

**IMPLEMENTASI *INTERNET OF THINGS* UNTUK
PENYIRAMAN DAN PENGKABUTAN OTOMATIS PADA
TANAMAN MENGGUNAKAN ALGORITMA *NAÏVE BAYES***
(Studi Kasus di Avicenna *Greenhouse*)

SKRIPSI

**Karya Tulis sebagai syarat memperoleh
Gelar Sarjana Komputer dari Fakultas Teknologi Informasi
Universitas Bale Bandung**

Disusun oleh:

ADAM SETIADI
NPM.301210013



**PROGRAM STRATA 1
PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAR BALE BANDUNG
BANDUNG
2025**

ABSTRAK

Perkembangan teknologi *Internet of Things (IOT)* telah membawa dampak besar dalam berbagai bidang, termasuk sektor pertanian. Salah satu implementasi yang menonjol adalah otomatisasi sistem penyiraman dan pengkabutan tanaman, khususnya di lingkungan *Greenhouse*. Kondisi ini menuntut pengelolaan lingkungan yang konsisten dan berkelanjutan agar pertumbuhan tanaman tetap optimal dan hasil pertanian yang dihasilkan berkualitas serta efisien.

Penelitian ini bertujuan merancang dan menerapkan sistem otomatis untuk penyiraman dan pengkabutan tanaman cabai berbasis *IOT*, dengan algoritma *Naïve bayes* sebagai metode prediksi dalam pengambilan keputusan. Sistem menggunakan sensor suhu udara, kelembapan udara, dan kelembapan tanah untuk memantau kondisi lingkungan secara langsung dan terus-menerus. Data dari sensor diolah oleh algoritma untuk menentukan apakah perlu dilakukan penyiraman atau pengkabutan, sesuai hasil klasifikasi kondisi. Sistem ini juga dilengkapi fitur notifikasi melalui antarmuka web yang memungkinkan pengguna memantau dan mengontrol sistem dari jarak jauh. Dengan klasifikasi cerdas berbasis data nyata, sistem dapat mengambil keputusan secara otomatis dan akurat tanpa campur tangan manual. Penerapan sistem ini diharapkan bisa menekan pemborosan air, mengurangi kerja manual, serta menjaga kestabilan iklim mikro di *Greenhouse* guna menunjang pertumbuhan tanaman secara optimal dan berkelanjutan.

Hasil pengujian menunjukkan bahwa sistem dapat berjalan secara otomatis, cepat merespons perubahan kondisi lingkungan, dan dikontrol dengan mudah melalui aplikasi berbasis web. Penerapan sistem ini di *Avicenna Greenhouse* menunjukkan adanya peningkatan efisiensi penggunaan air, pengurangan intervensi manual, serta perbaikan stabilitas suhu dan kelembapan di dalam *Greenhouse*. Sistem ini tidak hanya meningkatkan efektivitas proses penyiraman dan pengkabutan tanaman, tetapi juga mendukung untuk praktik pertanian presisi yang hemat sumber daya dan dapat diterapkan oleh petani di *Greenhouse*.

Kata Kunci: *Internet of Things, Naïve bayes, Pengkabutan, Penyiraman.*

ABSTRACT

The development of Internet of Things (IOT) technology has had a major impact on various fields, including the agricultural sector. One prominent implementation is the automation of plant watering and misting systems, especially in Greenhouse environments. This condition requires consistent and sustainable environmental management so that plant growth remains optimal and the agricultural products produced are of high quality and efficient.

This study aims to design and implement an automated system for watering and misting chili plants based on IOT, with the Naïve bayes algorithm as a prediction method in decision making. The system uses air temperature, air humidity, and Soil Moisture sensors to monitor environmental conditions directly and continuously. Data from the sensors is processed by the algorithm to determine whether watering or misting is necessary, according to the results of the condition classification. This system is also equipped with a notification feature via a web interface that allows users to monitor and control the system remotely. With intelligent classification based on real data, the system can make decisions automatically and accurately without manual intervention. The implementation of this system is expected to reduce water waste, reduce manual work, and maintain microclimate stability in the Greenhouse to support optimal and sustainable plant growth.

The test results show that the system can run automatically, respond quickly to changes in environmental conditions, and be easily controlled through a web-based application. The application of this system in the Avicenna Greenhouse shows an increase in the efficiency of water use, a reduction in manual intervention, and an improvement in the stability of temperature and humidity in the Greenhouse. This system not only increases the effectiveness of the plant watering and fogging process, but also supports for resource-efficient precision farming practices that can be applied by farmers in the Greenhouse.

Keywords: *Internet of Things, Misting, Naïve bayes, Watering.*

KATA PENGANTAR

Puji syukur atas kehadirat Allah SWT atas rahmat dan hidayah-Nya yang melimpah, sehingga penulis dapat menyelesaikan laporan skripsi yang berjudul “IMPLEMENTASI INTERNET OF THINGS UNTUK PENYIRAMAN DAN PENGKABUTAN OTOMATIS PADA TANAMAN MENGGUNAKAN ALGORITMA NAIVE BAYES (Studi Kasus di Avicenna Greenhouse)”. Shalawat serta salam tidak lupa disampaikan kepada junjungan Nabi Muhammad SAW yang senantiasa memberikan teladan dan petunjuk yang luhur dalam setiap aspek kehidupan.

Ucapan terima kasih yang tulus disampaikan kepada kedua Orang Tua yang senantiasa memberikan kasih sayang, dukungan, dan doa yang tidak terbatas. Serta, ucapan terima kasih juga penulis sampaikan kepada pihak yang telah membantu dalam penyusunan laporan skripsi, di antaranya:

1. Bapak Yudi Herdiana, S.T., M.T., selaku Dekan Fakultas Teknologi Informasi Universitas Bale Bandung.
2. Bapak Yusuf Muharam, S.Kom., M.Kom., selaku Ketua Program Studi Teknik Informatika Fakultas Teknologi Informasi Universitas Bale Bandung Sekaligus dosen pembimbing utama.
3. Bapak Yaya Suharya, S.Kom, M.Kom., selaku pembimbing pendamping.
4. Segenap Bapak/Ibu Dosen di Fakultas Teknologi Informasi.
5. Semua pihak yang telah membantu dan memberikan dukungan kepada penulis untuk menyelesaikan usulan penelitian ini.

Penulis menyadari usulan penelitian ini masih jauh dari sempurna, untuk itu kritik dan saran yang membangun penulis harapkan demi perbaikan dimasa yang akan datang. Akhir kata, penulis berharap semoga laporan skripsi ini diterima dan bermanfaat kepada berbagai pihak.

Bandung, Mei 2025

Penulis

DAFTAR ISI

| | |
|---|-------------|
| ABSTRAK | i |
| ABSTRACT | ii |
| KATA PENGANTAR..... | iii |
| DAFTAR ISI..... | iv |
| DAFTAR TABEL | vii |
| DAFTAR GAMBAR..... | viii |
| DAFTAR LAMPIRAN | x |
| BAB I PENDAHULUAN..... | 1 |
| 1.1 Latar Belakang | 1 |
| 1.2 Rumusan Masalah..... | 4 |
| 1.3 Batasan Masalah..... | 4 |
| 1.4 Tujuan Penelitian | 5 |
| 1.5 Metodologi Penelitian | 5 |
| 1.6 Sistematika Penulisan | 6 |
| BAB II TINJAUAN PUSTAKA..... | 8 |
| 2.1 Landasan Teori..... | 8 |
| 2.2 Dasar Teori..... | 13 |
| 2.2.1 <i>Internet of Things</i> | 13 |
| 2.2.2 Penyiraman Tanaman..... | 14 |
| 2.2.3 Pengkabutan Tanaman | 15 |
| 2.2.4 <i>Machine Learning</i> | 17 |
| 2.2.5 Algoritma <i>Naïve bayes</i> | 18 |
| 2.2.6 <i>ESP32</i> | 19 |
| 2.2.7 <i>Soil Moisture Sensor</i> | 21 |
| 2.2.8 <i>DHT11</i> | 22 |
| 2.2.9 <i>Nozzle Sprayer</i> | 23 |
| 2.2.10 <i>Relay</i> | 24 |
| 2.2.11 <i>Unified Modeling language</i> | 25 |
| 2.2.12 <i>Website</i> | 28 |
| 2.2.13 <i>Hosting</i> | 29 |

| | | |
|--------|--|-----------|
| 2.2.14 | <i>Metode Agile</i> | 29 |
| 2.2.15 | <i>Github</i> | 31 |
| 2.2.16 | <i>Arduino</i> | 32 |
| 2.2.17 | <i>Arduino IDE</i> | 33 |
| 2.2.18 | <i>MQTT</i> | 34 |
| 2.2.19 | <i>MySQL</i> | 35 |
| 2.2.20 | <i>Figma</i> | 36 |
| 2.2.21 | <i>Python</i> | 37 |
| 2.2.22 | <i>Flask</i> | 39 |
| 2.2.23 | <i>Draw.io</i> | 40 |
| 2.2.24 | <i>Visual Studio Code</i> | 41 |
| 2.2.25 | <i>Fritzing</i> | 41 |
| 2.2.26 | <i>HTML</i> | 42 |
| 2.2.27 | <i>CSS</i> | 43 |
| 2.2.28 | <i>JavaScript</i> | 44 |
| 2.2.29 | <i>HiveMQ</i> | 46 |
| 2.2.30 | <i>Whatsapp-web.js</i> | 46 |
| | BAB III METODOLOGI PENELITIAN | 48 |
| 3.1 | Kerangka Pikir | 48 |
| 3.2 | Deskripsi | 49 |
| 3.2.1 | Identifikasi Masalah..... | 49 |
| 3.2.2 | Pengumpulan Data | 49 |
| 3.2.3 | Perencanaan..... | 49 |
| 3.2.4 | Perancangan | 52 |
| 3.2.5 | Pengembangan | 53 |
| 3.2.6 | Pengujian..... | 54 |
| 3.2.7 | Penerapan | 54 |
| 3.2.8 | Peluncuran..... | 54 |
| 3.2.9 | Pembuatan Laporan..... | 55 |
| | BAB IV ANALISIS DAN PERANCANGAN | 56 |
| 4.1 | Analisis..... | 56 |
| 4.1.1 | Analisis Masalah | 56 |
| 4.1.2 | Analisis <i>Hardware</i> | 56 |
| 4.1.3 | Analisis <i>Software</i> | 57 |

| | | |
|---|--|------------|
| 4.1.4 | Analisis Pengguna | 58 |
| 4.1.5 | <i>User Interface</i> | 59 |
| 4.1.6 | Fitur - fitur..... | 59 |
| 4.1.7 | Analisis Data | 60 |
| 4.1.8 | Analisis Biaya | 61 |
| 4.2 | Perancangan | 63 |
| 4.2.1 | Blok Diagram Alat dan Sistem | 63 |
| 4.2.2 | Diagram Alur Alat dan Sistem..... | 65 |
| 4.2.3 | Denah <i>Greenhouse</i> | 67 |
| 4.2.4 | <i>Unified Modeling Language (UML)</i> | 69 |
| 4.2.5 | Struktur Tabel..... | 79 |
| 4.2.6 | Perancangan <i>Wiring Diagram</i> | 81 |
| 4.2.7 | Desain Antarmuka..... | 82 |
| BAB V IMPLEMENTASI DAN PENGUJIAN | | 87 |
| 5.1 | Implementasi | 87 |
| 5.1.1 | Listing Program..... | 87 |
| 5.1.2 | Implementasi Sistem | 106 |
| 5.1.3 | Spesifikasi Sistem | 106 |
| 5.1.4 | Instalasi Sistem | 107 |
| 5.1.5 | Menjalankan Sistem | 109 |
| 5.2 | Pengujian..... | 114 |
| BAB VI KESIMPULAN | | 119 |
| 6.1 | Kesimpulan | 119 |
| 6.2 | Saran..... | 119 |
| DAFTAR PUSTAKA | | 121 |
| LAMPIRAN | | 125 |

DAFTAR TABEL

| | |
|---|-----|
| Tabel 2. 1 Acuan Jurnal Penelitian | 8 |
| Tabel 2. 2 Pengujian Kelembapan Tanah | 14 |
| Tabel 2. 3 Pengujian <i>DHT11</i> | 15 |
| Tabel 2. 4 <i>Use Case Diagram</i> | 25 |
| Tabel 2. 5 <i>Activity Diagram</i> | 26 |
| Tabel 2. 6 <i>Entity Relationship Diagram</i> | 27 |
| Tabel 3. 1 Perangkat Keras | 51 |
| Tabel 3. 2 Perangkat <i>Internet of Things</i> | 51 |
| Tabel 3. 3 Perangkat Lunak | 51 |
| Tabel 4. 1 Analisis Biaya | 62 |
| Tabel 4. 2 Tabel diagram konsep | 64 |
| Tabel 4. 3 Deskripsi Diagram Alur Alat dan Sistem | 65 |
| Tabel 4. 4 Deskripsi Denah <i>Greenhouse</i> | 68 |
| Tabel 4. 5 Deskripsi Aktor | 70 |
| Tabel 4. 6 Tabel data sensor | 79 |
| Tabel 4. 7 Tabel jadwal penyiraman | 80 |
| Tabel 4. 8 tabel riwayat aksi | 80 |
| Tabel 4. 9 tabel nomor_hp | 80 |
| Tabel 4. 10 Deskripsi <i>Wiring Diagram</i> | 82 |
| Tabel 4. 11 <i>User Interface Login</i> | 83 |
| Tabel 4. 12 <i>User Interface Dashboard</i> | 83 |
| Tabel 4. 13 <i>User Interface Data Sensor</i> | 84 |
| Tabel 4. 14 <i>User Interface Kontrol Alat</i> | 85 |
| Tabel 4. 15 <i>User Interface Whatsapp</i> | 86 |
| Tabel 5. 1 Pengujian alat mikrokontroller..... | 114 |
| Tabel 5. 2 Pengujian <i>Dashboard Monitoring</i> | 115 |

DAFTAR GAMBAR

| | |
|---|----|
| Gambar 2. 1 Konsep <i>Internet of Things</i> | 13 |
| Gambar 2. 2 Skema <i>Artifial Intelligence dan Machine Learning</i> | 17 |
| Gambar 2. 3 <i>ESP32</i> | 20 |
| Gambar 2. 4 <i>Soil Mastuire Sensor</i> | 21 |
| Gambar 2. 5 <i>DHT11</i> | 22 |
| Gambar 2. 6 <i>Nozzle Sprayer</i> | 23 |
| Gambar 2. 7 <i>Relay</i> | 24 |
| Gambar 2. 8 Arsitektur <i>Website</i> | 28 |
| Gambar 2. 9 Metode <i>Agile</i> | 30 |
| Gambar 2. 10 <i>GitHub</i> | 31 |
| Gambar 2. 11 Bahasa Pemrograman <i>Arduino</i> | 32 |
| Gambar 2. 12 <i>Arduino IDE</i> | 34 |
| Gambar 2. 13 Arsitektur <i>MQTT</i> | 34 |
| Gambar 2. 14 Arsitektur <i>MySQL</i> | 35 |
| Gambar 2. 15 Tampilan <i>Figma</i> | 36 |
| Gambar 2. 16 Arsitektur <i>Python</i> | 38 |
| Gambar 2. 17 Arsitektur <i>Flask</i> | 39 |
| Gambar 2. 18 <i>Draw.io</i> | 40 |
| Gambar 2. 19 <i>Visual Studio Code</i> | 41 |
| Gambar 2. 20 <i>Fritzing</i> | 42 |
| Gambar 2. 21 Arsitektur <i>HTML</i> | 43 |
| Gambar 2. 22 Arsitektur <i>CSS</i> | 44 |
| Gambar 2. 23 Arsitektur <i>Javascript</i> | 45 |
| Gambar 3. 1 Kerangka Pikir..... | 48 |
| Gambar 4. 1 Blok diagram alat dan sistem | 63 |
| Gambar 4. 2 Diagram alur alat dan sistem..... | 65 |
| Gambar 4. 3 Denah <i>Greenhouse</i> | 68 |
| Gambar 4. 4 <i>Use Case Diagram</i> | 70 |
| Gambar 4. 5 <i>Activity Login</i> | 71 |
| Gambar 4. 6 <i>Activity Dashboard</i> | 72 |

| | |
|---|-----|
| Gambar 4. 7 <i>Activity Dashboard</i> Data Sensor | 73 |
| Gambar 4. 8 <i>Activity Whatsapp</i> | 74 |
| Gambar 4. 9 <i>Activity Kontrol Alat</i> | 75 |
| Gambar 4. 10 <i>Activity Logout</i> | 77 |
| Gambar 4. 11 <i>Class Diagram</i> | 78 |
| Gambar 4. 12 <i>Wiring Diagram</i> | 81 |
| Gambar 5. 1 Halaman <i>Login</i> | 109 |
| Gambar 5. 2 Halaman <i>Dashboard</i> | 109 |
| Gambar 5. 3 Gambar Halaman Data Sensor | 110 |
| Gambar 5. 4 Gambar Halaman Kontrol Alat | 111 |
| Gambar 5. 5 Halaman <i>whatsapp</i> | 112 |
| Gambar 5. 6 Notifikasi <i>Whatsapp</i> | 113 |

DAFTAR LAMPIRAN

| | |
|---|-----|
| Lampiran 1:Hasil Wawancara..... | 125 |
| Lampiran 2: Dokumentasi Wawancara..... | 127 |
| Lampiran 3: TOR | 128 |
| Lampiran 4: Instalasi Alat Di <i>Greenhouse</i> | 129 |

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam era digital saat ini, *Internet of Things (IOT)* telah banyak diterapkan dalam berbagai bidang, termasuk pertanian. Salah satu implementasi *IOT* yang semakin berkembang adalah sistem penyiraman dan pengkabutan otomatis yang bertujuan untuk meningkatkan efisiensi dan efektivitas penggunaan air. Dengan memanfaatkan sensor dan algoritma *Machine Learning*, sistem ini mampu menyesuaikan kondisi penyiraman dan pengkabutan secara *real-time*. Serta dilakukan penerapan algoritma *Naïve bayes* ke dalam sistem otomatisasi penyiraman tanaman berdasarkan data sensor dapat signifikan meningkatkan efisiensi dan efektifitas dalam manajemen penggunaan air (Alamsyah et al., 2024).

Penelitian ini dilaksanakan di Avicenna *Greenhouse*, yang berlokasi di Kp. Padarek Rt.03 Rw.02, Desa Drawati, Kecamatan Paseh, Kabupaten Bandung, Provinsi Jawa Barat. Avicenna *Greenhouse* merupakan sebuah rumah kaca yang difokuskan pada budidaya tanaman cabai dan telah berdiri sejak tahun 2024. Pengelolaan tanaman cabai di Avicenna *Greenhouse* saat ini, sistem penyiraman harus menyambungkan kabel pompa secara manual agar air dapat mengalir ke tanaman. Penyiraman tanaman dilakukan satu hingga maksimal dua kali dalam sehari, yaitu pada pukul 07.00–08.00 pagi dan 16.00–17.00 sore. Budidaya tanaman cabai dilakukan dalam sekitar 165 *polybag* yang disusun di atas lahan berukuran kurang lebih 10 x 12 meter. Masa panen tanaman cabai di Avicenna *Greenhouse* memerlukan waktu sekitar empat bulan sejak masa tanam.

Meskipun telah menerapkan sistem irigasi otomatis sederhana, proses penyiraman di Avicenna *Greenhouse* masih belum sepenuhnya praktis karena pengguna tetap harus menghubungkan kabel pompa secara manual setiap kali ingin menyiram tanaman. Hal ini membuat pengelolaan penyiraman menjadi kurang efisien dan masih bergantung pada kehadiran manusia. Selain itu, Avicenna *Greenhouse* belum memiliki sistem pengkabutan air untuk menjaga

suhu dan kelembapan udara serta di dalam ruangan. Cara kerja yang masih mengandalkan aktivitas manual ini berisiko menyebabkan ketidakpastian dalam perawatan tanaman dan berpotensi mempengaruhi hasil panen. Berdasarkan kondisi tersebut, diperlukan pengembangan sistem otomatis berbasis *Internet of Things* yang dapat mengatur penyiraman dan kelembapan udara secara otomatis, agar pengelolaan tanaman di Avicenna *Greenhouse* menjadi lebih efektif dan efisien.

Beberapa penelitian terdahulu dengan topik yang sama telah dilakukan oleh beberapa peneliti, seperti penelitian yang dilakukan oleh (Alamsyah et al., 2024), Mengungkapkan bahwa di Indonesia, banyak petani masih mengandalkan metode manual sehingga proses penyiraman belum optimal. Penelitian Oleh (M. Iqbal Hasani & Sri Wulandari, 2023), Mengungkapkan bahwa penyiraman tanaman umumnya masih dilakukan secara manual oleh tenaga manusia menggunakan peralatan sederhana seperti gayung, selang dan ember yang memerlukan waktu dan usaha yang signifikan. Penelitian Oleh (Muhamad Rusdi et al., 2023), mengungkapkan bahwa permasalahan mitra yaitu masih menggunakan cara konvensional dalam proses pemeliharaan tanaman. Proses penyiraman yang berlebihan dapat menyebabkan adanya pembusukan pada tanaman anggrek dan tidak adanya teknologi untuk memantau kelembapan dan suhu pada area budidaya. Permasalahan ini menunjukkan adanya kebutuhan akan solusi teknologi yang mampu meningkatkan efisiensi dan efektivitas dalam kegiatan budidaya tanaman. Dibandingkan dengan penelitian-penelitian terdahulu, penelitian ini memiliki beberapa kelebihan dan kebaruan. Sistem yang dikembangkan tidak hanya mengotomatisasi proses penyiraman dan pengkabutan, tetapi juga menggunakan algoritma *Naïve bayes* sebagai dasar pengambilan data sensor untuk melakukan prediksi berdasarkan sensor suhu, kelembapan udara, dan kelembapan tanah secara *real-time*. Selain itu, sistem ini dikembangkan menggunakan *MQTT* untuk komunikasi data antar perangkat, serta *framework Flask* untuk pengembangan aplikasi dan juga mengintegrasikan dengan *Whatsapp* untuk mengirimkan notifikasi kepada pengguna. Sistem ini dibangun dengan metode pengembangan *Agile*, dimana metode ini memiliki penyesuaian dinamis dalam setiap tahap pengembangan dan jika terdapat perubahan pada tahapan akan

fleksibel tidak harus mengulang ke tahapan awal. Algoritma *Naive Bayes* digunakan karena sesuai dengan penelitian yang akan dilakukan untuk melakukan klasifikasi dalam mengolah data dari sensor, sehingga dapat membantu dalam pengambilan keputusan seperti menentukan kebutuhan penyiraman dan pengkabutan. Oleh karena itu, penelitian ini difokuskan pada implementasi *Internet of Things* yang dapat mengotomatisasi proses penyiraman dan pengkabutan tanaman di dalam *Greenhouse*.

Solusi yang ditawarkan dalam penelitian ini adalah mengimplementasikan *Internet of Things* yang dapat mengotomatiskan proses penyiraman dan pengkabutan di Avicenna *Greenhouse* melalui pengembangan aplikasi dengan penerapan algoritma *Naïve bayes* sebagai prediksi suhu ruangan, kelembapan tanah pada tanaman, kelembapan udara pada ruangan. Sistem ini dikembangkan menggunakan metode *Agile* yang terdiri dari perencanaan, perancangan, pengembangan, pengujian, penerapan, peluncuran. Dalam proses perancangannya, aplikasi ini menggunakan sensor suhu, kelembapan udara dan kelembapan tanah. Data dari sensor tersebut dapat dilakukan prediksi untuk penyiraman dan pengkabutan pada tanaman supaya efektif menggunakan algoritma *Naïve bayes*. Aplikasi dikembangkan menggunakan bahasa pemrograman *Arduino*, *Flask Python*. Hasil dari penelitian ini berupa sebuah aplikasi web yang menyediakan *monitoring* suhu ruangan, kelembapan udara pada ruangan, kelembapan tanah pada pot tanaman, mengendalikan penyiraman dan pengkabutan secara otomatis. Manfaat yang diharapkan adalah peningkatan efisiensi penggunaan air, stabilitas kondisi lingkungan dalam *Greenhouse*, serta peningkatan produktivitas dan kualitas pertumbuhan tanaman. Dengan demikian penelitian ini di harapkan dapat memberikan kemudahan untuk pemilik dalam melakukan penyiraman dan pengkabutan otomatis di *Greenhouse*. Kesimpulannya, penelitian ini bertujuan untuk merancang dan membangun aplikasi berbasis *Internet of Things* yang dapat membantu pemilik atau petani dalam melakukan penyiraman dan pengkabutan secara otomatis, dengan judul “IMPLEMENTASI INTERNET OF THINGS UNTUK PENYIRAMAN DAN PENGKABUTAN OTOMATIS PADA TANAMAN MENGGUNAKAN ALGORITMA NAÏVE BAYES (Studi Kasus Di Avicenna *Greenhouse*)”.

1.2 Rumusan Masalah

Berdasarkan uraian pada latar belakang, maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana mengimplementasikan *IOT* untuk penyiraman dan pengkabutan otomatis?
2. Bagaimana penerapan algoritma *Naïve bayes* dapat membantu dalam pengambilan keputusan otomatis berdasarkan data sensor?
3. Bagaimana membangun aplikasi *monitoring* berbasis web yang dapat membantu pengguna memantau kondisi lingkungan *Greenhouse* secara *real-time* serta mengendalikan sistem penyiraman dan pengkabutan secara otomatis?

1.3 Batasan Masalah

Adapun batasan masalah terhadap penelitian yang diajukan adalah sebagai berikut:

1. Penelitian ini berfokus pada pengembangan aplikasi penyiraman dan pengkabutan otomatis berbasis *IOT* untuk tanaman cabai di Avicenna *Greenhouse*.
2. Aplikasi yang dikembangkan memanfaatkan sensor kelembapan tanah, suhu, dan kelembapan udara sebagai parameter utama dalam proses penyiraman dan pengkabutan.
3. Aplikasi ini menggunakan algoritma *Naïve bayes* dalam pengambilan keputusan untuk menentukan kapan penyiraman dan pengkabutan dilakukan berdasarkan data sensor sebagai notifikasi.
4. Pengembangan aplikasi mencakup integrasi dengan platform berbasis web untuk *monitoring* dan pengendalian sistem secara *real-time*, tetapi tidak mencakup fitur lanjutan seperti rekomendasi pemupukan atau analisis pertumbuhan tanaman.
5. Penelitian ini menggunakan metode *Agile* dalam pengembangan aplikasi guna meningkatkan fleksibilitas dalam implementasi dan evaluasi.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah dan batasan masalah diatas, maka tujuan penelitian ini adalah sebagai berikut:

1. Mengimplementasikan *IOT* untuk penyiraman dan pengkabutan otomatis pada *Greenhouse*.
2. Menerapkan algoritma *Naïve bayes* untuk membantu proses pengambilan keputusan berdasarkan data sensor.
3. Merancang dan membangun aplikasi *monitoring* berbasis web yang memungkinkan pengguna memantau kondisi *Greenhouse* secara *real-time* dan mengendalikan sistem penyiraman dan pengkabutan secara otomatis.

1.5 Metodologi Penelitian

1. Metode Pengumpulan Data

Beberapa Metode pengumpulan data digunakan untuk mendapatkan data yang akurat yaitu sebagai berikut:

a. Observasi

Metode ini dilakukan dengan mengamati secara langsung kondisi dan kegiatan di lokasi penelitian, yaitu Avicenna *Greenhouse* tempat tanaman cabai dibudidayakan. Observasi dilakukan untuk memahami proses penyiraman dan pengkabutan yang saat ini berjalan, serta kendala-kendala yang dihadapi dalam pengelolaan irigasi.

b. Wawancara

Wawancara dilakukan dengan pihak-pihak yang terlibat langsung dalam pengelolaan *Greenhouse*, seperti pemilik *Greenhouse* atau petani.

c. Studi Pustaka

Studi pustaka dilakukan dengan mengumpulkan dan mempelajari literatur dari berbagai sumber seperti jurnal, artikel ilmiah, buku, dan dokumentasi terkait *IOT*, penyiraman otomatis, pengkabutan, serta algoritma *Naïve bayes* dan sumber lain yang bersangkutan dengan topik penelitian. Melalui studi pustaka, peneliti memperoleh landasan teoritis dan pemahaman terhadap teknologi dan metode yang relevan.

2. Metode Pengembangan Sistem

Metode pengembangan perangkat lunak yang digunakan dalam penelitian ini adalah metode *Agile*. *Agile* merupakan metode yang mengandalkan proses berulang dan bertahap dalam siklus pengembangan perangkat lunak atau *Software Development Life Cycle (SDLC)*. Tujuannya adalah untuk menciptakan perangkat lunak yang dapat berkembang secara fleksibel dan menyesuaikan diri dengan perubahan kebutuhan di tengah proses pengembangan. Dalam metode *Agile*, seluruh proses pengembangan dibagi menjadi bagian-bagian kecil yang disebut iterasi atau *sprint*. Setiap *sprint* berlangsung dalam waktu yang relatif singkat dan mencakup beberapa tahapan penting, yaitu perencanaan, perancangan, pengembangan, pengujian, penerapan, peluncuran.

1.6 Sistematika Penulisan

Dalam menyusun laporan skripsi ini diatur dan disusun dalam enam bab, yang masing-masing terdiri dari beberapa sub bab. Adapun urutannya sebagai berikut:

BAB I PENDAHULUAN

Bagian ini berisi mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, metodologi penelitian dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bagian ini menjelaskan tentang landasan teori dan dasar teori pendukung dalam penelitian. Tinjauan Pustaka ini bersumber dari buku, jurnal dan *Website*.

BAB III: METODOLOGI PENELITIAN

Bab ini menjelaskan metodologi penelitian yang dipakai pada tahap-tahap penulis dalam melakukan penelitian di Avicenna *Greenhouse*.

BAB IV: ANALISIS DAN PERANCANGAN

Bab ini berisi tentang analisis, perancangan *Hardware* dan perancangan perangkat lunak, perancangan *database*, perancangan antarmuka serta penjelasan tentang perancangan perangkat lunak yang akan di bangun.

BAB V IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi penyajian tahap pengembangan aplikasi yang akan dijelaskan tiap langkahnya dan menampilkan tampilan dari setiap fitur dari aplikasi yang dibuat serta penjelasan bagaimana aplikasi berjalan.

BAB VI PENUTUP

Bab ini berisi kesimpulan dan penyajian tahap pembuatan yang dilakukan serta saran untuk implementasi *Internet of Things* selanjutnya.

BAB II

TINJAUAN PUSTAKA

2.1 Landasan Teori

Landasan teori adalah kumpulan teori yang berkaitan dengan penelitian yang akan dilakukan. Dalam penyusunan penelitian, merujuk pada penelitian sebelumnya sebagai referensi. Tujuan dari hal ini adalah untuk membandingkan kesamaan serta perbedaan antara penelitian yang sedang dilakukan dengan penelitian terdahulu.

Tabel 2. 1 Acuan Jurnal Penelitian

| No | Jurnal Penelitian | Masalah | Metode | Kesimpulan |
|----|---|--|--|---|
| 1 | Judul: Sistem Penyiraman Tanaman Otomatis Menggunakan Logika Fuzzy Dengan Teknologi Internet of Things Berbasis ESP8266 Dan Aplikasi Blynk Penulis: Ridho Alamsyah, Eddy Ratna Mufidah Ryansyah, Andari Yasinta Permana. | Di Indonesia, banyak petani masih mengandalkan metode manual sehingga proses penyiraman belum optimal. | Penelitian ini menggunakan metode studi literatur, perancangan perangkat, logika fuzzy yaitu Dalam penelitian ini dihasilkan keputusan menggunakan <i>fuzzy logic</i> yang terdiri dari beberapa tahap yaitu fuzzifikasi, mesin inferensi, dan defuzzifikasi. Pada <i>output</i> yang dihasilkan adalah keputusan akhir apakah pompa air otomatis akan | Penelitian ini memanfaatkan sensor kelembapan tanah untuk mendeteksi tingkat kelembapan tanam media tanam secara otomatis. Fokus utama dari sistem yang dikembangkan adalah otomasi penyiraman tanaman tanpa melibatkan aspek penerapan lainnya. Sistem ini juga terintegrasi dengan aplikasi |

| | | | | |
|---|--|--|--|--|
| | Tahun: 2024 | | menyala atau tidak dan pengujian perangkat. | <i>Blynk</i> yang berfungsi sebagai antarmuka untuk memantau serta mengendalikan proses penyiraman secara <i>real-time</i> melalui perangkat <i>mobile</i> . |
| 2 | <p>Judul: Implementasi <i>Internet of Things (IOT)</i> Pada Sistem Otomatisasi Penyiraman Tanaman Berbasis <i>Mobile</i></p> <p>Penulis: M. Iqbal Hasani, Sri Wulandari.</p> <p>Tahun: 2023</p> | <p>Penyiraman tanaman umumnya masih dilakukan secara manual oleh tenaga manusia menggunakan peralatan sederhana seperti gayung, selang dan ember yang memerlukan waktu dan usaha yang signifikan</p> | <p>Penelitian ini mengimplementasikan algoritma <i>Naïve bayes</i> digunakan untuk menganalisis data yang diakuisisi dari sensor dan menentukan tindakan yang harus dilakukan oleh sistem.</p> <p>Lalu melibatkan penerapan metodologi <i>System Development Life Cycle (SDLC)</i>, yang terdiri dari beberapa tahap, yaitu Analisis Kebutuhan, Desain Sistem, Implementasi, Pengujian, Peluncuran dan Pemeliharaan.</p> | <p>Sistem ini memanfaatkan sensor kelembapan tanah untuk mendeteksi tingkat kelembapan, serta sensor suhu untuk memantau suhu sekitar tanaman.</p> <p>Penelitian ini berfokus pada otomasi penyiraman tanaman tanpa mempertimbangkan aspek <i>IOT</i> lainnya. Aplikasi <i>Blynk</i> digunakan untuk memantau dan mengendalikan proses penyiraman secara <i>real time</i>. Selain itu, penerapan</p> |

| | | | | |
|---|---|---|--|---|
| | | | | algoritma <i>Naïve bayes</i> terbukti mampu mendukung pengambilan keputusan berdasarkan data sensor secara akurat. |
| 3 | Judul: Implementasi Teknologi Penyiraman Sistem Pengkabutan Otomatis dan <i>Monitoring</i> Pintar Berbasis Tenaga Surya untuk Tempat Budidaya Tanaman Anggrek UD fairus Di Kabupaten Merauke. Penulis: Muhamad Rusdi, Muriani, Rivaldo Pasca Corpatty, Mardiyasa Putra Yoga, Grace Christin Aditya | Permasalahan mitra yaitu masih menggunakan cara konvensional dalam proses pemeliharaan tanaman anggrek dari tahapan penyiraman tanaman anggrek secara langsung menggunakan selang hingga pemberian pupuk. | Metode pelaksanaan yang digunakan pada pengabdian ini terdiri dari beberapa tahapan yaitu: diskusi dan observasi, Sosialisasi Program kegiatan dan teknologi, <i>workshop</i> pengoperasian dan pemeliharaan teknologi, evaluasi kegiatan, publikasi dan capaian luaran kegiatan | Penelitian ini memanfaatkan sensor kelembapan tanah untuk mendekripsi tingkat kelembapan lingkungan serta sensor suhu untuk memantau suhu di sekitar tanaman. Sistem yang dikembangkan berfokus pada otomatisasi proses penyiraman dan pengkabutan tanaman secara efisien. Selain itu, aplikasi <i>Blynk</i> digunakan sebagai antarmuka pemantauan dan pengendalian sistem secara <i>real-time</i> , |

| | | | | |
|--|--|--|--|--|
| | Ronsumbre, Diah Bayu Titisari. Tahun: 2023 | budidaya. Selain itu, meningkatnya biaya operasional pemeliharaan tanaman anggrek yang disebabkan penggunaan mesin penyiraman berbahan bakar bensin. | | sehingga memungkinkan pengguna untuk mengatur penyiraman dan pengkabutan secara jarak jauh melalui perangkat <i>mobile</i> . |
|--|--|--|--|--|

Beberapa penelitian terdahulu menunjukkan potensi besar penerapan *Internet of Things (IOT)* dalam otomasi penyiraman dan pengkabutan tanaman. Penelitian oleh (Alamsyah et al., 2024) mengembangkan sistem penyiraman otomatis berbasis *ESP8266* dan aplikasi *Blynk*. Sistem tersebut menggunakan sensor kelembapan tanah untuk mengukur tingkat kelembapan media tanam dan mengaktifkan penyiraman secara otomatis, yang dapat dipantau secara *real-time* melalui perangkat *mobile*. Penelitian oleh (M. Iqbal Hasani & Sri Wulandari, 2023) juga memanfaatkan sensor kelembapan tanah dan suhu, dengan dukungan algoritma *Naïve bayes* untuk mendukung pengambilan keputusan penyiraman berdasarkan data sensor. Sistem ini diintegrasikan dengan aplikasi *Blynk* untuk pemantauan dan kendali secara jarak jauh. Sementara itu, penelitian oleh (Muhamad Rusdi et al., 2023) mengembangkan sistem penyiraman dan pengkabutan otomatis berbasis tenaga surya yang diterapkan pada budidaya anggrek. Sistem ini menggabungkan sensor kelembapan dan suhu serta aplikasi *Blynk* sebagai *interface* untuk memantau dan mengontrol penyiraman dan pengkabutan secara efisien.

