



INSTITUT TEKNOLOGI DEL

IoT Centralized Control and Monitoring System

TUGAS AKHIR

Oleh :

13318019	Rizky Romuel Sitorus
13318029	Michael Ollifarel Sagala
13318040	Yeremia Wirdyanti Tambunan

FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO

PROGRAM STUDI DIII TEKNOLOGI KOMPUTER

SITOLUAMA

2021



INSTITUT TEKNOLOGI DEL

IoT Centralized Control and Monitoring System

DOKUMEN TUGAS AKHIR

Oleh :

13318019	Rizky Romuel Sitorus
13318029	Michael Ollifarel Sagala
13318040	Yeremia Wirdyanti Tambunan

Diajukan sebagai salah satu syarat untuk memperoleh gelar Diploma Teknik

FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO

PROGRAM STUDI DIII TEKNOLOGI KOMPUTER

SITOLUAMA

2021

HALAMAN PERNYATAAN ORISINALITAS

Tugas Akhir ini adalah hasil karya saya sendiri, dan semua sumber baik yang dikutip maupun dirujuk telah saya nyatakan dengan benar.

Nama : Michael O Sagala

NIM : 13318029

Tanda Tangan : 

Tanggal : 18 Agustus 2021

Nama : Rizky R Sitorus

NIM : 13318019

Tanda Tangan : 

Tanggal : 18 Agustus 2021

Nama : Yeremia W Tambunan

NIM : 13318040

Tanda Tangan : 

Tanggal : 18 Agustus 2021

HALAMAN PENGESAHAN

Tugas Akhir ini diajukan oleh :

Nama : Michael O Sagala
NIM : 13318029
Program Studi : Diploma III Teknologi Komputer

Nama : Rizky R Sitorus
NIM : 13318019
Program Studi : Diploma III Teknologi Komputer

Nama : Yeremia W Tambunan
NIM : 13318040
Program Studi : Diploma III Teknologi Komputer

Judul Tugas Akhir : *IoT Centralized Control and Monitoring System*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai persyaratan yang diperlukan untuk memperoleh gelar Diploma Teknik, pada program studi Diploma III Teknologi Komputer, Fakultas Informatika dan Teknik Elektro, Institut Teknologi Del.

DEWAN PENGUJI

Pembimbing : Eka Stephani Sinambela, S.ST., M.Sc ()
Penguji 1 : Istas Pratomo Manalu, S.Si., M.Sc ()
Penguji 2 : Ahmad Zatnika Purwalaksana, S.Si., M.Si ()

Ditetapkan di : Laguboti

Tanggal : Agustus 2021

**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR
UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademik Institut Teknologi Del, kami yang bertanda tangan di bawah ini:

Nama	:	Michael O Sagala
NIM	:	13318029
Fakultas/Program Studi	:	Diploma III Teknologi Komputer
Nama	:	Rizky R Sitorus
NIM	:	13318019
Fakultas/Program Studi	:	Diploma III Teknologi Komputer
Nama	:	Yeremia W Tambunan
NIM	:	13318040
Fakultas/Program Studi	:	Diploma III Teknologi Komputer
Jenis Karya	:	Tugas Akhir

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Institut Teknologi Del **Hak Bebas Royalty Noneksklusif (Non-exclusive Royalty-Free Right)** atas karya ilmiah yang berjudul: **IoT Centralized Control and Monitoring System**

Beserta perangkat yang ada (jika diperlukan). Dengan **Hak Bebas Royalty Noneksklusif** ini Institut Teknologi Del berhak menyimpan, mengalih/media-format dalam bentuk pangkalan data (database), merawat, dan mempublikasikan tugas akhir ini selama tetap mencantumkan nama diatas sebagai penulis /pencipta dan sebagi pemilik hak cipta.

Demikian pernyataan ini kami buat dengan sebenarnya.

Dibuat di Laguboti , Agustus 2021

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas segala rahmat dan cinta kasih yang diberikan kepada penulis, sehingga dapat menyelesaikan Tugas Akhir ini dengan baik. Dengan harapan penulisan Tugas Akhir ini dapat memberikan informasi dan pengetahuan kepada pembaca yaitu *IoT Centralized Control and Monitoring System*. Adapun tujuan dalam penulisan Tugas Akhir ini adalah sebagai salah satu syarat dalam kelulusan Diploma III Teknologi Komputer di Institut Teknologi Del.

Dalam penulisan Tugas Akhir ini tentunya penulis menyampaikan terimakasih kepada :

1. Ibu Eka Stephani Sinambela, S.ST., M.Sc sebagai dosen pembimbing yang telah memberikan arahan, masukan serta dukungan kepada penulis selama pengerjaan Tugas Akhir ini.
2. Bapak Ahmad Zatnika Purwalaksana, S.Sc., M.Si dan Bapak Istas Pratomo Manalu, S.Si., M.Sc sebagai penguji yang telah memberikan masukan dalam pengerjaan Tugas Akhir ini dan sebagai koordinator Tugas Akhir D3 Teknologi Komputer 2020/2021.
3. Orang tua, saudara, sahabat dan semua pihak yang membantu dan memberikan doa serta dukungan selama pengerjaan Tugas Akhir ini.

Dengan penulisan Tugas Akhir ini penulis berharap sekiranya dapat memberikan manfaat bagi semua pihak yang membacanya dan dapat dijadikan sebagai bahan untuk pengembangan ide. Penulis juga menyadari dalam penulisan laporan Tugas Akhir ini banyak kekurangan. Oleh karena itu, penulis berharap saran maupun masukan yang membangun demi perbaikan penelitian ini di masa mendatang.

Situluama, Juni 2021

Michael O. P. Sagala

Rizky R Sitorus

Yeremia W Tambunan

ABSTRAK

Program Studi : Diploma 3 Teknologi Komputer

Judul : ***IoT Centralized Control and Monitoring System***

Penerapan Teknologi IoT (*Internet of Things*) yang mampu mengoptimalkan sistem dengan melakukan monitoring dan pengontrolan secara lebih realtime dan efisien di era sekarang ini. Pemanfaatan jaringan internet merupakan faktor yang sangat mendukung terhadap perkembangan teknologi yang bersifat otomatis maupun manual. Sehingga membuat beberapa sistem dijalankan ataupun di monitoring secara terpisah. Akan tetapi, pengontrolan secara desentralisasi masih umum dalam pengembangan sistem IoT. Sebagai contoh, sebuah sistem pemantauan tanaman bekerja menghidupkan lampu jika intensitas cahaya berada di bawah nilai 200. Saat mengganti tumbuhan yang memerlukan cahaya lebih banyak, maka mikrokontroler harus di *code* ulang agar menghidupkan lampu jika intensitas cahayanya berada di bawah 300. Hal ini tentu sangat memakan banyak waktu.

Oleh karena itu, penulis mencoba untuk membuat suatu sistem manajemen peralatan IoT untuk mempermudah kontrol dan monitoring *device* IoT. Pengguna dapat menggunakan berbagai sensor, dengan berbagai jumlah tanpa langsung melakukan *code* untuk menentukan sebuah nilai agar mikrokontroler dapat mengambil keputusan. Sistem ini juga mendukung melakukan kontrol secara langsung terhadap perangkat. Hal ini dapat diimplementasikan dengan mematikan atau menghidupkan lampu melalui jaringan. Data yang dikumpulkan sepenuhnya dimiliki oleh pengguna, sehingga tidak memerlukan pengaruh pihak ke-3 dalam pengelolaan datanya. Data tersebut akan disajikan melalui sebuah *Interface Web* yang sudah disediakan.

Dengan adanya sistem ini, diharapkan dapat mempermudah pengguna dalam manajemen perangkat IoT yang dimilikinya dengan kemampuan melakukan kontrol dan monitoring secara terpusat.

Kata Kunci : *IoT, device, Web Interface, control, monitoring*

ABSTRACT

Study Program : Diploma 3 Teknologi Komputer

Title : IoT Centralized Control and Monitoring System

Application of IoT (Internet of Things) technology that is able to optimize the system by monitoring and controlling in a more real-time and efficient manner in today's era. Utilization of the internet network is a very supportive factor for the development of technology that is both automatic and manual. Thus making several systems run or monitored separately. However, decentralized control is still common in the development of IoT systems. For example, a hydroponic system works to turn on the lights if the light intensity is below 200. When replacing plants that require more light, the microcontroller must be re-coded to turn on the lights if the light intensity is below 300. This is of course very consuming a lot time.

Therefore, the author tries to create an IoT equipment management system to facilitate the control and monitoring of IoT devices. Users can use various sensors, with various amounts without directly doing the code to determine a value so that the microcontroller can make decisions. .The system also supports direct control of the device. This can be implemented by turning the lights off or on over the network. The data collected is wholly owned by the user, so it does not require the influence of 3rd parties in managing the data. The data will be presented through a web interface that has been provided.

With this system, it is hoped that it will make it easier for users to manage their IoT devices with the ability to centrally control and monitor them.

Keyword : *IoT, device, Web Interface, control, monitoring*

DAFTAR ISI

KATA PENGANTAR.....	5
ABSTRAK	6
DAFTAR ISI	8
DAFTAR TABEL	10
DAFTAR GAMBAR.....	11
BAB I	14
1.1 Latar Belakang	14
1.2 Rumusan Masalah	17
1.3 Tujuan	17
1.4 Lingkup	17
1.5 Pendekatan	18
1.5.1 Studi Literatur	18
1.5.2 Analisis	18
1.5.3 Perancangan Desain	18
1.5.4 Implementasi Sistem.....	19
1.5.5 Testing	19
1.6 Sistematika Penyajian	19
BAB II	21
2.1 Landasan Teori.....	21
2.1.1 Internet of Things.....	21
2.1.2 Protokol HTTP.....	22
2.1.3 RESTful API.....	23
2.1.4 API Gateway.....	25
2.1.5 JSON (JavaScript Object Notation)	26
2.1.6 MongoDB	27
2.2 Kajian Penelitian Yang Relevan	28
2.2.1 Sistem Kontrol dan Monitoring Aplikasi IoT [22]	28
2.2.2 Perancangan Database IoT Berbasis Cloud dengan Restful API [23]	29
BAB III.....	30
3.1. Analisis Masalah	30
3.2 Analisis Pemecahan Masalah	31
3.3 Analisis dan Penentuan Kebutuhan	32
3.3.1 Analisis Kebutuhan Perangkat Keras (<i>Hardware</i>).....	32
3.3.2 Analisis Kebutuhan Perangkat Lunak (<i>Software</i>)	34
3.4 Perancangan dan Arsitektur Sistem.....	34
3.4.1 Gambaran Umum Sistem.....	34
3.4.2 Perancangan Arsitektur <i>Hardware</i>	39
3.4.3 Komunikasi Data	40
3.5 Fungsi dan Flowchart.....	41

3.5.1 Fungsi Kirim Data dari HTTP <i>Body</i> ke API.....	41
3.5.2 <i>Flowchart</i> Fungsi Kirim Data dari HTTP <i>Body</i> ke API.....	41
3.5.3 Fungsi Tranlasi JSON ke dalam query mongoDB	43
3.5.4 <i>Flowchart</i> Fungsi <i>Translasi</i> JSON ke dalam query mongoDB	43
3.5.5 Fungsi Membuat URL untuk Mengontrol <i>Agent</i>	44
3.5.6 <i>Flowchart</i> Fungsi Membuat URL untuk Mengontrol <i>Agent</i>	44
3.5.7 Fungsi <i>Update</i> Data ke MongoDB	46
3.5.8 <i>Flowchart</i> Fungsi <i>Update</i> Data ke MongoDB	46
3.5.9 Fungsi CRUD Data ke MongoDB	47
3.5.10 <i>Flowchart</i> Fungsi CRUD data ke MongoDB.....	48
3.5.11 Fungsi Terima <i>Request</i>	51
3.5.12 <i>Flowchart</i> Fungsi Terima <i>Request</i>	51
3.5.13 Fungsi Rata-rata data	53
3.5.14 <i>Flowchart</i> Fungsi Rata-rata data.....	53
3.6 Desain	54
3.6.1 Desain Web Sistem.....	55
3.6.2 Desain Database.....	62
3.7 Skenario Pengujian.....	63
3.7.1 Skenario Pengujian Fungsi CRUD <i>Agent</i> Pada Aplikasi Web.....	64
3.7.2 Skenario Pengujian Komunikasi <i>Agent</i> - API <i>Gateway</i> – Database – UI.....	66
BAB IV	67
4.1 Implementasi.....	67
4.1.1 Implementasi Wemos D1 ESP8266.....	67
4.1.2 Implementasi Rangkaian pada Prototype Kontroling Lampu	70
4.1.3 Implementasi Rangkaian pada Prototype Pemantauan Tanaman.....	70
4.1.4 Implementasi Kode	72
4.1.5 Implementasi Web Aplikasi.....	78
4.2 Pengujian.....	83
4.2.1 Pengujian Fungsi CRUD Pada Aplikasi Web	83
4.2.2 Pengujian Prototype Device IoT Tipe <i>Manual Control</i>	101
4.2.3 Pengujian Prototype Device IoT Tipe Automatic Control	107
4.2.4 Pengujian Komunikasi <i>Agent</i> ke API <i>Gateway</i> Menuju Database.....	111
4.2.4 Pengujian Database ke <i>User Interface</i>	113
4.2.5 Pengujian Waktu.....	115
4.2.6 Butir Uji Coba.....	117
BAB V	123
5.1 Kesimpulan	123
5.2 Saran	123
Daftar Pustaka	125
LAMPIRAN	127

DAFTAR TABEL

Tabel 1 Spesifikasi Kebutuhan Perangkat Keras	32
Tabel 2 Spesifikasi Kebutuhan Perangkat Lunak	34
Tabel 3 Hasil Pengujian Kontroling Lampu	101
Tabel 4 Hasil Pengujian Sensor DHT11	107
Tabel 5 Hasil Pengujian Sensor DHT11	108
Tabel 6 Hasil Pengujian Sensor Ultrasonik	108
Tabel 7 Hasil Pengujian Sensor LDR	109
Tabel 8 Butir Uji Fungsi Kirim Data dari Http Body ke API	117
Tabel 9 Butir Uji Fungsi Translasi JSON ke dalam Query MongoDB	118
Tabel 10 Butir Uji Fungsi Membuat URL untuk Mengontrol Agent	118
Tabel 11 Butir Uji Fungsi Update Data ke MongoDB	119
Tabel 12 Butir Uji Fungsi CRUD Data ke MongoDB	120
Tabel 13 Butir Uji Fungsi Terima Request	121
Tabel 14 Butir Uji Fungsi Rata-rata	122

DAFTAR GAMBAR

Gambar 1. Komunikasi Protokol HTTP	23
Gambar 2 Arsitektur API Gateway	26
Gambar 3 Gambaran Umum Sistem.....	35
Gambar 4 Komunikasi Data Secara Umum.....	38
Gambar 5 Desain Arsitektur Hardware	39
Gambar 6 Komunikasi Data	40
Gambar 7 Flowchart Fungsi Kirim Data dari HTTP Body ke API	42
Gambar 8 Flowchart Fungsi Translasi JSON ke dalam query mongoDB	43
Gambar 9 Flowchart Fungsi Membuat URL untuk Mengontrol Agent.....	45
Gambar 10 Flowchart Fungsi Update Data Ke MongoDB.....	47
Gambar 11 Flowchart Fungsi CRUD Data ke dalam MongoDB	48
Gambar 12 Flowchart Fungsi Create	49
Gambar 13 Flowchart Fungsi Update.....	50
Gambar 14 Flowchart Fungsi Delete.....	51
Gambar 15 Flowchart Fungsi Terima Request	52
Gambar 16 Flowchart Fungsi Rata-Rata Data.....	54
Gambar 17 Tampilan Halaman Awal Aplikasi Web	55
Gambar 18 Tampilan Halaman Login Aplikasi Web	56
Gambar 19 Tampilan Halaman Dashboard Aplikasi Web	57
Gambar 20 Tampilan Halaman Tambah Device	58
Gambar 21 Tampilan Halaman Tambah Device Automatic Control	59
Gambar 22 Tampilan Halaman Tambah Device Manual Control	60
Gambar 23 Tampilan Halaman Edit Device Automatic Control.....	61
Gambar 24 Tampilan Halaman Edit Device Manual Control	62
Gambar 25 Skenario Pengujian Fungsi CRUD	64
Gambar 26 Skenario Pengujian Komunikasi Agent-API Gateway-Database-UI	66
Gambar 27 Implementasi Wemos D1 R1 Pada Software Arduino (1)	67
Gambar 28 Implementasi Wemos D1 R1 Pada Software Arduino (2)	68
Gambar 29 Implementasi Wemos D1 R1 Pada Software Arduino (3)	68
Gambar 30 Implementasi Wemos D1 R1 Pada Software Arduino (4)	69
Gambar 31 Implementasi Wemos D1 R1 Pada Software Arduino (5)	69
Gambar 32 Implementasi Rangkaian Pada Prototype Kontroling Lampu.....	70
Gambar 33 Implementasi Rangkaian Pada Prototype Tanaman Sayur	71
Gambar 34 Tampilan Halaman Login	78
Gambar 35 Tampilan Halaman Dashboard	79
Gambar 36 Tampilan Halaman Tambah Agent.....	80
Gambar 37 Tampilan Halaman Tambah Agent Automatic Control	81

Gambar 38 Tampilan Halaman Tambah Agent Manual Control.....	81
Gambar 39 Tampilan Halaman Edit Agent Automatic Control	82
Gambar 40 Tampilan Halaman Edit Agent Manual Control	83
Gambar 41 Pengujian Penambahan Agent Kontroling Lampu (1)	84
Gambar 42 Pengujian Penambahan Agent Kontroling Lampu (2)	85
Gambar 43 Pengujian Penambahan Agent Kontroling Lampu (3)	85
Gambar 44 Pengujian Penambahan Agent Kontroling Lampu (4)	86
Gambar 45 Pengujian Edit Device Kontroling Lampu (1)	87
Gambar 46 Pengujian Edit Device Kontroling Lampu (2)	87
Gambar 47 Pengujian Edit Device Kontroling Lampu (3)	88
Gambar 48 Pengujian Edit Device Kontroling Lampu (4)	88
Gambar 49 Pengujian Delete Device Kontroling Lampu (1)	89
Gambar 50 Pengujian Delete Device Kontroling Lampu (2)	89
Gambar 51 Pengujian Penambahan Agent Automatic Control(1)	90
Gambar 52 Pengujian Penambahan Agent Automatic Control (2)	91
Gambar 53 Pengujian Penambahan Agent Automatic Control (3)	92
Gambar 54 Pengujian Penambahan Agent Automatic Control (4)	93
Gambar 55 Pengujian Penambahan Agent Automatic Control (5)	94
Gambar 56 Pengujian Edit Device Automatic Control (1)	95
Gambar 57 Pengujian Edit Device Automatic Control (2)	96
Gambar 58 Pengujian Edit Device Automatic Control (3)	97
Gambar 59 Pengujian Edit Device Automatic Control (4)	98
Gambar 60 Pengujian Edit Device Automatic Control (5)	99
Gambar 61 Pengujian Edit Device Automatic Control (6)	100
Gambar 62 Pengujian Delete Device Automatic Control (1)	101
Gambar 63 Pengujian Delete Device Automatic Control (2)	101
Gambar 64 Tampilan Halaman Kondisi Lampu OFF	102
Gambar 65 Tampilan Prototype Kondisi Lampu OFF	102
Gambar 66 Tampilan Web Kondisi Lampu Depan ON	103
Gambar 67 Tampilan Prototype Kondisi Lampu Depan ON	103
Gambar 68 Tampilan Web Kondisi Lampu Depan ON	104
Gambar 69 Tampilan Prototype Kondisi Lampu Tengah ON	104
Gambar 70 Tampilan Web Kondisi Lampu Belakang ON	105
Gambar 71 Tampilan Prototype Kondisi Lampu Belakang ON	105
Gambar 72 Tampilan Web Kondisi Semua Lampu ON	106
Gambar 73 Kondisi Prototype Semua Lampu ON	106
Gambar 74 Pengujian Monitoring Device Automatic Control	110
Gambar 75 Komunikasi Agent Ke API Gateway	111

Gambar 76 Data yang Tersimpan dalam Database.....	112
Gambar 77 Status Berhasil Penyimpanan Data	112
Gambar 78 Pemberitahuan Database dan Web Berhasil Terkoneksi.....	113
Gambar 79 Data Sensor Dari Pengguna	114
Gambar 80 Rata-rata nilai dari setiap sensor	115

BAB I

PENDAHULUAN

Pada bab ini berisi penjelasan mengenai latar belakang, tujuan pelaksanaan, ruang lingkup, pendekatan serta sistematika penyajian Dokumen.

1.1 Latar Belakang

Istilah “*Internet of Things*” (IoT) terdiri atas dua bagian utama yaitu internet yang mengatur konektivitas dan *things* yang berarti objek atau perangkat. IoT adalah sebuah gagasan dimana semua benda di dunia nyata dapat berkomunikasi satu dengan yang lain sebagai bagian dari satu kesatuan sistem terpadu yang menggunakan jaringan internet sebagai penghubung [1]. Konsep IoT dengan cara kerja mengacu pada 3 elemen utama pada arsitektur IoT, seperti barang fisik yang dilengkapi modul IoT, perangkat koneksi ke internet seperti modem dan Router, dan *cloud data center* tempat untuk menyimpan aplikasi beserta database [2]. Dalam arsitekturnya, IoT terdiri dari perangkat keras, perangkat lunak, dan web, karena perbedaan protokol antara perangkat keras dengan protokol web, maka diperlukan *embedded system* berupa *gateway* untuk menghubungkan dan menjembatani perbedaan protokol tersebut[3].

Manfaat yang dimiliki IoT dalam kehidupan sehari-hari sangat berpengaruh besar, diantaranya manfaat yang dimiliki IoT dapat berguna sebagai monitoring lingkungan contohnya pengecekan kondisi air, sebagai pengelolaan infrastruktur contohnya MRT yang dibuat untuk transportasi cepat, sebagai media sensor peralatan yang biasanya digunakan pada perusahaan pertambangan, dan masih banyak lagi. Seperti yang kita ketahui, teknologi *Internet Of Things* (IoT) telah menjadi bagian dari kehidupan sehari-hari. Misalnya sistem *smarthome* yang memanfaatkan teknologi IoT. Secara tidak langsung, IoT telah menyediakan berbagai solusi untuk permasalahan di sekitar kita. Dalam beberapa tahun terakhir, IoT ditingkatkan dengan adanya sensor dan aktuator (*Things*). *Things* itu merupakan sesuatu yang bisa dikendalikan melalui internet. Dapat dikaitkan pada sistem IoT ada perangkat komputasi yang saling terkait, dimana gabungan sensor dan aktuator berfungsi untuk meningkatkan interaksi objek terhadap kondisi sekeliling atau terhadap objek lain [4].

Kebanyakan dalam sebuah perangkat IoT, kontrol dan pemantauan biasanya menggunakan kontrol parameter manual oleh manusia. Dalam implementasinya, biasanya dalam satu perangkat IoT memiliki satu mikrokontroler sehingga melakukan konfigurasi harus secara manual terhadap setiap perangkat IoT tersebut. Hal ini tentu akan memakan banyak waktu jika dalam suatu ekosistem memiliki beberapa perangkat IoT yang bervariasi. Dalam suatu ekosistem IoT, terdapat sebuah keluaran yang penting, yaitu data. Data dapat berupa nilai sensor atau status dari sebuah aktuator[2]. Dalam pengolahan datanya, beberapa perangkat IoT menggunakan layanan pihak ke tiga, salah satu contohnya adalah Thingspeak. Thingspeak merupakan platform *open source* aplikasi IoT yang digunakan untuk menyimpan dan mengambil data dari sesuatu menggunakan protokol HTTP [5]. Salah satu kekurangan dari penggunaan layanan pihak ke tiga adalah akses terhadap data yang terbatas.

Dalam pengaplikasian IoT, sebuah perangkat IoT akan terdiri dari beberapa sensor dan aktuator, dan sebuah mikrokontroler. Untuk melakukan kontrol terhadap mikrokontroler itu, masih harus dilakukan penggantian secara manual di *source code* nya, dan melakukan *flash* ulang ke mikrokontrolernya. Hal ini tentu akan memakan banyak waktu jika harus melakukan hal yang sama ke beberapa perangkat IoT.

Didasari oleh permasalahan di atas, penulis mencoba untuk membangun sistem dengan konsep *agent* dan *master*, dimana *master* yang melakukan monitoring dan kontroling terhadap beberapa *agent*nya. *Agent* adalah sebuah perangkat IoT yang terdiri dari mikrokontroler yang bertugas sebagai otak untuk proses pengendalian [6], sensor dan aktuator, salah satu contohnya adalah suatu pemantauan yang terdiri dari beberapa sensor. *Agent* biasanya terdistribusi sesuai dengan kebutuhan dari pengguna. *Master* adalah sebuah aplikasi yang berfungsi untuk mengumpulkan data, mengelola dan melakukan kontrol terhadap *agent*. Dengan mengaplikasikan ini, pengguna dapat dengan mudah mengubah nilai acuan yang ada di dalam perangkat IoT melalui aplikasi *master*, sehingga tidak lagi harus melakukan perubahan secara langsung di mikrokontroler. Nilai acuan adalah sebuah nilai yang menjadi salah satu tolak ukur dari mikrokontroler dalam mengatur aktuatornya.

Dalam pengerjaannya, *agent* akan secara berkala memberikan datanya ke aplikasi *master*. Kemudian, aplikasi *master* akan membuat *decision* dari data yang diterima dan *state* yang diberikan oleh pengguna. Kemudian melalui *decision* ini akan melakukan controlling

terhadap perangkat IoT tersebut. *Master* juga akan menyimpan data dan menyajikan data tersebut ke pengguna.

Berdasarkan hal tersebut, pada pengerjaan tugas akhir ini kelompok membuat implementasi *master* dan *agent* dengan 2 jenis *agent* yang berbeda. *Agent* yang pertama berupa pengolahan sensor yang menggunakan lebih dari satu sensor dimana pengontrolannya diambil berdasarkan nilai sensor dan *state* yang dimasukkan oleh pengguna (otomatis) dimasukkan dalam studi kasus pemantauan tanaman sayuran. Jenis *agent* yang ke dua adalah melakukan kontrol terhadap beberapa lampu (*on/off*), pengontrolan ini dilakukan langsung oleh pengguna (manual).

Sebelumnya juga sudah ada penelitian yang mengerjakan proyek yang sama, yaitu sistem untuk mengontrol dan memonitoring aplikasi iot[22], dimana sistem yang dibangun ini mampu memenuhi semua tujuan, perangkat lunak untuk meminta, mengirim, dan menerima data serta penyimpanan posteriornya di DB yang telah dikembangkan. Sistem ini memungkinkan pemantauan jarak jauh atas data yang dipasok dari sensor webcam dan kontrol perangkat yang berbeda seperti aktuator, motor servo, dan LED. Keamanan komunikasi dalam sistem juga dijamin melalui SSH antara Raspberry Pi dan server.

Pada penelitian selanjutnya juga berhubungan dengan sistem yang dibangun yaitu mengenai perancangan database IoT berbasis *cloud* dengan Restful API[23], karena pada sistem yang dibangun juga menggunakan arsitektur Restful API dalam pengiriman datanya. Pada penelitian ini, membahas tentang penyimpanan data yang telah dikumpulkan oleh perangkat IoT. Seperti yang kita ketahui penyimpanan perangkat IoT itu sangat kecil, jadi dibutuhkan sebuah media yang digunakan untuk megumpulkan data tersebut. *API Gateway* yang akan menjadi penghubung antara perangkat IoT dengan database, yang akan mengelola segala komunikasi antara perangkat dengan basis data. *API gateway* yang menjadi sebuah jembatan untuk melakukan CRUD (*create, read, update and delete*) dalam sistem.

Dengan adanya sistem ini, diharapkan dapat membantu pengguna dalam memantau dan mengontrol beberapa *device* IoT yang terdistribusi dan bervariasi secara terpusat. Dan dengan adanya aplikasi berbasis *web* yang digunakan sebagai antar muka pengguna untuk memudahkan pengguna dalam memantau kondisi dari lingkungan dengan kemampuan penggunaan sensor secara fleksibel pada sistem yang dibangun.

1.2 Rumusan Masalah

Rumusan masalah berdasarkan latar belakang diatas adalah :

1. Bagaimana merancang dan membangun sistem yang dapat memonitoring dan mengontrol lebih dari satu *agent* secara terpusat dengan menggunakan aplikasi *master* ?

1.3 Tujuan

Adapun tujuan dari Tugas Akhir ini adalah untuk membangun sistem yang mampu mengatur dan memantau lebih dari satu *device* IoT yang terdistribusi secara sentralisasi dengan menggunakan konsep *master* dan *agent*, dimana *master* akan menyajikan data kepada pengguna dan *master* dapat digunakan oleh pengguna untuk mengontrol *agent*. Dengan demikian, pemantauan IoT yang sebelumnya memiliki sistem pengelolaan secara independen, terdistribusi dan bervariasi dapat diatur secara terpusat. Diharapkan juga, dengan adanya sistem pengaturan dan pemantauan tersentralisasi ini akses data dimiliki secara penuh oleh pengelola sistem.

1.4 Lingkup

Ruang lingkup dari Tugas Akhir ini merupakan gambaran yang akan dilakukan dalam perancangan produk yang dibangun dan batasan dalam produk.

Tugas Akhir ini memiliki ruang lingkup sebagai berikut :

1. Pembangunan aplikasi berbasis web sebagai *master* yang berfungsi untuk mengelola dan memantau beberapa *device* IoT.
2. Pembangunan sebuah sistem monitoring dan kontroling IoT yang terdiri dari mikrokontroler, sensor dan aktuator.
3. Mengumpulkan, dan menampilkan data yang diterima di aplikasi *master*.
4. Pembangunan 2 tipe *agent*, *agent* tipe pertama akan dikontrol secara otomatis berupa pengolahan beberapa sensor dan *agent* tipe kedua adalah pengontrolan secara manual berupa kontrol terhadap lampu.

Dalam pengerjaan tugas akhir ini terdapat batasan-batasan yang dimiliki oleh sistem yang harus diketahui oleh pengguna :

1. *User* diharuskan untuk memastikan tidak ada indeksasi data yang salah pada *client* dan *master*.
2. Ketersinambungan antara perangkat dengan *web* aplikasi tidak menggunakan nirkabel atau masih menggunakan kabel.
3. Pada aplikasi *web* yang dibangun belum tersedia fitur untuk memastikan status dari aktuator apakah sedang aktif/tidak aktif.

1.5 Pendekatan

Pendekatan yang dilakukan untuk mengembangkan produk *IoT Centralized Control and Monitoring System* ini adalah sebagai berikut :

1.5.1 Studi Literatur

Pada tahapan ini, kelompok akan mengumpulkan data atau informasi bagaimana tata cara merancang *device* iot dan mempelajari sumber pendukung yang relevan lainnya.

1.5.2 Analisis

Pada tahapan ini, kelompok menyiapkan kebutuhan dalam pembangunan proyek. Analisis kebutuhan yang dilakukan meliputi analisis masalah, usulan penyelesaian, identifikasi pengguna, kebutuhan proses, kebutuhan antarmuka dan kebutuhan perangkat lunak dan perangkat keras yang digunakan.

1.5.3 Perancangan Desain

Pada tahapan ini, kelompok menyiapkan rancangan sistem yang nantinya akan digunakan pada tahap implementasi. Membuat gambaran sistem yang akan dibangun, desain akan dibangun untuk menjelaskan bagaimana *device* iot dan komunikasi data melalui *master* dan *agent*.

Gambaran pada perancangan produk dan sistem ini yaitu ada dua *device* IoT yang bervariasi akan dikontrol melalui satu *master*. Mikrokontroler dari setiap *agent* adalah Wemos D1 ESP8266 atau Node MCU yang bertugas untuk mengolah data dari setiap sensor, mengirim data ke aplikasi *master*, menerima dan menjalankan perintah dari *master*, dan mengontrol aktuator.

1.5.4 Implementasi Sistem

Pada tahapan ini, kelompok mengaplikasikan hasil rancangan sistem yang telah dibuat. Pada tahapan implementasi juga diterapkan sejumlah materi yang telah dijelaskan pada tahapan sebelumnya.

Pengimplementasian tugas akhir ini berupa sistem yang mengontrol dan memonitoring beberapa *device* IoT secara terpusat. Dimana setiap komponen yang ada pada *device* IoT tersebut akan dikontrol secara terpusat dari sebuah web aplikasi yang akan dibangun.

1.5.5 Testing

Dalam tahap ini dilakukan pengujian mengenai kinerja sistem dan produk yang telah dibangun. Pengujian ini bertujuan untuk memeriksa kebenaran dari fitur dan alat yang telah dianalisis sebelumnya dan mengetahui tingkat akurasi dari sistem yang telah dikembangkan. Kemudian melalui hasil pengujian yang telah didapatkan maka akan dianalisis kembali tercapai atau tidaknya tujuan dari tugas akhir yang diharapkan. Adapun *testing* yang akan dilakukan dalam produk ini adalah pengujian terhadap setiap *device* IoT pada saat menerima data dari lingkungan kemudian data yang didapat diterima oleh mikrokontroler dan kemudian dikirimkan ke server kemudian akan dapat dikendalikan oleh web aplikasi master berjalan dengan baik.

1.6 Sistematika Penyajian

Dokumen ini disajikan dalam lima bab dengan sistematika penulisan sebagai berikut.

1. Bab I Pendahuluan

Bab I terdiri dari penjelasan mengenai latar belakang, tujuan pelaksanaan, lingkup, pendekatan yang dilakukan. Bab ini diakhiri dengan sistematika penulisan laporan.

2. Bab II Tinjauan Pustaka

Bab II menguraikan landasan teori yang digunakan pada pengerjaan tugas akhir.

3. Bab III Analisis dan Perancangan Sistem

Bab III menjelaskan tentang metode mengenai analisis dan perancangan sistem pada pengerjaan tugas akhir.

4. Bab IV Implementasi dan Pengujian

Bab IV menjelaskan implementasi dalam pembangunan sistem serta pengujian pada implementasi yang dilakukan.

5. Bab V Kesimpulan dan Saran

Bab V menjelaskan kesimpulan dari hasil pengerjaan sistem serta saran yang diperlukan untuk pengembangan sistem selanjutnya.

BAB II

TINJAUAN PUSTAKA

Pada bab tinjauan pustaka dijelaskan teori yang mendukung pengerjaan tugas akhir ini juga berdasarkan kebutuhan tugas akhir.

2.1 Landasan Teori

Pada bagian landasan teori ini akan diuraikan secara ringkas mengenai teori-teori atau konsep yang digunakan dalam membangun *IoT Centralized Control and Monitoring System*.

2.1.1 Internet of Things

Semakin berkembangnya teknologi informasi dan komunikasi, ada berbagai macam benda yang dapat terkoneksi dengan internet. Bukan hanya komputer dengan *smartphone* melainkan peralatan elektronik lainnya. *Internet of Things* atau sering disingkat dengan IoT merupakan istilah yang sering digunakan sebagai bukti berkembangnya teknologi informasi dan komunikasi.

Internet of Things merupakan struktur dimana objek disediakan dengan identitas eksklusif dan mempunyai kemampuan untuk memindahkan data melalui jaringan tanpa memerlukan dua arah antara manusia yaitu sumber ke tujuan atau interaksi manusia ke komputer [7]. Singkatnya, benda - benda disekitar kita dapat berkomunikasi satu sama lain melalui jaringan internet. Ide *Internet of Things* dimunculkan pertama kali oleh Kevin Ashton pada tahun 1999 [8] dan sudah banyak digunakan di oleh perusahaan, seperti microsoft, intel dan lainnya.

“A Things” pada *Internet of Things* adalah sebagai subjek, seperti mobil yang dilengkapi dengan sensor. IoT paling erat hubungannya dengan komunikasi *machine-to-machine* (M2M) di bidang manufaktur, listrik dan perminyakan. Produk dibangun dengan kemampuan komunikasi M2M sering disebut dengan sistem cerdas atau *smart* [9].

Dalam penerapannya, *Internet of Things* dapat mengidentifikasi, menemukan dan memantau objek secara otomatis dan *real time*. *Internet of Things* juga banyak ditemukan dalam berbagai aktivitas, seperti *live streaming*, *e-commerce*, pemesanan tiket *online* dan sebagainya. Dengan semakin maraknya penggunaan *Internet of Things* maka akan membuat segala sesuatunya menjadi mudah.

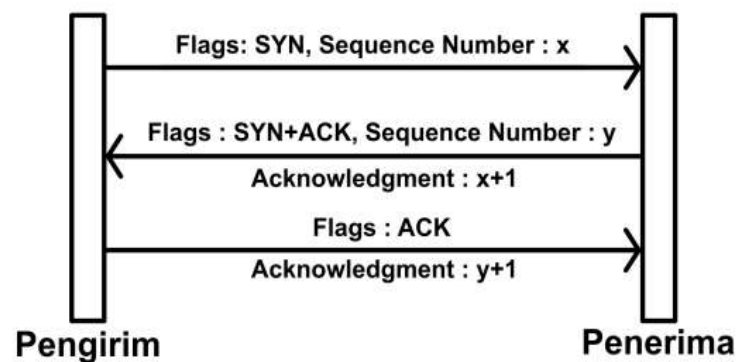
2.1.2 Protokol HTTP

Pada penyelesaian tugas akhir ini, protokol yang akan dibahas adalah HTTP. *Hypertext Transfer Protocol* (HTTP) adalah protokol tingkat aplikasi stateless untuk sistem informasi hiperteks terdistribusi, kolaboratif [13], dan menggunakan berbagai metode *request*, *header*, dan kode error dari protokol HTTP. HTTP menggunakan pesan *request* dari *client* ke *server* untuk melakukan suatu tugas tertentu kepada sebuah objek di server yang akan dibalas dengan pesan *response*, pesan mengenai status pengerjaan tugas tersebut, dari *server* ke *client*. HTTP memiliki beberapa komponen yang digunakan untuk melakukan *request* dan *response*. Beberapa komponen tersebut yaitu URI, *request* dan *response*, dan metode HTTP.

