *activity* diagram bisa dilihat pada tabel.

Tabel 2. 2 *Activity Diagram*

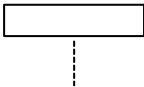
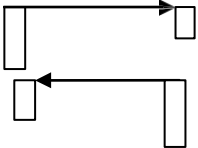
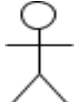
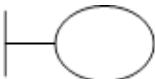




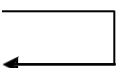
No	Gambar	Nama	Keterangan
1		Activity	Memperlihatkan bagaimana masing – masing kelas antarmuka saling berinteraksi satu sama lain.
2		Action	State dari sistem yang mencerminkan eksekusi suatu aksi.
3		Initial Node	Bagaimana objek dibentuk atau diawali.

No	Gambar	Nama	Keterangan
4		Final Node	Bagaimana objek dibentuk dan dihancurkan.
5		Fork Node	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran.
6		Decision	Pilihan untuk mengambil keputusan.
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		Time	Tanda waktu.
10		Send	Tanda pengiriman.

### 2.2.11 Sequence Diagram

*Sequence* diagram adalah *tool* yang sangat populer dalam pengembangan sistem informasi secara *objectoriented* untuk menampilkan interaksi antar objek. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara objek juga interaksi antara objek. komponen simbol *Sequence* bisa dilihat pada tabel.

Tabel 2. 3 *Sequence Diagram*

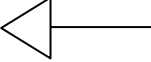
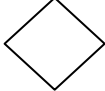
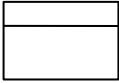


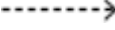

No	Gambar	Nama	Keterangan
1		LifeLine	Objek entity, antar muka yang saling berinteraksi.
2		Message	Spesifikasi dari komunikasi antar objek yang memuat informasi – informasi tentang aktifitas yang terjadi.
3		Actor	Menggambarkan orang yang sedang berinteraksi dengan sistem.
4		Boundary Class	Menggambarkan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

### 2.2.12 Class Diagram

*Class Diagram* adalah salah satu jenis diagram *UML* yang digunakan untuk menampilkan kelas maupun paket yang ada pada suatu sistem yang nantinya akan digunakan untuk memberikan

gambaran mengenai sistem maupun relasi-relasi yang terdapat pada sistem tersebut [15]. *Class* diagram menggambarkan struktur statis dalam sebuah sistem, untuk simbol dalam *class* diagram bisa dilihat pada tabel.

Tabel 2. 4 *Class* Diagram

No	Gambar	Nama	Keterangan
1		Generalization	Hubungan dimana objek anak ( <i>descendent</i> ) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk ( <i>ancestor</i> ).
2		Nary Association	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		Class	Himpunan dari objek – objek yang berbagi atribut serta operasi yang sama.
4		Collaboration	Deskripsi dari urutan aksi – aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5		Dependency	Operasi yang benar – benar dilakukan oleh suatu objek.
6		Dependency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri ( <i>independent</i> ) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7		Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.