Ketiga penelitian tersebut memiliki kesamaan dalam penggunaan teknologi *IOT* untuk otomatisasi proses penyiraman tanaman, serta penggunaan sensor kelembapan dan suhu sebagai indikator utama dalam pengambilan keputusan. Namun, berbeda dengan penelitian-penelitian tersebut, penelitian ini tidak hanya mengotomatiskan proses penyiraman, tetapi juga mengintegrasikan sistem pengkabutan tanaman dalam satu platform yang saling terhubung. Selain itu, sistem yang dikembangkan dalam penelitian ini menggunakan antarmuka berbasis web mandiri yang dibangun dengan *framework Flask*, bukan mengandalkan aplikasi pihak ketiga, sehingga memberikan fleksibilitas dan kontrol penuh terhadap fungsionalitas sistem. Penggunaan protokol *MQTT* dan integrasi notifikasi *Whatsapp* juga menjadi pembeda utama, yang tidak ditemukan pada tiga penelitian terdahulu.

Namun, terdapat beberapa keterbatasan pada studi sebelumnya. Pertama, fokus sebagian besar penelitian hanya terbatas pada proses penyiraman, tanpa mengintegrasikan sistem pengkabutan sebagai salah satu elemen penting dalam pengelolaan suhu dan kelembapan udara pada *Greenhouse*. Kedua, sistem yang dikembangkan sepenuhnya bergantung pada aplikasi pihak ketiga seperti *Blynk*, sehingga menimbulkan ketergantungan eksternal dan keterbatasan dalam fleksibilitas desain sistem. Ketiga, tidak ada integrasi terhadap notifikasi instan yang langsung terhubung ke pengguna seperti melalui *Whatsapp*, sehingga interaksi pengguna terhadap sistem cenderung pasif.

Penelitian ini menawarkan kelebihan dan kebaruan dengan merancang dan membangun sistem otomatisasi penyiraman dan pengkabutan tanaman cabai berbasis web tanpa ketergantungan pada aplikasi pihak ketiga. Sistem dikembangkan menggunakan *framework Flask (Python)* yang terintegrasi langsung dengan perangkat *IOT* melalui protokol *MQTT* untuk komunikasi data secara *real-time*. Algoritma *Naïve bayes* diterapkan sebagai sistem pengambilan keputusan berbasis data sensor suhu, kelembapan udara, dan kelembapan tanah untuk prediksi penyiraman dan pengkabutan. Selain itu, sistem ini juga menambahkan notifikasi langsung melalui aplikasi *Whatsapp* untuk memberikan informasi hasil prediksi dari *navie bayes* dan status aksi sistem kepada pengguna. Proses pengembangan sistem menggunakan metode *Agile*, memungkinkan

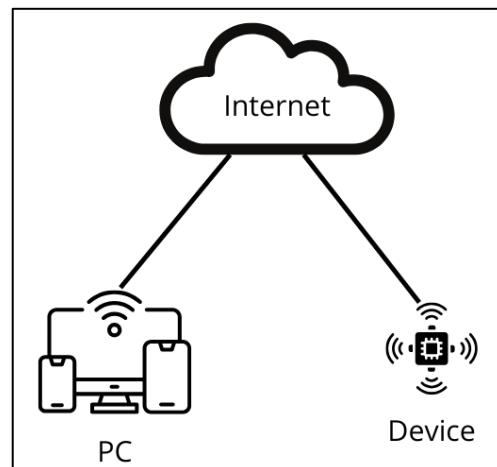
penyesuaian dinamis dalam setiap tahap pengembangan. Dengan pendekatan ini, penelitian ini memberikan kontribusi inovatif terhadap sistem otomasi pertanian berbasis *IOT* yang lebih efisien, fleksibel, dan responsif terhadap kebutuhan pengguna.

2.2 Dasar Teori

Adapun teori-teori yang ada sebagai acuan dalam implementasi *Internet of Things* yaitu sebagai berikut:

2.2.1 *Internet of Things*

Internet of Things didefinisikan kemampuan perangkat yang dapat berkomunikasi, terhubung, dan bertukar data melalui Internet. *Internet of Things* adalah teknologi yang memungkinkan adanya kontrol, komunikasi, kerja sama dengan berbagai perangkat keras, data di jaringan internet. (Ifa Susuek Anselmus Talli et al., 2023).



Gambar 2. 1 Konsep *Internet of Things*

Sumber: (Kelasplc, 2023)

Gambar 2.1 menunjukkan hubungan antara perangkat *PC* dan *device* (perangkat dengan sensor) yang terkoneksi ke internet. Perangkat-perangkat tersebut mengirimkan dan menerima data melalui jaringan internet, sehingga

memungkinkan terjadinya integrasi sistem, otomatisasi proses, dan pengambilan keputusan berbasis data secara *real-time*.

2.2.2 Penyiraman Tanaman

Menyiram tanaman berarti memberikan air secara langsung ke tanaman. Tujuannya adalah memasok air dan nutrisi yang dibutuhkan akar agar diserap dan didistribusikan ke seluruh bagian tanaman. Menyiram tanaman dilakukan dengan berbagai cara, seperti menuangkan atau meneteskan air di dekat perakaran, mengguyur pot dengan air, hingga menggunakan sistem irigasi otomatis (Kurniawan, 2023).

Tabel 2. 2 Pengujian Kelembapan Tanah

Sumber: (Ifa Susuek Anselmus Talli et al., 2023)

| No | Kondisi Tanah | <i>Sensor Soil Moisture</i> | Alat Ukur |
|----|-------------------------|-----------------------------|---------------|
| 1 | Tanah Berpupuk | 66% | Basah |
| 2 | Tanah Berpasir | 40% | Sangat Kering |
| 3 | Tanah Biasa (Liat) | 62% | Basah |
| 4 | Tanah Campur Kopi | 70% | Basah |
| 5 | Tanah Campur Garam | 61% | Basah |
| 6 | Tanah Disiram Air Teh | 60% | Basah |
| 7 | Tanah Disiram Air Sabun | 65% | Basah |
| 8 | Tanah ditaburkan kapur | 66% | Basah |
| 9 | Tanah ditaburkan micin | 65% | Basah |
| 10 | Tanah ditaburkan sekam | 64% | Basah |

Berdasarkan data dari Tabel 2.2 Pengujian Kelembapan Tanah, nilai kelembapan tanah bervariasi tergantung pada jenis dan perlakuan terhadap media tanam. Dari sepuluh sampel pengujian yang dilakukan, sebagian besar menunjukkan nilai kelembapan antara 60% hingga 70%, yang dikategorikan dalam kondisi "basah" menurut alat ukur kelembapan.

Untuk tanaman cabai, kelembapan tanah yang ideal berada dalam kisaran 60% hingga 80%, sebagaimana dijelaskan dalam penelitian oleh (S Nursuwars & Sujana, 2018). Kisaran ini penting karena sistem perakaran tanaman cabai memerlukan kelembapan yang cukup untuk menyerap air dan nutrisi secara optimal, tetapi tidak terlalu jenuh agar tidak menyebabkan pembusukan akar. Kelembapan di bawah 60% umumnya menunjukkan kondisi yang terlalu kering, seperti pada tanah berpasir dengan kelembapan 40%, yang dinyatakan sebagai "sangat kering". Tanah dengan kondisi ini berisiko menghambat pertumbuhan tanaman karena suplai air tidak mencukupi untuk proses fotosintesis dan penyerapan nutrisi. Sebaliknya, menjaga kelembapan tanah dalam kisaran 60% hingga 70% seperti pada tanah berpupuk, tanah liat, atau tanah yang dicampur dengan bahan organik (kopi, sekam) dapat memberikan kondisi yang stabil dan mendukung pertumbuhan tanaman cabai secara optimal.

2.2.3 Pengkabutan Tanaman

Misting atau pengkabutan merupakan salah satu teknik penting dalam merawat tanaman, terutama pada tanaman *hortikultura* dan tanaman hias yang sensitif terhadap perubahan suhu dan kelembapan. Teknik ini dilakukan dengan menyemprotkan butiran air sangat halus ke udara di sekitar tanaman. Proses ini memanfaatkan prinsip pendinginan *evaporatif*, yaitu saat butiran air menguap, panas dari udara sekitar diserap sehingga menyebabkan penurunan suhu lokal secara efektif. (Kurniawan, 2023).

Tabel 2. 3 Pengujian DHT11

Sumber: (Fahmi et al., 2022)

| No | Waktu | Suhu | Kelembapan Udara |
|----|---------------------------|---------|------------------|
| 1 | 2022-08-02T07:58:49+00:00 | 22,9°C | 77% |
| 2 | 2022-08-02T07:59:19+00:00 | 22,4 °C | 78% |
| 3 | 2022-08-02T07:00:22+00:00 | 26,8 °C | 71% |
| 4 | 2022-08-02T07:00:44+00:00 | 30,4 °C | 64% |

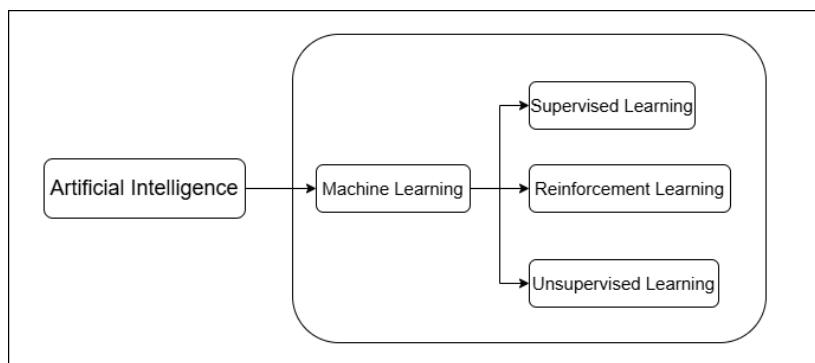
| | | | |
|----|---------------------------|---------|-----|
| 5 | 2022-08-02T07:01:31+00:00 | 23,9 °C | 71% |
| 6 | 2022-08-02T07:01:53+00:00 | 21,3 °C | 92% |
| 7 | 2022-08-02T07:02:17+00:00 | 23,4 °C | 77% |
| 8 | 2022-08-02T07:02:33+00:00 | 27,1 °C | 71% |
| 9 | 2022-08-02T07:02:49+00:00 | 27,8 °C | 70% |
| 10 | 2022-08-02T07:02:52+00:00 | 27,2 °C | 70% |
| 11 | 2022-08-02T07:03:10+00:00 | 27,2 °C | 70% |
| 12 | 2022-08-02T07:03:25+00:00 | 19,3 °C | 96% |
| 13 | 2022-08-02T07:03:50+00:00 | 23,3 °C | 67% |

Berdasarkan hasil uji coba terhadap sensor DHT11 yang dilakukan di Greenhouse budidaya tanaman cabai, diperoleh data suhu dan kelembapan udara yang bervariasi. Hasil pengujian sebagaimana tercantum pada Tabel 2.3 menunjukkan bahwa sensor mampu mendeteksi suhu dalam rentang 19,3°C hingga 30,4°C dan kelembapan udara antara 64% hingga 96%. Nilai-nilai ini memperlihatkan dinamika kondisi mikroklimat dalam Greenhouse selama periode pengamatan. Salah satu titik pembacaan menunjukkan suhu sebesar 27,3°C dan kelembapan udara 71%, yang dapat dikategorikan sebagai kondisi optimal untuk pertumbuhan tanaman cabai. Data ini juga menjadi acuan dalam menilai

keandalan sensor dalam membaca kondisi lingkungan. Sensor DHT11 cukup layak digunakan untuk mendukung pengaktifan sistem pengkabutan berdasarkan data suhu dan kelembapan udara. Pengkabutan akan menyala secara otomatis ketika suhu melebihi 27°C dan kelembapan udara turun di bawah 71%. Hal ini karena pada kondisi tersebut, udara menjadi lebih kering dan tanaman mengeluarkan lebih banyak air melalui daunnya. Dengan adanya sistem pengkabutan, kelembapan udara di sekitar tanaman bisa dijaga, dan suhu di dalam Greenhouse bisa menjadi lebih sejuk secara alami. Dengan begitu, tanaman tetap berada dalam kondisi yang baik untuk tumbuh. Data dari sensor digunakan sebagai dasar untuk menghidupkan pengkabutan secara otomatis jika suhu dan kelembapan sudah mencapai batas yang bisa memengaruhi pertumbuhan tanaman cabai.

2.2.4 Machine Learning

Machine Learning (ML) merupakan cabang kecerdasan buatan yang memungkinkan sistem belajar dari data dan membuat keputusan atau prediksi tanpa diprogram secara eksplisit. *ML* merupakan bidang studi yang fokus kepada desain dan analisis algoritma sehingga memungkinkan komputer untuk dapat belajar. Menurut Samuel, *Machine Learning* berisi sebuah algoritma yang bersifat generik atau umum di mana algoritma tersebut dapat menghasilkan sesuatu yang menarik atau bermanfaat dari sejumlah data tanpa harus menulis kode yang spesifik. (Ibnu Daqiqil Id, 2021).



Gambar 2. 2 Skema *Artificial Intelligence* dan *Machine Learning*

Sumber: (Roihan et al., 2019)

Gambar di atas menjelaskan bahwa *Machine Learning* merupakan salah satu bagian dari *Artificial Intelligence*. Di dalam *Machine Learning*, terdapat tiga pendekatan utama, yaitu *Supervised Learning*, *Unsupervised Learning*, dan *Reinforcement Learning*. *Supervised Learning* menggunakan dataset berlabel, di mana model dilatih untuk memetakan *input* ke *output* berdasarkan data yang telah diketahui hasilnya. *Unsupervised Learning* bekerja tanpa label dan bertujuan menemukan struktur atau pola tersembunyi dalam data. Sedangkan *Reinforcement Learning* menggunakan pendekatan *trial-and-error* dengan sistem penghargaan untuk menemukan strategi optimal dalam pengambilan keputusan. Pada penelitian ini, metode *Supervised Learning* digunakan karena data yang digunakan berupa dataset dengan label yang telah ditentukan sebelumnya, sehingga model dapat dilatih untuk melakukan klasifikasi atau prediksi berdasarkan data tersebut secara

akurat. Ketiga pendekatan tersebut memiliki keunggulan masing-masing tergantung pada karakteristik data dan tujuan analisis yang diinginkan. Dengan memilih pendekatan yang tepat, proses pembelajaran mesin dapat memberikan hasil yang lebih optimal dan relevan terhadap kebutuhan penelitian.

2.2.5 Algoritma *Naïve bayes*

Naive Bayes merupakan salah satu algoritma klasifikasi dalam *Machine Learning* yang didasarkan pada *Teorema Bayes*, yaitu sebuah metode probabilistik yang digunakan untuk memprediksi kelas dari suatu data berdasarkan probabilitas sebelumnya (*prior probability*) dan informasi baru (*posterior probability*). Meskipun teori ini pertama kali dikenalkan oleh Thomas Bayes pada abad ke-18, penerapannya dalam bidang ilmu komputer dan pembelajaran mesin menjadi sangat signifikan dalam beberapa dekade terakhir. Algoritma ini bekerja dengan mengasumsikan bahwa fitur-fitur pada dataset saling bebas secara kondisi (*naive assumption*), yang menjadikannya sangat sederhana namun efektif untuk berbagai tugas klasifikasi seperti *spam filtering*, *sentiment analysis*, dan diagnosis medis (Ibnu Daqiqil Id, 2021). Pada dasarnya Algoritma *Naive Bayes* memiliki beberapa varian yang digunakan sesuai dengan tipe data yang dianalisis:

- a. *Multinomial Naive Bayes*: Digunakan untuk data diskrit berupa jumlah frekuensi kemunculan fitur tertentu. Umumnya digunakan untuk klasifikasi teks, seperti *document classification* atau *spam filtering*.
- b. *Bernoulli Naive Bayes*: Cocok untuk data biner, yaitu fitur yang hanya memiliki dua nilai: 0 dan 1.
- c. *Gaussian Naive Bayes*: Digunakan untuk data numerik atau *kontinu* yang diasumsikan mengikuti distribusi *Gaussian* (normal).

Pada penelitian ini, varian yang digunakan adalah *Gaussian Naïve bayes*, yang merupakan salah satu tipe *Naïve bayes*. Prinsip dasarnya sama dengan *Naïve bayes*, namun pada *Naïve bayes* kita tidak dapat menghitung probabilitas data *continues*. Oleh karena itu kita menggunakan distribusi *gaussian* untuk menghitung probabilitas dengan asumsi data terdistribusi normal. Adapun *pdf* (*probability density function*) dari *gaussian distribution* adalah:

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Sumber: (Ibnu Daqiqil Id, 2021)

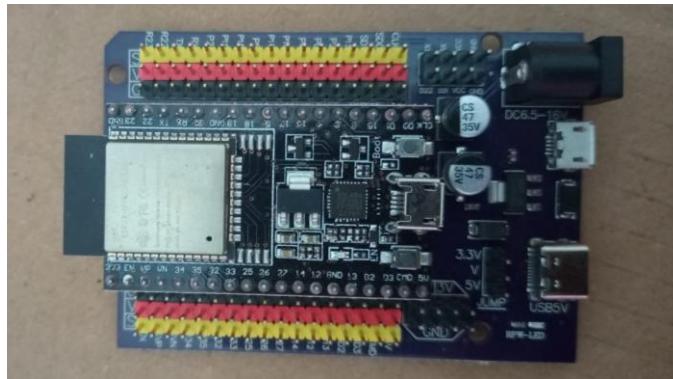
Rumus di atas digunakan untuk menghitung probabilitas suatu nilai fitur x_i yang termasuk dalam kelas y . Di mana μ_y adalah rata-rata (*mean*) dan σ_y^2 adalah varians dari fitur tersebut untuk kelas y . Fungsi ini menggambarkan seberapa besar kemungkinan nilai x_i muncul dalam distribusi *Gaussian* kelas tertentu.

Metode *Gaussian Naïve bayes* cocok diterapkan pada data yang bersifat *kontinu* dan mendekati distribusi normal. Pada penelitian ini, metode ini digunakan untuk mengklasifikasikan kondisi lingkungan di dalam *Greenhouse* berdasarkan tiga parameter sensor utama, yaitu suhu, kelembapan udara, dan kelembapan tanah. Setiap parameter sensor diklasifikasikan untuk menentukan apakah kondisi tersebut memerlukan penyiraman, pengkabutan, atau tidak perlu tindakan.

Sebagai contoh, sistem mengumpulkan data dari sensor secara *real-time*, lalu menghitung nilai probabilitas dari setiap parameter sensor terhadap kelas tindakan menggunakan rumus di atas. Setelah diperoleh nilai *mean* dan *varians* untuk masing-masing kelas tindakan berdasarkan data pelatihan, sistem dapat menentukan kelas mana yang memiliki probabilitas terbesar untuk kondisi saat itu. Misalnya, jika suhu tinggi dan kelembapan rendah, hasil klasifikasi kemungkinan besar akan menunjuk pada tindakan pengkabutan.

2.2.6 *ESP32*

ESP32 adalah nama dari mikrokontroler yang dirancang oleh perusahaan teknologi berbasis di Shanghai, Tiongkok, yaitu *Espressif Systems*. Mikrokontroler ini merupakan penerus dari seri sebelumnya, yaitu *ESP8266*, dengan peningkatan performa dan fitur yang lebih lengkap. *ESP32* menawarkan solusi jaringan *WiFi* dan *Bluetooth* yang mandiri sebagai jembatan antara sistem mikrokontroler dengan jaringan nirkabel (Kusumah & Pradana, 2019).



Gambar 2. 3 *ESP32*

Sumber: (Penulis, 2025)

Pada gambar di atas, *ESP32* terpasang pada sebuah *development board* yang telah dilengkapi dengan berbagai pin *header* untuk kemudahan koneksi ke komponen eksternal. Modul *ESP32* terletak di sisi kiri atas papan (ditandai dengan chip berwarna logam), yang merupakan inti dari sistem. *Board* ini menyediakan koneksi ke pin digital dan analog yang ditandai dengan label seperti D0, D1, dan seterusnya, serta port daya seperti 3.3V dan 5V untuk menyuplai arus ke sensor. Modul juga dilengkapi dengan port USB Type-C dan Micro USB untuk pemrograman dan komunikasi data, serta input catu daya eksternal (DC 6.5–16V) yang memungkinkan perangkat bekerja secara independen tanpa koneksi komputer. Keberadaan komponen tambahan seperti regulator tegangan, kapasitor, dan LED indikator juga berperan penting dalam memastikan kestabilan sistem.

Dalam penelitian ini, *ESP32* digunakan sebagai pusat kendali sistem *IOT* yang bertugas menerima data dari sensor suhu, kelembapan udara, dan kelembapan tanah. Data yang diterima akan diolah secara lokal, kemudian *ESP32* mengambil keputusan untuk mengaktifkan aktuator seperti pompa air (untuk penyiraman) atau *misting* (pengkabutan), serta mengirimkan data ke *broker MQTT* melalui jaringan *WiFi*.

Dalam penelitian ini, *ESP32* dipasang menggunakan *expansion shield plate* untuk memudahkan koneksi pin dan pemasangan sensor secara modular. Penggunaan *shield* juga membuat sistem lebih rapi dan stabil dalam pengembangan. Beberapa pin seperti GPIO26 dan GPIO27 digunakan untuk menghubungkan sensor dan aktuator secara langsung ke papan ekspansi.

2.2.7 Soil Moisture Sensor

Soil Moisture Sensor adalah sensor kelembapan yang dapat mendeteksi kelembapan dalam tanah. Sensor ini sangat sederhana, tetapi ideal untuk memantau taman kota, atau tingkat air pada tanaman pekarangan. Sensor ini terdiri dua *probe* untuk melewatkkan arus melalui tanah, kemudian membaca resistansinya untuk mendapatkan nilai tingkat kelembapan. Semakin banyak air membuat tanah lebih mudah menghantarkan listrik (resistansi kecil), sedangkan tanah yang kering sangat sulit menghantarkan listrik (resistansi besar) (Eka Candra & Maulana Universitas Putera Batam, 2019).



Gambar 2. 4 *Soil Mastuire Sensor*

Sumber: (Penulis, 2025)

Pada gambar di atas, terlihat bahwa sensor memiliki dua buah batang *probe* berbahan logam yang dilapisi pelindung, dengan masing-masing *probe* tersambung ke bagian rangkaian elektronik di bagian atas. Tulisan dalam karakter Mandarin yang tertera di papan menunjukkan bahwa sensor ini merupakan versi produksi dari pabrikan Cina. *Probe* tersebut didesain untuk ditancapkan langsung ke dalam tanah, dan sensor akan mendeteksi nilai resistansi berdasarkan kadar air yang menyentuh kedua elektroda. Sensor ini terhubung ke mikrokontroler seperti *ESP32* melalui pin analog. Pada praktiknya, nilai kelembapan dari sensor ini akan dibaca oleh mikrokontroler dan diubah menjadi data persentase kelembapan tanah. Data tersebut selanjutnya dapat digunakan untuk mengambil keputusan, seperti mengaktifkan sistem penyiraman otomatis ketika kelembapan tanah turun di bawah ambang batas yang telah ditentukan.

2.2.8 DHT11

Sensor *DHT11* adalah modul sensor yang mendeteksi suhu dan kelembapan udara. Modul sensor ini termasuk ke dalam elemen perangkat *resistive* seperti alat pengukur suhu NTC. Cara kerja sensor *DHT11* dengan mengukur perubahan resistansi yang disebabkan oleh suhu dan kelembapan udara, kemudian mengirimkan data sinyal digital yang terdiri dari pulsa tinggi dan pulsa rendah ke mikrokontroler yang mewakili nilai suhu dan kelembapan udara (Ifa Susuek Anselmus Talli et al., 2023).



Gambar 2. 5 *DHT11*

Sumber: (Penulis, 2025)

Pada gambar di atas terlihat modul *DHT11* yang memiliki bentuk fisik khas dengan bagian sensor berwarna biru dan kisi-kisi di bagian depan sebagai ventilasi untuk sirkulasi udara. Sensor ini dilengkapi dengan tiga pin konektor di sisi bawahnya yang digunakan untuk sambungan ke catu daya, *ground*, dan sinyal data. Bagian belakang sensor terhubung dengan kabel yang telah dilapisi pelindung hitam agar lebih aman dan rapi dalam instalasi. Modul *DHT11* ini biasa digunakan pada sistem monitoring berbasis mikrokontroler seperti *ESP32* untuk membaca kondisi suhu dan kelembapan lingkungan secara *real-time*. Sensor *DHT11* memiliki kelebihan berupa konsumsi daya rendah, ukuran kecil, dan kemudahan dalam integrasi ke berbagai proyek elektronik, terutama di bidang pertanian, sistem otomatisasi rumah, serta pemantauan lingkungan berbasis *IOT*. Nilai suhu dan kelembapan yang dihasilkan sensor dapat ditampilkan pada layar atau dikirim ke *server* atau aplikasi pemantauan jarak jauh melalui koneksi *WiFi*.

2.2.9 Nozzle Sprayer

Nozzle Sprayer adalah bagian dari alat semprot yang terdiri dari pipa yang menghubungkan tangki dengan ujung alat semprot. *Nozzle* ini berfungsi untuk mengubah aliran cairan menjadi semprotan dengan pola tertentu dan tekanan yang sesuai dengan kebutuhan. Bentuk dari *nozzle* bisa berbagai macam, seperti *konus*, *flat fan*, dan *full cone*. Setiap bentuk *nozzle* memiliki pola semprotan yang berbeda-beda dan cocok untuk kebutuhan yang berbeda-beda (Solahart Handal, 2025).



Gambar 2. 6 Nozzle Sprayer

Sumber: (Penulis, 2025)

Pada gambar di atas, terlihat *nozzle* dengan ujung berwarna jingga yang dapat disesuaikan, memungkinkan pengguna untuk mengatur intensitas atau pola semprotan, seperti kabut halus atau semprotan terarah. Komponen utama dari *nozzle* ini terdiri dari dua bagian: saluran utama berwarna hitam sebagai penghubung ke selang distribusi air atau cairan, dan bagian ujung *nozzle* yang berfungsi untuk mengatomisasi cairan menjadi partikel halus.

Desain ulir pada bagian jingga memungkinkan pengaturan manual tekanan dan *volume* semprotan, menjadikannya sangat ideal untuk sistem irigasi tetes atau penyemprotan otomatis berbasis sensor. *Nozzle* ini sangat umum digunakan dalam sistem pertanian modern, khususnya untuk penyemprotan pestisida, nutrisi cair, atau air dalam skala kecil dan terkontrol. Selain memberikan efisiensi penggunaan cairan, *nozzle* ini juga membantu menghindari pemborosan dan memastikan distribusi merata ke tanaman.

2.2.10 Relay

Modul *Relay* adalah sebuah saklar magnet, yang berfungsi untuk memutus dan menghubungkan arus listrik. Prinsip kerja *Relay* secara umum sama dengan kontakor magnet yaitu berdasarkan kemagnetan yang dihasilkan oleh kumparan *coil* (Effendi et al., 2022).



Gambar 2. 7 *Relay*

Sumber: (Penulis, 2025)

Pada gambar tersebut menunjukkan sebuah modul *Relay 2 Channel*. Dua kabel merah dan hitam terhubung pada terminal beban, yang biasanya digunakan untuk mengontrol perangkat eksternal seperti pompa air. Modul ini memiliki *optocoupler* dan *transistor* sebagai penguat sinyal dari mikrokontroler, serta *jumper* pengatur mode aktif. *Optocoupler* (atau *optoisolator*) berfungsi untuk menghubungkan dua rangkaian listrik tanpa kontak langsung, melainkan menggunakan cahaya sebagai media isolasi. *Transistor* bekerja sebagai saklar elektronik yang memungkinkan *Relay* aktif hanya saat sinyal dari mikrokontroler sesuai. Modul *Relay* ini umumnya membutuhkan tegangan kerja 5V atau 3.3V, tergantung spesifikasinya, dan memiliki indikator LED untuk menunjukkan status aktif (*ON*) atau tidak aktif (*OFF*) pada masing-masing *Channel*.

Dalam penelitian ini, *Relay* digunakan sebagai aktuator untuk mengontrol proses penyiraman dan pengkabutan otomatis di *Greenhouse* tanaman cabai. Perintah aktivasi *Relay* dikendalikan secara otomatis oleh sistem. Dengan penggunaan *Relay* ini, sistem dapat menghidupkan atau mematikan alat secara fisik, seperti pompa, sesuai kebutuhan tanaman secara cerdas dan efisien.

2.2.11 Unified Modeling language

UML (Unified Modeling Language) adalah sebuah bahasa visual yang digunakan untuk menggambarkan dan merancang sistem atau aplikasi secara jelas dan terstruktur. Dengan *UML*, pengembang bisa membuat berbagai diagram untuk menggambarkan bagaimana suatu sistem bekerja, siapa saja yang terlibat, dan bagaimana alur data berjalan di dalamnya (Irhan Hisyam Dwi Nugroho, 2024).

Berikut adalah diagram-diagram dalam *UML*:

1. Use Case Diagram

Use case diagram adalah satu dari berbagai jenis diagram *UML (Unified Modelling Language)* yang menggambarkan hubungan interaksi antara sistem dan aktor. *Use Case* dapat mendeskripsikan tipe interaksi antara si pengguna sistem dengan sistemnya (Irhan Hisyam Dwi Nugroho, 2024).

Daftar simbol pada *use case* diagram dapat dilihat pada tabel berikut:

Tabel 2. 4 *Use Case Diagram*

Sumber: (Irhan Hisyam Dwi Nugroho, 2024)

| No | Simbol | Keterangan |
|----|------------------------|--|
| 1 | Aktor | Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan <i>use case</i> |
| 2 | <i>Use Case</i> | Abstraksi dan interaksi antara sistem dan aktor |
| 3 | <i>Association</i> | Abstraksi dari penghubung antara aktor dengan <i>use case</i> |
| 4 | Generalisasi | Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan <i>use case</i> |
| 5 | <i>Include</i> | Menunjukkan bahwa suatu <i>use case</i> seluruhnya merupakan fungsionalitas dari <i>use case</i> lainnya |
| 6 | <i>Extend</i> | Menunjukkan bahwa suatu <i>use case</i> merupakan |

| | | |
|--|--|---|
| | | tambahan fungsional dari <i>use case</i> lainnya jika suatu kondisi terpenuhi |
|--|--|---|

2. Activity Diagram

Activity diagram adalah jenis diagram yang berguna untuk dapat membuat model dari berbagai proses dalam suatu sistem, urutan proses digambarkan secara vertikal. Diagram ini merupakan pengembangan dari *use case* dan menunjukkan alur aktivitas yang ditampilkan berupa rangkaian menu atau proses bisnis yang ada dalam sistem tersebut (Ucy Sugiarti, 2024).

Daftar simbol pada *activity diagram* dapat dilihat pada tabel berikut:

Tabel 2. 5 *Activity Diagram*

Sumber: (Ucy Sugiarti, 2024)

| No | Simbol | Keterangan |
|----|-----------------------------------|---|
| 1 | Status Awal | Sebuah diagram aktivitas memiliki sebuah status awal. |
| 2 | Aktivitas | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja. |
| 3 | Percabangan / <i>Decision</i> | Percabangan dimana ada pilihan aktivitas yang lebih dari satu. |
| 4 | Penggabungan / <i>Join</i> | Penggabungan dimana yang mana lebih dari satu aktivitas lalu digabungkan jadi satu. |
| 5 | Status Akhir | Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki status akhir. |
| 6 | <i>Swimlane</i> | <i>Swimlane</i> memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi |

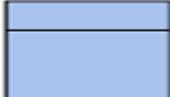
3. Class Diagram

Class Diagram adalah diagram yang digunakan dalam perancangan sistem berorientasi objek untuk menggambarkan struktur kelas, atribut, metode, serta hubungan antar kelas dalam sistem (Ayoni Sulthon, 2023).

Daftar simbol pada *Class Diagram* dapat dilihat pada tabel berikut:

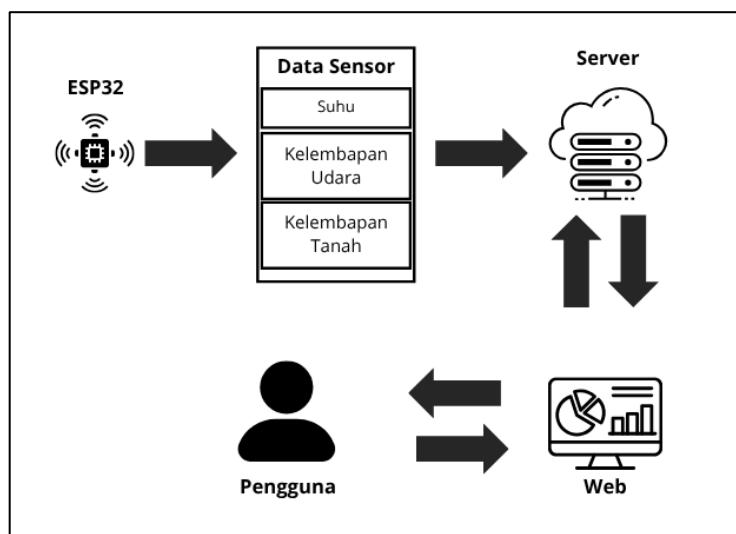
Tabel 2. 6 *Class Diagram*

Sumber: (Ayoni Sulthon, 2023)

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

2.2.12 Website

Website adalah kumpulan dari halaman-halaman situs yang terdapat dalam sebuah domain atau subdomain yang berada di *dalam World Wide Web (WWW)* di internet. *Website* dapat berisi teks, gambar, video, dan elemen interaktif lainnya yang dirancang untuk menyampaikan informasi, menyediakan layanan, atau menjalankan fungsi tertentu kepada pengunjungnya (Iftitah Nurul Laily, 2022).