Pesan *request* memiliki beberapa metode pengiriman yang digunakan untuk identifikasi proses yang akan dilakukan oleh server. Sehingga, berbagai proses yang berbeda dapat dilakukan pada satu URI yang sama. Berbagai metode yang terdapat pada HTTP adalah :

1. **OPTIONS.** Metode ini digunakan untuk mempresentasikan *request* informasi berkaitan dengan metode yang tersedia pada *server*.
2. **GET.** Metode yang digunakan untuk mengambil seluruh informasi yang teridentifikasi oleh Request URI. *Response* akan mengirimkan balik informasi yang di dapat dalam bentuk *message-body* di dalam pesan *response*.
3. **HEAD.** Digunakan untuk mendapatkan informasi *header* dari sebuah *response* tanpa *message-body* yang ada apabila menggunakan metode GET.
4. **POST.** Metode ini digunakan untuk mengirim *request* dengan sebuah data dalam *body* HTTP nya [14]. Respon dari *request* ini ditentukan oleh server.
5. **PUT.** Metode ini digunakan untuk modifikasi data apabila terdapat *resource* dengan URI yang sama, atau membuat *resource* baru apabila URI dari *request* tidak ada di server.
6. **DELETE.** Metode untuk menghapus *resource* sesuai dengan URI yang ada di dalam *request* [14].
7. **TRACE.** Metode yang digunakan untuk melihat apa yang diterima oleh server dari hasil *request*. Hal ini digunakan untuk keperluan diagnosis dan analisis masalah yang terjadi dalam metode *request*.

HTTP adalah protokol yang menggunakan TCP sebagai protokol transpornya. Pengiriman pesan menggunakan HTTP dijamin menggunakan TCP 3-way *handshake* [15]. TCP 3-way *handshake* adalah algoritma yang digunakan TCP untuk membuka dan menutup koneksi antara klien dan server. Algoritma ini bekerja dengan mengirimkan 3 (tiga) pesan antara kedua pihak untuk memastikan bahwa klien dapat menyetujui parameter untuk membuka koneksi TCP antara kedua pihak. Gambar dibawah akan menunjukkan bagaimana algoritma ini bekerja.



Gambar 1. Komunikasi Protokol HTTP

(Sumber : <https://jagad.id/pengertian-handshaking/>)

Pesan SYN yang dikirimkan klien akan mengirimkan *sequence number* yang akan diterima oleh server. Server kemudian akan merespon dengan pesan berisi ACK atau pengakuan *sequence number* tersebut dan *sequence number* dari server sendiri. Klien kemudian akan mengirimkan pesan terakhir ACK untuk pengakuan klien terhadap *sequence number* milik server. Apabila *handshake* tidak menerima respon yang diharapkan dalam jangka waktu tertentu, maka TCP akan melakukan transmit ulang untuk segmen tersebut.

2.1.3 RESTful API

RESTful API merupakan implementasi dari API (*Application Programming Interface*). REST (*Representational State Transfer*) merupakan seperangkat prinsip arsitektur yang melakukan transmisi data melalui antarmuka yang terstandarisasi seperti HTTP [17]. RESTful sendiri merupakan istilah yang dipakai untuk layanan web yang mengimplementasikan arsitektur REST. Dimana tujuannya adalah untuk membuat sebuah sistem yang memiliki performa yang baik, cepat dan mudah untuk digunakan [19]. Untuk mengakses suatu resource, ada beberapa hal yang harus diperhatikan oleh *developer* yaitu URI (*Uniform Resource Identifier*) sebagai identitas lokasi *resource*, HTTP *method*

(GET,POST,DELETE,PUT) [17], *header* sebagai meta informasi suatu *request*, dan *body* sebagai isi data yang ingin dikirimkan. Keluaran dari RESTful API dapat berupa format pertukaran JSON (*JavaScript Object Notation*) atau XML (*Extensible Markup Language*) [18]. Kedua format tersebut dapat digunakan pada berbagai bahasa pemrograman seperti PHP, Java, Python, dan lain sebagainya. Penggunaan API saat ini dapat membantu *developer* dalam menyingkat waktu pengembangan aplikasi. Sehingga *developer* tidak perlu membuat fitur yang sudah ada, tetapi langsung menggunakan layanan yang menyediakan fitur tersebut.

REST API bekerja layaknya seperti pada *web* biasa. *Client* dapat mengirimkan permintaan ke *server* melalui protokol HTTP dan kemudian *server* memberikan respons balik kepada klien. Arsitektur REST menjelaskan enam batasan, adapun keenam batasan arsitektur REST adalah sebagai berikut :

a. *Client Server*

Batasan *client server* menjelaskan suatu antarmuka yang memisahkan bagian *client* dan *server*. Melalui pemisahan antarmuka, REST memberikan keuntungan yaitu *client* tidak perlu berurusan dengan masalah penyimpanan . Hal tersebut meningkatkan portabilitas antarmuka pengguna di berbagai *platform* dan meningkatkan skalabilitas dengan menyederhanakan komponen *server*. Pemisahaan antarmuka pengguna memungkinkan komponen dapat terus berkembang secara independen. *Uniform Interface* yang menghubungkan antara *client* dan *server*.

b. *Stateless*

Batasan ini menjadi hal penting pada arsitektur REST. Batasan ini menjelaskan bahwa *server* tidak menyimpan *state* atau penanda *client*. Maksudnya adalah setiap pesan yang dikirimkan oleh *client* bersifat *self-descriptive* atau dengan kata lain setiap pesan yang dikirimkan memiliki informasi atau konteks yang cukup untuk *server* dalam memproses pesan tersebut. Setiap *state* atau penanda *session* tersimpan pada pihak *client*.

c. *Chaceable*

Batasan ini menjelaskan bahwa respon *server* bersifat *cacheable*. Setiap respons *server* dapat disimpan secara implisit, eksplisit, atau *negotiated*. Setiap browser biasanya akan melakukan *cache* terhadap semua *request* GET [16].

d. *Uniform Interface*

Uniform Interface menjelaskan antarmuka antara *client* dan *server*. Hal ini menyederhanakan dan memisahkan arsitektur kedua pihak, yang memungkinkan setiap bagian dapat berkembang secara independen. Batasan ini merupakan hal fundamental untuk desain RESTful. RESTful menggunakan *HTTP method* (GET, POST, PUT, DELETE) untuk menjelaskan metode permintaan, URI untuk mengidentifikasi nama sumber, *HTTP response (status, body)* untuk menjelaskan informasi yang dikembalikan oleh *server*.

e. *Layered System*

Batasan ini menjelaskan bahwa pihak *client* tidak bisa secara langsung terhubung ke *server*. Terdapat perantara antara *server* dan *client*. Hal tersebut berkaitan dengan poin pemisahan antara *client* dan *server*. Perantara tersebut meningkatkan skalabilitas dengan memungkinkan *load-balancing* dan dengan menyediakan *cache* bersama. Serta *layered system* dapat menerapkan kebijakan keamanan.

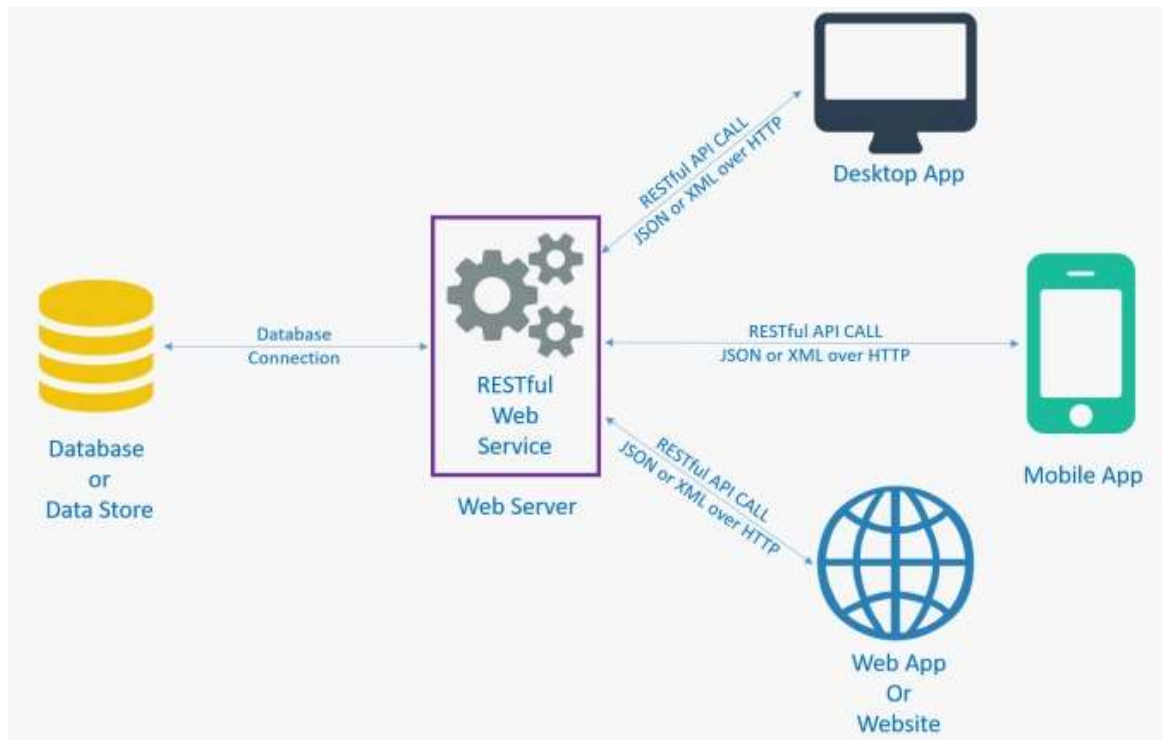
f. *Code on Demand* (Opsional)

Batasan ini menjelaskan bahwa *server* dapat memberikan atau menyesuaikan fungsionalitas *client* secara sementara dengan mentransferkan *logic* ke dalamnya sehingga dapat dijalankan. Contohnya yaitu komponen yang dikompilasi seperti Java applet dan JavaScript.

2.1.4 API Gateway

API Gateway merupakan gerbang dari beberapa API, bertugas sebagai management API, merge beberapa API, *authentication* API dan lain-lain. API Gateway juga memiliki *load balancer* yang memastikan bahwa permintaan akan di proses ke *endpoint* API yang ingin diakses sesuai *request* pengguna. *API Gateway* merupakan gerbang dari beberapa API, bertugas sebagai management API, *merge* beberapa API, *authentication* API dll. *API Gateway* tidak terlepas dari microservice yang merupakan kumpulan dari beberapa service atau kumpulan dari beberapa API.

Pada pengerjaan tugas akhir ini kelompok menggunakan *API Gateway* sebagai gerbang komunikasi antara database dan *device*. Arsitektur dari API dapat dilihat pada **Gambar 2** dibawah ini.



Gambar 2 Arsitektur API Gateway

2.1.5 JSON (JavaScript Object Notation)

JavaScript Object Notation (JSON) merupakan lightweight data interchange yang berbasis JavaScript Programming Language [20]. JSON berbasis teks/tulisan dengan format yang dapat dibaca dan dikenali oleh manusia untuk merepresentasikan struktur data sederhana dan array asosiatif. JSON merupakan bahasa independen yang lengkap dan menggunakan konvensi yang familiar bagi para programmer bahasa C, antara lain bahasa pemrograman C, C++, C#, Java, JavaScript, Perl, Python, dan lainnya. Kelompok lebih memilih menggunakan format data JSON daripada XML karena proses loading data lebih ringan karena ukuran file yang kecil dan penulisan kode yang cepat dengan kode yang sederhana.

Pada tugas akhir ini penerapan JSON digunakan untuk :

- Bertukar data *client* dan *server* atau antar aplikasi , contohnya RESTful API.
- Tempat menyimpan data , contohnya : Database MongoDB.
- Menyimpan konfigurasi proyek, contohnya : file composer.json pada proyek PHP dan package.json pada Nodejs.
- Menyimpan konfigurasi dan menyimpan data pada aplikasi.

Struktur dasar JSON , selalu diawali dengan tanda kurung kurawal { }. Lalu di dalam kurung kurawal terdapat data format key dan value, jika terdapat lebih dari satu data maka dipisah dengan tanda koma dan di data terakhir tidak diberikan koma. Dapat dilihat contoh code JSON:

```
{
  "id": "403:27:56062021",
  "sensor_value": [
    10,10,10,10
  ],
  "tanggal":null,
  "waktu":null
}
```

Pada contoh diatas, *value* data disimpan setelah nilai *key*. Sebagai contoh pada baris kedua, nilai 403:27:56062021 disimpan di *key* id. Sebuah *value* dapat berupa *int*, *string* atau sebuah *array*. Untuk mengakses nilai *value*, kebanyakan bahasa pemrograman hanya perlu mengakses *key*nya untuk mengakses *valuenya*.

2.1.6 MongoDB

MongoDB adalah sebuah *database* yang bersifat *Open Source* yang memiliki *High Performance*. MongoDB merupakan sebuah *database* dengan konsep manajemen *database* berorientasi dokumen yang dibuat menggunakan Bahasa pemrograman C++. *Database* berorientasi dokumen adalah sebuah program komputer yang dirancang untuk menyimpan, mengambil dan mengelola data yang berorientasi dokumen. MongoDB merupakan basis data yang tidak relasional, hal ini membuat MongoDB sangat cepat saat melakukan proses manipulasi data dari pada sistem basis data relasional (RDBMS), selain itu MongoDB berbasis dokumen sehingga tidak memiliki struktur yang tidak teratur seperti tabel. Kelebihan MongoDB dibandingkan *database* yang lain adalah dapat melakukan *searching* lebih cepat, tidak perlu membuat struktur tabel karena MongoDB otomatis membuatnya, jadi hanya perlu melakukan insert saja, mempercepat proses CRUD (*create, read, update, delete*) digunakan oleh banyak website.

MongoDB berisi *collection*. Setiap *collection* terdiri dari *fields*. Sebuah *collections* dapat di *indexes*, yang meningkatkan kinerja pengurutan data. Ada 6 konsep yang harus dimengerti adalah :

1. MongoDB memiliki konsep yang sama dengan *data storage* lainnya seperti MySQL atau Oracle. MongoDB dapat tidak memiliki *data storage* atau lebih dari satu *data storage*, masing-masingnya bertindak sebagai “*high level containers*”.
2. Sebuah *data storage* dapat tidak memiliki *collection* atau lebih dari satu *collection*. Sebuah *collection* memiliki banyak kesamaan dengan tabel tradisional pada *data storage* seperti MySQL. Sebuah *collection* dan tabel tradisional dalam hal ini dapat dianggap sama.
3. Sebuah *data store* dapat tidak memiliki *documents* atau lebih dari satu *documents*. Sebuah *documents* dapat dianggap sama dengan sebuah *row* pada tabel tradisional.
4. Sebuah *documents* terdiri dari satu atau lebih *fields* atau *columns*.
5. *Indexes* di MongoDB seperti *Indexes* di *Relational Data Management System*.
6. *Cursors* pada MongoDB digunakan untuk meminta atau memanggil data.

MongoDB menyimpan data berupa dokumen BSON. BSON adalah representasi binari oleh tipe data JSON. Struktur dasar dari BSON adalah *field: value*. Contoh struktur dokumen tipe BSON dapat dilihat:

```
{
  name: "pemantauan_tanaman 1",
  nilai_sensor: [10,2],
  status: "hidup"
}
```

Setiap *field* akan diikuti oleh *valuenya*. Pemisah dari *field* dan *value* ini adalah titik dua (':'). *Value* dapat berupa tipe data *integer*, *string*, *character*, dan *array*.

2.2 Kajian Penelitian Yang Relevan

Sub bab ini membahas tentang penelitian yang berkaitan dalam tugas akhir ini.

2.2.1 Sistem Kontrol dan Monitoring Aplikasi IoT [22]

Pada penelitian ini membahas terkait dengan sistem pengendalian dan pemantauan yang berfokus pada pemasangan *Internet Of Things* (IoT) yang berhasil diimplementasikan menggunakan perangkat lunak dan perangkat keras *open source* berdasarkan Raspberry Pi. Dimana Sistem yang dibangun mampu memenuhi semua tujuan, perangkat lunak untuk meminta, mengirim, dan menerima data serta penyimpanan posteriornya di DB yang telah dikembangkan. Sistem ini memungkinkan pemantauan jarak jauh atas data yang dipasok dari sensor webcam dan kontrol perangkat yang berbeda seperti aktuator, motor servo, dan LED. Parameter yang dimonitor adalah kecerahan, suhu dan kelembaban relatif yang

merupakan faktor lingkungan yang memungkinkan. Pengendalian dan pemantauan instalasi diwujudkan melalui server yang dikelola oleh seorang administrator. Perangkat yang mengatur penginstalan adalah Raspberry Pi, komputer mikro kecil dan kuat dalam satu papan dengan konsumsi rendah, biaya rendah, dan dapat diatur ulang. Keamanan komunikasi dalam sistem dijamin melalui SSH antara Raspberry Pi dan server. Ini termasuk konfigurasi server dan DB yang mengelola sistem. Sistem terbuka dan lengkap ini dapat digunakan di beberapa aplikasi IoT yang menghemat banyak energi dibandingkan dengan menggunakan komputer/laptop biasa.

2.2.2 Perancangan Database IoT Berbasis Cloud dengan Restful API [23]

Pada penelitian ini membahas fungsi sebuah *gateway API* kepada sebuah sistem IoT. Perangkat IoT memiliki kapasitas penyimpanan yang sangat kecil, sehingga tidak bisa digunakan sebagai media penyimpanan data. Oleh karena itu dibutuhkan sebuah media penyimpanan lainnya untuk menyimpan data yang telah dikumpulkan oleh perangkat IoT tersebut. *API gateway* dapat menjadi sebuah pintu penghubung antara *device* IoT dengan sebuah atau lebih basis data. Sebuah *API gateway* diharapkan dapat mengelola segala komunikasi antara *device* dengan basis data. Sebuah API Gateway dapat menjadi sebuah jembatan untuk melakukan *create*, *read*, *update* dan *delete* dalam sebuah sistem. Namun dalam sistem yang dilakukan, masih menggunakan basis data SQL.

BAB III

ANALISIS DAN DESAIN

Pada bab ini menjelaskan analisis yang akan dilakukan terhadap sistem perangkat keras dan tahapan perancangan lunak yang mencakup analisis masalah, analisis pemecahan masalah, analisis kebutuhan sistem, perancangan dan desain pada tugas akhir ini.

3.1. Analisis Masalah

Pengontrolan dan pemantauan terhadap suatu kondisi dalam sebuah *device* IoT saat ini masih dilakukan secara desentralisasi oleh manusia, baik dalam melakukan konfigurasi masih dilakukan secara manual terhadap masing-masing *device* IoT. Contoh kasus jika pada suatu tempat terdapat beberapa *device* IoT yang bervariasi, misalnya pemantauan tanaman menggunakan beberapa sensor, pengontrolan lampu (*on/off*) dll. Sistem monitoring dan pemantauan masih dilakukan secara terpisah oleh manusia. Untuk melakukan konfigurasi juga masih dilakukan secara manual terhadap setiap *device* IoT tersebut dengan melakukan konfigurasi di setiap *microcontroller*. Jika ada penambahan beberapa *device* IoT yang bervariasi di suatu tempat tersebut maka pengguna IoT harus melakukan konfigurasi ulang khusus untuk *device* IoT yang ditambahkan tersebut. Dan jika ada beberapa *device* IoT yang sedang berjalan dan beberapa dari *device* IoT tersebut sama maka dalam pengontrolannya masih dilakukan secara terpisah atau independen.

Adapun juga beberapa kasus biasanya, ada beberapa *device* IoT yang dijalankan dan dikontrol secara terpusat bahkan dalam satu tempat ada sistem yang sama, yang menjadi persoalannya adalah bagaimana cara mengintegrasikannya supaya dapat diketahui bahwa sistem tertentu berjalan sesuai perintah yang diberikan. Otomatis hal yang dilakukan adalah dengan melakukan konfigurasi manual terhadap masing-masing sistem.

Dengan adanya permasalahan seperti itu maka akan mengganggu dalam hal konfigurasi dan akan mengganggu jika penambahan suatu *device* IoT baru dapat menyebabkan gangguan untuk *device* IoT yang lainnya maka akan ada perbaikan kembali terhadap *device* IoT tersebut.

3.2 Analisis Pemecahan Masalah

Untuk mengatasi masalah yang didapat, maka dibutuhkan sistem sentral yang dapat berkomunikasi dengan setiap *device* IoT dan mengolah datanya di dalam satu jaringan. Sistem sentral ini juga harus bisa menerima data sefleksibel mungkin dari setiap *device* IoT, baik dalam bentuk jumlah sensor, aktuator dan tipe data yang diolah. Sistem sentral yang dapat diaplikasikan untuk masalah tersebut adalah membuat komunikasi dua arah antara *server* dan *device* IoTnya. Setiap data yang diperoleh oleh *device* akan dikirimkan menuju *server* untuk diolah. Dari data yang diolah, *server* kemudian akan mengirimkan perintah terhadap setiap *device* dari pengolahan data yang didapatkan dari setiap *device*.

Dalam pengiriman data, dapat diimplementasikan arsitektur RESTful API, yaitu suatu arsitektur metode komunikasi yang menggunakan protokol HTTP pada port 80 pada setiap *agent* dan *API Gateway* untuk pertukaran data. Setiap data akan dikirim menuju *server* dalam format data JSON. JSON merupakan format pertukaran data yang cepat karena sintaksnya kecil dan bobot data yang ringan. Setiap file JSON yang diterima, akan langsung disimpan dan diolah oleh *server*. Dalam komunikasi antara server dengan *device* IoT, *server* akan mengirimkan HTTP *request* kepada *device* IoT, kemudian akan menerima dan melakukan perintah yang dikirimkan oleh *server*.

Pada pengerjaan tugas akhir ini method *request* yang digunakan adalah Method POST. Karena jika menggunakan method POST pada saat pengiriman data keamanan datanya lebih aman dibanding dengan keamanan data method yang lainnya. Method POST digunakan untuk mengirim data yang biasanya digunakan untuk menambah atau merubah data pada server. Pada protokol HTTP, method POST dapat dikirim baik melalui query string maupun body. Adapun kelebihan dan kekurangan pengiriman data menggunakan metode POST adalah lebih aman dari metode GET karena data yang dikirim tidak terlihat, serta parameter yang dikirim tidak disimpan pada history browser, dapat mengirim data dalam jumlah besar, dapat mengirim berbagai jenis data seperti pada gambar, file, dll dan tidak harus teks, namun kekurangan dari method POST ini data tidak disimpan di history browser, data tidak dapat di bookmark karena dianggap sebagai data sensitive, maka ketika merefresh browser, akan muncul konfirmasi pengiriman ulang data.

Dengan sistem ini, maka akan memudahkan pengguna dalam melakukan konfigurasi terhadap *device* karena hanya perlu melakukan kontroling dan monitoring secara terpusat.

Dan untuk permasalahan mengintegrasikan setiap *device*, dapat diintegrasikan dengan membangun sebuah aplikasi berbasis web yang akan menjadi *User Interface* bagi sistem, dimana UI disini berfungsi untuk mengontrol dan memonitoring setiap *device* tersebut.

3.3 Analisis dan Penentuan Kebutuhan

Pada sub bab ini dijelaskan mengenai kebutuhan yang diperlukan dalam membangun *IoT Centralized Control and Monitoring System*. Dalam perangkat yang dibutuhkan dalam pembuatan sistem ini adalah perangkat keras (*hardware*) dan perangkat lunak (*software*). Berikut penjelasan dalam kebutuhan pembuatan sistem.

3.3.1 Analisis Kebutuhan Perangkat Keras (*Hardware*)

Dari pemecahan masalah yang dijelaskan pada subbab 3.2 dibutuhkan beberapa perangkat keras yang digunakan untuk pembangunan prototype alat pada sistem yang dibangun. Perangkat keras ini digunakan hanya sebagai mediator, dimana dari segi harga tidak terlalu mahal dan barangnya mudah di dapat, contohnya Wemos D1 yang digunakan sebagai mikrokontroler dari segi harga lebih murah dibanding mikrokontroler yang lain seperti *Raspberry pi* namun Wemos D1 mampu mengeksekusi perintah sederhana dan menggunakan modul *wifi*[10]. Sesuai dengan studi kasus yang sudah dibahas sebelumnya, untuk komponen studi kasus pemantauan tanaman menggunakan beberapa sensor dibutuhkan sensor ultrasonik, sensor DHT11, sensor LDR, dan sensor Soil Moisture. Untuk studi kasus pengontrolan lampu dibutuhkan LED, pada studi kasus ini tidak dibuhkan sensor dikarenakan hanya bekerja untuk menghidupkan atau mematikan lampu (*on/off*) yang dilakukan secara manual.

Berikut tabel kebutuhan mencakup *hardware*:

Tabel 1 Spesifikasi Kebutuhan Perangkat Keras

Perangkat Keras	Spesifikasi	Keterangan
Wemos D1	<ul style="list-style-type: none">- A 32 bit RISC CPU running at 80MHz- 64Kb of instruction RAM dan 96Kb of data RAM- 4MB flash memory- Wi-Fi- 16 GPIO pins- 12C, SPI- 12S	Perangkat yang berfungsi sebagai pusat kendali sistem, merespon perintah dari <i>server</i> .

Perangkat Keras	Spesifikasi	Keterangan
	- 1 ADC	
NodeMCU ESP8266	80 MHz Clock speed 4 MB / 64 KB Flash Memory/SRAM Wi-Fi 4.5V-10V Input Voltage	Perangkat yang berfungsi sebagai pusat kontrol aktuator
Sensor Ultrasonik	- HC-SR04 - Jangkauan : 2 cm – 400 cm - Resolusi : 0,3 cm - Jangkauan sudut : <15 derajat	Perangkat yang berfungsi untuk mendeteksi benda atau objek yang berada di hadapan sensor sehingga memancarkan sebuah gelombang suara ke arah depan.
LDR	- Resistansi : 10kOhm-100kOhm - Puncak spectral: 540nm (ukuran gelombang cahaya) - Waktu respon : 20ms-30ms - Suhu operasi : -30 ⁰ C sampai 70 ⁰ C	Perangkat yang berfungsi untuk mendeteksi keberadaan atau tingkat cahaya.
Relay	- Tegangan input 5 VDC , 12 VDC atau 48 VDC - Arus : 10 A - Coomon : 220 vac	Perangkat yang akan digunakan untuk mengalirkan arus listrik.
Soil Moisture	- YL – 69 - Tegangan input : 3.3V atau 5V - Tegangan output : 0 s\d 4.2V - Arus : 35 mA - Value range ADC sebesar 1024 mulai dari 0 s\d 1023 bit.	Perangkat yang berfungsi untuk membaca resistansi dari tanaman untuk mendapatkan nilai tingkat kelembaban tanah.
DHT11	- Tegangan input : 3,5 – 5 VDC - Sistem Komunikasi : Serial (Single – Wire two way) - Range suhu : 0 ⁰ C – 50 ⁰ C - Range kelembapan : 20% - 90% RH - Akurasi : $\pm 2^{\circ}\text{C}$ (temperature) $\pm 5\%$ RH (humadity)	Perangkat yang berfungsi untuk membaca suhu dari lingkungan.
Laptop	- Intel Core i5-7200U Processor (3M Cache, up to 10.3 GHz) - 8GB DDR4 RAM	Perangkat untuk menampilkan web aplikasi.

3.3.2 Analisis Kebutuhan Perangkat Lunak (*Software*)

Dari pemecahan masalah dibutuhkan juga beberapa perangkat lunak yang dibutuhkan dalam pengelolaan dan penyimpanan data dalam pembangunan *web* aplikasi. *Web* aplikasi yang dibangun menggunakan Bahasa pemrograman PHP, dimana developer membangun *web* aplikasi dikarenakan pengguna membutuhkan *Interface* dalam mengelola sistem dan pengguna dapat memperoleh data dari fitur *web* tersebut. Untuk pengelolaan database developer menggunakan MongoDB supaya tidak terbatas dengan banyaknya *value* sehingga datanya dinamis, dan MongoDB merupakan sistem yang terpisah sehingga tidak ada relasi antar tabel. Berikut tabel kebutuhan *software* :

Tabel 2 Spesifikasi Kebutuhan Perangkat Lunak

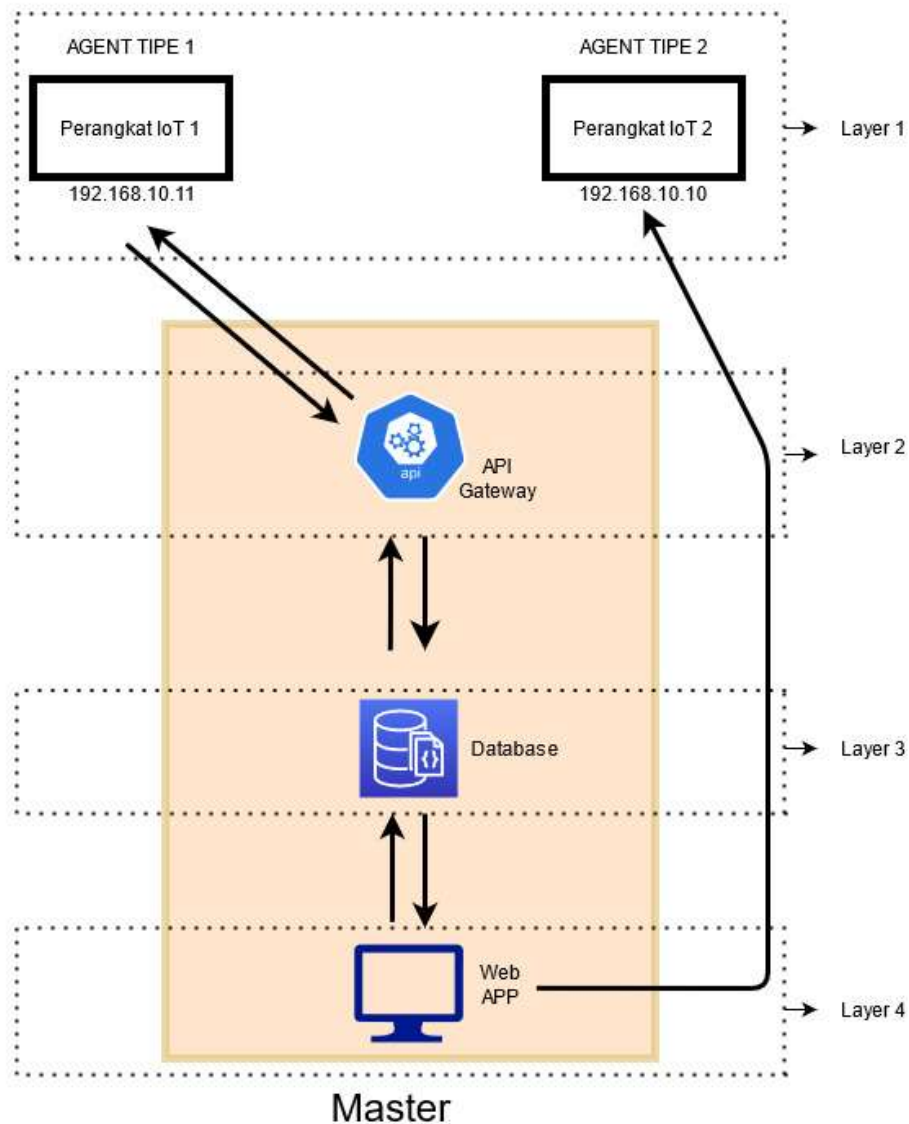
Nama Perangkat	Keterangan
Windows	Sistem Operasi yang digunakan dalam pembangunan aplikasi web, pengelola database dan penggunaan sistem
PHP, C++	Bahasa Pemrograman yang digunakan
MongoDB	Sistem pembuatan dan pengelolaan database
Software Arduino	OS untuk memprogram kode konfigurasi untuk wemos, node MCU dan code konfigurasi sensor dan aktuator.

3.4 Perancangan dan Arsitektur Sistem

Pada sub bab ini menjelaskan mengenai perancangan pada implementasi untuk pengerjaan tugas akhir ini baik dari sisi *hardware* maupun *software*. Perancangan sistem dalam pengujian ini yaitu pengujian HTTP *Request* dan HTTP *Response* antara *master* dan *client*, pengujian pengiriman antara *client* dan *master* begitu juga sebaliknya, pengujian terhadap *create*, *read*, *update* dan *delete* terhadap database. Dengan melakukan rancangan ini, maka diharapkan pengerjaan Tugas Akhir sesuai dengan *hardware* dan *software* yang dibutuhkan.

3.4.1 Gambaran Umum Sistem

Pada sub bab ini menjelaskan bagaimana gambaran umum alur kerja dari *IoT Centralized Control and Monitoring System* yang akan dibangun. Sistem yang dibangun ini memiliki 2 bagian utama yaitu *agent* dan *master*. Untuk gambaran umum dari sistem yang dibangun dapat dilihat pada Gambar 3 berikut.



Gambar 3 Gambaran Umum Sistem

Untuk penjelasan dari gambaran umum sistem kita dapat melihat apa-apa saja hal yang diterapkan dalam pembangunan proyek ini melalui pembagian layer-layer seperti yang tertera dibawah ini :

1. Layer 1

Pada layer pertama pada sistem yang dibangun ini merupakan *agent* yang merupakan perangkat iot atau *client-client* yang dalam satu *client* terdiri dari mikrokontroler, sensor dan ada juga aktuatornya. Dapat dilihat dari **Gambar 3** bagian Layar *Agent* merupakan bagian utama yang pertama menjelaskan tentang perangkat IoT yang akan dikontrol atau dimonitor oleh *master*. Dalam pembangunan sistem

ini terdapat terdapat dua jenis *client*, dimana jenis *client* yang pertama adalah *client* yang akan dikontrol secara otomatis berdasarkan nilai sensor yang diterima oleh master. Nilai dari sensor akan dikirimkan secara berkala ke aplikasi *master*. Jenis *client* yang kedua adalah *client* yang akan dikontrol secara manual oleh *administrator*. *Client* dan *master* terhubung oleh koneksi *wireless* dengan protokol HTTP.

Pada *Client* tipe kedua ini, aplikasi *master* melalui aplikasi *web* akan mengirimkan HTTP *request* kepada *client* melalui port 80. *Client* akan selalu melakukan *listen* melalui port 80. HTTP *request* ini berupa URL yang disertai data pin dan perintahnya. Berikut adalah contoh dari URL yang berisikan nomor pin dan perintah :

`http://192.10.10.2/2/on/3/off`

URL diatas terdiri dari IP *address agent* yang akan dikontrol (<http://192.10.10.2>), dan nomor pin serta perintah (2/on/3/off). Pada *agent* akan terdapat *code* yang akan membaca pin dan perintah sehingga *agent* dapat melakukan kontrol terhadap aktuaternya. Dengan perintah tersebut, *agent* akan menghidupkan pin nomor 2 dan mematikan pin nomor 3.

2. Layer 2

Pada layer dua pada sistem yang dibangun ini merupakan Server yang memiliki *Gateway* dan Database di dalamnya , dimana *gateway* disini yang menjadi gerbang komunikasi antara database dan *device* nya. Pada **Gambar 7** bagian API *Gateway* dijelaskan bahwa API *Gateway* berfungsi sebagai satu-satunya akses dari setiap *agent* untuk melakukan *create* data baru di dalam basis data, dan juga melakukan pengontrolan otomatis terhadap *agent*. API dibutuhkan untuk mentranslasikan Bahasa pemograman yang ada di mikrokontroler (C), *web* (PHP) dan query database. Saat menerima *file* JSON dari *agent*, API *Gateway* akan mentranslasikan file JSON tersebut menjadi *array* yang bisa diproses oleh Bahasa pemograman PHP. Sebelum memasukkan ke dalam database, API *Gateway* terlebih dahulu melakukan pengecekan apakah jumlah data yang diterima sesuai dengan jumlah sensor yang terdapat dalam basis data, jika hal ini tidak sesuai, maka proses akan berhenti. Jika jumlah data dan sensor sama, maka API *Gateway* akan melakukan pengecekan terhadap status sensor. Terdapat 2 jenis status sensor. Status pertama yaitu active,

yang berarti data yang diterima dari sensor tidak akan dilakukan perubahan. Status kedua yaitu *deactive*, yang berarti pengguna tidak menginginkan data dari sensor tersebut, oleh karena itu setiap data yang masuk akan diganti menjadi nol (0). Setelah pengecekan selesai, maka *API Gateway* akan menambahkan data tanggal dan waktu ke dalam *array* kemudian data akan disimpan di dalam database.

Untuk melakukan pengontrolan otomatis, *API Gateway* akan mengambil informasi nilai acuan sensor di basis data, kemudian melakukan perbandingan terhadap data yang diterima dari *client* dengan data acuan yang terdapat di basis data. Setelah melakukan perbandingan data, maka *API Gateway* kemudian akan membuat URL untuk melakukan kontroling terhadap *agent*. URL ini kemudian akan di *curl* menggunakan Bahasa pemrograman PHP untuk mengontrol *agent*.

3. Layer 3

Pada layer tiga ini pada sistem yang dibangun merupakan database store sebagai tempat penyimpanan data yang diterima dari *agent*.