Gambar 2. 8 Arsitektur *Website*

Sumber: (Wardani et al., 2021)

Gambar di atas menunjukkan alur kerja sistem berbasis *IOT* yang terhubung dengan *Website*. Sensor suhu, kelembapan udara, dan kelembapan tanah akan mengirimkan data ke mikrokontroler *ESP32*. Data tersebut kemudian dikirim ke *server* melalui jaringan internet. *Server* memproses data menggunakan algoritma *Naïve bayes* untuk menentukan apakah sistem penyiraman atau pengkabutan perlu dilakukan dan pengguna dapat memantau kondisi *Greenhouse* secara *real-time*.

Selain sebagai media *monitoring*, *Website* ini juga memungkinkan pengguna untuk mengendalikan sistem secara manual dari jarak jauh, seperti mengaktifkan atau mematikan penyiraman dan pengkabutan. Dengan demikian, *Website* menjadi bagian penting dalam sistem karena menghubungkan pengguna dengan perangkat *IOT* dan *server* secara interaktif dan efisien.

2.2.13 ***Hosting***

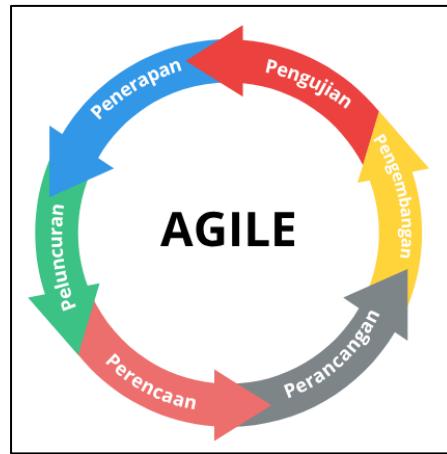
Hosting atau web *Hosting* adalah layanan penyimpanan yang digunakan untuk menampung seluruh kebutuhan sebuah *Website*, seperti *file*, gambar, video, *script*, *email*, aplikasi, dan *database*, agar *Website* tersebut dapat diakses secara *online* melalui internet. Tanpa layanan web *Hosting*, sebuah *Website* tidak dapat diakses oleh pengguna internet (Ariata C., 2024).

Terdapat beberapa jenis web *Hosting* yang dapat dipilih sesuai kebutuhan, antara lain:

- a. *Shared Hosting*: Berbagi sumber daya *server* dengan beberapa *Website* lain. Cocok untuk pemula dan situs dengan lalu lintas rendah.
- b. *VPS Hosting (Virtual Private Server)*: Memberikan lingkungan *server virtual* dengan sumber daya terdedikasi, meskipun masih berbagi fisik *server* dengan pengguna lain.
- c. *Dedicated Hosting*: Menyediakan satu *server* fisik secara eksklusif untuk satu *Website*. Cocok untuk situs dengan lalu lintas tinggi dan kebutuhan kustomisasi penuh.
- d. *Cloud Hosting*: Menggunakan sumber daya dari jaringan *server* yang saling terhubung, menawarkan fleksibilitas dan skalabilitas yang tinggi.

2.2.14 ***Metode Agile***

Metodologi *Agile Software Development Life Cycle (SDLC)* adalah pendekatan modern yang lebih fleksibel dibandingkan metode tradisional yang linier dan berurutan. *Agile* mengutamakan iterasi berkelanjutan dalam siklus pendek yang disebut *sprint*, sehingga memungkinkan penyesuaian cepat terhadap perubahan selama proses pengembangan. Dengan cara ini, tim dapat fokus pada pengembangan fitur yang berfungsi dan dapat diuji setiap siklusnya. Secara teori, *Agile* menekankan kolaborasi tim, komunikasi terbuka, dan pengiriman produk secara bertahap yang dapat diuji dan diperbaiki terus-menerus. Pendekatan ini mempercepat *feedback*, meningkatkan kualitas perangkat lunak, serta mengurangi risiko kegagalan proyek. *Agile* juga mendorong keterlibatan pengguna secara aktif agar hasil akhir lebih sesuai dengan kebutuhan (AgileTech Vietnam, 2025).



Gambar 2. 9 Metode *Agile*

Sumber: (AgileTech Vietnam, 2025)

Berikut adalah tahap-tahap pengembangan perangkat lunak yang digunakan dalam metode *Agile*:

1. Perencanaan

Tahap ini dilakukan untuk mengidentifikasi kebutuhan sistem serta menyusun tujuan yang ingin dicapai pada periode pengembangan tertentu yang disebut *sprint*.

2. Perancangan

Setelah kebutuhan dan tujuan ditentukan, tahap selanjutnya adalah merancang struktur sistem dan antarmuka pengguna.

3. Pengembangan

Pada tahap ini, sistem atau fitur mulai dibangun berdasarkan perancangan yang telah dibuat. Proses pengembangan dilakukan secara bertahap sesuai dengan durasi *sprint* yang telah ditetapkan.

4. Pengujian

Setelah proses pengembangan selesai, perangkat lunak yang dihasilkan diuji untuk memastikan fungsionalitasnya berjalan dengan baik dan sesuai dengan kebutuhan.

5. Penerapan

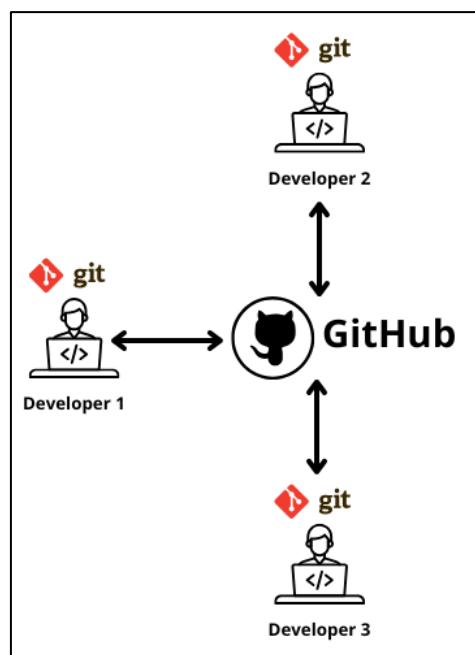
Jika perangkat lunak telah dinyatakan layak, maka hasil pengembangan akan diterapkan pada lingkungan produksi agar dapat digunakan oleh pengguna akhir.

6. Peluncuran

Setelah perangkat lunak melewati seluruh tahapan, lalu jika tidak ada perubahan pada tahapan sebelumnya maka sistem siap untuk diluncurkan.

2.2.15 *Github*

GitHub adalah platform berbasis *cloud* yang digunakan secara luas oleh pengembang perangkat lunak untuk mengelola proyek secara kolaboratif. Platform ini mendukung sistem kontrol versi *Git* yang memungkinkan pengguna melacak perubahan kode, bekerja dalam *GitHub* juga menyediakan berbagai fitur seperti *repository*, *branch*, *pull request*, dan *issue tracker* untuk mempermudah proses pengembangan tim (GitHub, 2025).



Gambar 2. 10 *GitHub*

Sumber: (AgileTech Vietnam, 2025)

Gambar di atas menggambarkan alur kerja kolaboratif dalam pengembangan perangkat lunak menggunakan *Git* dan *GitHub*. Setiap pengembang (*Developer 1*, *Developer 2*, dan *Developer 3*) melakukan perubahan kode secara lokal menggunakan *Git*, kemudian mendorong (*push*) atau menarik (*pull*) perubahan tersebut dari repository pusat yang disimpan di *GitHub*. Dengan pendekatan ini,

setiap perubahan kode dapat dilacak, dikelola, dan digabungkan (*merge*) dengan lebih mudah, sehingga mendukung pengembangan proyek secara terorganisir dan efisien.

Dalam konteks penelitian ini, *GitHub* digunakan sebagai media penyimpanan terpusat untuk mendokumentasikan perancangan dan menyimpan kode pengembangan sistem *IOT* penyiraman dan pengkabutan otomatis tanaman cabai. Pendekatan ini tidak hanya mendukung keteraturan pengembangan, tetapi juga mempermudah pengawasan, pemeliharaan, dan pembaruan sistem secara berkala, terutama saat melibatkan lebih dari satu pengembang atau saat sistem berkembang di masa mendatang.

2.2.16 *Arduino*

Bahasa pemrograman *Arduino* ditemukan oleh Hernando Barragan pada tahun 2003, pada saat ia mengembangkan sistem *wiring*. *Arduino* menggunakan bahasa pemrogramannya sendiri, yang mirip dengan C++. Terdapat tiga bagian utama yang menyusun bahasa pemrograman *Arduino* yaitu *function*, *value*, dan *structure*. *Function* memungkinkan dalam mengontrol *board*. Dengan menggunakan *function*, analisis data, operasi matematika, dan tugas lainnya dapat dilakukan. *Value* berfungsi mewakili konstanta dan variabel, tipe data yang digunakan seperti *array*, *boolean*, *char*, *float*, dan lainnya. (Togi, 2021).

```

1 // Header
2 #include <library.h>
3
4 // Deklarasi Variabel Global
5
6 // Setup Function
7 void setup() {
8     // Inisialisasi pin, variabel, atau kondisi awal
9 }
10
11 // Loop Function
12 void loop() {
13     // Kode yang akan dijalankan secara berulang
14 }
15

```

Gambar 2. 11 Bahasa Pemrograman *Arduino*

Sumber: (Elga Aris Prastyo, 2025)

Berikut ini merupakan penjelasan dari setiap bagian yang terdapat pada struktur program di atas:

1. Header

Bagian ini digunakan untuk menyertakan pustaka (*library*) yang diperlukan dalam program. Pustaka adalah kumpulan fungsi yang telah dibuat sebelumnya untuk memudahkan pemrograman, seperti mengakses sensor atau modul tertentu.

2. Deklarasi Variabel *Global*

Di bagian ini, variabel-variabel yang akan digunakan di seluruh bagian program baik di dalam *setup()* maupun *loop()* dideklarasikan. Variabel global memungkinkan penyimpanan data yang dapat diakses dari berbagai fungsi. Hal ini berguna untuk menyimpan nilai yang harus tetap tersedia selama program berjalan.

3. Setup Function

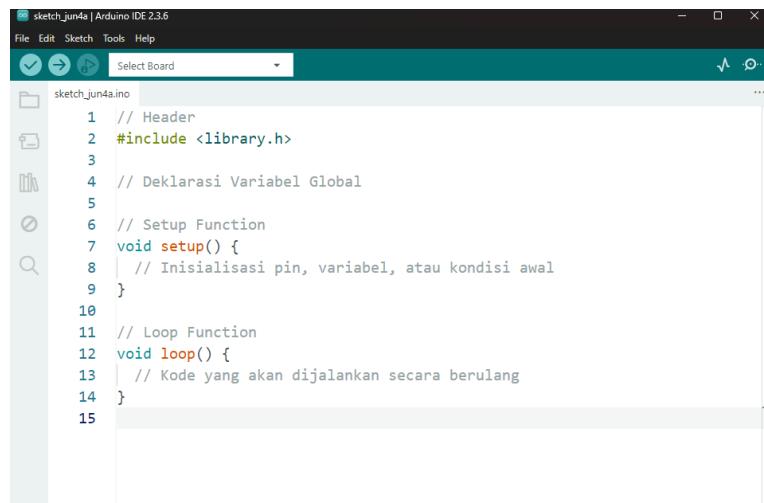
Fungsi *setup()* adalah fungsi khusus yang hanya dieksekusi satu kali saat program *Arduino* pertama kali dijalankan. Fungsi ini digunakan untuk melakukan inisialisasi, seperti mengatur pin, menginisialisasi variabel, atau menyiapkan kondisi awal.

4. Loop Function

Fungsi *loop()* adalah inti dari program *Arduino* yang berjalan terus-menerus setelah fungsi *setup()* selesai. Di dalam fungsi ini, perintah-perintah yang ingin dijalankan secara berulang diletakkan, seperti pembacaan sensor, pengendalian aktuator, atau logika pemrosesan data.

2.2.17 *Arduino IDE*

Arduino IDE (Integrated Development Environment) adalah perangkat lunak resmi yang digunakan untuk menulis, mengedit, dan mengunggah program ke papan mikrokontroler *Arduino*. *IDE* ini menyediakan antarmuka yang sederhana dan mudah digunakan. Terdapat dua versi utama *Arduino IDE* saat ini, yaitu *Arduino IDE 1.x* dan *Arduino IDE 2.x*. Versi 2.x merupakan pengembangan besar dari versi sebelumnya, dengan peningkatan kecepatan, antarmuka pengguna yang lebih modern, serta dukungan fitur lanjutan seperti *auto-completion* navigasi cepat, dan debugger bawaan. (Liam Aljundi, 2024).



```

1 // Header
2 #include <library.h>
3
4 // Deklarasi Variabel Global
5
6 // Setup Function
7 void setup() {
8 | // Inisialisasi pin, variabel, atau kondisi awal
9 }
10
11 // Loop Function
12 void loop() {
13 | // Kode yang akan dijalankan secara berulang
14 }
15

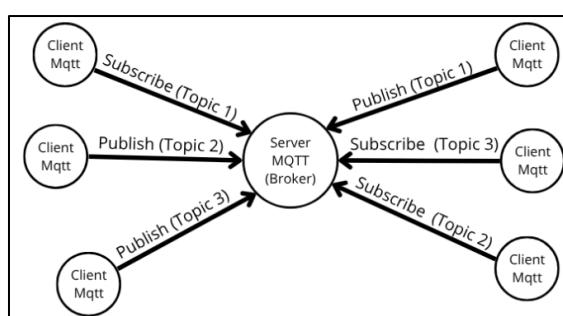
```

Gambar 2. 12 *Arduino IDE*

Sumber: (Penulis, 2025)

2.2.18 *MQTT*

MQTT (Message Queuing Telemetry Transport) adalah protokol *publish-subscribe* yang dirancang khusus untuk mengirimkan pesan antara perangkat *IOT* dengan *overhead* yang rendah dan konsumsi energi yang efisien. Dalam era *Internet of Things (IOT)* yang semakin berkembang, di mana jutaan perangkat terhubung dan bertukar data, diperlukan protokol komunikasi yang ringan, efisien, dan andal (Indobot Academy, 2023).



Gambar 2. 13 Arsitektur *MQTT*

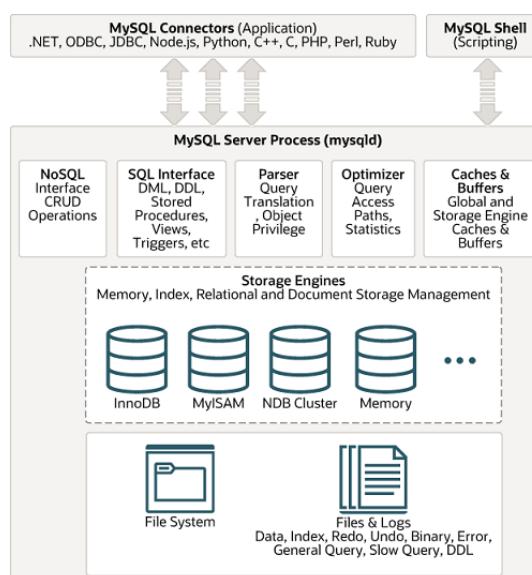
Sumber: (Valerie Lampkin, 2012)

Gambar di atas menggambarkan konsep dan cara kerja protokol *MQTT (Message Queuing Telemetry Transport)* yang berbasis arsitektur *publish-subscribe*. Dalam sistem ini, terdapat dua komponen utama, yaitu *client MQTT*

dan *server MQTT (broker)*. Setiap *client* dapat berperan sebagai *publisher* yang mengirimkan pesan ke suatu *topic*, atau sebagai *subscriber* yang menerima pesan dari *topic* tertentu yang telah didaftarkan. *Broker MQTT* bertindak sebagai perantara yang menerima pesan dari *publisher* dan meneruskannya ke semua *client* yang melakukan *subscribe* pada *topic* yang relevan. Sebagai contoh pada gambar, terdapat *client* yang melakukan *publish* pada *Topic 1*, dan *broker* akan mengirimkan pesan tersebut kepada *client* lain yang telah *subscribe* ke *Topic 1*. Mekanisme ini memastikan komunikasi yang efisien dan terdistribusi dengan baik antar perangkat *IOT* tanpa perlu mengetahui alamat satu sama lain. Dalam konteks penelitian ini, *MQTT* digunakan untuk menghubungkan perangkat sensor dan aktuator di *Greenhouse* dengan *server* dan antarmuka web.

2.2.19 MySQL

MySQL adalah sistem manajemen basis data berbasis *SQL (Structured Query Language)* yang memungkinkan pengguna untuk menyimpan, mengelola, dan mengambil data dengan cara yang terstruktur. Sebagai perangkat lunak *open-source*, *MySQL* memberikan kebebasan kepada pengembang untuk memodifikasi dan mendistribusikan perangkat lunak ini sesuai kebutuhan. (Yazid Yusuf, 2024).



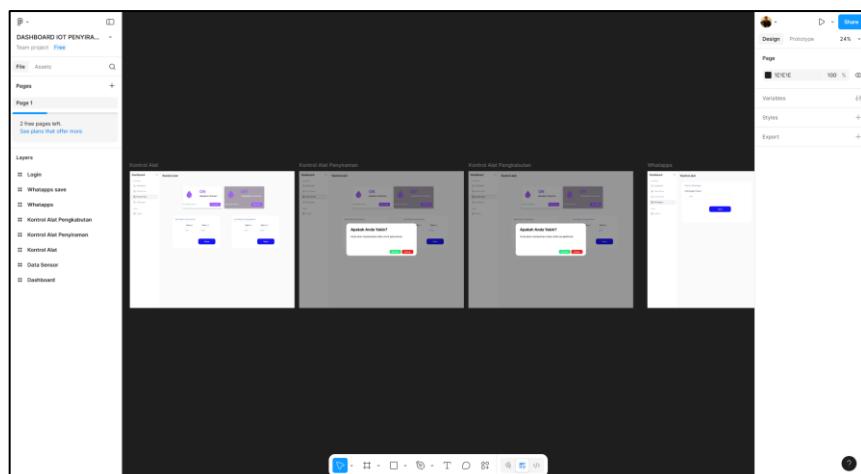
Gambar 2. 14 Arsitektur *MySQL*

Sumber: (MySQLTutorial.org, 2008)

Gambar di atas menjelaskan arsitektur dari MySQL menyediakan *MySQL Connectors* yang mendukung berbagai bahasa pemrograman seperti *Python*, *JavaScript* dan lainnya, memungkinkan integrasi lintas platform. *MySQL Server Process (mysqld)* menjadi komponen inti yang mengelola berbagai fungsi utama seperti antarmuka *SQL (DML, DDL, prosedur tersimpan, view, trigger)*, *parser* untuk penerjemahan *query*, serta *optimizer* untuk menentukan jalur akses terbaik ke data. Selain itu, terdapat *caches & buffers* yang bertugas mempercepat pemrosesan data melalui penyimpanan sementara. Di bagian bawah, arsitektur ini menunjukkan beragam *storage engine* seperti InnoDB, MyISAM, NDB Cluster, dan *Memory*, yang berperan dalam manajemen penyimpanan data. Semua data dan *log* terkait aktivitas basis data disimpan dalam file sistem, termasuk data transaksi, *redo log*, dan *error log*. Dalam konteks penelitian ini, MySQL digunakan sebagai media penyimpanan data sensor suhu, kelembapan tanah, dan kelembapan udara yang diperoleh dari perangkat *IOT* di *Greenhouse*.

2.2.20 Figma

Figma adalah rangkaian produk yang memungkinkan Anda membuat, berbagi, dan menguji desain, presentasi, dan papan tulis. Anda bisa bekerja dengan orang lain secara *real-time*, mengakses fitur-fitur canggih dalam *Mode Dev*, dan berintegrasi dengan alat populer seperti *GitHub* (Figma, 2025).



Gambar 2. 15 Tampilan *Figma*

Sumber: (Penulis, 2025)

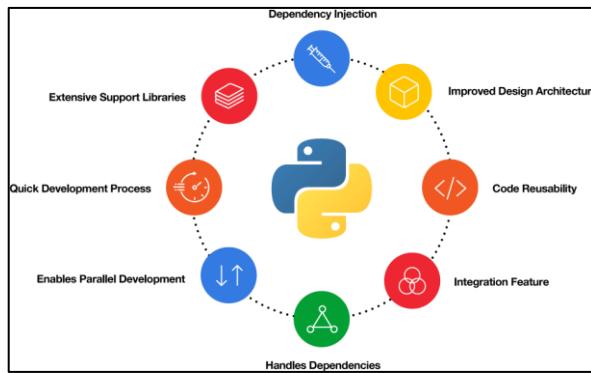
Figma merupakan platform desain antarmuka berbasis *cloud* yang mendukung kolaborasi secara *real-time*. Aplikasi ini memungkinkan tim desain dan pengembang bekerja bersama dalam satu ruang kerja digital, tanpa perlu menginstal perangkat lunak tambahan. Sebagaimana ditampilkan pada Gambar, antarmuka *Figma* terdiri dari panel *layer* di sisi kiri, area kerja utama di tengah, dan panel properti di sebelah kanan. Tampilan tersebut memperlihatkan fitur *horizontal layer scrolling* yang membantu pengguna menavigasi *layer* yang kompleks secara lebih mudah dan efisien.

Keunggulan utama *Figma* yang membuatnya populer dalam dunia desain digital antara lain:

- a. *Real-time functionality*: Setiap perubahan yang dilakukan akan langsung disimpan otomatis dan dapat dilihat oleh seluruh anggota tim secara serentak, memungkinkan kolaborasi yang dinamis.
- b. *Integrated prototyping*: *Figma* menyediakan fitur untuk membuat prototipe interaktif yang dapat langsung diuji pada perangkat target, tanpa perlu berpindah ke aplikasi lain.
- c. *Design library*: Fitur ini memungkinkan pengguna untuk membuat dan menggunakan kembali komponen, gaya, serta aset desain dalam berbagai proyek, sehingga meningkatkan konsistensi desain.
- d. *Easy sharing*: Projek yang dibuat di *Figma* dapat dengan mudah dibagikan melalui tautan kepada anggota tim (Nadifa Padantya Raihanah, 2023).

2.2.21 *Python*

Python adalah pemrograman tingkat tinggi yang diinterpretasikan, berorientasi objek, bahasa dengan semantik dinamis. Data bawaan tingkat tinggi struktur, dikombinasikan dengan pengetikan dinamis dan pengikatan dinamis, membuatnya sangat menarik untuk pengembangan aplikasi cepat, serta untuk digunakan sebagai bahasa skrip untuk menghubungkan komponen yang ada bersama. Sintaks *Python* yang sederhana dan mudah dipelajari menekankan keterbacaan dan karenanya mengurangi biaya pemeliharaan program. *Python* mendukung modul dan paket, yang mendorong program modularitas dan penggunaan kembali kode. (Python org, 2025).



Gambar 2. 16 Arsitektur *Python*

Sumber: (Web Idea Solution, 2025)

Pada gambar di atas, *Python* ditampilkan sebagai inti dari berbagai fitur utama yang membuatnya unggul dan relevan dalam pengembangan sistem modern, termasuk untuk penelitian ini. Di antaranya adalah:

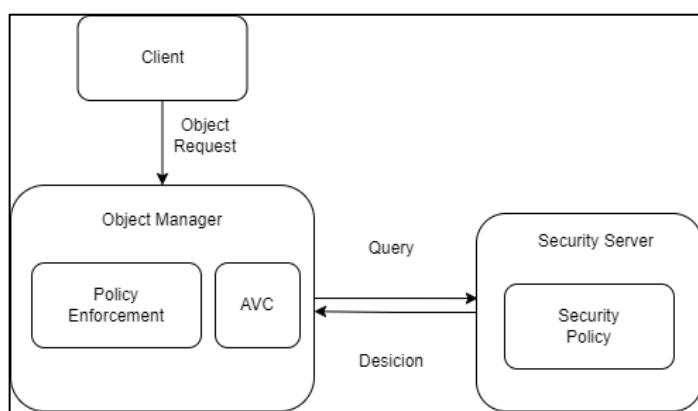
- a. *Dependency Injection*: Mempermudah pengelolaan dependensi antar komponen, sehingga lebih fleksibel dan mudah diuji.
- b. *Improved Design Architecture*: Mendukung pengembangan arsitektur sistem yang bersih dan terstruktur, termasuk pada sistem *IOT* berbasis sensor.
- c. *Code Reusability*: *Python* memungkinkan penggunaan ulang kode secara luas melalui modul dan pustaka.
- d. *Integration Feature*: *Python* sangat mudah diintegrasikan dengan berbagai sistem, protokol (seperti *MQTT*), dan *database*.
- e. *Handles Dependencies*: Mengelola ketergantungan pustaka dengan mudah menggunakan tools seperti *pip* atau *venv*.
- f. *Enables Parallel Development*: Mendukung pengembangan paralel antar tim melalui modul-modul terpisah.
- g. *Quick Development Process*: Cepat dalam proses pengembangan karena sintaksnya ringkas dan komunitasnya luas.
- h. *Extensive Support Libraries*: *Python* memiliki pustaka pendukung yang sangat lengkap.

Dalam konteks penelitian ini, *Python* menjadi komponen utama dalam membangun sistem *IOT* otomatis untuk penyiraman dan pengkabutan tanaman cabai. *Python* digunakan dalam proses integrasi data sensor (suhu, kelembapan

udara, dan tanah), penerapan algoritma *Naïve bayes* untuk pengambilan keputusan otomatis, serta pengembangan aplikasi web.

2.2.22 *Flask*

Flask adalah kerangka kerja aplikasi web yang ringan dan sesuai dengan standar *Web Server Gateway Interface (WSGI)*. Kerangka kerja ini dirancang untuk memudahkan pengembangan aplikasi secara cepat dan sederhana, serta memiliki fleksibilitas untuk dikembangkan menjadi aplikasi yang lebih kompleks (Pallets, 2025).



Gambar 2. 17 Arsitektur *Flask*

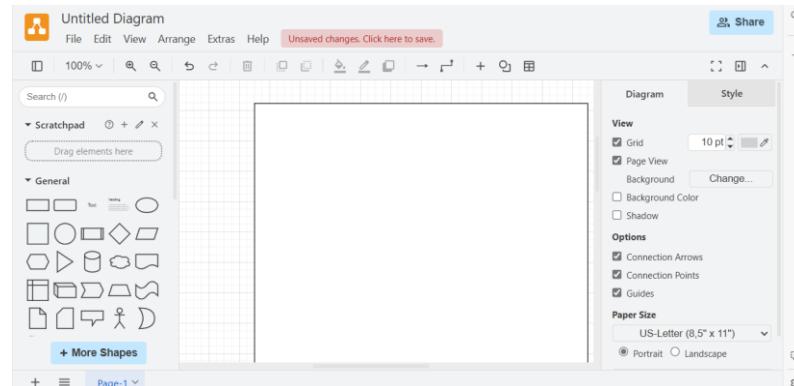
Sumber: (Ding et al., 2012)

Gambar di atas menggambarkan konsep kerja *Flask* dalam konteks keamanan dan otorisasi berbasis kebijakan. Saat seorang *client* mengirimkan permintaan akses terhadap objek, permintaan ini dikelola oleh *Object Manager* yang terdiri dari dua komponen utama: *Policy Enforcement* (untuk menegakkan kebijakan) dan *AVC* (*Access Vector Cache*), yang menangani kontrol akses. *Object Manager* kemudian mengajukan *query* ke *Security Server*, yang berisi komponen *Security Policy*. Berdasarkan kebijakan ini, *server* memberikan keputusan (*decision*) apakah permintaan diizinkan atau ditolak. Dalam konteks penelitian ini, *Flask* digunakan sebagai kerangka kerja utama untuk membangun aplikasi *monitoring* berbasis web. Aplikasi ini memungkinkan pengguna untuk melihat kondisi *real-time* dari *Greenhouse* melalui data sensor seperti suhu,

kelembapan udara, dan kelembapan tanah. *Flask* juga mengatur kontrol akses dan komunikasi antara antarmuka pengguna dengan *backend*, termasuk dalam proses pengambilan keputusan otomatis menggunakan algoritma *Naïve bayes*.

2.2.23 *Draw.io*

Draw.io adalah platform penggambaran grafik, *flowchart*, *chartnetwork*, diagram dan lain-lain. *Draw.io* juga menyediakan fitur pembuatan diagram berbasis web yang bekerja sama dengan *Google Drive* dan *Dropbox* untuk menyimpan proyeknya. *Draw.io* ditemukan atau didirikan pada tahun 2000 oleh Gaudenz Alderdi Norpathapton (Walid & Susanto, 2024).



Gambar 2. 18 *Draw.io*

Sumber: (Penulis, 2025)

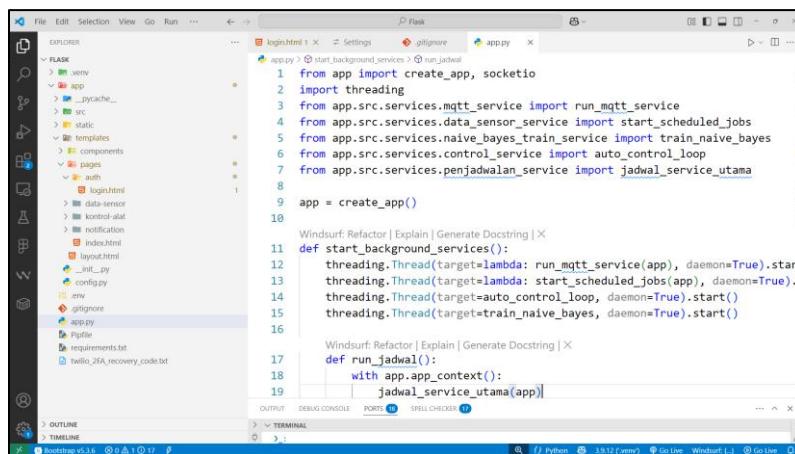
Platform *Draw.io* mendukung kolaborasi berbasis *cloud* karena terintegrasi dengan layanan penyimpanan seperti *Google Drive* dan *Dropbox*, sehingga memungkinkan penyimpanan dan akses proyek secara *online* secara *real-time*. Dalam konteks perancangan sistem *IOT*, kemampuan ini sangat bermanfaat untuk mendukung kerja tim lintas perangkat dan lokasi yang membutuhkan akses cepat terhadap diagram rancangan. *Draw.io* sangat berguna dalam merancang sistem karena menawarkan kemudahan penggunaan, antarmuka yang intuitif, serta kemampuan ekspor dalam berbagai format seperti PNG, PDF, dan XML yang mempermudah dokumentasi dan pelaporan.

Dengan fitur yang fleksibel dan lengkap tersebut, *Draw.io* dipilih sebagai alat bantu utama dalam merancang sistem *IOT* penyiraman dan pengkabutan

otomatis tanaman cabai, karena mampu menggambarkan struktur, alur data, dan integrasi antar komponen dengan jelas dan efisien, sekaligus mendukung proses pengembangan secara terstruktur dan terdokumentasi.

2.2.24 Visual Studio Code

Visual Studio Code adalah editor kode yang disederhanakan dengan dukungan untuk operasi pengembangan seperti penelusuran kesalahan, menjalankan tugas, dan kontrol versi. Ini bertujuan untuk menyediakan alat yang dibutuhkan pengembang untuk siklus pembuatan *code-debug* cepat dan meninggalkan alur kerja yang lebih kompleks untuk *IDE* berfitur yang lebih lengkap (Microsoft, 2025).



Gambar 2. 19 *Visual Studio Code*

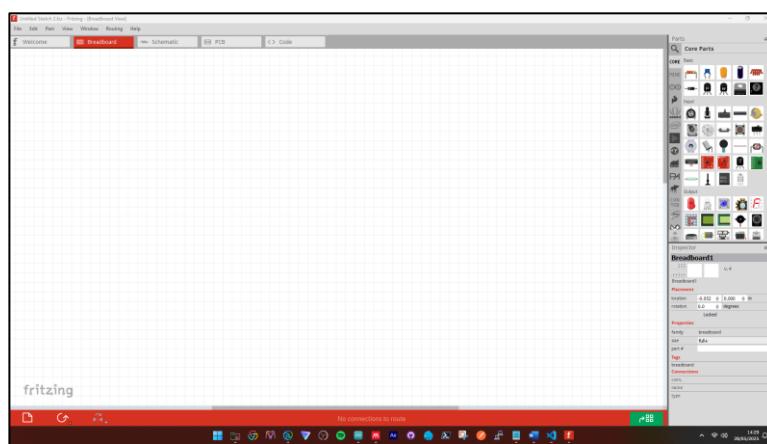
Sumber: (Penulis,2025)

Gambar di atas menampilkan antarmuka *VS Code* yang digunakan dalam menulis kode program untuk projek pembuatan aplikasi *IOT* untuk sistem penyiraman dan pengkabutan otomatis tanaman cabai di *Greenhouse*.

2.2.25 Fritzing

Fritzing adalah inisiatif perangkat keras sumber terbuka yang membuat elektronik dapat diakses sebagai materi kreatif bagi siapa saja. Menawarkan alat perangkat lunak, situs web komunitas, dan layanan dalam pemrosesan dan

Arduino, menumbuhkan ekosistem kreatif yang memungkinkan pengguna untuk mendokumentasikan *prototipe* mereka (Fritzing, 2025).



Gambar 2. 20 *Fritzing*

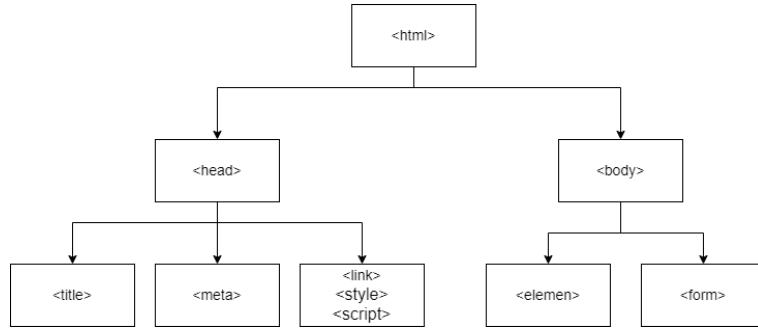
Sumber: (Penulis, 2025)

Dalam penelitian ini, *Fritzing* digunakan untuk merancang diagram *wiring* sistem *IOT* otomatis untuk penyiraman dan pengkabutan tanaman cabai di *Greenhouse*. Diagram *wiring* ini mencakup koneksi antara sensor suhu, kelembapan udara, dan kelembapan tanah dengan mikrokontroler, serta pengaturan aktuator seperti pompa dan *sprayer*. Dengan tampilan intuitif dan fitur visualisasi koneksi kabel yang jelas, *Fritzing* memudahkan dalam mendokumentasikan jalur koneksi secara akurat sebelum diimplementasikan secara fisik.

Fitur seperti *Breadboard View*, *Schematic View*, dan *PCB View* mendukung proses desain mulai dari prototipe awal hingga ke perancangan sirkuit cetak (*PCB*), menjadikan *Fritzing* alat penting dalam pengembangan sistem elektronik berbasis *IOT* yang terstruktur dan dapat direproduksi.

2.2.26 *HTML*

HTML (*HyperText Markup Language*) adalah bahasa yang digunakan untuk membuat struktur dasar halaman web. *HTML* terdiri dari elemen-elemen seperti tag, atribut, dan konten yang membentuk tampilan dan susunan informasi di web (Publikasi et al., 2025).



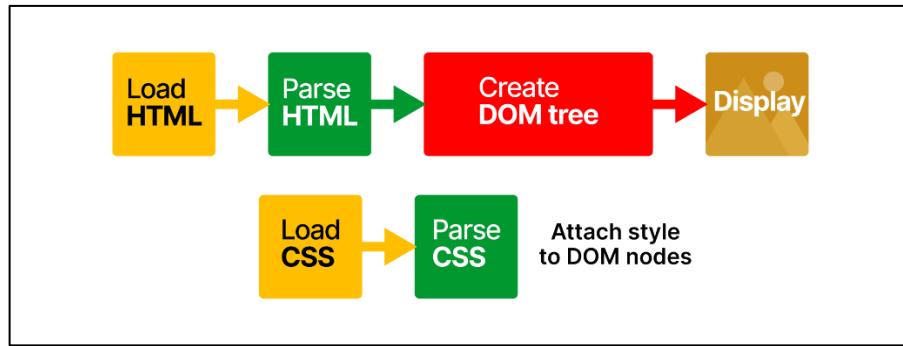
Gambar 2. 21 Arsitektur *HTML*

Sumber: (Ade Roni, 2025)

Gambar di atas menunjukkan struktur hierarki dasar dokumen *HTML*. Elemen <html> berfungsi sebagai elemen utama yang membungkus seluruh isi dokumen. Di dalamnya terdapat dua bagian utama, yaitu <head> dan <body>. Bagian <head> berisi elemen-elemen yang bersifat metadata dan tidak langsung ditampilkan di halaman web, seperti <title> (judul halaman), <meta> (informasi *metadata* seperti *charset* dan deskripsi), serta elemen-elemen tambahan seperti <link> (menghubungkan file eksternal seperti *CSS*), <style> (gaya *CSS* internal), dan <script> (kode *JavaScript*). Sementara itu, bagian <body> memuat konten utama yang akan ditampilkan di halaman web, seperti elemen teks, gambar, dan form interaktif. Gambar ini menggambarkan bahwa struktur *HTML* bekerja secara hierarkis dan terstruktur, dimulai dari tag utama hingga elemen-elemen turunan yang membentuk antarmuka pengguna serta logika interaksi dalam sebuah halaman web. Struktur ini penting untuk memastikan halaman web dapat dibaca dan ditampilkan dengan benar oleh browser.

2.2.27 CSS

CSS adalah singkatan dari *Cascading Style Sheets* digunakan untuk menata dan memperindah tampilan halaman web. *CSS* memungkinkan pengembang web mengontrol gaya visual dari elemen-elemen *HTML* seperti warna latar belakang, ukuran dan jenis *font*, spasi antar elemen, serta tata letak keseluruhan halaman. Dengan *CSS*, desain antarmuka web menjadi lebih konsisten, menarik, dan responsif terhadap berbagai ukuran layar perangkat (Publikasi et al., 2025).



Gambar 2. 22 Arsitektur CSS

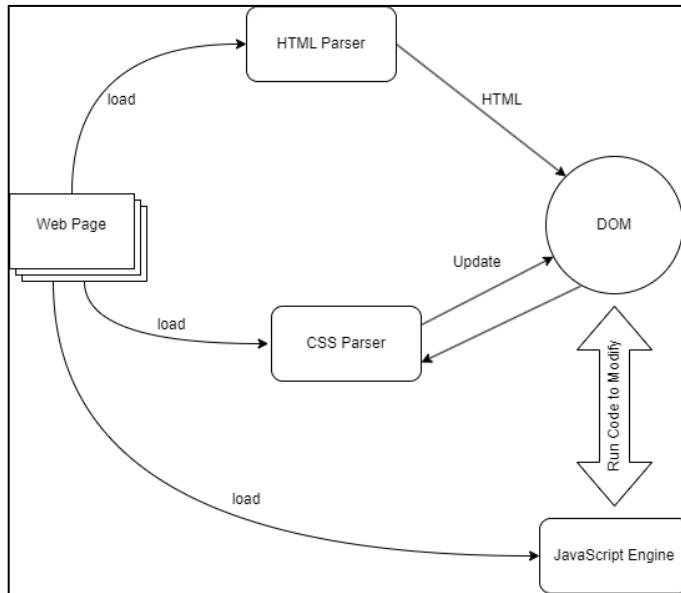
Sumber: (Mozilla Foundation., 2025)

Gambar di atas menjelaskan alur kerja browser saat merender halaman web. Proses diawali dengan pemuatan (*load*) dan penguraian (*parse*) file *HTML* untuk membentuk struktur *DOM (Document Object Model) tree*. Secara paralel, browser juga memuat dan mengurai file *CSS* untuk membentuk *CSSOM (CSS Object Model)*. Kombinasi antara *DOM* dan *CSSOM* ini kemudian digunakan oleh browser untuk merender dan menampilkan antarmuka halaman web secara visual kepada pengguna.

Dalam konteks penelitian ini, yaitu merancang dan membangun aplikasi *IOT* untuk penyiraman dan pengkabutan otomatis tanaman cabai di *Greenhouse*, *CSS* berperan penting dalam membentuk tampilan antarmuka dari aplikasi monitoring berbasis web. Aplikasi ini memungkinkan pengguna memantau data sensor suhu, kelembapan udara, dan kelembapan tanah secara *real-time*, serta mengontrol sistem penyiraman dan pengkabutan dari jarak jauh. *CSS* memastikan bahwa informasi yang disajikan dapat ditampilkan secara responsif, mudah dipahami, dan menarik secara visual baik di perangkat *desktop* maupun *mobile*. Dengan tampilan yang baik, pengguna dapat mengambil keputusan dengan cepat dan efisien berdasarkan data lingkungan yang disajikan.

2.2.28 JavaScript

JavaScript adalah bahasa skrip lintas platform berorientasi objek yang digunakan untuk membuat halaman web interaktif misalnya, memiliki animasi, tombol yang dapat diklik, menu *popup*, dll (Mozilla Foundation, 2025).



Gambar 2. 23 Arsitektur *Javascript*

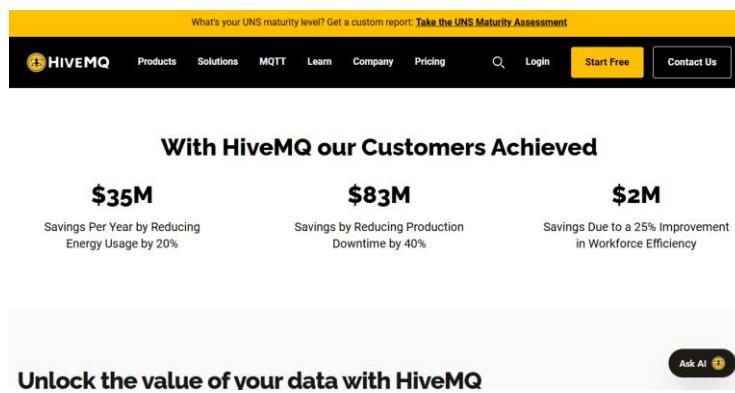
Sumber: (Rifqi Mulyawan Digital, 2025)

Gambar di atas menunjukkan bagaimana *JavaScript* bekerja dalam proses rendering halaman web di browser. Setelah halaman web dimuat, konten *HTML* dikirim ke *HTML Parser* untuk membentuk struktur *DOM* (*Document Object Model*). Sementara itu, file *CSS* juga dimuat dan diproses oleh *CSS Parser* untuk menentukan tampilan elemen-elemen pada *DOM*. Di sisi lain, *JavaScript Engine* memuat skrip *JavaScript* yang dapat berjalan secara asinkron dan digunakan untuk memanipulasi atau memperbarui struktur *DOM* secara dinamis. Proses ini memungkinkan interaksi waktu nyata, seperti pembaruan nilai sensor atau tampilan kondisi tanpa perlu *reload* halaman.

Dalam konteks penelitian ini, *JavaScript* sangat penting dalam pengembangan aplikasi *monitoring* berbasis web untuk sistem penyiraman dan pengkabutan otomatis tanaman cabai di *Greenhouse*. Melalui *JavaScript*, antarmuka aplikasi dapat menampilkan data sensor suhu, kelembapan udara, dan kelembapan tanah secara *real-time*, serta memungkinkan pengguna mengontrol sistem dari jarak jauh. *JavaScript* juga bekerja sama dengan *CSS* dan *HTML* untuk memberikan respons visual instan saat terjadi perubahan kondisi lingkungan, serta menjalankan logika interaktif seperti aktivasi penyiraman berbasis hasil prediksi dari algoritma *Naïve bayes*. Dengan begitu, *JavaScript* mendukung efisiensi dan kepraktisan dalam pengawasan dan pengendalian *Greenhouse* secara otomatis.

2.2.29 *HiveMQ*

HiveMQ adalah platform *MQTT* kelas dunia yang siap digunakan di lingkungan perusahaan. Platform ini dirancang untuk pergerakan data yang cepat, efisien, dan andal antara perangkat *IOT* dan sistem perusahaan. *HiveMQ* sepenuhnya mendukung protokol *MQTT*, memungkinkan komunikasi dua arah secara instan dan *real-time*, sehingga mendukung otomatisasi, analitik, dan pemrosesan data berbasis *AI*.



Gambar 2. 24 Tampilan *HiveMQ*

Sumber: (Penulis, 2025)

HiveMQ dipilih sebagai *broker MQTT* dalam sistem ini karena menawarkan komunikasi data yang cepat, ringan, dan andal antara perangkat *IOT* (seperti *ESP32*) dengan *server*. *HiveMQ* dirancang khusus untuk mendukung komunikasi *real-time* pada skala industri, serta kompatibel sepenuhnya dengan protokol *MQTT*. Platform ini memungkinkan pertukaran data dua arah secara instan, sehingga sangat sesuai untuk sistem otomatisasi seperti penyiraman dan pengkabutan tanaman yang memerlukan respon cepat berdasarkan data sensor.

2.2.30 *Whatsapp-web.js*

Whatsapp-web.js adalah sebuah *library Node.js* yang memungkinkan pengembang untuk mengintegrasikan dan mengontrol *Whatsapp* melalui *Whatsapp Web* secara otomatis. *Library* ini bekerja dengan cara meluncurkan aplikasi *Whatsapp Web* di dalam browser yang dikendalikan oleh *Puppeteer*,

sebuah pustaka pengendali *headless browser*. Melalui pendekatan ini, *Whatsapp-web.js* dapat mengakses berbagai fungsi internal *Whatsapp Web*, sehingga memungkinkan pengguna untuk mengirim dan menerima pesan, membaca status, mengelola grup, dan fitur lainnya, hampir sama seperti menggunakan *Whatsapp Web* secara manual. Karena *library* ini memanfaatkan antarmuka resmi *Whatsapp Web*, risiko pemblokiran menjadi lebih rendah dibandingkan pendekatan tidak resmi lainnya.



Gambar 2. 25 *Whatsapp-web.js*

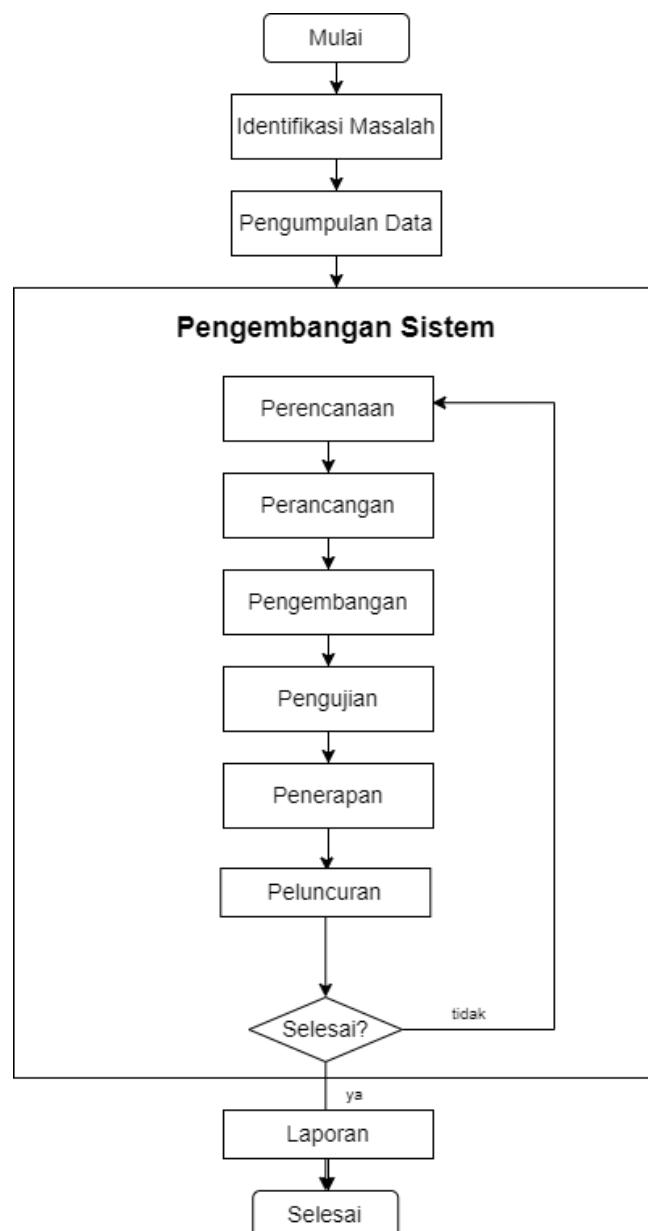
Sumber: (Pengguna, 2025)

BAB III

METODOLOGI PENELITIAN

3.1 Kerangka Pikir

Kerangka pikir merupakan jalur pemikiran yang dirancang berdasarkan kegiatan penyusun yang dilakukan. Berikut adalah kerangka pikir yang merupakan langkah-langkah yang akan dilakukan dalam penelitian ini:



Gambar 3. 1 Kerangka Pikir

3.2 Deskripsi

Penelitian ini menggunakan pendekatan metode *Agile* dalam pengembangan sistem. Berikut adalah tahapan-tahapan penelitian yang dilaksanakan:

3.2.1 Identifikasi Masalah

Identifikasi masalah adalah upaya untuk menjelaskan permasalahan. Identifikasi ini dilakukan sebagai langkah awal penelitian. Dimulai dengan meminta izin kepada pemilik Avicenna *Greenhouse* untuk melakukan penelitian kemudian melakukan pengamatan secara langsung terhadap objek penelitian.

3.2.2 Pengumpulan Data

Dalam pengumpulan data, penulis menggunakan beberapa metode untuk mendapatkan data yang akurat yang diperlukan dalam implementasi alat *Internet of Things*, penyusunan proposal, dan penyusunan laporan, yaitu sebagai berikut:

1. Observasi

Penulis melakukan pengamatan langsung di Avicenna *Greenhouse* untuk memahami kondisi dan kebutuhan penyiraman tanaman secara manual.

2. Wawancara

Penulis melakukan sesi tanya jawab dengan pemilik Avicenna *Greenhouse* guna memperoleh informasi terkait metode penyiraman yang digunakan, kendala dalam penyiraman dan pengkabutan tanaman, serta harapan terhadap sistem penyiraman otomatis berbasis *Internet of Things*.

3. Studi Pustaka

Penulis mengumpulkan dan menganalisis data dari berbagai jurnal, buku, website serta sumber relevan lainnya yang membahas konsep *Internet of Things*, sistem penyiraman otomatis, dan pengkabutan pada tanaman.

3.2.3 Perencanaan

Setelah melakukan pengumpulan data, penulis melakukan tahap perencanaan sistem. Tahapan ini bertujuan untuk menetapkan kebutuhan sistem serta

merancang alur pengembangan yang akan dilaksanakan dalam beberapa iterasi (*sprint*). Perencanaan meliputi identifikasi kebutuhan sistem, baik dari sisi fungsional maupun *non-fungsional*, agar proses implementasi dapat berjalan sesuai dengan tujuan penelitian.

1. Analisis kebutuhan fungsional

Analisis kebutuhan fungsional merupakan proses untuk mengidentifikasi fitur-fitur utama yang harus dimiliki oleh sistem berdasarkan fungsionalitas yang diharapkan. Adapun kebutuhan fungsional sistem yang dirancang adalah sebagai berikut:

a. Autentikasi Pengguna

- 1) Sistem menyediakan *form Login* untuk *admin*.
- 2) Sistem memverifikasi kredensial pengguna sebelum memberikan akses ke halaman utama.

b. Dashboard Admin

- 1) Menampilkan data dari sensor suhu, kelembapan tanah, dan kelembapan udara.
- 2) Menampilkan status perangkat penyiram dan pengkabut.

c. Integrasi IOT dan Proses Pengambilan Keputusan

- 1) Sistem menggunakan algoritma *Naive Bayes* untuk menentukan kapan melakukan penyiraman dan pengkabutan.

d. Pengendalian Perangkat

- 1) Sistem menerima data dari *ESP32* melalui protokol *MQTT*.
- 2) Sistem mengirim perintah ke *ESP32* untuk mengontrol perangkat.

e. Notifikasi dan *Monitoring*

- 1) *Admin* dapat mengaktifkan/menonaktifkan penyiraman dan pengkabutan secara manual melalui notifikasi via *Whatsapp*.

2. Kebutuhan Non-Fungsional

Analisis kebutuhan non-fungsional mencakup spesifikasi teknis dan kualitas sistem yang memengaruhi performa dan keandalan sistem secara keseluruhan. Kebutuhan ini tidak secara langsung berkaitan dengan fungsi sistem, namun sangat penting untuk menjamin bahwa sistem berjalan dengan efisien dan aman.

Adapun kebutuhan non-fungsional dalam sistem ini meliputi:

1) Perangkat Keras

Tabel 3. 1 Perangkat Keras

| | |
|------------------|-------------------------|
| <i>Processor</i> | Intel Dual-Core 3.10GHz |
| <i>Hard Disk</i> | 500 GB |
| <i>Memory</i> | 4 GB |
| Monitor | Resolusi 1280 x 800px |

2) Perangkat *Internet of Things*

Tabel 3. 2 Perangkat *Internet of Things*

| | |
|----------------------------------|--|
| Mikrokontroler | <i>ESP32</i> |
| Sensor Suhu dan Kelembapan Udara | <i>DHT11</i> |
| Sensor Kelembapan Tanah | <i>Soil Moisture Sensor</i> |
| Pompa Air | <i>Mini water pump DC 3–6V & Sanyo</i> |
| Pengkabutan | <i>Nozzle Sprayer</i> |
| <i>Relay Module</i> | <i>2 Channel Relay 5V</i> |
| Power Supply & Adaptor | Adapter 5V/2A |
| Selang | Selang plastik PE (<i>Polyethylene</i>) |
| Internet | <i>Hostpot HP</i> |
| Wadah Alat | <i>Box Casing</i> |

3) Perangkat Lunak

Tabel 3. 3 Perangkat Lunak

| | |
|------------------------------------|--|
| Sistem Operasi | Window 11 / Ubuntu 22.04 |
| Bahasa Pemrograman | <i>Python, Arduino</i> |
| <i>Backend</i> dan <i>Frontend</i> | <i>Flask</i> |
| <i>Database</i> | <i>MySQL</i> |
| <i>Broker MQTT</i> | <i>HiveMQ</i> |
| Editor Kode | <i>Visual Studio Code, Arduino IDE</i> |
| Diagram & Wireframe | <i>Draw.io</i> |

| | |
|-----------------------|------------------------|
| <i>User Interface</i> | <i>Figma</i> |
| <i>Wiring</i> | <i>Fritzing</i> |
| <i>Laporan</i> | <i>Microsoft Word</i> |
| <i>Whatsapp</i> | <i>Whatsapp-web.js</i> |

3.2.4 Perancangan

1. Perancangan Diagram

a. *Use Case Diagram*

Diagram ini menggambarkan keterhubungan antara aktor dan fungsionalitas sistem. Aktor dalam sistem ini adalah *Admin*, yang memiliki hak akses penuh terhadap pengelolaan data dan pengendalian perangkat.

- 1) Melakukan *Login* ke *Dashboard*
- 2) Melihat data sensor secara *real-time*
- 3) Melihat hasil prediksi sistem (penyiraman/pengkabutan)
- 4) Mengaktifkan/menonaktifkan perangkat secara manual
- 5) Menerima notifikasi hasil prediksi dan mengirim link untuk akses *Dashboard* melalui *Whatsapp*.

b. *Activity Diagram*

Diagram aktivitas menggambarkan alur proses sistem mulai dari pengambilan data sensor hingga pengiriman hasil prediksi. Adapun alur aktivitas yang digambarkan meliputi:

- 1) Aktivitas *Login admin*
- 2) Aktivitas membaca data dari sensor
- 3) Aktivitas memproses data dengan algoritma *Naive Bayes*
- 4) Aktivitas mengirim hasil prediksi ke *WhatsApp*
- 5) Aktivitas untuk kontrol penyiraman dan pengkabutan

c. *Wiring Diagram*

Wiring Diagram (skema rangkaian) menunjukkan koneksi fisik antar komponen perangkat keras seperti sensor, *ESP32*, dan aktuator. Beberapa komponen seperti:

- 1) *ESP32* (mikrokontroler)
 - 2) Sensor *DHT11* (untuk suhu dan kelembapan udara)
 - 3) Sensor kelembapan tanah
 - 4) *Nozzle* kabut dan kipas
 - 5) *Breadboard* dan kabel jumper
- d. *Class Diagram*
- Perancangan sistem menggunakan *Class Diagram* untuk menggambarkan struktur kelas dalam sistem serta hubungan antar kelas. *Class Diagram* membantu dalam proses implementasi sistem berbasis objek, terutama dalam pengembangan *backend* dan integrasi dengan sensor *IOT*. Berikut adalah penjelasan dari masing-masing kelas:
- 1) Data Sensor
 - 2) Riwayat Aksi
 - 3) Jadwal Penyiraman
 - 4) Hasil Prediksi
2. Perancangan Antarmuka Pengguna

Pada tahapan ini, desain antarmuka aplikasi dibuat menggunakan aplikasi *Figma* dengan desain antarmuka yang sederhana dan tentunya memperhatikan *user experience*.

3.2.5 Pengembangan

Tahap ini melibatkan proses penerjemahan seluruh desain sistem menjadi kode program sesuai dengan spesifikasi yang telah dirancang. Sistem ini dikembangkan menggunakan bahasa pemrograman *Python* dengan *framework Flask* sebagai web *framework*.

Komunikasi antara perangkat *IOT* dan *server* dilakukan menggunakan protokol *MQTT* untuk memastikan pengiriman data sensor dan perintah kontrol secara efisien dan *real-time*.

Pada sisi klien, aplikasi antarmuka pengguna dikembangkan dengan *Flask* untuk menampilkan data sensor dan kontrol perangkat. Sistem ini juga menggunakan *MySQL* sebagai basis data untuk menyimpan data *historis* sensor dan *log* kontrol perangkat.

3.2.6 Pengujian

Tahap pengujian dilakukan untuk memastikan bahwa seluruh fitur dalam sistem berfungsi dengan baik sesuai dengan spesifikasi dan bebas dari kesalahan atau bug yang signifikan.

Pengujian untuk menguji fungsionalitas sistem tanpa melihat struktur kode secara langsung. Selain itu, dilakukan pengujian oleh pengguna untuk memastikan bahwa sistem sesuai dengan kebutuhan pengguna di lapangan, khususnya dalam konteks *monitoring* dan pengendalian penyiraman dan pengkabutan di *Greenhouse*.

3.2.7 Penerapan

Tahap penerapan adalah proses penempatan sistem ke lingkungan sebenarnya, yaitu dengan melakukan *deploy* aplikasi ke *server* dan menghubungkannya dengan perangkat *IOT* di lapangan. Pada tahap ini, sistem yang telah dikembangkan mulai dijalankan dan dikonfigurasikan agar dapat berfungsi sesuai dengan kondisi operasional.

Proses penerapan mencakup pengunggahan kode program ke *server*, pengaturan koneksi *MQTT*, integrasi dengan basis data, serta pengujian awal untuk memastikan bahwa sistem dapat berjalan dengan baik. Penerapan dilakukan secara bertahap sesuai dengan pendekatan metode *Agile*, di mana setiap bagian sistem yang telah selesai dapat langsung diuji di lapangan.

3.2.8 Peluncuran

Setelah sistem dinyatakan layak berdasarkan hasil evaluasi, maka dilakukan tahap peluncuran. Tahap ini menandai bahwa sistem telah siap digunakan secara penuh oleh pengguna di lapangan.

Peluncuran dilakukan dengan memberikan akses ke seluruh fitur sistem, menyosialisasikan cara penggunaannya, serta memastikan bahwa semua komponen perangkat keras dan perangkat lunak berjalan dengan baik.

3.2.9 Pembuatan Laporan

Tahapan terakhir adalah penyusunan laporan hasil penelitian yang menguraikan secara detail tahapan-tahapan yang dijalankan dalam penelitian dan hasil yang di peroleh. Laporan ini bertujuan untuk secara sistematis menyampaikan informasi kepada pembaca tentang isi dari penelitian yang telah dilakukan.

BAB IV

ANALISIS DAN PERANCANGAN

4.1 Analisis

Analisis dilakukan sebagai langkah awal penelitian untuk mengidentifikasi kebutuhan-kebutuhan yang diperlukan dalam merancang sistem. Berikut adalah analisis yang telah penulis lakukan untuk membantu dalam perancangan sistem penyiraman dan pengkabutan otomatis pada tanaman berbasis *Internet of Things (IOT)*. Analisis ini mencakup pembahasan mengenai analisis masalah, analisis perangkat keras dan perangkat lunak.

4.1.1 Analisis Masalah

Langkah pertama pada tahap ini adalah mengidentifikasi permasalahan yang terjadi di lingkungan Avicenna *Greenhouse* terkait efisiensi dan efektivitas dalam proses penyiraman dan pengkabutan tanaman. Proses dilakukan dengan mengamati kondisi lapangan, di mana kegiatan penyiraman dan pengkabutan masih dilakukan secara manual tanpa memperhatikan parameter lingkungan secara *real-time*, seperti suhu, kelembaban udara, dan kelembaban tanah. Hal ini menyebabkan ketidak tepatan waktu dalam pemberian air atau kabut, yang dapat berdampak pada kualitas pertumbuhan tanaman. Selain itu, tidak adanya sistem monitoring dan kontrol jarak jauh menyulitkan pengguna untuk melakukan pengawasan dan pengambilan keputusan secara cepat.

4.1.2 Analisis *Hardware*

Untuk membangun sistem penyiraman dan pengkabutan otomatis pada tanaman di *Greenhouse*, diperlukan perangkat keras yang dapat mendeteksi kondisi lingkungan, mengontrol aktuator, serta mengirimkan data secara *real-time* ke *server*. Komponen-komponen ini saling terintegrasi melalui jaringan internet dan dikendalikan oleh sistem berbasis web. Berikut adalah perangkat keras yang digunakan dalam sistem:

- 1) Sensor *DHT11*, digunakan untuk membaca suhu dan kelembapan udara di dalam *Greenhouse* secara berkala.
- 2) *Soil Moisture Sensor*, berfungsi untuk mengukur kelembapan tanah guna menentukan tingkat kebutuhan air pada tanaman.
- 3) *Mini Water Pump DC 3–6V & Sanyo*, digunakan untuk mengalirkan air ke tanaman saat sistem memutuskan untuk melakukan penyiraman.
- 4) *Nozzle Sprayer*, digunakan untuk menghasilkan kabut air.
- 5) *Relay Module 2 Channel 5V*, berfungsi sebagai saklar elektronik untuk mengontrol *on/off* pompa air dan sprayer secara otomatis.
- 6) Adapter 5V/2A, menyediakan catu daya utama bagi mikrokontroler dan komponen lainnya.
- 7) Selang Plastik *PE (Polyethylene)*, digunakan untuk menyalurkan air dari pompa ke tanaman atau *Nozzle Sprayer*.
- 8) *Hotspot HP*, digunakan sebagai sumber koneksi internet agar *ESP32* dapat terhubung dengan *server* dan mengirim data secara real-time.
- 9) *Box Casing*, berfungsi sebagai wadah pelindung untuk seluruh rangkaian elektronik agar aman dari air dan gangguan luar.

4.1.3 Analisis Software

Untuk mendukung pengembangan sistem penyiraman dan pengabutan otomatis berbasis *Internet of Things (IOT)*, diperlukan sejumlah perangkat lunak yang berfungsi sebagai alat bantu dalam proses pemrograman, pengujian, pengambilan keputusan, dan pengelolaan data. Berikut adalah perangkat lunak yang digunakan dalam sistem:

- 1) *Arduino IDE* digunakan untuk memprogram mikrokontroler *ESP32* agar dapat membaca data sensor, mengaktifkan *Relay*, serta mengirim dan menerima data melalui koneksi *WiFi*.
- 2) *Python* digunakan sebagai bahasa pemrograman untuk mengimplementasikan algoritma *Naive Bayes*.
- 3) *Flask framework backend* berbasis *Python* yang digunakan untuk membangun aplikasi web dan integrasi antara *server* dengan perangkat *IOT*.

- 4) *MySQL* sistem manajemen basis data relasional yang digunakan untuk menyimpan data sensor, hasil prediksi, jadwal penyiraman, dan riwayat aksi.
- 5) *MQTT (Message Queuing Telemetry Transport)* protokol komunikasi yang digunakan untuk pertukaran data secara *real-time* antara *ESP32* dan *server*.
- 6) *Whatsapp* digunakan untuk mengirim notifikasi kepada pengguna terkait status penyiraman atau pengkabutan, baik otomatis maupun manual. Integrasi dilakukan menggunakan *library Whatsapp-web.js*, yang memungkinkan pengiriman pesan melalui antarmuka *Whatsapp Web* dengan bantuan *Puppeteer*. Pendekatan ini memungkinkan notifikasi *real-time* tanpa perlu akses *API* resmi *Whatsapp*.
- 7) *Visual Studio Code (VS Code)* sebagai editor kode utama yang digunakan dalam pengembangan program *backend*, algoritma, dan antarmuka.
- 8) *Fritzing* alat bantu visual untuk merancang dan mendokumentasikan rangkaian perangkat keras *IOT*.
- 9) *Draw.io* digunakan untuk membuat berbagai diagram seperti *UML* yang mendukung dokumentasi sistem.
- 10) *Figma* platform desain antarmuka pengguna (*UI*) yang digunakan untuk merancang tampilan halaman *Dashboard* sistem monitoring dan kontrol.

4.1.4 Analisis Pengguna

Analisis pengguna dalam sistem penyiraman dan pengkabutan otomatis ini berfokus pada siapa saja yang akan menggunakan sistem serta bagaimana interaksinya terhadap fitur-fitur yang tersedia. Pengguna utama sistem adalah pengelola *Greenhouse* yang bertanggung jawab dalam pemantauan dan perawatan tanaman. Pengguna dapat mengakses *Dashboard* untuk melihat data sensor secara *real-time*, mengatur jadwal penyiraman dan pengkabutan, serta memantau riwayat aktivitas yang dilakukan sistem secara otomatis maupun manual. Selain itu, pengguna juga akan menerima notifikasi melalui *Whatsapp* sebagai bentuk pemberitahuan terkait aksi yang telah dilakukan oleh sistem. Meskipun sistem ini bersifat otomatis, pengguna tetap memiliki kendali penuh untuk menyesuaikan pengaturan sesuai kebutuhan. Oleh karena itu, antarmuka sistem dirancang

dengan tampilan yang sederhana, informatif, dan mudah dioperasikan guna mendukung kenyamanan, keandalan, serta efektivitas dalam merawat tanaman di *Greenhouse* secara modern dan efisien.

4.1.5 User Interface

User Interface pada sistem penyiraman dan pengkabutan otomatis ini terdiri dari dua bagian utama, yaitu antarmuka perangkat fisik dan antarmuka digital berbasis web. Pembagian ini bertujuan untuk memfasilitasi interaksi pengguna secara langsung melalui perangkat *IOT* di lapangan, serta mendukung pengelolaan dan pemantauan sistem secara jarak jauh melalui *Dashboard* web yang responsif dan informatif.

Antarmuka web memungkinkan pengguna (*admin/pengelola Greenhouse*) untuk mengelola sistem secara menyeluruh melalui *Dashboard* Web. Fitur-fitur utama yang disediakan dalam tampilan web antara lain:

- a. *User Interface* Halaman *Login*
- b. *User Interface* Halaman *Dashboard* Utama
- c. *User Interface* Halaman Data Sensor
- d. *User Interface* Halaman Kontrol alat
- e. *User Interface* Halaman *Whatsapp*

4.1.6 Fitur - fitur

Fitur-fitur yang tersedia dalam sistem penyiraman dan pengkabutan otomatis ini dirancang untuk memberikan kemudahan kepada pengguna dalam memantau kondisi lingkungan tanaman serta mengontrol penyiraman dan pengkabutan secara efisien. Berikut adalah fitur-fitur yang disediakan dalam sistem:

- a. Menampilkan data suhu, kelembapan udara, dan kelembapan tanah secara *real-time*. Data ini ditampilkan dalam bentuk komponen *card* pada *Dashboard* untuk memudahkan pemahaman pengguna terhadap kondisi tanaman.
- b. Sistem secara otomatis memproses data yang diperoleh dari sensor menggunakan algoritma *Naive Bayes* untuk menentukan apakah perlu

dilakukan penyiraman, pengkabutan, atau tidak melakukan tindakan sama sekali. Setiap kali algoritma dijalankan, hasil prediksi beserta data sensornya akan disimpan secara otomatis ke dalam *database* sebagai *log* historis. Selain itu, sistem juga menyediakan opsi untuk menghapus data prediksi yang telah tersimpan, sehingga pengguna dapat mengelola riwayat data sesuai kebutuhan.

- c. Pengguna dapat mengaktifkan atau menonaktifkan penyiraman dan pengkabutan secara manual melalui *Dashboard* jika ingin melakukan kontrol secara langsung.
- d. Pengguna dapat mengatur jadwal penyiraman atau pengkabutan pada waktu tertentu secara berulang setiap hari.
- e. memungkinkan sistem untuk secara otomatis menghentikan proses penyiraman atau pengkabutan ketika nilai sensor telah mencapai batas tertentu yang telah ditentukan.
- f. Sistem mencatat setiap tindakan penyiraman dan pengkabutan, baik yang dilakukan secara otomatis maupun manual. Riwayat ini dapat ditinjau untuk evaluasi dan dokumentasi.
- g. Sistem mengirimkan pesan otomatis ke *Whatsapp* pengguna setiap kali terjadi penyiraman atau pengkabutan, lengkap dengan informasi waktu dan jenis aksi yang dilakukan.
- h. *Admin* dapat menambahkan dan menghapus nomor *Whatsapp* pengguna melalui halaman pengelolaan *Whatsapp*.