4. Layer 4

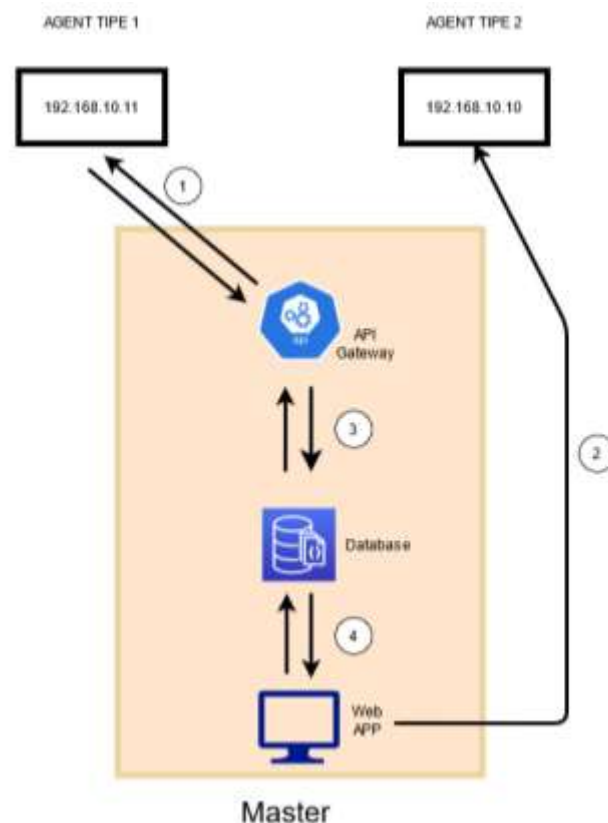
Pada layer empat ini pada sistem yang dibangun merupakan *User Interface* yang digunakan sebagai antar muka pengguna untuk memantau , mengontrol dan memonitoring setiap sensor dan menampilkan data yang diterima dari sensor.

Dapat dilihat dari **Gambar 3** bagian utama yang kedua adalah *Master*, dimana *master* adalah aplikasi *master* yang terdiri dari 3 bagian utama yaitu, *API Gateway*, database, dan aplikasi *web* yang terdapat pada layer 2,3 dan 4. *API Gateway* akan menerima data dari setiap *agent*, kemudian *API Gateway* akan melakukan modifikasi terhadap datanya, menyimpan data ke dalam basis data, dan melakukan kontrol terhadap *agent*. Database akan menyimpan semua data, termasuk data setiap sensor, dan data informasi perangkat yang dimasukkan oleh pengguna. Kemudian Aplikasi *web* akan menampilkan data, sebagai antarmuka untuk pengguna melakukan CRUD (*create, read, update, delete*) terhadap data di database, melakukan kontroling secara langsung terhadap setiap *agent*.

Untuk *flow* aliran proses pada sistem ini, pada *client* tipe pertama, data yang dikirimkan dari *client* berupa data JSON yang dikirimkan melalui *body HTTP request* menuju port 80 dari *API gateway*. Dalam pengaplikasian Rest API, terdapat 2 jenis data yang kerap digunakan, yaitu JSON dan XML[19]. Dibandingkan dengan XML, JSON lebih baik

digunakan untuk transmisi data yang sederhana[20,21], waktu respons JSON lebih cepat 30-40% daripada XML[21]. Hal ini membuat pengguna JSON lebih cocok daripada penggunaan XML. Data yang dikirimkan oleh *client* yang berformat JSON tersebut kemudian akan diproses oleh *API Gateway* pada aplikasi *master*.

Dari gambaran umum yang sudah dijelaskan sebelumnya kita dapat melihat komunikasi data secara umum dari sistem yang dibangun.



Gambar 4 Komunikasi Data Secara Umum

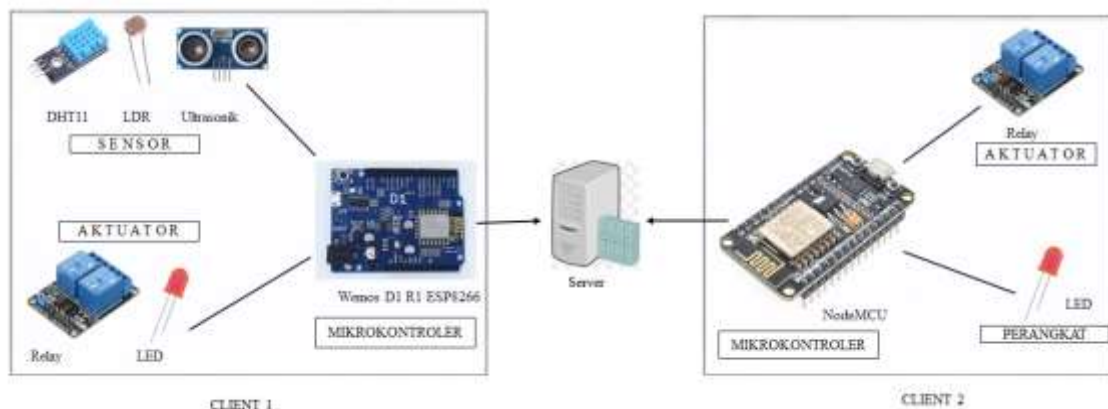
Dari keterangan penomoran pada **Gambar 4** diatas dapat kita lihat bahwa komunikasi yang terjadi di antara layer adalah sebagai berikut :

1. *API Gateway* melakukan komunikasi dua arah dengan *agent*, yaitu pada saat memberikan perintah dan menerima data sensor.
2. *Web* aplikasi melakukan komunikasi satu arah menuju *agent* saat melakukan kontroling secara langsung.
3. *API Gateway* melakukan komunikasi dua arah dengan basis data saat mengupdate data dan mengambil data.

4. *Web* aplikasi melakukan komunikasi dua arah dengan basis data, yaitu saat melakukan pengeditan data dan *me-retrieve* data saat akan menampilkan data sensor.

3.4.2 Perancangan Arsitektur *Hardware*

Desain arsitektur *hardware* digunakan untuk gambaran developer dalam melakukan implementasi tugas akhir. Sistem yang akan dibangun menggunakan beberapa perangkat keras yang saling terhubung. Desain perangkat keras *IoT Centralized Control and Monitoring System* dapat dilihat pada **Gambar 5** sebagai berikut.



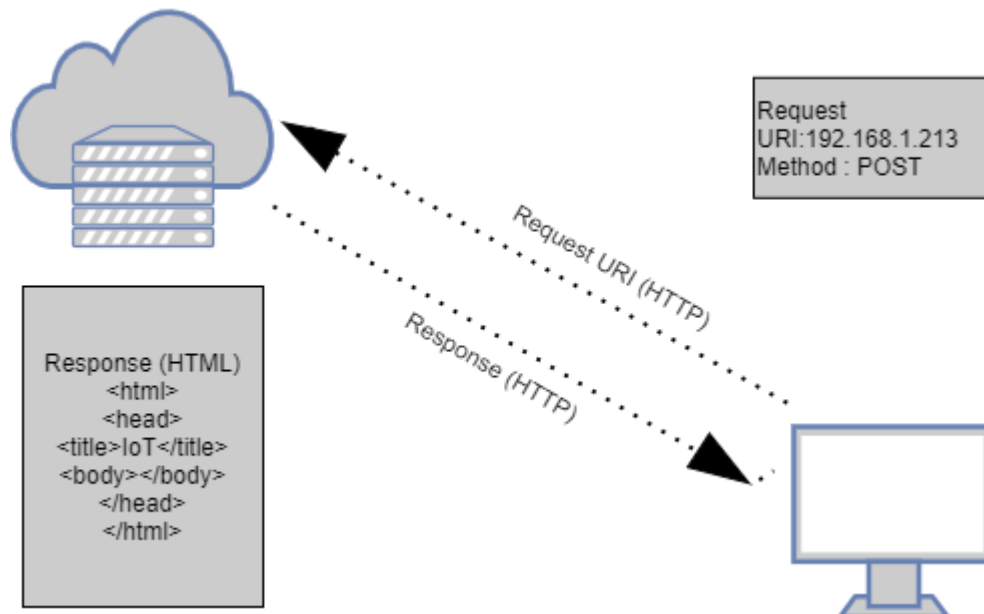
Gambar 5 Desain Arsitektur *Hardware*

Pada gambar diatas, dapat dilihat bahwa sistem memiliki dua bagian utama, yaitu *server* sebagai pusat data, dan *client* sebagai pengumpul data. *Client* adalah sistem IoT yang bertugas untuk mengumpulkan data, baik melalui sensor ataupun status dari setiap aktuator. *Client* merupakan sistem tertanam tradisional yang memiliki sensor, aktuator, dan aktuator. Namun, pada hal ini setiap *device* harus memiliki kemampuan untuk terhubung kedalam jaringan *wifi*. Maka dari itu, developer akan menggunakan Wemos D1 R1 sebagai *microcontroller*. Wemos D1 R1 memiliki kemampuan untuk terhubung kedalam jaringan *wifi* karena didalamnya sudah terdapat modul ESP8266. ESP8266 merupakan sebuah modul dimana modul tersebut memungkinkan sebuah *microcontroller* untuk terhubung kedalam sebuah jaringan *wifi*. *Server* adalah pusat dari sistem yang kami bangun. *Server* akan menerima setiap data yang dikirim oleh setiap *client* dan melakukan pengolahan terhadap data tersebut (*create, read, update, delete*) pada *database*. Setelah melakukan pengolahan terhadap data, *server* juga dapat mengirimkan perintah terhadap setiap *device*,

hal ini bergantung dari data yang diberikan dan masukan dari *user* terhadap *client* yang bersangkutan.

3.4.3 Komunikasi Data

Pada sub bab ini penulis akan membahas bagaimana sistem kami dapat berkomunikasi. Gambar dibawah ini adalah diagram dari komunikasi data kami.



Gambar 6 Komunikasi Data

Dari **Gambar 6** dapat dilihat bahwa komunikasi data dalam sistem kami adalah sistem komunikasi dua arah, yang memungkinkan *client* dapat mengirimkan data kepada *server* dan *server* dapat mengirimkan perintah kepada *client*.

3.4.3.1 Komunikasi *Client* dan *Server*

Pada sub bab ini penulis akan membahas tentang cara komunikasi antara *client* dengan *server*. *Client* akan mengirimkan data secara simultan ke *server*. Data yang dikirimkan berupa file json melalui *body HTTP Request*. *Server* kemudian mengolah file *JSON* yang dikirimkan oleh agent untuk selanjutnya disimpan di dalam basis data.

3.4.3.2 Komunikasi *Server* ke *Client*

Pada sub bab ini penulis akan membahas tentang cara komunikasi antara *client* dengan *server*. *Server* akan mengirimkan *HTTP Request* kepada *client* yang berisikan perintah berupa nomor pin yang akan di *set on* atau *off*. Saat *server* ingin mengirimkan *http request* kepada sebuah *device*, maka dapat digunakan perintah *curl*. *Curl* adalah sebuah program

dan library untuk mengirim dan mengambil data melalui URL. Client akan langsung mengeksekusi perintah yang dikirimkan oleh server, berupa mematikan atau menghidupkan perangkat.

3.5 Fungsi dan Flowchart

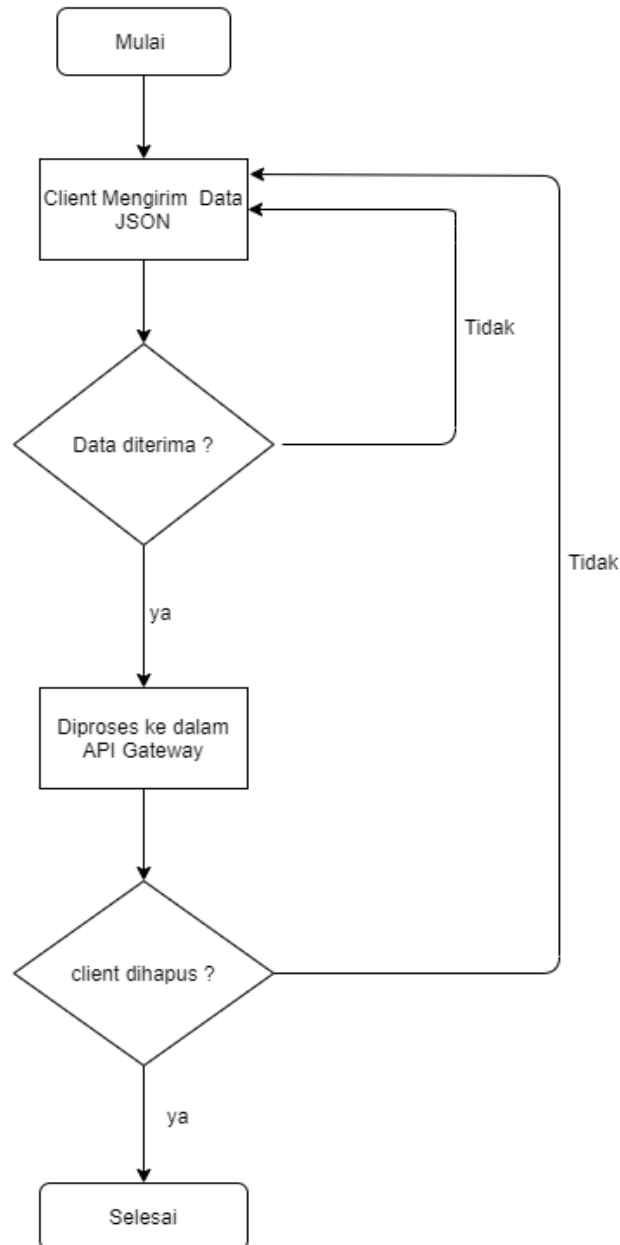
Sub bab ini menjelaskan tentang deskripsi dari setiap fungsi yang ada dalam tugas akhir ini baik dalam *hardware* maupun dalam *software*.

3.5.1 Fungsi Kirim Data dari HTTP *Body* ke API

Fungsi kirim data dari HTTP *body* ke API terjadi antara *client* dan API Gateway. Proses yang terjadi di dalam *client* yaitu *client* akan mengambil data melalui sensor terhadap lingkungan. Data yang diambil berbentuk JSON dan dikirimkan melalui *body* HTTP *request* melalui port 80 ke *API Gateway*, dimana data akan dikirimkan setiap 1 menit sekali.

3.5.2 Flowchart Fungsi Kirim Data dari HTTP *Body* ke API

Alur dalam kirim data dari HTTP *body* ke API dapat direpresentasikan menggunakan *flowchart* pada **Gambar 7** dibawah ini.



Gambar 7 Flowchart Fungsi Kirim Data dari HTTP Body ke API

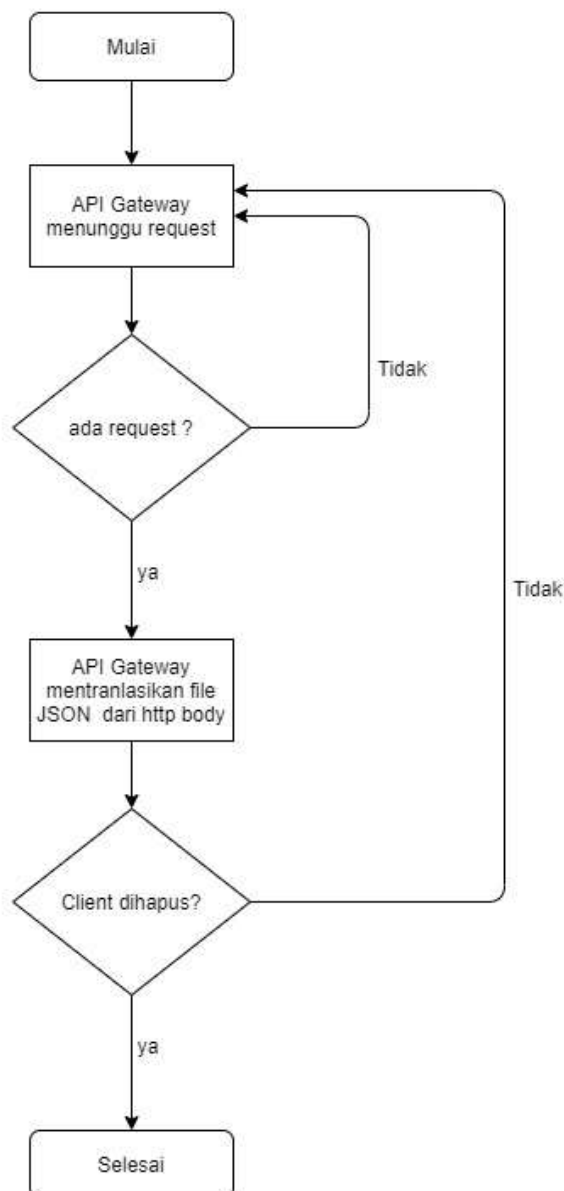
Flowchart ini menggambarkan bagaimana alur dalam pengiriman data dari *client* melalui *body* HTTP. Pada saat *client* mengirimkan data JSON dari *client* melalui *body* HTTP *request* melalui port 80 API Gateway. Data diterima oleh API Gateway, maka data tersebut akan diproses. Namun jika pengiriman data gagal, maka *agent* akan tetap mengirimkan data menuju API Gateway. Data akan diproses di dalam API Gateway dan ketika data tersebut sudah selesai diproses, maka *client* terus mengulang proses dari awal. Proses ini akan berhenti jika *client* sudah tidak ada lagi.

3.5.3 Fungsi Tranlasi JSON ke dalam query mongoDB

Fungsi tranlasi JSON ke dalam query mongoDB berlangsung di dalam API Gateway. Dimana data yang diterima dari *client* melalui HTTP *body* berbentuk JSON akan ditranslasi ke dalam query mongoDB. Proses pentranlasian akan menghasilkan bentuk array yang bisa diproses oleh bahasa php.

3.5.4 Flowchart Fungsi Translasi JSON ke dalam query mongoDB

Alur dalam pentranlasian JSON ke dalam query mongoDB dapat direpresentasikan menggunakan *flowchart* pada **Gambar 8** dibawah ini.



Gambar 8 Flowchart Fungsi Translasi JSON ke dalam query mongoDB

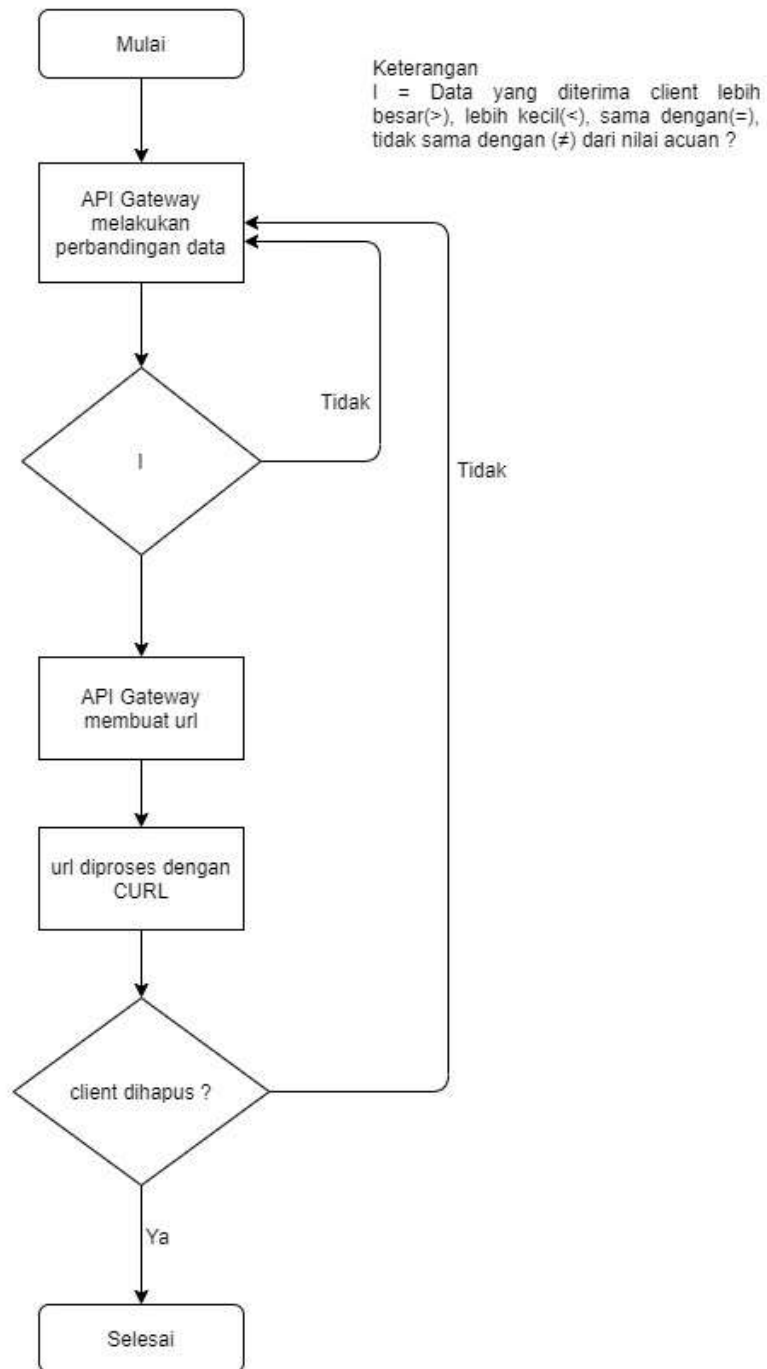
API Gateway akan menunggu *request* yang dilakukan oleh *client*, ketika tidak ada *request* maka proses akan terus berulang hingga API Gateway menerima *request*. Ketika API Gateway menerima *request* maka data JSON yang diterima akan ditranslasikan oleh API Gateway menjadi bentuk array yang bisa diproses oleh bahasa pemrograman php. Proses tersebut akan terus berlanjut selama *client* tidak dihapus.

3.5.5 Fungsi Membuat URL untuk Mengontrol Agent

Fungsi membuat url untuk mengontrol agent akan berlangsung di API Gateway. Dalam proses ini API Gateway akan membandingkan dua nilai yaitu nilai yang sudah ditentukan untuk menjadi acuan basis data dan nilai yang diterima dari *client*. Berdasarkan hasil perbandingan maka akan membuat url yang menjadi kontrol untuk menjalankan aktuator. Dalam hal ini agent yang akan dikontrol adalah pemantauan tanaman/sayuran , nilai acuan yang dimaksud adalah nilai yang sudah ditentukan yaitu lebih besar (>), lebih kecil(<), sama dengan(=), tidak sama dengan(\neq) dan nilai yang diterima dari *client* adalah data sensor

3.5.6 Flowchart Fungsi Membuat URL untuk Mengontrol Agent

Alur dalam membuat url untuk mengontrol agent dapat direpresentasikan menggunakan *flowchart* pada **Gambar 9** dibawah ini.



Gambar 9 Flowchart Fungsi Membuat URL untuk Mengontrol Agent

API Gateway akan melakukan perbandingan terhadap data , dimana API Gateway mengambil informasi nilai yang menjadi acuan basis data , dan mengambil informasi nilai yang diterima dari *client*. Nilai inilah yang akan menjadi perbandingan lebih besar(>), lebih kecil(<), sama dengan (=), dan tidak sama dengan (≠). Setelah melakukan perbandingan maka API Gateway akan membuat url , url ini kemudian akan di curl menggunakan bahasa

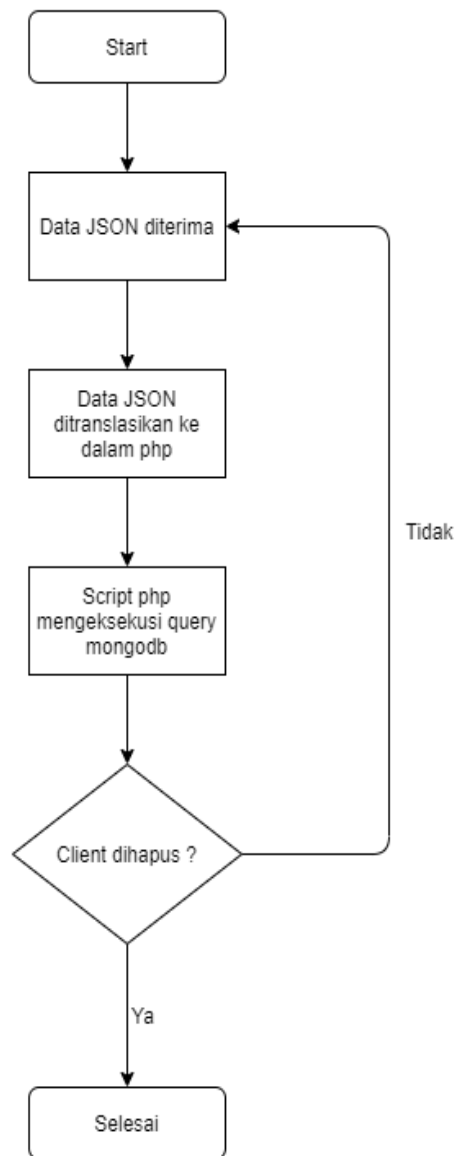
pemrograman PHP yang berguna sebagai pengontrolan . Proses akan terus berlanjut selama client tidak dihapus.

3.5.7 Fungsi *Update* Data ke MongoDB

Fungsi *update* data ke mongoDB ini akan berlangsung di dalam API Gateway. Sebelum data di update ke mongoDB data JSON yang diterima dari client terlebih dahulu ditranslasikan agar mongoDB dapat menjalankan script yang akan melakukan update ke mongoDB.

3.5.8 *Flowchart* Fungsi *Update* Data ke MongoDB

Alur dalam update data ke dalam mongoDB dapat direpresentasikan menggunakan *flowchart* pada **Gambar 10** dibawah ini.



Gambar 10 Flowchart Fungsi Update Data Ke MongoDB

Ketika API Gateway menerima data JSON dari *client*, data tersebut akan ditranslasikan ke dalam bentuk array php. Array ini kemudian akan dieksekusi oleh *script* dalam bahasa pemrograman php untuk melakukan update kedalam MongoDB. Proses ini akan terus berulang jika ada perubahan ataupun masih menerima data dan terus berlanjut selama client tidak dihapus.

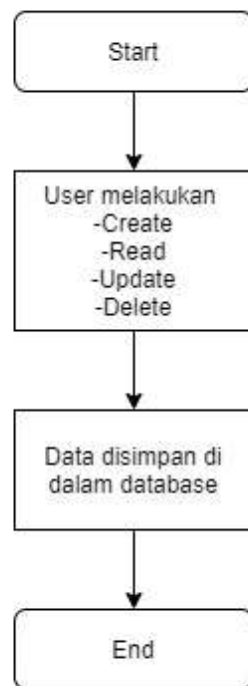
3.5.9 Fungsi CRUD Data ke MongoDB

Fungsi CRUD (Create, Read, Update, Delete), fungsi ini akan dijalankan melalui web aplikasi. Hasil CRUD di web aplikasi akan ditambahkan ke dalam database. Fungsi create adalah akan membuat *device* baru melalui web, fungsi update akan melakukan

pembaharuan data, dan fungsi delete adalah menghapus sebuah *device* , maupun sensor yang ada didalam sebuah *device* IoT.

3.5.10 Flowchart Fungsi CRUD data ke MongoDB

Alur dalam CRUD ke dalam mongoDB dapat direpresentasikan menggunakan *flowchart* pada **Gambar 11** dibawah ini.



Gambar 11 Flowchart Fungsi CRUD Data ke dalam MongoDB

User akan melakukan perintah CRUD (menambah, menghapus, melakukan perubahan terhadap *client*) melalui aplikasi *web*, kemudian web akan memproses perintah dari user dengan melakukan modifikasi di dalam basis data. Proses itu akan berlanjut ketika user ingin melakukan CRUD.

3.5.10.1 Flowchart Fungsi Create Oleh Administrator

Berdasarkan flowchart CRUD maka akan dipecah lagi menjadi beberapa bagian. *Flowchart create* akan ditampilkan pada **Gambar 12** dibawah ini.



Gambar 12 Flowchart Fungsi Create

Flowchart ini akan menjelaskan alur bagaimana user melakukan *create*. User akan melakukan *create* melalui web, kemudian melakukan input data yaitu nama agent, ip address, jumlah pin dan aktuator, tipe dan nama sensor. Setelah user selesai melakukan input maka data yang sudah diinput akan di tambahkan ke dalam database.

3.5.10.2 Flowchart Fungsi Update Oleh Administrator

Flowchart update akan ditampilkan pada Gambar 13 dibawah ini.

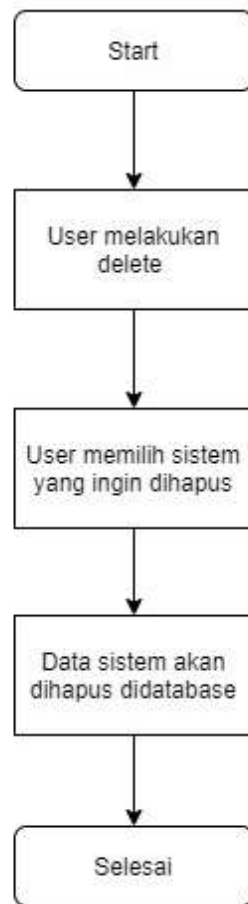


Gambar 13 Flowchart Fungsi Update

Flowchart ini akan menjelaskan alur bagaimana user melakukan *update*. User akan melakukan *update* melalui web, kemudian melakukan input data yaitu nama agent, ip address, jumlah pin dan aktuator, tipe dan nama sensor yang ingin diupdate. Setelah user selesai melakukan input maka data yang sudah diinput akan di perbaharui di dalam database.

3.5.10.5 Flowchart Fungsi Delete Oleh Administrator

Flowchart delete akan ditampilkan pada Gambar 14 dibawah ini.



Gambar 14 Flowchart Fungsi Delete

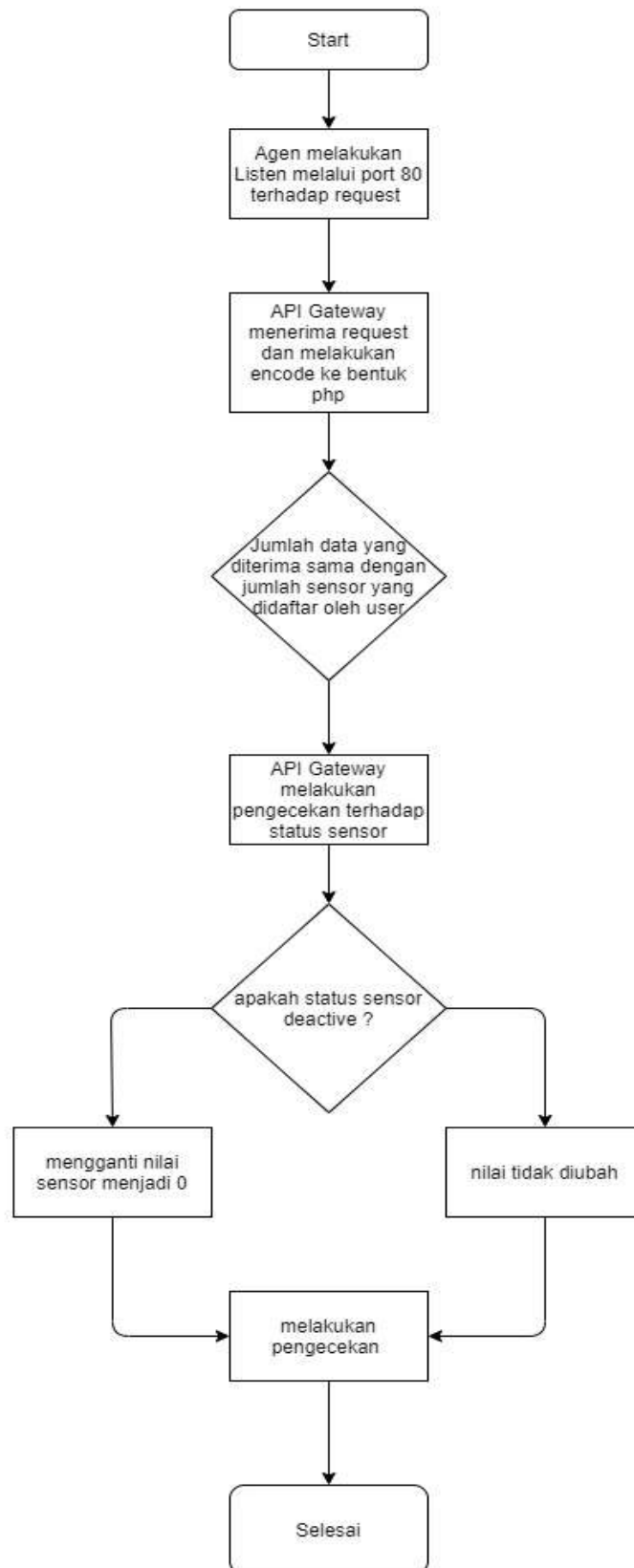
Flowchart ini akan menjelaskan alur bagaimana user melakukan *delete*. User akan melakukan *delete* melalui web. User akan memilih *device* yang ingin dihapus, setelah user berhasil melakukan delete , maka data *device* akan terhapus di dalam database.

3.5.11 Fungsi Terima Request

Fungsi ini akan berlangsung di API Gateway. Fungsi ini akan melakukan listen terhadap HTTP request oleh *agent* melalui port 80 agar API Gateway dapat menerima request dan melakukan perbandingan data yang diterima dengan data yang menjadi acuan.

3.5.12 Flowchart Fungsi Terima Request

Alur dalam Terima Request dapat direpresentasikan menggunakan *flowchart* pada Gambar 15 dibawah ini.



Gambar 15 Flowchart Fungsi Terima Request

Flowchart Terima request oleh *Agent* akan melakukan listen terhadap *request* yang datang melalui port 80. Kemudian API Gateway menerima sebuah *request*, API Gateway akan melakukan *encode body* HTTP kedalam bentuk variabel PHP. Kemudian API Gateway akan melakukan pengecekan apakah jumlah data yang diterima sama banyaknya dengan jumlah sensor yang didaftarkan oleh pengguna di dalam basis data. Jika jumlah data tidak sama banyak dengan jumlah sensor yang didaftarkan, maka proses akan berhenti sampai disini dan API Gateway akan melakukan *listen* kembali.

Jika jumlah data yang diterima sama dengan jumlah sensor yang ada di dalam basis data, maka melakukan pengecekan terhadap status dari sensor. Jika sensor memiliki status *deactive*, maka API Gateway akan mengganti nilai sensor tersebut menjadi 0, namun jika sensor memiliki status *active*, maka data sensor tidak akan diubah sama sekali. Setelah melakukan pengecekan, API Gateway akan memasukkan waktu dan tanggal kedalam *array* data tersebut. Setelahnya, maka API Gateway akan memasukkan data tersebut ke dalam basis data.

Setelah berhasil memasukkan data, maka API Gateway akan melakukan perbandingan untuk membuat *url* terhadap data yang diterima dan data yang ada di dalam basis data, dengan acuan operator perbandingan yang dimasukkan oleh pengguna di dalam basis data

3.5.13 Fungsi Rata-rata data

Fungsi ini merupakan fungsi yang digunakan untuk melakukan rata rata terhadap data sensor yang terdapat pada *agent* pengolahan beberapa sensor. Data ini akan dikumpulkan berdasarkan menit yang telah di set , kemudian akan dilakukan penghitungan nilai data untuk mendapatkan hasil rata – rata. Hasil rata-rata ini akan ditampilkan dalam bentuk chart melalui web.

3.5.14 Flowchart Fungsi Rata-rata data

Alur dalam melakukan rata-rata dapat direpresentasikan menggunakan *flowchart* pada **Gambar 16** dibawah ini.



Gambar 16 Flowchart Fungsi Rata-Rata Data

Web terlebih dahulu mengambil data dari database, data ini berupa data dari setiap sensor yang tersimpan di basis data. Kemudian web akan menghitung rata rata dari data yang diterima dan menampilkannya di dalam web dalam bentuk chart . Proses pengratarataan ini akan berulang dan berlanjut berdasarkan waktu yang telah diatur.

3.6 Desain

Pada sub Bab ini menjelaskan tentang desain dari web yang akan dibangun dan juga desain database. Tampilan web akan terdiri web *user* saja yang dimana *user* disini berperan sebagai *administrator*.

3.6.1 Desain Web Sistem

3.6.1.1 Komponen Layar Awal Web

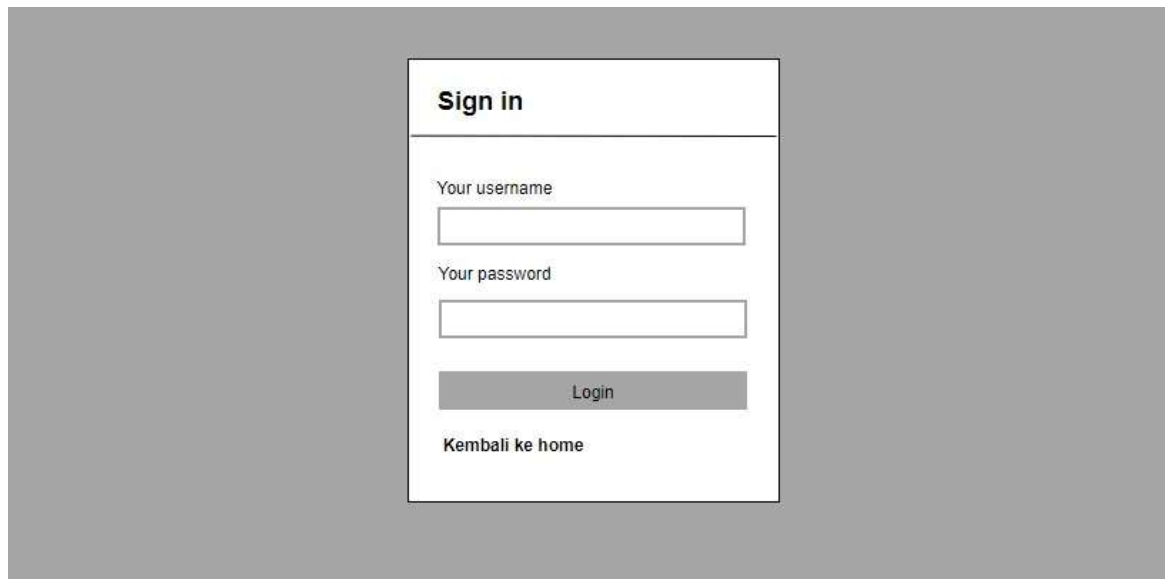
Berikut desain halaman web. Agar *user* dapat melakukan kegiatan dalam web sistem dibutuhkan autentikasi dengan *login* terlebih dahulu. Pada **Gambar 17** dapat dilihat tampilan awal dari halaman *web* aplikasi yang akan dibangun.