4.1.7 Analisis Data

Untuk mendukung perancangan sistem penyiraman dan pengkabutan otomatis berbasis *IOT* dan web, penulis memerlukan beberapa jenis data sebagai bahan analisis dan dasar pengembangan sistem. Data-data ini diperoleh dari hasil pembacaan sensor, proses prediksi menggunakan algoritma *Naive Bayes*, serta interaksi pengguna terhadap sistem melalui *Dashboard* dan notifikasi *Whatsapp*. Berikut dibawah ini adalah jenis data yang digunakan dan dianalisis dalam penelitian, yaitu sebagai berikut:

a. Data Sensor

Data ini berasal dari pembacaan sensor suhu dan kelembapan udara, serta sensor kelembapan tanah. Data ini dikumpulkan secara berkala dan digunakan sebagai input utama dalam proses pengambilan keputusan otomatis oleh sistem. Analisis terhadap data ini dilakukan untuk memahami kondisi lingkungan di *Greenhouse* serta menguji konsistensi pembacaan sensor.

b. Data Prediksi Sistem

Data ini dihasilkan dari pemrosesan algoritma *Naive Bayes* yang menentukan status tindakan: penyiraman, pengkabutan, atau tidak melakukan aksi. Setiap hasil prediksi disimpan ke dalam *database* untuk dianalisis lebih lanjut, baik dalam konteks validasi algoritma maupun evaluasi akurasi sistem terhadap kondisi sebenarnya.

c. Data Interaksi Pengguna

Data ini mencakup aktivitas pengguna melalui *Dashboard*, seperti pengaturan jadwal, kontrol manual, serta penghapusan data prediksi. Selain itu, interaksi berupa tanggapan terhadap notifikasi *Whatsapp* juga menjadi bagian dari data yang dianalisis untuk menilai sejauh mana sistem memberikan kemudahan dan kontrol bagi pengguna.

d. Data Kinerja Sistem

Data ini meliputi waktu respons sistem dalam membaca sensor, menjalankan prediksi, menyalakan atau mematikan perangkat (pompa atau *sprayer*), serta kecepatan pengiriman notifikasi ke *Whatsapp*. Data ini dianalisis untuk mengevaluasi kestabilan dan efisiensi sistem secara keseluruhan, termasuk keterandalan koneksi antara *ESP32* dan *server*.

4.1.8 Analisis Biaya

Untuk mendukung pembuatan sistem penyiraman dan pengkabutan otomatis berbasis *Internet of Things (IOT)* dengan metode *agile* dan algoritma *Naive Bayes*, penulis melakukan analisis terhadap kebutuhan biaya yang mencakup pengadaan perangkat keras, perangkat lunak, serta komponen pendukung lainnya yang digunakan dalam proses pengembangan dan implementasi sistem.

Tabel 4. 1 Analisis Biaya

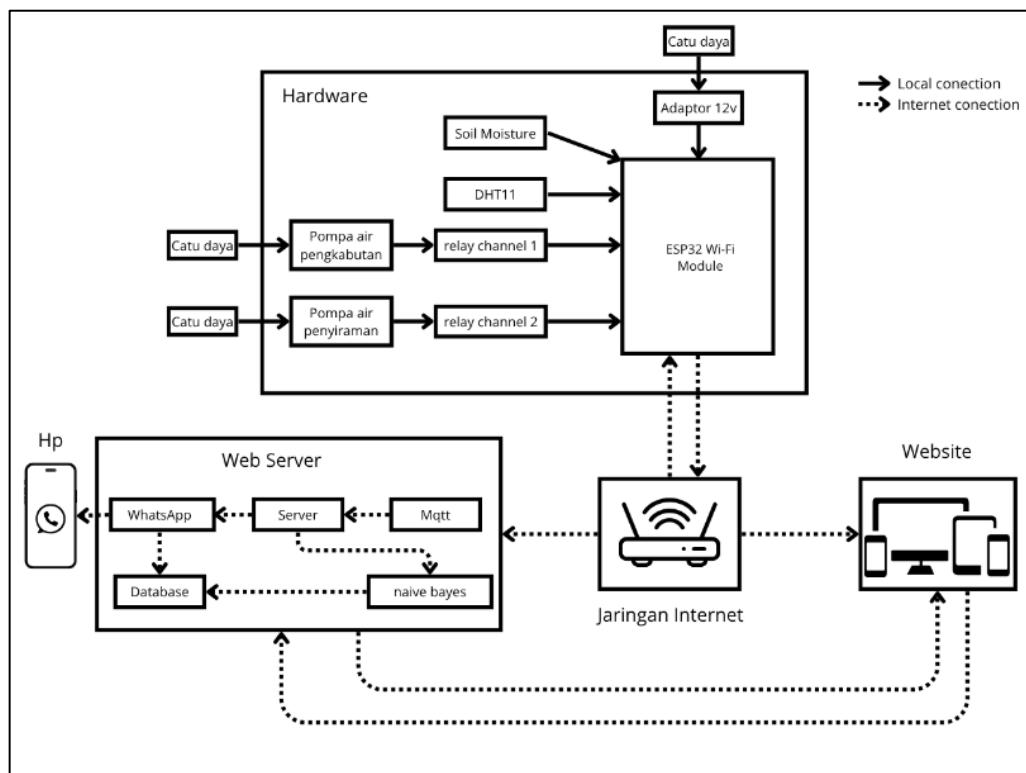
| No | Jenis Kebutuhan | Biaya |
|--------------|--|----------------|
| 1 | Biaya ATK | Rp. 2.000.000 |
| 2 | <i>DHT11</i> | Rp. 10.000 |
| 3 | 2 Channel <i>Relay</i> | Rp. 23.000 |
| 4 | Kabel USB | Rp. 15.000 |
| 5 | Kabel AWG 24 3 roll (Hitam, Hijau, Merah) | Rp. 80.000 |
| 6 | Tang Kabel | Rp. 34.000 |
| 7 | <i>Heatshrink Tube</i> Isolasi bakar | Rp. 13.000 |
| 8 | Adaptor DC 12V | Rp. 8.000 |
| 9 | <i>Expansion Board ESP32</i> 38 pin | Rp. 29.000 |
| 10 | <i>ESP32 DOIT</i> 38 pin | Rp. 66.500 |
| 11 | <i>Soil Moisture sensor</i> | Rp. 9.000 |
| 12 | <i>Duppont Connector Female</i> 100pcs | Rp. 11.000 |
| 13 | Selang PE 6 MM 20 meter | Rp. 39.000 |
| 14 | <i>Mist Nozzle Sprayer</i> 15pcs | Rp. 21.000 |
| 15 | Stop kontak steker saklar <i>on off</i> 2pcs | Rp. 23.000 |
| 16 | Avometer Digital | Rp. 43.000 |
| 17 | <i>Box Componen</i> | Rp. 17.000 |
| 18 | Baut sekrup 20pc | Rp. 3.000 |
| 19 | Pompa air sanyo | Rp. 200.000 |
| 20 | Pompa air aquarium | Rp. 80.000 |
| 21 | Kabel Listrik hitam 7mm 4 meter | Rp. 20.000 |
| 22 | Alat set Bor | Rp. 362.000 |
| 23 | Lem tembak + <i>glue gun</i> | Rp. 63.000 |
| 24 | Set solder Listrik + timah solder | Rp. 140.000 |
| 25 | Token Listrik | Rp. 50.000 |
| 26 | Kuota Internet | Rp. 100.000 |
| 27 | <i>Hosting VPS</i> 4 core 4 gb ram 100gb ssd 3 bln | Rp. 600.000 |
| 28 | Transportasi Kendaraan | Rp. 500.000 |
| 29 | Pembuatan Aplikasi | Rp. 5.000.0000 |
| 31 | Selang Air | Rp. 50.000 |
| 32 | Pipa paralon | Rp. 100.000 |
| Total Jumlah | | Rp 9.809.500 |

4.2 Perancangan

Sebelum memasuki tahap pembuatan sistem penyiraman dan pengkabutan otomatis berbasis *IOT*, diperlukan perancangan sistem yang terencana dan menyeluruh. Tahapan ini mencakup pembuatan blok diagram untuk menggambarkan alur kerja sistem penyiraman dan pengkabutan secara otomatis. Selanjutnya, dilakukan perancangan rangkaian perangkat keras menggunakan aplikasi *Fritzing* untuk memvisualisasikan koneksi antar komponen seperti sensor, mikrokontroler, pompa, dan *sprayer*. Pada sisi perangkat lunak, sistem dirancang dalam bentuk *Dashboard* web, yang dimodelkan menggunakan *Unified Modeling Language (UML)* melalui penyusunan *Use Case Diagram*, *Activity Diagram*, dan *Class Diagram* sebagai acuan dalam pengembangan fungsionalitas dan alur sistem.

4.2.1 Blok Diagram Alat dan Sistem

Perancangan ini menunjukkan blok diagram konsep yang menggambarkan integrasi antara perangkat keras dan perangkat lunak.



Gambar 4. 1 Blok diagram alat dan sistem

Penjelasan mengenai gambar diagram di atas dapat secara terinci diuraikan dalam sebuah tabel deskripsi yang telah disediakan:

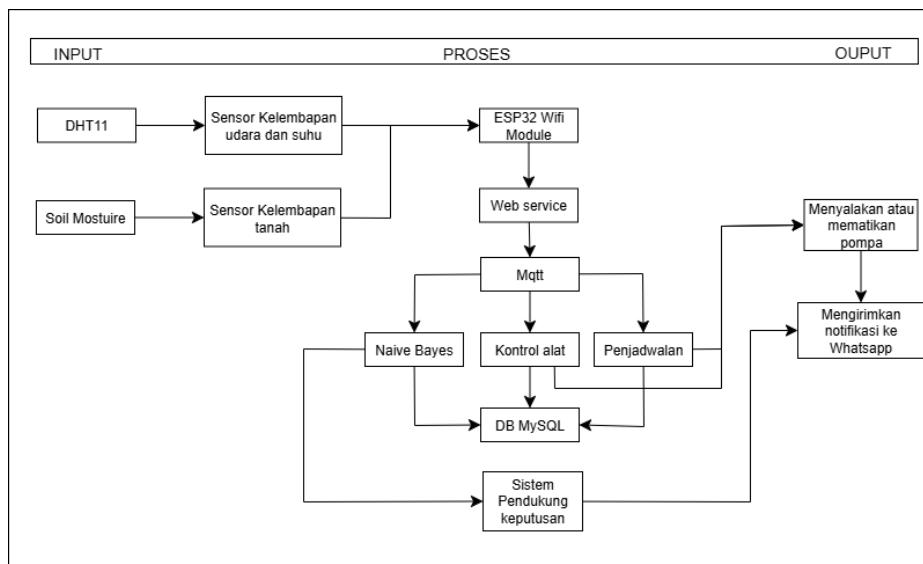
Tabel 4. 2 Tabel diagram konsep

| Lingkup | Deskripsi |
|-------------------|---|
| <i>Hardware</i> | Perangkat keras terdiri dari <i>ESP32 WiFi</i> Modul sebagai pusat kendali, yang terhubung dengan sensor <i>DHT11</i> (suhu dan kelembaban udara) dan <i>Soil Moisture</i> (kelembaban tanah). Modul <i>ESP32</i> juga mengontrol dua <i>Relay channel</i> , masing-masing terhubung ke pompa air penyiraman dan pompa pengkabutan, dengan catu daya terpisah. |
| Jaringan Internet | Modul <i>ESP32</i> terhubung ke jaringan internet melalui koneksi jaringan internet, memungkinkan pertukaran data antara perangkat fisik dan sistem <i>backend (web server serta website)</i> . Komunikasi menggunakan protokol <i>MQTT</i> dan jalur internet digunakan untuk proses pengiriman dan penerimaan data sensor serta kontrol pompa. |
| <i>Web server</i> | <i>Web server</i> berfungsi sebagai pusat logika sistem. Data dari sensor yang dikirim melalui <i>MQTT</i> diolah oleh <i>server backend</i> , yang memanfaatkan algoritma <i>Naive Bayes</i> untuk melakukan prediksi tindakan (siram, kabut, atau tidak). Seluruh data disimpan dalam <i>database</i> untuk kebutuhan histori dan pengambilan keputusan. Selain itu, <i>server</i> juga mengirimkan notifikasi otomatis ke <i>Whatsapp</i> pengguna setiap kali sistem melakukan penyiraman atau pengkabutan, baik secara otomatis maupun manual. |
| <i>Website</i> | <i>Website</i> menampilkan data sensor, histori aksi, dan menyediakan kontrol manual untuk penyiraman dan |

| | |
|--|---|
| | pengkabutan. Komunikasi dilakukan secara <i>real-time</i> melalui integrasi dengan sistem <i>MQTT</i> dan <i>database</i> . |
|--|---|

4.2.2 Diagram Alur Alat dan Sistem

Perancangan ini menggambarkan alur kerja sistem penyiraman dan pengkabutan otomatis pada tanaman, mulai dari proses pembacaan data sensor oleh *ESP32*, hingga pengambilan keputusan penyiraman atau pengkabutan menggunakan algoritma *Naive Bayes* di *web server* dan pengguna juga menerima notifikasi status aksi secara *real-time* melalui *Whatsapp*.



Gambar 4. 2 Diagram alur alat dan sistem

Penjelasan mengenai gambar diagram di atas dapat secara terinci diuraikan dalam sebuah tabel deskripsi yang telah disediakan:

Tabel 4. 3 Deskripsi Diagram Alur Alat dan Sistem

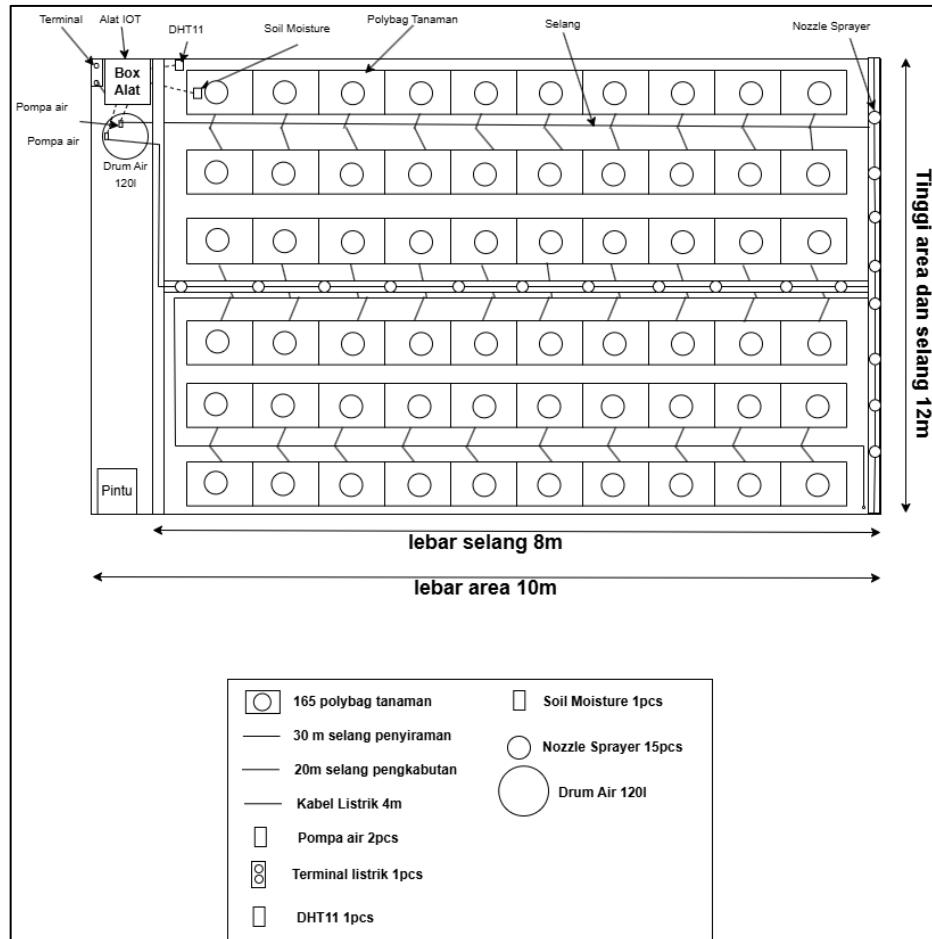
| Komponen | Kategori | Deskripsi |
|----------|----------|--|
| DHT11 | Input | Sensor yang digunakan untuk mengukur suhu dan kelembapan udara di lingkungan <i>Greenhouse</i> . |

| | | |
|--------------------------|--------------|--|
| <i>Soil Moisture</i> | <i>Input</i> | Sensor yang digunakan untuk mendekripsi tingkat kelembapan tanah guna menentukan kebutuhan penyiraman. |
| Sensor Kelembapan & Suhu | Proses | Data sensor yang dibaca dari <i>DHT11</i> . |
| Sensor Kelembapan Tanah | Proses | Data sensor yang dibaca dari <i>Soil Moisture</i> |
| <i>ESP32 WiFi Modul</i> | Proses | Mikrokontroler yang berfungsi mengirim data sensor ke <i>web service</i> dan menerima instruksi penyiraman atau pengkabutan. |
| <i>Web Service</i> | Proses | <i>Server backend</i> yang mengelola komunikasi data dari <i>ESP32</i> dan mendistribusikannya ke berbagai modul logika sistem. |
| <i>MQTT</i> | Proses | Protokol komunikasi ringan yang digunakan untuk mengirim data sensor dari <i>ESP32</i> ke <i>server</i> secara <i>real-time</i> dan mengirim instruksi kembali ke perangkat. |
| <i>Naive Bayes</i> | Proses | Algoritma klasifikasi yang menganalisis data sensor untuk memutuskan tindakan penyiraman, pengkabutan, atau tidak bertindak. |
| Kontrol Alat | Proses | Modul yang mengeksekusi hasil prediksi untuk mengontrol pompa penyiraman atau pengkabutan sesuai hasil keputusan. |
| Penjadwalan | Proses | Modul yang memungkinkan |

| | | |
|---|---------------|--|
| | | sistem melakukan penyiraman atau pengkabutan otomatis sesuai waktu yang telah ditentukan. |
| <i>DB MySQL</i> | Proses | Basis data yang menyimpan semua nomor <i>Whatsapp</i> , riwayat aksi, jadwal, dan hasil prediksi untuk keperluan analisis dan dokumentasi. |
| Sistem Pendukung Keputusan | Proses | Komponen logika tambahan yang memanfaatkan data historis dalam <i>database</i> untuk menyempurnakan proses pengambilan keputusan sistem. |
| Menyalakan & Mematikan Pompa | <i>Output</i> | Hasil akhir berupa aksi langsung sistem untuk mengaktifkan atau menonaktifkan pompa berdasarkan kontrol manual maupun penjadwalan. |
| Mengirimkan Notifikasi ke <i>Whatsapp</i> | <i>Output</i> | Sistem mengirimkan pemberitahuan otomatis ke <i>Whatsapp</i> pengguna mengenai status penyiraman atau pengkabutan. |

4.2.3 Denah *Greenhouse*

Tata letak sistem penyiraman dan pengkabutan pada penelitian ini dirancang dengan mempertimbangkan efisiensi distribusi air serta kestabilan dalam pembacaan sensor di dalam *Greenhouse*. Penempatan perangkat dilakukan secara strategis agar mendukung kinerja sistem secara optimal, mulai dari pengambilan data, pemrosesan keputusan, hingga eksekusi penyiraman atau pengkabutan.

Gambar 4. 3 Denah *Greenhouse*

Tata letak sistem penyiraman dan pengkabutan otomatis ini diterapkan pada *Greenhouse* dengan luas area 10meter x 12meter, yang berisi 165 *polybag* tanaman tersusun dalam beberapa baris sejajar. Desain tata letak dirancang untuk memastikan distribusi air dan kabut yang merata, serta memudahkan instalasi dan pemeliharaan sistem secara keseluruhan. Beberapa komponen penting yang ditata secara strategis meliputi:

Tabel 4. 4 Deskripsi Denah *Greenhouse*

| Komponen | Deskripsi |
|---------------------|--|
| Box Alat & Alat IOT | Diletakkan di sisi kiri atas area <i>Greenhouse</i> dekat dengan terminal listrik, berfungsi sebagai pusat kontrol yang berisi <i>ESP32 WiFi</i> modul, <i>Relay</i> , dan konektor sensor |

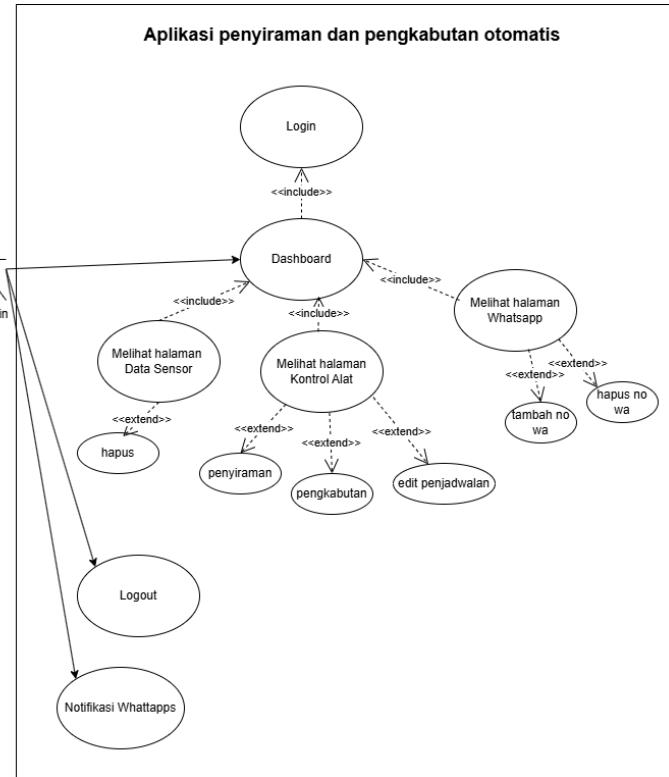
| | |
|---|---|
| Sensor <i>DHT11 & Soil Moisture</i> | <i>DHT11</i> diletakkan di dekat box alat untuk mengukur suhu dan kelembapan udara secara sentral. Sensor <i>Soil Moisture</i> ditanam pada salah satu baris tanaman untuk mewakili kondisi kelembapan tanah dari area tanam |
| Pompa Air & Drum 120 Liter | Dua pompa air diposisikan di dekat drum air 120L pada sisi kiri area tanam. Satu pompa digunakan untuk penyiraman, dan satu lagi untuk pengkabutan. Keduanya terhubung ke jaringan pipa melalui selang |
| Jaringan Selang dan <i>Nozzle</i> | Sistem selang penyiraman sepanjang 30meter dan selang pengkabutan sepanjang 20meter ditata memanjang mengikuti barisan tanaman. 15 <i>nozzle sprayer</i> dipasang merata di titik-titik strategis di atas tanaman untuk menyebarkan air kabut secara efisien dan merata. Lebar penempatan selang adalah 8 meter, dengan panjang bentangan mengikuti tinggi area 12meter |
| Terminal Listrik & Kabel | Satu terminal listrik diletakkan di bagian pojok atas dekat box alat untuk menyuplai daya ke pompa dan mikrokontroler. Sistem menggunakan 4meter kabel listrik yang ditata dengan rapi mengikuti sisi dalam <i>Greenhouse</i> agar aman dan tidak mengganggu jalur tanam |

4.2.4 Unified Modeling Language (UML)

Untuk memvisualisasikan proses dan struktur sistem secara konseptual, digunakan pendekatan *Unified Modeling Language (UML)*. *UML* berfungsi untuk memodelkan interaksi antara pengguna dan sistem, alur kegiatan dalam sistem. Pada tahap ini ditampilkan beberapa jenis diagram *UML*, yaitu:

1. Use Case Diagram

Use Case Diagram berikut menggambarkan interaksi antara aktor dengan sistem penyiraman dan pengkabutan otomatis berbasis *IOT* di *Greenhouse*. Diagram ini dirancang untuk memperlihatkan fungsi-fungsi utama yang dapat dilakukan oleh masing-masing aktor.



Gambar 4. 4 Use Case Diagram

Penjelasan mengenai gambar diagram di atas dapat secara terinci diuraikan dalam sebuah tabel deskripsi yang telah disediakan:

Tabel 4. 5 Deskripsi Aktor

| Aktor | Deskripsi |
|--------------|--|
| <i>Admin</i> | <i>Admin</i> merupakan pengguna utama sistem yang memiliki akses penuh terhadap seluruh fitur aplikasi penyiraman dan pengkabutan otomatis. <i>Admin</i> dapat melakukan <i>Login</i> untuk masuk ke <i>Dashboard</i> , melihat data sensor, serta menghapus data yang tidak diperlukan. <i>Admin</i> juga dapat mengontrol penyiraman dan pengkabutan secara manual, mengatur jadwal otomatisasi alat, dan mengelola daftar nomor <i>WhatsApp</i> yang digunakan untuk menerima notifikasi. Selain itu, <i>admin</i> dapat menambahkan atau menghapus nomor <i>WhatsApp</i> , serta menerima pemberitahuan otomatis dari sistem mengenai status penyiraman atau pengkabutan yang telah dilakukan. Setelah |

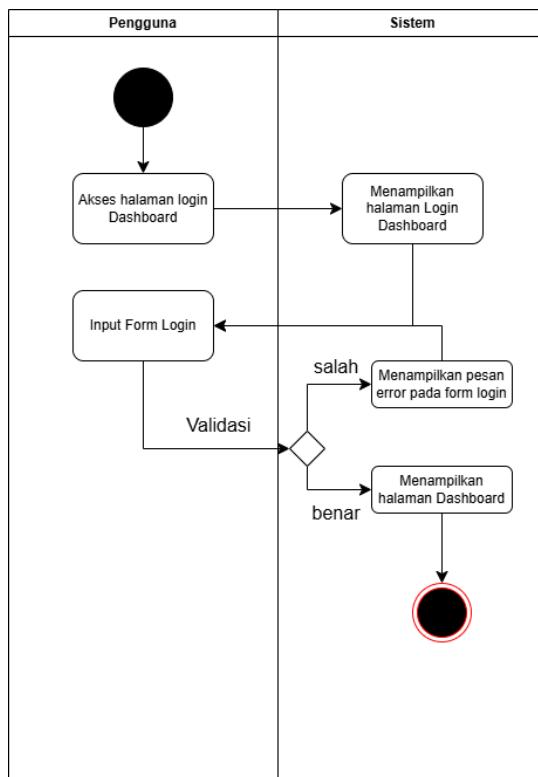
| | |
|--|--|
| | seluruh aktivitas selesai, <i>admin</i> dapat melakukan <i>Logout</i> dari sistem. |
|--|--|

2. Activity Diagram

Activity diagram digunakan untuk menggambarkan urutan aktivitas dalam suatu sistem serta alur kerja dari satu proses ke proses lainnya.

a. Activity Login

Dalam *Activity Login*, menggambarkan tahapan aktivitas pada saat melakukan *Login*, sebagaimana yang tergambar pada gambar di bawah ini:



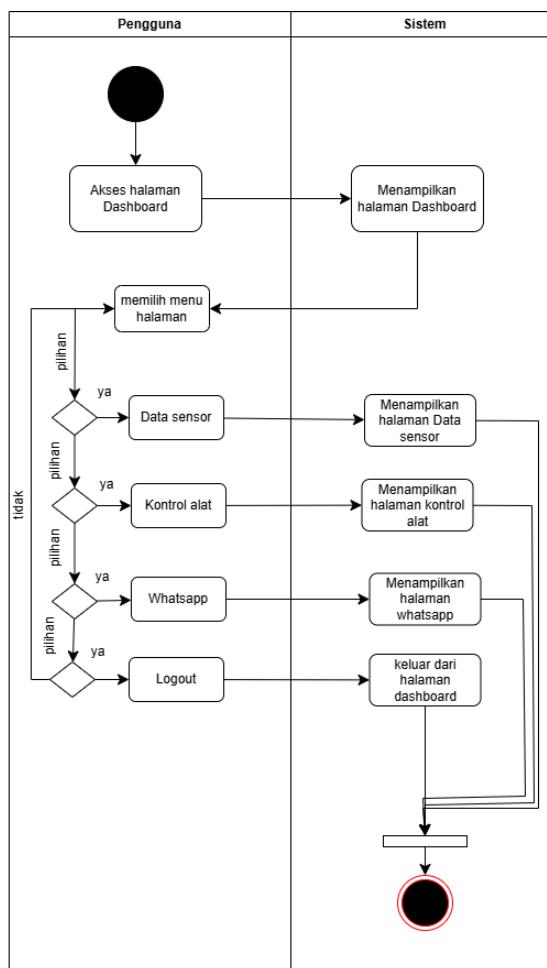
Gambar 4. 5 *Activity Login*

Gambar tersebut menunjukkan *activity diagram* yang menjelaskan proses pengguna saat melakukan *Login* ke halaman *Dashboard*. Proses diawali ketika pengguna mengakses halaman *Login* melalui aplikasi. Sistem kemudian menampilkan form *Login* yang memungkinkan pengguna untuk memasukkan informasi autentikasi, seperti *email* atau *username* dan *password*. Setelah

pengguna mengisi form dan mengirimkannya, sistem melakukan validasi terhadap data yang dimasukkan. Jika data yang dimasukkan salah, sistem akan menampilkan pesan error pada form *Login* dan pengguna diminta untuk mengulang proses input. Namun, jika data valid, sistem akan menampilkan halaman *Dashboard* sebagai tanda bahwa proses *Login* berhasil.

b. *Activity Dashboard*

Dalam *Activity Dashboard*, menggambarkan tahapan aktivitas pada saat masuk *Dashboard*, sebagaimana yang tergambar pada gambar di bawah ini:



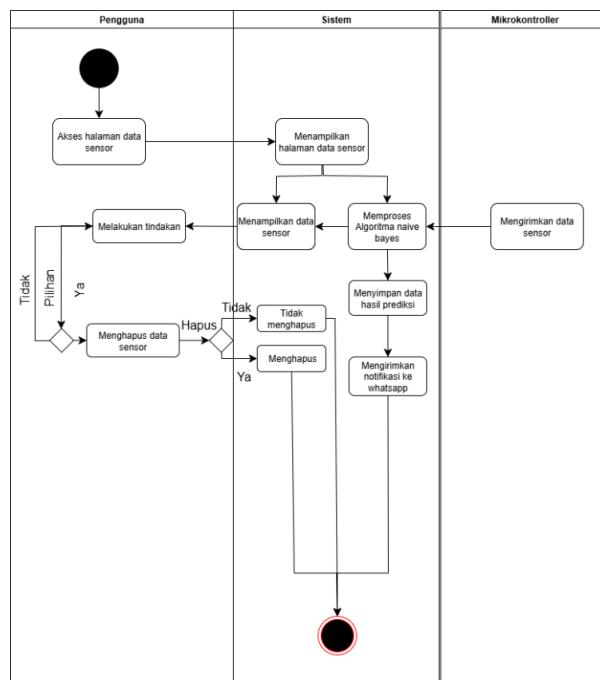
Gambar 4. 6 *Activity Dashboard*

Gambar tersebut menunjukkan *activity* diagram yang menggambarkan alur aktivitas pengguna saat mengakses dan berinteraksi dengan halaman *Dashboard* sistem. Proses diawali ketika pengguna mengakses halaman *Dashboard*. Sistem kemudian menampilkan halaman *Dashboard* sebagai tampilan awal. Selanjutnya,

pengguna dapat memilih berbagai menu halaman yang tersedia. Jika pengguna memilih menu Data Sensor, maka sistem akan menampilkan halaman Data Sensor. Jika pengguna memilih menu Kontrol Alat, sistem akan menampilkan halaman kontrol alat. Jika pengguna memilih menu *Whatsapp*, sistem akan menampilkan halaman *Whatsapp*. Sementara itu, jika pengguna memilih opsi *Logout*, maka sistem akan keluar dari halaman *Dashboard* dan proses berakhir.

c. Activity *Dashboard* Data Sensor

Dalam *Activity Dashboard* data sensor, menggambarkan tahapan aktivitas pada saat masuk *Dashboard* data sensor, pada gambar di bawah ini:



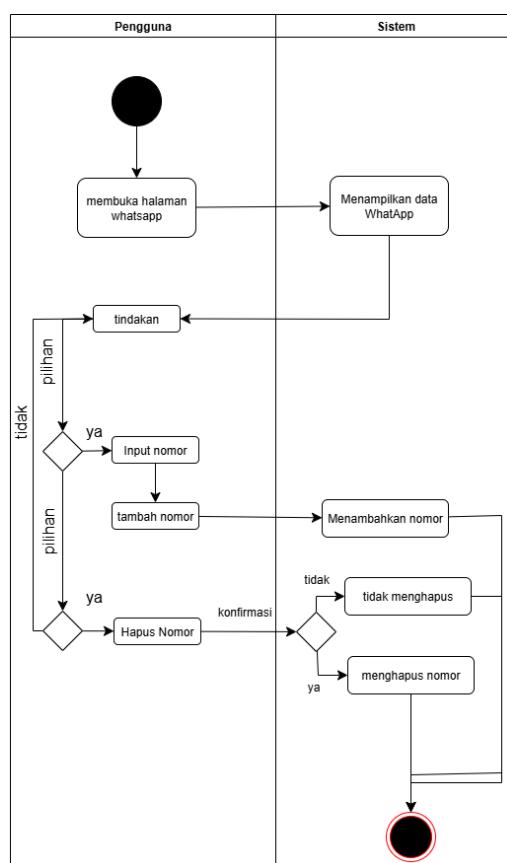
Gambar 4. 7 *Activity Dashboard* Data Sensor

Proses diawali ketika pengguna mengakses halaman data sensor. Sistem kemudian menampilkan halaman yang berisi data sensor yang telah dikirimkan sebelumnya oleh mikrokontroler. Setelah data ditampilkan, sistem langsung memproses data tersebut menggunakan algoritma *Naive Bayes* untuk menghasilkan prediksi. Hasil dari proses prediksi kemudian disimpan oleh sistem, dan sistem juga dapat mengirimkan notifikasi ke *Whatsapp*. Selanjutnya, pengguna dapat memilih untuk melakukan tindakan, seperti menghapus data sensor. Jika pengguna memilih untuk menghapus data, sistem akan menanyakan

kembali konfirmasi penghapusan. Jika pengguna menyetujui penghapusan, sistem akan menghapus data tersebut. Jika tidak, maka data tetap disimpan.

d. *Activity Whatsapp*

Dalam *Activity Whatsapp*, menggambarkan tahapan aktivitas pada saat masuk *Whatsapp*, pada gambar di bawah ini:



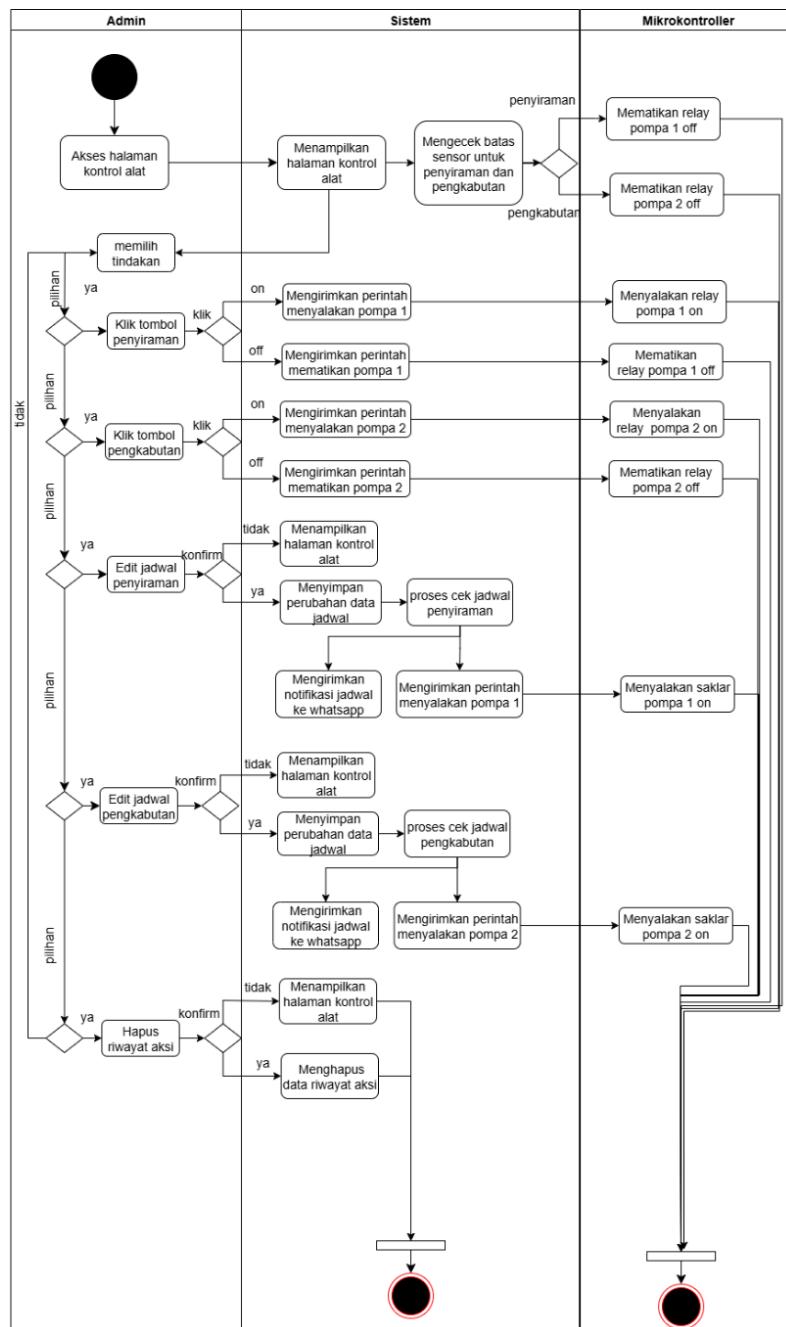
Gambar 4. 8 *Activity Whatsapp*

Gambar tersebut menunjukkan *activity* diagram yang menggambarkan proses interaksi pengguna dengan halaman *Whatsapp*. Proses diawali ketika pengguna membuka halaman *Whatsapp*. Setelah itu, sistem akan menampilkan data *Whatsapp* yang telah tersimpan sebelumnya. Pengguna kemudian dapat melakukan tindakan, seperti menambahkan nomor *Whatsapp* baru atau menghapus nomor yang sudah ada. Jika pengguna memilih untuk menambahkan nomor, maka pengguna perlu menginput nomor terlebih dahulu dan memilih opsi “tambah nomor”. Setelah itu, sistem akan menyimpan nomor tersebut ke dalam *database*. Selain itu, pengguna juga dapat memilih untuk menghapus nomor

Whatsapp. Jika opsi ini dipilih, sistem akan meminta konfirmasi terlebih dahulu. Jika pengguna mengonfirmasi penghapusan, sistem akan menghapus nomor tersebut. Jika tidak, maka sistem tidak akan menghapusnya.

e. Activity Kontrol Alat

Dalam *Activity* kontrol alat, menggambarkan tahapan aktivitas pada saat masuk kontrol alat, pada gambar di bawah ini:



Gambar 4. 9 *Activity* Kontrol Alat

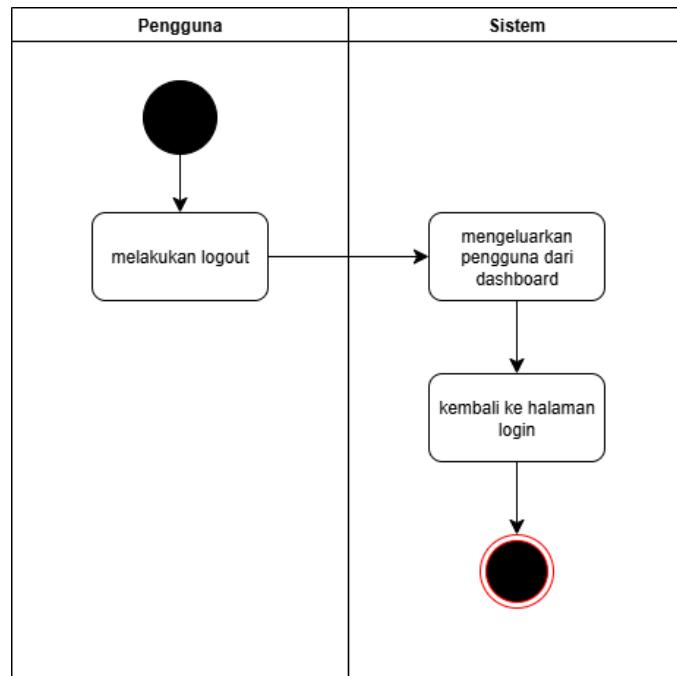
Gambar tersebut menunjukkan *activity diagram* yang menggambarkan alur interaksi antara *admin*, sistem, dan mikrokontroler dalam proses pengendalian alat penyiraman dan pengkabutan. Aktivitas dimulai ketika *admin* mengakses halaman kontrol alat. Setelah itu, sistem akan menampilkan halaman kontrol serta secara otomatis melakukan pengecekan batas sensor untuk menentukan apakah kondisi saat ini harus menonaktifkan proses penyiraman atau pengkabutan. Jika nilai sensor berada dalam batas tertentu, sistem akan mengirimkan perintah kepada mikrokontroler untuk mematikan *Relay* pompa penyiraman dan pengkabutan.

Selanjutnya, *admin* dapat memilih tindakan yang ingin dilakukan melalui halaman kontrol. Apabila *admin* mengklik tombol penyiraman dan memilih mode "on", maka sistem akan mengirimkan perintah untuk menyalakan *Relay* pompa penyiraman ke mikrokontroler. Sebaliknya, jika *admin* memilih mode "off", maka sistem akan memerintahkan mikrokontroler untuk mematikan *Relay* pompa tersebut. Hal serupa juga berlaku pada tombol pengkabutan, di mana sistem akan mengirimkan perintah untuk mengaktifkan atau menonaktifkan *Relay* pompa pengkabutan sesuai dengan perintah yang diberikan oleh *admin*.

Selain itu, *admin* juga memiliki opsi untuk mengedit jadwal penyiraman maupun pengkabutan. Ketika *admin* memilih opsi tersebut, sistem akan menampilkan halaman pengaturan jadwal dan menunggu konfirmasi perubahan dari *admin*. Setelah data jadwal disimpan, sistem akan memproses pengecekan terhadap waktu penyiraman atau pengkabutan yang telah dijadwalkan. Jika sesuai, sistem akan mengirimkan notifikasi melalui *Whatsapp* dan mengirimkan perintah ke mikrokontroler untuk menyalakan saklar pompa sesuai dengan jadwal yang telah ditentukan. *Admin* juga dapat melakukan penghapusan terhadap data riwayat aksi yang tercatat. Jika *admin* mengonfirmasi tindakan tersebut, maka sistem akan menghapus data riwayat aksi dari basis data dan menampilkan Kembali ke halaman kontrol alat sebagai penutup proses. Secara keseluruhan, diagram ini menggambarkan bagaimana *admin* dapat mengontrol perangkat secara manual maupun otomatis, serta bagaimana sistem berperan sebagai penghubung yang mengelola perintah dan respon antara antarmuka pengguna dan perangkat mikrokontroler dalam sistem pengendalian alat untuk penyiraman dan pengkabutan berbasis *Internet of Things*.

f. *Activity Admin Logout*

Dalam *admin Logout*, menggambarkan tahapan aktivitas pada saat melakukan *Logout*, pada gambar di bawah ini:



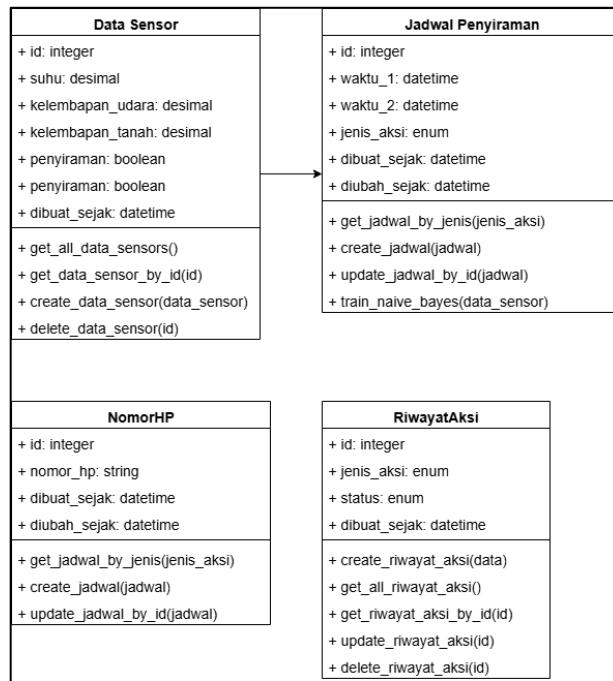
Gambar 4. 10 *Activity Logout*

Gambar tersebut menunjukkan *activity* diagram yang menggambarkan tahapan aktivitas saat pengguna, dalam hal ini *admin*, melakukan proses logout dari sistem. Proses diawali ketika pengguna memilih opsi logout melalui antarmuka aplikasi. Setelah perintah logout dikirimkan, sistem segera memproses permintaan tersebut dengan mengeluarkan pengguna dari sesi aktif dan menghentikan akses ke halaman *Dashboard*. Setelah pengguna berhasil dikeluarkan dari *Dashboard*, sistem secara otomatis akan mengarahkan pengguna kembali ke halaman *Login*.

Langkah ini menandakan bahwa sesi autentikasi telah berakhir, sehingga untuk mengakses kembali sistem, pengguna perlu melakukan proses *Login* ulang. Diagram ini mencerminkan bagaimana sistem menjaga keamanan akses pengguna dengan memastikan bahwa setiap sesi ditutup secara eksplisit melalui proses *logout* yang *valid*. Hal ini penting untuk mencegah akses tidak sah terhadap data atau fitur sistem jika perangkat digunakan oleh orang lain setelah sesi berakhir.

3. Class Diagram

Class Diagram berikut menggambarkan struktur sistem dan hubungan antara kelas-kelas yang terlibat dalam sistem penyiraman dan pengkabutan otomatis berbasis *IOT* di *Greenhouse*.



Gambar 4. 11 *Class Diagram*

Gambar tersebut menunjukkan *Class Diagram* yang menggambarkan struktur utama sistem penyiraman dan pengkabutan otomatis berbasis *IOT* yang digunakan di *Greenhouse*. Diagram ini terdiri dari empat kelas utama, yaitu DataSensor, JadwalPenyiraman, NomorHP, dan RiwayatAksi, yang masing-masing memiliki atribut dan fungsi spesifik sesuai dengan tanggung jawabnya. Kelas DataSensor berfungsi untuk merepresentasikan data hasil pembacaan dari berbagai sensor seperti suhu, kelembapan udara, dan kelembapan tanah. Kelas ini juga mencatat status penyiraman dan pengkabutan serta waktu pencatatan data. Fungsi-fungsi di dalamnya meliputi pengambilan seluruh data sensor, pengambilan data berdasarkan ID, pembuatan data baru, dan penghapusan data sensor.

Kelas JadwalPenyiraman digunakan untuk menyimpan dan mengatur jadwal penyiraman maupun pengkabutan. Kelas ini memiliki atribut waktu, jenis aksi

(penyiraman atau pengkabutan), serta informasi waktu pembuatan dan pembaruan data. Terdapat relasi antara kelas ini dengan DataSensor, yang digunakan dalam proses pelatihan algoritma *Naive Bayes* guna menentukan jadwal penyiraman otomatis berdasarkan pola data sensor yang telah direkam sebelumnya. Fungsi pada kelas ini mencakup pengambilan jadwal berdasarkan jenis aksi, pembuatan dan pembaruan jadwal, serta pelatihan model prediksi. Kelas NomorHP menyimpan daftar nomor *Whatsapp* yang digunakan untuk menerima notifikasi sistem. Atribut utama di dalamnya adalah nomor HP, serta waktu pembuatan dan pembaruan data. Kelas ini menyediakan fungsi untuk mengelola data nomor berdasarkan jenis aksi, serta untuk membuat dan memperbarui data nomor. Kelas RiwayatAksi mencatat semua aktivitas atau perintah yang dilakukan pada sistem, baik secara manual oleh *admin* maupun otomatis oleh sistem. Setiap riwayat berisi jenis aksi, status, dan waktu pencatatan. Fungsi di dalamnya memungkinkan sistem untuk mencatat, memperbarui, mengambil, dan menghapus data riwayat.

4.2.5 Struktur Tabel

Tabel-tabel yang terdapat dalam basis data yang digunakan dalam aplikasi *IOT* sebagai pengukur kinerja proses penyiraman dan pengkabutan di avicenna *Greenhouse* adalah sebagai berikut

- Perancangan tabel data_sensor

Tabel 4. 6 Tabel data sensor

| Field | Type | Size | Indeks | Deskripsi |
|------------------|-----------------|------|--------------------|---|
| id | Integer | 10 | <i>Primary key</i> | Id <i>primary key</i> tabel data_sensor |
| suhu | Float | - | - | Kolom untuk data sensor suhu |
| kelembapan udara | Float | - | - | Kolom untuk data sensor udara |
| kelembapan tanah | Float | - | - | Kolom untuk data sensor udara |
| penyiraman | Integer | 1 | - | Kolom untuk data aksi penyiraman |
| pengkabutan | Integer | 1 | - | Kolom untuk data aksi pengkabutan |
| dibuat_sejak | <i>Datetime</i> | - | - | Waktu data sensor dibuat |

b. Perancangan tabel jadwal penyiraman

Tabel 4. 7 Tabel jadwal penyiraman

| Field | Type | Size | Indeks | Deskripsi |
|--------------|----------|----------------------------|-------------|--|
| id | Integer | 10 | Primary key | Id <i>primary key</i> tabel jadwal penyiraman |
| waktu_1 | Time | - | - | Kolom untuk waktu kesatu penyiraman atau pengkabutan |
| waktu_2 | Time | - | - | Kolom untuk waktu kedua penyiraman atau pengkabutan |
| jenis_aksi | Enum | 'penyiraman','pengkabutan' | - | Kolom untuk jenis aksi yaitu penyiraman atau pengkabutan |
| dibuat_sejak | Datetime | - | - | Waktu dibuatnya data jadwal penyiraman |
| diubah_sejak | Datetime | - | - | Waktu diubahnya data jadwal penyiraman |

c. Perancangan tabel riwayat aksi

Tabel 4. 8 tabel riwayat aksi

| Field | Type | Size | Indeks | Deskripsi |
|--------------|----------|----------------------------|-------------|--|
| id | Integer | 10 | Primary key | Id <i>primary key</i> tabel riwayat aksi |
| Jenis_aksi | Enum | 'penyiraman','pengkabutan' | - | Kolom untuk jenis aksi yaitu penyiraman atau pengkabutan |
| status | Enum | 'aktif','nonaktif' | - | Kolom untuk status penyiraman atau pengkabutan |
| dibuat_sejak | Datetime | - | - | Waktu dibuatnya data jadwal riwayat aksi |

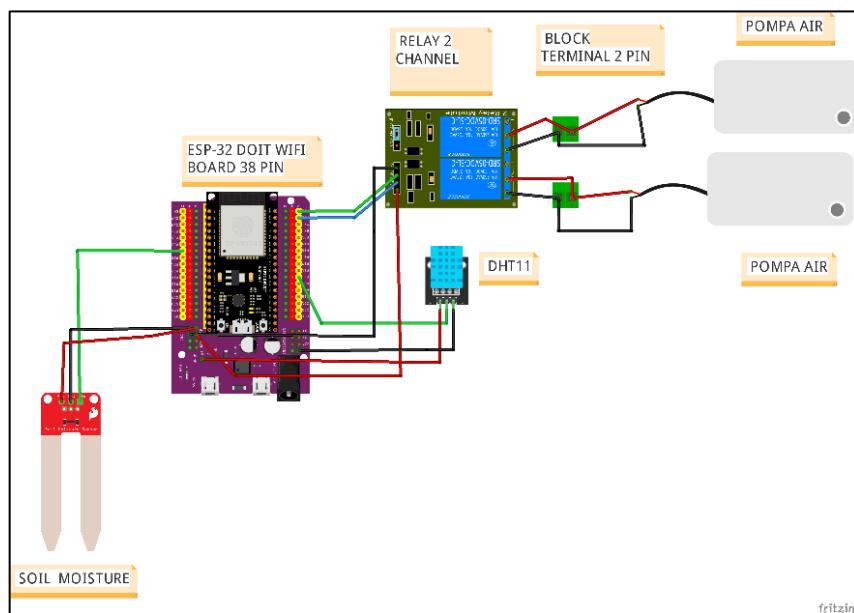
d. Perancangan tabel nomor_hp

Tabel 4. 9 tabel nomor_hp

| Field | Type | Size | Indeks | Deskripsi |
|--------------|----------|------|-------------|--------------------------------------|
| id | Integer | 10 | Primary key | Id <i>primary key</i> tabel nomor_hp |
| Nomor_hp | String | 15 | - | Kolom untuk data nomor Whatsapp |
| dibuat_sejak | Datetime | - | - | Waktu dibuatnya data nomor_hp |
| diubah_sejak | Datetime | - | - | Waktu diubahnya data nomor_hp |

4.2.6 Perancangan *Wiring Diagram*

Sebelum sistem dapat berjalan, diperlukan perancangan rangkaian pada perangkat keras supaya seluruh komponen dapat bekerja dengan terintegrasi sesuai dengan fungsi nya masing-masing. Pada tahap ini, dilakukan penyusunan rangkaian elektronik menggunakan aplikasi *Fritzing* untuk memvisualisasi koneksi antar komponen *IOT*. Rancangan ini berfungsi sebagai acuan dalam proses perakitan fisik sistem *Internet of Things* penyiraman dan pengkabutan.



Gambar 4. 12 *Wiring Diagram*

Gambar di atas menunjukkan *Wiring Diagram* dari sistem *IOT* yang dirancang untuk melakukan penyiraman dan pengkabutan secara otomatis berdasarkan parameter lingkungan seperti kelembapan tanah, suhu, dan kelembapan udara. Perancangan ini dibuat menggunakan aplikasi Fritzing untuk memvisualisasikan koneksi antar komponen secara jelas sebelum dilakukan perakitan fisik. Diagram ini menjadi acuan dalam proses integrasi perangkat keras, sehingga seluruh komponen dapat berfungsi dengan baik sesuai dengan perannya masing-masing dalam sistem. Perancangan wiring ini bertujuan agar seluruh komponen terhubung dan bekerja secara terintegrasi sesuai dengan logika kontrol yang dirancang dalam sistem. Komponen yang digunakan dalam rangkaian ini antara lain:

Tabel 4. 10 Deskripsi *Wiring Diagram*

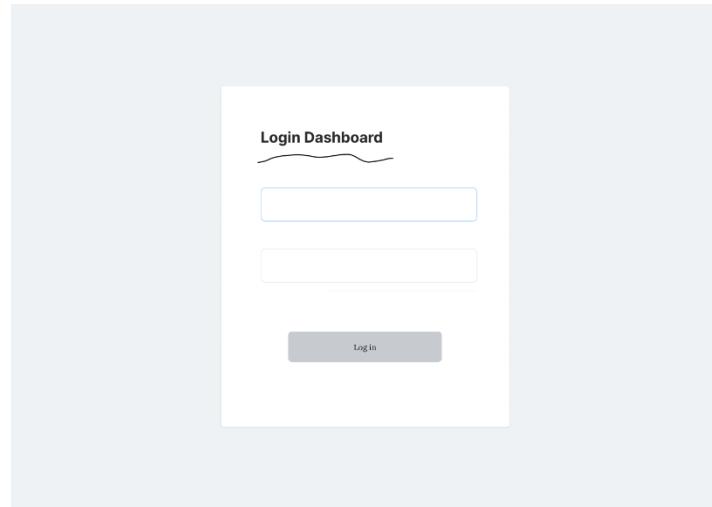
| Komponen | Deskripsi |
|--------------------------------------|---|
| <i>ESP32 DOIT Dev Board (38 pin)</i> | Sebagai pusat kendali sistem, yang menerima data dari sensor dan mengirimkan perintah ke aktuator (<i>Relay</i>) |
| <i>Sensor Soil Moisture</i> | Untuk mengukur tingkat kelembapan tanah. Sensor ini mengirimkan data ke <i>ESP32</i> guna menentukan kebutuhan penyiraman tanaman |
| <i>Sensor DHT11</i> | Berfungsi untuk mengukur suhu dan kelembapan udara. Data dari sensor ini digunakan untuk memutuskan kapan sistem harus mengaktifkan pengkabutan |
| <i>Relay 2 Channel</i> | <i>Relay</i> ini digunakan untuk menghidupkan atau mematikan dua pompa air, masing-masing untuk sistem penyiraman dan sistem pengkabutan |
| Blok terminal 2 pin | Digunakan untuk menghubungkan pompa air ke rangkaian <i>Relay</i> dengan aman |
| Pompa air | Berfungsi sebagai aktuator untuk menyiram tanaman dan melakukan pengkabutan |

4.2.7 Desain Antarmuka

Desain antarmuka (*User Interface*) merupakan proses perancangan tampilan sistem yang bertujuan untuk memberikan kemudahan dan kenyamanan bagi pengguna dalam mengoperasikan aplikasi. Pada sistem penyiraman dan pengkabutan otomatis tanaman berbasis *Internet of Things* ini, antarmuka dirancang dalam bentuk *Dashboard* web yang memungkinkan *admin* untuk memantau kondisi lingkungan tanaman, melihat data historis aktivitas penyiraman dan pengkabutan, serta mengatur notifikasi. Berikut merupakan tampilan antarmuka dari sistem yang diimplementasikan di Avicenna *Greenhouse*:

1. Desain tampilan antarmuka *Login*

Berikut merupakan tampilan antarmuka form *Login* untuk sistem penyiraman dan pengkabutan yang akan dirancang:

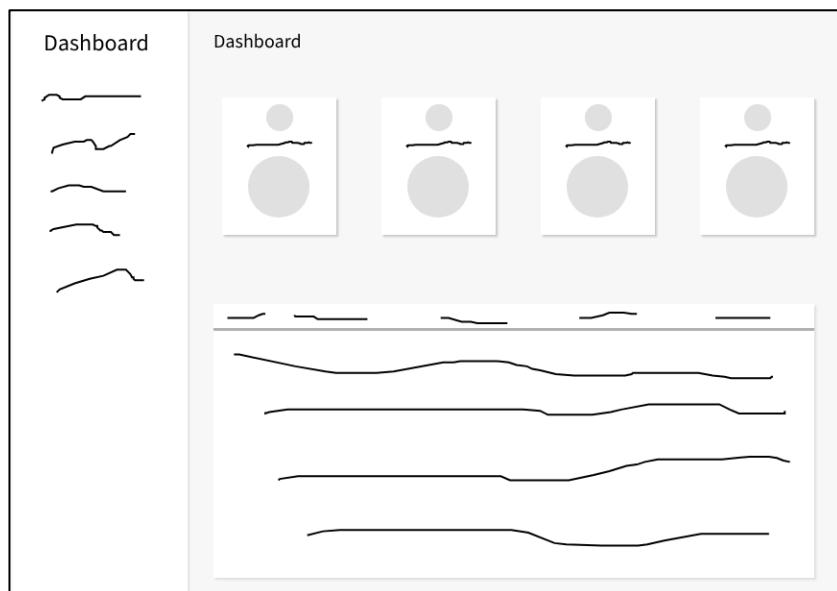


Tabel 4. 11 *User Interface Login*

Gambar di atas menunjukkan antarmuka *Login* yang terdiri dari dua kolom input, yaitu *username/email* dan *password*, serta tombol *Login* yang digunakan untuk masuk ke dalam sistem. Setelah data diisikan dengan benar, pengguna akan diarahkan menuju *Dashboard* web untuk memantau kondisi lingkungan dan mengatur sistem.

2. Desain tampilan antarmuka *Dashboard*

Berikut merupakan tampilan antarmuka *Dashboard* untuk sistem penyiraman dan pengkabutan yang akan dirancang:

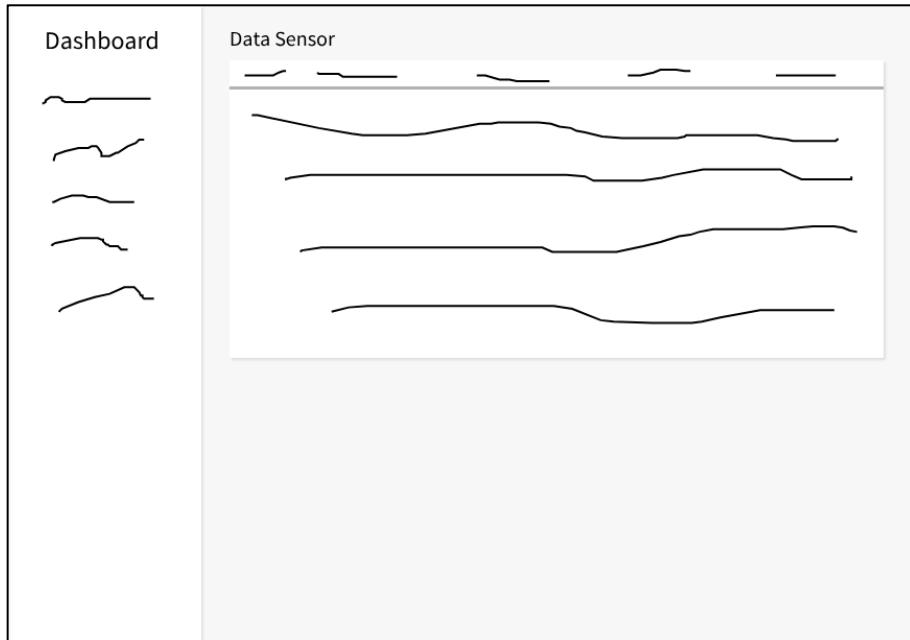


Tabel 4. 12 *User Interface Dashboard*

Antarmuka pengguna pada sistem penyiraman dan pengkabutan otomatis menampilkan *Dashboard* yang dirancang untuk memberikan informasi lingkungan secara menyeluruh. Tampilan ini menyajikan data sensor, seperti kelembapan tanah, suhu, dan kelembapan udara, dalam bentuk visual yang mudah dipahami. Di bagian atas terdapat kartu informasi untuk menampilkan status atau nilai terkini dari masing-masing parameter. Sedangkan di bagian bawah, terdapat tabel atau grafik histori yang menampilkan riwayat aktivitas penyiraman dan pengkabutan.

3. Desain tampilan antarmuka data sensor

Berikut merupakan tampilan antarmuka data sensor untuk sistem penyiraman dan pengkabutan yang akan dirancang:

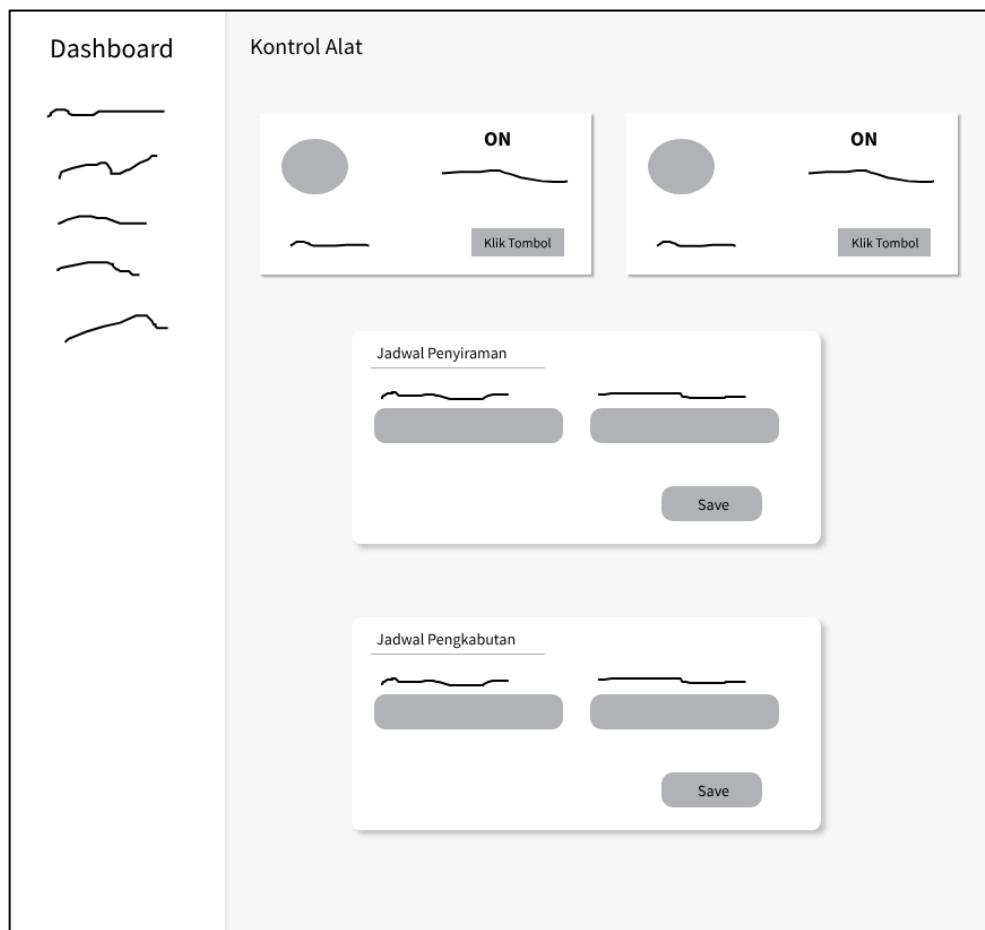


Tabel 4. 13 *User Interface* Data Sensor

Antarmuka data sensor pada sistem penyiraman dan pengkabutan otomatis dirancang untuk menampilkan informasi hasil pembacaan dari berbagai sensor secara *real-time*. Tampilan ini menyajikan data lingkungan seperti kelembapan tanah, suhu udara, kelembapan udara, dan parameter lainnya dalam bentuk tabel atau grafik garis yang mudah dipahami. Melalui antarmuka ini, *admin* dapat memantau perubahan nilai sensor dari waktu ke waktu, yang menjadi dasar pengambilan keputusan oleh sistem berbasis algoritma *Naïve bayes*.

4. Desain tampilan antarmuka kontrol alat

Berikut merupakan tampilan antarmuka kontrol alat untuk sistem penyiraman dan pengkabutan yang akan dirancang:

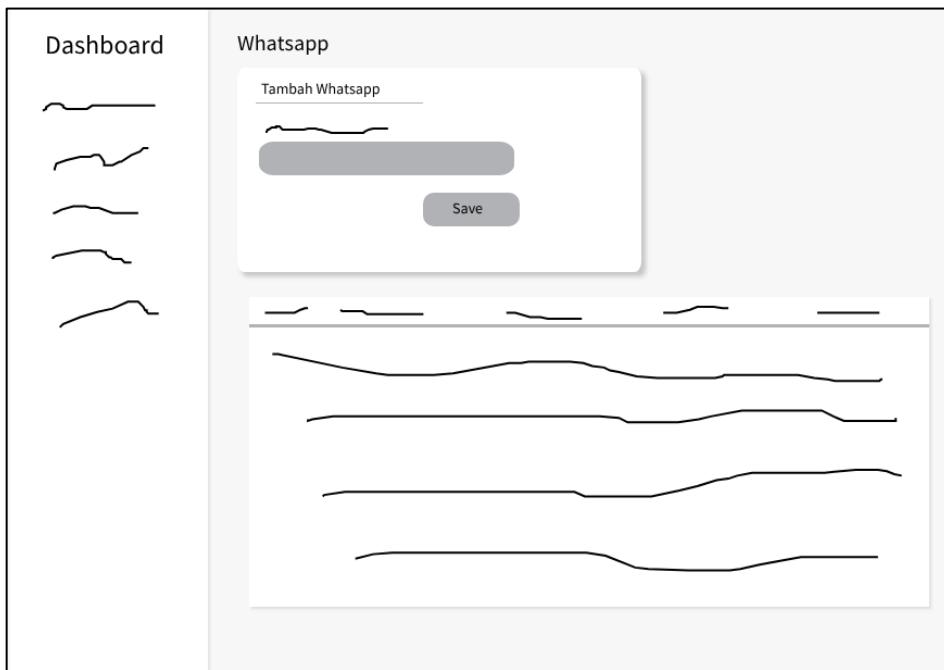


Tabel 4. 14 *User Interface* Kontrol Alat

Antarmuka kontrol alat pada sistem penyiraman dan pengkabutan otomatis dirancang untuk memberikan kemudahan kepada *admin* dalam mengendalikan perangkat secara manual. Tampilan ini menampilkan dua tombol kontrol utama, masing-masing untuk menghidupkan atau mematikan pompa penyiraman dan pengkabutan. Status perangkat ditampilkan secara *real-time* dengan indikator *on/off*. Selain itu, antarmuka ini juga dilengkapi dengan fitur pengaturan batas sensor, yang memungkinkan *admin* mengubah nilai ambang batas suhu, kelembapan udara, atau kelembapan tanah sesuai kebutuhan. Perubahan dapat disimpan dengan menekan tombol *save*.

5. Desain tampilan antarmuka *Whatsapp*

Berikut merupakan tampilan antarmuka *Whatsapp* untuk sistem penyiraman dan pengkabutan yang akan dirancang:



Tabel 4. 15 *User Interface Whatsapp*

Gambar di atas menunjukkan desain wireframe antarmuka fitur *Whatsapp* dalam sistem *IOT* penyiraman dan pengkabutan otomatis. Fitur ini memungkinkan *admin* untuk menambahkan dan mengelola nomor *Whatsapp* yang akan digunakan sebagai tujuan notifikasi otomatis dari sistem. Pada bagian atas tampilan, terdapat form input dengan label "Tambah Whatsapp", di mana *admin* dapat memasukkan nomor telepon yang ingin ditambahkan. Di bawahnya terdapat tombol "Save" untuk menyimpan nomor tersebut ke dalam sistem. Bagian bawah wireframe menampilkan tabel atau daftar nomor *Whatsapp* yang telah didaftarkan. Setiap entri kemungkinan menampilkan informasi seperti nomor yang terdaftar, status pengiriman pesan, serta waktu pengiriman notifikasi terakhir. Desain antarmuka ini dirancang sederhana dan fungsional agar memudahkan pengguna dalam mengelola daftar kontak yang menerima notifikasi, seperti peringatan batas sensor terlampaui atau status perangkat.