Gambar 17 Tampilan Halaman Awal Aplikasi Web

3.6.1.2 Komponen Layar Login

Halaman *login* berfungsi untuk menjaga keamanan dan privasi. Rancangan tampilan halaman *login* dapat dilihat dibawah ini. Untuk *user* yang melakukan *login* hanya *administrator*, sehingga *username* dan *password* sesuai dengan yang sudah didaftarkan. Pada **Gambar 18** merupakan desain dari layar login yang akan dibangun.



Gambar 18 Tampilan Halaman Login Aplikasi Web

3.6.1.3 Komponen Layar Utama (Dashboard)

Pada Gambar 19 menunjukkan rancangan untuk *dashboar* dari *web* yang akan dibangun. Ketika *user* berhasil *login* ke dalam web aplikasi sebagai *administrator*, akan langsung ditampilkan bagian *dashboard* yang menyajikan tampilan *device* *iot* yang sudah ada terdaftar sebelumnya, menu tambah agent, hapus agent dan edit agent. Berikut deskripsi menu dan fungsi utama dari *Web* :

- 1) Daftar *Agent*, *agent* yang dimaksudkan disini adalah *device* *IoT*-nya terdiri dari 2 jenis yaitu *device* yang *Automatic Control* dan *Manual Control*, sesuai dengan studi kasus yang dijelaskan di awal, pada bagian *Automatic Control* dimasukkan orkestrasi beberapa sensor yaitu pemantauan tanaman yang menggunakan beberapa sensor dan pada bagian *Manual Control* dimasukkan Kontroling untuk menghidupkan dan mematikan Lampu.
- 2) Tambah *Agent* : Dalam menu ini *user* dapat melakukan penambahan *agent*, dimana *agent* yang dimaksud disini adalah *device* *IoT* nya, *user* akan menambahkan *device* *IoT* dan memilih apakah *agent* yang ditambahkan tersebut dikontrol secara otomatis atau manual, kemudian menambahkan keterangan pin dari sensor yang digunakan pada *agent* tersebut untuk dilakukan konfigurasi.
- 3) Edit *Agent* : Pada bagian aksi di *button* Edit disini *user* dapat melakukan pengeditan deskripsi dari setiap *device* *IoT* yang sudah didaftarkan sebelumnya, misalnya *user*

ingin menghapus sensor tertentu ataupun menambahkan beberapa sensor lagi ataupun mengubah pin dari setiap sensor.

- 4) Hapus *Agent* : Pada bagian aksi di *button* hapus disini *user* dapat melakukan *remove* pada *device* IoT yang sudah didaftarkan sebelumnya, dimana ketika *user* memilih *button* hapus maka keseluruhan perangkat yang berada dalam *device* IoT yang ingin dihapus akan ikut terhapus juga.

Selamat datang admin

[+ Tambah](#) [Logout](#)

Data Pengguna - Manual Control

Show entries Search

No	ID	Nama Device	IP Address	Aksi
1				Edit Hapus
2				Edit Hapus

[Pervious](#) [Next](#)

Data Pengguna - Automatic Control

Show entries Search

No	ID	Nama Device	IP Address	Jumlah pin dan aktuator	Aksi
1					Edit Hapus
2					Edit Hapus

[Pervious](#) [Next](#)

[User Manual](#)

Gambar 19 Tampilan Halaman Dashboard Aplikasi Web

3.6.1.4 Komponen Layar Tambah Agent

Berikut desain tampilan halaman Tambah *device* dapat dilihat pada Gambar 20. Dalam halaman ini *user* akan menginput data dari *agent* yang ingin ditambahkan. *User* juga akan memilih apakah *agent* yang ditambahkan menggunakan *manual control* ataupun *automatic control*.

Kembali Logout

Tambah Device

Nama agent

Ip address

Jumlah Sensor

Type

+ Tambah

Gambar 20 Tampilan Halaman Tambah Device

Dalam halaman ini pengguna akan melakukan input nama *agent* yaitu nama dari *device* IoT yang ingin ditambahkan. Kemudian melakukan input *ip address*, dimana *ip address* ini didapatkan ketika *user* sudah menjalankan kode pada *software Arduino*. *Field* jumlah sensor merupakan jumlah sensor yang ada pada *device* sesuai dengan rangkaian. *Field type* merupakan pilihan tipe yang digunakan, dalam studi kasus pengontrolan lampu maka perlu memilih *type manual control* dan jika dalam studi kasus pemantauan tanaman menggunakan beberapa sensor perlu memilih *type automatic control*.

3.6.1.5 Komponen Layar Tambah Agent Tipe Automatic Control

Ketika *user* memilih tipe *automatic control* maka desain tampilannya terlihat seperti pada **Gambar 21** berikut.

Kembali
Logout

Tambah Data Agent

Nama

IP

Pin

Type Automatic Control

IP Controller

Nama sensor	Pin Sensor	State	Nilai Sensor	Satuan	Pin Aktuator

+ Tambah

Gambar 21 Tampilan Halaman Tambah Device Automatic Control

Dalam tampilan berikut maka akan ditampilkan nama, ip, pin, dan *type automatic control* yang sudah dibahas sebelumnya. Proses tambah data tipe *automatic control* ini *user* akan melakukan input *Ip controller*, *Ip controller* yang dimaksud didapatkan ketika kita sudah selesai melakukan *compiling* terhadap kode di *software Arduino*. Kemudian *user* melakukan input nama sensor yang digunakan, pin sensor sesuai dengan rangkaian, kemudian *state*, dimana *state* yang dimaksud adalah lebih besar (>)/lebih kecil(<)/sama dengan(=)/tidak sama dengan(≠) dan juga mengisi bagian nilai sensor yang merupakan inputan *user* yang akan menjadi acuan dari *state* untuk menjalankan aktuator.

3.6.1.6 Komponen Layar Tambah Agent Tipe Manual Control

Ketika *user* memilih tipe *manual control* maka desain tampilannya terlihat seperti pada **Gambar 22** berikut.

KembaliLogout

Tambah Data Agent

Nama
IP
Pin
Type Manual Control

Nama alat	Pin sensor

+ Tambah

Gambar 22 Tampilan Halaman Tambah Device Manual Control

Dalam tampilan halaman berikut maka akan ditampilkan nama, ip , pin, dan *type manual control* yang sudah dibahas sebelumnya. *User* kemudian menginput nama alat yang ada di dalam kontrol lampu, seperti lampu depan dan menginput pin sensor sesuai dengan rangkaian.

3.6.1.7 Komponen Layar Edit Agent Tipe Automatic Control

Tampilan desain halaman edit untuk *agent* dengan tipe *automatic control* dapat dilihat pada Gambar 23 berikut.

Kembali
Logout

Edit Device - Automatic Control

Nama

IP Address Sensor

IP Address Controller

NO	Nama sensor	Pin Sensor	State	Nilai sensor	Satuan	Pin Aktuator	Aksi
1							Hapus
2							Hapus

Tambah baris | Hapus baris

+ Tambah Data

Gambar 23 Tampilan Halaman Edit Device Automatic Control

User dapat melakukan pengeditan terhadap data yang telah ditambahkan sebelumnya, dengan mengubah nama *device*, *ip address*. Kemudian jika ada perubahan rangkaian *user* dapat melakukan pengeditan pada nama sensor dan pin sensor. *User* juga dapat melakukan pengurangan baris jika ada sensor yang tidak dibutuhkan lagi dan penambahan baris jika ada sensor baru yang akan digunakan dengan menyesuaikan pin pada rangkaian.

3.6.8 Komponen Layar Edit Agent Tipe Manual Control

Tampilan desain halaman edit untuk *agent* dengan tipe *manual control* dapat dilihat pada **Gambar 24** berikut.

Kembali
Logout

Edit Device - Manual Control

Nama

IP Address Sensor

NO	Nama sensor	Pin Sensor	Aksi
1			Hapus
2			Hapus

Tambah baris | Hapus baris

+ Tambah Data

Gambar 24 Tampilan Halaman Edit Device Manual Control

User dapat melakukan pengeditan terhadap data yang telah ditambahkan sebelumnya, dengan mengubah nama *device*, *ip address* da *ip controller*. Jika ada perubahan pada rangkaian *user* dapat melakukan perubahan pada nama sensor, pin sensor, *state*, nilai sensor, dan pin aktuator. *User* juga dapat melakukan pengurangan baris jika ada sensor yang tidak dibutuhkan lagi dan penambahan baris jika ada sensor baru yang akan digunakan dengan menyesuaikan pin pada rangkaian.

3.6.2 Desain Database

Database yang digunakan dalam pembangunan sistem ini adalah MongoDB karena MongoDB tidak memiliki relasi dan mempercepat proses CRUD (*create*, *read*, *update*, *delete*). MongoDB berisi *collection*, dimana dalam pembangunan sistem ini akan nada 2 *collection* di dalam basis data. *Collection* dianalogikan sebagai tabel di basis data yang memiliki relasi. *Collection* pertama akan menyimpan informasi umum *client*, seperti nama *client*, *ip address*, jumlah pin, nama sensor dan aktuator dan id dari *client* tersebut.

```

{
  "_id": {
    "$oid": "6108f65d982c0000720017db"
  },
  "id": "109:21:36082021",
  "agent_ip": "172.20.40.120",
  "controller_ip": "172.20.40.125",
  "agent": "pemantauan tanaman",
  "sensor_pin": ["7", "0", "6"],
  "sensor_value": ["8", "100", "20"],
  "actuator_pin": ["3", "4", "5"],
  "desc_sensor": ["sensor jarak", "sensor ldr", "suhu"],
  "state": [">", ">", ">"],
  "status": ["active", "active", "active"],
  "tipe": "2"
}

```

Untuk *collection* yang kedua akan menyimpan nilai sensor dari setiap *agent*.

```

{
  "_id": {
    "$oid": "60c5e3158e75000061002f72"
  },
  "id": "112:20:45062021",
  "sensor_value": [35, 195, 51],
  "tanggal": "2021-06-13",
  "waktu": "17:51:01"
}

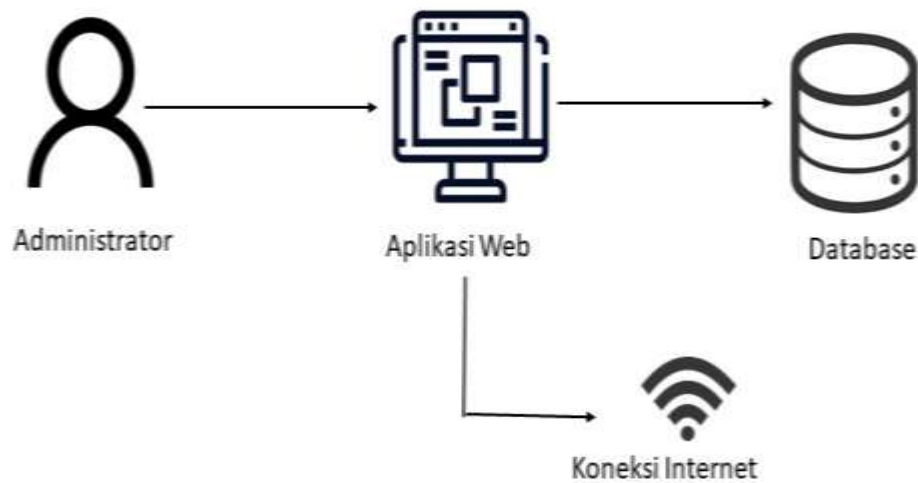
```

Setiap data dari setiap sensor akan di simpan di dalam *collection* kedua. Pengguna dapat melakukan monitoring dan melihat data setiap *agent* melalui aplikasi *web*. Aplikasi *web* mengambil nilai sensor dari *agent* tertentu untuk ditampilkan, namun sebelumnya data akan diolah untuk mengambil nilai rata-rata sensor setiap 5 menit, kemudian nilai rata-rata ini akan ditampilkan dalam bentuk grafik pada aplikasi *web*. Melalui aplikasi *web*, *user* juga dapat melakukan aksi CRUD untuk menambah *client* baru, menghapus *client*, dan melakukan edit terhadap *client*. Setiap perubahan akan langsung diaplikasikan di dalam basis data mongoDB oleh aplikasi *web*.

3.7 Skenario Pengujian

Pada bagian ini menjelaskan beberapa scenario pengujian dari *IoT Centralized Control and Monitoring System* dengan 2 studi kasus yaitu pengontrolan lampu dan pemantauan tanaman yang menggunakan beberapa sensor yang akan diimplementasikan pada tahap pengujian.

3.7.1 Skenario Pengujian Fungsi CRUD *Agent* Pada Aplikasi Web



Gambar 25 Skenario Pengujian Fungsi CRUD

Pada Gambar 25 di atas menjelaskan skenario fungsi CRUD pada *agent*, dengan tujuan menguji web aplikasi yang dibangun dapat melakukan CRUD pada setiap *agent*. Aktivitas dimulai dari *user* sebagai *administrator* memeriksa koneksi internet pada PC supaya dapat mengakses aplikasi *web*. Aplikasi *web* disini berfungsi sebagai antar muka pengguna dalam memantau dan mengontrol *device* IoT dan juga *web* aplikasi akan menampilkan data. *User* akan melakukan CRUD (*create, read, update and delete*) di dalam aplikasi *web* terhadap data di database dan melakukan kontroling secara langsung terhadap *device* IoT.

1. Skenario Pengujian Fungsi *Create Agent*

Pada saat melakukan *create agent* aktivitas dimulai dari *user* sebagai *administrator* login ke *website*. Kemudian setelah *user* berhasil masuk, *user* akan dialihkan ke halaman *dashboard*, pada halaman *dashboard* terdapat menu *user manual* yang berfungsi untuk memberikan informasi kepada *user* bagaimana cara untuk menambahkan *agent* baru. *User* harus menjalankan kode yang sudah disediakan terlebih dahulu pada *software Arduino ide* untuk mendapatkan Ip address dari *device* yang akan ditambahkan, *user* juga harus menjalankan kode program *hardware* dari *device* yang akan ditambahkan, dan memastikan untuk mengatur SSID dan *password* dari koneksi jaringan yang digunakan supaya proses *create agent* berjalan dengan baik. Setelah *user* mendapatkan ip address, maka akan dilanjut

ke halaman penambahan *agent* baru dengan memasukkan data dari *agent* yaitu nama *agent*, ip address yang sudah didapatkan sebelumnya, jumlah sensor dan tipe dari *agent* yang ditambahkan. Jika tipe dari *agent* yang ditambahkan adalah *manual control*, selanjutnya *user* akan dialihkan ke halaman deskripsi *manual control* yang membutuhkan data berupa nama alat dan pin sensor. Jika tipe dari *agent* yang ditambahkan adalah *automatic control*, selanjutnya *user* akan dialihkan ke halaman deskripsi *automatic control* yang membutuhkan data berupa *ip controller* yang didapatkan dari kode program *device* jika sudah berhasil dijalankan, nama sensor, pin sensor, state, nilai acuan dan pin aktuator. Pada saat menambahkan sensor tipe *automatic control*, dipastikan pengguna harus menginput data berurutan sesuai dengan urutan pada code program *hardware* nya. Setelah pengisian data selesai, *agent* yang baru ditambahkan akan muncul pada *dashboard web* yang menandakan fungsi *create* berhasil diimplementasikan.

2. Skenario Pengujian Fungsi Read Agent

Untuk skenario pengujian fungsi *read* data *agent* akan berlangsung ketika *user* sudah berhasil menambahkan *agent* baru. Maka *agent* yang baru ditambahkan akan muncul pada halaman *dashboard*, dimana *agent* yang ditambahkan tertera pada tabel yang sesuai dengan tipe masing-masing *agent*. Setiap *agent* yang sudah berhasil ditambahkan dapat diakses dan dikontrol oleh *user* yang menandakan fungsi *read* berhasil diimplementasikan.

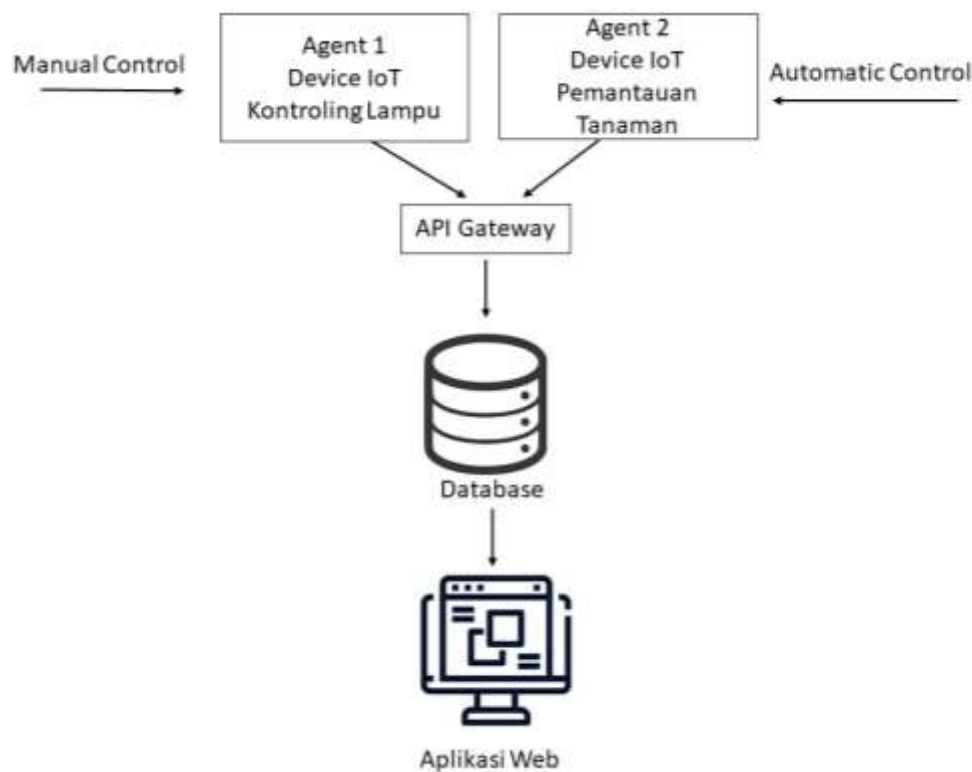
3. Skenario Pengujian Fungsi Update Agent

Untuk skenario pengujian fungsi *update agent* berlangsung ketika *user* melakukan perubahan terhadap data *agent*, misalnya *user* mengubah nama *agent*, mengubah *ip address*, mengubah nama alat/sensor, mengubah pin sensor/aktuator, menambahkan atau mengurangi baris dari data *agent* maka setelah melakukan perubahan tersebut, data yang terbaru akan ter-*update* sesuai dengan perubahan yang dilakukan oleh *user* pada *dashboard web* yang menandakan fungsi *update* berhasil diimplementasikan.

4. Skenario Pengujian Fungsi Delete Agent

Untuk skenario pengujian fungsi *delete agent* berlangsung ketika *user* menghapus baris dari deskripsi *agent* atau menghapus *agent* secara keseluruhan maka, *agent* ataupun baris yang sudah dihapus tidak akan tertampil pada *dashboard web* sistem yang menandakan fungsi *delete* berhasil diimplementasikan.

3.7.2 Skenario Pengujian Komunikasi *Agent* - *API Gateway* – *Database* – *UI*



Gambar 26 Skenario Pengujian Komunikasi *Agent*-*API Gateway*-*Database*-*UI*

Pada Gambar 26 di atas menjelaskan skenario jalannya sistem dan relasi dari masing-masing komponen, dengan tujuan menguji sistem yang dibangun berjalan dengan baik. Pada sistem terdiri dari 2 bagian utama yaitu *agent* yang merupakan *device* IoT dan *master* yang terdiri dari 3 bagian utama yaitu *API Gateway*, *Database* dan *Aplikasi web*. Dimulai dari *agent* yang terbagi dalam dua jenis yaitu yang dikontrol secara manual dan otomatis. Perlu diketahui untuk setiap *agent* tertentu sudah dipastikan bahwa *device* 1 untuk *automatic control* dan *device* 2 untuk *manual control* sehingga untuk *agent* 1 dan 2 tidak dapat dilakukan *switch* atau pertukaran posisi dikarenakan akan banyak perubahan data.

Untuk *agent* yang dikontrol secara otomatis akan mengirimkan nilai dari masing-masing sensor secara berkala ke aplikasi *master*, sementara untuk *agent* yang dikontrol secara manual melalui aplikasi *web* akan mengirimkan *HTTP request* kepada *client*. Pada bagian *API Gateway* akan menerima data dari setiap *agent* dan akan melakukan modifikasi data terhadap datanya dan kemudian data tersebut akan disimpan ke dalam *database*. Data yang sudah disimpan akan ditampilkan pada aplikasi *web* yang digunakan sebagai antar muka pengguna untuk melakukan *CRUD*.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

Pada bab ini menjelaskan mengenai implementasi terhadap sistem yang telah dibangun. Dalam proses implementasi terdapat beberapa hal yang dilakukan berupa implementasi perangkat yaitu melakukan konfigurasi , implementasi pada prototype dan implementasi pada aplikasi web. Dalam pengujian yang dilakukan berdasarkan tujuan dari sistem yang dijelaskan pada bab sebelumnya.

4.1 Implementasi

Pada sub bab ini menjelaskan implementasi yang dilakukan pada pembangunan sistem yaitu sistem yang melakukan kontroling dan monitoring beberapa *device* IoT yang menggunakan Wemos D1 ESP8266 sebagai mikrokontrolernya.

4.1.1 Implementasi Wemos D1 ESP8266

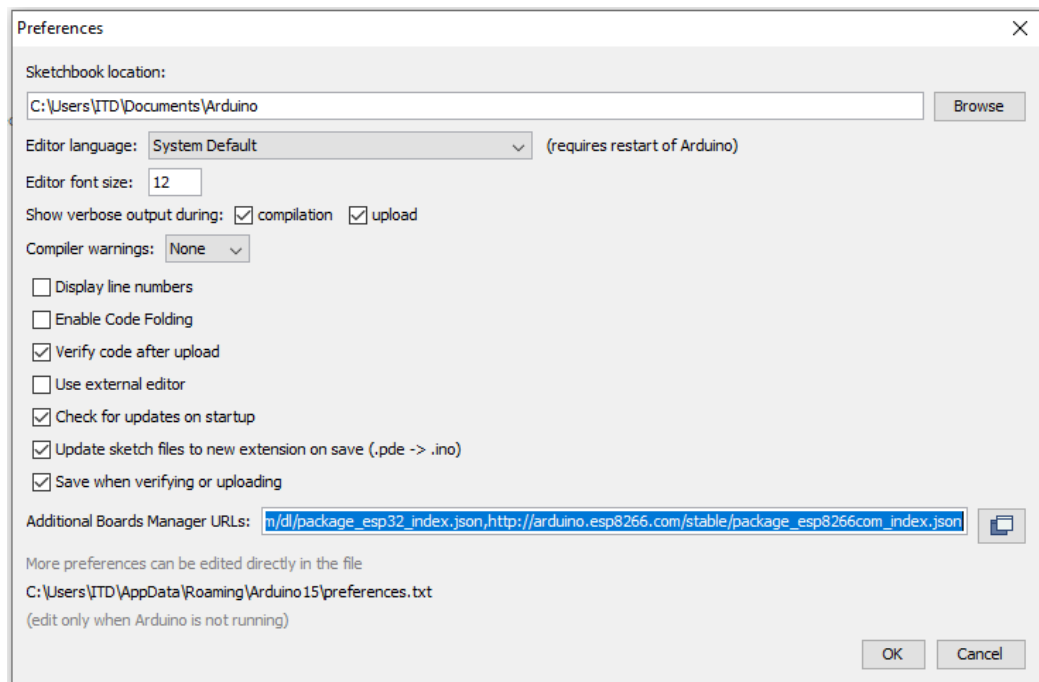
Pada subbab ini akan dijelaskan cara menginstal Wemos D1 R1 pada Arduino IDE supaya dapat diprogram. Untuk Instalasi Wemos D1 R1 dapat dilihat pada link berikut : <http://www.jogjarobotika.com/blog/tutorial-install-wemos-pada-arduino-ide-b135.html>

1. Download *software* Arduino IDE dari web resminya pada link berikut <https://www.arduino.cc/en/software> . Kemudian melakukan penginstalan Arduino IDE di PC anda.
2. Kemudian buka aplikasi Arduino yang sudah selesai di *install*, kemudian beralih ke menu file -> preference



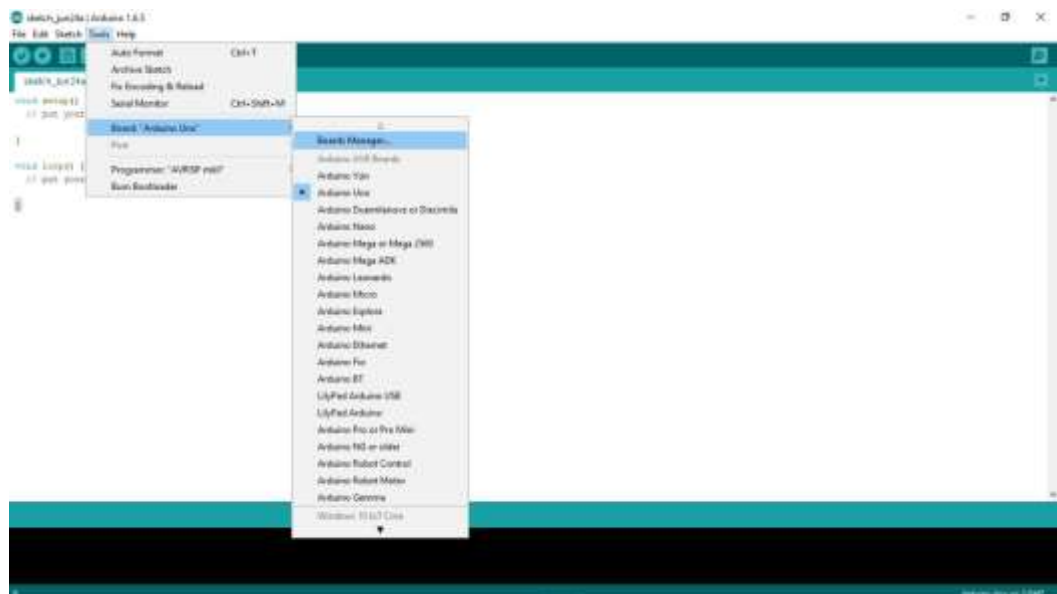
Gambar 27 Implementasi Wemos D1 R1 Pada Software Arduino (1)

3. Kemudian pada Additional Board Manager URL, masukkan URL berikut, http://arduino.esp8266.com/stable/package_esp8266com_index.json



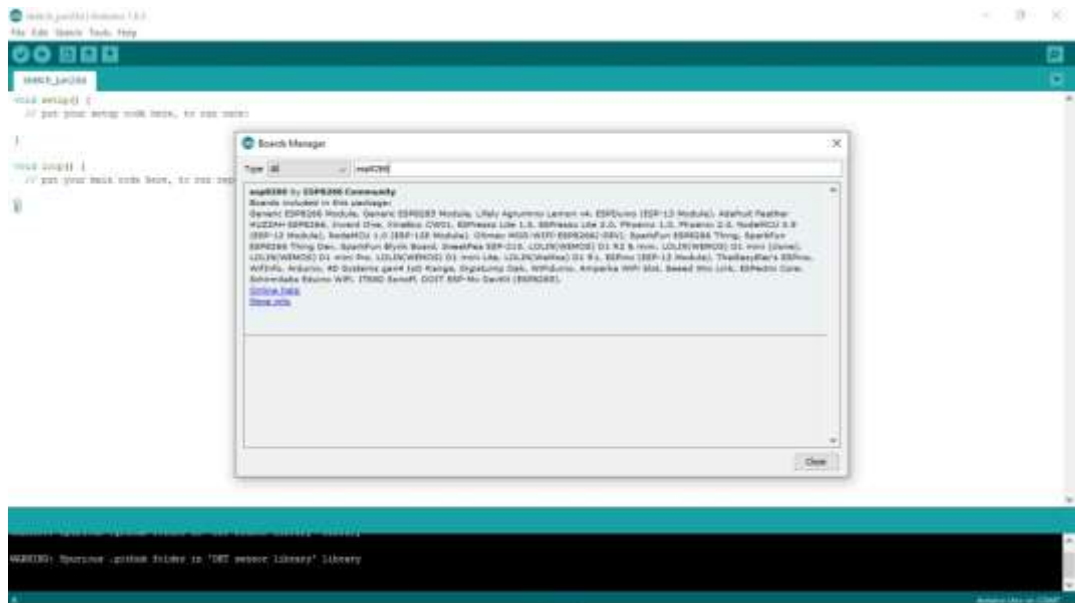
Gambar 28 Implementasi Wemos D1 R1 Pada Software Arduino (2)

4. Kemudian klik pada menu tools -> board -> board manager.



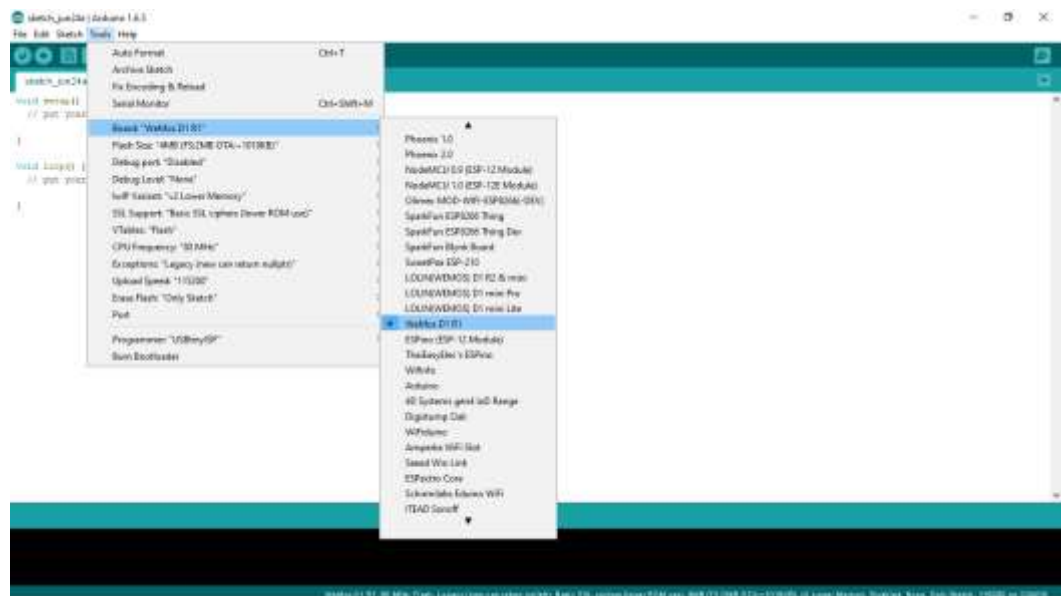
Gambar 29 Implementasi Wemos D1 R1 Pada Software Arduino (3)

5. Pada kolom pencarian ketik “esp”, maka akan muncul pilihan esp8266 by ESP8266 Community. Klik *install* dan tunggu hingga proses penginstalan selesai.



Gambar 30 Implementasi Wemos D1 R1 Pada Software Arduino (4)

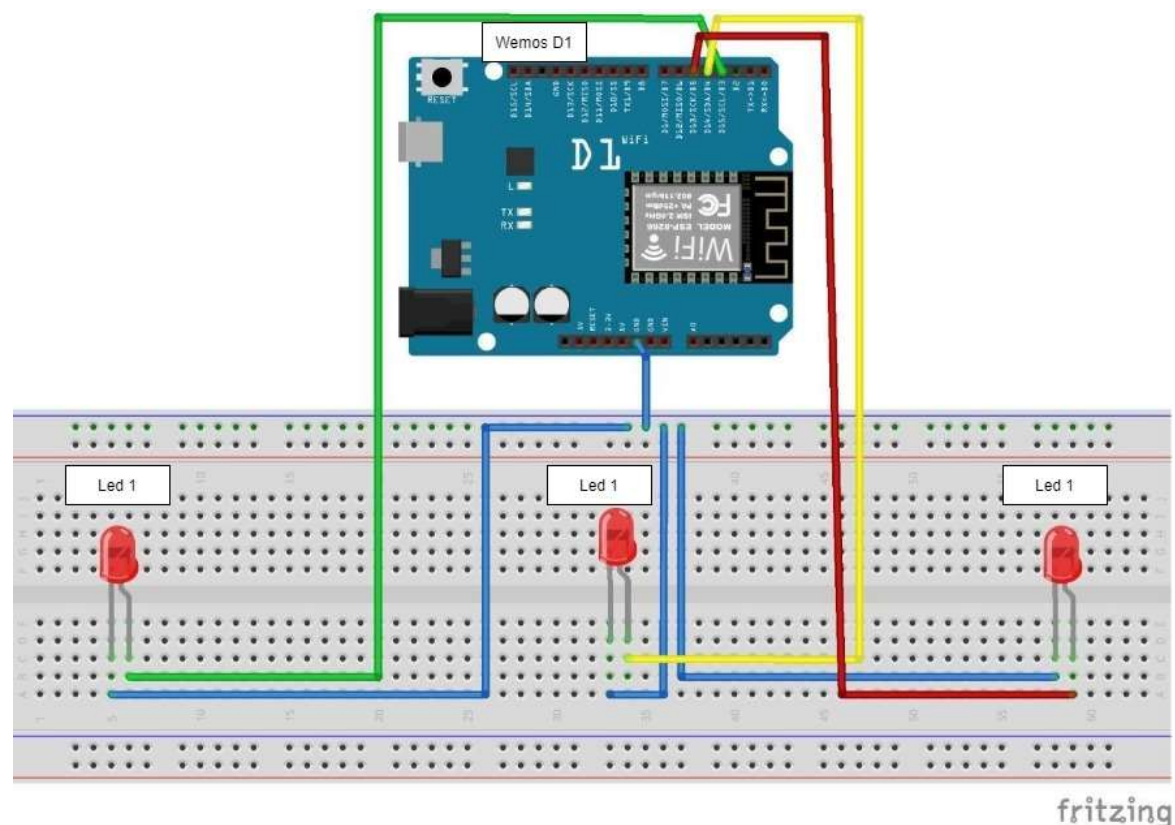
6. Setelah selesai di install, kita sudah dapat menggunakan pada Board kita Wemos D1 R1.



Gambar 31 Implementasi Wemos D1 R1 Pada Software Arduino (5)

4.1.2 Implementasi Rangkaian pada Prototype Kontroling Lampu

Implementasi rangkaian pada prototype pengontrolan hidup/mati adalah menghubungkan lampu led dengan pin wemos . Dalam rangkaian terdapat 3 lampu , dimana kaki Negatif pada led akan terhubung dengan GND. Kaki positif pada lampu led 1 akan terhubung dengan pin D3. Kaki positif pada lampu led 2 akan terhubung dengan pin D4. Kaki positif pada lampu led 3 akan tergubung dengan pin D5. Untuk rangkaian prototype dapat dilihat pada **Gambar 32** berikut.

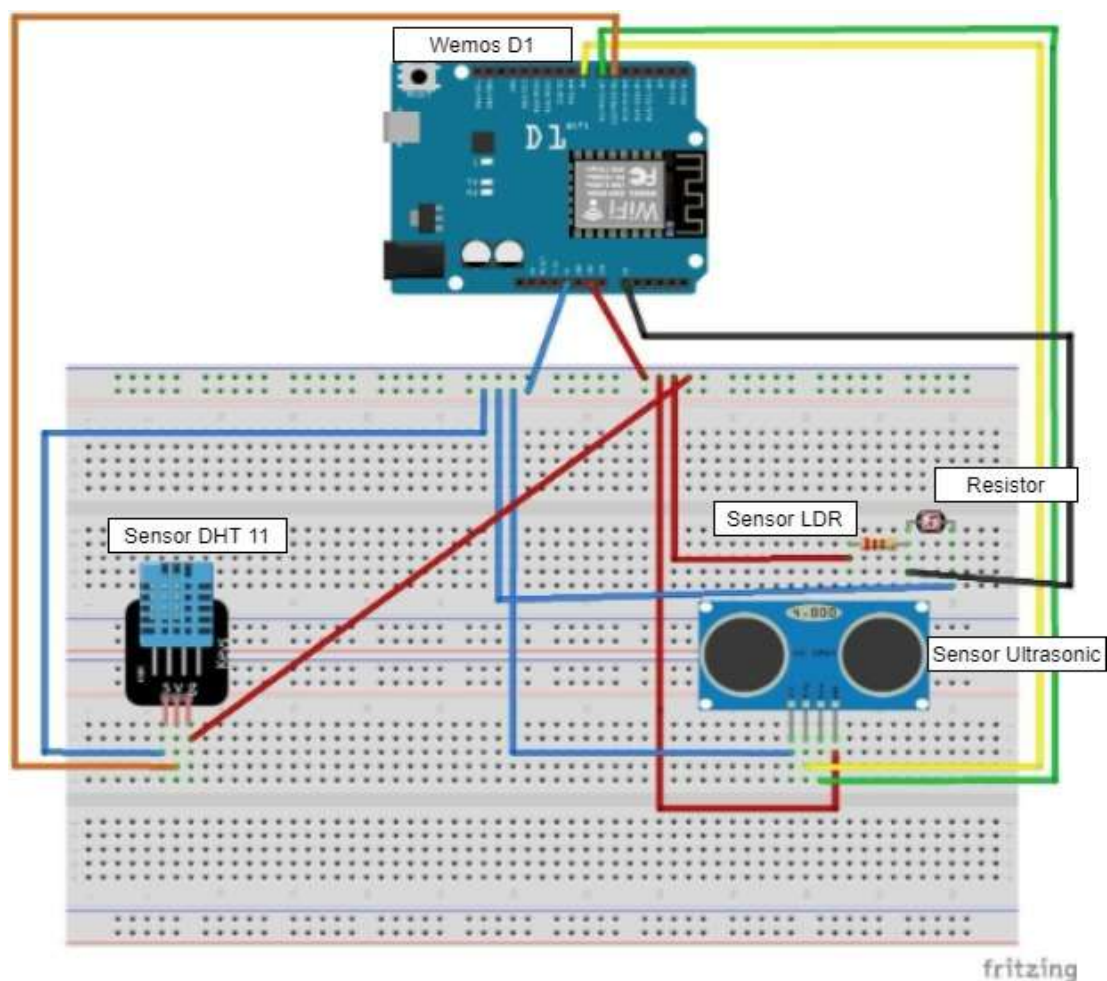


Gambar 32 Implementasi Rangkaian Pada Prototype Kontroling Lampu

4.1.3 Implementasi Rangkaian pada Prototype Pemantauan Tanaman

Implementasi rangkaian pada Prototype pemantauan tanaman yang menggunakan beberapa sensor adalah membuat penghubung antara pin . Pin yang ada pada sensor dihubungkan dengan menggunakan kabel *jumper*. Pada sensor DHT11 pin yang digunakan adalah pin VCC, *ground*, Data . Pin tersebut akan dihubungkan pada Wemos D1R1. Pin VCC dihubungkan menggunakan jumper berwarna biru ke pin 5 V sebagai tegangan positif terhadap sensor. Pin *ground* dihubungkan menggunakan jumper berwarna merah ke pin GND yang merupakan 0 V *power supply*. Pin tersebut menjadi sumber tegangan *negative*

sensor. Pin Data dihubungkan menggunakan jumper berwarna orange ke pin D6. Pada sensor Ultrasonic pin yang digunakan adalah pin VCC, *ground*, Echo, data 1 dan data 2. Pin VCC dihubungkan menggunakan jumper berwarna biru ke pin 5 V. Pin *ground* dihubungkan menggunakan jumper berwarna merah ke pin GND. Pin data 1 atau biasa disebut sebagai trigger terhubung pada kabel jumper kuning ke pin D8. Pin ini berfungsi untuk membangkitkan sinyal ultrasonik. Pin data 2 atau biasa disebut pin echo terhubung pada kabel hijau ke pin D7. Pin ini berfungsi mendeteksi sinyal pantulan ultrasonik. Pada sensor LDR pin yang digunakan adalah pin VCC, *ground* dan A0. Pin VCC akan dihubungkan dengan menggunakan kabel jumper berwarna biru. Pin *ground* akan dihubungkan dengan menggunakan kabel jumper berwarna merah. Pin A0 akan dihubungkan dengan pin A0, diman pin A0 berfungsi untuk input dan output digital yang dapat mengubah sinyal analog menjadi nilai digital yang mudah diukur. Untuk rangkaian prototype dapat dilihat pada **Gambar 33** berikut.



Gambar 33 Implementasi Rangkaian Pada Prototype Tanaman Sayur

4.1.4 Implementasi Kode

Pada subbab ini akan dijelaskan mengenai kode yang digunakan pada fungsi utama sistem yang dibangun.

4.1.4.1 Kode Fungsi CRUD Data Agent

Pada subbab ini dijelaskan kode yang digunakan untuk CRUD data dari *agent*, yang menjadi fungsi utamanya adalah tambah data, menampilkan data, *edit* data dan *delete*.

1. Kode fungsi tambah data *agent*

Pada kode fungsi data ini digunakan untuk menampilkan dan menjalankan fungsi untuk menambahkan data *agent* yang ditambahkan oleh *user*. Dimana *user* dapat menambahkan data *agent* melalui *button* tambah pada *dashboard* aplikasi *web*. Pada kode program ini akan menyimpan data *agent* dari *form* ke dalam variabel. Kemudian *master* akan menyimpan setiap nilai variabel tersebut ke dalam sebuah *array* JSON. *Array* *Json* ini kemudian akan disimpan ke dalam basis data dengan menggunakan fungsi `insertOne()`.

```
$pin_sensor = array();
    $pin_aktuator = array();
    $desc_sensor = array();
    $status = array();
    $id = $collection->count().''.date("h:i:sdmY");
    $ip = $_POST['ip'];
    $agent = $_POST['name'];
    $tipe = $_POST['tipe'];
    for ($i=1; $i <= $count ; $i++) {
        $str_desc      = 'desc'.$i;
        $str_sensor    = 'sensor'.$i;
        $pin_sensor[] = $_POST[$str_sensor];
        $desc_sensor[] = $_POST[$str_desc];
        $status[] = "mati";
    }
    $final = array(
        "id" => $id,
        "agent_ip" => $ip,
        "agent" => $agent,
        "sensor_pin" => $pin_sensor,
```

```

        "desc_sensor" => $desc_sensor,
        "status" => $status,
        "tipe" => $tipe
    );
    if($collection->insertOne($final)){
        header('Location: http://localhost/TA/TA/');
    } else {
        echo "nope";
    }
}

```

2. Kode fungsi *read data agent*

Pada fungsi *read* ini berfungsi untuk menunjukkan data yang sudah ditambahkan pada aplikasi *web* atau untuk *show data agent*. Pada kode program ini akan menyimpan data dari basis data ke dalam variabel *key*. Kemudian nilai setiap elemen variabel *key* akan ditampilkan oleh *master* di tampilan *web*.

```

for ($a=0; $a < $coun ; $a++) {
    echo "<tr>";
    echo "<td>".($a+1). "</td>";
    echo "<td>".$key->desc_sensor[$a]. "</td>";
    echo "<td>".$key->sensor_pin[$a]. "</td>";
    echo "<td>".$key->sensor_value[$a]. "</td>";
    echo "<td>".$key->actuator_pin[$a]. "</td>";
    echo "<td>".$key->status[$a]. "</td>";

    if ($key->status[$a] == 'active') {
        echo "<td> <center> <a
href='../deleterow.php?tipe=deact&job=deact&id=".$key-
>id."&row=".$a."'><button type='button' class='btn btn-
danger'>Deactivate</button></a></center></td>";

    }else{
        echo "<td> <center> <a
href='../deleterow.php?tipe=deact&job=activ&id=".$key-
>id."&row=".$a."'><button type='button' class='btn btn-
success'>Activate</button></a></center></td>";
    }

    echo "</tr>";
}
}

```

3. Kode fungsi *edit data agent*

Kode fungsi *edit* digunakan untuk mengubah data dari *agent* jika ada yang perlu dirubah dan fungsi *edit* sama dengan fungsi tambah data. Pada kode program ini akan menyimpan data *agent* dari *form* ke dalam *variabel*. Kemudian *master* akan menyimpan setiap nilai variabel tersebut ke dalam sebuah *array* JSON. Array Json ini kemudian akan disimpan ke dalam basis data dengan menggunakan fungsi `inserOne()`.

```
$pin_sensor = array();
$pin_aktuator = array();
$desc_sensor = array();
$status = array();
$id = $collection->count().''.date("h:i:sdmY");
$ip = $_POST['ip'];
$agent = $_POST['name'];
$tipe = $_POST['tipe'];
for ($i=1; $i <= $count ; $i++) {
    $str_desc      = 'desc'.$i;
    $str_sensor    = 'sensor'.$i;
    $pin_sensor[] = $_POST[$str_sensor];
    $desc_sensor[] = $_POST[$str_desc];
    $status[] = "mati";
}
$final = array(
    "id" => $id,
    "agent_ip" => $ip,
    "agent" => $agent,
    "sensor_pin" => $pin_sensor,
    "desc_sensor" => $desc_sensor,
    "status" => $status,
    "tipe" => $tipe
);
if($collection->insertOne($final)){
    header('Location: http://localhost/TA/TA/');
} else {
    echo "nope";
}
```

4. Kode fungsi *delete data agent*

Fungsi *delete* digunakan untuk menghapus data *agent* atau *agent* sendiri yang mungkin tidak dibutuhkan lagi. Pada kode program ini, file JSON yang memiliki *Id* yang sama dengan variabel *id* akan di hapus dengan menggunakan fungsi `deleteOne()`.

```
$id = $_GET['id'];
    if ($collection->deleteOne(['id' => $id])) {
        header('Location: http://localhost/TA/TA/');
    }
    else {
        echo "fail";}
```

4.1.4.2 Kode Fungsi Koneksi ke Database

Pada subbab ini diterakan kode fungsi koneksi untuk database. Pada kode program ini, variabel *con* akan menghubungkan aplikasi *web* dengan basis data MongoDB yang berjalan pada port 27017.

```
<?php
    require 'vendor/autoload.php';
    $con = new MongoDB\Client("mongodb://localhost:27017");
    ?>
```

4.1.4.3 Kode Fungsi Kirim Data *Agent* ke API

Pada subbab berikut diterakan kode fungsi kirim data *agent* ke API yang dilakukan melalui HTTP *body*. Pada kode program ini, file JSON akan dikirimkan melalui *body* HTTP oleh *client* dengan menggunakan fungsi `http.POST()`.

```
inhttpResponseCode=http.POST("{\"id\":\"201:209322012\",\"sensor_value\":
[sensor1,++++sensor2]}");
```

4.1.4.4 Kode Fungsi Translasi JSON ke dalam *Query MongoDB*

Pada subbab ini diterakan kode fungsi translasi JSON ke dalam *query* mongoDB yang berlangsung di dalam API *Gateway*. Pada kode program ini, file JSON akan terlebih dahulu disimpan pada variable *data*. Kemudian, *master* akan melakukan pengecekan terhadap status dari sensor, jika sensor memiliki status “aktif” maka nilai aslinya akan tetap disimpan, sementara jika sensor memiliki status “deaktif” maka nilai akan di set menjadi nol.

```

$data = json_decode(file_get_contents("php://input"));
for ($a=0; $a < count($data->sensor_value) ; $a++) {
    if($checkID->status[$a] == 'active'){
        $data->sensor_value[$a] = $data->sensor_value[$a];
    }else{
        $data->sensor_value[$a] = 0;
    }
}

```

4.1.4.5 Kode Fungsi Url untuk Mengontrol *Agent*

Pada subbab ini akan diterakan kode fungsi untuk membuat Url untuk mengontrol *agent* yang akan berlangsung di *API Gateway*. Pada kode program ini akan melakukan pengecekan terhadap setiap nilai sensor yang di set oleh pengguna di dalam basis data dengan nilai sensor yang diterima. Setiap kondisi yang bersifat benar akan menambahkan */on/* ke dalam url, namun jika salah akan menambahkan */off/* ke dalam url.

```

$url=$checkID->controller_ip."/";
$separate = "/";
// echo $checkID->state[$i];
if ( $checkID->state[$i] == "=") {
    if ($kiri == $kanan) {
        echo $kiri. " ". $kanan;
        $url = $url."".$checkID->actuator_pin[$i]."/on/";
    }else{
        $url = $url."".$checkID->actuator_pin[$i]."/off/";
    }
}elseif($checkID->state[$i] == ">"){
    if ($kiri > $kanan) {
        $url = $url."".$checkID->actuator_pin[$i]."/on/";
    }else{
        $url = $url."".$checkID->actuator_pin[$i]."/off/";
    }
}elseif($checkID->state[$i] == "<"){
    if ($kiri < $kanan) {

```

```

                $url = $url."".$checkID-
>actuator_pin[$i]."/on/";
            }else{
                $url = $url."".$checkID-
>actuator_pin[$i]."/off/";
            }
        }elseif($checkID->state[$i] == "!="){
            if ($kiri != $kanan) {
                $url = $url."".$checkID-
>actuator_pin[$i]."/on/";
            }else{
                $url = $url."".$checkID-
>actuator_pin[$i]."/off/";
            }
        }
    }
}

```

4.1.4.6 Kode Fungsi Update Data ke MongoDB

Pada subbab ini diterakan kode fungsi untuk *update* data ke mongodb yang akan berlangsung di *API Gateway*. Pada kode program ini, file JSON yang sudah siap untuk disimpan ke dalam basis data akan disimpan di dalam variabel data. Data akan di masukkan ke dalam basis data dengan menggunakan fungsi `insertOne()` .

```
$collection->insertOne($data)
```

4.1.4.7 Kode Fungsi Rata-Rata

Pada subbab ini diterakan kode fungsi yang digunakan untuk melakukan rata-rata terhadap data sensor yang terdapat pada studi kasus orkestrasi beberapa sensor dan akan ditampilkan dalam bentuk *chart* pada aplikasi *web*. Pada kode program ini, akan mengelompokkan data sesuai dengan interval waktu tertentu. *Master* akan mencari rata-rata dari setiap interval dan kemudian akan ditampilkan pada *web*.

```

$date = strtotime($keys[$abc]->tanggal." ".$keys[$abc]->waktu);
                                if($date >= ($min + $bot
*$timeinterval) && $date <($min + ($bot+1) *$timeinterval) ){
                                for ($abc=0; $abc < $coun;
$abc++) {

$penampung[$abc][$bdata] = $keys[$abc]->sensor_value[$abc] +
$penampung[$abc][$bdata];

```

```
}  
$rata2 ++;}
```

4.1.4.8 Kode Fungsi CRUD User

Pada subbab ini diterakan fungsi CRUD pada *user*. Pada kode program ini, file JSON yang sudah siap untuk disimpan ke dalam basis data disimpan di dalam variabel data. Data akan di masukkan ke dalam basis data dengan menggunakan fungsi `insertOne()` .

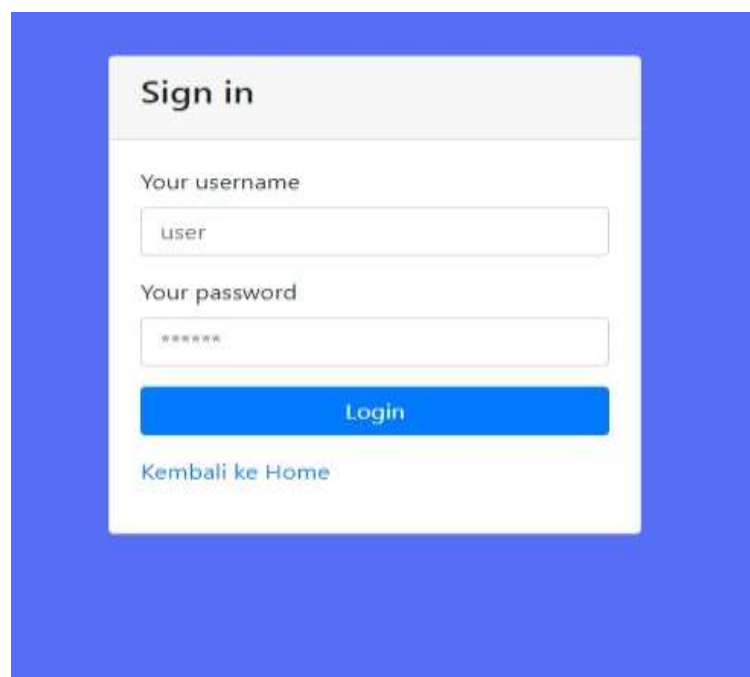
```
$collection->insertOne($final)
```

4.1.5 Implementasi Web Aplikasi

Dalam sub bab ini menjelaskan mengenai implementasi pembangunan web aplikasi yang diharapkan dapat membantu *user* dalam melakukan kontroling dan monitoring terhadap beberapa *device* IoT. Web aplikasi yang dibangun menggunakan Bahasa pemrograman PHP dengan *tools* Sublime.

1. Komponen Layar Login

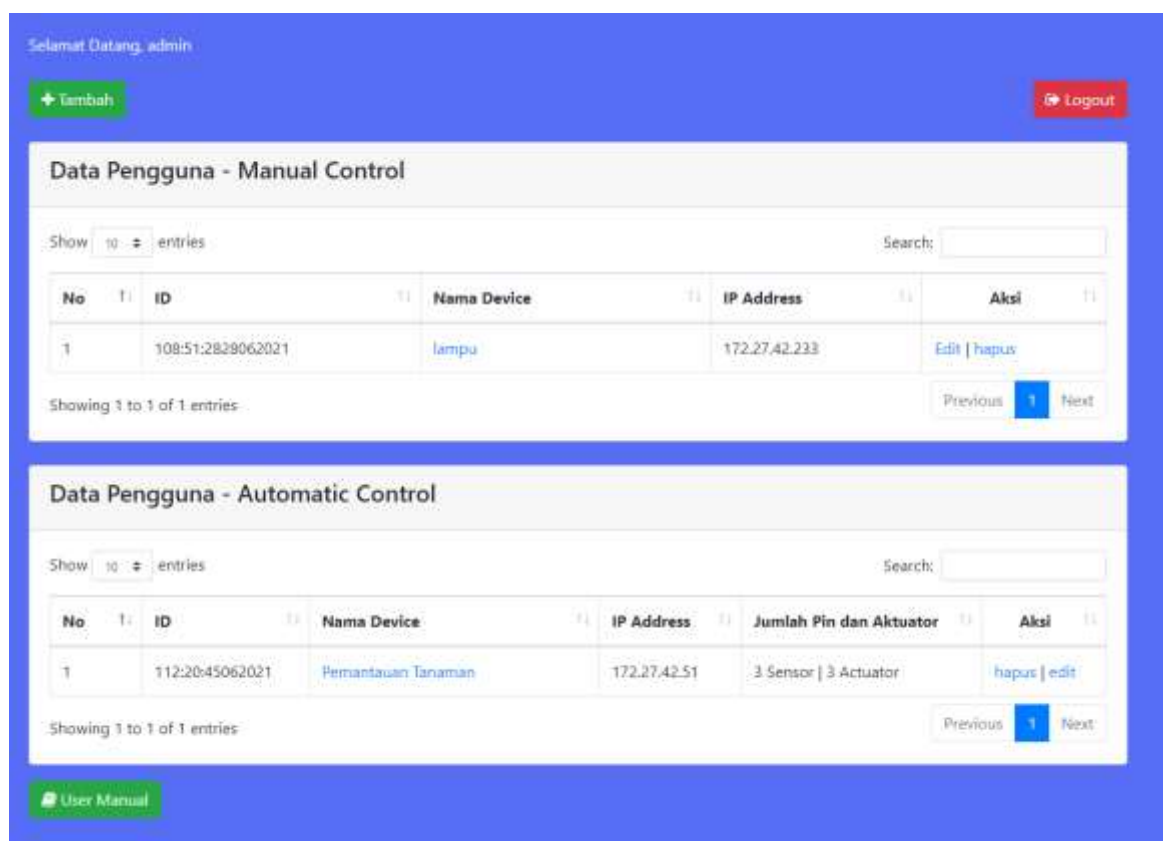
Pada komponen layar login, *user* sebagai *administrator* dapat menggunakan web aplikasi ini dengan melakukan autentikasi terlebih dahulu agar dapat memasuki *dashboard* web aplikasi. Berikut tampilan web aplikasi untuk melakukan *login* dapat dilihat pada **Gambar 34**.



Gambar 34 Tampilan Halaman Login

2. Komponen Layar Utama (Dashboard)

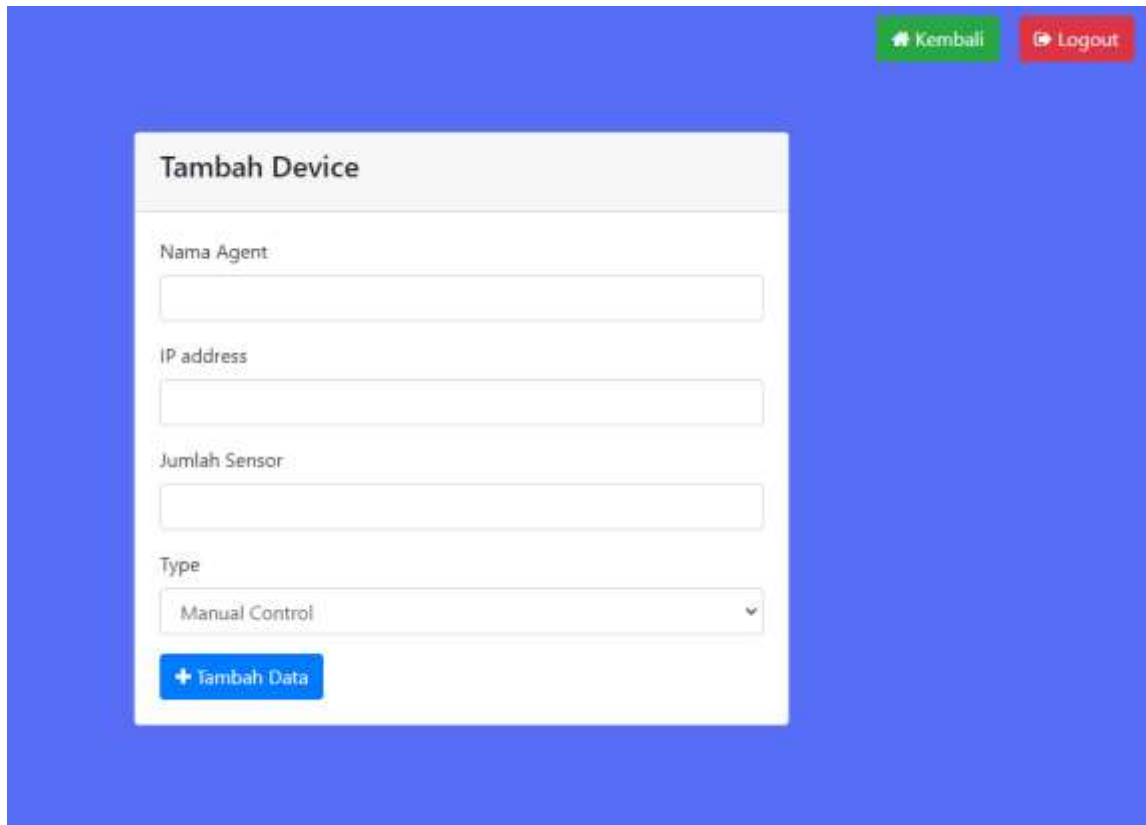
Pada komponen layar utama ketika *user* berhasil melakukan login, maka akan di arahkan ke tampilan *dashboard* yang langsung menunjukkan tabel *agent* yang dikontrol secara manual dan otomatis. Pada tampilan *dashboard* sesuai dari studi kasus yang diambil, pada bagian tabel “Data Pengguna - ManualControl” dimasukkan studi kasus pengontrolan *on/off* lampu, terdapat ID dari agent, nama dari setiap *device*, ip address, dan aksi yang berisi edit dan hapus agent. Pada bagian tabel “Data Pengguna – AutomaticControl” dimasukkan studi kasus ”Pemantauan Tanaman” , terdapat ID dari *agent*, nama dari setiap *device*, Jumlah pin dan aktuator lalu aksi yang berisi edit dan hapus *agent*. Pada halaman *dashboard* juga terdapat menu *user manual* yang dapat digunakan pengguna sebagai panduan penggunaan *web* yang berisi informasi untuk menambahkan *agent* dan juga dilampirkan kode dari *device* IoT yang akan didaftarkan. Berikut tampilan komponen layar utama yang bisa dilihat pada Gambar 35.



Gambar 35 Tampilan Halaman Dashboard

3. Komponen Layar Tambah *Agent*

Pada komponen layar tambah, jika *user* memilih *button* tambah *agent*, akan ditampilkan halaman form pengisian informasi dari *agent* yang akan ditambahkan. *User* akan memasukkan nama *agent*, Ip address, jumlah sensor beserta tipe dari *agent* apakah *agent* yang ditambahkan dikontrol secara manual atau otomatis. Tampilan halaman seperti pada **Gambar 36** sebagai berikut.



Gambar 36 Tampilan Halaman Tambah *Agent*

4. Komponen Layar Tambah *Agent Tipe Automatic Control*

Komponen layar tambah *agent* untuk yang dikontrol secara otomatis, jika *user* memilih tipe dari *agent* yang ditambahkan adalah “Automatic Control” maka selanjutnya *user* akan diarahkan ke halaman baru untuk memberi informasi lebih terperinci mengenai sensor apa saja yang akan digunakan, dengan mengisi nama-nama dari sensor, pin dari masing-masing sensor, nilai dari masing-masing sensor dan juga pin dari aktuator yang akan dihubungkan ke masing-masing sensor nantinya saat melakukan konfigurasi. Tampilan dari halaman seperti pada **Gambar 37** dibawah ini.

Gambar 37 Tampilan Halaman Tambah Agent Automatic Control

5. Komponen Layar Tambah Agent Tipe *Manual Control*

Komponen layar tambah *agent* untuk yang dikontrol secara manual, jika *user* memilih tipe dari *agent* yang ditambahkan adalah “Manual Control” maka selanjutnya *user* akan diarahkan ke halaman baru untuk memberi informasi mengenai nama dari alat dan pin sensornya. Tampilan halaman seperti pada Gambar 38 dibawah ini.

Gambar 38 Tampilan Halaman Tambah Agent Manual Control

6. Komponen Layar Edit Agent Tipe Automatic Control

Untuk komponen layar edit, pada bagian aksi yang berisi *button* “hapus” dan “edit” *user* dapat menghapus dan mengedit setiap agent yang dipilih. Ketika *user* memilih untuk menghapus agent tertentu, *user* akan klik hapus maka agent yang dipilih akan terhapus lalu semua perangkat-perangkat yang berada di dalam agent tersebut juga akan ikut terhapus.

Pada bagian *button* edit untuk “Automatic Control”, *user* dapat melakukan edit deskripsi dari agent yang sudah ditambahkan. Tampilan seperti pada **Gambar 39** sebagai berikut.

No	Nama Sensor	Pin Sensor	State	Nilai Sensor	Satuan	pin aktuator	Aksi
1	Sensor Jarak	7	>	8	cm	3	Hapus
2	Sensor LDR	0	>	100	cd	4	Hapus
3	Suhu	6	>	20	C	5	Hapus

Gambar 39 Tampilan Halaman Edit Agent Automatic Control

7. Komponen Layar Edit Agent Tipe Manual Control

Pada komponen layar edit bagian *button* untuk “Manual Control”, *user* dapat melakukan edit menghapus salah lampu yang sudah didaftarkan. Tampilan seperti **Gambar 40** sebagai berikut.

No	Nama Sensor	Pin Sensor	Aksi
1	1	4	Hapus
2	2	5	Hapus
3	2	3	Hapus

Gambar 40 Tampilan Halaman Edit Agent Manual Control

4.2 Pengujian

Pada sub bab ini menjelaskan mengenai pengujian yang dilakukan setelah melakukan implementasi. Pengujian ini dilakukan untuk memastikan produk yang dibangun berfungsi sesuai dengan tujuan dan dapat bekerja dengan baik. Dalam pengujian produk yang telah dibangun terdapat beberapa pengujian yang dilakukan, untuk deskripsinya dapat dijelaskan pada sub sub bab dibawah ini.

4.2.1 Pengujian Fungsi CRUD Pada Aplikasi Web

4.2.1.1 Pengujian Fungsi *Create Agent Tipe Manual Control*

Pada pengujian *create* akan diuji pada saat melakukan penambahan *device* pengontrolan *on/off* lampu, dimana akan ditampilkan halaman sebagai berikut. Kemudian *user* akan menginput data *device* yang ingin dimasukkan dan *IP Address* yang sudah ditentukan. Mengisi jumlah sensor misalnya seperti yang tertera pada gambar berjumlah 3, dimana pin sensor tersebut disesuaikan berdasarkan rangkaian yang telah dibuat. Dalam *case* pengontrolan *on/off* lampu, pengontrolan akan dilakukan secara manual. Pada pengujian penambahan *device* ini, *user* mengisi data sebagai berikut :

Nama Agent	:	Lampu Rumah
IP Address	:	192.17.128.17
Jumlah Sensor	:	3
Type	:	Manual Control

Gambar 41 Pengujian Penambahan Agent Kontroling Lampu (1)

Kemudian mengisi nama alat dan pin sensor sesuai dengan rangkaian yang telah dibuat. Berikut data yang diinput oleh *user* untuk deskripsi *agent* baru :

Nama Alat	Pin sensor
Lampu depan	3
Lampu tengah	4
Lampu teras	5

[Kembali](#)
[Logout](#)

Tambah Data Agent

Nama: Lampu Rumah
 IP : 192.17.128.17
 Pin: 3
 Tipe: Manual Control

Nama Alat

Lampu Depan

Pin Sensor 1

3

Nama Alat

Lampu tengah

Pin Sensor 2

4

Nama Alat

Lampu teras

Pin Sensor 3

5

+ Tambah Data

Gambar 42 Pengujian Penambahan Agent Kontroling Lampu (2)

Lampu telah berhasil ditambahkan dapat dilihat pada **Gambar 43** berikut atas nama *device* Lampu Rumah. Perlu diketahui pin yang dimasukkan oleh pengguna harus sesuai dengan deklarasi pin pada kode rangkaian prototype.

Selamat Datang, admin

[+ Tambah](#)
[Logout](#)

Data Pengguna - Manual Control

Show 10 entries Search:

No	ID	Nama Device	IP Address	Aksi
1	310:15:3627072021	Lampu Rumah	192.17.128.17	Edit hapus
2	108:51:2828062021	lampu	172.27.42.233	Edit hapus

Showing 1 to 2 of 2 entries

[Previous](#)
[1](#)
[Next](#)

Gambar 43 Pengujian Penambahan Agent Kontroling Lampu (3)

Dan Ketika *user* mengakses nama *agent* yang baru ditambah, akan menampilkan halaman seperti **Gambar 44** berikut. Dimana pada halaman ini *user* akan mengatur pengontrolan *on/off* pada masing-masing lampu. Perlu diketahui, bahwa pada pembangunan sistem ini lampu yang digunakan masih jenis LED karena dalam proyek ini prototype yang dibangun masih berupa mediator, namun untuk penamaan dalam uji coba menambahkan *agent* baru menggunakan nama “Lampu Rumah”.



Gambar 44 Pengujian Penambahan Agent Kontroling Lampu (4)

4.2.1.2 Pengujian Fungsi *Edit Tipe Manual Control*

Pada pengujian fungsi *edit* akan dilakukan pada *agent* dengan nama “Lampu Rumah” yang sudah di *create* sebelumnya. *User* akan memilih *button edit* pada aksi, maka akan beralih ke halaman seperti pada **Gambar 45** berikut, dimana pin yang diubah sudah disesuaikan dengan rangkaian yang dibuat. Pada saat melakukan edit, *user* juga dapat menambah dan mengurangi alat, mengganti nama alat, mengganti ip address dari *device*. Setelah selesai melakukan pengeditan, *user* dapat menekan tombol tambah data.

1. Pada pengujian fungsi *edit* kali ini, akan diuji pada saat mengubah nama alat dan pin sensor. Berikut data yang diubah oleh *user* yang diubah dari *create data agent* sebelumnya :

Nama Alat	Pin sensor
Lampu depan	3
Lampu dapur	4
Lampu Kamar Mandi	6

Edit Device - Manual Control

Nama:

IP Address sensor:

No	Nama Sensor	Pin Sensor	Aksi
1	<input type="text" value="Lampu depan"/>	<input type="text" value="3"/>	Hapus
2	<input type="text" value="Lampu Dapur"/>	<input type="text" value="4"/>	Hapus
3	<input type="text" value="Lampu Kamar Mandi"/>	<input type="text" value="6"/>	Hapus

[Tambah Baris](#) | [Hapus Baris](#)

[+ Tambah Data](#)

Gambar 45 Pengujian Edit Device Kontroling Lampu (1)

Berikut tampilan halaman *agent* Lampu Rumah setelah berhasil di edit, dapat dilihat pada **Gambar 46**.

Home [Logout](#)

Nama Agent: Lampu Rumah
192.17.128.17

Show entries Search:

No	Nama Alat	Status	Aksi
1	Lampu depan	● mati	Hidup Mati
2	Lampu Dapur	● mati	Hidup Mati
3	Lampu Kamar Mandi	● mati	Hidup Mati

Showing 1 to 3 of 3 entries Previous **1** Next

Gambar 46 Pengujian Edit Device Kontroling Lampu (2)

2. Pada pengujian fungsi *edit* selanjutnya, akan diuji pada saat mengubah nama *agent*, ip address dan mengurangi baris atau jumlah alat. Untuk mendapatkan ip address yang baru, pengguna harus kembali menjalankan kode program *device* untuk mendapatkan ip address

yang baru. Berikut data yang diubah oleh *user* yang diubah dari *create data agent* sebelumnya :

Nama Agent	Ip address	Nama Alat	Pin Sensor
Kontrol Lampu	172.20.40.54	Lampu Depan	3

Gambar 47 Pengujian Edit Device Kontroling Lampu (3)

Berikut tampilan halaman home Kotrol Lmapu setelah berhasil diedit, dapat dilihat pada Gambar 48.

Gambar 48 Pengujian Edit Device Kontroling Lampu (4)

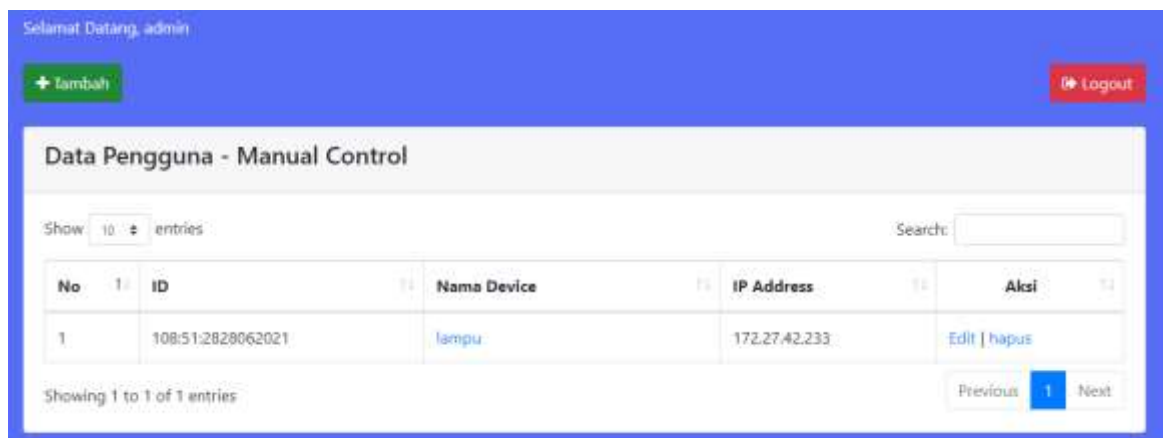
4.2.1.3 Pengujian Fungsi *Delete Agent Tipe Manual Control*

Pada pengujian fungsi *delete* kali ini akan dilakukan pada *agent* dengan nama “Lampu Rumah”, dimana pada halaman home bagian *Manual Control*, *user* memilih aksi hapus dimana fungsi ini akan menghapus keseluruhan *device* tersebut termasuk perangkat yang ada di dalam sistem tersebut.



Gambar 49 Pengujian Delete Device Kontroling Lampu (1)

Berikut tampilan halaman *home* bagian *manual control* setelah berhasil di hapus.



Gambar 50 Pengujian Delete Device Kontroling Lampu (2)

4.2.1.4 Pengujian Fungsi *Create Agent Tipe Automatic Control*

Pada saat melakukan pengujian fungsi *create* akan dilakukan pada saat melakukan penambahan *device* yang baru , dimana akan ditampilkan halaman sebagai berikut. Kemudian *user* akan menginput data dari *device* yang ingin dimasukkan dan *IP Address* yang sudah ditentukan. Mengisi jumlah sensor misalnya seperti yang tertera pada gambar berjumlah 3 yang disesuaikan berdasarkan rangkaian yang telah dibuat. Dalam *case*

pemantauan tanaman menggunakan beberapa sensor, pengontrolan akan dilakukan secara Otomatis Pada pengujian penambahan pemantauan tanaman, *user* mengisi data sebagai berikut :

Nama Agent	:	Pemantauan Tanaman
IP Address	:	192.168.17.122
Jumlah sensor	:	3
Type	:	Automatic Control

Gambar 51 Pengujian Penambahan Agent Automatic Control(1)

Kemudian pada halaman selanjutnya yang dilakukan *user* adalah mengisi Nama sensor yang diperlukan untuk *device* yang baru, pin sensor , *state* , nilai sensor sebagai acuan, satuan dari nilai sensor, pin aktuator dan juga jumlah sensor sesuai dengan rangkaian yang telah dibuat. *State* berfungsi untuk menjadi acuan server untuk memberikan perintah kepada *agent*. Jika *state* bernilai "=", maka aktuator akan dihidupkan jika nilai sensor yang didapatkan sama dengan nilai sensor yang berada di dalam server. Berikut data yang dimasukkan oleh *user* :

Nama Sensor	Pin Sensor	State	Nilai Sensor	Satuan	Pin Aktuator
Kelembapan	6	>	75	RH	2
Temperatur	0	>	25	C	3
Intensitas Cahaya	7	>	100	Cd	4

Gambar 52 Pengujian Penambahan Agent Automatic Control (2)

Pada halaman tambah data *agent* pada Gambar 52 berikut, nilai yang berada pada kolom pin sensor merupakan informasi yang diberikan kepada pengguna mengenai letak pin dari masing-masing sensor, dan juga pin sensor disini tidak berhubungan dengan komunikasi data, pengontrolan dan yang lainnya. Nilai yang ada pada kolom nilai sensor merupakan acuan sensor terhadap kondisi lingkungan. Nilai yang ada pada kolom pin aktuator merupakan nilai output yang sudah dideklarasikan pada kode *hardware*. Dalam kasus *automatic control* ini, aktuator yang dipakai masih menggunakan LED sebagai mediator yang digunakan untuk memastikan aktuatornya *on/off*.

Perlu diketahui terkait pin sensor dan aktuator yang diinput oleh pengguna di *web* pada saat menambahkan *agent* baru, harus sesuai dengan deklarasi pin yang sudah dikonfigurasi pada kode program *hardware* sebelumnya. Karena jika pengguna menginput pin aktuator yang salah, aktuator tersebut tidak akan bisa dikontrol dan jika pengguna menginput pin

sensor yang tidak sesuai dengan deklarasi maka akan ada kesalahan data yang akan diterima oleh pengguna.