BAB V

IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi

Setelah melaksanakan analisis dan perancangan maka selanjutnya adalah pengimplementasian untuk menjalankan analisis dan perancangan yang sudah dibuat ke dalam bentuk aplikasi.

5.1.1 Listing Program

1. Listing program Kontrol alat

```
from app.src.repositories.riwayat_aksi_repositories import
create_riwayat_aksi_repository,get_all_riwayat_aksi_repository
from app.src.services.mqtt_service import kirim_perintah_siram,
kirim_perintah_kabut
# Akses latest_sensor_data dari mqtt_service
from app.src.services.mqtt_service import latest_sensor_data
from app.src.services.notification_service import notify_sensor_data_Service
import time
from datetime import datetime

from flask import current_app as app
def kontrol_penyiraman_service(perintah):
    kirim_perintah_siram(perintah)
    created_notification_service(perintah,"penyiraman")

def kontrol_pengkabutan_service(perintah):
    kirim_perintah_kabut(perintah)
    created_notification_service(perintah,"pengkabutan")

def created_notification_service(perintah,action=None):
```

```

data = {
    "jenis_aksi": action,
    "status": "aktif" if perintah == "1" else "nonaktif"
}
status = "Aktif" if perintah == "1" else "Mati"
waktu = datetime.now().strftime('Tanggal %d-%m-%Y, jam :%H:%M')
pesan = f"⚠️ Pengkabutan: {status}. {waktu}"
notify_sensor_data_Service(pesan, app)
create_riwayat_aksi_repository(data)
notify_sensor_data_Service(pesan, app)

def get_jadwal_penyiraman_service():
    # Implement the logic to retrieve the watering schedule from the database
    get_all_riwayat_aksi = get_all_riwayat_aksi_repository()
    return get_all_riwayat_aksi

def auto_control_loop():
    print("🚀 Auto Control Service started")
    while True:
        kelembapan_tanah = latest_sensor_data.get("Kelembapan Tanah")
        suhu_udara = latest_sensor_data.get("Suhu Udara")
        kelembapan_udara = latest_sensor_data.get("Kelembapan Udara")

        if kelembapan_tanah is not None and kelembapan_tanah > 80:
            kirim_perintah_siram("0")
        if suhu_udara is not None and kelembapan_udara is not None:
            print(f"Suhu Udara: {suhu_udara}, Kelembapan Udara: {kelembapan_udara}")
            if suhu_udara < 27 and kelembapan_udara > 71:
                kirim_perintah_kabut("0") # Matikan kabut
                time.sleep(2)

```

2. Listing program *Naïve bayes*

```

import os
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.multioutput import MultiOutputClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
model = None
required_features = ['Suhu Udara', 'Kelembapan Udara', 'Kelembapan Tanah']
target_columns = ['Penyiraman', 'Pengkabutan']

def load_data_from_csv(filename='sensor_data.csv'):
    file_path = os.path.join(os.path.dirname(os.path.abspath(__file__)), filename)
    if not os.path.isfile(file_path):
        print(f"❌ File tidak ditemukan: {file_path}")
        return None
    try:
        df = pd.read_csv(file_path)
        # Pastikan kolom target numerik
        df['Penyiraman'] = df['Penyiraman'].map({'Perlu': 1, 'Tidak perlu': 0})
        df['Pengkabutan'] = df['Pengkabutan'].map({'Perlu': 1, 'Tidak perlu': 0})
        if df[target_columns].isnull().any().any():
            print("❌ Data target mengandung nilai tak valid setelah konversi.")
            return None
        return df
    except Exception as e:
        print(f"❌ Gagal membaca file CSV: {e}")
        return None

def train_naive_bayes(csv_path='sensor_data.csv'):
    global model
    df = load_data_from_csv(csv_path)
    if df is None or len(df) < 5:

```

```

print(" ❌ Data terlalu sedikit atau tidak valid.")

return

X = df[required_features]
y = df[target_columns]
model = MultiOutputClassifier(GaussianNB())
model.fit(X, y)
y_pred = model.predict(X)
acc_penyiraman = accuracy_score(y['Penyiraman'], y_pred[:, 0])
acc_pengkabutan = accuracy_score(y['Pengkabutan'], y_pred[:, 1])
print(f" ⚡ Akurasi Penyiraman: {acc_penyiraman:.2f}")
print(f" ⚡ Akurasi Pengkabutan: {acc_pengkabutan:.2f}")

def predict_status(sensor_data):
    global model
    if model is None:
        print(" ! Model belum dilatih.")
        return None
    try:
        data_input = pd.DataFrame([
            'Suhu Udara': float(sensor_data['Suhu Udara']),
            'Kelembapan Udara': float(sensor_data['Kelembapan Udara']),
            'Kelembapan Tanah': float(sensor_data['Kelembapan Tanah'])
        ])
        prediction = model.predict(data_input)[0]
        # logic_result = apply_manual_logic(data_input.iloc[0])
        result = {
            'Penyiraman': 'Perlu' if prediction[0] else 'Tidak perlu',
            # 'Penyiraman (Logika)': 'Perlu' if logic_result[0] else 'Tidak perlu',
            'Pengkabutan': 'Perlu' if prediction[1] else 'Tidak perlu',
            # 'Pengkabutan (Logika)': 'Perlu' if logic_result[1] else 'Tidak perlu'
        }
        return result
    
```

```

except Exception as e:
    print(f'❌ Gagal memproses prediksi: {e}')
    return None

```

3. Listing program Penjadwalan

```

from app.src.repositories.jadwal_penyiraman_repositories import
    update_jadwal_by_id_repository, create_jadwal_penyiraman_repository
from apscheduler.schedulers.background import BackgroundScheduler
from apscheduler.triggers.cron import CronTrigger
import pytz
from app.src.services.control_service import
    kontrol_penyiraman_service, kontrol_pengkabutan_service
from app.src.repositories.jadwal_penyiraman_repositories import
    get_jadwal_penyiraman_by_jenis_repository
from app.src.services.notification_service import notify_sensor_data_Service
import datetime
scheduler = BackgroundScheduler(timezone=pytz.timezone('Asia/Jakarta'))
# penjadwalan_service.py
scheduler = BackgroundScheduler(timezone=pytz.timezone('Asia/Jakarta'))
app = None # global app instance
def init_app(app_instance):
    global app
    app = app_instance
def update_jadwal_service(jadwal_id, waktu_1, waktu_2, jenis):
    try:
        if jadwal_id is None:
            create_jadwal = {
                "waktu_1": waktu_1,
                "waktu_2": waktu_2,
                "jenis_aksi": jenis
            }
    
```

```

jadwal = create_jadwal_penyiraman_repository(create_jadwal)
return {"success": True, "jadwal": {
    "id": jadwal.id,
    "waktu_1": jadwal.waktu_1.strftime("%H:%M:%S"),
    "waktu_2": jadwal.waktu_2.strftime("%H:%M:%S"),
    "jenis_aksi": jadwal.jenis_aksi
}}
jadwal = update_jadwal_by_id_repository(jadwal_id, waktu_1, waktu_2,
jenis)
return {"success": True, "jadwal": {
    "id": jadwal.id,
    "waktu_1": jadwal.waktu_1.strftime("%H:%M:%S"),
    "waktu_2": jadwal.waktu_2.strftime("%H:%M:%S"),
    "jenis_aksi": jadwal.jenis_aksi
}}
except Exception as e:
    print("Error update_jadwal:", e)
    return {"success": False}
def kontrol_penyiraman_service_wrapper(perintah):
    with app.app_context():
        kontrol_penyiraman_service(perintah)
def kontrol_pengkabutan_service_wrapper(perintah):
    with app.app_context():
        kontrol_pengkabutan_service(perintah)
def jadwal_penyiraman_service():
    penyiraman =
get_jadwal_penyiraman_by_jenis_repository(jenis_aksi='penyiraman')
    if not penyiraman:
        print("⚠️ Jadwal penyiraman tidak ditemukan di database.")
        return
    times = [penyiraman.waktu_1, penyiraman.waktu_2]
    for idx, time_obj in enumerate(times):

```

```

try:
    hour = time_obj.hour
    minute = time_obj.minute
    job_id = f"penyiraman_{idx+1}_{hour:02d}{minute:02d}"
    if not scheduler.get_job(job_id):
        scheduler.add_job(
            func=kontrol_penyiraman_service_wrapper,
            trigger=CronTrigger(hour=hour, minute=minute, second=0),
            id=job_id,
            replace_existing=True,
            kwargs={"perintah": "1"} # <-- Inject app
        )
except Exception as e:
    print(f"🔴 Gagal menambahkan job untuk waktu '{time_obj}': {e}")
if not scheduler.running:
    scheduler.start()
    print("🕒 Penjadwalan tugas penyiraman dimulai.")
else:
    print("🕒 Scheduler sudah berjalan.")

def jadwal_pengkabutan_service():
    pengkabutan =
    get_jadwal_penyiraman_by_jenis_repository(jenis_aksi='pengkabutan')
    if not pengkabutan:
        print("⚠️ Jadwal pengkabutan tidak ditemukan di database.")
        return
    times = [pengkabutan.waktu_1, pengkabutan.waktu_2]
    for idx, time_obj in enumerate(times):
        try:
            hour = time_obj.hour
            minute = time_obj.minute
            job_id = f"pengkabutan_{idx+1}_{hour:02d}{minute:02d}"

```

```

if not scheduler.get_job(job_id):
    scheduler.add_job(
        func=kontrol_pengkabutan_service_wrapper,
        trigger=CronTrigger(hour=hour, minute=minute, second=0),
        id=job_id,
        replace_existing=True,
        kwargs={"perintah": "1"} # <-- Inject app
    )
except Exception as e:
    print(f"🔴 Gagal menambahkan job untuk waktu '{time_obj}': {e}")
if not scheduler.running:
    scheduler.start()
    print("🕒 Penjadwalan tugas pengkabutan dimulai.")
else:
    print("🕒 Scheduler sudah berjalan.")

def refresh_jadwal_service(jenis_aksi: str):
    """ Hapus semua job untuk jenis aksi lalu jadwalkan ulang berdasarkan data terbaru."""
    prefix = f"{jenis_aksi}_"
    job_ids = [job.id for job in scheduler.get_jobs() if job.id.startswith(prefix)]
    for job_id in job_ids:
        scheduler.remove_job(job_id)
    if jenis_aksi == "penyiraman":
        jadwal_penyiraman_service()
    elif jenis_aksi == "pengkabutan":
        jadwal_pengkabutan_service()

def jadwal_service_utama(app_instance):
    init_app(app_instance) # Simpan app global
    jadwal_penyiraman_service()
    jadwal_pengkabutan_service()

```

4. Listing program Notifikasi

```

import os
import requests
from app.src.repositories.nohp_repositories import get_all_nomor_hp
from dotenv import load_dotenv
def notify_sensor_data_Service(msg, app=None):
    load_dotenv(override=True)
    wa_server_url = os.getenv('WA_SERVER_URL')
    session_id = os.getenv('WA_SESSION_ID')
    if not wa_server_url:
        print("🔴 WA_SERVER_URL tidak ditemukan di .env")
        return
    if not session_id:
        print("🔴 WA_SESSION_ID tidak ditemukan di .env")
        return
    with app.app_context():
        try:
            numbers = get_all_nomor_hp()
            if not numbers:
                print("⚠️ Tidak ada nomor WA yang ditemukan di database.")
                return
            for record in numbers:
                phone_number = record.nomor_hp if hasattr(record, 'nomor_hp') else
record['nomor_hp']
                payload = {
                    "number": phone_number,
                    "message": msg,
                    "sessionId": session_id
                }
                headers = {
                    "Content-Type": "application/json"
                }
                response = requests.post(wa_server_url, json=payload, headers=headers)
                if response.status_code != 200:
                    print(f"Error: {response.status_code} - {response.text}")
        except Exception as e:
            print(f"An error occurred: {e}")

```

```

    }

try:

    response = requests.post(wa_server_url, json=payload,
headers=headers)

    print(f" 📡 Mengirim pesan ke {phone_number}...")

    print(f'Payload: {payload}')

    response.raise_for_status()

    res_json = response.json()

    if res_json.get("status") == "success":

        print(f" ✅ Pesan berhasil dikirim ke {phone_number}")

    else:

        print(f" ❌ Gagal kirim ke {phone_number}: {res_json}")

except requests.exceptions.RequestException as err:

    print(f" ❌ Error koneksi ke {phone_number}: {err}")

except Exception as e:

    print(f" ❌ Error saat mengambil nomor dari database: {e}")

```

5. Listing program *Mqtt*

```

import paho.Mqtt.client as paho

from paho import Mqtt

import os

import time

import requests

from datetime import datetime, timedelta

import ssl

import certifi

from dotenv import load_dotenv

from app import socketio # Untuk emit ke client

from app.src.services.notification_service import notify_sensor_data_Service

from Flask import current_app

```

```

# Memuat variabel lingkungan dari file .env
load_dotenv()

# 🔑 Konfigurasi MQTT & Flask API
BROKER = os.environ.get('MQTT_BROKER', "")
PORT = 8883
USERNAME = os.environ.get('MQTT_USERNAME')
PASSWORD = os.environ.get('MQTT_PASSWORD')
FLASK_URL = os.environ.get('FLASK_URL')

# Mqtt_service.py
app_context = None
TOPICS = [
    ("otomatis/siram/status", 1),
    ("otomatis/kabut/status", 1),
    ("sensor/kelembapan_tanah", 1),
    ("sensor/suhu_udara", 1),
    ("sensor/kelembapan_udara", 1)
]
# 💡 Data sensor & waktu terakhir update
latest_sensor_data = {
    "Suhu Udara": None,
    "Kelembapan Udara": None,
    "Kelembapan Tanah": None
}
sensor_last_seen = {
    "Suhu Udara": None,
    "Kelembapan Udara": None,
    "Kelembapan Tanah": None
}
sensor_last_value = {
    "Suhu Udara": None,
    "Kelembapan Udara": None,
    "Kelembapan Tanah": None
}

```

```

}

sensor_last_change = {
    "Suhu Udara": None,
    "Kelembapan Udara": None,
    "Kelembapan Tanah": None
}

sensor_alert_sent = {
    "Suhu Udara": False,
    "Kelembapan Udara": False,
    "Kelembapan Tanah": False
}

check_times = [1, 5, 15, 24] # jam
SENSOR_RESET_HOURS = 24 # reset jika 24 jam tidak ada perubahan nilai
SENSOR_STALE_SECONDS = 60 # peringatan jika 1 menit tidak berubah
client = None

# ● Callback koneksi MQTT
def on_connect(client, userdata, flags, rc, properties=None):
    if rc == 0:
        print("✓ Terhubung ke MQTT Broker")
        for topic, qos in TOPICS:
            client.subscribe(topic, qos)
            print(f"📡 Berlangganan ke: {topic}")
    else:
        print(f"✗ Koneksi MQTT gagal, kode: {rc}")

# 📈 Callback untuk data sensor

def handle_sensor_data(client, userdata, msg):
    topic = msg.topic
    value = msg.payload.decode('utf-8').strip()
    mapping = {

```

```

    "sensor/kelembapan_tanah": "Kelembapan Tanah",
    "sensor/suhu_udara": "Suhu Udara",
    "sensor/kelembapan_udara": "Kelembapan Udara",
}

label = mapping.get(topic)
if label:
    try:
        value_float = float(value)
    except ValueError:
        print(f" ❌ Nilai sensor tidak valid: {value}")
        return
    now = datetime.now()
    last_val = sensor_last_value[label]
    # Deteksi perubahan nilai
    if last_val != value_float:
        sensor_last_change[label] = now
        sensor_last_value[label] = value_float
        latest_sensor_data[label] = value_float
        sensor_last_seen[label] = now
        socketio.emit("sensor_update", latest_sensor_data)

# ⚠ Cek timeout data dan perubahan nilai
def check_sensor_status():
    now = datetime.now()
    updated = False
    for label in latest_sensor_data:
        last_seen = sensor_last_seen.get(label)
        last_change = sensor_last_change.get(label)
        val = latest_sensor_data[label]
        # Reset jika data sudah usang
        if last_change and (now - last_change).total_seconds() >
            SENSOR_RESET_HOURS * 3600:

```

```

if val is not None:
    latest_sensor_data[label] = None
    updated = True
    print(f"⚠️ Data {label} tidak berubah selama
{SENSOR_RESET_HOURS} jam Di-reset.")
    # Peringatan jika tidak berubah dalam 1 jam
    if last_change and (now - last_change).total_seconds() >
SENSOR_STALE_SECONDS:
        notify_sensor_data_Service(
            f"⚠️ Peringatan: {label} tidak berubah selama lebih dari 1 jam.\n"
            f" Cek sinyal atau perangkat.\n"
            f" Cek Dashboard: {os.environ.get('FLASK_URL')}", app_context
        )
        print(f"⚠️ WARNING: {label} tidak berubah selama lebih dari 1 jam")
if updated:
    socketio.emit("sensor_update", latest_sensor_data)
# Inisialisasi waktu pengecekan berikutnya
def init_next_check_times():
    now = datetime.now()
    next_times = {}
    for jam in check_times:
        scheduled_time = now.replace(hour=jam % 24, minute=0, second=0,
microsecond=0)
        if scheduled_time <= now:
            scheduled_time += timedelta(days=1) # Jadwalkan untuk hari
berikutnya
            next_times[jam] = scheduled_time
    return next_times
# Inisialisasi waktu pengecekan berikutnya
next_check_time = init_next_check_times()

```

```
# 🚀 Jalankan MQTT

def run_Mqtt_service(app_instance):
    global app_context
    app_context = app_instance
    global client
    print(f'🔌 Menghubungkan ke MQTT {BROKER}:{PORT}')
    client = paho.Client(client_id="otomatisasi_penyiram",
    protocol=paho.MQTTv5)
    client.username_pw_set(USERNAME, PASSWORD)
    client.tls_set(ca_certs=certifi.where(),
    tls_version=ssl.PROTOCOL_TLS_CLIENT)
    client.on_connect = on_connect
    client.message_callback_add("sensor/kelembapan_tanah",
    handle_sensor_data)
    client.message_callback_add("sensor/suhu_udara", handle_sensor_data)
    client.message_callback_add("sensor/kelembapan_udara",
    handle_sensor_data)
    try:
        client.connect(BROKER, PORT)
        client.loop_start()
    except Exception as e:
        print(f'🔴 Gagal koneksi: {e}')
        socketio.emit("Mqtt_error", {"error": str(e)})
        return
    try:
        while True:
            now = datetime.now()
            for jam in check_times:
                if now >= next_check_time[jam]:
                    print(f'⌚ Menjalankan pengecekan sensor untuk jam ke-{jam}')
                    check_sensor_status()
```

```

# Jadwalkan ulang jam ke-jam berikutnya
next_check_time[jam] += timedelta(days=1)

time.sleep(60) # cek setiap 1 menit apakah waktunya eksekusi

except KeyboardInterrupt:

    print("🔴 Menutup koneksi MQTT...")
    client.disconnect()
    client.loop_stop()

# 🚧 Perintah manual

def kirim_perintah_siram(perintah):
    if client:
        client.publish("otomatis/siram/status", perintah)
        print(f"📤 Kirim perintah: {perintah}")
        socketio.emit("status_siram", {"status": perintah})

def kirim_perintah_kabut(perintah):
    if client:
        client.publish("otomatis/kabut/status", perintah)
        socketio.emit("status_kabut", {"status": perintah})
        print(f"📤 Kirim perintah: {perintah}")

```

6. Listing program *Arduino ESP32*

```

#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <DHT.h>

// Konfigurasi WiFi
const char* ssid = "adm";
const char* password = "adams161121..";
// Konfigurasi MQTT

```

```

const char* Mqtt_server =
"5ef609a0286140bb9cb1dd89dabdc139.s1.eu.hivemq.cloud";
const int Mqtt_port = 8883; // Gunakan 1883 untuk koneksi tanpa SSL
const char* Mqtt_user = "hivemq.webclient.1746975391787";
const char* Mqtt_password = "8HZK!l@%q3?Ah7o0TrxU";
// Topik MQTT
const char* topic_siram_status = "otomatis/siram/status";
const char* topic_kabut_status = "otomatis/kabut/status";
const char* topic_kelembapan_tanah = "sensor/kelembapan_tanah";
const char* topic_suhu_udara = "sensor/suhu_udara";
const char* topic_kelembapan_udara = "sensor/kelembapan_udara";
const char* topic_ketinggian_air = "sensor/ketinggian_air";
// Konfigurasi pin
#define RELAY1_PIN 22
#define RELAY2_PIN 23
#define DHTPIN 4
#define DHTTYPE DHT11
#define SOIL_MOISTURE_PIN 33 // A0
const int DRY_VALUE = 4000;
const int WET_VALUE = 1200;
WiFiClientSecure espClient;
PubSubClient client(espClient);
DHT dht(DHTPIN, DHTTYPE);

// === Callback MQTT ===
void callback(char* topic, byte* payload, unsigned int length) {
    String message;
    for (unsigned int i = 0; i < length; i++) {
        message += (char)payload[i];
    }
    if (String(topic) == topic_siram_status) {
        digitalWrite(RELAY1_PIN, message == "1" ? LOW : HIGH);
    }
}

```

```
    } else if (String(topic) == topic_kabut_status) {
        digitalWrite(RELAY2_PIN, message == "1" ? LOW : HIGH);
    }
}

// === Koneksi ke WiFi ===

void setup_WiFi() {
    delay(10);
    Serial.println();
    Serial.print("Menghubungkan ke ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi terhubung");
    Serial.println("Alamat IP: ");
    Serial.println(WiFi.localIP());
}

// === Setup Awal ===

void setup() {
    Serial.begin(115200);
    dht.begin();
    pinMode(RELAY1_PIN, OUTPUT);
    pinMode(RELAY2_PIN, OUTPUT);
    digitalWrite(RELAY1_PIN, HIGH); // Relay non-aktif (aktif LOW)
    digitalWrite(RELAY2_PIN, HIGH);
    setup_WiFi();
    espClient.setInsecure(); // Nonaktifkan verifikasi sertifikat SSL (hanya untuk
                           pengujian)
```

```

client.setServer(Mqtt_server, Mqtt_port);
client.setCallback(callback);
}

// === WIFI ===

void reconnect() {
    while (!client.connected()) {
        Serial.print("Menghubungkan ke MQTT...");
        if (client.connect("ESP32Client", Mqtt_user, Mqtt_password)) {
            Serial.println("terhubung");
            client.subscribe(topic_siram_status);
            client.subscribe(topic_kabut_status);
        } else {
            Serial.print("gagal, rc=");
            Serial.print(client.state());
            Serial.println(" coba lagi dalam 5 detik");
            delay(5000);
        }
    }
}

// === Loop ===

void loop() {
    if (!client.connected()) {
        reconnect();
    }
    client.loop();
    // Baca sensor
    float suhu = dht.readTemperature();
    float kelembapan = dht.readHumidity();
    int nilaiSensorTanah = analogRead(SOIL_MOISTURE_PIN);
    // Konversi nilai kelembapan tanah ke persentase
    int kelembapanTanah = map(nilaiSensorTanah, DRY_VALUE,
WET_VALUE, 0, 100);
}

```

```

kelembapanTanah = constrain(kelembapanTanah, 0, 100);
// Kirim data ke MQTT
client.publish(topic_suhu_udara, String(suhu).c_str(), true);
client.publish(topic_kelembapan_udara, String(kelembapan).c_str(), true);
client.publish(topic_kelembapan_tanah, String(kelembapanTanah).c_str(),
true);
delay(3000); // Kirim data setiap 3 detik
}

```

5.1.2 Implementasi Sistem

Implementasi sistem adalah tahap penerapan sistem yang akan dilakukan jika sistem telah disetujui termasuk program yang telah dibuat pada tahap perancangan sistem agar siap untuk dioperasikan. Adapun waktu dan tempat penerapan sistem yang sudah dibuat sebagai berikut:

1. Waktu dan tempat implementasi

Tempat : Avicenna *Greenhouse*

Alamat : Kp. Padarek Rt. 03 Rw. 02 Desa Drawati Kec. Paseh Kab. Bandung, Prov. Jawa Barat

Waktu : 5 Juli 2025

5.1.3 Spesifikasi Sistem

Spesifikasi sistem menjelaskan secara rinci kebutuhan perangkat keras dan perangkat lunak yang digunakan dalam pengembangan serta implementasi sistem penyiraman dan pengkabutan otomatis berbasis *IOT*. Informasi ini penting untuk memastikan sistem dapat berjalan secara optimal dalam memantau kondisi lingkungan dan mengaktifkan penyiraman atau pengkabutan sesuai parameter yang telah ditentukan, seperti kelembapan tanah, suhu udara, dan kelembapan udara. Spesifikasi sistem dibagi menjadi dua bagian utama, yaitu:

1. Spesifikasi Perangkat Keras

Berikut adalah perangkat keras yang digunakan untuk mengembangkan sistem:

| | |
|------------------|--|
| Processor | Core 2 |
| RAM | 2 GB |
| Disk Penyimpanan | 50GB |
| Mikrokontroller | <i>ESP32</i> |
| Sensor | <i>DHT11, Soil Moisture</i> |
| Modul | <i>Relay 2 Channel, Pompa air, ESP32</i> |

2. Spesifikasi Perangkat Lunak

Berikut adalah perangkat lunak yang digunakan dalam pengembangan sistem ini adalah sebagai berikut:

| | |
|--------------------|-----------------------------|
| Sistem Operasi | Linux Ubuntu 20 / Window 10 |
| Database | <i>MySQL</i> |
| Bahasa Pemrograman | <i>Python, Arduino</i> |
| Hosting | <i>VPS</i> |

5.1.4 Instalasi Sistem

Instalasi sistem ini memuat penjelasan mengenai langkah-langkah yang harus dilakukan untuk instalasi aplikasi dan *database* di *VPS*, serta konfigurasi perangkat *IOT* menggunakan mikrokontroler *ESP32*.

1. Instalasi Aplikasi

Instalasi aplikasi dilakukan di dua bagian, yaitu di *server VPS* dan di perangkat *ESP32*. Aplikasi *server* dibangun menggunakan *Python (Flask)*, sementara *ESP32* diprogram melalui *Arduino IDE*.

a. Instalasi Aplikasi

1. Mengakses *VPS* melalui *SSH*
2. Mengupdate sistem (*apt update && apt upgrade*)
3. Menginstal *Python* dan *pip*

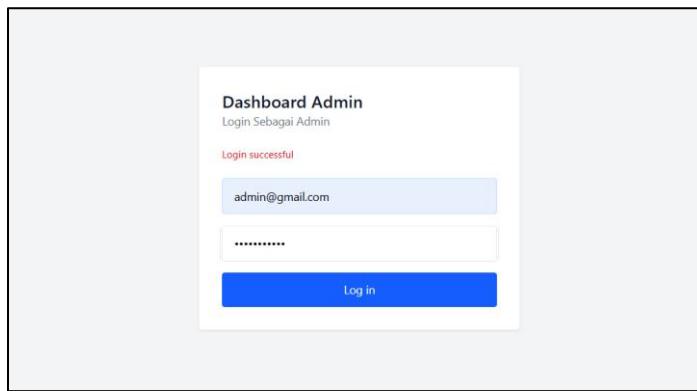
4. Menginstal *virtual environment* (*python3 -m venv venv*)
 5. Mengaktifkan *virtual environment*
 6. Menginstal *library Python* yang dibutuhkan (*pip install -r requirements.txt*)
 7. Mengunggah file aplikasi ke *VPS* (via *Git/SCP*)
 8. Masuk ke direktori aplikasi
 9. Mengatur *file konfigurasi* (*env/database/mqtt*)
 10. Memastikan semua konfigurasi aman.
 11. Menjalankan *Flask*
- b. Instalasi *Database*
1. Masuk ke *server VPS* menggunakan *SSH* dari komputer lokal.
 2. Melakukan instalasi perangkat lunak *MySQL* sebagai sistem manajemen basis data (*DBMS*) yang akan digunakan.
 3. Mengakses konsol *MySQL* untuk melakukan konfigurasi awal.
 4. Menyiapkan basis data yang akan digunakan untuk menyimpan data sensor, log aktivitas, dan konfigurasi system.
 5. Menambahkan pengguna baru dengan *username* dan *password* khusus untuk aplikasi *IOT*.
 6. Memberikan hak penuh (*privileges*) kepada user untuk mengelola *database* yang telah dibuat.
 7. Menguji koneksi dari aplikasi ke *database* untuk memastikan konfigurasi berhasil.
- c. Instalasi Alat *IOT*
1. Mengunggah program ke *ESP32* melalui *Arduino IDE*
 2. Menyiapkan hotspot/*WiFi* untuk *ESP32*
 3. Menyiapkan *broker MQTT*.
 4. Menguji komunikasi *WIFI* (*subscribe/publish*) dengan *server*
 5. Menempatkan alat dalam *box casing*
 6. Menyambungkan adaptor *ESP32* dan pompa ke terminal listrik
 7. Melakukan uji coba manual (perintah siram/kabut dari *server*)
 8. Menata tata letak posisi alat dan sensor sesuai pada denah *Greenhouse*.
 9. Menyambungkan selang, pompa air dan terminal listrik dengan mikrokontroller

5.1.5 Menjalankan Sistem

Pada bagian ini menjelaskan bagaimana langkah – langkah menjalankan sistem aplikasi:

1. Halaman *Login*

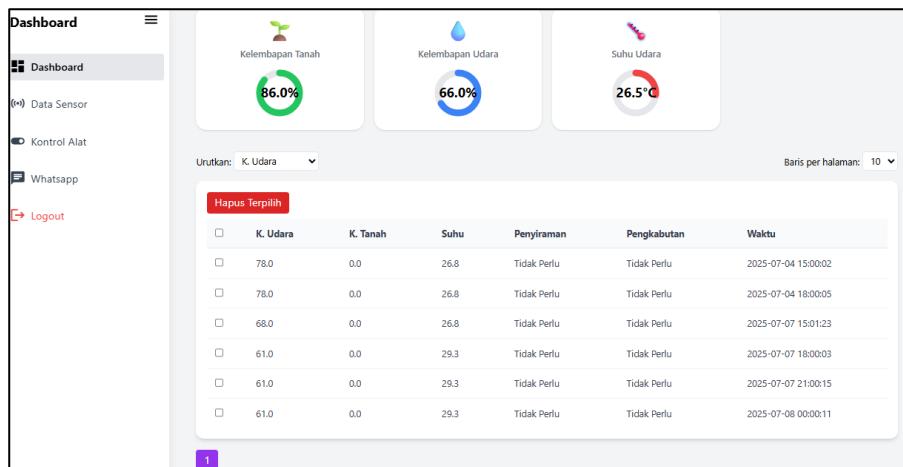
Sebelum mengakses *Dashboard IOT* penyiraman dan pengkabutan, *admin* diharuskan untuk *Login* menggunakan *email* dan *password* yang telah ditetapkan.



Gambar 5. 1 Halaman *Login*

2. Halaman *Dashboard*

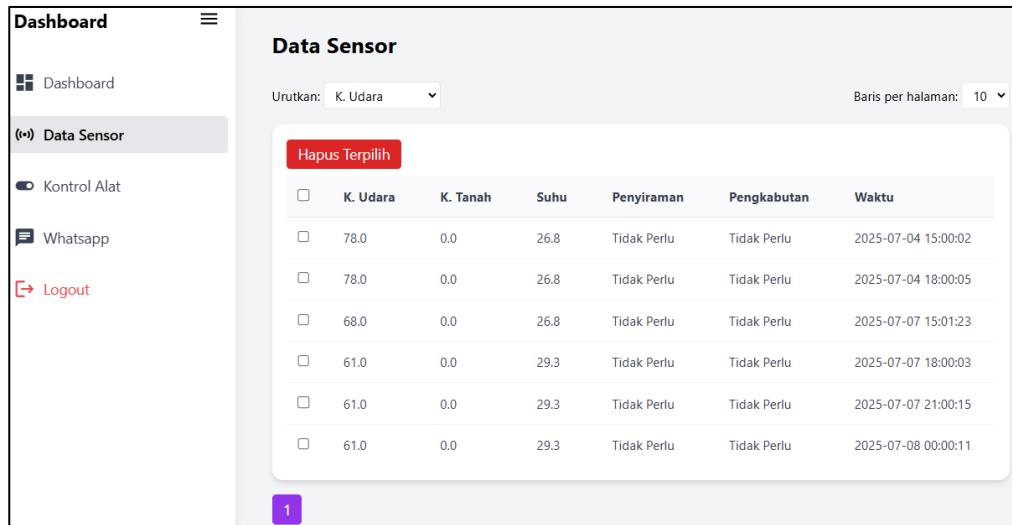
Setelah melakukan *Login*, sistem akan menampilkan halaman awal pada *Dashboard* yang menampilkan data sensor pada *card* komponen dan juga tabel data sensor yang telah dilakukan prediksi *naïve bayes*. Pada tabel dapat melakukan filter data sesuai sensor dan juga dapat menghapus data.



Gambar 5. 2 Halaman *Dashboard*

3. Halaman Data Sensor

Halaman ini menampilkan data hasil pembacaan sensor yang dikirim secara berkala lalu disimpan ke dalam basis data. Sistem dijadwalkan untuk merekam data setiap tiga jam, yaitu pada pukul 00:00, 03:00, 06:00, 09:00, 12:00, 15:00, 18:00, dan 21:00. Setiap entri data mencakup informasi seperti kelembaban udara, kelembaban tanah, suhu, penyiraman dan pengkabutan serta waktu pengambilan data. Setelah data diterima, sistem secara otomatis menjalankan algoritma *Naive Bayes* untuk melakukan analisis dan klasifikasi terhadap kondisi lingkungan. Hasil dari analisis ini akan menentukan apakah penyiraman dan pengkabutan tanaman diperlukan, dengan hasil keputusan ditampilkan langsung dalam kolom penyiraman dan pengkabutan pada tabel. Sistem menyediakan fitur pengurutan berdasarkan nilai dari masing-masing sensor. Selain itu, terdapat pilihan untuk menentukan jumlah data yang ditampilkan pada tabel. Fitur lainnya adalah kemampuan untuk menghapus data yang tidak lagi diperlukan.