Agent yang baru telah berhasil ditambahkan dapat dilihat pada **Gambar 55** berikut atas nama *device* Pemantauan Tanaman.

Data Pengguna - Automatic Control

Show10entries

Search:

No	ID	Nama Device	IP Address	Jumlah Pin dan Aktuator	Aksi
1	109:21:36082021	pemantauan Sayur	172.20.40.120	3 Sensor 3 Aktuator	hapus edit
2	404:30:55082021	pemantauan tanaman	192.168.17.122	3 Sensor 3 Aktuator	hapus edit

Showing 1 to 2 of 2 entries

Previous

1

Next

Gambar 53 Pengujian Penambahan Agent Automatic Control (3)

Dan Ketika *user* mengakses nama *agent* yang baru ditambah, akan menampilkan halaman seperti gambar berikut. Dimana pada halaman ini *user* akan menerima data dari masing-masing sensor yang diterima dari lingkungan, status dari masing-masing sensor dan juga aksi yang dapat dikontrol oleh *user*. Pada *page automatic control device* akan diterakan template *script* untuk kirim data yang dapat digunakan oleh pengguna untuk mengurangi adanya *human error* atau mempermudah pengguna pada saat melakukan indeksasi data pada *agent* dapat dilihat pada **Gambar 54**.

Home
Logout

Nama Sistem: pemantauan tanaman

ID : 404:30:55062021

IP Sensor : 192.168.17.122

IP Controller : 192.168.17.132

Data Sensor

No	Desc Sensor	Pin Sensor	Nilai Batas Sensor	Pin Aktuator	Status	Aksi
1	Kelembapan	6	75 RH	2	active	Deactivate
2	Temperatur	0	25 C	3	active	Deactivate
3	Intensitas Cahaya	7	100 Cd	4	active	Deactivate

Snippet

```
String str = "(" + id + "," + "112:20:45062021" + "," + sensor_value + ")";
String comma = ",";
String str2 = "[" + String(Kelembapan) + comma + String(Temperatur) + comma + String(IntensitasCahaya) + "]";
String str3 = "[]";
String str4 = str + str2 + str3;
int httpResponseCode = http.POST(str4);
```

No	Kelembapan (RH)	Temperatur (C)	Intensitas Cahaya (Cd)	Waktu
----	-------------------	------------------	--------------------------	-------

Gambar 54 Pengujian Penambahan Agent Automatic Control (4)

Pada Gambar 55 berikut dapat kita lihat hasil dari *agent* pemantauan tanaman dari setiap sensor yang didapatkan dari lingkungan setiap 6 detik dan nilai rata-rata sensor yang ditampilkan dalam bentuk *chart*.



Gambar 55 Pengujian Penambahan Agent Automatic Control (5)

4.2.1.5 Pengujian Fungsi *Edit Agent Tipe Automatic Control*

Pada pengujian fungsi *edit* akan dilakukan pada *agent* dengan nama “Pemantauan Sayur” yang sudah di *create* sebelumnya. *User* akan memilih *button edit* pada aksi, maka akan beralih ke halaman seperti pada gambar berikut, dimana pin yang diubah sudah disesuaikan dengan rangkaian yang dibuat. Pada saat melakukan *edit*, *user* juga dapat menambah dan mengurangi sensor, mengganti nama sensor, mengganti ip address sensor dan controllernya. Setelah selesai melakukan pengeditan, *user* dapat menekan tombol tambah data.

1. Pada pengujian fungsi *edit* kali ini, akan diuji pada saat mengubah keterangan dari setiap sensor pada semua baris dan menambahkan satu baris untuk sensor baru. Berikut data yang diubah oleh *user* yang diubah dari *create* data *agent* sebelumnya :

Nama Sensor	Pin Sensor	State	Nilai Sensor	Satuan	Pin Aktuator
Kelembapan	6	>	75	RH	2
Temperatur	0	>	25	C	3
Intensitas Cahaya	7	>	100	Cd	4
Jarak	9	=	30	Cm	6

Gambar 56 Pengujian Edit Device Automatic Control (1)

Berikut tampilan halaman *agent* Pemantauan Tanaman setelah berhasil di edit, dapat dilihat pada **Gambar 57** berikut.

[Home](#)
[Logout](#)

Nama Sistem: pemantauan tanaman

ID : 404:30:55082021

IP Sensor : 192.168.17.122

IP Controller: 192.168.17.132

Data Sensor

Show entries
 Search:

No	Desc Sensor	Pin Sensor	Nilai Batas Sensor	Pin Aktuator	Status	Aksi
1	Kelembapan	6	75 RH	2	active	Deactivate
2	Temperatur	0	25 C	3	active	Deactivate
3	Intensitas Cahaya	7	100 Cd	4	active	Deactivate
4	Jarak	8	10 cm	5	active	Deactivate

Showing 1 to 4 of 4 entries

[Previous](#)
[1](#)
[Next](#)

Snippet

```
String str = "{id\":\"112:20:45062021\", \"sensor_value\"";
String comma = ",";
String str2 = "[" + String(Kelembapan) + comma + String(Temperatur) + comma + String(IntensitasCahaya) + comma + String(Jarak) ;
String str3 = "]\"";
String str4 = str + str2 + str3;
int httpResponseCode = http.POST(str4);
```

Gambar 57 Pengujian Edit Device Automatic Control (2)

Pada Gambar 58 berikut dapat kita lihat hasil dari *agent* pemantauan tanaman yang sudah diedit , kita dapat mengetahui data nilai dari setiap sensor yang didapatkan dari lingkungan setiap 6 detik dan nilai rata-rata sensor yang ditampilkan dalam bentuk *chart*.



Gambar 58 Pengujian Edit Device Automatic Control (3)

2. Pada pengujian fungsi *edit* selanjutnya, akan diuji pada saat mengubah nama *agent*, ip address, ip controller dan mengurangi baris atau jumlah sensor. Untuk mendapatkan ip address dan ip controller yang baru, pengguna harus kembali menjalankan kode program *device* untuk mendapatkan ip address dan ip controller yang baru. Berikut data yang diubah oleh *user* yang diubah dari *create* data *agent* sebelumnya, nama *agent* “pemantauan tanaman angrek”, ip address “172.20.40.42”, ip controller “172.20.40.44”.

Nama Sensor	Pin Sensor	State	Nilai Sensor	Satuan	Pin Aktuator
Kelembapan	6	>	75	RH	2
Jarak Tanaman	8	>	10	C	5

[Kembali](#)
[Logout](#)

Edit Device - Automatic Control

Nama

IP Address sensor

IP Address Controller

No	Nama Sensor	Pin Sensor	State	Nilai Sensor	Satuan	pin aktuator	Aksi
1	<input type="text" value="Kelembapan"/>	<input type="text" value="6"/>	<input type="text" value=">"/>	<input type="text" value="75"/>	<input type="text" value="RH"/>	<input type="text" value="2"/>	Hapus
2	<input type="text" value="Jarak"/>	<input type="text" value="8"/>	<input type="text" value=">"/>	<input type="text" value="10"/>	<input type="text" value="C"/>	<input type="text" value="5"/>	Hapus

[Tambah Baris](#) | [Hapus Baris](#)

[+ Tambah Data](#)

Gambar 59 Pengujian Edit Device Automatic Control (4)

Pada **Gambar 60** berikut merupakan tampilan halaman *home* pamantauan tanaman setelah berhasil diedit, dapat dilihat pada gambar berikut.

[Home](#)
[Logout](#)

Nama Sistem: pemantauan tanaman anggrek

ID : 404:30:55082021

IP Sensor : 172.20.40.42

IP Controller: 172.20.40.44

Data Sensor

Show entries

Search:

No	Desc Sensor	Pin Sensor	Nilai Batas Sensor	Pin Aktuator	Status	Aksi
1	Kelembapan	6	75 RH	2	active	Deactivate
2	Jarak	8	10 C	5	active	Deactivate

Showing 1 to 2 of 2 entries

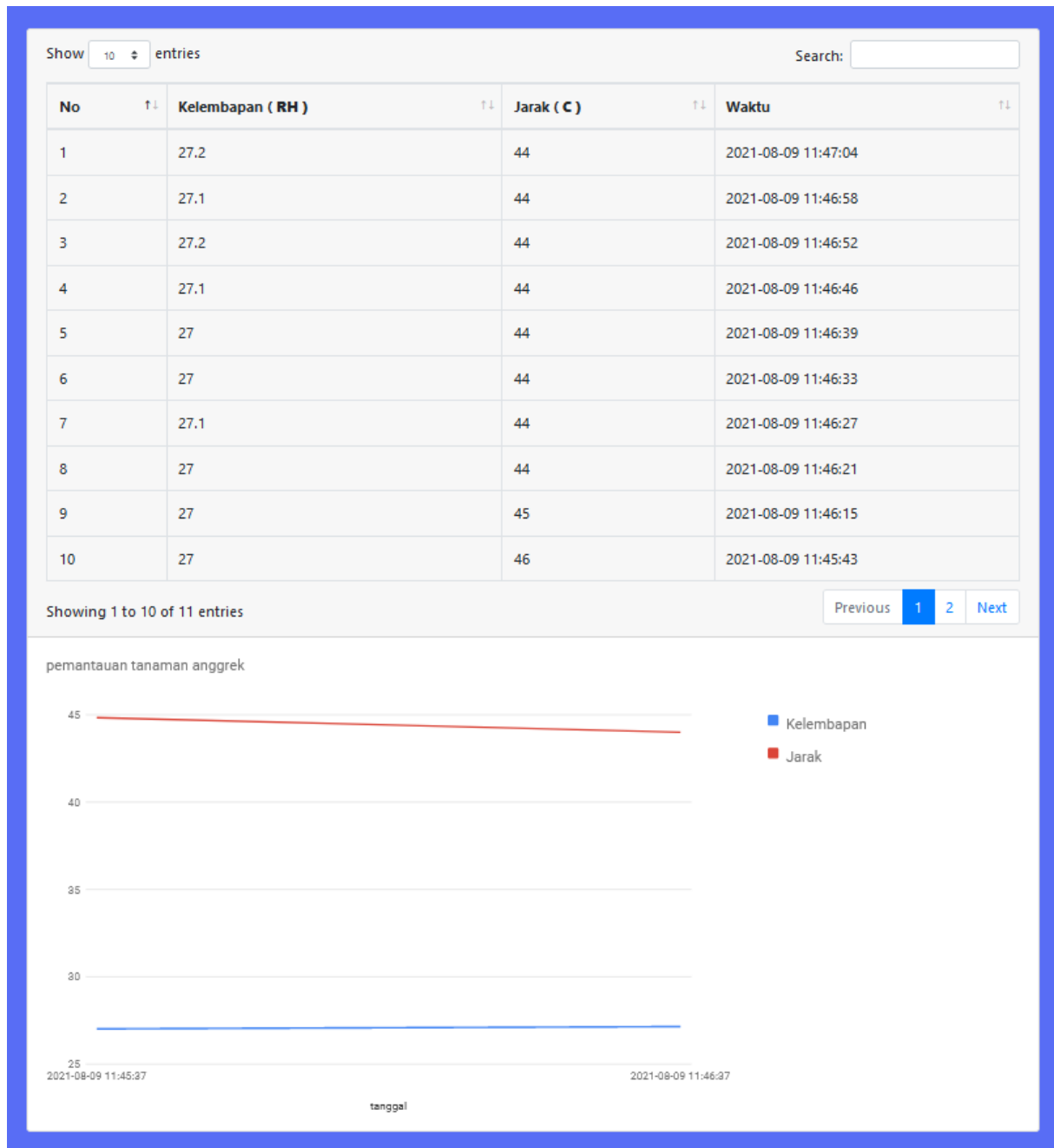
[Previous](#)
[1](#)
[Next](#)

Snippet

```
String str = "{id\":\"112:20:45062021\", \"sensor_value\"";
String comma = ",";
String str2 = "[" + String(Kelembapan) + comma + String(Jarak);
String str3 = "]";
String str4 = str + str2 + str3;
int httpResponseCode = http.POST(str4);
```

Gambar 60 Pengujian Edit Device Automatic Control (5)

Pada **Gambar 61** berikut dapat kita lihat hasil dari *agent* pemantauan tanaman anggrek yang sudah berhasil diedit yang menampilkan data nilai dari setiap sensor yang didapatkan dari lingkungan setiap 6 detik dan nilai rata-rata sensor yang ditampilkan dalam bentuk *chart*.



Gambar 61 Pengujian Edit Device Automatic Control (6)

4.2.1.6 Pengujian Fungsi *Delete Agent Tipe Automatic Control*

Pada pengujian fungsi *delete* kali ini akan dilakukan pada *agent* dengan nama “Pemantauan Sayur”, dimana pada halaman home bagian *Automatic Control*, *user* memilih aksi hapus dimana fungsi ini akan menghapus keseluruhan isi dari *device* tersebut termasuk sensor dan aktuator yang ada di dalam *device* tersebut.

Data Pengguna - Automatic Control						
Show 10 entries		Search:				
No	ID	Nama Device	IP Address	Jumlah Pin dan Aktuator	Aksi	
1	210:23:35072021	Pemantauan Sayur	192.168.17.128	4 Sensor 4 Aktuator	hapus	edit
2	112:20:45062021	Pemantauan Tanaman	172.27.42.51	3 Sensor 3 Aktuator	hapus	edit
Showing 1 to 2 of 2 entries					Previous	Next

Gambar 62 Pengujian Delete Device Automatic Control (1)

Berikut tampilan halaman *home* bagian *automatic control* setelah berhasil di hapus.

Data Pengguna - Automatic Control						
Show 10 entries		Search:				
No	ID	Nama Device	IP Address	Jumlah Pin dan Aktuator	Aksi	
1	112:20:45062021	Pemantauan Tanaman	172.27.42.51	3 Sensor 3 Aktuator	hapus	edit
Showing 1 to 1 of 1 entries					Previous	Next

Gambar 63 Pengujian Delete Device Automatic Control (2)

4.2.2 Pengujian Prototype Device IoT Tipe *Manual Control*

Pada saat melakukan pengujian pada pengontrolan lampu, developer menggunakan 3 led pada satu *device* IoT yang dikontrol secara manual melalui web aplikasi dengan menekan aksi *On/Off*. Pada saat pengguna melakukan kontroling lampu dengan sistem yang telah dibangun sesuai dengan skenario yang telah dijelaskan pada bab sebelumnya. Berikut hasil pengujian untuk melakukan kontroling lampu.

Tabel 3 Hasil Pengujian Kontroling Lampu

Tahapan Uji Coba	Hasil Uji Coba
Aksi ON/OFF lampu depan	Berhasil
Aksi ON/OFF lampu dapur	Berhasil
Aksi ON/OFF lampu kamar mandi	Berhasil

Tampilan web aplikasi ketika seluruh kondisi lampu mati.



Home Logout

Nama Agent: kontrol lampu
172.20.40.8

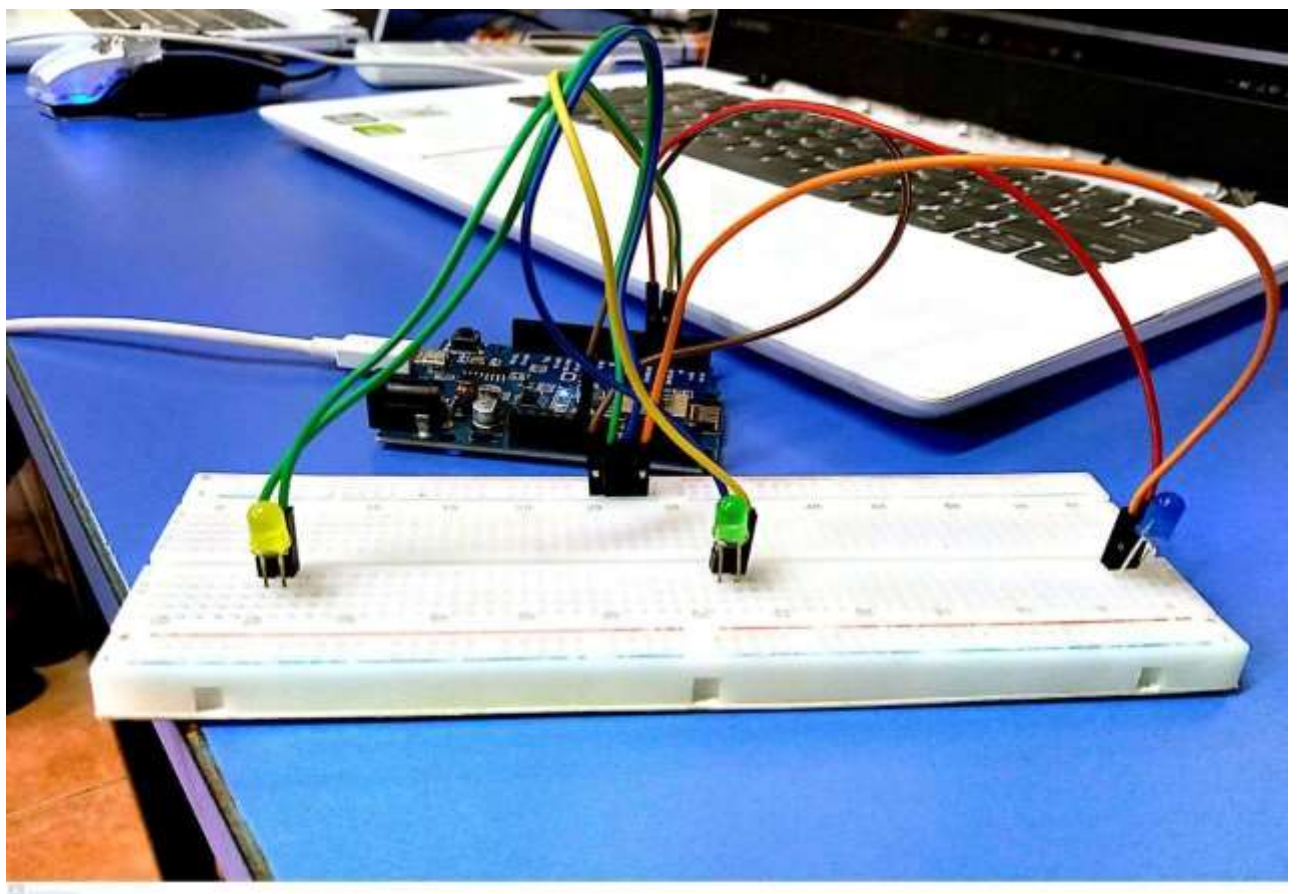
Show 10 entries Search:

No	Nama Alat	Status	Aksi
1	Lampu depan	mati	Hidup Mati
2	Lampu Dapur	mati	Hidup Mati
3	Lampu kamar mandi	mati	Hidup Mati

Showing 1 to 3 of 3 entries Previous 1 Next

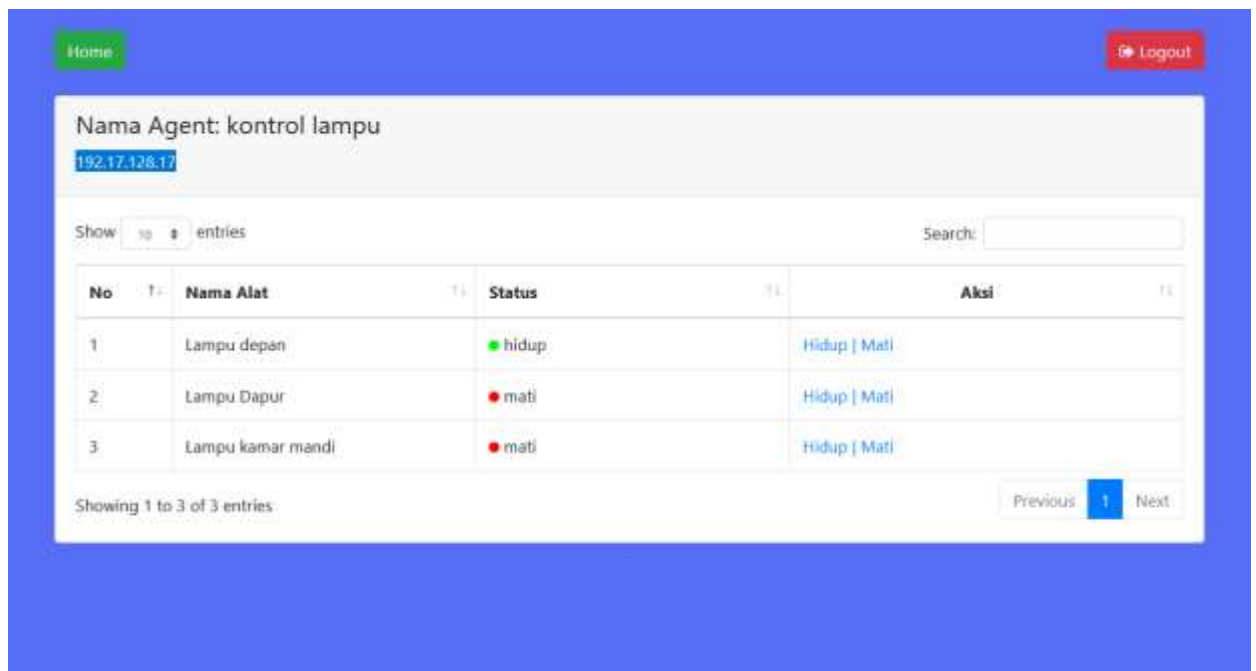
Gambar 64 Tampilan Halaman Kondisi Lampu OFF

Kondisi lampu yang masih dalam keadaan OFF dapat dilihat pada gambar dibawah.



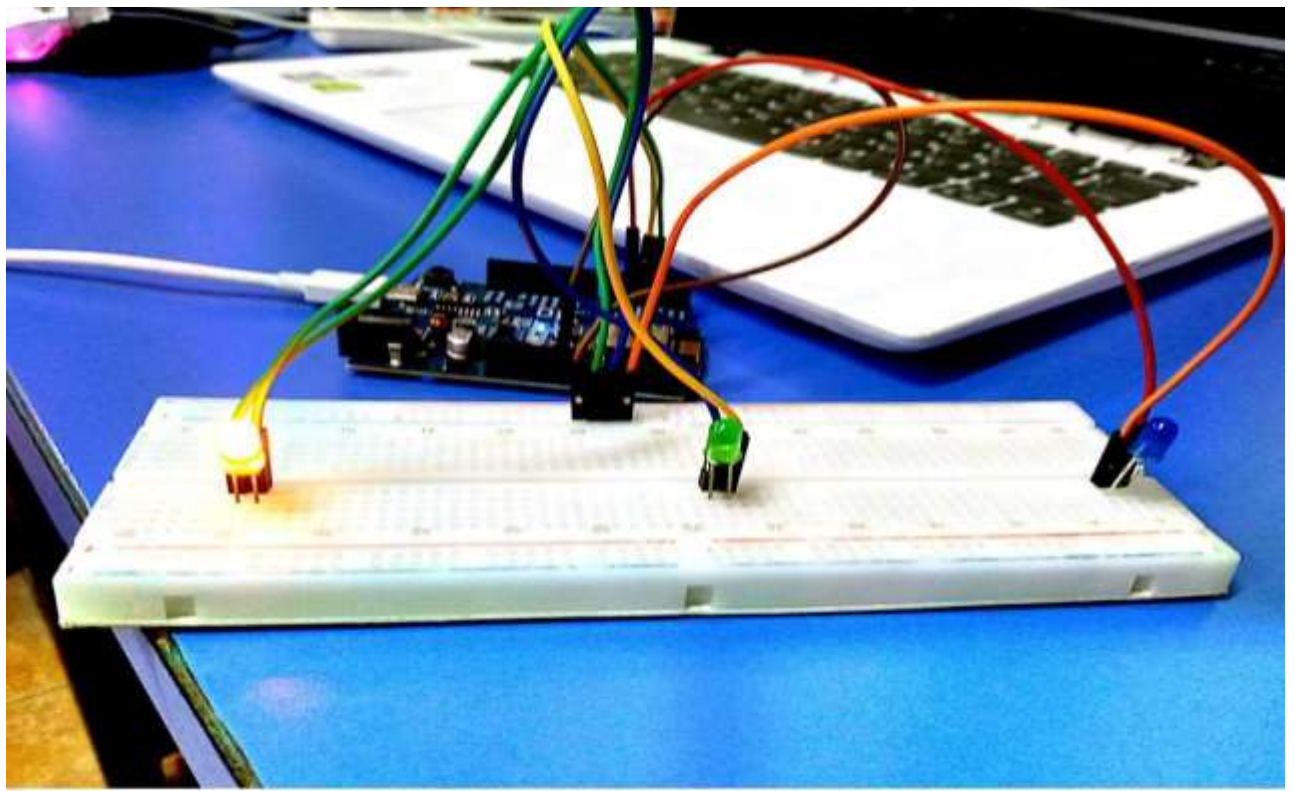
Gambar 65 Tampilan Prototype Kondisi Lampu OFF

User memberikan aksi ON pada lampu depan, berikut tampilan dari web aplikasinya.



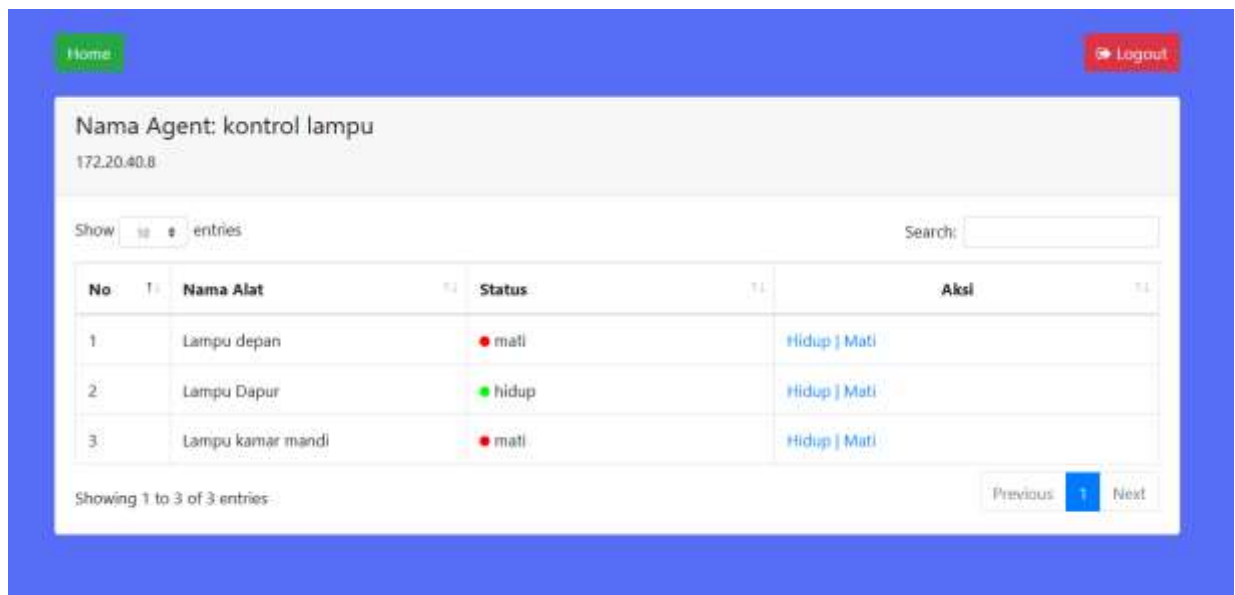
Gambar 66 Tampilan Web Kondisi Lampu Depan ON

Kondisi lampu depan menyala saat diberi aksi ON oleh user.



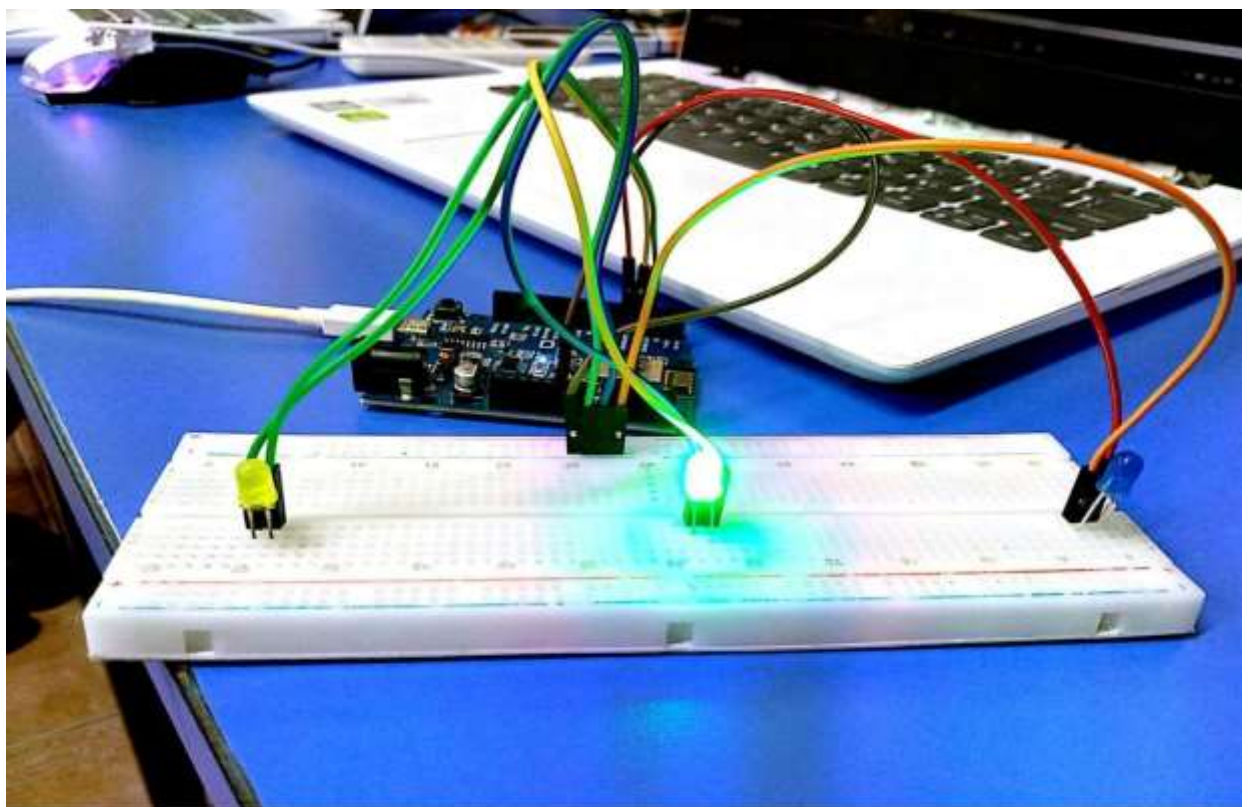
Gambar 67 Tampilan Prototype Kondisi Lampu Depan ON

User memberikan aksi ON pada lampu dapur, berikut tampilan dari web aplikasinya.



Gambar 68 Tampilan Web Kondisi Lampu Depan ON

Kondisi lampu dapur saat diberi aksi ON oleh user.



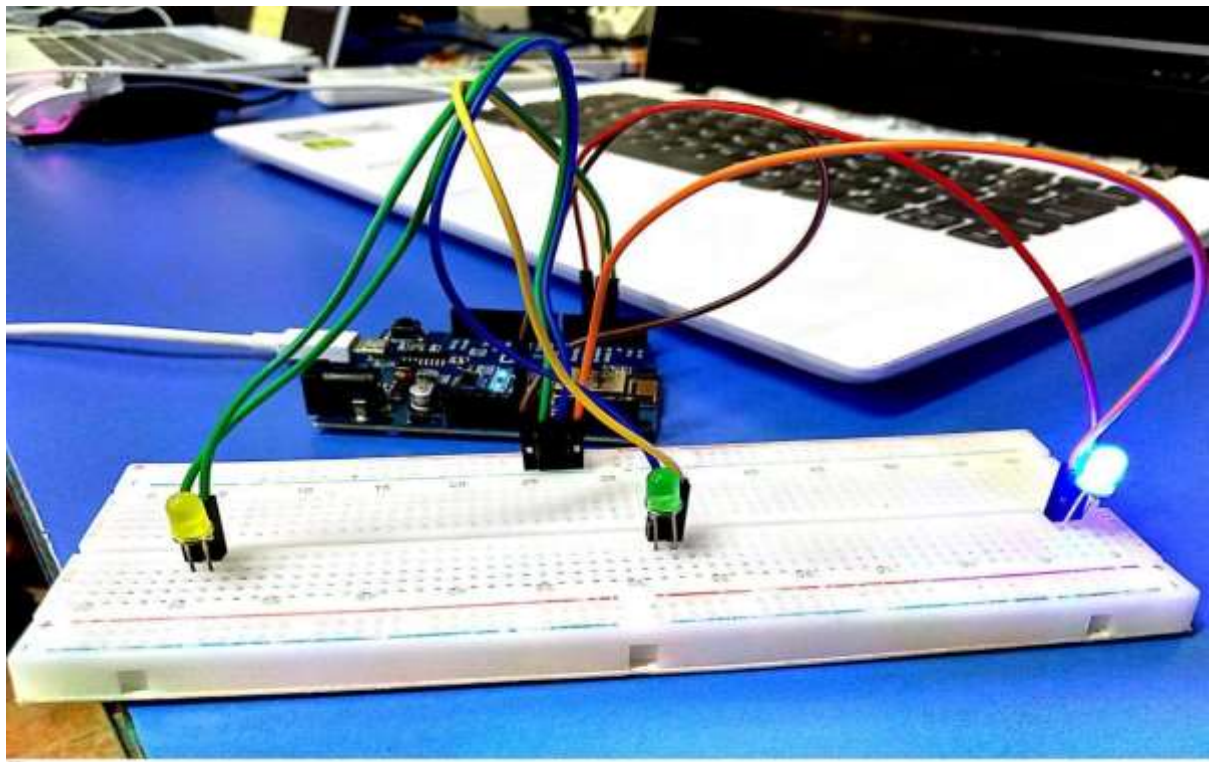
Gambar 69 Tampilan Prototype Kondisi Lampu Tengah ON

User memberikan aksi ON pada lampu kamar mandi, berikut tampilan dari web aplikasinya.



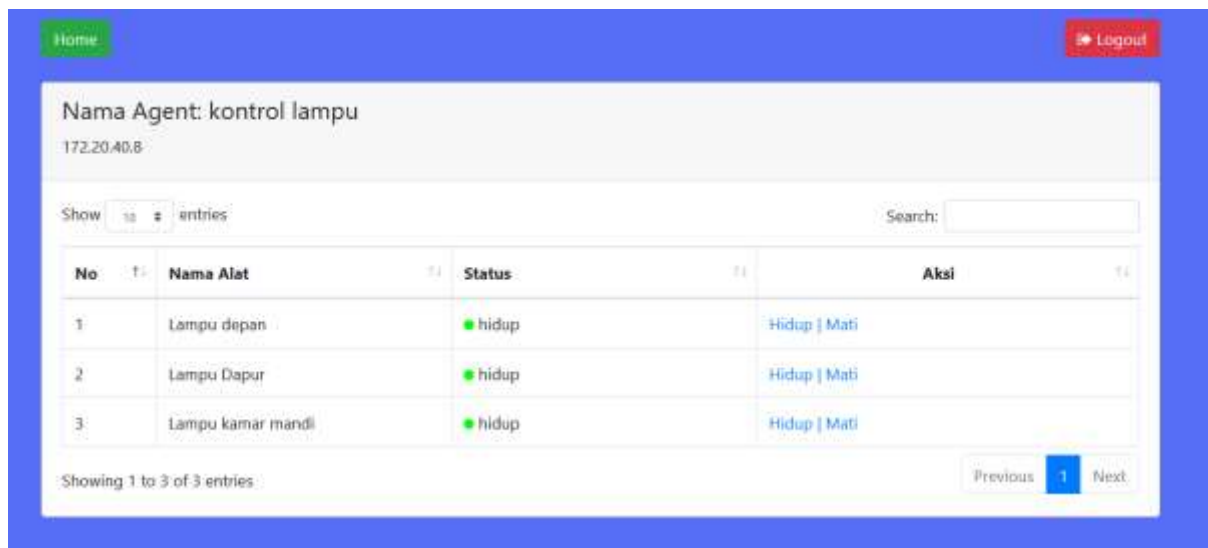
Gambar 70 Tampilan Web Kondisi Lampu Belakang ON

Kondisi dari lampu kamar mandi saat diberi aksi ON oleh user.



Gambar 71 Tampilan Prototype Kondisi Lampu Belakang ON

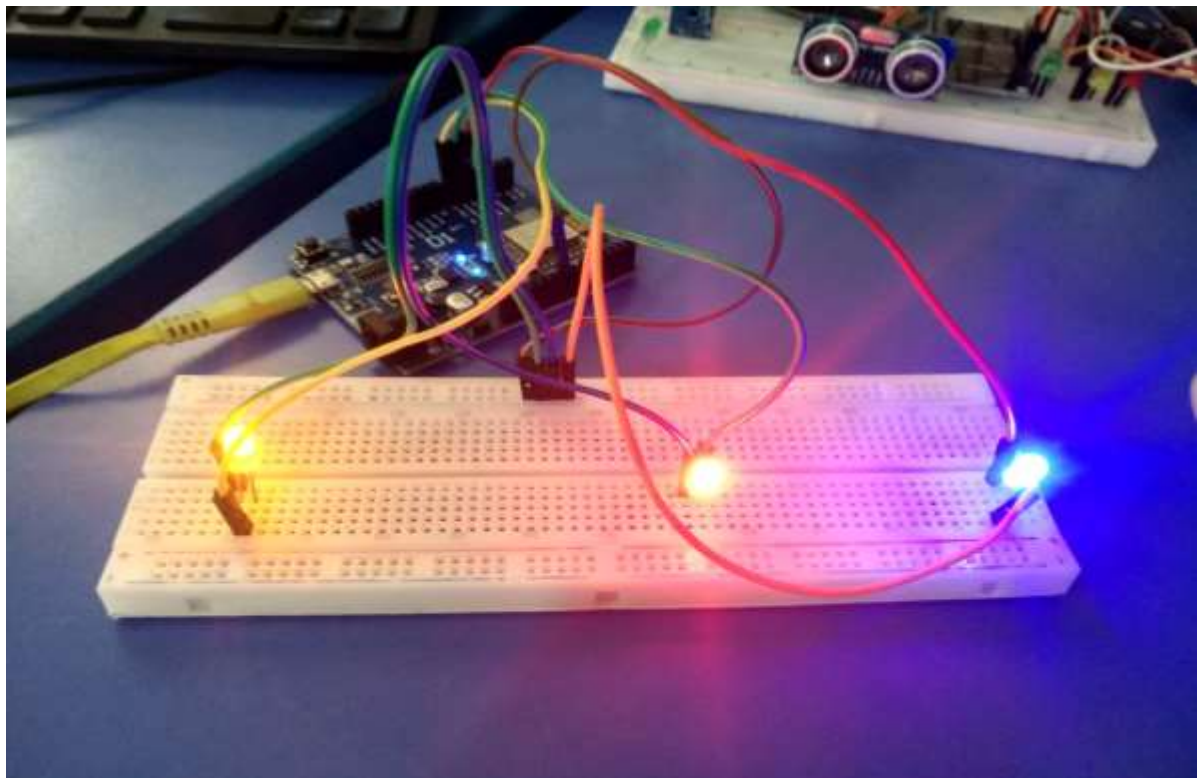
User memberi aksi ON pada semua lampu, berikut tampilan halaman dari *web* aplikasi.



No	Nama Alat	Status	Aksi
1	Lampu depan	hidup	Hidup Mati
2	Lampu Dapur	hidup	Hidup Mati
3	Lampu kamar mandi	hidup	Hidup Mati

Gambar 72 Tampilan Web Kondisi Semua Lampu ON

Kondisi semua lampu dalam keadaan ON dapat dilihat pada gambar berikut.



Gambar 73 Kondisi Prototype Semua Lampu ON

Dari skema pengujian diatas, fitur untuk mematikan dan menghidupkan lampu sudah bisa bekerja.

4.2.3 Pengujian Prototype Device IoT Tipe Automatic Control

Pengujian sistem ini untuk studi kasus pengontrolan dan monitoring tanaman, ini dilakukan terhadap sensor yang terhubung kedalam prototype.

4.2.3.1 Pengujian Temperatur Menggunakan Sensor DHT11

Dalam pengujian sensor DHT11, dilakukan berdasarkan skenario yang telah dijelaskan di sub bab sebelumnya. Pengujian sensor didasari dengan penggunaan library DHT11 untuk memastikan bahwa sensor yang digunakan berjalan dengan baik. Pin yang digunakan adalah D6. Pada proses ini sensor akan mendeteksi suhu

4.2.3.2 Hasil Pengujian Temperatur Menggunakan Sensor DHT11

Hasil pengukuran yang didapatkan setelah sensor DHT11 diuji maka akan ditampilkan pada tabel berikut.

Tabel 4 Hasil Pengujian Sensor DHT11

Suhu (°C)
44
44
44
44
44
44
45
46

4.2.3.3 Pengujian Kelembaban Menggunakan Sensor DHT11

Dalam pengujian sensor kelembaban tanah yang digunakan , dilakukan sebuah skenario dimana sensor akan ditanamkan ke dalam tanah yang kering dan basah. Hasil yang ingin didapatkan dari pengujian ini adalah sensor dht11 dapat mengukur pH dari tanah, dan Wemos D1 R1 dapat mengolah dan menampilkan datanya kepada pengguna.

4.2.3.4 Hasil Pengujian Kelembaban Menggunakan DHT11

Hasil pengukuran yang didapatkan setelah sensor dht11 diuji maka akan ditampilkan pada tabel berikut. Semakin kecil nilai yang didapatkan oleh sensor, maka kelembaban tanah semakin kering. Jika nilainya besar, maka tanah akan semakin basah.

Tabel 5 Hasil Pengujian Sensor DHT11

Kelembapan Tanah (%)
27.2
27.1
27.2
27.1
27
27

4.2.3.5 Pengujian Jarak Menggunakan Sensor Ultrasonic

Dalam pengujian sensor ultrasonic yang digunakan, dilakukan sebuah skenario yang diukur dengan mendekatkan sebuah bidang datar, maka sensor akan membaca jarak dari sensor ultrasonic dengan bidang datar yang didekatkan. Hasil yang ingin didapatkan dari pengujian ini adalah sensor ultrasonic dapat mengukur jarak dari sensor dengan objek yang berada di depannya, dan Wemos D1 R1 Dapat mengolah dan menampilkan datanya kepada pengguna.

4.2.3.6 Hasil Pengujian Jarak Menggunakan Sensor Ultrasonic

Hasil pengukuran yang didapatkan setelah sensor ultrasonic diuji maka akan ditampilkan pada tabel berikut. Keluaran dari sensor ultrasonic adalah jarak antara sensor dengan benda yang berada di depannya.

Tabel 6 Hasil Pengujian Sensor Ultrasonik

Jarak yang ditampilkan Sensor Ultrasonic (cm)
2
5
15
15
4
5
6
8

4.2.3.7 Pengujian Intensitas Cahaya Menggunakan Sensor LDR

Dalam pengujian sensor LDR yang digunakan, dilakukan sebuah skenario yang diukur dengan menutup sensor dan menerangi sensor dengan senter. Hasil yang ingin didapatkan dengan pengujian ini adalah apakah sensor LDR dapat mengukur intensitas cahaya dan Wemos D1 R1 dapat melakukan pengolahan data terhadap data yang diterima.

4.2.3.8 Hasil Pengujian Intensitas Cahaya Menggunakan Sensor LDR

Hasil pengukuran yang didapatkan setelah sensor LDR diuji maka akan ditampilkan pada tabel berikut. Semakin besar nilai yang ditampilkan, maka keadaan cahaya yang berada di sekitar sensor LDR dalam kondisi terang. Sebaliknya, jika nilai yang didapat semakin kecil, maka keadaan cahaya di sekitar sensor dalam kondisi gelap.

Tabel 7 Hasil Pengujian Sensor LDR

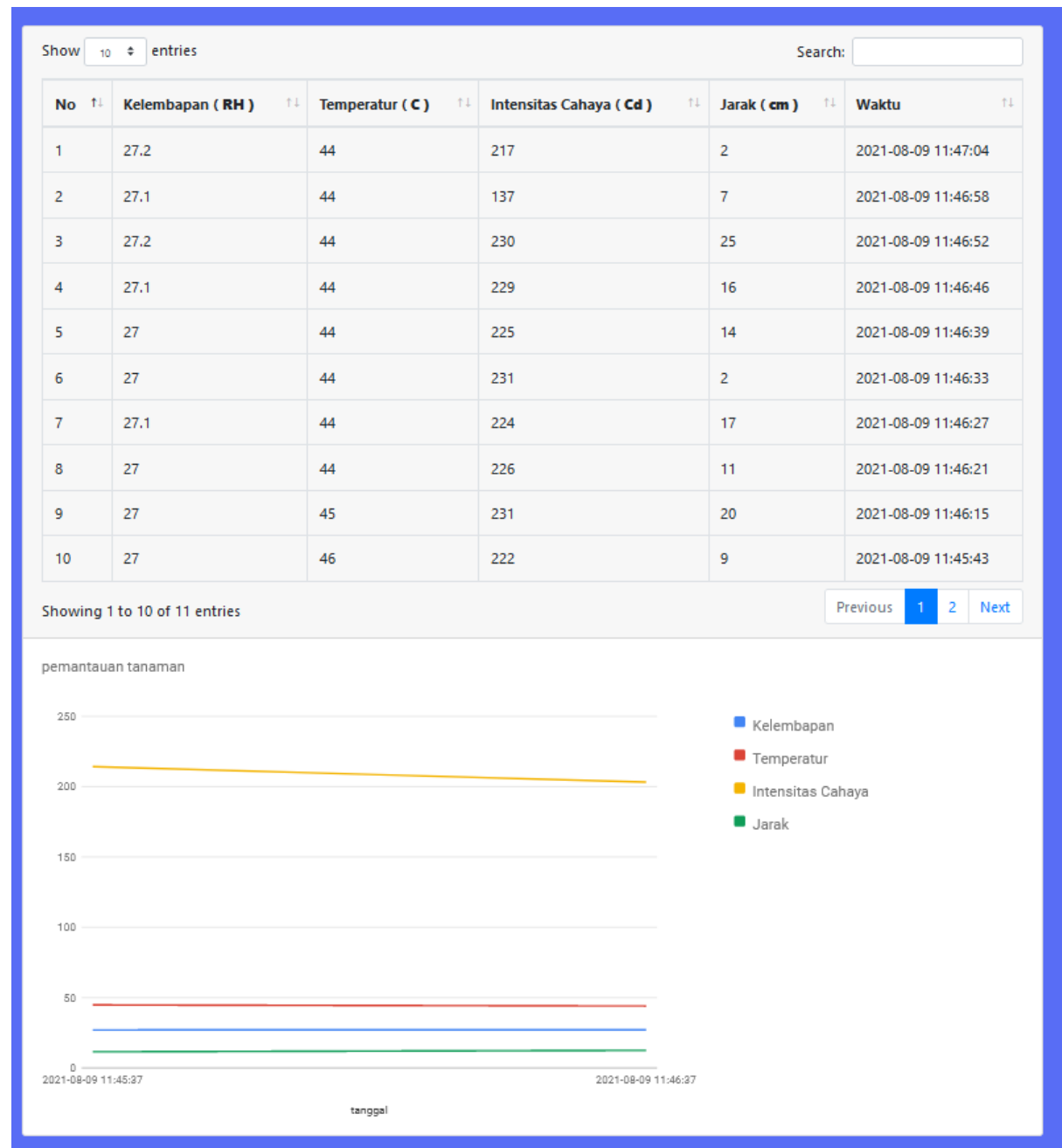
Intensitas Cahaya yang ditampilkan Sensor LDR (Cd)
120
100
108
29
98
102
103
110

4.2.3.9 Pengujian Monitoring Tanaman/Sayuran

Pada saat melakukan pengujian pada monitoring tanaman dan sayuran, developer menggunakan 4 sensor pada satu *device* yang dikontrol secara otomatis. Pada saat pengguna melakukan monitoring setiap sensor dengan sistem yang telah dibangun sesuai dengan skenario yang telah dijelaskan pada bab sebelumnya. Maka ketika kita mengklik nama *device* yaitu Pemantauan Tanaman maka akan menampilkan data dari wemos yang disimpan di dalam mongoDB. Data disajikan dalam dua bentuk, yaitu dengan bentuk tabel dan grafik. Tabel menampilkan semua data yang berada di dalam basis data, mulai dari waktu ditambahkan dan nilai setiap sensor. Bentuk kedua adalah dalam bentuk grafik,

dalam bentuk grafik, tidak semua data ditampilkan, melainkan rata-rata nilai tiap sensor setiap 5 menit.

Berikut hasil pengujian saat melakukan Monitoring pemantauan tanaman dengan menggunakan 4 sensor.



Gambar 74 Pengujian Monitoring Device Automatic Control

4.2.4 Pengujian Komunikasi *Agent* ke *API Gateway* Menuju Database

Pada gambar dibawah ini menunjukkan bahwa *agent* mengirimkan 7 data sensor. Maka data dari sensor tersebut akan dilakukan beberapa pengecekan di dalam *API Gateway* kemudian akan disimpan di database.

```
{ "id": "112:20:45062021", "sensor_value": [5, 92, 53.00] }  
HTTP Response code: 200  
{ "id": "112:20:45062021", "sensor_value": [5, 91, 53.00] }  
HTTP Response code: 200  
{ "id": "112:20:45062021", "sensor_value": [5, 90, 53.00] }  
HTTP Response code: 200  
{ "id": "112:20:45062021", "sensor_value": [5, 32, 53.00] }  
HTTP Response code: 200  
{ "id": "112:20:45062021", "sensor_value": [5, 32, 53.00] }  
HTTP Response code: 200  
{ "id": "112:20:45062021", "sensor_value": [5, 116, 53.00] }  
HTTP Response code: 200  
{ "id": "112:20:45062021", "sensor_value": [5, 117, 53.00] }  
HTTP Response code: 200
```

Gambar 75 Komunikasi Agent Ke API Gateway

Data yang sudah berhasil disimpan ke dalam database dapat dilihat seperti pada **Gambar 76** berikut, yang dimana data dari setiap sensor diterakan tanggal dan waktu yang ditambahkan oleh *API Gateway*. Adapun juga *delay* yang dibutuhkan dimulai dari pengiriman data, pengecekan data di *API Gateway* dan menyimpan data ke dalam database adalah sekitar 10-20ms.



Gambar 76 Data yang Tersimpan dalam Database

Berikut pemberitahuan status ketika data dari *agent* berhasil disimpan ke dalam database yang akan ditampilkan di MongoDB.



Gambar 77 Status Berhasil Penyimpanan Data

4.2.4 Pengujian Database ke *User Interface*

Pada subbab ini dijelaskan proses testing yang terjadi dari database menuju aplikasi web.

1. Test Koneksi Database ke UI

Pada gambar dibawah ini menunjukkan dengan menggunakan kode fungsi koneksi dari database ke aplikasi *web* telah berhasil.

```
<?php
    require 'panggil.php';
    include_once '../conf.php';
    if ($con->TugasAkhir) {
        echo "koneksi database dan web berhasil";
    }
?>
```

Pada gambar dibawah ini menampilkan status ketika database berhasil terkoneksi dengan aplikasi *web* yang akan ditampilkan pada aplikasi *web*.



Gambar 78 Pemberitahuan Database dan Web Berhasil Terkoneksi

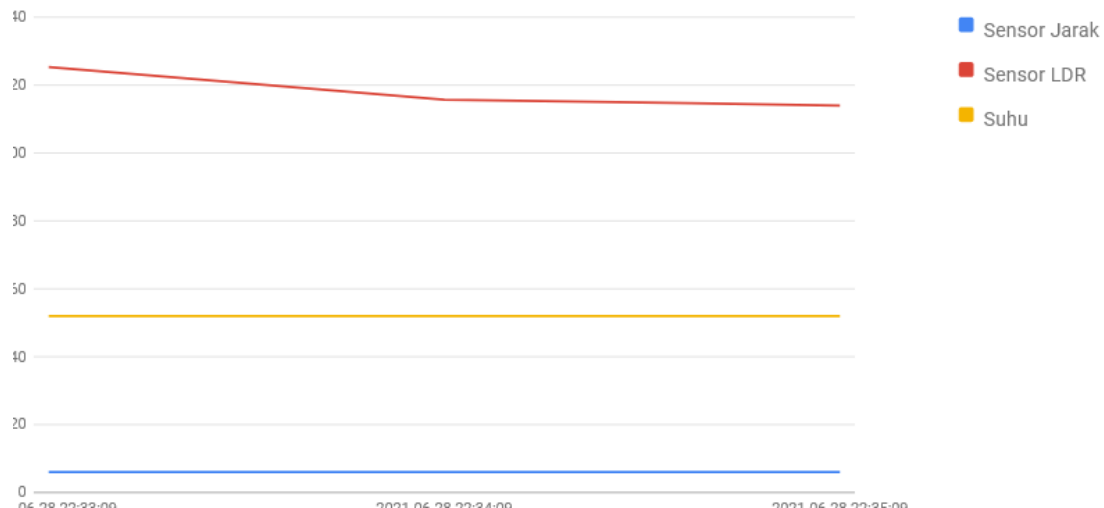
2. Testing Fungsi Rata-Rata

Pada **Gambar 79** dibawah diterakan testing untuk menampilkan rata-rata nilai dari setiap sensor dan rata-rata tersebut berhasil di tampilkan pada aplikasi *web*. Seperti yang diterakan pada gambar dibawah terdapat 30 data dari pengguna.



Gambar 79 Data Sensor Dari Pengguna

Kemudian rata-rata tersebut ditampilkan dalam bentuk *chart* pada aplikasi *web* seperti yang bisa dilihat pada **Gambar 80** berikut.



Gambar 80 Rata-rata nilai dari setiap sensor

4.2.5 Pengujian Waktu

1. Pada saat melakukan pengujian terhadap *device* iot sebelum adanya system yang dibangun atau ketika menggunakan sistem tradisional , sewaktu melakukan flash data waktu yang dibutuhkan yaitu sekitar 35 detik dan untuk melakukan penggantian code sekitar 15 detik, disini sudah termasuk untuk menambahkan *agent* dikarenakan pada sistem tradisional saat menambahkan *agent* perangkatnya yang langsung ditambahkan melalui kode program, maka berdasarkan itu didapat waktu yang digunakan untuk melakukan sekali perubahan *device* iot dan perubahan terhadap sensor adalah

$$\begin{aligned}
 &50 \text{ detik} + 1 * 50 \text{ detik} \\
 &= 100 \text{ detik} \\
 &= 1 \text{ menit } 40 \text{ detik}
 \end{aligned}$$

Kemudian ketika terjadi perubahan data sebanyak 10 kali , maka kita asumsikan bahwa n adalah banyaknya perubahan data. Maka waktu yang di dapat ketika melakukan 10 kali perubahan adalah:

$$\begin{aligned}
 &50 \text{ detik} + n * 50 \text{ detik} \\
 &= 50 \text{ detik} + 10 * 50 \text{ detik} \\
 &= 550 \text{ detik} \\
 &= 9 \text{ menit } 10 \text{ detik}
 \end{aligned}$$

Kemudian ketika terjadi perubahan data sebanyak 15 kali , maka waktu yang diperlukan untuk melakukan perubahan data adalah

$$\begin{aligned} & 50 \text{ detik} + n * 50 \text{ detik} \\ & = 50 \text{ detik} + 15 * 50 \text{ detik} \\ & = 800 \text{ detik} \\ & = 13 \text{ menit } 20 \text{ detik} \end{aligned}$$

2. Pada saat melakukan pengujian terhadap *device* iot setelah menggunakan sistem yang dibangun memerlukan waktu sekitar 2 menit 6 detik untuk mendapatkan alamat IP dan melakukan *flash* atau mengupload code, disini sudah termasuk saat menambahkan *agent* melalui aplikasi *web* dan menambahkan sensor. Waktu yang dibutuhkan untuk melakukan penggantian ataupun perubahan adalah 5 detik . maka berdasarkan itu didapat waktu yang digunakan untuk melakukan sekali pergantian *device* iot dan perubahan terhadap sensor adalah :

$$\begin{aligned} & 2 \text{ menit } 6 \text{ detik} + 1 * 5 \text{ detik} \\ & = 2 \text{ menit } 11 \text{ detik} \end{aligned}$$

Kemudian ketika terjadi perubahan data sebanyak 10 kali , maka kita asumsikan bahwa n adalah banyaknya perubahan data. Maka waktu yang di dapat ketika melakukan 10 kali perubahan data adalah :

$$\begin{aligned} & 2 \text{ menit } 6 \text{ detik} + 10 * 5 \text{ detik} \\ & = 2 \text{ menit } 6 \text{ detik} + 50 \text{ detik} \\ & = 2 \text{ menit } 56 \text{ detik} \end{aligned}$$

Kemudian ketika terjadi perubahan data sebanyak 15 kali , maka waktu yang diperlukan untuk melakukan perubahan data adalah :

$$\begin{aligned} & 2 \text{ menit } 6 \text{ detik} + 15 * 5 \text{ detik} \\ & = 2 \text{ menit } 6 \text{ detik} + 75 \text{ detik} \\ & = 3 \text{ menit } 21 \text{ detik} \end{aligned}$$

4.2.6 Butir Uji Coba

Pada sub bab ini akan dijelaskan butir uji coba terhadap setiap fungsi yang ada pada sistem.

1. Butir Uji Fungsi Kirim Data dari Http *Body* ke API

Pada fungsi ini akan dilakukan pengambilan data dari lingkungan melalui setiap sensor yang ada pada *agent* yang akan diproses di dalam API *gateway*.

Tabel 8 Butir Uji Fungsi Kirim Data dari Http Body ke API

No. Fungsi	3.5.1		
Nama Butir Uji	Fungsi kirim data dari http <i>body</i> ke API		
Tujuan	Uji coba untuk mengambil data melalui sensor dari lingkungan.		
Deskripsi	Mengirim data dari http <i>body</i> ke API		
Kondisi Awal	-		
Skenario Pengujian			
Client mengambil data melalui sensor dari lingkungan, kemudian data yang diambil berbentuk JSON dan dikirimkan melalui <i>body</i> HTTP <i>request</i> melalui port 80 ke <i>API Gateway</i> .			
Kriteria Evaluasi Hasil			
Data yang diterima oleh <i>API gateway</i> akan diproses di dalam <i>API Gateway</i> dan ketika data tersebut sudah selesai diproses, maka <i>client</i> terus mengulang proses dari awal.			
Kasus dan Hasil Pengujian			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Data sensor dari lingkungan	Data diproses di <i>API gateway</i> jika <i>client</i> melakukan proses yang sama dan data akan berhenti jika <i>client</i> sudah tidak ada lagi	Data diproses di <i>API gateway</i> jika <i>client</i> melakukan proses yang sama dan data akan berhenti jika <i>client</i> sudah tidak ada lagi	Diterima
Catatan			
-			

2. Butir Uji Fungsi Translasi JSON ke dalam *Query* MongoDB

Pada fungsi ini akan dilakukan translasi JSON ke dalam *query* mongodb, dimana data yang sudah berhasil ditranslasikan akan menghasilkan bentuk *array* dan kemudian akan diproses oleh Bahasa pemrograman.

Tabel 9 Butir Uji Fungsi Translasi JSON ke dalam Query MongoDB

No. Fungsi	3.5.3		
Nama Butir Uji	Fungsi Translasi JSON ke dalam <i>Query</i> MongoDB		
Tujuan	Menstranlasikan data yang diterima dari <i>client</i> melalui HTTP <i>body</i> ke dalam <i>query</i> MongoDB.		
Deskripsi	Translasi JSON ke dalam <i>query</i> mongoDB		
Kondisi Awal	-		
Skenario Pengujian			
Mentranslasikan data yang diterima oleh <i>client</i> dari http <i>body</i> ke dalam <i>query</i> mongoDB.			
Kriteria Evaluasi Hasil			
Data yang sudah selesai ditranslasikan akan menghasilkan bentuk <i>array</i> supaya dapat diproses oleh bahasa php.			
Kasus dan Hasil Pengujian			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Ketika API <i>gateway</i> menerima <i>request</i> maka data JSON akan ditranslasikan.	Data yang sudah dalam bentuk <i>array</i> dapat diproses oleh Bahasa php, dan data tidak akan hilang selama <i>client</i> masih ada.	Data yang sudah dalam bentuk <i>array</i> dapat diproses oleh Bahasa php, dan data tidak akan hilang selama <i>client</i> masih ada.	Diterima
Catatan			
-			

3. Butir Uji Fungsi Membuat URL untuk Mengontrol Agent

Pada fungsi ini akan membuat url untuk mengontrol *agent*, yang didapat dari perbandingan nilai yang digunakan untuk mengontrol aktuator.

Tabel 10 Butir Uji Fungsi Membuat URL untuk Mengontrol Agent

No. Fungsi	3.5.5
Nama Butir Uji	Fungsi membuat url untuk mengontrol <i>agent</i>
Tujuan	Mendapatkan url yang digunakan untuk mengontrol aktuator
Deskripsi	Membuat URL untuk Mengontrol <i>Agent</i>
Kondisi Awal	-
Skenario Pengujian	
API <i>gateway</i> melakukan perbandingan dari antara dua nilai, yaitu nilai yang sudah ditentukan untuk menjadi acuan basis data dan nilai yang diterima dari <i>client</i> , kemudian dari nilai perbandingan tersebut akan membuat url untuk mengontrol aktuator.	
Kriteria Evaluasi Hasil	

Setelah melakukan perbandingan maka API <i>gateway</i> akan membuat url , url ini kemudian akan di curl menggunakan bahasa pemrograman PHP yang berguna sebagai pengontrolan.			
Kasus dan Hasil Pengujian			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Api <i>gateway</i> membuat perbandingan antara nilai acuan dan nilai yang diterima oleh <i>client</i> .	Api <i>gateway</i> akan membuat url kemudian akan di curl menggunakan Bahasa pemograman PHP yang digunakan sebagai pengontrolan.	Api <i>gateway</i> akan membuat url kemudian akan di curl menggunakan Bahasa pemograman PHP yang digunakan sebagai pengontrolan.	Diterima
Catatan			
-			

4. Butir Uji Fungsi *Update* Data ke MongoDB

Pada fungsi ini akan melakukan *update* data ke mongodb, dimana mongodb akan menjalankan *script* agar dapat melakukan *update* data ke mongodb.

Tabel 11 Butir Uji Fungsi *Update* Data ke MongoDB

No. Fungsi	3.5.7		
Nama Butir Uji	Fungsi <i>update</i> data ke mongodb		
Tujuan	Melakukan update data ke mongodb		
Deskripsi	Update Data ke MongoDB		
Kondisi Awal	-		
Skenario Pengujian			
Setelah data ditranslasikan, mongodb akan menjalankan <i>script</i> -nya supaya dapat melakukan <i>update</i> ke mongodb.			
Kriteria Evaluasi Hasil			
Setelah data yang ditranslasikan menjadi <i>array</i> , maka data tersebut akan dieksekusi dengan <i>script</i> bahasa pemograman sehingga bisa di <i>update</i> ke mongodb.			
Kasus dan Hasil Pengujian			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Data JSON yang sudah ditranslasikan ke dalam bentuk <i>array</i> .	Data <i>array</i> tersebut dieksekusi dengan <i>script</i> kemudian di <i>update</i> di mongodb.	Data <i>array</i> tersebut dieksekusi dengan <i>script</i> kemudian di <i>update</i> di mongodb.	Diterima
Catatan			
-			

5. Butir Uji Fungsi CRUD Data ke MongoDB

Pada fungsi ini *user* akan melakukan CRUD atau *create*, *read*, *update* dan *delete* data *agent* melalui aplikasi *web* sesuai dengan instruksi yang sudah dijelaskan. Kemudian data *agent* tersebut akan disimpan di dalam mongodb dan ditampilkan pada aplikasi *web*.

Tabel 12 Butir Uji Fungsi CRUD Data ke MongoDB

No. Fungsi	3.5.9		
Nama Butir Uji	Fungsi CRUD (<i>create, update, read and delete</i>) data <i>agent</i> dan disimpan ke mongodb		
Tujuan	Menyimpan data yang di CRUD ke MongoDB		
Deskripsi	Melakukan CRUD terhadap data <i>agent</i> kemudian disimpan ke MongoDB.		
Kondisi Awal	-		
Skenario Pengujian			
Pengguna melakukan CRUD terhadap data <i>agent</i> melalui aplikasi <i>web</i> .			
Kriteria Evaluasi Hasil			
Pengguna melakukan CRUD terhadap data <i>agent</i> melalui aplikasi <i>web</i> , kemudian data tersebut disimpan ke dalam mongodb (database).			
Kasus dan Hasil Pengujian			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Data <i>agent</i>	Data disimpan dalam mongodb.	Data tersimpan dalam mongodb.	Diterima
Catatan			
-			

6. Butir Uji Fungsi Terima Request

Pada fungsi ini API *gateway* menerima *request* dari *client* dan melakukan perbandingan data yang diterima dengan menjadikannya nilai acuan dan data tersebut akan disimpan di dalam mongodb atau database.

Tabel 13 Butir Uji Fungsi Terima Request

No. Fungsi	3.5.11		
Nama Butir Uji	Fungsi terima <i>request</i>		
Tujuan	API <i>gateway</i> menerima request dan melakukan perbandingan data yang diterima dengan data dijadikan acuan.		
Deskripsi	API <i>gateway</i> Menerima <i>Request</i> dari <i>Agent</i>		
Kondisi Awal	-		
Skenario Pengujian			
Agent akan melakukan <i>listen</i> terhadap HTTP <i>request</i> port 80 agar API Gateway dapat menerima request dan melakukan perbandingan data yang diterima dengan data yang menjadi acuan. Kemudian API <i>gateway</i> melakukan encode ke bentuk php, jumlah data yang diterima sama dengan jumlah sensor yang ditambahkan oleh <i>user</i> , kemudian API <i>gateway</i> akan melakukan pengecekan, setelah pengecekan berhasil API <i>gateway</i> akan memasukkan waktu dan tanggal kedalam <i>array</i> data tersebut dan data tersebut disimpan ke dalam database.			
Kriteria Evaluasi Hasil			
Data berhasil dimasukkan data, maka API Gateway akan melakukan perbandingan untuk membuat <i>url</i> terhadap data yang diterima dan data yang ada di dalam basis data, dengan acuan operator perbandingan yang dimasukkan oleh pengguna di dalam basis data.			
Kasus dan Hasil Pengujian			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Data <i>agent</i>	Data disimpan dalam <i>mongodb</i> dan dilakukan perbandingan untuk membuat <i>url</i> .	Data tersimpan dalam <i>mongodb</i> dan dilakukan perbandingan untuk membuat <i>url</i> .	Diterima
Catatan			
-			

7. Butir Uji Fungsi Rata-Rata

Pada fungsi ini akan menampilkan nilai rata-rata dari setiap sensor pada aplikasi *web* dalam bentuk *chart*.

Tabel 14 Butir Uji Fungsi Rata-rata

No. Fungsi	3.5.13		
Nama Butir Uji	Fungsi Rata-rata		
Tujuan	Menampilkan hasil rata-rata nilai dari setiap sensor pada <i>agent</i> pengolahan sensor di dalam <i>web</i> aplikasi dalam bentuk <i>chart</i> .		
Deskripsi	Menampilkan hasil rata-rata dari sensor pada aplikasi <i>web</i> .		
Kondisi Awal	-		
Skenario Pengujian			
Web mengambil data dari database, kemudian data tersebut dihitung rata-ratanya kemudian data tersebut ditampilkan di <i>web</i> aplikasi.			
Kriteria Evaluasi Hasil			
Web mengambil data dari database, kemudian data tersebut dihitung rata-ratanya kemudian data tersebut ditampilkan di <i>web</i> aplikasi.			
Kasus dan Hasil Pengujian			
Data Masukan	Yang diharapkan	Pengamatan	Kesimpulan
Data <i>agent</i>	Hasil rata-rata yang sudah dihitung tertampil di <i>web</i> aplikasi dalam bentuk <i>chart</i> .	Hasil rata-rata yang sudah dihitung tertampil di <i>web</i> aplikasi dalam bentuk <i>chart</i> .	Diterima
Catatan			
-			

BAB V

KESIMPULAN DAN SARAN

Pada bab ini menguraikan kesimpulan tugas akhir yang telah dikerjakan serta saran yang diperlukan untuk pembangunan sistem yang sama pada pengembangan selanjutnya.

5.1 Kesimpulan

Setelah melakukan implementasi *IoT Centralized Control and Monitoring System* diperoleh kesimpulan yaitu:

1. Dengan menggunakan *IoT Centralized Control and Monitoring System* dapat membantu pengguna dalam memonitoring beberapa *agent* secara terpusat melalui aplikasi *master*.
2. Sistem yang dibangun terbagi atas dua jenis pengontrolan yakni, pengontrolan secara otomatis dan pengontrolan secara manual.
3. Data yang diperoleh dari *agent* tersimpan ke dalam database setelah dimodifikasi oleh *API Gateway*.
4. Dengan menggunakan *web* aplikasi sebagai antar muka pengguna, pada *web* tersebut pengguna dapat melakukan CRUD terhadap data *agent*.
5. Dengan menggunakan aplikasi *web* pengguna dapat mengontrol dan memantau *agent*.
6. Aplikasi *web* yang diharapkan dapat menampilkan nilai dan hasil rata-rata dari setiap nilai sensor yang diperoleh dalam bentuk *chart*.

5.2 Saran

Adapun saran yang diberikan dalam pengembangan selanjutnya adalah sebagai berikut :

1. Sistem yang dibangun dapat dengan mudah digunakan oleh pengguna dengan memberikan panduan yang dapat digunakan pengguna sebagai referensi dalam mengelola *web* aplikasi.
2. Supaya pengguna mengetahui status dari *aktuator*, diharapkan untuk pengembang selanjutnya membuat fitur status pada *web* aplikasi untuk memastikan aktuatornya *active/non-active*, karena pada sistem yang dibangun saat ini untuk memastikan aktuator aktif atau tidak aktif penulis masih menggunakan LED yang harus dilihat secara langsung perangkat kerasnya bahwa jika LED menyala artinya aktuator sedang aktif begitu pula sebaliknya.

3. Sistem yang dibangun saat ini hanya menyertakan dua studi kasus yaitu pemantauan tanaman dan pengontrolan lampu, untuk kedepannya di harapkan untuk mencoba studi kasus berbeda yang dapat dimonitoring melalui aplikasi *web*.
4. Supaya sistem ini dapat diimplementasikan secara langsung oleh pengguna, disarankan pada pengembang selanjutnya mengimplementasikan keseluruhan perangkat dan menyesuaikan terhadap lingkungan dikarenakan sistem yang sekarang perangkat yang digunakan hanya sebagai mediator.

Daftar Pustaka

- [1] A. N. Baharsyah, "Internet Of Things," 2019.
- [2] Y. Setiawan, H. Tanudjaja and S. Octaviani, "Penggunaan Internet Of Things untuk Pemantauan dan Pengendalian Sistem Hidroponik," *TESLA*, vol. 20, 2018.
- [3] D. Prihatmoko, "Penerapan *Internet Of things* (IoT) Dalam Pembelajaran di UNISNU Jepara", vol. 7, 2016
- [4] S. Hossain, A. Jahid and E. Haque, "A smart IoT Based System for Monitoring and Controlling the Substation Equipment," 2019.
- [5] I. Alfannizar and Y. Rahayu, "Perancangan dan Pembuatan Alat Home electricity Based Home Appliance Controller Berbasis Internet Of Things".
- [6] H. Yuliansyah, "Uji Kinerja Pengiriman Data Secara Wireless menggunakan Modul ESP8266 Berbasis Rest Architecture".
- [7] Hambali, "Internet Of Things (IoT)".
- [8] A. Junaldi, "Internet Of Things, Sejarah, Teknologi dan Penerapannya," 2015.
- [9] M. B. Laili, "Sistem Internet Of Things Berbasis Cloud Computing".
- [10] D. M. Putri, "Mengenal Wemos D1 Dalam Dunia IoT," p. 2017.
- [11] m. Hossain, "A Smart IoT Based System for Monitoring and Controlling the Sub-Station Equipment," *Internet Of Things*, 2019.
- [12] Markovic, D., Koprivica, R., Pesovic, U., & Randic, S. (2015). Application of IoT in monitoring and controlling agricultural production.
- [13] Fielding, R., & Reschke, J. (2014). Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. *RFC, 7231*, 1-101.
- [14] Isaac Kim (2012). Penetration Testing Of A Web Application Using Dangerous HTTP Methods
- [15] Cerf, V., & Kahn, R. (2021). A Protocol for Packet Network Intercommunication (1974). Ideas That Created the Future
- [16] Says, C. (2020, December 22). REST API Caching – Cache Headers. REST API Tutorial. <https://restfulapi.net/caching/>.
- [17] Neumann, A., Laranjeiro, N., & Bernardino, J. (2018). An Analysis of Public REST Web Service APIs. *IEEE Transactions on Services Computing*, 1-1.
- [18] Rodríguez, C., Baez, M., Daniel, F., Casati, F., Trabucco, J., Canali, L., & Percannella, G. (n.d.). REST APIs: A Large-Scale Analysis of Compliance with Principles and Best Practices.
- [19] M. Maleshkova, C. Pedrinaci, and J. Domingue, "Investigating Web APIs on the World Wide Web," in 2010 Eighth IEEE European Conference on Web Services, 2010, pp. 107–114.