The screenshot shows a web-based application interface. On the left is a sidebar with navigation links: 'Dashboard', 'Data Sensor' (which is highlighted in grey), 'Kontrol Alat', 'Whatsapp', and 'Logout'. The main content area has a title 'Data Sensor'. It includes a dropdown menu 'Urutkan:' set to 'K. Udara' and a dropdown for 'Baris per halaman:' set to '10'. Below this is a table titled 'Hapus Terpilih' (Delete Selected) with a red button. The table lists sensor data with columns: K. Udara, K. Tanah, Suhu, Penyiraman, Pengkabutan, and Waktu. The data rows are:

| | K. Udara | K. Tanah | Suhu | Penyiraman | Pengkabutan | Waktu |
|--------------------------|----------|----------|------|-------------|-------------|---------------------|
| <input type="checkbox"/> | 78.0 | 0.0 | 26.8 | Tidak Perlu | Tidak Perlu | 2025-07-04 15:00:02 |
| <input type="checkbox"/> | 78.0 | 0.0 | 26.8 | Tidak Perlu | Tidak Perlu | 2025-07-04 18:00:05 |
| <input type="checkbox"/> | 68.0 | 0.0 | 26.8 | Tidak Perlu | Tidak Perlu | 2025-07-07 15:01:23 |
| <input type="checkbox"/> | 61.0 | 0.0 | 29.3 | Tidak Perlu | Tidak Perlu | 2025-07-07 18:00:03 |
| <input type="checkbox"/> | 61.0 | 0.0 | 29.3 | Tidak Perlu | Tidak Perlu | 2025-07-07 21:00:15 |
| <input type="checkbox"/> | 61.0 | 0.0 | 29.3 | Tidak Perlu | Tidak Perlu | 2025-07-08 00:00:11 |

A small purple button with the number '1' is visible at the bottom left of the table.

Gambar 5. 3 Gambar Halaman Data Sensor

4. Halaman Kontrol Alat

Halaman ini menyediakan dua mode kontrol, yaitu manual dan otomatis, sehingga pengguna dapat dengan mudah memonitor dan mengendalikan alat sesuai kebutuhan. Di bagian atas halaman, ditampilkan status terkini dari penyiraman dan pengkabutan, apakah dalam kondisi aktif (*ON*) atau nonaktif

(OFF). Pengguna dapat menekan tombol "Toggle penyiraman" dan "Toggle pengkabutan" untuk mengaktifkan atau menonaktifkan alat secara manual. Tombol ini memungkinkan kontrol langsung terhadap perangkat, terutama dalam situasi mendesak atau saat dilakukan uji coba sistem. Terdapat juga fitur pengaturan waktu otomatis untuk penyiraman dan pengkabutan. Pengguna dapat mengatur waktu mulai dan waktu selesai untuk setiap proses, lalu menyimpannya dengan tombol *Save*. Ketika waktu yang ditentukan tercapai, sistem akan mengaktifkan alat secara otomatis sesuai dengan jadwal yang telah diset sebelumnya.

Selain berdasarkan waktu, sistem juga menerapkan logika otomatis berbasis pembacaan sensor untuk menghentikan proses penyiraman dan pengkabutan. Proses penyiraman akan berhenti secara otomatis apabila sensor kelembapan tanah mendeteksi nilai lebih dari 80%. Sementara itu, proses pengkabutan akan dihentikan jika suhu udara turun di bawah 27°C dan kelembapan udara lebih dari 71%. Logika ini memastikan bahwa tanaman tidak mengalami kelebihan air atau kelembaban yang berlebih. Di bagian bawah halaman, tersedia tabel Riwayat Aksi yang mencatat semua aktivitas alat, baik yang dilakukan secara otomatis maupun manual. Setiap data mencakup informasi seperti jenis alat, status (aktif/nonaktif), dan waktu pelaksanaan. Tabel ini dilengkapi dengan fitur filter berdasarkan jenis alat, serta opsi penghapusan data secara selektif melalui tombol *Hapus Terpilih*, setelah pengguna mencentang baris data yang diinginkan.

| Jenis Alat | Status | Waktu | |
|-------------|----------|---------------------|--|
| pengkabutan | nonaktif | 2023-07-04 00:15:26 | |
| pengkabutan | aktif | 2023-07-05 00:10:00 | |
| penyiraman | aktif | 2023-07-05 00:15:00 | |
| pengkabutan | aktif | 2023-07-05 00:15:00 | |
| penyiraman | aktif | 2023-07-05 00:15:00 | |
| penyiraman | aktif | 2023-07-06 00:10:00 | |

Gambar 5. 4 Gambar Halaman Kontrol Alat

5. Halaman *Whatsapp*

Pada halaman ini, admin dapat menambahkan nomor *Whatsapp* baru dengan mengisi kolom input yang tersedia, kemudian menekan tombol Save. Nomor yang ditambahkan akan langsung masuk ke dalam daftar, dan sistem akan mulai mengirimkan notifikasi ke nomor tersebut secara otomatis. Tabel yang terdapat di bawah form input menampilkan daftar seluruh nomor *Whatsapp* yang telah terdaftar. Informasi yang disajikan meliputi nomor *Whatsapp*, tanggal dan waktu saat nomor tersebut dibuat, serta kapan terakhir kali diperbarui. Setiap baris juga dilengkapi dengan tombol Hapus yang memungkinkan pengguna menghapus nomor tertentu dari daftar jika tidak lagi diperlukan.

Selain itu, sistem menyediakan fitur pengurutan berdasarkan nomor *Whatsapp* dan pengaturan jumlah baris data yang ingin ditampilkan per halaman. Hal ini memudahkan pengguna dalam mencari, mengelola, dan mengorganisir daftar kontak penerima notifikasi.

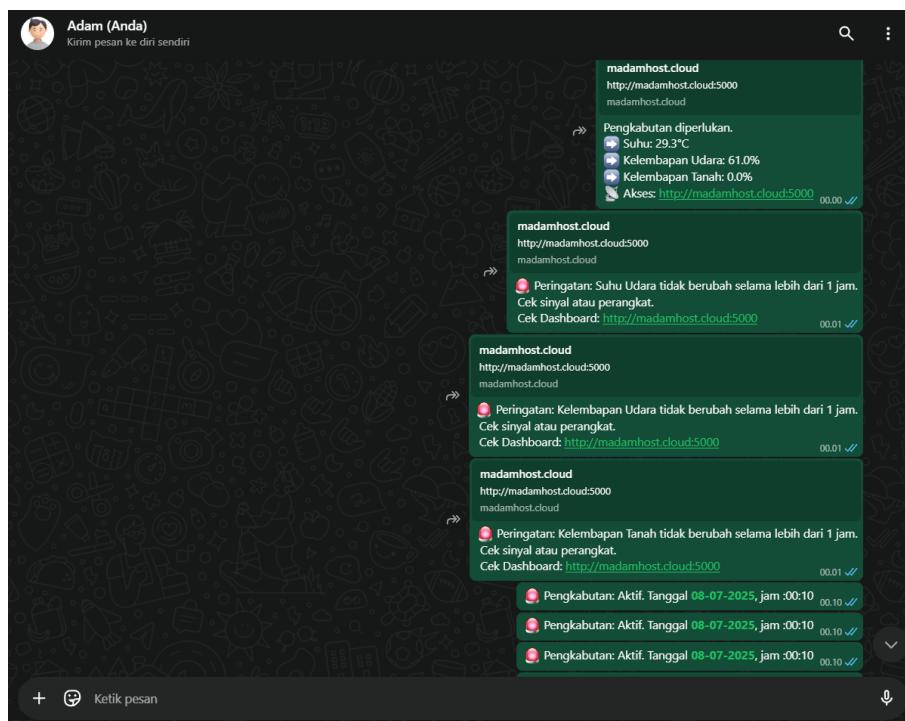
| Nomor WA | Dibuat Sejak | Diubah Sejak | Jenis Aksi |
|---------------|---------------------|---------------------|------------------------|
| 6282118783378 | 2025-06-05 23:06:37 | 2025-06-05 23:06:37 | <button>Hapus</button> |
| 6288220355910 | 2025-07-10 00:28:39 | 2025-07-10 00:28:39 | <button>Hapus</button> |

Gambar 5. 5 Halaman whatsapp

6. Notifikasi *Whatsapp*

Fitur Notifikasi *Whatsapp* merupakan bagian dari sistem monitoring *IoT* yang berfungsi untuk mengirimkan pemberitahuan secara otomatis kepada pengguna melalui pesan *Whatsapp*. Pesan notifikasi dikirimkan secara otomatis oleh *server* berdasarkan analisis data sensor yang diterima. Terdapat beberapa jenis notifikasi yang dikirim, antara lain:

- a. Pertama, sistem akan mengirimkan peringatan kondisi sensor, misalnya jika suhu udara, kelembapan udara, atau kelembapan tanah tidak berubah selama lebih dari satu jam. Notifikasi ini bertujuan untuk menginformasikan kemungkinan gangguan pada sensor atau koneksi perangkat, sehingga pengguna dapat segera melakukan pengecekan.
- b. Kedua, sistem mengirimkan notifikasi prediksi kebutuhan penyiraman atau pengkabutan berdasarkan hasil perhitungan dari algoritma Naive Bayes. Jika kondisi lingkungan menunjukkan bahwa penyiraman atau pengkabutan diperlukan, maka pengguna akan menerima pesan yang menjelaskan nilai suhu, kelembapan udara, dan kelembapan tanah secara detail, serta status akses perangkat. Informasi ini membantu pengguna mengetahui secara langsung kapan tindakan akan atau telah diambil oleh sistem secara otomatis.
- c. Ketiga, sistem juga mengirimkan pesan ketika penyiraman atau pengkabutan diaktifkan, baik berdasarkan jadwal otomatis maupun intervensi manual oleh pengguna. Notifikasi ini mencantumkan waktu dan tanggal pengaktifan alat, sehingga pengguna dapat memverifikasi aktivitas sistem dengan mudah dan cepat.



Gambar 5. 6 Notifikasi *Whatsapp*

5.2 Pengujian

Setelah tahap perancangan dan pembuatan aplikasi selesai, kemudian dilakukan pengujian menggunakan metode *black-box testing* untuk memastikan bahwa fitur-fitur utama berfungsi sesuai dengan spesifikasi. Pengujian dilakukan pada alat mikrokontroller dan aplikasi monitoring, di sajikan dalam dua tabel. Hasil pengujian sistem ditampilkan pada tabel berikut:

1. Pengujian Alat Mikrokontroller

Tabel 5. 1 Pengujian alat mikrokontroller

| No | Item | Skenario | Hasil yang diharapkan | Hasil Pengujian |
|----|----------------------|--|---|-----------------|
| 1 | ESP32 | Mengirim dan menerima data melalui MQTT saat terhubung ke WiFi | ESP32 berhasil publish dan subscribe ke topik MQTT | Valid |
| 2 | DHT11 | Membaca suhu dan kelembapan udara | Data suhu dan kelembapan udara tampil di aplikasi dan dapat memicu pengkabutan jika suhu $> 27^{\circ}\text{C}$ dan kelembapan udara $< 71\%$ | Valid |
| 3 | <i>Soil Moisture</i> | Membaca tingkat kelembapan tanah | Data kelembapan tanah tampil di aplikasi dan dapat memicu penyiraman jika $< 80\%$ | Valid |
| 4 | <i>Relay</i> | Menerima perintah ON/OFF dari ESP32 sesuai perintah | <i>Relay</i> aktif dan nonaktif sesuai perintah dari | Valid |

| | | | | |
|---|-----------------------|---|---|-------|
| | | MQTT | aplikasi (otomatis atau manual) | |
| 5 | Pompa air penyiraman | Aktif saat perintah penyiraman dikirim dan kelembapan tanah < 80% | Pompa menyala saat kondisi terpenuhi dan berhenti saat kelembapan > 80% | Valid |
| 6 | Pompa air pengkabutan | Aktif saat suhu udara > 27°C dan kelembapan < 71% serta menerima perintah pengkabutan | Pompa menyala saat kondisi terpenuhi dan berhenti saat suhu < 27°C dan kelembapan udara > 71% | Valid |

2. Pengujian *Dashboard Monitoring*

Tabel 5. 2 Pengujian *Dashboard Monitoring*

| No | Item | Skenario | Hasil yang diharapkan | Hasil Pengujian |
|----|---------------------------------|---|---|-----------------|
| 1 | <i>Login Admin</i> | Admin memasukkan <i>email</i> dan <i>password</i> yang valid | Sistem menampilkan halaman <i>Dashboard</i> | Valid |
| 2 | Menampilkan data sensor | Sistem menerima data <i>MQTT</i> dari <i>ESP32</i> dan menampilkannya pada <i>card</i> komponen <i>html</i> | Data suhu, kelembapan tanah, dan kelembapan udara tampil secara <i>realtime</i> | Valid |
| 3 | Menyimpan data sensor, prediksi | Sistem menyimpan data sensor dan hasil prediksi dari algoritma Naive | Data sensor dan hasil prediksi tersimpan | Valid |

| | | | | |
|----|---|---|--|-------|
| | penyiraman dan pengkabutan menggunakan algoritma <i>naïve bayes</i> | Bayes ke database secara otomatis setiap jam (03:00, 06:00, 09:00, 12:00, 15:00, 18:00, 21:00, 00:00) | otomatis setiap jam pada waktu yang telah ditentukan | |
| 4. | Menghapus data sensor dan prediksi pada tabel | Admin menekan tombol hapus pada tabel data sensor | Data yang dipilih terhapus dari tabel dan <i>database</i> | valid |
| 5 | Mengaktifkan atau mematikan penyiraman melalui tombol penyiraman | Admin menekan tombol penyiraman manual | <i>Relay</i> pompa air aktif/nonaktif, dan status penyiraman dikirim ke mikrokontroller | Valid |
| 6 | Mengaktifkan atau mematikan pengkabutan melalui tombol pengkabutan | Admin menekan tombol pengkabutan manual | <i>Relay</i> pompa kabut aktif/nonaktif, dan status pengkabutan dikirim ke mikrokontroller | Valid |
| 7 | Mengedit waktu penyiraman | Admin mengatur jadwal penyiraman otomatis | Jadwal penyiraman diperbarui dan disimpan, sistem mengeksekusi sesuai waktu yang diatur | Valid |
| 8 | Mengedit | Admin mengatur jadwal | Jadwal | Valid |

| | | | | |
|----|---|--|--|-------|
| | waktu pengkabutan | pengkabutan otomatis | pengkabutan diperbarui dan disimpan, sistem mengeksekusi sesuai waktu yang diatur | |
| 9 | Menyimpan otomatis Riwayat aksi penyiraman atau pengkabutan | Setiap kali penyiraman/pengkabutan aktif, sistem menyimpan data riwayat ke <i>database</i> | Riwayat tercatat otomatis lengkap dengan waktu, jenis aksi, dan metode pemicunya (otomatis/manual) | Valid |
| 10 | Menghapus data Riwayat aksi | Admin menghapus salah satu atau semua riwayat | Riwayat terhapus dari tampilan dan database | Valid |
| 11 | Menambahkan nomor untuk notifikasi <i>whatsapp</i> | Admin memasukkan nomor telepon dan menyimpannya | Nomor berhasil disimpan ke database dan ditampilkan dalam daftar penerima notifikasi | Valid |
| 12 | Menghapus nomor untuk notifikasi <i>whatsapp</i> | Admin menghapus salah satu nomor dari daftar | Nomor tidak lagi muncul dan tidak menerima notifikasi lagi | Valid |
| 13 | Menampilkan notifikasi <i>whatsapp</i> | Sistem mendeteksi kondisi pemicu (prediksi siram/kabut, suhu/kelembapan dan mengirimkan notifikasi | <i>Whatsapp</i> terkirim ke seluruh nomor yang terdaftar sesuai isi pesan | Valid |

| | | | | |
|----|---------------|------------------------------------|---|-------|
| | | ke <i>whatsapp</i> | yang telah ditentukan | |
| 14 | <i>Logout</i> | Admin menekan tombol <i>logout</i> | Sistem keluar dari sesi dan kembali ke halaman <i>Login</i> | Valid |

BAB VI KESIMPULAN

6.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan oleh penulis melalui beberapa tahapan pada bab-bab sebelumnya, maka penulis dapat menyimpulkan bahwa:

1. Aplikasi penyiraman dan pengkabutan otomatis berbasis *Internet of Things (IoT)* berhasil dibangun dan diimplementasikan menggunakan mikrokontroler ESP32, yang mampu membaca data dari sensor suhu, kelembapan udara, serta kelembapan tanah secara *real-time*.
2. Integrasi sistem dengan algoritma *Naive Bayes* berhasil digunakan untuk melakukan prediksi kebutuhan penyiraman dan pengkabutan secara otomatis berdasarkan parameter lingkungan, sehingga dapat membantu menjaga kondisi optimal untuk pertumbuhan tanaman.
3. Sistem *Dashboard monitoring* berbasis web yang dibangun menggunakan *framework Flask* mampu menampilkan data sensor, hasil prediksi, serta memberikan kontrol manual terhadap penyiraman dan pengkabutan, baik melalui jadwal maupun tombol kontrol.
4. Fitur notifikasi *WhatsApp* berhasil diimplementasikan untuk mengirimkan informasi peringatan, prediksi, dan status aktif penyiraman atau pengkabutan kepada pengguna, sehingga memudahkan pemantauan jarak jauh secara efisien.
5. Berdasarkan hasil pengujian menggunakan metode *black-box testing*, seluruh fitur sistem berjalan dengan baik sesuai skenario yang telah ditentukan, termasuk proses penyimpanan data sensor dan riwayat aksi ke *database*.

6.2 Saran

Berdasarkan hasil penelitian sistem *IoT* untuk penyiraman dan pengkabutan otomatis berbasis *Naive Bayes*, penulis menyadari bahwa masih terdapat beberapa keterbatasan yang dapat diperbaiki dan dikembangkan ke depannya. Adapun beberapa saran yang dapat dipertimbangkan, antara lain:

1. Disarankan menggunakan jaringan WiFi dengan koneksi internet yang lebih stabil dan kuat, terutama jika sistem akan diterapkan di area perkebunan atau *Greenhouse* yang luas dan jauh dari pusat jaringan, agar komunikasi data antara ESP32 dan *server MQTT* tetap lancar.
2. Penggunaan sensor kelembapan tanah dengan tingkat akurasi yang lebih tinggi dapat meningkatkan keandalan sistem dalam mengambil keputusan penyiraman secara otomatis.
3. Penggunaan perangkat *hosting* yang efisien dan memiliki kinerja tinggi sangat disarankan agar proses pengolahan data sensor, prediksi, serta pengiriman notifikasi dapat berjalan lebih cepat, stabil, dan responsif.
4. Sistem keamanan data dapat ditingkatkan dengan menambahkan autentikasi pengguna, enkripsi koneksi *MQTT*, dan log aktivitas untuk mencegah penyalahgunaan sistem.
5. Diperlukan pengujian lanjutan di lingkungan yang lebih beragam, seperti lahan pertanian terbuka dengan berbagai kondisi cuaca, untuk memastikan stabilitas sistem dan ketahanan perangkat keras di lapangan.

DAFTAR PUSTAKA

- Ade Roni. (2025). *Pengertian dan Struktur Dasar HTML*. Aderoni.Com. <https://aderoni.com/pemrograman/pengertian-dan-struktur-dasar-html/>
- AgileTech Vietnam. (2025). *Traditional vs Agile SDLC: 7 Key Practices To Skyrocket Your Project With Agile Model in 2025*. AgileTech Vietnam. <https://agilettech.vn/traditional-sdlc-vs-agile-sdlc/>
- Alamsyah, R., Ryansyah, E., Permana, A. Y., & Mufidah, R. (2024). SISTEM PENYIRAMAN TANAMAN OTOMATIS MENGGUNAKAN LOGIKA FUZZY DENGAN TEKNOLOGI INTERNET OF THINGS BERBASIS ESP8266 DAN APLIKASI BLYNK. *Jurnal Informatika Dan Teknik Elektro Terapan*, 12(2). <https://doi.org/10.23960/jitet.v12i2.4007>
- Ariata C. (2024, May 21). *Apa Itu Hosting Web? Pengertian, Fungsi, dan Jenisnya*. Hostinger. <https://www.hostinger.com/id/tutorial/apa-itu-web-hosting>
- Ayoni Sulthon. (2023, May 27). *Cara membuat ERD: Simbol, Entitas, Atribut Termudah*. DomaiNesia. <https://www.domainesia.com/berita/pengertian-erd-adalah/>
- Ding, B., Yao, F., Wu, Y., & He, Y. (2012). Improving Flask Implementation Using Hardware Assisted In-VM Isolation. In *IFIP AICT* (Vol. 376).
- Effendi, N., Ramadhani, W., & Farida, F. (2022). Perancangan Sistem Penyiraman Tanaman Otomatis Menggunakan Sensor Kelembapan Tanah Berbasis IoT. *Jurnal CoSciTech (Computer Science and Information Technology)*, 3(2), 91–98. <https://doi.org/10.37859/coscitech.v3i2.3923>
- Eka Candra, J., & Maulana Universitas Putera Batam, A. (2019). Penerapan Soil Moisture Sensor Untuk Desain System Penyiram Tanaman Otomatis. *Seminar Nasional Ilmu Sosial Dan Teknologi*.
- Elga Aris Prastyo. (2025). *Memahami Struktur Program Arduino untuk Pemula*. PT Teknolab Caraka Internasional. <https://www.arduinoindonesia.id/2024/05/memahami-struktur-program-arduino-untuk-pemula.html>
- Fahmi, A., Fathul Hadi, C., & Yusa, A. M. (2022). Prototype Sistem Monitoring Suhu Dan Kelembapan Udara Pada Tanaman Cabai Berbasis (IOT). *Zetroem*, Vol 04. No 02.
- Figma, Inc. (2025, May 1). *What is Figma?* Figma, Inc. <https://help.figma.com/hc/en-us/articles/14563969806359-What-is-Figma>
- Fritzing. (2025, May 5). *Fritizing*. Fritzing.

- GitHub, I. (2025, May 1). *About GitHub and Git*. GitHub, Inc. <https://docs.github.com/en/get-started/start-your-journey/about-github-and-git>
- Ibnu Daqiqil Id. (2021). *MACHINE LEARNING: Teori , Studi Kasus dan Implementasi Menggunakan Python* (Ibnu Daqiqil Id, Ed.). UR PRESS . <https://play.google.com/books/reader?id=JvBPEAAAQBAJ&pg=GBS.PR1&hl=id>
- Ifa Susuek Anselmus Talli, W., Dedy Irawan, J., & Xaverius Ariwibisono, F. (2023). RANCANG BANGUN SISTEM MONITORING KUALITAS TANAH UNTUK TANAMAN CABAI BERBASIS IOT (INTERNET OF THINGS). *Jurnal Mahasiswa Teknik Informatika*, 7(5).
- Iftitah Nurul Laily. (2022, February 7). *Pengertian Website Menurut Para Ahli, Beserta Jenis dan Fungsinya*. Katadata.Co.Id. <https://katadata.co.id/lifestyle/edukasi/6200a2a9697ec/pengertian-website-menurut-para-ahli-beserta-jenis-dan-fungsinya>
- Indobot Academy. (2023, June 27). *Mengenal Protokol MQTT dan Perbedaan dengan HTTP*. Indobot Academy. <https://blog.indobot.co.id/mengenal-protokol-mqtt-dan-perbedaan-dengan-http/>
- Irhan Hisyam Dwi Nugroho. (2024, July 12). *Apa itu UML? Pengertian, Jenis, Fungsi, dan Contoh Diagram*. PT Dibimbing Digital Indonesia. <https://dibimbing.id/blog/detail/apa-itu-uml-definisi-fungsi-jenis-contohnya-lengkap>
- Kelasplc. (2023). *Pengertian Internet Of Things : Fitur, Arsitektur dan Contohnya*. Kelas PLC. <https://www.kelasplc.com/pengertian-internet-of-things/>
- Kurniawan. (2023, December 9). *Perbedaan Misting dan Menyiram Tanaman*. DaunSuper. <https://daunsuper.com/perbedaan-misting-dan-menyiram-tanaman/>
- Kusumah, H., & Pradana, R. A. (2019). PENERAPAN TRAINER INTERFACING MIKROKONTROLER DAN INTERNET OF THINGS BERBASIS ESP32 PADA MATA KULIAH INTERFACING. *Journal Cerita*, 5 No 2.
- Liam Aljundi. (2024, January 16). *Using the Arduino Software (IDE)*. Arduino. <https://docs.arduino.cc/learn/startng-guide/the-arduino-software-ide/>
- M. Iqbal Hasani, & Sri Wulandari. (2023). Implementasi Internet of Things (IoT) Pada Sistem Otomatisasi Penyiraman Tanaman Berbasis Mobile. *ILKOMNIKA: Journal of Computer Science and Applied Informatics*, 5(3), 149–161. <https://doi.org/10.28926/ilkomnika.v5i3.573>

- Maya Maharani, D., Malin Sutan, S., & Arimurti, P. (2018). Pengontrolan Suhu Dan Kelembaban (Rh) Terhadap Pertumbuhan Vegetatif Cabai Merah (*Capsicum Annuum L.*) Pada Plant factory. *Jurnal Keteknikan Pertanian Tropis Dan Biosistem*, 6(2), 120–134.
- Microsoft. (2025, May 5). *What is the difference between Visual Studio Code and Visual Studio IDE?* Visual Studio Code. https://code.visualstudio.com/docs/supporting/faq#_what-is-the-difference-between-visual-studio-code-and-visual-studio-ide
- Mozilla Foundation. (2025). *What is CSS?* Mozilla Corporation's. https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Styling_basics/What_is_CSS
- Mozilla Foundation. (2025, May 5). *Introduction Javascript*. Mdn Web Docs. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction>
- Muhamad Rusdi, Muriani, Rivaldo Pasca Corpusty, Mardiyasa Putra Yoga, Grace Christin Aditya Ronsumbre, & Diah Bayu Titisari. (2023). IMPLEMENTASI TEKNOLOGI PENYIRAMAN SISTEM PENGKABUTAN OTOMATIS DAN MONITORING PINTAR BERBASIS TENAGA SURYA UNTUK TEMPAT BUDIDAYA TANAMAN ANGGREK UDFAIRUS DI KABUPATEN MERAUKE. *JURNAL PENGABDIAN MASYARAKAT*, 1(Vol. 1 No. 2 (2023): AKSELERASI: Jurnal Pengabdian Masyarakat), 53–59. <https://doi.org/https://doi.org/10.70210/ajpm.v1i2.40>
- MySQLTutorial.org. (2008). *MySQL Architecture*. MySQLTutorial.Org. <https://www.mysqltutorial.org/mysql-administration/mysql-architecture/>
- Nadifa Padantya Raihanah. (2023, January 20). *Apa Itu Figma? Fitur, Kelebihan dan Kekurangan*. Alterra Academy. <https://academy.alterra.id/blog/apa-itu-figma/>
- Pallets. (2025, May 1). *Welcome to Flask*. Pallets. <https://flask.palletsprojects.com/en/stable/>
- Publikasi, A. J., Mardiansyah, A., Kasah, B. N., Zamzami, H. R., Arabu, Y., Nasro, M. A., Kristanto, N., Paojiah, R., & Wulandari, Y. (2025). PENGENALAN DASAR HTML DAN CSS: LANGKAH PERTAMA DALAM PENGEMBANGAN WEB. *Abdi Jurnal Publikasi*, 3(3), 165–170. <https://jurnal.portalpublikasi.id/index.php/AJP/index>
- python org. (2025, May 5). *What is Python? Executive Summary*. Python.Org. <https://www.python.org/doc/essays/blurb/>
- Rifqi Mulyawan Digital. (2025). *Pengertian JavaScript: Sejarah, Cara Kerja JavaScript dan Manfaatnya*. Rifqi Mulyawan Digital. <https://rifqimulyawan.com/blog/pengertian-javascript/>

- Roihan, A., Abas Sunarya, P., & Rafika, A. S. (2019). IJCIT (Indonesian Journal on Computer and Information Technology) Pemanfaatan Machine Learning dalam Berbagai Bidang: Review paper. *IJCIT (Indonesian Journal on Computer and Information Technology)*, 5(1), 75–82.
- S Nursuwars, F. M., & Sujana, D. G. (2018). IoT: Kelembaban Tanah dan Suhu Ruang sebagai Parameter Sistem Otomatis Penyiraman Air Bawah dan Atas Tanah. *Jurnal Transistor Elektro Dan Informatika (TRANSISTOR EI)*, 3(3).
- Solahart Handal. (2025, May 5). *Apa itu Nozzle Sprayer?* Solahart Handal. <https://www.solaharthandal.com/jenis-jenis-nozzle-sprayer/>
- Togi. (2021, September 14). *Mengenal Bahasa Pemrograman Arduino Secara Lengkap yang Mudah Dipelajari untuk Pemula.* PT Tekno Gemilang Indonesia. <https://toghrl.com/bahasa-pemrograman-arduino/>
- Ucy Sugiarti. (2024, November 12). *Activity Diagram: Komponen, Elemen, Beserta Contohnya.* Lawencon International. <https://www.lawencon.com/activity-diagram/>
- Valerie Lampkin, W. T. L. L. O. S. R. N. S. R. X. G. K. N. K. S. F. M. K. D. L. (2012). *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry:* Vol. (1st Edition). IBM Redbooks. https://play.google.com/books/reader?id=F_HHAgAAQBAJ&pg=GBS.PR9.w.10.0.32_29&hl=id
- Walid, Moh., & Susanto, A. (2024). Penyiraman Otomatis Menggunakan Arduino Uno pada Tanaman Greenhouse MA. Nurul Khoiroh. *Jurnal Ilmu Komputer Dan Informatika*, 4(1), 11–20. <https://doi.org/10.54082/jiki.121>
- Wardani, Hadi, S., & Budiarto, J. (2021). Rancang Bangun Sistem Monitoring Suhu dan Kelembaban Udara Pada Ruang Server Berbasis Wireless Sensor Network. *JURNAL TEKNOLOGI TERPADU*, 9(2).
- Web Idea Solution. (2025). Hire Python Developer to Upscale Your Product Development Capabilities. *Web Idea Solution.* <https://webideasole.com/python-developer/>
- Yazid Yusuf. (2024, December 16). *Apa Itu MySQL? Pengertian MySQL, Cara Kerja, dan Kelebihannya.* Telkom University. <https://bif.telkomuniversity.ac.id/apa-itu-mysql/>

LAMPIRAN

Lampiran 1:Hasil Wawancara

Wawancara ini bertujuan untuk memperoleh data penelitian tentang proses penyiraman dan pengkabutan di Avicenna *Greenhouse*.

Berikut adalah detail wawancara yang dilakukan:

Narasumber : Resa Aldiana
Jabatan / Posisi : Pemilik *Greenhouse*
Hari, tanggal : 12 Mei 2025
Lokasi : Avicenna *Greenhouse*
Alamat : Kp. Padarek Rt. 03 Rw. 02 Desa Drawati Kec. Paseh Kab. Bandung, Prov. Jawa Barat

Berikut adalah hasil wawancara:

| No | Pertanyaan | Jawaban |
|----|--|---|
| 1 | Kapan Avicenna <i>Greenhouse</i> mulai dibangun dan digunakan untuk menanam cabai? | Avicenna <i>Greenhouse</i> mulai dibangun dan digunakan sejak tahun 2024 |
| 2 | Mengapa Bapak memilih menanam cabai di dalam <i>Greenhouse</i> ? | Supaya tanaman bisa lebih terlindungi dari serangan hama dan cuaca ekstrim. |
| 3 | Berapa lama waktu yang dibutuhkan tanaman cabai sampai bisa dipanen? | Tanaman cabai biasanya membutuhkan waktu sekitar 4 bulan dari masa tanam hingga panen. |
| 4 | Bagaimana metode penyiraman tanaman dilakukan saat ini | Saat ini penyiraman masih manual. Kami menyalakan keran, lalu air dialirkan melalui selang ke setiap <i>polybag</i> . |
| 5 | Apakah ada permasalahan dalam proses penyiraman tanaman? | Kadang penyiraman tergantung pada kehadiran orang. Jika tidak ada yang menyiram, tanaman bisa kekurangan air. |
| 6 | Apakah pengkabutan juga sudah dilakukan di <i>Greenhouse</i> ini? | Saat ini, sistem pengkabutan belum diterapkan sama sekali. |
| 7 | Untuk penyiraman pada | Biasanya dilakukan satu kali, maksimal dua |

| | | |
|---|--|--|
| | tanaman dilakukan pada waktu jam berapa saja? | kali sehari, yaitu jam 7 pagi dan jam 5 sore. |
| 8 | Bagaimana pandangan Bapak terhadap penggunaan sistem <i>IOT</i> untuk otomatisasi penyiraman dan pengkabutan di <i>Greenhouse</i> ini? | Sangat mendukung penggunaan <i>IOT</i> . Dengan adanya sistem otomatis berbasis sensor suhu dan kelembapan, proses penyiraman dan pengkabutan bisa lebih efisien, hemat tenaga, dan menunjang pertumbuhan tanaman lebih optimal. |

Bandung, 12 Mei 2025

Pewawancara



Adam Setiadi

Narasumber



Resa Aldiana

Lampiran 2: Dokumentasi Wawancara

Narasumber : Resa Aldiana
Jabatan : Pemilik *Greenhouse*
Tanggal : 12 Mei 2025
Lokasi : Avicenna *Greenhouse*
Alamat : Kp. Padarek Rt. 03 Rw. 02 Desa Drawati Kec. Paseh Kab. Bandung, Prov. Jawa Barat



Lampiran 3: TOR

Sebelum melaksanakan penelitian, penulis mengumpulkan data melalui observasi, wawancara, dan studi pustaka di Avicenna *Greenhouse*. Penelitian ini difokuskan pada proses penyiraman dan pengkabutan tanaman yang masih dilakukan secara manual. Tujuan dari penelitian ini adalah mengimplementasikan sistem *Internet of Things* untuk mengontrol penyiraman dan pengkabutan secara otomatis, serta memanfaatkan data sensor untuk prediksi dan akurasi proses tersebut. Untuk memastikan fokus penelitian adapun batasan masalah yang ditetapkan adalah sebagai berikut:

1. Penelitian ini berfokus pada pengembangan aplikasi penyiraman dan pengkabutan otomatis berbasis *IOT* untuk tanaman cabai di *Greenhouse* Avicenna.
2. Aplikasi yang dikembangkan memanfaatkan sensor kelembapan tanah, suhu, dan kelembapan udara sebagai parameter utama dalam proses penyiraman dan pengkabutan.
3. Aplikasi ini menggunakan algoritma *Naïve bayes* dalam pengambilan keputusan untuk menentukan kapan penyiraman dan pengkabutan dilakukan berdasarkan data sensor.
4. Pengembangan aplikasi mencakup integrasi dengan platform berbasis web untuk monitoring dan pengendalian sistem secara *real-time*, tetapi tidak mencakup fitur lanjutan seperti rekomendasi pemupukan atau analisis pertumbuhan tanaman.
5. Penelitian ini menggunakan metode *Agile* dalam pengembangan aplikasi guna meningkatkan fleksibilitas dalam implementasi dan evaluasi.

Bandung, 12 Mei 2025

Mahasiswa

Pemilik Avicenna *Greenhouse*



Adam Setiadi



Resa Aldiana

Lampiran 4: Instalasi Alat Di *Greenhouse*

1. Tata Letak Alat Mikrokontroller box



2. Tata Letak Pompa air pengkabutan



3. Tata Letak Pompa air penyiraman



4. Tata Letak Terminal Listrik



5. Tata Letak Selang Penyiraman



6. Tata Letak Selang Pengkabutan