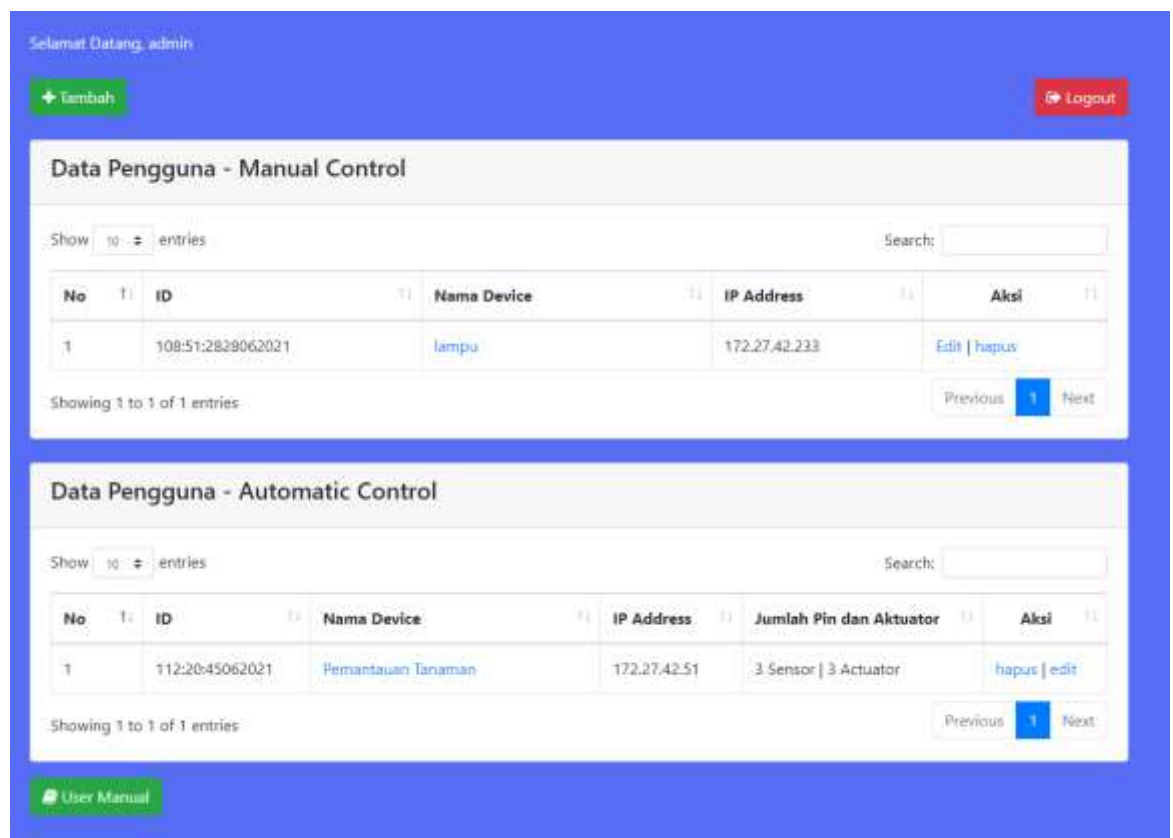
- [20] Šimec, Alen & Magličić,. (2014). Comparison of JSON and XML Data Formats.
- [21] Breje, Anca-Raluca & Gyorodi, Robert & Győrödi, Cornelia & Zmaranda, Doina & Pecherle, George. (2018). Comparative study of data sending methods for XML and JSON models. International Journal of Advanced Computer Science and Applications. 9.
- [22] Lima, Zebenzuy, Hugo García-Vázquez , Raúl Rodríguez , Sunil L. Khemchandani, Fortunato Dualibe 2 and Javier del Pino, “A System For Controlling and Monitoring IoT Applications”, 2018
- [23] Anwar, Kasyful Muhammad, Tjahjanto, “Perancangan *Database* IoT Berbasis *Cloud* dengan Restful API”, 20-2021

LAMPIRAN

Panduan Pemakaian *IoT Centralized Control and Monitoring System*

Langkah-langkah penggunaan *IoT Centralized Control and Monitoring System* untuk pengguna :

1. Hal yang pertama dilakukan adalah , pengguna memeriksa apakah sudah tersedia koneksi internet pada PC/Laptop.
2. Kemudian pengguna menjalankan XAMPP dan mengakses web <http://localhost/folder> , dimana folder web disimpan.
3. Pengguna kemudian melakukan *login* dengan menggunakan, *username* : admin dan *password* : admin. Kemudian pengguna akan masuk ke *dashboard web* seperti yang terlihat pada gambar berikut. Pada *dashboard* terdapat menu *user manual* yang berisi informasi mengenai bagaimana cara untuk menambahkan *agent* baru dan berisi *code* untuk mendapatkan ip address dari *device* iot yang akan di daftarkan.



4. Setelah pengguna masuk ke *dashboard web*, hal yang dilakukan sebelum menambahkan *agent* yang baru untuk tipe kontrol secara manual adalah

mendapatkan IP *address* dari *microcontroller*, disini pengguna dapat menggunakan kode yang sudah disediakan dan akan di *copy* ke software Arduino ide.

```
#include <ESP8266WiFi.h>

const char *ssid = "Students_2021"; //ganti nama wifi
const char *pass = "mahasiswa";//ganti password
WiFiClient client;

void setup()
{
    Serial.begin(115200);
    delay(10);

    Serial.print(" Menghubungkan ke : ");
    Serial.println(ssid);

    WiFi.begin(ssid, pass);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print("...");
    }
    Serial.print("\n");
    Serial.print("IP address : ");
    Serial.print(WiFi.localIP());
    Serial.print("\n");
    Serial.print("MAC : ");
    Serial.println(WiFi.macAddress());
    Serial.println("");
    Serial.print("Terhubung dengan : ");
    Serial.println(ssid);
}

void loop() { }
```

5. Sebelum pengguna menjalankan kode program tersebut, pengguna harus mengganti SSID dan *password* sesuai dengan *wifi* pengguna yang terhubung dan mengatur PORT yang terbuka, kemudian klik “upload”.
6. Setelah proses upload kode selesai, pengguna membuka serial monitor untuk mendapatkan Ip address dari mikrokontroler tersebut. Misalnya, Ip address yang didapatkan adalah 192.17.128.17.
7. Kemudian pengguna juga harus menjalankan kode untuk *device* IoT nya untuk melakukan pengontrolan, disini kita ambil dari kasus pengontrolan (*on/off*) lampu.

```
#include <ESP8266WiFi.h>

// di bagian ini menconnectkan ke jaringan
const char* ssid      = "Students_2021";
```

```

        const char* password = "mahasiswa";

// untuk port server dan kita menggunakan port server 80
        WiFiServer server(80);
// untuk menyimpan permintaan server http
        String header;
// variabel bantuan utk menyimpan status keluaran saat ini
        String output3State = "off";
        String output4State = "off";
        String output5State = "off";

// untuk pin inputan
        const int output3 = D7;
        const int output4 = D6;
        const int output5 = D5;
        void setup() {
            Serial.begin(115200);
// untuk keluaran
            pinMode(output3, OUTPUT);
            pinMode(output4, OUTPUT);
            pinMode(output5, OUTPUT);

// membuat hasil output LOW
            digitalWrite(output3, HIGH);
            digitalWrite(output4, HIGH);
            digitalWrite(output5, HIGH);

// terhubung ke wifi dengan ssid dan pass
            Serial.print("Connecting to ");
            Serial.println(ssid);
            WiFi.begin(ssid, password);
            while (WiFi.status() != WL_CONNECTED) {
                delay(500);
                Serial.print(".");
            }

// membuat alamat local dan mulai web server
            Serial.println("");
            Serial.println("WiFi connected.");
            Serial.println("IP address: ");
            Serial.println(WiFi.localIP());
            server.begin();
        }

        void loop(){
WiFiClient client = server.available();// melihat client ygmasuk
if (client) {           // jika client yg terhubung,
Serial.println("New Client."); // cetak pesan di port serial
String currentLine = ""; // membuat String untuk menyimpan data
yang masuk dari klien

```

```

while (client.connected()) { // loop sementara
    klien terhubung
    if (client.available()) { // jika ada byte
        untuk dibaca dari klien,
        char c = client.read(); // baca byte, lalu
        Serial.write(c); // cetaklah monitor
        serialnya
        header += c;
        if (c == '\n') { // jika byte adalah
            karakter baris baru

// jika baris saat ini kosong, Anda mendapat dua karakter baris
baru berturut-turut.

// itulah akhir dari permintaan HTTP klien, jadi kirim respons:
        if (currentLine.length () == 0) {

// Header HTTP selalu dimulai dengan kode respons (mis. HTTP /
1.1 200 OK)

// dan tipe konten agar klien tahu apa yang akan terjadi,
kemudian baris kosong:

            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

// menyalakan dan mematikan GPIO

            if (header.indexOf("GET /3/on") >= 0) {
                Serial.println("GPIO 3 on");
                output3State = "on";
                digitalWrite(output3, HIGH);
            } else if
            (header.indexOf("GET /3/off") >= 0) {
                Serial.println("GPIO 3 off");
                output3State = "off";
                digitalWrite(output3, LOW);
            } else if (header.indexOf("GET /4/on") >= 0) {
                Serial.println("GPIO 4 on");
                output4State = "on";
                digitalWrite(output4, HIGH);
            } else if (header.indexOf("GET /4/off") >= 0) {
                Serial.println("GPIO 4 off");
                output4State = "off";
                digitalWrite(output4, LOW);
            } else if (header.indexOf("GET /5/on") >= 0) {
                Serial.println("GPIO 5 on");
                output5State = "on";
                digitalWrite(output5, HIGH);
            } else if (header.indexOf("GET /5/off") >= 0) {
                Serial.println("GPIO 5 off");
                output5State = "off";
                digitalWrite(output5, LOW);
            }
        }

// Tampilkan halaman web HTML

        client.println("<!DOCTYPE html><html>");
        client.println("<head><meta name=\"viewport\"
content=\"width=device-width, initial-scale=1\">");
        client.println("<link rel=\"icon\" href=\"data:;\">");

```

```

// CSS untuk menata tombol on / off
// Jangan ragu untuk mengubah warna latar dan atribut ukuran
font agar sesuai dengan preferensi Anda

    client.println("<style>html { font-family: Helvetica;
display: inline-block; margin: 0px auto; text-align:
center;});");
    client.println(".button { background-color: #195B6A;
border: none; color: white; padding: 16px 40px;});");
    client.println("text-decoration: none; font-size: 30px;
margin: 2px; cursor: pointer;});");
    client.println(".button2 {background-color:
#77878A;}</style></head>");

// Tajuk Halaman Web
    client.println("<body><h1>TA Kelompok 2</h1>");

// Tampilkan status saat ini, dan tombol ON / OFF untuk Pin 3
    client.println("<p>GPIO 3 - State " + output3State +
"</p>");

// Jika output3 State mati, ini akan menampilkan tombol ON
    if (output3State=="off") {
    client.println("<p><a href=\""/3/on\"><button
class=\"button\">ON</button></a></p>");
    } else {
    client.println("<p><a href=\""/3/off\"><button
class=\"button button2\">OFF</button></a></p>");
    }

// Tampilkan status saat ini, dan tombol ON / OFF untuk Pin 4
    client.println("<p>GPIO 4 - State " + output4State +
"</p>");

// Jika output4 State mati, ini akan menampilkan tombol ON
    if (output4State=="off") {
    client.println("<p><a href=\""/4/on\"><button
class=\"button\">ON</button></a></p>");
    } else {
    client.println("<p><a href=\""/4/off\"><button
class=\"button button2\">OFF</button></a></p>");
    }

// Tampilkan status saat ini, dan tombol ON / OFF untuk Pin 5
    client.println("<p>GPIO 5 - State " + output5State +
"</p>");

// Jika output5 State mati, ini akan menampilkan tombol ON
    if (output5State=="off") {
    client.println("<p><a href=\""/5/on\"><button
class=\"button\">ON</button></a></p>");
    } else {
    client.println("<p><a href=\""/5/off\"><button
class=\"button button2\">OFF</button></a></p>");
    }
    client.println("</body></html>");

```

```

// Respons HTTP berakhir dengan baris kosong lain
    client.println();
// Keluar dari loop sementara
    break;
} else { // jika Anda mendapat baris baru, maka bersihkan
currentLine
    currentLine = "";
}
} else if (c != '\r') { // jika Anda mendapatkan hal lain
selain karakter carriage return,
currentLine += c; // tambahkan ke akhir currentLine
}
}
}

// Bersihkan variabel tajuk
    header = "";
// tutup koneksi
client.stop();
Serial.println("Client disconnected.");
Serial.println("");
}
}

```

8. Setelah berhasil, pengguna dapat menambahkan *agent* baru. Pada halaman tambah *agent* baru, pengguna memasukkan nama *agent*, IP address dari mikrokontroler yang sudah didapatkan sebelumnya, kemudian menginput jumlah lampu yang akan digunakan dan tipe dari *agent*-nya.

Tambah Device

Nama Agent
Lampu Rumah

IP address
192.17.128.17

Jumlah Sensor
3

Type
Manual Control

+ Tambah Data

9. Kemudian setelah menambahkan *agent* baru, pada halaman selanjutnya pengguna memasukkan nama alat dan nomor pin dari sensor sesuai dengan yang sudah dibuat dalam rangkaian *prototype* nya kemudian klik tambah data.

Tambah Data Agent

Nama: Lampu Rumah
IP : 192.17.128.17
Pin: 3
Tipe: Manual Control

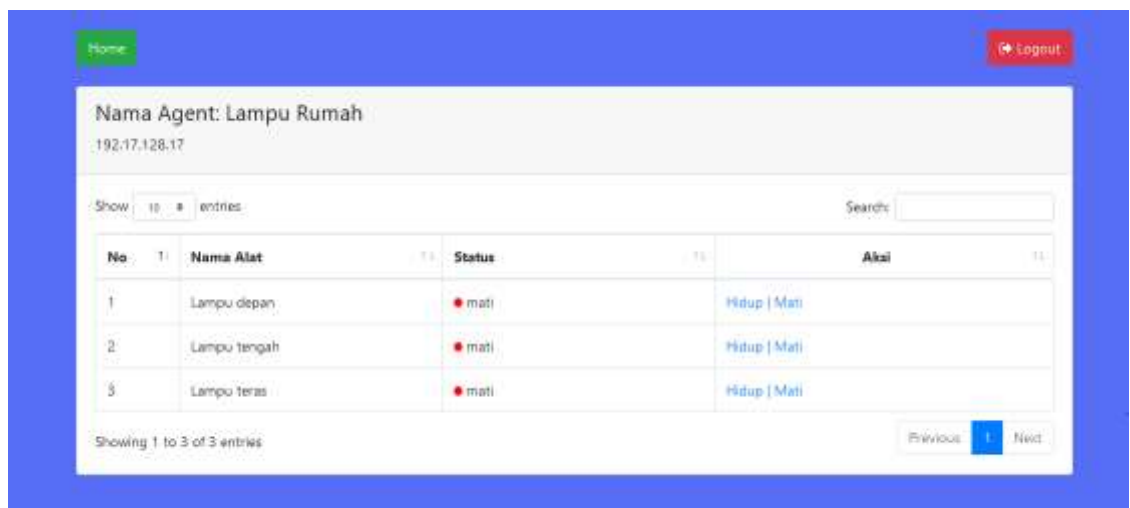
Nama Alat Lampu Depan	Pin Sensor 1 3
Nama Alat Lampu tengah	Pin Sensor 2 4
Nama Alat Lampu teras	Pin Sensor 3 5

+ Tambah Data

10. Setelah *agent* yang baru berhasil ditambahkan, maka akan muncul pada bagian *dashboard* manual kontrol.



11. Kemudian pengguna data memilih *device* mana yang akan dikontrol, seperti yang tertera pada gambar pengguna dapat mengontrol hidup dan matinya lampu melalui aplikasi *web*. Pengguna juga dapat melakukan aksi hapus dan edit pada *device* yang ingin dieksekusi.



12. Pada bagian studi kasus pengontrolan *device* iot secara otomatis untuk menambahkan *agent* baru menggunakan hal yang sama dengan pengontrolan secara manual, dengan mendapatkan IP address dari mikrokontroler terlebih dahulu dan menjalankan kode untuk pemantauan tanamannya.

Untuk kode mikrokontroler pemantauan tanaman, pengguna dapat menggunakan kode yang sudah disediakan :

```
#include <ESP8266WiFi.h>

// di bagian ini menconnectkan ke jaringan
const char* ssid      = "Students_2021";
const char* password  = "mahasiswa";
```

```

// untuk port server dan kita menggunakan port server 80
WiFiServer server(80);

// untuk menyimpan permintaan server http
String header;

// variabel bantuan utk menyimpan status keluaran saat ini
String output3State = "off";
String output4State = "off";
String output5State = "off";

// untuk pin inputan
const int output3 = D3;
const int output4 = D4;
const int output5 = D5;

void setup() {
  Serial.begin(115200);
  // untuk keluaran
  pinMode(output3, OUTPUT);
  pinMode(output4, OUTPUT);
  pinMode(output5, OUTPUT);

  // membuat hasil output LOW
  digitalWrite(output3, LOW);
  digitalWrite(output4, LOW);
  digitalWrite(output5, LOW);

  // terhubung ke wifi dengan ssid dan pass
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // membuat alamat local dan mulai web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop(){
  WiFiClient client = server.available(); // melihat
  client yg masuk

  if (client) { // jika client
    yg terhubung,
    Serial.println("New Client."); // cetak pesan
    di port serial
    String currentLine = ""; // membuat
    String untuk menyimpan data yang masuk dari klien
    while (client.connected()) { // loop sementara
      klien terhubung

```



```

        if (client.available()) { // jika ada byte
            untuk dibaca dari klien,
            char c = client.read(); // baca byte,
            lalu
            Serial.write(c); // cetaklah
            monitor serialnya
            header += c;
            if (c == '\n') { // jika byte
                adalah karakter baris baru
                // jika baris saat ini kosong, Anda mendapat dua
                karakter baris baru berturut-turut.
                // itulah akhir dari permintaan HTTP klien, jadi
                kirim respons:
                if (currentLine.length () == 0) {
                    // Header HTTP selalu dimulai dengan kode
                    respons (mis. HTTP / 1.1 200 OK)
                    // dan tipe konten agar klien tahu apa yang
                    akan terjadi, kemudian baris kosong:
                    client.println("HTTP/1.1 200 OK");
                    client.println("Content-type:text/html");
                    client.println("Connection: close");
                    client.println();

                    // menyalakan dan mematikan GPIO
                    if (header.indexOf("GET /3/on/4/on/5/on") >= 0)
{ //1
//
                    Serial.println("GPIO 3 on");
                    output3State = "on";
                    output4State = "on";
                    output5State = "on";

                    digitalWrite(output3, HIGH);
                    digitalWrite(output4, HIGH);
                    digitalWrite(output5, HIGH);

                }
                else if (header.indexOf("GET /3/off/4/on/5/on")
>= 0) { //2
                    output3State = "off";
                    output4State = "on";
                    output5State = "off";
                    digitalWrite(output3, LOW);
                    digitalWrite(output4, HIGH);
                    digitalWrite(output5, HIGH);
                    Serial.println("MATIIIIIIIIII");
                }
                else if (header.indexOf("GET
/3/on/4/off/5/on") >= 0) { //3
                    output3State = "on";
                    output4State = "off";
                    output5State = "on";
                    digitalWrite(output3, HIGH);
                    digitalWrite(output4, LOW);
                    digitalWrite(output5, HIGH);
                }
                else if (header.indexOf("GET
/3/on/4/on/5/off") >= 0) { //4
                    output3State = "on";
                    output4State = "on";
                    output5State = "off";

```

```

        digitalWrite(output3, HIGH);
        digitalWrite(output4, HIGH);
        digitalWrite(output5, LOW);
    } else if (header.indexOf("GET
/3/off/4/off/5/off") >= 0) { //5
        output3State = "off";
        output4State = "off";
        output5State = "off";
        digitalWrite(output3, LOW);
        digitalWrite(output4, LOW);
        digitalWrite(output5, LOW);
    } else if (header.indexOf("GET
/3/off/4/off/5/on") >= 0) { //6
        output3State = "off";
        output4State = "off";
        output5State = "on";
        digitalWrite(output3, LOW);
        digitalWrite(output4, LOW);
        digitalWrite(output5, HIGH);
    } else if (header.indexOf("GET
/3/off/4/on/5/off") >= 0) { //7
        output3State = "off";
        output4State = "on";
        output5State = "off";
        digitalWrite(output3, LOW);
        digitalWrite(output4, HIGH);
        digitalWrite(output5, LOW);
    } else if (header.indexOf("GET
/3/on/4/off/5/off") >= 0) { //8
        output3State = "on";
        output4State = "off";
        output5State = "off";
        digitalWrite(output3, HIGH);
        digitalWrite(output4, LOW);
        digitalWrite(output5, LOW);
    } else if (header.indexOf("GET /3/on/4/off") >=
0) { //8
        output3State = "on";
        output4State = "off";
        output5State = "off";
        digitalWrite(output3, HIGH);
        digitalWrite(output4, LOW);
        digitalWrite(output5, LOW);
    } else if (header.indexOf("GET /3/off/4/off") >=
0) { //8
        output3State = "off";
        output4State = "off";
        output5State = "off";
        digitalWrite(output3, LOW);
        digitalWrite(output4, LOW);
        digitalWrite(output5, LOW);
    } else if (header.indexOf("GET /3/on/4/on") >=
0) { //8
        output3State = "off";
        output4State = "off";
        output5State = "off";
        digitalWrite(output3, HIGH);
        digitalWrite(output4, HIGH);

```

```

//          digitalWrite(output5, LOW);
    }else if (header.indexOf("GET /3/off/4/on") >=
0) { //8
        output3State = "off";
        output4State = "off";
//          output5State = "off";
        digitalWrite(output3, LOW);
        digitalWrite(output4, HIGH);
//          digitalWrite(output5, LOW);
    }

//          else if (header.indexOf("GET /5/on") >= 0) {
//              Serial.println("GPIO 5 on");
//              output5State = "on";
//              digitalWrite(output5, HIGH);
//          } else if (header.indexOf("GET /5/off") >= 0)
//          {
//              Serial.println("GPIO 5 off");
//              output5State = "off";
//              digitalWrite(output5, LOW);
//          }

// Tampilkan halaman web HTML
client.println("<!DOCTYPE html><html>");
client.println("<head><meta    name=\"viewport\"
content=\"width=device-width, initial-scale=1\">");
client.println("<link                rel=\"icon\"
href=\"data:,\">");
// CSS untuk menata tombol on / off
// Jangan ragu untuk mengubah warna latar dan
atribut ukuran font agar sesuai dengan preferensi Anda
client.println("<style>html    {    font-family:
Helvetica; display: inline-block; margin: 0px auto; text-
align: center;}");
client.println(".button    {    background-color:
#195B6A; border: none; color: white; padding: 16px 40px;}");
client.println("text-decoration: none; font-
size: 30px; margin: 2px; cursor: pointer;}");
client.println(".button2    {background-color:
#77878A;}</style></head>");

// Tajuk Halaman Web
client.println("<body><h1>TA Kelompok 2</h1>");

// Tampilkan status saat ini, dan tombol ON /
OFF untuk Pin 3
client.println("<p>GPIO    3    -    State    "    +
output3State + "</p>");
// Jika output3 State mati, ini akan menampilkan
tombol ON
if (output3State=="off") {
    client.println("<p><a href=\""/3/on\"><button
class=\"button\">ON</button></a></p>");
} else {
    client.println("<p><a
href=\""/3/off\"><button                                class=\"button
button2\">OFF</button></a></p>");
}

```

```

        // Tampilkan status saat ini, dan tombol ON /
        OFF untuk Pin 4
        client.println("<p>GPIO 4 - State " +
        output4State + "</p>");
        // Jika output4 State mati, ini akan menampilkan
        tombol ON
        if (output4State=="off") {
            client.println("<p><a href=\""/4/on\"><button
            class=\"button\">ON</button></a></p>");
        } else {
            client.println("<p><a
            href=\""/4/off\"><button                                class=\"button
            button2\">OFF</button></a></p>");
        }

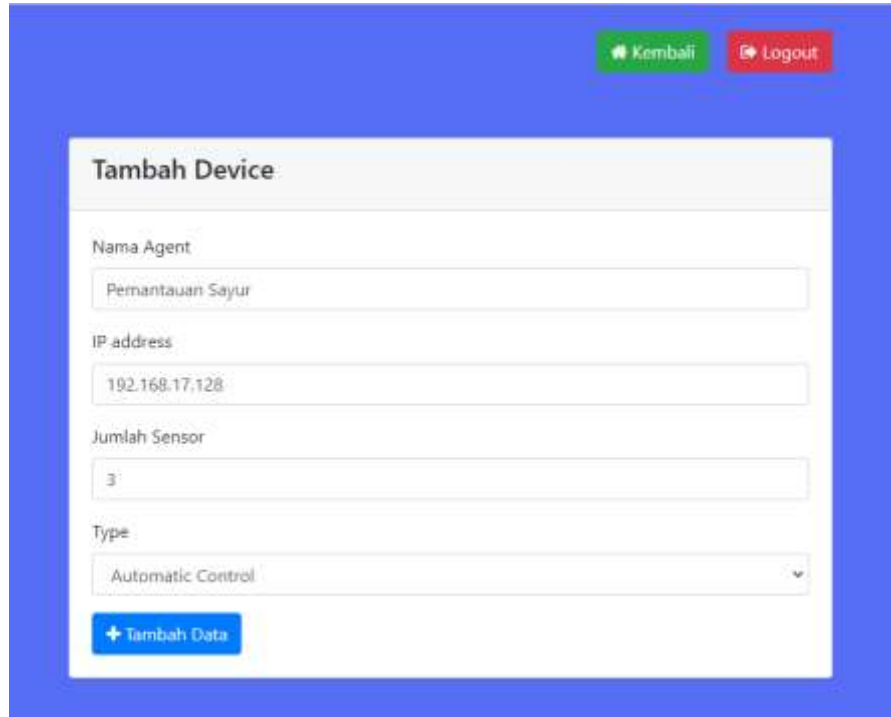
        // Tampilkan status saat ini, dan tombol ON /
        OFF untuk Pin 5
        client.println("<p>GPIO 5 - State " +
        output5State + "</p>");
        // Jika output5 State mati, ini akan menampilkan
        tombol ON
        if (output5State=="off") {
            client.println("<p><a href=\""/5/on\"><button
            class=\"button\">ON</button></a></p>");
        } else {
            client.println("<p><a
            href=\""/5/off\"><button                                class=\"button
            button2\">OFF</button></a></p>");
        }

        client.println("</body></html>");

        // Respons HTTP berakhir dengan baris kosong
        lain
        client.println();
        // Keluar dari loop sementara
        break;
    } else { // jika Anda mendapat baris baru, maka
        bersihkan currentLine
        currentLine = "";
    }
    } else if (c != '\r') { // jika Anda mendapatkan
        hal lain selain karakter carriage return,
        currentLine += c; // tambahkan ke akhir
        currentLine
    }
    }
    // Bersihkan variabel tajuk
    header = "";
    // tutup koneksi
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
}
}

```

13. Setelah berhasil mendapatkan Ip address dan berhasil menjalankan kode untuk pemantauan tanaman, maka selanjutnya pengguna akan menambahkan *agent* baru tipe *automatic controlling*. Untuk keterangan dapat dilihat seperti pada gambar berikut :



The image shows a web application interface with a blue background. At the top right, there are two buttons: a green 'Kembali' button and a red 'Logout' button. In the center, there is a white form titled 'Tambah Device'. The form contains four input fields: 'Nama Agent' with the value 'Pemantauan Sayur', 'IP address' with the value '192.168.17.128', 'Jumlah Sensor' with the value '3', and 'Type' with a dropdown menu showing 'Automatic Control'. At the bottom of the form is a blue button with a white plus icon and the text '+ Tambah Data'.

14. Kemudian pada halaman selanjutnya , pengguna memasukkan data *agent* dimulai dari mengisi ip *controller* yang sudah didapatkan sebelumnya saat menjalankan kode pemantauan tanaman, kemudian memasukkan nomor pin sensor dan aktuator serta *state* yang sudah ditentukan sebelumnya pada *prototype* rangkaian.

Kembali
Logout

Tambah Data Agent

Nama: pemantauan tanaman
 IP : 192.168.17.122
 Pin: 3
 Tipe: Automatic Control

IP Controller :

Nama Sensor	Pin Aktuator 1	State 1	Nilai sensor 1	Satuan 1	Pin Aktuator 1
<input type="text" value="Kelembapan"/>	<input type="text" value="6"/>	<input type="text" value=""/> > <	<input type="text" value="75"/>	<input type="text" value="RH"/>	<input type="text" value="2"/>
Nama Sensor	Pin Aktuator 2	State 2	Nilai sensor 2	Satuan 2	Pin Aktuator 2
<input type="text" value="Temperatur"/>	<input type="text" value="0"/>	<input type="text" value=""/> > <	<input type="text" value="25"/>	<input type="text" value="C"/>	<input type="text" value="3"/>
Nama Sensor	Pin Aktuator 3	State 3	Nilai sensor 3	Satuan 3	Pin Aktuator 3
<input type="text" value="Intensitas Cahaya"/>	<input type="text" value="7"/>	<input type="text" value=""/> > <	<input type="text" value="100"/>	<input type="text" value="Cd"/>	<input type="text" value="4"/>

+ Tambah Data

15. Dan untuk kode pemantauan tanaman pengguna dapat menggunakan kode yang sudah disediakan, pada serial monitor juga dapat dilihat untuk ip controller yang akan digunakan saat mengisi data penambahan *agent* . Setelah ditambahkan , untuk mendapatkan data yang akan ditampilkan di dalam sensor , kita harus menambahkan atau mengubah id ke dalam code pemantauan, dapat menggunakan snipped code yang sudah disediakan pada halaman home *automatic control*.

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include "DHT.h" //
#define DHTPIN D6      // pin yang terconnect dalam dht 11
                        // yaitu D6
#define DHTTYPE DHT11  // DHT 11

const char* ssid = "Students_2021";
const char* password = "mahasiswa";
const char* serverName = "http://172.30.43.122/rest-api/items/read";
unsigned long lastTime = 0;
unsigned long timerDelay = 5000;

int sensor_pin = A0;
int value ;
DHT dht(DHTPIN, DHTTYPE);
int trigPin = D8;    // Trigger
int echoPin = D7;    // Echo
```

```

long duration, cm, inches;
void setup()
{
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address:");
  Serial.println(WiFi.localIP());

  Serial.println("Timer set to 5 seconds (timerDelay variable), it will take 5 seconds before publishing the first reading.");

  Serial.println("DHT11 test!");
  dht.begin();
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop()
{
  //sensor cahaya
  int sensorValue = analogRead(sensor_pin);
  // Serial.println("Cahaya:");
  // Serial.println(sensorValue);
  delay(1000);

  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);

  // Convert the time into a distance
  cm = (duration/2) / 29.1;    // Divide by 29.1 or multiply
  by 0.0343
  inches = (duration/2) / 74;  // Divide by 74 or multiply
  by 0.0135
  //
  // Serial.print(inches);
  // Serial.print("in, ");
  // Serial.print(cm);
  // Serial.print("cm");
  // Serial.println();
  // Wait a few seconds between measurements.
  // delay(2000);

```

```

//untuk sesor dht 11
float h = dht.readHumidity();
float t = dht.readTemperature();
float f = dht.readTemperature(true);

//Periksa apakah ada pembacaan yang gagal dan keluar lebih
awal (untuk mencoba lagi).
if (isnan(h) || isnan(t) || isnan(f))
{
    Serial.println("Failed to read from DHT sensor!");
    return;
}

//menampilkan hasil pembacaan sensor
// Serial.print("Humidity: ");
// Serial.print(h);
// Serial.print(" %\t");
// Serial.print("Temperature: ");
// Serial.print(t);
// Serial.print(" *C ");
// Serial.print(f);
// Serial.println(" *F");

if ((millis() - lastTime) > timerDelay) {
    if(WiFi.status()== WL_CONNECTED){
        HTTPClient http;
        http.begin(serverName);
        http.addHeader("Content-Type", "application/x-www-
form-urlencoded");
        // If you need an HTTP request with a content type:
        application/json, use the following:
        http.addHeader("Content-Type", "application/json");
        String str =
        "{\"id\":\"109:21:36082021\",\"sensor_value\":";
        String comma = ",";
        String str2 = "[" + String(cm) + comma+String(sensorValue)
        ;
        String str3 = "]\"";
        String str4 = str +str2 +str3;
        int httpResponseCode = http.POST(str4);

        // If you need an HTTP request with a content type:
        text/plain
        //http.addHeader("Content-Type", "text/plain");
        //int httpResponseCode = http.POST("Hello, World!");

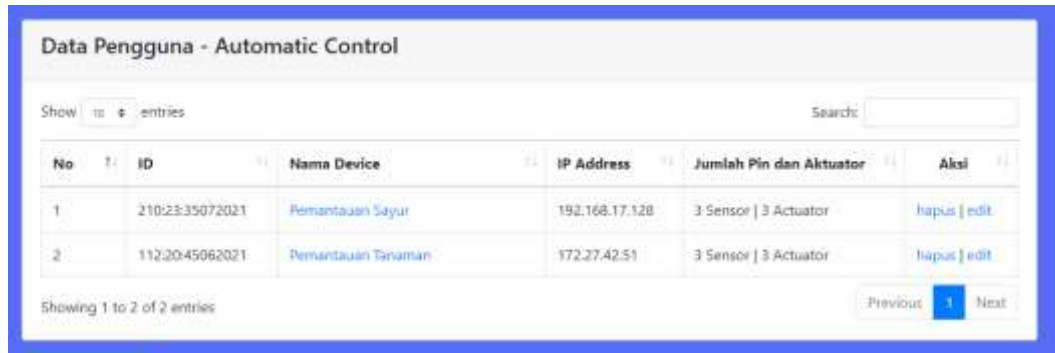
        Serial.print("HTTP Response code: ");
        Serial.println(httpResponseCode);
        http.end();
    }
    else {
        Serial.println("WiFi Disconnected");
    }
    lastTime = millis();
}

```



```
}
```

16. Setelah *agent* baru tipe kontrol otomatis berhasil ditambahkan, maka pengguna dapat memantau tanaman melalui *web* aplikasi dengan memilih *agent* mana yang akan dipantau. Pengguna juga dapat melakukan aksi “hapus” dan “edit” terhadap *agent* yang ingin dieksekusi.

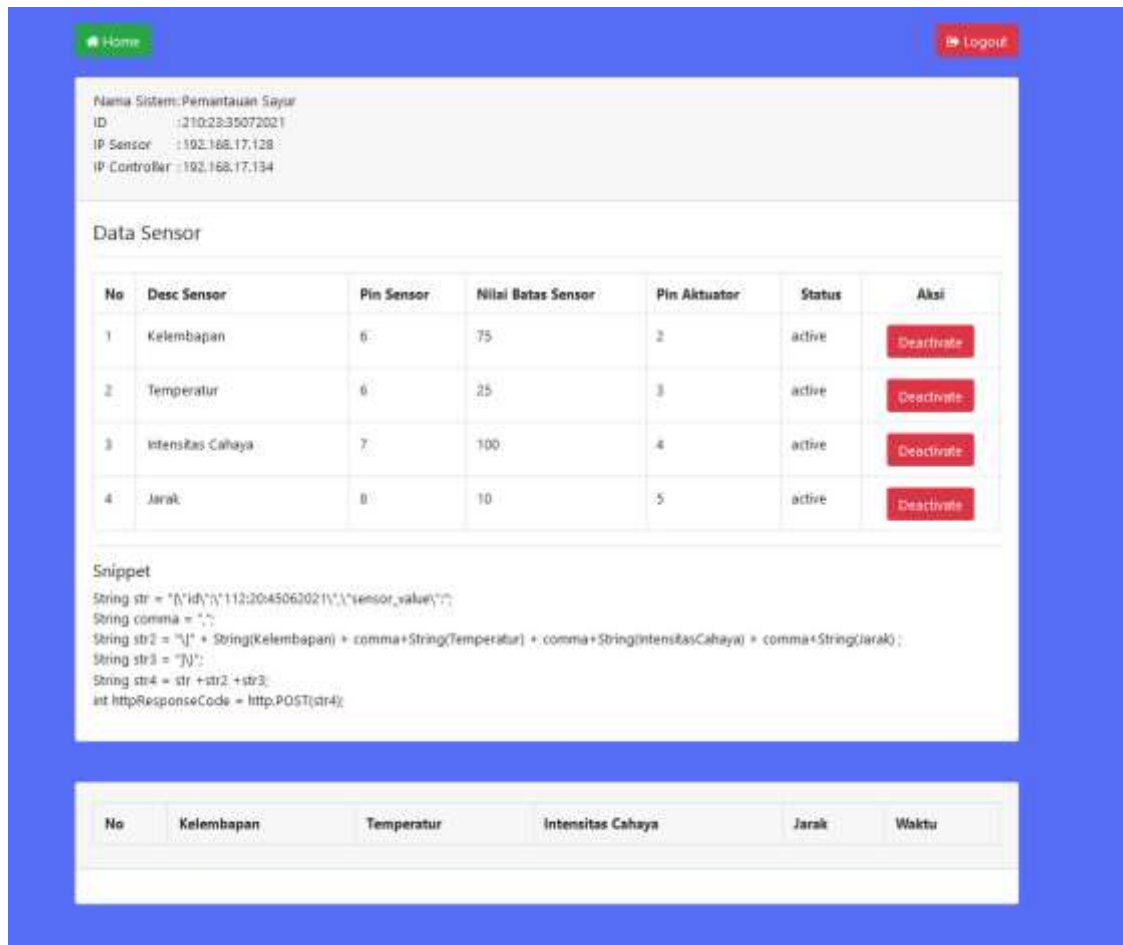


No	ID	Nama Device	IP Address	Jumlah Pin dan Aktuator	Aksi
1	210:23:35072021	Pemantauan Sayur	192.168.17.128	3 Sensor 3 Aktuator	hapus edit
2	112:20:45062021	Pemantauan Tanaman	172.27.42.51	3 Sensor 3 Aktuator	hapus edit

Showing 1 to 2 of 2 entries

Previous 1 Next

17. Pada halaman pemantauan tanaman akan menampilkan grafik hasil data yang diterima dari lingkungan dan pada halaman home pemantauan tersebut terdapat template *script* yang dapat digunakan pengguna sebagai contoh untuk menghindari adanya kesalahan pemasukan data pin sensor, karena data dari pin sensor yang dimasukkan pada *web* aplikasi harus sesuai dengan urutan yang ada pada *script* tersebut.



Pada link github berikut penulis menyediakan program dan *user manual* dari sistem yang dibangun.

Link Github : <https://github.com/michaelollifare105/TugasAkhir2.git>

BIAYA PENGELUARAN DALAM PEMBANGUNAN PROYEK

No	Description	Price @	Quantity	Total
1	Wemos D1 R1 ESP8266	Rp. 73.500,-	2	Rp. 147.000,-
2	NodeMCU	Rp. 46.500,-	2	Rp. 93.000,-
3	Relay 2 channel	Rp. 23.000,-	3	Rp. 69.000,-
4	Relay 8 channel	Rp. 53.500,-	1	Rp. 53.500,-
5	Sensor Ultrasonik	Rp. 28.000,-	2	Rp. 56.000,-
6	Sensor Moisture	Rp. 22.000,-	2	Rp. 44.000,-
7	Sensor DHT11	Rp. 19.500,-	2	Rp. 39.000,-
8	Sensor LDR	Rp. 800,-	5	Rp. 4.000,-
9	LED	Rp. 500,-	18	Rp. 9.000,-
10	Papan Breadboard	Rp. 19.500,-	2	Rp. 39.000,-
11	Kabel Jumper (Male/Female)	Rp. 11.00,-	3	Rp. 33.000,-
Total				Rp. 586.500,-